On Division Versus Saturation in Pseudo-Boolean Solving

Stephan Gocht[1] , **Jakob Nordström**[2,1] and **Amir Yehudayoff**[3]
[1]KTH Royal Institute of Technology
[2]University of Copenhagen
[3]Technion – Israel Institute of Technology
gocht@kth.se, jn@di.ku.dk, amir.yehudayoff@gmail.com

Abstract

The conflict-driven clause learning (CDCL) paradigm has revolutionized SAT solving over the last two decades. Extending this approach to pseudo-Boolean (PB) solvers doing 0-1 linear programming holds the promise of further exponential improvements in theory, but intriguingly such gains have not materialized in practice. Also intriguingly, most PB extensions of CDCL use not the division rule in cutting planes as defined in [Cook *et al.*, '87] but instead the so-called saturation rule. To the best of our knowledge, there has been no study comparing the strengths of division and saturation in the context of conflict-driven PB learning, when all linear combinations of inequalities are required to cancel variables.

We show that PB solvers with division instead of saturation can be exponentially stronger. In the other direction, we prove that simulating a single saturation step can require an exponential number of divisions. We also perform some experiments to see whether these phenomena can be observed in actual solvers. Our conclusion is that a careful combination of division and saturation seems to be crucial to harness more of the power of cutting planes.

1 Introduction

Although the Boolean satisfiability (SAT) problem is NP-complete [Cook, 1971], and hence expected to be intractable from a theoretical point of view, there has been enormous progress in performance in the last 15–20 years of SAT solvers based on *conflict-driven clause learning (CDCL)* [Marques-Silva and Sakallah, 1999; Moskewicz *et al.*, 2001].[1] Today CDCL solvers are routinely used for large-scale real-world problems in a wide range of areas [Biere *et al.*, 2009].

Annoyingly, however, there also exist tiny formulas that are completely beyond reach even for the best CDCL solvers, which highlights two limitations of this approach:

- The conjunctive normal form (CNF) used for CDCL input is a weak formalism for encoding constraints.
- The *resolution* method of reasoning used in CDCL solvers is quite weak.

Pseudo-Boolean (PB) constraints can be exponentially more concise than CNF, and PB reasoning (which can be thought of as 0-1 integer linear programming with conflict analysis) is exponentially more powerful than resolution in theory. Extending the conflict-driven framework to a pseudo-Boolean setting would therefore seem like an attractive option. However, although there are crafted benchmark formulas on which PB solvers exponentially outperform CDCL-based approaches, in practice they are often less efficient.

In this work, we study the rules of reasoning used in PB solvers and how they compare to the cutting planes method on which they are based. Interestingly, the most popular conflict-driven PB solvers use the so-called saturation rule instead of the division rule in [Cook *et al.*, 1987]. Our focus is on understanding the relative strengths of division and saturation.

Let us review some background. In this paper, by pseudo-Boolean (PB) constraints we always mean 0-1 integer linear constraints. In what follows, all such constraints are assumed to be written in *normalized form* as non-negative linear combinations of literals $\sum_i a_i \ell_i \geq A$, where the *coefficients* $a_i \in \mathbb{N}_0$ are non-negative integers, the *degree (of falsity)* $A \in \mathbb{N}_+$ is a positive integer, and *literals* ℓ_i represent variables x_i or negated variables $\overline{x}_i$ (which *cancel* to produce $x_i + \overline{x}_i = 1$, and where at most one of x_i and $\overline{x}_i$ appears in any constraint).

A disjunctive clause $x \vee \overline{y} \vee z$ is just a special case $x + \overline{y} + z \geq 1$ of a PB constraint, and we will refer to collections of such constraints as *CNF formulas*. *Cardinality constraints* are another special case where all coefficients are 1 but the degree can be larger. By a *pseudo-Boolean (PB) formula* we mean any collection of (general) PB constraints.

One approach to solving PB formulas is to convert them to CNF, either lazily by learning clauses from PB constraints during conflict analysis, as in one of the version in the *Sat4j* library [Le Berre and Parrain, 2010], or eagerly by re-encoding the whole formula to CNF and running a CDCL solver as in, e.g., *MiniSat+* [Eén and Sörensson, 2006], *Open-WBO* [Martins *et al.*, 2014], or *NaPS* [Sakai and Nabeshima, 2015]. Here we are more interested in solvers doing native pseudo-Boolean reasoning, such as *PRS* [Dixon and Ginsberg, 2002], *Galena* [Chai and Kuehlmann, 2005], *Pueblo* [Sheini and Sakallah, 2006], *Sat4j* [Le Berre and Parrain, 2010], and *RoundingSat* [Elffers and Nordström, 2018] (related, but

[1]A similar idea in the context of constraint satisfaction problems was independently developed in [Bayardo Jr. and Schrag, 1997].

slightly different, ideas were also explored in *bsolo* [Manquinho and Marques-Silva, 2006]). Needless to say, this discussion is far from a complete overview of PB solving or the even richer area of PB optimization—see, e.g., the excellent survey in [Biere *et al.*, 2009, Chapter 22] for more information.

The *cutting planes* proof system [Cook *et al.*, 1987] can be defined as consisting of rules for *literal axioms*

$$\frac{}{\ell_i \geq 0} \,, \tag{1}$$

linear combination

$$\frac{\sum_i a_i\ell_i \geq A \qquad \sum_i b_i\ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B} \; [c_A, c_B \in \mathbb{N}_0] \;, \tag{2}$$

and *division*

$$\frac{\sum_i a_i\ell_i \geq A}{\sum_i \lceil a_i/c \rceil \ell_i \geq \lceil A/c \rceil} \; [c \in \mathbb{N}_+] \;. \tag{3}$$

A toy example just to illustrate the rules is the derivation

$$\dfrac{\dfrac{6x + 2y + 3z \geq 5 \qquad x + 2y + w \geq 1}{8x + 6y + 3z + 2w \geq 7}\,\text{Linear comb.}}{3x + 2y + z + w \geq 3}\,\text{Division} \tag{4}$$

The setting in this paper is that the input is a PB formula without 0-1 solutions, and the goal is to prove unsatisfiability by deriving $0 \geq 1$. For readers more interested in optimization, this is also the situation when the solver should prove that the objective function cannot be better than in the current solution.

