MaxSAT Resolution with Inclusion Redundancy

Ilario Bonacina ✉ 🄳
Universitat Politècnica de Catalunya, Barcelona, Spain

Maria Luisa Bonet ✉ 🄳
Universitat Politècnica de Catalunya, Barcelona, Spain

Massimo Lauria ✉ 🄳
Sapienza Università di Roma, Italy

──── **Abstract** ────

Popular redundancy rules for SAT are not necessarily sound for MaxSAT. The works of [Bonacina-Bonet-Buss-Lauria'24] and [Ihalainen-Berg-Järvisalo'22] proposed ways to adapt them, but required specific encodings and more sophisticated checks during proof verification. Here, we propose a different way to adapt redundancy rules from SAT to MaxSAT. Our rules do not require specific encodings, their correctness is simpler to check, but they are slightly less expressive. However, the proposed redundancy rules, when added to MaxSAT-Resolution, are already strong enough to capture Branch-and-bound algorithms, enable short proofs of the optimal cost of notable principles (e.g., the Pigeonhole Principle and the Parity Principle), and allow to break simple symmetries (e.g., XOR-ification does not make formulas harder).

1 Introduction

MaxSAT is the problem of finding an assignment that minimizes the number of falsified clauses in a given CNF formula. Several variants of MaxSAT exist that, for example, allow to give different weights to clauses, or enforce some clauses to be hard requirements for the solution. While all state-of-the-art SAT-solvers are more or less based on the same theoretical approach, there is more variety among state-of-the-art MaxSAT solvers, e.g., core-guided, minimum-hitting-set, branch-and-bound, and MaxSAT Resolution [27, 4]. Here we focus mostly on MaxSAT Resolution and we make some observations about branch-and-bound (in Section 5). MaxSAT Resolution was first defined in [15] and proved complete for MaxSAT in [12]. Although MaxSAT is a much harder problem than SAT, in some cases MaxSAT solvers can be adapted to be more efficient than CDCL SAT-solvers on hard problems, for instance dual-rail MaxSAT Resolution [10] has short proofs of the Pigeonhole Principle.

We propose new proof systems for MaxSAT by incorporating *redundancy rules* into MaxSAT resolution. Redundancy rules were introduced in SAT solving to allow the introduction of clauses that preserve satisfiability even though they are not logical consequences.

In other words, redundant clauses formalize the notion of reasoning "*without loss of generality*" [29] by restricting the space of solutions without killing it entirely. The first type of redundancy rules considered were Blocked Clauses (BC) [26] and Resolution Asymmetric Tautologies (RAT) [23, 18]. BC, RAT and stronger redundancy rules are extensively used in proof logging of pre-processing and in-processing of state-of-the-art SAT-solvers, and hence are extensively studied in the literature, for instance in [16, 17, 19, 21, 24, 30]. Particularly relevant to this article is the work of Buss and Thapen [13]. Redundancy rules strengthen significantly the Resolution proof system, for instance allowing to prove efficiently the Pigeonhole Principle [21].

Redundancy rules for MaxSAT can only add clauses that do not increase the minimum number of falsified clauses, and the usual redundancy checks used in SAT do not provide any guarantees on that. Recently, some papers have proposed ways to integrate and study redundancy rules in MaxSAT. The work of Ihalainen, Berg and Järvisalo [22], building on [6], studies the advantage of redundancy to preprocess MaxSAT instances. Proof system veriPB [7] includes redundancy rules, among many others. Being rooted in cutting planes, veriPB is particularly apt at certifying optimality, and it can log the reasoning of MaxSAT solver strategies that are way out of reach of MaxSAT resolution [5]. In contrast to these works, the explicit goal of the of Bonacina, Buss, Bonet, Lauria [8] is to study the proof complexity of redundancy rules for MaxSAT in a similar vein of what [13] does for SAT, something that is beyond the scope of those other works, more focused on proof logging actual solvers. To witness that a clause is redundant for SAT it is sufficient to provide an appropriate variable substitution, and even that is not necessary for the weakest forms of redundancy like BC. For MaxSAT, a proof must witness that the redundant clause preserves optimality. In [22] the authors do not provide a polynomially checkable methods for their most powerful rule, but they do for the simplest ones. In [7] they leverage the underlying language cutting planes, usually strong enough to prove the redundancy of a clause explicitly. The work in [8] gives a simple condition that is easy to check and allow to reuse all redundancy rule from [13] in the MaxSAT setting.

The scope of this paper is to understand the power of redundancy rules in the context of MaxSAT and MaxSAT resolution. We analyze forms of redundancy that are simple, and yet add non-trivial power to MaxSAT resolution. In particular, we adapt the rules from [13], namely Literal Propagation Redundancy (LPR), Subset Propagation Redundancy (SPR), Propagation Redundancy (PR), or Substitution Redundancy (SR), to the context of MaxSAT. For simplicity of presentation, we develop our redundancy rules in a setting where all clauses are soft. Everything generalizes easily to a setting with soft/hard clauses. There, the hard clauses would be manipulated by standard resolution, while the soft clauses would be subjected to MaxSAT resolution and our new redundancy rules. That setting would make the arguments more cumbersome, without providing any further technical insight.

We stress again that this paper does not provide strong proof systems for MaxSAT. Even MaxSAT-Resolution +iSR, which the stronger proof system presented here, is easily simulated by veriPB, a very strong proof system that captures most proof techniques in SAT/CP solving and optimization. Our goal is quite the opposite: we add redundancy in the least intrusive way possible to a simple proof system for MaxSAT, and we investigate its strength. In this setting lower bounds could be reachable, at least for weak redundancy rules as iSPR. On the contrary lower bounds for veriPB are way beyond the reach of current techniques.

Moreover, regarding proof logging applications, the framework we propose captures simple branch-and-bound algorithms, it is plausible it could capture *some* branch-and-bound usage from MaxCDCL [28], and can also witness some preprocessing techniques. On the other hand,

it is unlikely that MaxSAT-Resolution +iSR could capture core-guided MaxSAT. Nevertheless, a small advantage over veriPB is that DRAT/PR verifiers can be adapted to work with our framework with minimal effort.

The redundancy rules in [22, 8, 7] are all variants of rules LPR, SPR, PR, SR as given in [13]. All these variants rely on a specific, although common, *blocking variables* encoding of MaxSAT instances. Furthermore, every time a redundant clause is added to the proof, the verifier must check that cost is preserved.

In this work we follow a different approach. Essentially, we show that BC and suitable variants of LPR, SPR, PR and SR rules are already cost-preserving, thus no additional check is necessary. Moreover, our approach works even without the blocking variable encoding. To give a concrete idea, when studying satisfiability we say that a clause C is redundant w.r.t. a set of clauses Γ if there exists a partial assignment α that satisfies C and such that $\Gamma|_{\overline{C}} \models \Gamma|_\alpha$; that is, the set of clauses $\Gamma|_{\overline{C}}$, i.e., the clauses of Γ restricted by the assignment which is the negation of C, logically imply all the clauses in $\Gamma|_\alpha$. In other words, if we can find a solution of Γ assuming C is false, there is also a solution of Γ assuming the partial assignment α. Unfortunately the relation $\models$ is not polynomially checkable therefore the relation is substituted with unit-propagation $\vdash_1$, a simpler form of logical implication which is efficiently checkable. This notion is fine for satisfiability, but unfortunately the $\vdash_1$ relation is not cost-preserving: the central idea in this paper is to consider $\subseteq$ instead of $\vdash_1$ in the redundancy condition.

