In Between Resolution and Cutting Planes: A Study of Proof Systems for Pseudo-Boolean SAT Solving

Marc Vinyals[2], Jan Elffers[1], Jesús Giráldez-Cru[1], Stephan Gocht[1], and Jakob Nordström[1(✉)]

[1] KTH Royal Institute of Technology, Stockholm, Sweden
{elffers,giraldez,gocht,jakobn}@kth.se
[2] Tata Institute of Fundamental Research, Mumbai, India
marc.vinyals@tifr.res.in

Abstract. We initiate a proof complexity theoretic study of subsystems of cutting planes (CP) modelling proof search in conflict-driven pseudo-Boolean (PB) solvers. These algorithms combine restrictions such as that addition of constraints should always cancel a variable and/or that so-called saturation is used instead of division. It is known that on CNF inputs cutting planes with cancelling addition and saturation is essentially just resolution. We show that even if general addition is allowed, this proof system is still polynomially simulated by resolution with respect to proof size as long as coefficients are polynomially bounded.

As a further way of delineating the proof power of subsystems of CP, we propose to study a number of *easy* (but tricky) instances of problems in NP. Most of the formulas we consider have short and simple tree-like proofs in general CP, but the restricted subsystems seem to reveal a much more varied landscape. Although we are not able to formally establish separations between different subsystems of CP—which would require major technical breakthroughs in proof complexity—these formulas appear to be good candidates for obtaining such separations. We believe that a closer study of these benchmarks is a promising approach for shedding more light on the reasoning power of pseudo-Boolean solvers.

1 Introduction

The efficiency of modern Boolean satisfiability (SAT) solvers is one of the most fascinating success stories in computer science. The SAT problem lies at the foundation of the theory of NP-completeness [13], and as such is believed to be completely beyond reach from a computational complexity point of view. Yet solvers based on *conflict-driven clause learning (CDCL)* [4,38,40] are nowadays used routinely to solve instances with millions of variables.

From a theoretical point of view, it is an intriguing question *how to explain* the performance of state-of-the-art SAT solvers, and unfortunately our understanding of this remains quite limited. Perhaps the only tool currently available

O. Beyersdorff and C. M. Wintersteiger (Eds.): SAT 2018, LNCS 10929, pp. 292–310, 2018.
https://doi.org/10.1007/978-3-319-94144-8_18

for giving rigorous answers to such questions is provided by *proof complexity* [14], where one essentially ignores the question of algorithmic proof search and instead studies the power and limitations of the underlying method of reasoning.

Conflict-Driven Clause Learning and Resolution. It is well-known (see, e.g., [5]) that CDCL solvers search for proofs in the proof system *resolution* [7]. Ever since resolution-based SAT solvers were introduced in [16,17,46], subsystems of resolution corresponding to these algorithms, such as tree-like and regular resolution, have been studied. Exponential lower bounds for general resolution proofs were established in [11,29,51], and later it was proven that general resolution is exponentially stronger than regular resolution, which in turn is exponentially stronger than tree-like resolution (see [1,6,52] and references therein). More recently, CDCL viewed as a proof system was shown to simulate general resolution efficiently [3,43] (i.e., with at most a polynomial blow-up), though an algorithmic version of this result seems unlikely in view of [2].

A problem that is arguably even more intriguing than the analysis of CDCL solver performance is why attempts to build SAT solvers on stronger methods of reasoning than resolution have had such limited success so far. Resolution lies very close to the bottom in the hierarchy of proof systems studied in proof complexity, and even quite a limited extension of this proof system with algebraic or geometric reasoning holds out the prospect of exponential gains in performance.

Pseudo-Boolean Solving and Cutting Planes. In this paper we consider one such natural extension to *pseudo-Boolean (PB) solving* using linear inequalities over Boolean variables with integer coefficients, which is formalized in the proof system *cutting planes (CP)* [10,15,28]. By way of a brief overview, Hooker [31,32] considered generalizations of resolution to linear constraints and investigated the completeness of such methods. More general algorithms were implemented by Chai and Kuehlman [9], Sheini and Sakallah [49], and Dixon et al. [18–20]. The focus in all of these papers is mostly on algorithmic questions, however, and not on properties of the corresponding proof systems.

Papers on the proof complexity side have studied tree-like cutting planes [34] and CP with bounded constant terms in the inequalities [27], and resolution has been shown to simulate cutting planes when this upper bound is constant [30]. Exponential lower bounds for cutting planes with coefficients of polynomial magnitude were obtained in [8], and for general cutting planes with coefficients of arbitrary size strong lower bounds were proven in [45] and (very recently) in [26,33]. These papers consider more general derivation rules than are used algorithmically, however, and in contrast to the situation for resolution we are not aware of any work analysing the proof complexity of subsystems of CP corresponding to the reasoning actually being used in pseudo-Boolean solvers.

Our Contributions. We initiate a study of proof systems intended to capture the reasoning in pseudo-Boolean solvers searching for cutting planes proofs. In this work we focus on *cdcl-cuttingplanes* [21] and *Sat4j* [36,48], which are the two CP-based solvers that performed best in the relevant satisfiability problems

category *DEC-SMALLINT-LIN* in the *Pseudo-Boolean Competition 2016* [44].[1] Our subsystems of CP combine algorithmically natural restrictions such as that addition should always cancel a variable and/or that *saturation* is used instead of the more expensive to implement division rule. We stress that these derivation rules are nothing new—indeed, the point is that they are already used in practice, and they are formally defined in, e.g., the excellent survey on pseudo-Boolean solving [47]. Our contribution is to initiate a systematic study of concrete combinations of these rules, using tools from proof complexity to establish concrete limitations on what solvers using these rules can achieve.

PB solvers typically perform poorly on inputs in conjunctive normal form (CNF), and it has been known at least since [31,32] that in this case CP with cancelling addition and saturation degenerates into resolution. We observe that strengthening just one of these rules is not enough to solve this problem: CP with cancelling addition and division is easily seen still to be resolution, and resolution can also polynomially simulate the saturation rule plus unrestricted additions as long as the coefficients are of polynomial magnitude. The issue here is that while all versions of CP we consider are *refutationally complete*, meaning that they can prove unsatisfiability of an inconsistent set of constraints, the subsystems of CP are not *implicationally complete*, i.e., even though some linear constraint is implied by a set of other constraints there might be no way of deriving it. This makes reasoning in these subsystems very sensitive to exactly how the input is encoded. Thus, a strong conceptual message of our paper is that in order to function robustly over a wide range of input formats (including, in particular, CNF), PB solvers will need to explore a stronger set of reasoning rules.

In a further attempt to understand the relative strength of these subsystems of cutting planes, we present some (to the best of our knowledge) new combinatorial formulas encoding NP-complete problems, but with the concrete instances chosen to be "obviously" unsatisfiable. We then investigate these formulas, as well as the even colouring formulas in [37], from the point of view of proof complexity. Most of these formulas have very short and simple proofs in general cutting planes, and these proofs are even tree-like. With some care the applications of addition in these proofs can also be made cancelling, but having access to the division rule rather than the saturation rule appears critical. Although we are not able to establish any formal separations between the subsystems of cutting planes that we study (other than for the special case of CNF inputs as noted above), we propose a couple of formulas which we believe are promising candidates for separations. Obtaining such results would require fundamentally new techniques, however, since the tools currently available for analysing CP cannot distinguish between subsystems defined in terms of different sets of syntactic rules.[2]

[1] There is now an updated version of *cdcl-cuttingplanes* called *RoundingSat* [24], but any theoretical claims we make in this paper hold for this new version also.