When we want to understand the power of a method of reasoning, we ignore algorithmic aspects and study what can be achieved assuming optimal use of the derivation rules. (This can also be a fruitful perspective because the sophisticated heuristics in modern solvers are typically beyond rigorous analysis.) In this context, it is known that cutting planes is exponentially stronger than the resolution proof system underlying CDCL [Haken, 1985; Cook *et al.*, 1987].

It can be noted that literal axioms and linear combinations are sound also over the reals, so division is where the power of cutting planes lies. In example (4) no information is lost when dividing the constraint, but this does not hold in general—for instance, a further division by 3 would yield the clause $x + y + z + w \geq 1$, which is a strictly weaker constraint.

In conflict-driven solving, linear combinations (2) are always made to cancel some variable on which the two constraints disagree, giving rise to the more restricted *generalized resolution* rule (going back to [Hooker, 1988; 1992])

$$\frac{a_j x_j + \sum_{i \neq j} a_i\ell_i \geq A \qquad b_j \overline{x}_j + \sum_{i \neq j} b_i\ell_i \geq B}{\sum_{i \neq j} \big(\frac{c}{a_j} a_i + \frac{c}{b_j} b_i\big)\ell_i \geq \frac{c}{a_j} A + \frac{c}{b_j} B - c} \,, \tag{5}$$

where $c = \mathrm{lcm}(a_j, b_j)$. What is more, PB solvers based on [Chai and Kuehlmann, 2005] do not use the division rule (3) but instead the *saturation rule*

$$\frac{\sum_i a_i\ell_i \geq A}{\sum_i \min\{a_i, A\} \cdot \ell_i \geq A} \tag{6}$$

saying that no variable coefficient need be larger than the maximum contribution required from that variable. Note that saturation, too, is a "Boolean" rule in that it is not sound over the reals. The derivation

$$\dfrac{\dfrac{2x + y + z \geq 2 \qquad 3\overline{x} + 2y + u + w \geq 3}{7y + 3z + 2u + 2w \geq 6}\,\text{Res. on } x}{6y + 3z + 2u + 2w \geq 6}\,\text{Saturation} \tag{7}$$

shows how resolution and saturation can be combined.

As discussed in [Vinyals *et al.*, 2018], this leads to the following combinations of cutting planes rules to consider from an applied PB solving perspective (where 1(b)+2(b) corresponds to [Cook *et al.*, 1987]):

1. Boolean rule: (a) saturation or (b) division.
2. Linear combinations: (a) resolution or (b) no restrictions.

The use of generalized resolution seems inherent in a conflict-driven context, but which Boolean rule to prefer is less clear. Saturation was used in the seminal paper [Chai and Kuehlmann, 2005] and has also been the rule of choice in what is arguably the most popular PB solver *Sat4j* [Le Berre and Parrain, 2010]. Division appeared only recently in *RoundingSat* [Elffers and Nordström, 2018] (although it was suggested in a more general integer linear programming setting in [Jovanovic and de Moura, 2013]).

But before choosing between these two Boolean rules, it seems natural to ask how they compare in strength! Very little is known about this. [Vinyals *et al.*, 2018] initiated a study of different subsystems of cutting planes, but in the context of PB solving, when linear combinations are restricted to be instances of generalized resolution, they failed to differentiate between division and saturation. This limited understanding stands in striking contrast to the extensive research on different versions of the resolution proof system in the context of CDCL (in, e.g., [Beame *et al.*, 2004; Buss *et al.*, 2008; Atserias *et al.*, 2011; Pipatsrisawat and Darwiche, 2011]).

In this work, we obtain the following results:

1. For cutting planes with saturation, it holds that linear combinations can be restricted to generalized resolution without (any significant) loss of proof power.
2. Cutting planes with division and generalized resolution can be exponentially stronger than cutting planes with saturation and unrestricted linear combinations.
3. To simulate a single combination of generalized resolution plus saturation (as in example (7)) can require a number of division steps that is exponential in the bitsize of the coefficients in the constraints, even if unrestricted linear combinations are allowed.

The first contribution is a strengthening of [Vinyals *et al.*, 2018], which obtained an analogous result when the input is in CNF and all coefficients in the inequalities are restricted to be of at most polynomial magnitude, but as far as we are aware the second and third results are the first of their kind.

As a complement to these theoretical contributions, we also report on a limited empirical evaluation of whether these separations can be observed in practice as well.

The rest of this paper is organized as follows. We present the proofs of the three results listed above in Sections 2, 3, and 4, respectively. After discussing the results from our experiments in Section 5, we make some concluding remarks in Section 6.

2 On the Strength of Generalized Resolution

Let us start by investigating how much of a restriction the generalized resolution rule is. It is known that this can be a severe limitation—on CNF inputs it causes cutting planes to collapse to the much weaker resolution proof system regardless of whether division or saturation is used [Vinyals *et al.*, 2018]. Hence, for cutting planes with division this restriction incurs an exponential loss in strength. However, we show that combined with saturation the generalized resolution rule in fact affords the same power as unrestricted linear combinations.

By way of a quick review of preliminaries, a *cutting planes derivation* π from a PB formula F is a sequence of PB constraints $\pi = (C_1, C_2, \ldots, C_L)$ such that each C_i is either from F or is derived from previous constraints using some subset of the rules (1)–(3) and (5)–(6) (depending on the flavour of cutting planes under study). The *length* of a derivation is the number of constraints in it. We say that a derivation π is a *proof (of unsatisfiability) for* F, or *refutation of* F, if $C_L \doteq 0 \geq A$ for $A \in \mathbb{N}_+$ (where $\doteq$ denotes syntactic equality). For the rest of the paper we will use the following terminology:

- A *resolution derivation* is a derivation where (a) the input is in CNF and (b) the only derivation rule is generalized resolution (5) followed by saturation (6) in one step.
- *Cutting planes with division* is the proof system using rules (1)–(3), and a *division derivation (refutation)* is a derivation (refutation) in this proof system.
- *Cutting planes with saturation* is the system with rules (1), (2), and (6) yielding *saturation derivations*.

If the linear combination rule (2) is restricted to be an instance of generalized resolution (5), we say that we have a *division/saturation derivation with (generalized) resolution*. With these conventions we can now state our first theorem.

Theorem 2.1. *If a PB formula F over n variables has a saturation refutation π in length L, then F also has a a saturation refutation π' with generalized resolution in length* $\mathrm{O}\big(n^2 \cdot L\big)$.