In the context of MaxSAT, we say that a clause C is redundant w.r.t. a *multiset* of clauses Γ, when $\Gamma|_{\overline{C}} \supseteq \Gamma|_\alpha$ for some α that satisfies C. Recall that inclusion between multiset takes multiple occurrences in account. We define rules iLPR, iSPR, iPR and iSR that are the "inclusion" versions of redundancy rules LPR, SPR, PR, and SR (Definition 3.2). There is no inclusion version of BC since it is already defined using inclusion. This change makes the rules immediately cost-preserving, thus avoids the extra check on the cost and the blocking variables encoding, needed in [22, 8, 7]. The relation $\supseteq$ is weaker than $\vdash_1$, therefore, in principle, our rules are formally less expressive. This does not seem to be a limitation: in the context of SAT, all the upper bounds for hard tautologies showed in [13], while described using the SPR rule, do not use unit-propagation but only inclusion, even if the SPR rule would have allowed it.

Adding redundancy rules to the resolution proof system makes it stronger. We add our new rules to the MaxSAT-Resolution proof system (Definition 3.4) and we see that indeed it becomes more powerful: we see that some hard contradictions for MaxSAT-Resolution become easy (the Pigeonhole Principle and the Parity Principle, Theorem 4.3 and Theorem 4.4 resp.), and we show that we can "undo" the effect of xorifications (Theorem 4.5). Both the Pigeonhole Principle and the Parity Principle are exponentially hard for MaxSAT-Resolution, since a MaxSAT-Resolution proof of cost at least 1 is also, syntactically, a Resolution refutation, and the principles above are exponentially hard for Resolution [14].

One goal of proof systems for MaxSAT is to capture the reasoning of MaxSAT solvers and it turns out that the iPR rule is quite apt at logging *Branch-and-Bound* (BnB) approaches to optimization. In its simplest form, the BnB approach explores the possible assignments for a CNF formula in a tree-like fashion, in order to find one that satisfies the largest number of clauses, cutting the branches that are bound to give solutions worse than the best one discovered so far. This works well in many scenarios, but historically has not performed well on MaxSAT industrial instances. Recently, though, a better integration with CDCL has made BnB competitive again [28]. As a proof of concept we show how the basic BnB approach can be simulated in MaxSAT-Resolution+iPR.

Structure of the Paper

Section 2 contains notation and preliminaries. In Section 3 we introduce the redundancy rules for MaxSAT and proof systems for MaxSAT based on them. Section 4 showcases the strength of the system. We give short refutation of the Pigeonhole Principle, the Parity Principle, and show how to undo xor-ifications. In Section 5 we show how to simulate in the system Branch-and-bound algorithms. Finally, Section 6 contains some concluding remarks and open problems.

2 Preliminaries

Basic Notation. For $n \in \mathbb{N}$, let $[n] = \{1, \dots, n\}$. We denote with capital Roman or Greek letters sets and multisets. The *size* of a multiset S is $|S|$, the number of elements in S (counted with multiplicity). Given two multisets S, T, S is *included* in T ($S \subseteq T$) if each element of S appears in T with multiplicity at least the multiplicity it had in S.

A *Boolean variable* x is a variable that takes values $\top$ (true) or $\bot$ (false). A *literal* is a Boolean variable x or its negation $\overline{x}$. We consider fixed a finite set X of Boolean variables and let $\overline{X} = \{\overline{x} \mid x \in X\}$. A *clause* is a finite disjunction of literals. A clause D is a *weakening* of a clause C if $D = C \vee C'$ for some clause C'. The empty disjunction is $\bot$.

A map $\sigma\colon X \cup \overline{X} \cup \{\top, \bot\} \to X \cup \overline{X} \cup \{\top, \bot\}$ is a *substitution* if σ is the identity on $\{\top, \bot\}$, and for each $\overline{x} \in \overline{X}$, $\sigma(\overline{x}) = \overline{\sigma(x)}$, where $\overline{\top} = \bot$, $\overline{\bot} = \top$, and for each Boolean variable x, $\overline{\overline{x}} = x$. The composition $\sigma \circ \tau$ of two substitutions σ, τ is also a substitution, where $\sigma \circ \tau(v) = \sigma(\tau(v))$ for every $v \in X \cup \overline{X} \cup \{\top, \bot\}$.

A substitution σ is an *assignment* if for every $\ell \in X \cup \overline{X}$, either $\sigma(\ell) = \ell$ or $\sigma(\ell) \in \{\top, \bot\}$. An assignment τ *extends* an assignment σ if $\sigma^{-1}(\{\top, \bot\}) \subseteq \tau^{-1}(\{\top, \bot\})$. An assignment σ is *total* if for every $\ell \in X \cup \overline{X}$, $\sigma(\ell) \in \{\top, \bot\}$. The *domain* of an assignment σ is $\mathrm{dom}(\sigma) = \sigma^{-1}(\{\top, \bot\})$.

Given a clause $\bigvee_{x \in L} \ell$ and a substitution σ, the *restriction* of C under σ is $C|_\sigma = \bigvee_{x \in L} \sigma(\ell)$, simplified using the usual logic rules ($\top \vee \ell = \top$, $\ell \vee \ell = \ell$, etc.). A substitution σ *satisfies* a clause C if $C|_\sigma$ is a *tautology*, i.e., $C|_\sigma$ is a weakening either of $\top$ or $x \vee \overline{x}$ for some variable x. Given a clause $C = \bigvee_{\ell \in L} \ell$, we denote with $\overline{C}$ the assignment given by $\sigma(\ell) = \bot$ and $\sigma(\overline{\ell}) = \top$ if $\ell \in L$ and the identity otherwise. In particular, $C|_{\overline{C}} = \bot$.

Given a set of clauses Γ and a substitution σ, the *restriction* of Γ under σ is the multiset $\Gamma|_\sigma = \{C|_\sigma : C \in \Gamma \text{ and } \sigma \not\models C\}$. A substitution σ *satisfies* a multiset of clauses Γ ($\sigma \models \Gamma$) if σ satisfies all the clauses in Γ, i.e., $\Gamma|_\sigma = \emptyset$. A multiset of clauses Γ *entails* a multiset of clauses Δ if for every substitution σ s.t. $\sigma \models \Gamma$ it holds $\sigma \models \Delta$.

▶ **Fact 1.** If S, T are multisets of clauses with $S \subseteq T$ and σ is a substitution then $S|_\sigma \subseteq T|_\sigma$.

▶ **Fact 2** ([13, Lemma 1.1]). If S is a multiset of clauses and σ, τ are substitutions, then $(S|_\sigma)|_\tau = S|_{\tau \circ \sigma}$.

MaxSAT and MaxSAT-Resolution. Given a multiset of clauses S, MaxSAT asks to find the maximum number of clauses in S that can be simultaneously satisfied. Equivalently, find a total assignment mapping to $\bot$ the smallest number of clauses in S possible. The *cost* of a multiset of clauses S is $\mathrm{cost}(S)$, the minimum of size of $S|_\alpha$ over all possible total assignments α. Notice that for a total assignment α, $S|_\alpha$ is a multiset of the form $\{\bot, \dots, \bot\}$. The goal of proof systems for MaxSAT is to show lower bounds on the cost of MaxSAT instances. One of such systems is MaxSAT-Resolution which was introduced in [15] and proved complete in [12]. We present the system using the rules of [2], first used in the context of MaxSAT in [11].

▶ **Definition 2.1** (MaxSAT-Resolution). *A sequence of multisets* $(S_i)_{i \in [m]}$ *is a* derivation *of* S_m *from* S_1 *in* MaxSAT-Resolution *if for each* $i \in [m]$ *either*

(i) $S_{i+1} = (S_i \setminus \{C \vee x, \ C \vee \overline{x}\}) \cup \{C\}$ *where* $\{C \vee x, \ C \vee \overline{x}\} \subseteq S_i$ (SYMM. CUT)*; or*

(ii) $S_{i+1} = (S_i \setminus \{C\}) \cup \{C \vee x, \ C \vee \overline{x}\}$ *where* $C \in S_i$ (SPLIT)*.*

The size *of the derivation is* $\sum_{i \in [m]} |S_i|$.