[2] Essentially all lower bound proofs for CP work for any semantically sound proof system operating on pseudo-Boolean constraints, completely ignoring the syntactic rules, and the one exception [25] that we are aware of uses a very specific trick to separate fully semantic (and non-algorithmic) CP from the syntactic version.

We also consider these formulas for other ranges of parameter values and show that for such values the formulas are very easy even for the weakest subsystems of CP that we consider. This would seem to imply that solving such instances should be well within reach for *cdcl-cuttingplanes* and *Sat4j*. However, as reported in [22] many of these instances are instead very challenging in practice. This suggests that in order to make significant advances in pseudo-Boolean solving one crucial aspect is to make full use of the division rule in cutting planes, and we believe that further study of these benchmarks is a promising approach for gaining a deeper understanding of the theoretical reasoning power of pseudo-Boolean solvers implementing conflict-driven proof search.

Organization of This Paper. We discuss conflict-driven proof search in resolution and cutting planes and give formal definitions of proof systems in Sect. 2. In Sect. 3 we prove simulation results for different subsystems of CP, and in Sect. 4 we present our new combinatorial formulas providing candidates for separations. We make some brief concluding remarks in Sect. 5. We refer the reader to the upcoming full-length version of the paper for all missing proofs.

2 Proof Systems for Pseudo-Boolean SAT Solving

Let us start by giving a more formal exposition of the proof systems studied in this paper. Our goal in this section is to explain to complexity theorists without much prior exposure to applied SAT solving how these proof systems arise naturally in the context of pseudo-Boolean (PB) solving, and to this end we start by reviewing resolution and conflict-driven clause learning (CDCL) solvers. By necessity, our treatment is very condensed, but an excellent reference for more in-depth reading on PB solving is [47], and more details on proof complexity material relevant to this paper can be found, e.g., in [41,42].

We use the standard notation $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ and $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$ for natural numbers and positive natural numbers, respectively, and write $[n] = \{1, 2, \ldots, n\}$ and $[n, m] = \{n, n+1, \ldots, m\}$ for $m, n \in \mathbb{N}^+$, $m > n$.

Resolution and Conflict-Driven Clause Learning. Throughout this paper we identify 1 with *true* and 0 with *false*. A *literal* over a Boolean variable x is either a *positive literal* x or a *negative* or *negated literal* $\overline{x}$. It will also be convenient to write x^σ, $\sigma \in \{0, 1\}$, to denote $x^1 = x$ and $x^0 = \overline{x}$. A *clause* $C = \ell_1 \vee \cdots \vee \ell_k$ is a disjunction of literals over pairwise disjoint variables. A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. We write $\mathit{Vars}(F)$ to denote the set of variables appearing in a formula F. We think of clauses and formulas as sets, so that order is irrelevant and there are no repetitions.

We can represent a partial truth value assignment ρ as the set of literals set to true by ρ. We write $\rho(x^\sigma) = 1$ if $x^\sigma \in \rho$, $\rho(x^\sigma) = 0$ if $x^{1-\sigma} \in \rho$, and $\rho(x^\sigma) = *$ otherwise (i.e., when ρ does not assign any truth value to x). A clause C is satisfied by ρ if it contains some literal set to true by ρ; falsified if ρ sets all literals in C to false; and undetermined otherwise. The *restricted* clause $C\restriction_\rho$ is

the trivial clause 1 if ρ satisfies C and otherwise C with all literals falsified by ρ removed, i.e., $C|_\rho = C \setminus \{x^\sigma \mid x^{1-\sigma} \in \rho\}$. A *unit clause* is a clause with only one literal. We say that C is *unit under* ρ if $C|_\rho = \{x^\sigma\}$ is a unit clause, and if so C is also said to *propagate* x^σ *under* ρ.

A *resolution refutation* π of F is a sequence of clauses $\pi = (D_1, D_2, \ldots, D_L)$ such that $D_L = \bot$ is the empty clause without literals and each D_i is either an *axiom clause* $D_i \in F$ or a *resolvent* on the form $D_i = B \vee C$ derived from $D_j = B \vee x$ and $D_k = C \vee \overline{x}$ for $j, k < i$ by the *resolution rule*

$$\frac{B \vee x \qquad C \vee \overline{x}}{B \vee C} \ . \tag{1}$$

It is sometimes convenient to add also a *weakening rule*

$$\frac{B}{B \vee C} \ , \tag{2}$$

which allows to derive any strictly weaker clause from an already derived clause, but it is not hard to show that any use of weakening in a resolution refutation can be eliminated without loss of generality. It is a standard fact that resolution is *implicationally complete*, meaning that a clause C can be derived from a formula F if and only if F semantically implies C.[3] In particular, F is unsatisfiable if and only if there exists a resolution refutation of F.

The *length* $L(\pi)$ of a refutation π is the number of clauses in it. Viewing the list of clauses π as annotated with explanations how they were obtained, we define an associated directed acyclic graph (DAG) G_π with vertices $\{v_1, v_2, \ldots, v_L\}$ labelled by the clauses $\{D_1, D_2, \ldots, D_L\}$ and with edges from resolved clauses to resolvents. We say that π is *tree-like* if G_π is a tree, or, equivalently, if every clause D_i is used at most once as a premise in the resolution rule (repetitions of clauses in π are allowed; i.e., different vertices can be labelled by the same clause). The *(clause) space* at step i in π is the number of clauses D_j, $j < i$ used to obtain resolvents $D_{j'}$, $j' \geq i$, plus 1 for the clause D_i itself, and the space $Sp(\pi)$ of the refutation is the maximal space at any step in π.

Turning next to CDCL solvers, we give a simplified description below that is sufficient for our needs—a more complete (theoretical) treatment can be found in [23]. In one sentence, a CDCL solver running on a CNF formula F repeatedly decides on variable assignments and propagates values that follow from such assignments until a clause is falsified, at which point a learned clause is added to the clause database $\mathcal{D}$ (where we always have $F \subseteq \mathcal{D}$) and the search backtracks. In a bit more detail, the solver maintains a current partial assignment ρ, where every assignment also has a *decision level* (the initial state is at decision level 0 with $\rho = \emptyset$ and $\mathcal{D} = F$). If there is a clause $C \in \mathcal{D}$ that is unit under ρ, the solver adds the propagated literal $x^\sigma = C|_\rho$ to ρ with *reason clause* C and repeats the check for unit clauses until either (i) some clause $D \in \mathcal{D}$ is falsified

[3] In case the definition of resolution without weakening is used, the notion of implicational completeness is adapted in the natural way to mean that resolution can derive either C or some clause C' that *subsumes* C, i.e., such that $C' \subsetneq C$.

by the current assignment (referred to as a *conflict clause*), or else (ii) there are no propagating clauses. In the latter case the solver makes a *decision* $y = \nu$ and adds y^ν to ρ with decision level increased by 1 (unless there are no more variables left, in which case ρ is a satisfying assignment for F). In the former case, the solver instead performs a *conflict analysis* as described next.

