Redundancy Rules for MaxSAT

Ilario Bonacina
UPC Universitat Politècnica de Catalunya, Barcelona, Spain

Maria Luisa Bonet
UPC Universitat Politècnica de Catalunya, Barcelona, Spain

Sam Buss
University of California, San Diego, CA, USA

Massimo Lauria
Sapienza Università di Roma, Rome, Italy

Abstract

The concept of redundancy in SAT leads to more expressive and powerful proof search techniques, e.g., able to express various inprocessing techniques, and originates interesting hierarchies of proof systems [Heule *et.al*'20, Buss-Thapen'19]. Redundancy has also been integrated in MaxSAT [Ihalainen *et.al*'22, Berg *et.al*'23, Bonacina *et.al*'24].

In this paper, we define a structured hierarchy of redundancy proof systems for MaxSAT, with the goal of studying its proof complexity. We obtain MaxSAT variants of proof systems such as SPR, PR, SR, and others, previously defined for SAT.

All our rules are polynomially checkable, unlike [Ihalainen *et.al*'22]. Moreover, they are simpler and weaker than [Berg *et.al*'23], and possibly amenable to lower bounds. This work also complements the approach of [Bonacina *et.al*'24]. Their proof systems use different rule sets for soft and hard clauses, while here we propose a system using only hard clauses and blocking variables. This is easier to integrate with current solvers and proof checkers.

We discuss the strength of the systems introduced, we show some limitations of them, and we give a short cost-SR proof that any assignment for the weak pigeonhole principle PHP^m_n falsifies at least $m - n$ clauses.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases MaxSAT, Redundancy Rules, Pigeonhole Principle

Digital Object Identifier 10.4230/LIPIcs.SAT.2025.7

Funding *Ilario Bonacina*: The author was supported by grant PID2022-138506NB-C22 (PROOFS BEYOND) funded by AEI.
Maria Luisa Bonet: The author was supported by grant PID2022-138506NB-C21 (PROOFS BEYOND) funded by AEI.
Massimo Lauria: The author has been supported by the project PRIN 2022 "Logical Methods in Combinatorics" N. 2022BXH4R5 of the Italian Ministry of University and Research (MIUR).

Acknowledgements The authors would like to thank the Simons Institute for the Theory of Computing: part of this work has been done during the "Extended Reunion: Satisfiability" program (Spring 2023). Another part of this work has been done during the Oberwolfach workshop 2413 "Proof Complexity and Beyond" and during the 2023 Workshop on Proof Theory and its Applications organized by the Proof Society.

1 Introduction

This paper investigates new proof systems for MaxSAT that incorporate redundancy inferences tailored to work for MaxSAT. Redundancy inferences were introduced as extensions to SAT solvers to allow non-implicational inferences that preserve satisfiability and non-satisfiability. For resolution and SAT solvers, the first redundancy inferences were based on blocked

clauses (BC) [25] and Resolution Asymmetric Tautology (RAT) [22, 15]. Other work on redundancy reasoning includes [14, 16, 20, 23, 17]; and, of particular relevance to the present paper, are the work of Heule, Kiesl, and Biere [19], and the work of Buss and Thapen [9]. Redundancy inferences formalize "*without loss of generality*" reasoning [30] and can substantially strengthen resolution and, in some cases, the effectiveness of SAT solvers for hard problems such as the pigeonhole principle (PHP) [20]. Indeed, in their full generality, redundancy inferences allow resolution to polynomially simulate extended resolution.

MaxSAT is a generalization of SAT; it is the problem of determining a truth assignment for a CNF formula that minimizes the number of falsified clauses. Although the MaxSAT problem is inherently more difficult than SAT, in some cases MaxSAT can be adapted to be more efficient in practice than CDCL solvers for hard problems such as PHP [7]. There are several approaches to MaxSAT solvers, including MaxSAT resolution [8, 26], core-guided MaxSAT [12, 1, 28, 27, 29], and maximum-hitting-set MaxSAT [2, 11, 31] ; the present paper discusses only MaxSAT resolution. The MaxSAT resolution proof system was first defined by Larrosa and Heras [26] and proved completed by Bonet, Levy and Manya [8].

We define cost preserving redundancy rules for MaxSAT mirroring the redundancy rules for SAT, called "cost-BC", "cost-LPR", "cost-SPR", "cost-PR", and "cost-SR" (see Definition 3.2). The strongest of these is "cost-SR" based on the substitution redundancy (SR) [9]. All five of these new inferences are sound for MaxSAT reasoning (Theorem 4.2). Furthermore, we prove that cost-SPR, cost-PR and cost-SR are complete for MaxSAT (Theorem 4.3). On the other hand, we prove that cost-LPR and cost-BC are incomplete for MaxSAT (Corollary 5.2). We illustrate the power of cost-SR by giving polynomial size proofs of the cost of the blocking-variable version of weak pigeonhole principle bPHP_n^m for arbitrary numbers $m > n$ of pigeons and holes (Theorem 6.3).