It is known that MaxSAT-Resolution is *sound* and *complete* for MaxSAT:

Soundness Whenever $(S_i)_{i \in [m]}$ is a MaxSAT-Resolution derivation, $\text{cost}(S_1) = \text{cost}(S_m)$. Therefore, if $\bot$ appears in S_m with multiplicity k, then trivially $\text{cost}(S_m) \geq k$ and we say that the MaxSAT-Resolution derivation *certifies* $\text{cost}(S_1) \geq k$, since $\text{cost}(S_1) = \text{cost}(S_m)$.

Completeness Whenever $\text{cost}(S_1) = k$ there is a MaxSAT-Resolution derivation $(S_i)_{i \in [m]}$ where $\bot$ occurs k times in S_m, and all the remaining clauses in S_m are satisfiable [12, Theorem 10].

3 Cost-preserving rules

In the context of SAT, a clause C is *redundant* w.r.t. a set of clauses F if adding C to F does not affect its satisfiability or unsatisfiability, i.e., $F \cup \{C\}$ is satisfiable if and only if F is satisfiable [26]. In particular if $F \models C$, then C is redundant.

In the context of MaxSAT the intuition is similar, a clause C is *redundant* w.r.t. a multiset of clauses S if adding C to S does not affect the cost. In this context, even when $F \models C$, it is not clear that C is redundant w.r.t. S. The notion of redundancy for MaxSAT was introduced in [22] in a slightly different setting than ours.

▶ **Definition 3.1** (redundant clause). *A clause C is* redundant *w.r.t. a multiset of clauses S if* $\text{cost}(S) = \text{cost}(S \cup \{C\})$.

Notice that C is redundant w.r.t. S if and only if there exists a total assignment β with $|S|_\beta = \text{cost}(S)$ and $\beta \models C$. In particular, if C is redundant w.r.t. S, then the clause C could be added to S with arbitrary multiplicity without changing the cost, i.e., $\text{cost}(S) = \text{cost}(S \cup \{C\}) = \text{cost}(S \cup \{C, \dots, C\})$.

Unless $\mathsf{P} = \mathsf{NP}$, it is not polynomially checkable whether a clause C is redundant w.r.t. a multiset of clauses S, therefore, as in the context of SAT, we consider polynomially verifiable notions of redundancy, i.e., ways of adding redundant clauses (as per Definition 3.1) while certifying efficiently their redundancy. In [8] the authors described a systematic way of porting to MaxSAT the notions of efficiently certifiable redundancy already studied in the literature of SAT [20], for example the systems SR/PR/SPR/LPR/BC . This relied on (1) a particular form of the MaxSAT instance considered, and (2) an additional condition to enforce the correctness for MaxSAT.

By considering limited versions of the rules SR/PR/SPR/LPR/BC, we show a conceptually simpler way of adding redundancy rules to MaxSAT. This approach is alternative to the one in [8, 22].

▶ **Definition 3.2** (Inclusion Substitution Redundant, iSR). *A clause C is* Inclusion Substitution Redundant (iSR) *w.r.t. a multiset of clauses S if exists a substitution σ s.t.*

$$(S \cup \{C\})|_\sigma \subseteq S|_{\overline{C}} \ .$$

If the substitution σ has some additional structure, we also have the following redundancy rules, listed in decreasing order of generality:

Inclusion Propagation Redundant (iPR) if σ is an assignment.

Inclusion Subset Propagation Redundant (iSPR) if σ is an assignment with the same domain of $\overline{C}$. Hence σ differs from $\overline{C}$ in the value given to some variables.

Inclusion Literal Propagation Redundant (iLPR) if σ is an assignment with the same domain of $\overline{C}$ and differs from $\overline{C}$ in the value given to exactly one variable.

Blocked Clause (BC) if σ is an assignment with the same domain of $\overline{C}$ and differs from $\overline{C}$ in the value given to exactly one variable x, and moreover for every clause in $D \in S$ containing the variable x, $\sigma \models D$.[1]

Notice that while BC and iLPR might look very similar, they are distinct concepts. For instance, the clause $C = x$ is iLPR w.r.t. $S = \{y, y \vee x, y \vee \overline{x}\}$ but it is not BC w.r.t. the same set. Another redundancy rule is SBC, a generalization of BC defined in [25] which we do not address explicity. As it happens for BC and iLPR, SBC is a valid redundancy rule for MaxSAT and a proper special case of iSPR. For instance, the clause $C = \{x \vee z\}$ is iSPR w.r.t. $F = \{y, y \vee x, y \vee \overline{x}, y \vee x \vee \overline{z}, y \vee \overline{x} \vee z\}$, but it is not SBC (nor iLPR).

▶ **Lemma 3.3.** *If a clause C is* iSR *w.r.t a multiset of clauses S, then C is redundant w.r.t. S, i.e.,* $\text{cost}(S) = \text{cost}(S \cup \{C\})$.

Proof. Clearly $\text{cost}(S) \leq \text{cost}(S \cup \{C\})$. To prove the other inequality, let $k = \text{cost}(S)$ and let β be a total assignment such that $|S|_\beta| = k$. If $\beta \models C$, then we are done. Suppose then $C|_\beta = \bot$. That is β extends $\overline{C}$. By assumption, there is a substitution σ such that $(S \cup \{C\})|_\sigma \subseteq S|_{\overline{C}}$. Therefore $(S \cup \{C\})|_{\beta \circ \sigma} \subseteq S|_{\beta \circ \overline{C}} = S|_\beta$. Hence $|(S \cup \{C\})|_{\beta \circ \sigma}| \leq |S_\beta| = k$, and $\text{cost}(S \cup \{C\}) \leq k$.   ◀

Checking whether a clause is iSR w.r.t. S, given the substitution σ, is doable in polynomial time. Therefore we can extend any proof system for MaxSAT with a rule that introduces iSR clauses. We now consider such extension for MaxSAT-Resolution.

▶ **Definition 3.4** (MaxSAT-Resolution + iSR)**.** *A sequence of multisets $(S_i)_{i \in [m]}$ is a* derivation *of S_m from S_1 in* MaxSAT-Resolution + iSR *if for each $i \in [m]$ either one of the cases* (i), (ii) *of the definition of* MaxSAT-Resolution *occur, or*
(iii) $S_{i+1} = S_i \cup \{C\}$ *where C is* iSR *w.r.t. S_i;*
(iv) $S_{i+1} = S_i \setminus \{C\}$ *where C is* iSR *w.r.t. $S_i \setminus \{C\}$.*
Each occurrence of the rules (iii) and (iv) is accompanied by the corresponding substitution σ witnessing the validity of the rule. The size *of the derivation is $\sum_{i \in [m]} |S_i|$. The definition of* MaxSAT-Resolution+R *for any $R \in \{\text{iPR}, \text{iSPR}, \text{iLPR}, \text{BC}\}$ is analogous.*

We only consider the case where MaxSAT-Resolution + iSR derivation are not allowed to introduce new variables, since introducing new variables makes the systems as strong as Extended Resolution [26]. To be consistent with [13], the rules/systems should be called MaxSAT-Resolution+ iSR⁻, where the "−" is used to indicate that the systems are not allowed to introduce new variables. We ignore that convention to ease notation. The system MaxSAT-Resolution + iSR is *sound* and *complete*.

Soundness Lemma 3.3 and the soundness of MaxSAT-Resolution immediately imply that MaxSAT-Resolution + iSR is also sound, i.e., whenever $(S_i)_{i \in [m]}$ is a MaxSAT-Resolution + iSR derivation, $\text{cost}(S_1) = \text{cost}(S_m)$. Therefore, as in the case of MaxSAT-Resolution, we say that the MaxSAT-Resolution + iSR derivation *certifies* $\text{cost}(S_1) \geq k$ if S_m contains $\bot$ with multiplicity at least k.