Suppose for concreteness that the last propagated literal in ρ before reaching the conflict clause D was x^σ with reason clause $C = C^* \vee x^\sigma$. Since this propagation caused a conflict the variable x must appear with the opposite sign in D, which can hence be written on the form $D = D^* \vee x^{1-\sigma}$. The solver can therefore resolve $C^* \vee x^\sigma$ and $D^* \vee x^{1-\sigma}$ to get $D' = C^* \vee D^*$, after which x^σ is removed from ρ. We refer to D' as the new *conflict-side clause*. During conflict analysis the conflict-side clause D' is resolved in reverse chronological order with the reason clauses propagating literals in D' to false, and these literals are removed one by one from ρ. An important invariant during this process is that the current conflict-side clause is always falsified by the partial assignment ρ after removing the literal just resolved over. Therefore, every derived clause on the conflict side provides an "explanation" why the corresponding partial assignment fails.

The conflict analysis loop ends when the conflict-side clause contains only one literal from the current decision level at which point the solver *learns* this clause and adds it to the database $\mathcal{D}$. By the invariant, this learned clause is still falsified by ρ, and so the solver removes further literals from ρ in reverse chronological order until the decision level decreases to that of the second largest decision level of any literal in the learned clause. At this point the solver returns from conflict analysis and resumes the main loop described above. By design it now holds that the newly learned clause immediately causes unit propagation, flipping some previously assigned literal to the opposite value. Learned clauses having this property are called *asserting*, and a common feature of essentially all clause learning schemes used in practice is that they learn such asserting clauses.

The CDCL solver terminates either when it finds a satisfying assignment or when it detects unsatisfiability by learning the empty clause $\bot$ (or, more precisely, when it reaches a conflict at decision level 0, in which case the conflict analysis is guaranteed to derive the empty clause). There are, of course, lots of details that we are omitting above. The important conclusions, as we prepare to generalize the description of CDCL to a pseudo-Boolean context, is that the CDCL solver decides on variables and propagates values based on the clauses currently in the database, and that when a conflict is reached a new clause is added to the database obtained by a resolution derivation from the conflict and reason clauses. This means that from any run of CDCL on an unsatisfiable formula F we can extract a resolution refutation of F.

Cutting Planes and Pseudo-Boolean Solving. Recall that throughout this paper we are considering pseudo-Boolean constraints encoded as linear inequalities over Boolean variables with integral coefficients (and all linear inequalities discussed are assumed to be over $\{0, 1\}$-valued variables unless stated otherwise). In order to give a description of cutting planes that is suitable when we want to reason about pseudo-Boolean solvers, it is convenient to keep negated literals as objects in their own right, and to insist that all inequalities consist of positive

linear combinations of literals. Therefore, we will write all linear constraints in *normalized form*

$$\sum_{i\in[n],\,\sigma\in\{0,1\}} a_i^\sigma x_i^\sigma \geq A \ , \tag{3}$$

where for all $a_i^\sigma \in \mathbb{N}$ with $i \in [n]$ and $\sigma \in \{0,1\}$ at least one of a_i^0 or a_i^1 equals 0. (variables occur only with one sign in any given inequality), and where the right-hand constant term $A \in \mathbb{N}$ is called the *degree of falsity* (or just *degree*). Note that the normalization is only a convenient form of representation and does not affect the strength of the proof system. If the input is a CNF formula F we just view every clause $C = x_1^{\sigma_1} \vee \cdots \vee x_w^{\sigma_w}$ as a linear constraint $x_1^{\sigma_1} + \cdots + x_w^{\sigma_w} \geq 1$, i.e., a constraint on the form (3) with $a_i^\sigma \in \{0,1\}$ and $A = 1$.

When generalizing CDCL to a pseudo-Boolean setting we want to build a solver that decides on variable values and propagates forced values until conflict, at which point a new linear constraint is learned and the solver backtracks. The main loop of a conflict-driven PB solver can be made identical to that of a CDCL solver, except that we change the word "clause" to "constraint." However, a naive generalization of the conflict analysis does not work. For an example of this, suppose we have $\rho = \{\overline{x}_1, x_2, \overline{x}_3\}$ under which $x_1 + 2\overline{x}_2 + x_3 + 2x_4 + 2x_6 \geq 3$ unit propagates x_6 to true, causing a conflict with $x_3 + 2x_5 + 2\overline{x}_6 \geq 3$. By analogy with the CDCL conflict analysis, we "resolve" (i.e., add and normalize) these two constraints to eliminate x_6, yielding $x_1 + 2\overline{x}_2 + 2x_3 + 2x_4 + 2x_5 \geq 3 + 3 - 2 = 4$ (since $2x_6 + 2\overline{x}_6 = 2$). But now the important invariant that the derived constraint is falsified by the current partial assignment fails, because the new constraint is not falsified by $\rho = \{\overline{x}_1, x_2, \overline{x}_3\}$! There are different ways of modifying the pseudo-Boolean conflict analysis to address this problem, and these different approaches are partly reflected in the different proof systems studied in this paper.

Starting with the most general version of the *cutting planes* proof system used in the proof complexity literature, using the normalized form (3) we can define the derivation rules[4] to be *literal axioms*

$$\frac{}{x_i^\sigma \geq 0} \ , \tag{4a}$$

linear combination

$$\frac{\sum_i a_i^\sigma x_i^\sigma \geq A \qquad \sum_i b_i^\sigma x_i^\sigma \geq B}{\sum_i (\alpha a_i^\sigma + \beta b_i^\sigma) x_i^\sigma \geq \alpha A + \beta B} \ \alpha, \beta \in \mathbb{N}^+ \ , \tag{4b}$$

and *division*

$$\frac{\sum_i a_i^\sigma x_i^\sigma \geq A}{\sum_i \lceil a^\sigma / \alpha \rceil x_i^\sigma \geq \lceil A/\alpha \rceil} \ \alpha \in \mathbb{N}^+ \ , \tag{4c}$$

[4] Attentive readers might note that division looks slightly stronger in our definition than the standard rule in the proof complexity literature, but the two versions are easily verified to be equivalent up to a linear factor in length. It is important to note that multiplication is only ever performed in combination with addition.

where in the linear combination rule we tacitly assume that the cancellation rule $x^\sigma + x^{1-\sigma} = 1$ is applied to bring the derived constraint into normalized form, as in the example we just saw. Just as in this example, for any linear combination that arises during conflict analysis it will be the case that there is a literal x_i^σ for which $\alpha a_i^\sigma = \beta b_i^{1-\sigma} > 0$. We say that this is an instance of *cancelling linear combination* since the variable x_i vanishes, and we also require for such linear combinations that α and β are chosen so that $\alpha a_i^\sigma = \beta b_i^{1-\sigma}$ is the least common multiple of a_i^σ and $b_i^{1-\sigma}$. We remark that this is also referred to as *generalized resolution* in the literature [31,32], since it is a natural generalization of (1) from disjunctive clauses to general linear constraints, and we will sometimes refer to the resulting constraint as a *(generalized) resolvent*.