Furthermore, if F is a CNF formula, then the refutation π' obtained in Theorem 2.1 can be converted to a resolution refutation of F of the same length as π'. To see this, it is sufficient to verify that the degree of falsity in π' can never go above 1, meaning that the constraints are always semantically equivalent to clauses. Note that coefficients larger than 1 are not an issue when the degree is 1—they can simply be viewed as clauses containing the same literal multiple times. Formalizing this argument properly yields the following corollary.

Corollary 2.2. *If a CNF formula F over n variables has a saturation refutation π in length L, then there is a resolution refutation of F in length* $\mathrm{O}\big(n^2 \cdot L\big)$.

A weaker form of Corollary 2.2 was shown in [Vinyals *et al.*, 2018], namely with the added (and significant) restriction that all coefficients in the original refutation π have to be small.

In what remains of this section we will prove Theorem 2.1. Starting with the refutation $\pi = (C_1, C_2, \ldots, C_L)$, we will construct π' by representing each C_i by a set of m_i constraints $D_{ij} \doteq \sum_{k=1}^{n} a_{ijk}\ell_k \geq A_{ij}$ for $j \in [m_i]$ and associated factors $\delta_{ij} \in \mathbb{N}_+$, writing $\mathcal{D}_i = \big\{(\delta_{ij}, D_{ij}) \,\big|\, j \in [m_i]\big\}$ to denote the constraints and factors for C_i. We will require the following invariants to hold for all $i \in [L]$:

1. $\mathcal{D}_i$ represents C_i in the sense that $\sum_{j=1}^{m_i} \delta_{ij} \cdot D_{ij} = \gamma \cdot C_i$ for some $\gamma \in \mathbb{N}_+$ (where the summation notation denotes taking linear combinations as in (2) but of arbitrary arity).
2. $\mathcal{D}_i$ has size $|\mathcal{D}_i| = m_i \leq n + 1$.
3. Every variable in $\mathcal{D}_i$ occurs with only one *polarity* (i.e., cannot appear both negated and unnegated in $\mathcal{D}_i$).
4. If C_i is derived from $C_{i'}$ (and $C_{i''}$) then all constraints in $\mathcal{D}_i$ can be derived from $\mathcal{D}_{i'}$ (and $\mathcal{D}_{i''}$) using at most $\mathrm{O}\big(n^2\big)$ generalized resolution and saturation steps.

Let us argue that Theorem 2.1 follows immediately from a construction maintaining these invariants. The refutation π' will consist of the constraints in the sets $\mathcal{D}_i$ concatenated with the intermediate derivation steps in invariant 4, using only generalized resolution and saturation. Since the final line in the refutation π is $C_L \doteq 0 \geq A$ for some $A \in \mathbb{N}_+$, it follows that all constraints in $\mathcal{D}_L$ are of the form $0 \geq A'$, $A' \in \mathbb{N}_+$ (since adding all constraints in $\mathcal{D}_i$ must yield a multiple of $0 \geq A$ by invariant 1 and no variables can cancel by invariant 3). Finally, the length of π' is $\mathrm{O}\big(n^2 \cdot L\big)$ because each constraint C_i is replaced by $n + 1$ constraints by invariant 2 in addition to the $\mathrm{O}\big(n^2\big)$ constraints that are used to derive C_i by invariant 4.

We can take care of invariant 2 directly, arguing similarly to the proof of Caratheodory's theorem. We omit the proof due to space constraints, but the idea is that if a positive integer linear combination of a set of constraints $\mathcal{D}$ yields some constraint C, then we only need a linearly independent subset of at most $n + 1$ constraints to get a multiple of C. We now present an inductive construction that maintains the other invariants.

Base Case: $C_i \in F$ or $C_i \doteq \ell \geq 0$. Set $\mathcal{D}_i = \{(1, C_i)\}$. The invariants hold trivially.

Saturation: If C_i is obtained by saturation of $C_{i'}$, which we denote $C_i = \mathsf{sat}(C_{i'})$, then we let $\mathcal{D}_i$ consist of the set $\big\{(\delta, \mathsf{sat}(D)) \,\big|\, (\delta, D) \in \mathcal{D}_{i'}\big\}$ plus possibly $\{(\delta_k, \ell_k \geq 0)\}$ for some literals ℓ_k in $\mathcal{D}_{i'}$ as discussed below. Invariant 3 holds by construction, as it already holds for $\mathcal{D}_{i'}$. Let us argue that Invariants 1 and 4 can be made to hold as well.

First note that if we would sum over $\mathcal{D}_{i'}$ and then saturate, we would obtain the desired constraint $\mathsf{sat}(\sum_{j=1}^{m_{i'}} \delta_{i'j} D_{i'j}) = \mathsf{sat}(\gamma C_{i'})$ using invariant 1 and the fact that $\mathsf{sat}(\gamma C_{i'}) = \gamma \cdot \mathsf{sat}(C_{i'}) = \gamma C_i$, but now we are saturating before taking the summation. However, the degree of falsity in the final constraint does not depend on the order of saturation and summation, as there are no cancellations when adding the constraints in $\mathcal{D}_{i'}$ due to invariant 3, and saturation does not affect the degree. Therefore, the only difference when saturating first are the coefficients, and the only thing that can happen to them is that they get smaller, making the final constraint stronger.

For a fixed literal ℓ_k, the coefficient when saturation happens first is $\sum_{j=1}^{m_i} \delta_{ij} a_{ijk} = \sum_{j=1}^{m_{i'}} \delta_{i'j} \min(a_{i'jk}, A_{i'j}) \leq \min(\sum_{j\in[m_{i'}]} \delta_{i'j} a_{i'jk}, \sum_{j\in[m_{i'}]} \delta_{i'j} A_{i'j})$, where the last expression is the coefficient if summation is done before saturation. Therefore, all that is needed to get $\sum_{j=1}^{m_i} \delta_{ij} D_{ij} = \gamma C_i$ is to add $\ell_k \geq 0$ to $\mathcal{D}_i$ for literals ℓ_k with too small coefficients.