Completeness The completeness of MaxSAT-Resolution + iSR is immediate from the completeness of the system MaxSAT-Resolution.

---

[1] This is not the usual definition of BC but it is equivalent, as shown in [8].

▶ **Remark 3.5.** The choice of MaxSAT-Resolution in Definition 3.4 is in some sense arbitrary: the rule iSR (i.e., items (iii) and (iv) in Definition 3.4) could be added or easily adapted to any sound proof system for MaxSAT with substitution rules, for instance, the weighted resolution proof system from [11]. In particular, since weighted resolution is equivalent to Sherali-Adams, and restricted weighted resolution is equivalent to Nullstellensatz [11, 9], this means the iSR rule could be also added to those (semi-)algebraic proof systems.

▶ **Remark 3.6.** Our goal is to adapt rules like SR from the SAT framework to MaxSAT. Nevertheless the iSR rule and its restrictions iPR/iSPR/iLPR/BC still make sense for SAT. If we apply them to sets instead of multisets, they immediately become special cases of the original redundancy rules for SAT. For example iSR in this context is a special case of the SR rule that we spell here for convenience. A clause C is *Substitution Redundant* (SR) w.r.t. a set of clauses S if

$$S|_{\overline{C}} \vdash_1 (S \cup \{C\})|_\sigma ,$$

where $\vdash_1$ indicates *unit propagation*, an efficiently checkable form of entailment. To the best of our knowledge, the iSR/iPR/iSPR/iLPR rules are presented here for the first time, both in the context of SAT and MaxSAT. We observe, though, that the upper bounds for pigeonhole principle, bit-pigeonhole principles, clique-coloring, parity, xor-ification, and Tseitin formulas in [13, Section 4] are stated for the rule SPR in the case of SAT, but in fact fulfill the inclusion condition as in iSPR. On the other hand, these results cannot be automatically adapted to MaxSAT-Resolution+iSPR due to the MaxSAT-Resolution rule restrictions.

4 Certifying the cost of some hard tautologies

In this section we exemplify the power of MaxSAT-Resolution + iSR by (1) efficiently certifing the optimum cost of the pigeonhole principle PHP_n^m, (2) efficiently certifying the optimum cost of the parity principle, and (3) reversing the hardness increase due to xor-ification of CNFs. To do so we use few simple, yet useful, lemmas.

▶ **Lemma 4.1.** *Given a clause C and multisets of clauses S and T, if there is a substitution π such that $S|_\pi \subseteq S$ and $C|_\pi \vee C$ is a tautology and for every clause $D \in T$, $D|_\pi \vee C$ is also a tautology, then C is iSR w.r.t. $S \cup T$.*

Proof. Let π such that $S|_\pi \subseteq S$ and $C|_\pi \vee C$ is a tautology. Let $\sigma = \overline{C} \circ \pi$. Since $C|_\pi \vee C$ is a tautology we have $\overline{C} \models C|_\pi$ and $\sigma \models C$. Similarly for every clause $D \in T$, $\sigma \models D$. That is

$$(S \cup T \cup \{C\})|_\sigma = S|_\sigma = S|_{\overline{C} \circ \pi} = (S|_\pi)|_{\overline{C}} \subseteq S|_{\overline{C}} \subseteq (S \cup T)|_{\overline{C}} . \qquad \blacktriangleleft$$

It is well known that pure literals are blocked clauses. For convenience here we state essentially the same fact for iLPR.

▶ **Lemma 4.2.** *Given a multiset of clauses Γ and a literal ℓ such that $\overline{\ell}$ does not occur in Γ. We can derive Γ' from Γ using the iLPR rule where $\Gamma' \subseteq \Gamma$ and Γ' is a multiset not containing any clause with the literal ℓ.*

Proof. It is sufficient to consider the symmetric difference to be a single clause $C \vee \ell$ and show that $C \vee \ell$ is iLPR w.r.t. to Γ. The main claim follows by repeated applications of iLPR rules.

Let $\Gamma = \Gamma_0 \cup \Gamma_\ell$ where Γ_0 are the clauses containing neither ℓ nor $\overline{\ell}$, and Γ_ℓ are the clauses with literal ℓ. We fix $\sigma = \{\overline{C} \wedge \ell = 1\}$, and observe that

$$(\Gamma \cup \{C \vee \ell\})|_\sigma = \Gamma_0|_{\overline{C}} = \Gamma_0|_{\overline{C} \wedge \ell = 0} \subseteq \Gamma|_{\overline{C} \wedge \ell = 0} . \qquad \blacktriangleleft$$

4.1   Pigeonhole principle PHP_n^m

Let $m, n \in \mathbb{N}$ with $m > n$. The propositional encoding of the Pigeonhole Principle PHP_n^m uses Boolean variables $p_{i,j}$ with $i \in [m]$ and $j \in [n]$ with intended meaning that $p_{i,j}$ is true if and only if the pigeon i flies to hole j. For $i < k$ let the *injectivity axiom* $\mathsf{Inj}_{i,k,j}$ be the clause $\overline{p_{i,j}} \vee \overline{p_{k,j}}$, expressing that the two pigeons i, k cannot fly at the same time to hole j; and let the *totality axiom* $\mathsf{Tot}_{i,n}$ be the clause $\bigvee_{j \in [n]} p_{i,j}$, expressing that the pigeon i must fly somewhere among the n holes. The CNF encoding of Pigeonhole Principle is

$$\mathsf{PHP}_n^m = \{\mathsf{Tot}_{i,n} \mid i \in [m]\} \cup \{\mathsf{Inj}_{i,k,j} \mid i, k \in [m], \ j \in [n] \text{ and } i < k\} \ .$$

An assignment that maps the first n pigeons to the n holes, and leaves the other pigeons unassigned, falsifies $m - n$ totality axioms and no injectivity axioms. Hence $\mathrm{cost}(\mathsf{PHP}_n^m) \leq m - n$, and we can prove that this is optimal in $\mathsf{MaxSAT\text{-}Resolution} + \mathsf{iSR}$.

▶ **Theorem 4.3.** *There is a polynomial size derivation in* $\mathsf{MaxSAT\text{-}Resolution} + \mathsf{iSR}$ *showing that* $\mathrm{cost}(\mathsf{PHP}_n^m) \geq m - n$.

Proof. It is enough to show how to derive PHP_{n-1}^{m-1} from PHP_n^m, since repeating this process n times gives PHP_0^{m-n}. This latter formula contains no other clauses than $m - n$ totality axioms $\mathsf{Tot}_{i,0}$, that are indeed copies of the empty clause $\bot$. This would conclude the proof.

To derive PHP_{n-1}^{m-1} from PHP_n^m we use iSR to enforce one by one all the pigeons of index below m not to fly into hole n. Namely for $1 \leq i \leq m$ we have the intermediate sets

$$\Gamma_i = \big(\big(\mathsf{PHP}_n^m \setminus \{\mathsf{Inj}_{\ell,k,n} \ : \ \ell < i \text{ and } k \neq \ell\}\big) \setminus \{\mathsf{Tot}_{\ell,n} \ : \ \ell < i\}\big) \cup \{\mathsf{Tot}_{\ell,n-1} \ : \ \ell < i\} \ ,$$

In particular, $\Gamma_1 = \mathsf{PHP}_n^m$ and $\Gamma_m = \mathsf{PHP}_{n-1}^{m-1} \cup \{\mathsf{Tot}_{m,n}\}$. The variable $x_{m,n}$ appears only in $\mathsf{Tot}_{m,n}$ therefore, by Lemma 4.2, this clause can be removed from Γ_m to get PHP_{n-1}^{m-1}.