We want to highlight that in the division rule (4c) we can divide *and round up* to the closest integer, since we are only interested in $\{0,1\}$-valued solutions. This division rule is where the power of cutting planes lies. And indeed, this is how it must be, since the other rules are sound also for real-valued variables, and so without the division rule we would not be able to distinguish sets of linear inequalities that have real-valued solutions but no $\{0,1\}$-valued solutions.

Pseudo-Boolean solvers such as *Sat4j* [36,48] and *cdcl-cuttingplanes* [21] do not implement the full set of cutting planes derivation rules as described above, however. In proofs generated by these solvers the linear combinations will always be *cancelling*. Division is used in *cdcl-cuttingplanes* only in a restricted setting to ensure that the learned constraint is always conflicting, and *Sat4j* omits this rule pretty much completely and instead applies the *saturation* rule

$$\frac{\sum_{(i,\sigma)} a_i^\sigma x_i^\sigma \geq A}{\sum_{(i,\sigma)} \min\{a_i^\sigma, A\} \cdot x_i^\sigma \geq A} \ , \tag{5a}$$

saying that no coefficient on the left need be larger than the degree on the right. (For instance, saturation applied to $3x_1 + x_2 + x_3 \geq 2$ yields that $2x_1 + x_2 + x_3 \geq 2$ holds.) As the division rule, the saturation rule is sound only for integral solutions. It is an interesting question how the division and saturation rules are related. Saturation can be simulated by division, but it is not clear whether this simulation can be made efficient in general. In the other direction, we give examples in this paper of when division is exponentially stronger than saturation.

We remark that another rule that is important in practice is *weakening*

$$\frac{\sum_{(i,\sigma)} a_i^\sigma x_i^\sigma \geq A}{\sum_{(i,\sigma)\neq(i^*,\sigma^*)} a_i^\sigma x_i^\sigma \geq A - a_{i^*}^{\sigma^*}} \ , \tag{5b}$$

which—perhaps somewhat counter-intuitively—is used during conflict analysis to maintain the invariant that the constraint being learned is conflicting with respect to the current partial assignment. In contrast to the weakening rule in resolution, the rule (5b) is crucial for pseudo-Boolean solvers, but since this rule

can be implemented using (4a) and a cancelling linear combination we do not need to include it in our formal proof system definitions.

In order to try to understand the reasoning power of pseudo-Boolean solvers such as *cdcl-cuttingplanes* and *Sat4j*, in this paper we study the following four subsystems of cutting planes, where for brevity we will write just *cancellation* instead of *cancelling linear combination*:

General CP: Rules (4a), (4b), and (4c).
CP with saturation: Rules (4a), (4b), and (5a).
CP with saturation and cancellation: Rules (4a) and (5a) plus the cancelling version of (4b); essentially corresponding to *Sat4j*.
CP with division and cancellation: Rules (4a) and (4c) plus the cancelling version of (4b); strong enough to capture *cdcl-cuttingplanes*.

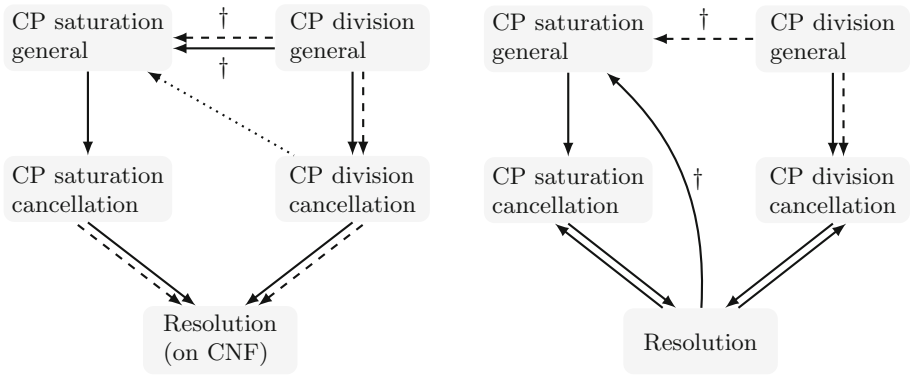
General cutting planes is refutationally complete in that it can disprove any inconsistent set of linear inequalities [28]: One can show that there is no $\{0,1\}$-valued solution by using the cutting planes rules (4a)–(4c) to derive the contradiction $0 \geq A$ for some $A > 0$, which is the pseudo-Boolean equivalent of the empty clause, from the given linear inequalities. The *length* of such a cutting planes refutation is the total number of inequalities in it, and the *size* also sums the sizes of all coefficients (i.e., the bit size of representing them). We can also define a *line space* measure analogous to the clause space measure counting the number of inequalities in memory during a proof.

It is not hard to show—as we will argue shortly—that the three restricted versions of CP defined above are also refutationally complete. However, while general cutting planes is also implicationally complete [10], meaning that it can derive any inequality that is implied by a set of linear equations, the subsystems we consider are not even *weakly implicationally complete*.

Let us pause to explain what we mean by this terminology. For disjunctive clauses C and D it is not hard to see that the only way C can imply D is if $C \subseteq D$. In a pseudo-Boolean context, however, there are infinitely many ways to express a linear threshold function over the Boolean hypercube as a linear inequality (for instance, by multiplying the inequality by an arbitrary positive integer). We say, therefore, that a PB proof system is *weakly implicationally complete* if when some set of inequalities implies $\sum_{(i,\sigma)} a_i^\sigma x_i^\sigma \geq A$ it holds that the proof system can derive some potentially syntactically different inequality $\sum_{(i,\sigma)} b_i^\sigma x_i^\sigma \geq B$ implying $\sum_{(i,\sigma)} a_i^\sigma x_i^\sigma \geq A$, and that it is *(strongly) implicationally complete* if it can derive a constraint on the exact syntactic form $\sum_{(i,\sigma)} a_i^\sigma x_i^\sigma \geq A$.

Returning to our previous discussion, given the constraint $\sum_{i=1}^{k} x_i \geq d$ written as a set of disjunctive clauses $\left\{\sum_{i\in S} x_i \geq 1 \,\middle|\, S \subseteq [k], |S| = k - d + 1\right\}$ (in pseudo-Boolean notation), it is not hard to see that there is no way CP with cancellation can derive any inequality implying the former encoding from the constraints in the latter encoding [31].[5] A slightly less obvious fact, which we

[5] This is so since the only possibility to apply cancelling linear combinations is to use literal axioms (4a) yielding (trivial) constraints on the form $\sum_{i\in S} x_i \geq 0$ for $|S| \geq 0$, and the set of such constraints is invariant under both division and saturation.