Linear Combination: If C_i is derived by linear combination, i.e., $C_i = cC_{i'} + c'C_{i''}$ then we join the sets $\mathcal{D}_{i'}, \mathcal{D}_{i''}$ and multiply the factors for each constraint by c or c' to obtain $\mathcal{D}'_i = \{(c\delta, D) \mid (\delta, D) \in \mathcal{D}_{i'}\} \cup \{(c'\delta, D) \mid (\delta, D) \in \mathcal{D}_{i''}\}$. Summing all constraints in $\mathcal{D}'_i$ will yield a multiple of C_i by invariant 1 and because addition of constraints is associative, i.e., the order of addition does not matter.

However, $\mathcal{D}'_i$ might be violating invariant 3. To fix this, for every variable x and every pair of constraints containing x with opposite polarities we can replace one of the constraints by their resolvent over x as in (5). After each such step, it is still true that the constraints in $\mathcal{D}'_i$ can be summed up to yield a multiple of C_i, possibly after adapting the δ-factors. To formalize this argument we need the next lemma.

Lemma 2.3. *Let C_1, C_2 be any two constraints in which a variable x occurs with opposite polarities, and let $\delta_1, \delta_2 \in \mathbb{N}_+$. Then generalized resolution can be used to obtain constraints D_1, D_2 such that x does not occur in D_1 and there are $\gamma, \widehat{\delta}_1, \widehat{\delta}_2 \in \mathbb{N}_0$ for which $\gamma(\delta_1 C_1 + \delta_2 C_2) = \widehat{\delta}_1 D_1 + \widehat{\delta}_2 D_2$.*

Proof. Let a_1, a_2 be the coefficients of x and $\overline{x}$ respectively in C_1, C_2 and let $c = \operatorname{lcm}(a_1, a_2)$, $c_1 = c/a_1$, and $c_2 = c/a_2$. Apply generalized resolution to derive $D_1 = c_1 \cdot C_1 + c_2 \cdot C_2$ (which by construction does not contain x). Assuming without loss of generality (because of symmetry) that $\delta_1 c_2 \geq \delta_2 c_1$, set $D_2 = C_1$, $\widehat{\delta}_1 = \delta_2$, $\widehat{\delta}_2 = \delta_1 c_2 - \delta_2 c_1$, and $\gamma = c_2$. Then it holds that $\gamma \cdot (\delta_1 C_1 + \delta_2 C_2) = (\delta_1 c_2 - \delta_2 c_1) \cdot C_1 + \delta_2 c_1 \cdot C_1 + \delta_2 c_2 \cdot C_2 = \widehat{\delta} D_1 + \widehat{\delta}_2 D_2$, establishing the lemma. □

To restore invariant 3, we apply Lemma 2.3 repeatedly to variables x occurring with opposite polarities in $\mathcal{D}'_i$ as follows. Each time Lemma 2.3 is invoked one constraint (out of at most $n + 1$) is replaced by another one that does not contain x. This continues until x occurs with only one polarity or has vanished completely, and this is maintained when the process is repeated for the next variable. Therefore, we will obtain a set $\mathcal{D}_i$ that no longer violates invariant 3 after $\mathrm{O}(n^2)$ applications of the generalized resolution rule.

3 On the Strength of Division

We now turn to studying how the division and saturation rules compare in strength assuming that linear combinations are restricted to generalized resolution, as is the case in conflict-driven PB solving. Without this restriction, [Vinyals *et al.*, 2018] exhibited a family of CNF formulas witnessing that division can be exponentially stronger than saturation in cutting planes. CNF formulas are of no use here, since for such inputs cutting planes with generalized resolution is the same as the resolution proof system regardless of which Boolean rule is used, but nevertheless it is helpful to study these separating CNF formulas. They contain many subsets of clauses

$$\ell_1 + \ell_2 + \ell_3 \qquad \geq 1 \tag{8a}$$
$$\ell_1 + \ell_2 \qquad + \ell_4 \geq 1 \tag{8b}$$
$$\ell_1 \qquad + \ell_3 + \ell_4 \geq 1 \tag{8c}$$
$$\ell_2 + \ell_3 + \ell_4 \geq 1 \tag{8d}$$

which can be summed up to get

$$3\ell_1 + 3\ell_2 + 3\ell_3 + 3\ell_4 \geq 4 \tag{9}$$

after which division by 3 recovers the cardinality constraint

$$\ell_1 + \ell_2 + \ell_3 + \ell_4 \geq 2 \ . \tag{10}$$

Although the constraints (9) and (10) are semantically equivalent over the integers the former constraint is weaker over the reals, and it turns out to be crucial to have constraints of the latter form in order to prove contradiction efficiently.

The reason this yields nothing in a setting with generalized resolution is that there are no literals with opposite polarity in (8a)–(8d), and so there is no legal way to sum up these constraints to give division the chance to go from (9) to (10). However, a moment of thought reveals that we can "cheat" by changing our formula to a "morally equivalent" but syntactically different one. The trick is to re-encode (8a)–(8d) by introducing helper variables x, y and z, writing

$$x + y + z + \ell_1 + \ell_2 + \ell_3 \qquad \geq 1 \tag{11a}$$
$$\overline{x} + \ell_1 + \ell_2 \qquad + \ell_4 \geq 2 \tag{11b}$$
$$\overline{y} + \ell_1 \qquad + \ell_3 + \ell_4 \geq 2 \tag{11c}$$
$$\overline{z} + \qquad \ell_2 + \ell_3 + \ell_4 \geq 2 \tag{11d}$$

(where x, y, z are unique to this subset of constraints). Since the helper variables cancel, it is now legal to apply generalized resolution to all constraints. This results in (9), after which division yields the inequality (10) as desired. Applying this re-encoding trick to the separating CNF formulas used in [Vinyals *et al.*, 2018] leads to the following theorem.

Theorem 3.1. *There is a family of PB formulas $\{F_n\}_{n \in \mathbb{N}_+}$ with $\mathrm{O}(n)$ variables and constraints that can be refuted in length $\mathrm{O}(n)$ in cutting planes with division and generalized resolution, but for which any saturation refutations, even with unrestricted linear combinations, have length $\exp(\Omega(n))$.*

Proof. Let $\{F_n\}$ be *subset cardinality formulas* as in [Mikša and Nordström, 2014] that require exponential length for the resolution proof system but that, once cardinality constraints are recovered, have short refutations in cutting planes with generalized resolution as shown in [Vinyals *et al.*, 2018].