Suppose now $1 \leq i < m$, we have the database of clauses is Γ_i and we want to obtain Γ_{i+1}.

Step 1.   The clause $C = p_{m,n} \vee \overline{p_{i,n}}$ is iSR for Γ_i, and we witness that with permutation π that exchanges pigeons i and m. That is, $\pi(p_{m,j}) = p_{i,j}$, $\pi(p_{i,j}) = p_{m,j}$, $\pi(\overline{p_{m,j}}) = \overline{p_{i,j}}$ and $\pi(\overline{p_{i,j}}) = \overline{p_{m,j}}$ for every $j \in [n]$. On the other variables π is the identity. The permutation π maps Γ_i to itself: totality axioms $\mathsf{Tot}_{m,n}$ and $\mathsf{Tot}_{i,n}$ are both in Γ_{i-1} and get swapped; the injectivity axioms $\mathsf{Inj}_{\ell,k,n}$ in Γ_i all have $\ell, k \geq i$, therefore π maps this set of axioms to itself, the set of remaining axioms is also mapped to itself. Applying Lemma 4.1 with $T = \emptyset$ we get that $p_{m,n} \vee \overline{p_{i,n}}$ is iSR wrt Γ_i. Once we add $p_{m,n} \vee \overline{p_{i,n}}$ to Γ_i, we cut it with the injectivity axiom $\mathsf{Inj}_{i,m,n}$ to get $\overline{p_{i,n}}$. Now the database of clauses is $\Gamma_i \setminus \{\mathsf{Inj}_{i,m,n}\} \cup \{\overline{p_{i,n}}\}$.

Step 2.   To cut $\overline{p_{i,n}}$ with $\mathsf{Tot}_{i,n}$, i.e., $\bigvee_{j \in [n]} p_{i,j}$ we need first to split $\overline{p_{i,n}}$ repeatedly getting the database of clauses

$$\big(\Gamma_i \setminus \{\mathsf{Inj}_{i,m,n}\}\big) \cup \{\overline{p_{i,n}} \vee \big(\bigvee_{\ell \in [j-1]} p_{i,\ell}\big) \vee \overline{p_{i,j}} \ : \ j \in [n-1]\} \cup \{\overline{p_{i,n}} \vee p_{i,1} \vee p_{i,2} \vee \cdots \vee p_{i,n-1}\} \ .$$

Now cut $\overline{p}_{i,n} \vee p_{i,1} \vee p_{i,2} \vee \cdots \vee p_{i,n-1}$ with the totality axiom $\mathsf{Tot}_{i,n}$ to obtain $\mathsf{Tot}_{i,n-1}$ and the database of clauses

$$\Delta = \big(\Gamma_i \setminus \{\mathsf{Inj}_{i,m,n}, \mathsf{Tot}_{i,n}\}\big) \cup \{\mathsf{Tot}_{i,n-1}\} \cup \{\overline{p_{i,n}} \vee \big(\bigvee_{\ell \in [j-1]} p_{i,\ell}\big) \vee \overline{p_{i,j}} \ : \ j \in [n-1]\} \ .$$

Step 3. The database Δ only contains variable $p_{i,n}$ with negative polarity, hence, Lemma 4.2 allows to remove all the clauses containing $\overline{p_{i,n}}$, and get the database of clauses

$$\left(\left(\Gamma_i \setminus \{\mathsf{Inj}_{i,k,n} \ : \ i < k\}\right) \setminus \{\mathsf{Tot}_{i,n}\}\right) \cup \{\mathsf{Tot}_{i,n-1}\} = \Gamma_{i+1} \ . \qquad \blacktriangleleft$$

The proof of Theorem 4.3 is a generalization of the argument to prove efficiently the unsatisfiability of PHP_n^{n+1} in [13, Example 1.4], which in turn is based on [20].

4.2 Parity principle

The *Parity Principle* claims that there is a perfect matching between an odd number of elements. The propositional encoding of this principle (Parity_n) is minimally unsatisfiable, and here we show that it has a short proof in $\mathsf{MaxSAT\text{-}Resolution}+\mathsf{iSR}$. This is interesting since the formula is hard for Sherali-Adams and Sum-of-Squares proof systems [3, 1].

The set of clauses Parity_n has Boolean variables $x_{\{i,j\}}$ for $i, j \in [n]$ with $i \neq j$, where $x_{\{i,j\}}$ means that elements i and j are matched together. To ease the notation we use $x_{i,j}$ and $x_{j,i}$ as alternative notations for $x_{\{i,j\}}$. For each $i, n' \in [n]$ we define the set of clauses

$$\mathsf{AtLeast}_i^{n'} = \bigvee_{j \in [n'] \setminus \{i\}} x_{i,j} \qquad \mathsf{AtMost}_i^{n'} = \{\overline{x}_{i,j} \vee \overline{x}_{i,j'} \ : \ j, j' \in [n'], \ i, j, j' \text{ all distinct}\} \ .$$

Their informal meaning is that the element i matches with at least and at most one distinct element in $[n']$ respectively. The set of clauses Parity_n is then

$$\mathsf{Parity}_n = \{\mathsf{AtMost}_i^n, \ \mathsf{AtLeast}_i^n \ : \ i \in [n]\} \ .$$

For n odd, there is an assignment of the variables satisfying all but one clause, that is Parity_n is minimally unsatisfiable.

▶ **Theorem 4.4.** *For odd n, there is a polynomial size derivation in* $\mathsf{MaxSAT\text{-}Resolution}$ $+\mathsf{iSR}$ *showing that* $\mathrm{cost}(\mathsf{Parity}_n) \geq 1$.

Proof. The strategy of the proof is to start with a clauses of Parity_n and deduce from it the clauses of Parity_{n-2}. Since n is odd at some point we get to Parity_1, which contains $\mathsf{AtLeast}_1^1$ which is the empty disjunction, i.e., $\bot$.

To reduce Parity_n to Parity_{n-2} we enforce the elements $n - 1$ and n to match.

Step 1. Derive clauses $C_i = \overline{x_{n,i}} \vee x_{n,(n-1)}$ for every $1 \leq i \leq n - 2$ in this order one by one. To derive the clause C_i we use Lemma 4.1 with $S = \mathsf{Parity}_n$ and $T = \{C_j \ : \ j < i\}$. As witnessing substitution we use π, the variable permutation induced by swapping indices i and $n - 1$. By symmetry $\mathsf{Parity}_n|_\pi = \mathsf{Parity}_n$. Then, observe that the clause $C_i|_\pi$ and all clauses $C_j|_\pi$ for $C_j \in T$ contain the literal $x_{n,i}$, while C_i contains $\overline{x_{n,i}}$. Therefore $C_i|_\pi \vee C_i$ and all $C_j|_\pi \vee C_i$ are tautologies and Lemma 4.1 applies.