(a) Over pseudo-Boolean inputs (b) Over CNF inputs

Fig. 1. Relations between proof systems. $A \longrightarrow B$: A polynomially simulates B; $A \dashrightarrow B$: B cannot simulate A (there is an exponential separation); $A \cdots\!\blacktriangleright B$: candidate for a separation, †: known only for coefficients of polynomial magnitude.

shall prove in Sect. 3, is that even with general addition and saturation it is not possible to recover a cardinality constraint from its CNF encoding.

We want to emphasize again that we make no claims of originality when it comes to defining the derivation rules—they arise naturally in the context of pseudo-Boolean solving, and indeed all of them are described in [47]. However, we are not aware of any previous work defining and systematically studying the subsystems of CP described above from a proof complexity point of view, i.e., proving upper and lower bounds on proof resources. This is the purpose of the current paper, and we study the strength of these proof systems both for CNF inputs and general (linear) pseudo-Boolean inputs.

As a final remark for completeness, we want to point out that one further important rule, which is used, e.g., in [9], is *rounding to cardinality constraints* We leave as future work a study of formal proof systems using this rule.

3 Relations Between Subsystems of Cutting Planes

We now proceed to examine how having saturation instead of division and/or requiring linear combinations to be cancelling affects the reasoning power of cutting planes. The conclusions of this section are pictorially summarized in Fig. 1.

For starters, it is an easy observation that all the subsystems of cutting planes that we consider can simulate resolution when the input is in CNF, and we show that this is still the case when we start with a pseudo-Boolean input for cutting planes and the straightforward encoding into CNF of that input for resolution. This is immediate if we have the division rule, but in fact it is not hard to prove that the simulation also works with saturation.

Let us make these claims formal. We say that a set of clauses $\widehat{I}$ *represents* a pseudo-Boolean constraint I if both expressions are over the same variables[6] and encode the same Boolean function, and a CNF formula $\widehat{F}$ is said to represent a set of inequalities F if $\widehat{F} = \bigcup_{I \in F} \widehat{I}$ (where it is important to note that each CNF subformula $\widehat{I}$ represents one linear constraint I). Then the next lemma says that even if each linear constraint $I \in F$ is rewritten to a semantically equivalent but obfuscated constraint I' in some awkward way, but encoded into a CNF representation $\widehat{F}$ in some nice way, it is still the case that even the weakest version of CP applied to $F' = \bigcup I'$ can efficiently simulate resolution on $\widehat{F}$.

Lemma 1. *Let F be a set of pseudo-Boolean constraints over n variables and let $\widehat{F}$ be any CNF representation of F as described above. Then if there is a resolution refutation $\widehat{\pi}$ of $\widehat{F}$ in length L and clause space s, there is also a CP refutation π of F in length $\mathrm{O}(nL)$ and line space $s+\mathrm{O}(1)$ using only cancellation and saturation. If $\widehat{\pi}$ is tree-like, then π is also tree-like.*

It follows from Lemma 1 that CP with saturation is refutationally complete.

Corollary 2. *Any unsatisfiable set of pseudo-Boolean constraints over n variables has a tree-like CP refutation in length $\mathrm{O}(n2^n)$ and line space $\mathrm{O}(n)$ using cancellation and saturation.*

In the other direction, cutting planes with cancellation is equivalent to resolution when restricted to CNF inputs, and this is so regardless of whether division or saturation is used. The reason for this is that cancelling linear combinations of disjunctive clauses can only produce inequalities with degree of falsity 1, which are equivalent to clauses. This is essentially just an observation from [31] rewritten in the language of proof complexity, but let us state it here for the record.

Lemma 3. *If cutting planes with cancellation and either division or saturation can refute a CNF formula F in length L and line space s, then there is a resolution refutation of F in length L and clause space s.*

We can use this observation to show that systems allowing general linear combinations can be strictly stronger than systems with cancellation. To see this, consider *subset cardinality formulas* [39,50,53] defined in terms of $0/1$ $n \times n$ matrices $A = (a_{i,j})$, which have variables $x_{i,j}$ for all $a_{i,j} = 1$ and constraints claiming that in each row there is a majority of positive variables but in each column there is a majority of negative variables, i.e.,

$$\sum\nolimits_{j \in R_i} x_{i,j} \geq \lceil |R_i|/2 \rceil \qquad i \in [n] \tag{6a}$$

$$\sum\nolimits_{i \in C_j} x_{i,j} \leq \lfloor |C_i|/2 \rfloor \qquad j \in [n] \tag{6b}$$

where $R_i = \{j \mid a_{i,j} = 1\}$ and $C_j = \{i \mid a_{i,j} = 1\}$. In the case when all rows and columns have $2k$ variables, except for one row and column that have $2k+1$

[6] We do not allow encodings with extension variables, since then formulas are no longer semantically equivalent and it becomes very hard to make meaningful comparisons.

variables, these formulas are unsatisfiable and are easily refutable in general CP, but if the matrix is *expanding* in a certain sense, then resolution proofs require exponential length [39]. This yields the following corollary of Lemma 3.

Corollary 4. *There are formulas on n variables that can be refuted in length $\mathrm{O}(n)$ in general CP but require length $\exp(\Omega(n))$ in CP with cancellation.*

When it comes to comparing division versus saturation, it was observed in [9] that saturation can be simulated by repeated division. Working out the details, we obtain the following proposition.

Proposition 5. *If a set of pseudo-Boolean constraints has a CP refutation with saturation in length L and coefficients bounded by A, then there is a CP refutation with division in length AL.*

We remark that a direct simulation may lead to an exponential blow-up if the proof uses coefficients of exponential magnitude.

Our main contribution in this section is to show that when the input is in CNF, then cutting planes proofs with saturation and unrestricted addition can in fact be efficiently simulated by resolution assuming that all CP coefficients are of polynomial magnitude. Observe that this last condition also implies that the the degree of falsity has polynomial magnitude, which is the slightly more precise assumption used in the next theorem.

Theorem 6. *If a CNF formula F has a CP refutation π with saturation in length L and every constraint in π has degree of falsity at most A, then there is a resolution refutation of F in length $\mathrm{O}(AL)$.*

We can then use subset cardinality formulas again to separate CP with division from CP with saturation. The formal claim follows below, where the constant hidden in the asymptotic notation depends on the size of the coefficients.

Corollary 7. *There are formulas on n variables that can be refuted in length $\mathrm{O}(n)$ in general CP but require length $\exp(\Omega(n))$ in CP with saturation if all coefficients in the proofs have polynomial magnitude.*

The idea behind the proof of Theorem 6 is to maintain for every inequality with degree of falsity A a set of A clauses that implies the inequality. We simulate linear combination steps by resolving the sets of clauses corresponding to the two inequalities over the variables that cancel, and we do not do anything for saturation steps.

Note that this approach does not work if the input is not in CNF. For instance, if we start with the pseudo-Boolean constraint $x + y + z \geq 2$ with degree of falsity 2, which is equivalent to the clauses $(x \vee y) \wedge (y \vee z) \wedge (x \vee z)$, then it is not possible to pick any 2 out of these 3 clauses that would imply the inequality.