Let F'_n be the formula obtained from F_n by introducing helper variables as in (11a)–(11d). We argued above that generalized resolution followed by a division step recovers (10), after which we can use the efficient refutation with generalized resolution in [Vinyals *et al.*, 2018]. It remains to argue why F'_n is exponentially hard for cutting planes with saturation.

Note that if we assign the helper variables to false in (11a)–(11d), then we get back the clauses (8a)–(8d) in the original formula F_n. Letting ρ be the partial assignment, or *restriction*, that sets all helper variables in the whole formula to false, we write $C\!\restriction_\rho$ for the result of applying ρ to a constraint C, and extend this notation to sets of constraints by taking unions. It is not hard to show that if π is a saturation (or division) refutation of F with unrestricted linear combinations, then applying ρ to the lines of π results in a saturation (or division, respectively) refutation $\pi\!\restriction_\rho$ of $F\!\restriction_\rho$, except that we might need to insert some linear combinations with literal axioms to make sure that the derivation stays syntactically valid (but this can only increase the length by a factor of n.)

Let π' be a saturation refutation of F'_n. Applying ρ yields a saturation refutation $\pi'\!\restriction_\rho$ of $F'_n\!\restriction_\rho = F_n$ that is at most a factor $\mathrm{O}(n)$ longer. Appealing to Corollary 2.2, we obtain a resolution refutation π^* at most a factor $\mathrm{O}(n^2)$ longer than $\pi' \upharpoonright_\rho$. But by [Mikša and Nordström, 2014] we know that any resolution refutation π^* of F_n must have exponential length, and hence an exponential lower bound holds also for the saturation refutation π' of F'_n. The theorem follows. $\square$

4 On the Strength of Saturation

So far in the paper we have given evidence that the division rule can be exponentially stronger than the saturation rule in certain contexts. In this section we show that there are settings in which saturation can be significantly stronger than division. Unfortunately, we are not able to get a analogous result to Theorem 3.1, but what we can prove is that in order to go from

$$C_1(R) \doteq Rx + Ry + \textstyle\sum_{j=1}^{R} z_j \geq R \tag{12a}$$

$$C_2(R) \doteq Rx + R\overline{y} + \textstyle\sum_{j=R+1}^{2R} z_j \geq R \tag{12b}$$

to

$$C_L(R) \doteq Rx + \textstyle\sum_{j=1}^{2R} z_j \geq R \ , \tag{13}$$

which can be done with one resolution step followed by one saturation step, at least $\Omega\big(\sqrt{R}\big)$ applications of the division rule are required (in addition to other steps). Note that this is exponential in the bitsize of R.

A formal proof follows below, but let us first sketch the idea. It can be shown by a simple inductive argument that for any constraint containing negated literals $\overline{x}$ or $\overline{z}_j$ we can instead derive the same constraint without these literals. Therefore, it is only necessary to consider constraints C_i of the form

$$a_i x + b_i y + c_i \overline{y} + \textstyle\sum_{j=1}^{2R} d_{ij} z_j \geq A_i \tag{14}$$

(where $\min\{b_i, c_i\} = 0$). For such a constraint C_i we write $B_i = 2a_i + b_i + c_i$ and define the *potential* $\mathcal{P}(C_i)$ to be

$$\mathcal{P}(C_i) = \ln\big(B_i/A_i\big) \ . \tag{15}$$

Note that $\mathcal{P}(C_1(R)) = \mathcal{P}(C_2(R)) = \ln(3) > \ln(2) = \mathcal{P}(C_L(R))$. What we will prove in Lemma 4.2 is that only division can decrease the potential of derived constraints, and only does so by an amount of at most $1/\sqrt{R}$. This shows that $\Omega(\sqrt{R})$ divisions are required to derive $C_L(R)$ from $C_1(R)$ and $C_2(R)$. Our formal result is as follows.

Theorem 4.1. *Let $R = K^2$ for $K \in \mathbb{N}_+$ and let π be a division derivation of $C_L(R)$ from $C_1(R)$ and $C_2(R)$ (with unrestricted linear combinations). Then π contains $\Omega(\sqrt{R})$ applications of the division rule.*

We remark that the lower bound is tight except possibly for the square root. As discussed in [Vinyals *et al.*, 2018], for constraints with coefficients of size at most R it is always possible to simulate saturation with $\mathrm{O}(R)$ division and unrestricted linear combination steps.

To establish Theorem 4.1, we start with a preprocessing step to ensure that the degree A_i of any constraint $C_i \in \pi$ obtained by division is sufficiently large, namely $A_i \geq \sqrt{R} + 1$.

Suppose $A_{i_1} < \sqrt{R} + 1$ for some constraint C_{i_1} resulting from division. Since $\sqrt{R}$ is an integer by assumption, we have $A_{i_1} \leq \sqrt{R}$. We claim that C_{i_1} can be satisfied by setting at most $\sqrt{R}$ variables z_j to true. To see why, note that $C_1(R)$ and $C_2(R)$ are satified by setting all z_j to true, and hence any constraint derived from them must also be satisfied by this assignment since the proof system is sound. Furthermore, for C_{i_1} it must be sufficient to assign a subset of at most $\sqrt{R}$ variables z_j, since every z_j contributes at least 1 to the left-hand side and the degree on the right is $A_{i_1} \leq \sqrt{R}$. Let ρ_1 be such a partial assignment to at most $\sqrt{R}$ variables z_j fixing C_{i_1} to true and consider the restricted derivation $\pi\upharpoonright_{\rho_1}$ (where $C_{i_1}\upharpoonright_\rho$ has been removed since it is now a trivial constraint).

Suppose the derivation $\pi\upharpoonright_{\rho_1}$, contains some constraint C_{i_2} with degree $A_{i_2} < \sqrt{R} + 1$ (note that degrees might have decreased after the restriction ρ_1). Argue as above to find a restriction ρ_2 to at most $\sqrt{R}$ variables z_j satisfying C_{i_2}, and continue with the derivation $\pi\upharpoonright_{\rho_1 \cup \rho_2}$. We repeat this procedure for T steps if possible as long as $T \leq \sqrt{R}/6$. If at the end of this process there is still some constraint C_i with degree $A_i < \sqrt{R} + 1$, then we have counted $\sqrt{R}/6$ division steps, which is enough to obtain the lower bound in Theorem 4.1. Otherwise, it now holds for $\rho = \rho_1 \cup \cdots \cup \rho_T$ that all constraints in $\pi\upharpoonright_\rho$ have degree $A_i \geq \sqrt{R} + 1$ and that ρ assigns at most $(\sqrt{R}/6) \cdot \sqrt{R} = R/6$ variables z_j.