Step 2. Derive clauses $D_i = \overline{x_{(n-1),i}} \vee x_{(n-1),n}$ for every $1 \leq i \leq n - 2$ in this order, using almost the same strategy of Step 1. To derive the clause D_i we again use Lemma 4.1 with substitution π induced by the variable permutation induced by swapping indices i and n. Everything works as in the previous paragraph, $S = \mathsf{Parity}_n$ and

$$T = \{D_j \ : \ j < i\} \cup \{C_k \ : \ k \in [n-2]\} \ .$$

Clause $D_i|_\pi$ contains the literal $x_{(n-1),i}$ and the same happens for all $D_j|_\pi$ with $j < i$, and for all $C_k|_\pi$ with $k \in [n-2]$. Hence, as in Step 1, all $D_j|_\pi \vee D_i$ with $j \leq i$ and all $C_k|_\pi \vee D_i$ with $k \in [n-2]$ are tautologies, and Lemma 4.1 applies. The current database of clauses is

$$\Gamma = \mathsf{Parity}_n \cup \{C_i \ : \ i \in [n-2]\} \cup \{D_j \ : \ j \in [n-2]\} \ .$$

Step 3. For all $i \in [n-2]$ we do a symmetric cut between the clause $C_i = \overline{x_{n,i}} \vee x_{n,(n-1)}$, introduced in Step 2, and the Parity_n clause $\overline{x_{n,i}} \vee \overline{x_{n,(n-1)}}$, to obtain the set of unit clauses

$$\{\overline{x_{n,i}} \ : \ i \in [n-2]\} \ ,$$

consuming all the clauses of the form C_i and all the clauses $\overline{x_{n,i}} \vee \overline{x_{n,(n-1)}}$ with $i \in [n-2]$. Similarly, for all $j \in [n-2]$ we do a symmetric cut between the clause $D_j = \overline{x_{(n-1),j}} \vee x_{(n-1),n}$, introduced in Step 1, and the Parity_n clause $\overline{x_{(n-1),j}} \vee \overline{x_{(n-1),n}}$, to obtain the set of unit clauses

$$\{\overline{x_{(n-1),j}} \ : \ j \in [n-2]\} \ ,$$

consuming from Γ all the clauses of the form D_j and all the clauses $\overline{x_{(n-1),j}} \vee \overline{x_{(n-1),n}}$ with $j \in [n-2]$. As a result the current database of clauses is

$$\Gamma' = (\mathsf{Parity}_n \backslash \{\overline{x_{(n-1),i}} \vee \overline{x_{(n-1),n}}, \ \overline{x_{n,i}} \vee \overline{x_{n,(n-1)}} \ : \ i \in [n-2]\}) \cup \{\overline{x_{i,(n-1)}}, \ \overline{x_{i,n}} \ : \ i \in [n-2]\} \ .$$

Step 4. In Γ' literal $\overline{x_{n,n-1}}$ does not occur, so we can use Lemma 4.2 to remove both $\mathsf{AtLeast}_{n-1}^n$ and $\mathsf{AtLeast}_n^n$ from Γ'. The clause database becomes

$$\Gamma'' = \{\mathsf{AtMost}_i^{n-2}, \mathsf{AtLeast}_i^n \ : \ i \in [n-2]\} \cup \{\overline{x_{i,(n-1)}}, \ \overline{x_{i,n}} \ : \ i \in [n-2]\} \ .$$

Step 5. To conclude the derivation of Parity_{n-2} we need to shorten the all the clauses $\mathsf{AtLeast}_i^n$ into $\mathsf{AtLeast}_i^{n-2}$ for each $i \in [n-2]$. We show how to derive $\mathsf{AtLeast}_i^{n-1}$ from unit $\overline{x_{i,n}}$ and $\mathsf{AtLeast}_i^n$. The same procedure then works to get $\mathsf{AtLeast}_i^{n-2}$ from unit $\overline{x_{i,n-1}}$ and $\mathsf{AtLeast}_i^{n-1}$.

First split $\overline{x_{i,n}}$ into $\overline{x_{i,1}} \vee \overline{x_{i,n}}$ and $x_{i,1} \vee \overline{x_{i,n}}$, then the latter clause into $x_{i,1} \vee \overline{x_{i,2}} \vee \overline{x_{i,n}}$ and $x_{i,1} \vee x_{i,2} \vee \overline{x_{i,n}}$, and so on up to get $\bigvee_{j \in [n-1]\backslash\{i\}} x_{i,j} \vee \overline{x_{i,n}}$. We do symmetric cut between this last clause and $\mathsf{AtLeast}_i^n$ to get $\mathsf{AtLeast}_i^{n-1}$. Notice that all intermediate clauses from the splits left in the clause database contain the literal $\overline{x_{i,n}}$.

Repeating this procedure using the unit clause $\overline{x_{i,n-1}}$ and $\mathsf{AtLeast}_i^{n-1}$ gives $\mathsf{AtLeast}_i^{n-2}$ and several intermediate clauses containing literal $\overline{x_{i,n-1}}$.

We do this for every $i \in [n-2]$, so that in the clause database we have Parity_{n-2} plus clauses containing literals of the form either $\overline{x_{i,n-1}}$ or $\overline{x_{i,n}}$. We can remove all such clauses using Lemma 4.2 because the opposites of these literals do not occur in Parity_{n-2}. This concludes the derivation of Parity_{n-2}. ◀

Therefore, a consequence of Theorem 4.4 is that neither Sherali-Adams nor Sum-of-Squares as proof systems for MaxSAT can simulate MaxSAT-Resolution $+$ iSR.

4.3 XOR-ification

We show that MaxSAT-Resolution+iSPR can "undo" the effect of common techniques used to make hard instances of propositional tautologies. For concreteness we do it for xor-ifications. This is analogous to the case of SAT, where the SPR rule can be used to "undo" the effects of xor-ifications [13, Section 4.6].

Given a multiset of clauses F, the mth *xor-ification* of a variable x is the set of clauses where the variable x is substituted by the XOR of m new variables $x_1 \oplus \cdots \oplus x_m$ and the resulting formula is expanded again as a CNF formula. The mth xor-ification of F (denoted $F[\oplus^m]$) is the procedure above applied to all the variables of F. Notice that $\mathrm{cost}(F) = \mathrm{cost}(F[\oplus^m])$.

▶ **Theorem 4.5.** *Let F be a multiset of clauses with a* MaxSAT-Resolution *derivation showing that* $\mathrm{cost}(F) \geq k$ *in size* s, *then there is a* MaxSAT-Resolution+iSPR *derivation showing that* $\mathrm{cost}(F[\oplus^m]) \geq k$ *of size polynomial in* s *and the number of clauses of* $F[\oplus^m]$.

Proof. The idea is to remove all the symmetries among the xor-ified variables at the beginning one by one, and then do the MaxSAT-Resolution derivation. We show how to "undo" the xor-ification of a variable x xor-ified into $x_1 \oplus x_2 \oplus \cdots \oplus x_m$. Without loss of generality we assume m to be even, therefore for each clause in F of the form $C \vee x$, $F[\oplus^m]$ contains the clause $C \vee x_1 \vee x_2 \vee \cdots \vee x_m$; similarly, for each clause in F of the form $C \vee \overline{x}$, $F[\oplus^m]$ contains the clause $C \vee \overline{x_1} \vee x_2 \vee \cdots \vee x_m$. Let ℓ be the number of occurrences of the variable x in F.

Step 1. Let $\Gamma_i = F[\oplus^m] \cup \{\underbrace{\overline{x_j} \ldots \overline{x_j}}_{\ell \text{ copies}} : 2 \leq j \leq i\}$, so that $\Gamma_1 = F[\oplus^m]$. For $i \geq 2$, we see how to derive Γ_i from Γ_{i-1}.

The clauses $x_1 \vee \overline{x_i}$ is iSPR w.r.t. Γ_{i-1} and $\overline{x_1} \vee \overline{x_i}$ is iSPR w.r.t. $\Gamma_{i-1} \cup \{x_1 \vee \overline{x_i}\}$. For the first application of the iSPR rule we set $\sigma = \{x_1 := \top, x_i := \bot\}$. To check that it is a valid application notice that for any clause $C \vee (\overline{x_1} \vee x_i \vee \ldots)$ that gets restricted but not satisfied in $\Gamma_{i-1}|_\sigma$, there is another clause $C \vee (x_1 \vee \overline{x_i} \vee \ldots)$ that gets restricted in the same way in $\Gamma_{i-1}|_{\{x_1 := \bot, x_i := \top\}}$. For the second application $\sigma = \{x_1 := \bot, x_i := \bot\}$ and we apply a similar reasoning. In both cases we add $x_1 \vee \overline{x_i}$ and $\overline{x_1} \vee \overline{x_i}$ to Γ_{i-1} with multiplicity ℓ. Afterward, by symmetric cut we obtain ℓ copies of $\overline{x_i}$, and this gives us Γ_i. We keep going until we get to Γ_m.