We remark that we do not know of any separation between CP with saturation and division except those exhibited by CNF formulas. These separations are

somewhat artificial in that they crucially use that the implicationally incomplete subsystems of CP cannot recover the the cardinality constraints "hidden" in the CNF encodings. In Sect. 4 we propose more natural candidates for separations between CP with division and cancellation and CP with saturation, where the difficulty would not be due to an "obfuscated" CNF encoding.

We conclude this section by the observation that any version of CP considered in this paper can easily refute any set of linear constraints that define an empty polytope over the reals, i.e., for which there is no real-valued solution. For general addition this is an immediate consequence of Farkas' lemma, and we can make the additions cancelling using the Fourier–Motzkin variable elimination procedure.

Lemma 8. *If a set of linear inequalities on n variables defines an empty polytope over the reals, then there is a tree-like CP refutation using only addition in length $\mathrm{O}(n)$ and space $\mathrm{O}(1)$, and a CP refutation using only cancelling addition in length $\mathrm{O}(n^2)$ and space $\mathrm{O}(n)$.*

As a consequence of Corollary 7 and Lemma 8 we obtain the following theorem.

Theorem 9. *CP with saturation is not (even weakly) implicationally complete.*

4 Tricky Formulas Based on Easy NP Instances

In this section we present candidates for achieving separations between the subsystems of cutting planes studied in this paper, and where these separations would not be a consequence of presenting pseudo-Boolean constraints as "obfuscated" CNF formulas but would highlight fundamental differences in pseudo-Boolean reasoning power between the proof systems.

All of our candidate formulas have short proofs for CP with division (and all refutations have constant-size coefficients unless stated otherwise), but for appropriately chosen parameter values it seems plausible that some of them are not possible to refute efficiently using the saturation rule. We also show that it is possible to chose other parameter values for these formulas to generate instances that are very easy in theory even for the weakest subsystem of CP that we consider. This is in striking contrast to what one can observe empirically when running pseudo-Boolean solvers on these instances, as reported in [22]—in practice, many of these theoretically easy instances appear to be very challenging.

Even Colouring. The even colouring formula $EC(G)$ [37] over a connected graph $G = (V, E)$ with all vertices of even degree consists of the constraints

$$\sum\nolimits_{e \in E(v)} x_e = \deg(v)/2 \qquad\qquad v \in V \tag{7}$$

(where $E(v)$ denotes the set of edges incident to v), claiming that each vertex has an equal number of incident 0- and 1-edges. The formula is unsatisfiable if and only if $|E|$ is odd, which we assume is always the case it what follows.

Even colouring formulas have short CP proofs: just add all positive and negative inequalities separately, divide by 2 and round up, and add the results. We can make the additions cancelling by processing inequalities in breadth-first order, alternating between positive and negative inequalities.

Proposition 10. *Tree-like CP with division and cancellation can refute* $EC(G)$ *in length* $\mathrm{O}(n)$ *and space* $\mathrm{O}(1)$.

If the graph is *t-almost bipartite*, by which we mean that removing t edges yields a bipartite graph, then we can make the proof work with saturation instead of division at the price of an exponential blow-up in t (which becomes a constant factor if t is constant).

Proposition 11. *If* G *is a* t*-almost bipartite graph then the formula* $EC(G)$ *can be refuted in length* $\mathrm{O}(2^t + n)$ *and space* $\mathrm{O}(t)$ *by CP with saturation and cancellation, and the refutation can be made tree-like in length* $\mathrm{O}(2^t n)$.

An example of such graphs are rectangular $m \times n$ grids (where edges wrap around the borders to form a torus) and where we subdivide one edge into a degree-2 vertex to get an odd number of edges. If both m and n are even, then the graph is bipartite except for 1 edge, so we have cutting planes proofs with saturation and cancellation of length $\mathrm{O}(mn)$, and if m is even and n is odd, then the graph is bipartite except for $m + 1$ edges, so we have proofs of length $\mathrm{O}(2^m + mn)$. In all cases even colouring formulas on grids have resolution refutations of length $2^{\mathrm{O}(m)}n$ and space $2^{\mathrm{O}(m)}$ which we can simulate.

We conjecture that these formulas are exponentially hard for CP with saturation when the graph is a square grid of odd side length, i.e., $m = n = 2\ell + 1$ (so that the graph is far from bipartite), or is a $2d$-regular random graph.

Vertex Cover. Recall that a *vertex cover* of a graph $G = (V, E)$ is a subset of vertices $V' \subseteq V$ such that every edge $(u, v) \in E$ is incident to some vertex in V'. A graph G has a vertex cover of size at most $S \in \mathbb{N}^+$ if and only if the formula $VC(G, S)$ given by the constraints

$$x_u + x_v \geq 1 \qquad (u, v) \in E; \tag{8a}$$

$$\sum\nolimits_{v \in V} x_v \leq S \tag{8b}$$

has a $\{0, 1\}$-valued solution.

We consider vertex cover instances over grid graphs $R_{m,n}$ Since a grid has degree 4 any cover must have size at least $mn/2$. This bound is not achievable when one dimension, say n, is odd, in which case the minimal cover size is $m\lceil n/2 \rceil$. We can choose the parameter S in (8b) in the interval $\left[mn/2, m\lceil n/2 \rceil - 1\right]$ to obtain unsatisfiable formulas with different levels of overconstrainedness.

Vertex cover formulas have short cutting planes proofs: add the horizontal edge inequalities (8a) for every row, divide by 2 (which rounds up the degree of falsity), and add all of these inequalities to find a contradiction with the upper bound (8b), and these additions can be reordered to be made cancelling.

Proposition 12. *CP with division and cancellation can refute* $VC(R_{m,n}, S)$ *with* n *odd and* $S < m\lceil n/2 \rceil$ *in length* $\mathrm{O}(mn)$ *and space* $\mathrm{O}(1)$.

A similar approach works with saturation instead of division, but since we cannot round up every row we need a stronger cover size constraint (8b).

Proposition 13. *Tree-like CP with saturation and cancellation is able to refute* $VC(R_{m,n}, S)$ *with* n *odd and* $S \leq \lfloor mn/2 \rfloor$ *in length* $\mathrm{O}(mn)$ *and space* $\mathrm{O}(1)$.

Alternatively, using what we find to be a rather nifty approach it turns out to be possible to derive all the 2^m clauses over the m variables corresponding to vertices in the first column, after which one can simulate a brute-force resolution refutation of this formula.

Proposition 14. *Tree-like CP with saturation and cancellation is able to refute* $VC(R_{m,n}, S)$ *with* n *odd and* $S < m\lceil n/2 \rceil$ *in length* $\mathrm{O}(2^m mn)$ *and space* $\mathrm{O}(m)$.

We conjecture that the exponential gap between Propositions 12 and 14 for $m = \Theta(n)$ and $S = m\lceil n/2 \rceil - 1$ is real and is due to the weakness of saturation.