For the rest of the proof we will focus on the restricted derivation $\pi\upharpoonright_\rho$. It is immediate from (15) that the potential of the constraints can only increase compared to π, since restricting variables can only cause the degree of falsity to go down. Hence, for the initial constraints we have $\min\{\mathcal{P}(C_1(R)\upharpoonright_\rho), \mathcal{P}(C_2(R)\upharpoonright_\rho)\} \geq \ln(3)$. But the potential does not increase too much—since ρ sets at most $R/6$ variables z_j, for the final constraint we have $\mathcal{P}(C_L(R)\upharpoonright_\rho) \leq \ln(2R/(R - R/6)) = \ln(12/5)$. Therefore, the difference is still at least $\ln(3) - \ln(12/5) = \ln(15/12) \geq 1/6$. We will now show that the potential can only decrease after a division step, and only by $1/\sqrt{R}$, which establishes Theorem 4.1.

For convenience, we split the linear combination rule (2) into two rules for multiplying a constraint and adding two constraints. Also, when dividing a constraint C by $k \in \mathbb{N}_+$ as in (3), which we denote $\mathsf{div}(C, k)$, we assume k divides all coefficients in C. This is without loss of generality, since literal axioms (1) can be added as needed to make this true.

Lemma 4.2. *For any constraints C_i and $C_{i'}$ of the form* (14) *derived from* (12a) *and* (12b) *(possibly with some variables z_j restricted to true), for any literal axiom E as in* (1), *and for any $k \in \mathbb{N}_+$, it holds that:*

1. *$\mathcal{P}(k \cdot C_i) = \mathcal{P}(C_i)$.*
2. *$\mathcal{P}(C_i + k \cdot E) \geq \mathcal{P}(C_i)$.*
3. *$\mathcal{P}(C_i + C_{i'}) \geq \min\{\mathcal{P}(C_i), \mathcal{P}(C_{i'})\}$.*
4. *$\mathcal{P}(\mathsf{div}(C_i, k)) \geq \mathcal{P}(C_i) - 1/\sqrt{R}$, assuming that $\mathsf{div}(C_i, k)$ has degree at least $\sqrt{R} + 1$.*

Proof. Part 1 is obvious from the definition in (15).

Part 2 is trivially true if no cancellation occurs as only the numerator in the potential can increase. Suppose that the degree decreases by $k' \leq k$ through cancellation on x.

Analogously to the argument in the preprocessing step, setting x to true satisfies $C_1(R)$ and $C_2(R)$ and hence also C_i. Thus, $a_i \geq A_i$ and $B_i \geq 2 \cdot A_i$, from which it follows that $\exp(\mathcal{P}(C_i + k \cdot E)) = (B_i - 2k')/(A_i - k') \geq B_i/A_i = \exp(\mathcal{P}(C_i))$. It is straightforward to verify that if the cancellation is due to some other variable than x, then the numerator can only be larger, and hence so will the potential.

For part 3, assuming that $\mathcal{P}(C_i) \leq \mathcal{P}(C_{i'})$ (without loss of generality due to symmetry), we have $(B_i + B_{i'})/(A_i + A_{i'}) \geq B_i/A_i$. As in part 2, we have $B_i \geq 2 \cdot A_i$ and $B_{i'} \geq 2 \cdot A_{i'}$. Let k be the decrease in degree due to cancellation on y (other variables do not occur negated in (14) and cannot cancel). Then we get $\exp(\mathcal{P}(C_i + C_{i'})) = (B_i + B_{i'} - 2k)/(A_i + A_{i'} - k) \geq (B_i + B_{i'})/(A_i + A_{i'}) \geq B_i/A_i = \exp(\mathcal{P}(C_i))$.

For part 4, since all coefficients are divisible by k this also holds for B_i. Therefore, $\mathcal{P}(C_i) - \mathcal{P}(\mathsf{div}(C_i, k)) = \ln(B_i/A_i) - \ln\left((B_i/k)/\lceil A_i/k \rceil\right) = \ln\left(\lceil A_i/k \rceil/(A_i/k)\right) \leq \ln\left((A_i/k + 1)/(A_i/k)\right) \leq k/A_i \leq 1/\sqrt{R}$, where the second to last inequality is just $\ln(1+x) \leq x$ and the last inequality holds since $A_i/k \geq \lceil A_i/k \rceil - 1 \geq \sqrt{R}$ by the assumption about the degree $\lceil A_i/k \rceil$ of $\mathsf{div}(C_i, k)$. $\square$

5 Empirical Evaluation

We have shown that division and saturation are incomparable in strength as rules of reasoning. It is important to understand, however, that these results speak only about the *existence* of proofs and not about whether pseudo-Boolean solvers will actually be able to *find* such proofs. Although our main focus in this paper is on the former question, in this section we report on some limited experiments to shed some light on the latter.

We have run instrumented versions of *Sat4j*, which uses saturation, and *RoundingSat*, which defaults to division but has an option to use saturation instead. All experiments were performed on 4 AMD Opteron 6238 (Interlagos) 12-core 2.6 GHz processors with 128 GB RAM with a 5000-second time-out.

The heuristics for PB solvers are not at all as well-tuned as those for CDCL solvers, and small changes in internal settings can have huge, and currently not so well understood, effects on performance. In order to measure the overall impact of division versus saturation—rather than of some other, unrelated settings—we have therefore run the solvers with several different parameter settings and measured for each PB instance the best result with division and saturation, thus obtaining *virtual best solvers (VBS)* for division and saturation, respectively. More details about the experiments and full results can be found at www.csc.kth.se/~jakobn/DivisionVsSaturation.

To obtain benchmarks that are easy for division but hard for saturation, we use subset cardinality formulas as in Section 3, generated from 4-regular random bipartite graphs with an additional random edge added (see [Mikša and Nordström, 2014] for more details). Almost all the constraints in these formulas are of the form $\ell_1 + \ell_2 + \ell_3 + \ell_4 \geq 2$, and we have compared solver performance on this "unobfuscated" version with the "division-friendly" version based on the clausal encoding in (8a)–(8d), enhanced with helper variables as in (11a)–(11d). We have run the solvers on instances of increasing size to see how the performance scales asymptotically.