Step 2. Now in the clause database we have ℓ copies each of the sequence of unit clauses $\overline{x_2}, \ldots, \overline{x_m}$. A positive occurrence of original variable x in a clause $C \vee x \in F$ induces a clause $C \vee x_1 \vee x_2 \vee \cdots \vee x_m \in F[\oplus^m]$. To resolve that with $\overline{x_2}, \ldots, \overline{x_m}$ we first apply splits to the units, and eventually we can apply a series of symmetric cuts and obtain $C \vee x_1$. In the same way, for $C \vee \overline{x} \in F$ we work on the corresponding clause $C \vee \overline{x_1} \vee x_2 \vee \cdots \vee x_m$ to get clause $C \vee \overline{x_1}$. In the end the clause database contains a copy of F up to variable renaming. ◀

5 Simulating branch-and-bound with MaxSAT redundancy rules

Given a set of clauses F, the basic *branch-and-bound* (BnB) procedure explores the space of all possible assignments for F in a depth-first way. At every node, the BnB procedure has computed an upper bound UB and a lower bound LB: the UB is the cost of the best solution found so far, while the LB is the number of falsified clauses in the current branch. At the beginning of the procedure UB is the number of clauses in F and the LB is 0. At each node the procedure compares LB and UB at that node: if $LB \geq UB$ the algorithm prunes the branch, i.e., it does not continue to explore the subtree and backtracks to a previous node, since we are exploring an assignment that we already discovered it is not optimal. If $LB < UB$ the algorithm instantiates one more variable and continues the exploration. The solution is the value of UB after exploring the whole search tree.

Here we simulate the basic BnB approach via MaxSAT-Resolution+ iPR. Let S be a multiset of clauses, let T be the BnB decision tree for S and let t be the number of leaves of T. We identify the leaves of T with the partial assignments $\beta_1, \beta_2, \ldots, \beta_t$ that label the branches of the tree, enumerated according to the visit order. Each leaf has an associated cost k_i, which is the LB at node i, the number of clauses of S falsified by β_i. Each leaf β_i is of one of two types:

- *Pruning:* when $k_i \geq k_j$ for some $1 \leq j < i$; k_j corresponds to the UB;
- *Improvement:* when $k_i < k_j$ for all $1 \leq j < i$ and each clause in S is either satisfied or falsified by β_i, because β_i is a leaf. In this case k_i will be the new UB.

▶ **Theorem 5.1.** *Consider a BnB procedure for MaxSAT on clauses S and let T be the BnB decision tree associated showing* $\text{cost}(S) = k$. *Then there is a* MaxSAT-Resolution+ iPR *proof of length* $O(k \cdot |T|)$ *that* $\text{cost}(S) \geq k$.

Proof. Let t be the number of leaves of T, and $m = |S|$. *First phase:* the proof simulates the BnB by considering the leaves $\beta_1, \ldots, \beta_t$ one by one. The derivation maintains a multiset of clauses that forbid all leaves seen so far, except for the leaf with the current best value. If the next leaf is pruned, then the proof adds a clause to forbid it. If the next leaf is an improvement, then the proof forbids the leaf corresponding to the previous best value. We will use several times the following fact.

▶ Fact 3. For any $i \neq j$, assignment β_j satisfies clause $\overline{\beta}_i$, since β_i and β_j must disagree on the value of some variable.

For $1 \leq i \leq t$, we show how to get the clause database $\Gamma_i = S \cup \{\overline{\beta}_1, \ldots, \overline{\beta}_i\} \setminus \{\overline{\alpha}_i\}$ where α_i is the assignment corresponding to the leaf of minimum cost among $\{\beta_1, \ldots, \beta_i\}$. Leaf β_1 is trivially an improvement, hence $\Gamma_1 = S$. To derive Γ_{i+i} from Γ_i we deal with the two types of leaves separately.

If β_{i+1} was pruned, we forbid it by adding $\overline{\beta}_{i+1}$ to the database Γ_i using the iPR rule with current best assignment α_i as witness. We need to check that $(\Gamma_i \cup \{\overline{\beta}_{i+1}\})|_{\alpha_i} \subseteq \Gamma_i|_{\beta_{i+1}}$. By the previous Fact, all clauses in $\Gamma_i \setminus S = \{\overline{\beta}_1, \ldots, \overline{\beta}_i\} \setminus \{\overline{\alpha}_i\}$ are satisfied both by α_i and by β_{i+1}, and furthermore $\overline{\beta}_{i+1}|_{\alpha_i} = \top$. The check reduces to verifying that $S|_{\alpha_i} \subseteq S|_{\beta_{i+1}}$. The right hand side $S|_{\beta_{i+1}}$ contains k_{i+1} copies of $\bot$ by definition (together possibly with other clauses), and $S|_{\alpha_i}$ contains only the clause $\bot$ with multiplicity at most k_{i+1}.

If leaf β_{i+1} corresponds to an improvement, we forbid α_i by adding $\overline{\alpha}_i$ to the database via iPR using β_{i+1} as witness, that is $(\Gamma_i \cup \{\overline{\alpha}_i\})|_{\beta_{i+1}} \subseteq \Gamma_i|_{\alpha_i}$. The procedure is the same as before, but with the role of α_i and β_{i+1} reversed. In this case $\Gamma_{i+1} = \Gamma_i \cup \{\overline{\alpha}_i\}$.

Second phase: Let α be the assignment corresponding to the optimal leaf, and k be its cost. The clause database now contains $S \cup \{\overline{\beta}_1 \ldots, \overline{\beta}_t\} \setminus \{\overline{\alpha}\}$. Our goal now is to derive k copies of all clauses forbidding all assignments at the leaves of T. From there, we derive k copies of $\bot$ by doing symmetric cuts on the tree branches, and that would conclude the proof.

To derive the missing $k-1$ copies of clauses $\overline{\beta}_i$, for $\beta_i \neq \alpha$, we use the iPR rule with α as witnessing substitution. This is similar to the pruning step, but the right hand side of the inclusion has even more copies of $\bot$. These applications of iPR rule are correct regardless of their order in the proof.

Now we derive k copies of $\overline{\alpha}$: S contains k clauses falsified by α, hence $\overline{\alpha}$ is a weakening of each of them. We do not have a weakening rule in our proof system, but we can simulate it using the split rule. We explain with an example how we do it: let us say that $\overline{\alpha} = x_1 \vee x_2 \vee \cdots \vee x_\ell$ and that $C = x_1 \vee x_2 \vee \cdots \vee x_j \in S$ for some $j \leq \ell$. We use the split rule on C to get $C \vee \overline{x}_{j+1}$ and $C \vee x_{j+1}$, then on the latter to get $C \vee x_{j+1} \vee \overline{x}_{j+2}$ and $C \vee x_{j+1} \vee x_{j+2}$, and so on. In this way we derive a copy of $\overline{\alpha}$ from each of the k falsified clauses, each of them in $O(|\alpha|)$ steps.

In total we do $k(|T| - 1)$ applications of iPR rule, and $O(k|\alpha|)$ to derive the clauses corresponding to the optimal leaf. Finally we get the k empty clauses in $O(|T|)$ steps each, resolving bottom to top in the tree using symmetric cuts.   ◀

The division of the proof into first and second phase looks artificial, but we used it to highlight how part of the proof can be logged during the BnB procedure. In the second phase we use the optimum k, that is only known at the end of the procedure. As an alternative, for any non-optimal branch $\beta_i \neq \alpha$ we could produce a sufficient number of copies of $\overline{\beta}_i$ when we add it, and use only k of them at the end, since the optimal branch α can only be produced with multiplicity k (as in the previous proof). The sufficient number of copies could be the current UB of the branch, which is always greater or equal than k. This, on the other hand, makes the proof longer, but could be avoided using the language of weighted clauses.