Dominating Set. A *dominating set* of a graph $G = (V, E)$ is a subset of vertices $V' \subseteq V$ such that every vertex in $V \setminus V'$ has a neighbour in V'. G has a dominating set of size $S \in \mathbb{N}^+$ if and only if there is a $\{0, 1\}$-valued solution to the set of constraints $DS(G, S)$ defined as

$$x_v + \sum\nolimits_{u \in N(v)} x_u \geq 1 \qquad v \in V; \tag{9a}$$

$$\sum\nolimits_{v \in V} x_v \leq S \tag{9b}$$

We consider dominating set formulas over hexagonal grid graphs $H_{m,n}$, which can be visualized as brick walls. As it turns out these formulas have short proofs even in CP with saturation and cancellation, but the proofs are not obvious and the formulas have a surprisingly rich structure and present particularly challenging benchmarks in practice.

Since a hexagonal grid has degree 3, the minimum size of a dominating set is $\lceil |V|/4 \rceil = \lceil mn/4 \rceil$, so we set $S = \lfloor mn/4 \rfloor$. Whether these formulas are satisfiable depends on the largest power of 2 that divides m and n—also known as the 2-*adic valuation* or v_2. Formulas where $v_2(mn) = 1$ are unsatisfiable and can be refuted by adding all inequalities, and these additions can be made cancelling with some care.

Proposition 15. *Tree-like CP with cancellation can refute* $DS(H_{m,n}, \lfloor mn/4 \rfloor)$ *with* $v_2(mn) = 1$ *in length* $\mathrm{O}(mn)$ *and space* $\mathrm{O}(1)$.

Formulas where $v_2(mn) = 2$ are unsatisfiable and the proof follows by dividing the resulting inequalities in the previous proof by 2 and rounding up.

Proposition 16. *Tree-like CP with division and cancellation is able to refute* $DS(H_{m,n}, mn/4)$ *with* $v_2(mn) = 2$ *in length* $\mathrm{O}(mn)$ *and space* $\mathrm{O}(1)$.

When $v_2(n) \geq 2$ the dominating set must in fact define a *tiling* of the hexagonal grid. If furthermore $v_2(m) \geq 1$ then formulas are satisfiable. Among the remaining formulas some are satisfiable and some are not, and the next lemma sums up our knowledge in this matter.

Lemma 17. *Dominating set formulas over hexagonal grids are unsatisfiable if*

- $v_2(m) \geq 2$ *and* $v_2(n) = 1$, *or*
- $v_2(m) = 0$ *and* $v_2(n) \geq 3$ *and* $v_2(n) \leq v_2(4\lfloor m/4 \rceil)$, *or*
- $v_2(n) = 0$ *and* $v_2(m) \geq 3$ *and* $v_2(m) \leq v_2(4\lfloor n/4 \rceil)$.

We conjecture that Lemma 17 in fact provides an exact characterization.

To find CP refutations of the unsatisfiable dominating set instances, we can derive tiling constraints $x_v + \sum_{u \in N(v)} = 1$ for all vertices using only cancelling addition. CP with saturation and cancellation can then easily refute these formulas with tiling constraints in polynomial length.

Proposition 18. *If* $DS(H_{m,n}, mn/4)$ *is as in Lemma 17, then it can be refuted in length* $\mathrm{O}((nm)^2)$ *in CP with saturation and cancellation.*

5 Concluding Remarks

In this paper, we investigate subsystems of cutting planes motivated by pseudo-Boolean proof search algorithms. Using tools from proof complexity, we differentiate between the reasoning power of different methods and show that current state-of-the-art pseudo-Boolean solvers are inherently unable to exploit the full strength of cutting planes even in theory, in stark contrast to what is the case for CDCL solvers with respect to resolution.

Some of these limitations are in some sense folklore, in that it is known that pseudo-Boolean solvers perform badly on input in CNF, but we show that this is true for all natural restrictions suggested by current solvers that fall short of full-blown cutting planes reasoning. Also, we propose a number of new crafted benchmarks as a way of going beyond CNF-based lower bounds to study the inherent limitations of solvers even when given natural pseudo-Boolean encodings. We show how the parameters for these benchmarks can be varied to yield versions that appear to be hard or easy for different subsystems of cutting planes.

Although we cannot establish any formal separations between the subsystems of cutting planes studied in this paper—this would seem to require the development of entirely new proof complexity techniques—it is our hope that further investigations of these benchmarks could yield more insights into the power and limitations of state-of-the-art pseudo-Boolean solvers.

Acknowledgements. We are most grateful to Daniel Le Berre for long and patient explanations of the inner workings of pseudo-Boolean solvers, and to João Marques-Silva for helping us get an overview of relevant references for pseudo-Boolean solving. We would like to thank Susanna F. de Rezende, Arnold Filtser, and Robert Robere for helpful discussions on polytopes. We also extend our gratitude to

the *SAT 2018* anonymous reviewers for the many detailed comments that helped to improve the paper considerably.

Some empirical pseudo-Boolean solver experiments made within the context of this work were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at the High Performance Computing Center North (HPC2N) at Umeå University. For these experiments we also used the tool `CNFgen` [12,35], for which we gratefully acknowledge Massimo Lauria.

The first author performed part of this work while at KTH Royal Institute of Technology. All authors were funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611. The first author was also supported by the Prof. R Narasimhan post-doctoral award, and the fourth and fifth authors received support from Swedish Research Council grants 621-2012-5645 and 2016-00782.

References

1. Alekhnovich, M., Johannsen, J., Pitassi, T., Urquhart, A.: An exponential separation between regular and general resolution. Theory Comput. **3**(5), 81–102 (2007). preliminary version in STOC 2002
2. Alekhnovich, M., Razborov, A.A.: Resolution is not automatizable unless W[P] is tractable. SIAM J. Comput. **38**(4), 1347–1363 (2008). Preliminary version in FOCS 2001
3. Atserias, A., Fichte, J.K., Thurley, M.: Clause-learning algorithms with many restarts and bounded-width resolution. J. Artif. Intell. Res. **40**, 353–373 (2011). Preliminary version in SAT 2009
4. Bayardo Jr., R.J., Schrag, R.: Using CSP look-back techniques to solve real-world SAT instances. In: Proceedings of the 14th National Conference on Artificial Intelligence (AAAI 1997), pp. 203–208, July 1997
5. Beame, P., Kautz, H., Sabharwal, A.: Towards understanding and harnessing the potential of clause learning. J. Artif. Intell. Res. **22**, 319–351 (2004). Preliminary version in IJCAI 2003
6. Ben-Sasson, E., Impagliazzo, R., Wigderson, A.: Near optimal separation of tree-like and general resolution. Combinatorica **24**(4), 585–603 (2004)
7. Blake, A.: Canonical Expressions in Boolean Algebra. Ph.D. thesis, University of Chicago (1937)
8. Bonet, M., Pitassi, T., Raz, R.: Lower bounds for cutting planes proofs with small coefficients. J. Symbolic Logic **62**(3), 708–728 (1997). Preliminary version in STOC 1995
9. Chai, D., Kuehlmann, A.: A fast pseudo-Boolean constraint solver. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **24**(3), 305–317 (2005). Preliminary version in DAC 2003
10. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Math. **4**(1), 305–337 (1973)
11. Chvátal, V., Szemerédi, E.: Many hard examples for resolution. J. ACM **35**(4), 759–768 (1988)
12. CNFgen: Combinatorial benchmarks for SAT solvers. https://github.com/MassimoLauria/cnfgen
13. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC 1971), pp. 151–158 (1971)