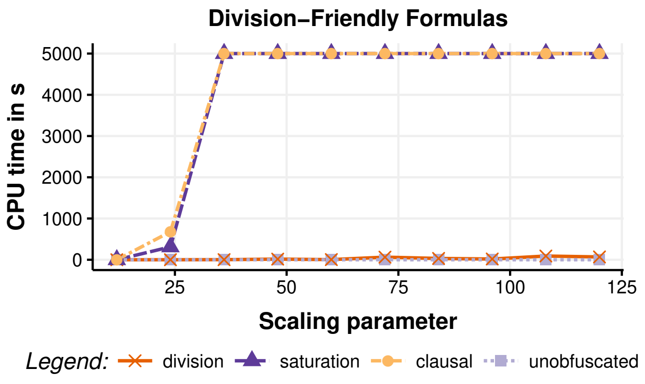

Figure 1: Virtual best division/ saturation solver for division-friendly formula compared to clausal and unobfuscated encoding.

As shown in Figure 1, we have consistently very poor solver performance for the clausal encoding. This is as expected, since this version is exponentially hard for both division- and saturation-based solvers. Furthermore, while the unobfuscated encoding is easy for both division and saturation, for the division-friendly encoding the saturation-VBS again struggles whereas the division-VBS is able to harness the helper variables to achieve a performance close to that of the unobfuscated encoding. This all fits perfectly with theory. Before we get too carried away by this, however, it should be noted that this result is rather fragile. Whether the division-based solver works well or not depends heavily also on how other internal parameters are adjusted, and it turns out to be even more crucial exactly how the helper variables are added.

It is not known whether there exist PB formulas that are easy for saturation but provably hard for division—this is a very interesting question left open by our work. What we can do to obtain interesting benchmarks, though, is to use inspiration from Section 4 to design formulas that are easy for saturation but *appear* to be tricky for division. To this end, we construct a pigeonhole principle-like formula which we think of as being defined in terms of a $(2R + 2) \times (2R + 1)$ matrix, where we scale R to increase the instance size. The variables are x_{ij}, $i \in [2R + 2]$, $j \in [2R + 1]$, with coefficients $a_{ii} = R$ and $a_{ij} = 1$ for $i \neq j$. We first consider an "unobfuscated" formula with constraints

$$\textstyle\sum_{j=1}^{2R+1} a_{ij} x_{ij} \geq R \qquad \text{for } i \in [2R + 2] \tag{16a}$$

$$\textstyle\sum_{i=1}^{2R+2} a_{ij} x_{ij} \leq R \qquad \text{for } j \in [2R + 1] \tag{16b}$$

which is easily seen to be unsatisfiable by adding all row constraints (16a) and all column constraints (16b) separately. This proof can be carried out using only generalized resolution, and so these formulas are easy in theory for all PB solvers regardless of which Boolean rule they use.

To get a formula that is easy for saturation but potentially tricky for division, we observe that for the rows $i < 2R + 2$ the constraint (16a) is of the form (13) and can be "split" into

$$Rx_{i,i} + Ry_i + \textstyle\sum_{j=1;j\neq i}^{R'_i} a_{ij} x_{ij} \geq R \tag{17a}$$

$$Rx_{i,i} + R\overline{y}_i + \textstyle\sum_{j=R'_i+1;j\neq i}^{2R+1} a_{ij} x_{ij} \geq R \tag{17b}$$

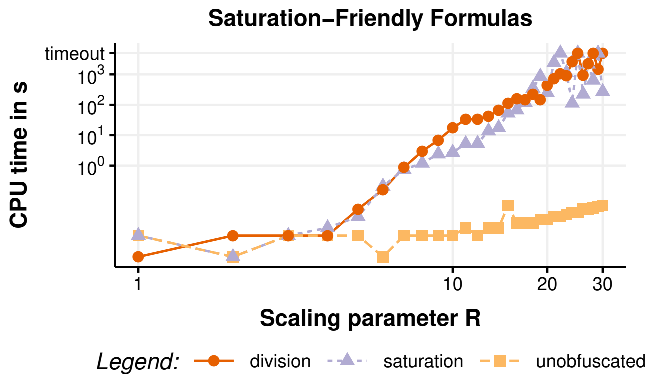

Figure 2: Virtual best division/ saturation solver for saturation-friendly formula compared to unobfuscated encoding as baseline

similar to (12a)–(12b), where $R'_i = R + 1$ if $i \leq R$ and $R'_i = R$ if $i > R$. It is straightforward to verify that this does not change anything from a saturation point of view—the new formula can still be solved in $\mathrm{O}(R)$ steps by first using generalized resolution plus saturation to go from (17a) and (17b) back to (16a), reverting the obfuscation, and then adding all constraints (16a) and (16b) in a cancelling way as discussed above. If we use division instead of saturation, the formula can be solved in $\mathrm{O}\big(R^2\big)$ steps, since each recovery of a constraint (16a) from (17a)–(17b) can be done with at most R divisions and linear combinations [Vinyals *et al.*, 2018], but we know from Section 4 that any proof starting by such a "recovery phase" requires $\Omega\big(R \cdot \sqrt{R}\big) = \Omega\big(R^{3/2}\big)$ steps. Furthermore, it should be noted that the refutation in length $\mathrm{O}\big(R^2\big)$ employs non-cancelling linear combinations, and it is not clear what the best approach is if we insist on using generalized resolution, as PB solvers do. Thus, we could hope that formulas of this type should be significantly harder for solvers using division than for solvers using saturation.

Sadly, however, the experimental results fail to confirm this intuition. The problem is not that these formulas are too easy for division, but rather that they are too hard for saturation (see Figure 2). While the baseline version is easy as expected, the obfuscated formula is hard for both division and saturation with no clear difference between the two.