6 Conclusions and open problems

We convert redundancy rules SR/PR/SPR/LPR for SAT into rules iSR/iPR/iSPR/iLPR that are sound for MaxSAT. Adding such rules to MaxSAT-Resolution produces new proof systems. We exemplify their strength with short proofs of the optimal cost of hard tautologies (Section 4), and with a simulation of simple BnB procedures (Section 5). We conclude with a list of open problems.

- Can MaxSAT-Resolution+iSPR prove either $\mathrm{cost}(\mathsf{PHP}_n^m) = m - n$ or $\mathrm{cost}(\mathsf{Parity}_n) = 1$ efficiently? Polynomial-size proofs of the unsatisfiability of PHP_n^{n+1} and Parity_n are known for the system SR^- [13] but the arguments don't seem to adapt to MaxSAT-Resolution +iSPR.

- Sparse versions of Pigeonhole Principle allow each pigeon to only fly into a small selection of holes, i.e., some variables $p_{i,j}$ are set to $\bot$. In proof systems that are closed under variable restrictions, the sparse version is at least as easy as the standard version. But this closure property does not hold in proof using redundancy rules, thus it is interesting to ask whether sparse versions of pigeonhole principle are easy for MaxSAT-Resolution $+$ iSR or SR^-.

- How does MaxSAT-Resolution $+$ iSR compare with MaxSAT-Resolution+cost-SR from [8]?

- We simulate a plain BnB in MaxSAT-Resolution $+$ iPR. This suggests that iPR rule could be instrumental to simulate more sophisticated BnB algorithms. For example algorithms that integrate CDCL reasoning [28].

References

1 Albert Atserias and Tuomas Hakoniemi. Size-degree trade-offs for Sums-of-Squares and Positivstellensatz proofs. In *34th Computational Complexity Conference (CCC)*, volume 137, pages 24:1–24:20, 2019. `doi:10.4230/LIPIcs.CCC.2019.24`.

2 Albert Atserias and Massimo Lauria. Circular (yet sound) proofs in propositional logic. *ACM Trans. Comput. Log.*, 24(3):20:1–20:26, 2023. Conference version appeared in SAT'19. `doi:10.1145/3579997`.

3 Per Austrin and Kilian Risse. Perfect matching in random graphs is as hard as Tseitin. *TheoretiCS*, 1, 2022. `doi:10.46298/THEORETICS.22.2`.

4 Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maximum satisfiabiliy. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 929–991. IOS Press, 2021. `doi:10.3233/FAIA201008`.

5 Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, , and Dieter Vandesande. Certified core-guided maxsat solving. In *Proceedings of the 29th International Conference on Automated Deduction (CADE-29), July 2023*, 2023.

6   Jeremias Berg and Matti Järvisalo. Unifying reasoning and care-guided search for maximum satisfiability. In *16th European Conf. on Logics in Artificial Intelligence (JELIA)*, pages 287–303, 2019.

7   Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified dominance and symmetry breaking for combinatorial optimisation. *J. Artif. Intell. Res.*, 77:1539–1589, 2023. `doi:10.1613/JAIR.1.14296`.

8   Ilario Bonacina, Maria Luisa Bonet, Sam Buss, and Massimo Lauria. Redundancy rules for MaxSAT. *Electron. Colloquium Comput. Complex.*, TR24-045, 2024. URL: `https://eccc.weizmann.ac.il/report/2024/045`.

9   Ilario Bonacina, Maria Luisa Bonet, and Jordi Levy. Weighted, circular and semi-algebraic proofs. *Journal of Artificial Intelligence Research (JAIR)*, 79:447–482, February 2024. `doi:10.1613/jair.1.15075`.

10   Maria Luisa Bonet, Sam Buss, Alexey Ignatiev, João Marques-Silva, and Antonio Morgado. MaxSAT resolution with the dual rail encoding. In *32nd Intl. AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

11   Maria Luisa Bonet and Jordi Levy. Equivalence between systems stronger than resolution. In *23rd International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 166–181, Cham, 2020.

12   Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for max-SAT. *Artif. Intell.*, 171(8-9):606–618, 2007.

13   Sam Buss and Neil Thapen. DRAT and propagation redundancy proofs without new variables. *Logical Methods in Computer Science*, Volume 17, Issue 2, April 2021. Conference version appeared in SAT'19.

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

15   Federico Heras and Javier Larrosa. New inference rules for efficient Max-SAT solving. In *21st National Conference on Artificial Intelligence and 18th Innovative Applications of Artificial Intelligence Conference*, pages 68–73, 2006.

16   Marijn J. H. Heule and Armin Biere. What a difference a variable makes. In *24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 75–92, 2018.

17   Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Trimming while checking clausal proofs. In *Formal Methods in Computer-Aided Design (FMCAD)*, pages 181–188, 2013.

18   Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Verifying refutations with extended resolution. In *24th International Conference on Automated Deduction (CADE)*, pages 345–359, 2013.

19   Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Expressing symmetry breaking in DRAT proofs. In *25th International Conference on Automated Deduction (CADE)*, pages 591–606, 2015.

20   Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Strong extension-free proof systems. *Journal of Automated Reasoning*, 64(3):533–554, 2019. Conference version appeared in CADE'17. `doi:10.1007/s10817-019-09516-0`.

21   Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere. PRuning through satisfaction. In *Hardware and Software: Verification and Testing - 13th International Haifa Verification Conference (HVC)*, pages 179–194, 2017.

22   Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo. Clause redundancy and preprocessing in maximum satisfiability. In *Automated Reasoning*, pages 75–94. Springer International Publishing, 2022.

23   Matti Järvisalo, Marijn J. H. Heule, and Armin Biere. Inprocessing rules. In *6th International Joint Conference on Automated Reasoning (IJCAR)*, pages 355–270, 2012.

24 Benjamin Kiesl, Adrián Rebola-Pardo, and Marijn J. H. Heule. Extended resolution simulates DRAT. In *6th International Joint Conference on Automated Reasoning (IJCAR)*, pages 516–531, 2018.

25 Benjamin Kiesl, Martina Seidl, Hans Tompits, and Armin Biere. Super-blocked clauses. In *8th International Joint Conference on Automated Reasoning (IJCAR)*, volume 9706 of *Lecture Notes in Computer Science*, pages 45–61. Springer, 2016. `doi:10.1007/978-3-319-40229-1_5`.

26 Oliver Kullmann. On a generalization of extended resolution. *Discrete Applied Mathematics*, 96-97:149–176, 1999. `doi:10.1016/S0166-218X(99)00037-2`.

27 Chu Min Li and Felip Manyà. MaxSAT, hard and soft constraints. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 903–927. IOS Press, 2021. `doi:10.3233/FAIA201007`.

28 Chu-Min Li, Zhenxing Xu, Jordi Coll, Felip Manyà, Djamal Habet, and Kun He. Combining clause learning and branch and bound for MaxSAT. In *Constraint Programming (CP)*, volume 210, pages 38:1–38:18, 2021.

29 Adrián Rebola-Pardo and Martin Suda. A theory of satisfiability-preserving proofs in SAT solving. In *22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, pages 583–603, 2018.

30 Emre Yolcu and Marijn J. H. Heule. Exponential separations using guarded extension variables. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 251 of *LIPIcs*, pages 101:1–101:22, 2023. `doi:10.4230/LIPICS.ITCS.2023.101`.