14. Cook, S.A., Reckhow, R.: The relative efficiency of propositional proof systems. J. Symbolic Log. **44**(1), 36–50 (1979)
15. Cook, W., Coullard, C.R., Turán, G.: On the complexity of cutting-plane proofs. Discrete Appl. Math. **18**(1), 25–38 (1987)
16. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem proving. Commun. ACM **5**(7), 394–397 (1962)
17. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**(3), 201–215 (1960)
18. Dixon, H.E., Ginsberg, M.L., Hofer, D.K., Luks, E.M., Parkes, A.J.: Generalizing Boolean satisfiability III: implementation. J. Artif. Intell. Res. **23**, 441–531 (2005)
19. Dixon, H.E., Ginsberg, M.L., Luks, E.M., Parkes, A.J.: Generalizing Boolean satisfiability II: theory. J. Artif. Intell. Res. **22**, 481–534 (2004)
20. Dixon, H.E., Ginsberg, M.L., Parkes, A.J.: Generalizing Boolean satisfiability I: Background and survey of existing work. J. Artif. Intell. Res. **21**, 193–243 (2004)
21. Elffers, J.: CDCL-cuttingplanes: A conflict-driven pseudo-Boolean solver (2016). Submitted to the Pseudo-Boolean Competition 2016
22. Elffers, J., Giráldez-Cru, J., Nordström, J., Vinyals, M.: Using combinatorial benchmarks to probe the reasoning power of pseudo-Boolean solvers. In: Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT 2018), July 2018. To appear
23. Elffers, J., Johannsen, J., Lauria, M., Magnard, T., Nordström, J., Vinyals, M.: Trade-offs between time and memory in a tighter model of CDCL SAT solvers. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 160–176. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_11
24. Elffers, J., Nordström, J.: Divide and conquer: towards faster pseudo-Boolean solving. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-ECAI 2018), July 2018. To appear
25. Filmus, Y., Hrubeš, P., Lauria, M.: Semantic versus syntactic cutting planes. In: Proceedings of the 33rd International Symposium on Theoretical Aspects of Computer Science (STACS 2016). Leibniz International Proceedings in Informatics (LIPIcs), vol. 47, pp. 35:1–35:13, February 2016
26. Fleming, N., Pankratov, D., Pitassi, T., Robere, R.: Random $\theta(\log n)$-CNFs are hard for cutting planes. In: Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017), pp. 109–120, October 2017
27. Goerdt, A.: The cutting plane proof system with bounded degree of falsity. In: Proceedings of the 5th International Workshop on Computer Science Logic (CSL 1991), pp. 119–133, October 1991
28. Gomory, R.E.: An algorithm for integer solutions of linear programs. In: Graves, R., Wolfe, P. (eds.) Recent Advances in Mathematical Programming, pp. 269–302. McGraw-Hill, New York (1963)
29. Haken, A.: The intractability of resolution. Theoret. Comput. Sci. **39**(2–3), 297–308 (1985)
30. Hirsch, E.A., Kojevnikov, A., Kulikov, A.S., Nikolenko, S.I.: Complexity of semialgebraic proofs with restricted degree of falsity. J. Satisfiability Boolean Model. Comput. **6**, 53–69 (2008). Preliminary version in SAT 2005 and SAT 2006
31. Hooker, J.N.: Generalized resolution and cutting planes. Ann. Oper. Res. **12**(1), 217–239 (1988)
32. Hooker, J.N.: Generalized resolution for 0-1 linear inequalities. Ann. Math. Artif. Intell. **6**(1), 271–286 (1992)

33. Hrubeš, P., Pudlák, P.: Random formulas, monotone circuits, and interpolation. In: Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017), pp. 121–131, October 2017
34. Impagliazzo, R., Pitassi, T., Urquhart, A.: Upper and lower bounds for tree-like cutting planes proofs. In: Proceedings of the 9th Annual IEEE Symposium on Logic in Computer Science (LICS 1994). pp. 220–228, July 1994
35. Lauria, M., Elffers, J., Nordström, J., Vinyals, M.: CNFgen: a generator of crafted benchmarks. In: Gaspers, S., Walsh, T. (eds.) SAT 2017. LNCS, vol. 10491, pp. 464–473. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66263-3_30
36. Le Berre, D., Parrain, A.: The SAT4J library, release 2.2. J. Satisfiability Boolean Model. Comput. **7**, 59–64 (2010)
37. Markström, K.: Locality and hard SAT-instances. J. Satisfiability Boolean Model. Comput. **2**(1–4), 221–227 (2006)
38. Marques-Silva, J.P., Sakallah, K.A.: GRASP: A search algorithm for propositional satisfiability. IEEE Trans. Comput. **48**(5), 506–521 (1999). Preliminary version in ICCAD 1996
39. Mikša, M., Nordström, J.: Long proofs of (seemingly) simple formulas. In: Sinz, C., Egly, U. (eds.) SAT 2014. LNCS, vol. 8561, pp. 121–137. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09284-3_10
40. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient SAT solver. In: Proceedings of the 38th Design Automation Conference (DAC 2001), pp. 530–535, June 2001
41. Nordström, J.: Pebble games, proof complexity and time-space trade-offs. Log. Methods Comput. Sci. **9**(3), 15:1–15:63 (2013)
42. Nordström, J.: On the interplay between proof complexity and SAT solving. ACM SIGLOG News **2**(3), 19–44 (2015)
43. Pipatsrisawat, K., Darwiche, A.: On the power of clause-learning SAT solvers as resolution engines. Artif. Intell. **175**(2), 512–525, February 2011. Preliminary version in CP 2009
44. Pseudo-Boolean competition 2016. http://www.cril.univ-artois.fr/PB16/, July 2016
45. Pudlák, P.: Lower bounds for resolution and cutting plane proofs and monotone computations. J. Symbol. Log. **62**(3), 981–998 (1997)
46. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965)
47. Roussel, O., Manquinho, V.M.: Pseudo-Boolean and cardinality constraints. In: Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, chap. 22, pp. 695–733. IOS Press, February 2009
48. SAT4J: The Boolean satisfaction and optimization library in Java. http://www.sat4j.org/
49. Sheini, H.M., Sakallah, K.A.: Pueblo: A hybrid pseudo-Boolean SAT solver. J. Satisfiability Boolean Model. Comput. **2**(1–4), 165–189, March 2006. Preliminary version in DATE 2005
50. Spence, I.: sgen1: A generator of small but difficult satisfiability benchmarks. J. Exp. Algorithmics **15**, 1.2:1–1.2:15, March 2010
51. Urquhart, A.: Hard examples for resolution. J. ACM **34**(1), 209–219 (1987)
52. Urquhart, A.: A near-optimal separation of regular and general resolution. SIAM J. Comput. **40**(1), 107–121 (2011). Preliminary version in SAT 2008
53. Van Gelder, A., Spence, I.: Zero-one designs produce small hard SAT instances. In: Strichman, O., Szeider, S. (eds.) SAT 2010. LNCS, vol. 6175, pp. 388–397. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14186-7_37