This finding illustrates what we discussed at the beginning of this section, namely the difference between the non-constructive existence of short proofs and the constructive, algorithmic search for such short proofs. In this case, not only the choice of rules is important, but also the order in which these rules are applied. To illustrate this, let $R = 2$ and consider, e.g., the "split" row constraint (17a)–(17b) for $i = 1$ and the column constraint (16b) for $j = 2$, i.e.,

$$2x_{11} + 2y_1 + x_{12} + x_{13} \geq 2 \qquad (18a)$$
$$2x_{11} + 2\overline{y}_1 + x_{14} + x_{15} \geq 2 \qquad (18b)$$
$$\overline{x}_{12} + 2\overline{x}_{22} + \overline{x}_{32} + \overline{x}_{42} \geq 3 \qquad (18c)$$

(where (18c) is just (16b) written in normalized form). Performing generalized resolution on (18a) and (18b) followed by saturation recovers the unobfuscated row constraint

$$2x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \geq 2 \ , \qquad (19)$$

and resolving this constraint with (18c) yields

$$2x_{11} + x_{13} + x_{14} + x_{15} + 2\overline{x}_{22} + \overline{x}_{32} + \overline{x}_{42} \geq 4 \ . \qquad (20)$$

If we instead apply the resolution rule on (18c) with (18a) and then resolve the resulting constraint with (18b), we get

$$4x_{11} + x_{13} + x_{14} + x_{15} + 2\overline{x}_{22} + \overline{x}_{32} + \overline{x}_{42} \geq 4 \ . \qquad (21)$$

Note that the difference between (20) and (21) is that the coefficient of x_{11} is 4 instead of 2 in the latter, resulting in a strictly weaker constraint.

It is possible to force a saturation-based solver to find short proofs for formulas with obfuscated constraints (17a)–(17b) (in particular, by hard-coding a specific decision order for the variables), but this is nothing that a solver with default heuristics is currently able to do. This suggests that in addition to carefully choosing the set of derivation rules, optimizing the order in which these rules are applied during search is an important part of improving pseudo-Boolean solvers further.

6 Concluding Remarks

In this work we study the relative strength of division and saturation in pseudo-Boolean reasoning. We show that there are formulas for which PB solvers using division can be exponentially faster than solvers using saturation. In the other direction, we prove that the number of division steps needed to simulate a single saturation step can be exponential, but leave open the question of whether saturation-based solvers can ever be strictly stronger than division-based solvers.

By necessity, the formulas we use to obtain these results are crafted so as to be amenable to rigorous mathematical analysis. It would be nice to find more natural benchmarks, and also to study whether the difference in reasoning power between division and saturation ever comes into play in an applied context. Our limited experiments on crafted benchmarks indicate that other aspects of the search heuristics can easily become more important, but this also points to room for improvement of these heuristics (perhaps by, e.g., adaptively choosing between, or combining, division and saturation).

Acknowledgements

We are grateful to Jan Elffers and Marc Vinyals for extensive and enlightening discussions about cutting planes and pseudo-Boolean solving. We would also like to thank the anonymous reviewers for *IJCAI* and *Pragmatics of SAT* for several insightful comments that helped improve the exposition considerably.

Part of this work was carried out while the second and third authors visited the Simons Institute for the Theory of Computing in association with the DIMACS/Simons Collaboration on Lower Bounds in Computational Complexity, which is conducted with support from the National Science Foundation. Our computational experiments used resources provided by the Swedish National Infrastructure for Computing (SNIC).

The first and second authors were funded by the Swedish Research Council (VR) grant 2016-00782. The second author was also supported by the Knut and Alice Wallenberg grant KAW 2016.0066 and the VR grant 621-2012-5645. The third author was partially supported by the ISF grant 1162/15.

References

[Atserias *et al.*, 2011] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*, 40:353–373, 2011.

[Bayardo Jr. and Schrag, 1997] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, 1997.

[Beame *et al.*, 2004] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.

[Biere *et al.*, 2009] Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[Buss *et al.*, 2008] Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning. *Logical Methods in Computer Science*, 4(4:13), 2008.

[Chai and Kuehlmann, 2005] Donald Chai and Andreas Kuehlmann. A fast pseudo-Boolean constraint solver. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(3):305–317, 2005.

[Cook *et al.*, 1987] William Cook, Collette Rene Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987.

[Cook, 1971] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71)*, pages 151–158, 1971.

[Dixon and Ginsberg, 2002] Heidi E. Dixon and Matthew L. Ginsberg. Inference methods for a pseudo-Boolean satisfiability solver. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02)*, pages 635–640, 2002.

[Eén and Sörensson, 2006] Niklas Eén and Niklas Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, 2006.

[Elffers and Nordström, 2018] Jan Elffers and Jakob Nordström. Divide and conquer: Towards faster pseudo-Boolean solving. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI '18)*, pages 1291–1299, 2018.

[Haken, 1985] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, 1985.

[Hooker, 1988] John N. Hooker. Generalized resolution and cutting planes. *Annals of Operations Research*, 12(1):217–239, 1988.

[Hooker, 1992] John N. Hooker. Generalized resolution for 0-1 linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 6(1):271–286, 1992.

[Jovanovic and de Moura, 2013] Dejan Jovanovic and Leonardo de Moura. Cutting to the chase solving linear integer arithmetic. *Journal of Automated Reasoning*, 51(1):79–108, 2013.

[Le Berre and Parrain, 2010] Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, 2010.

[Manquinho and Marques-Silva, 2006] Vasco M. Manquinho and João Marques-Silva. On using cutting planes in pseudo-Boolean optimization. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:209–219, 2006.

[Marques-Silva and Sakallah, 1999] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[Martins *et al.*, 2014] Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-WBO: A modular MaxSAT solver. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2014.

[Mikša and Nordström, 2014] Mladen Mikša and Jakob Nordström. Long proofs of (seemingly) simple formulas. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2014.

[Moskewicz *et al.*, 2001] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, 2001.

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

[Sakai and Nabeshima, 2015] Masahiko Sakai and Hidetomo Nabeshima. Construction of an ROBDD for a PB-constraint in band form and related techniques for PB-solvers. *IEICE Transactions on Information and Systems*, 98-D(6):1121–1127, 2015.

[Sheini and Sakallah, 2006] Hossein M. Sheini and Karem A. Sakallah. Pueblo: A hybrid pseudo-Boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):165–189, 2006.

[Vinyals *et al.*, 2018] Marc Vinyals, Jan Elffers, Jesús Giráldez-Cru, Stephan Gocht, and Jakob Nordström. In between resolution and cutting planes: A study of proof systems for pseudo-Boolean SAT solving. In *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT '18)*, volume 10929 of *Lecture Notes in Computer Science*, pages 292–310. Springer, 2018.