Ours is not the first paper bringing redundancy reasoning to the context of optimization. For instance, the work of Ihalainen, Berg and Järvisalo [21], building on [4], introduced versions of redundancy inferences that work with MaxSAT. In contrast to the system CPR of [21], all of our "cost-" inferences are polynomial-checkable for validity, and thus all give traditional Cook-Reckhow proof systems for MaxSAT.

Another system with redundancy rules for certifying unsatisfiability and optimization is veriPB [5]. 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 [3]. The propositional fragment of veriPB is as strong as Extended Resolution and DRAT, and plausibly even stronger [24]. In contrast, our systems are plausibly weaker and simpler, but yet strong enough to prove efficiently interesting formulas. Still our systems might be amenable to proving lower bounds. Indeed, our explicit goal is to study the proof complexity of redundancy rules for MaxSAT in a similar vein of what [9] does for SAT, something that is beyond the scope of veriPB, focused on proof logging actual solvers. As a starting point, we show that "cost-BC" and "cost-LPR" are not complete (Corollary 5.2) and we show width lower bounds on "cost-SPR" and an analogue of a width lower bound for "cost-SR" (Corollary 5.3). Finally there is another work that tries to modify the redundancy rules to make them compatible with the MaxSAT model of soft and hard clauses [6]. That work has a similar spirit to this one, but departs in the redundancy test, which checks for inclusion rather than for reverse unit propagation. Such choice makes the inference weaker but preserves the number of unsatisfied clauses, hence the rule applies directly to soft clauses.

Before proceeding with the description and analysis of our systems, we should highlight two aspects of redundancy inference that are somehow orthogonal to the choice of the concrete inference rule: clause deletion and new variable introduction. The applicability

of a redundancy rule depends on the clauses present in the database, and it seems that allowing deletion of past clauses makes the system stronger, and indeed collapses together the power of several types of inference rules (see [9]). Likewise the possibility of introducing new variables in redundant clauses makes all such systems as powerful as extended resolution [25]. Coherently with the stated goal of having systems that are simple and amenable to proof complexity analysis, in this paper we do not allow neither clause deletion nor new variables.

Structure of the paper

Section 2 contains all the necessary preliminaries, including notation on MaxSAT and the blocking variables encoding of MaxSAT instances (blocking variables are also used by [21]). Section 3 introduces the redundancy rules for MaxSAT, proves their basic properties, and defines calculi based on those rules. Section 4 shows their soundness, and their completeness. Section 5 shows the incompleteness of cost-BC/cost-LPR and some limitations of cost-SPR and even cost-SR. Section 6 gives examples of applications of the redundancy rules, including a polynomial size proof of the optimal cost of the weak Pigeonhole Principle and a general result about the polynomial size provability of minimally unsatisfiable formulas. Section 7 gives some concluding remarks.

2 Preliminaries

For a natural number n, let $[n]$ be the set $\{1, \dots, n\}$. Sets and multi-sets are denoted with capital Roman or Greek letters.

Propositional logic notation

A *Boolean variable* x takes values in $\{0, 1\}$. A *literal* is either a variable x or its negation $\overline{x}$. A *clause* is a finite disjunction of literals, i.e., $C = \bigvee_i \ell_i$. The empty clause is $\bot$. A formula in *Conjunctive Normal Form* (CNF) is a conjunction of clauses $\Gamma = \bigwedge_j C_j$. We identify a CNF with the multiset of its clauses, and denote as $|\Gamma|$ the number of its clauses (counted with multiplicity). We denote as $\mathrm{Var}(\Gamma)$ the set of variables in Γ.

Substitutions and assignments

A *substitution* σ for a set of variables X is a function so that $\sigma(x)$ is either 0, 1 or some literal defined on X. For convenience, we extend a substitution σ to constants and literals, setting $\sigma(0) = 0$, $\sigma(1) = 1$, and $\sigma(\overline{x}) = \overline{\sigma(x)}$ for any variable $x \in X$. The composition of two substitutions σ, τ is the substitution $\sigma \circ \tau$, where $\sigma \circ \tau(x) = \sigma(\tau(x))$ for $x \in X$. A substitution σ is an *assignment* when $\sigma(x) \in \{0, 1, x\}$ for any $x \in X$. The *domain* of an assignment σ is $\mathrm{dom}(\sigma) = \sigma^{-1}(\{0, 1\})$, and σ is a *total assignment* over X if X is its domain, i.e., σ maps all variables in X to Boolean values. Given a clause $C = \bigvee_i \ell_i$ and a substitution σ, the clause C *restricted* by σ, is $C\restriction_\sigma = \bigvee_i \sigma(\ell_i)$, simplified using the usual logic rules, i.e., $D \vee 0 = D$, $D \vee 1 = 1$, and $D \vee \ell \vee \ell = D \vee \ell$. Another notation for $C\restriction_\sigma$ is $\sigma(C)$. If $\sigma(C) = 1$ or $\sigma(C)$ is tautological we say that $\sigma \vDash C$, i.e., σ *satisfies* C.

The *restriction* of a CNF formula Γ by σ, denoted as $\Gamma\restriction_\sigma$, is the conjunction of all clauses $C\restriction_\sigma$ where $C \in \Gamma$ and $\sigma(C) \neq 1$. The CNF $\Gamma\restriction_\sigma$ is also a multiset. We say that σ *satisfies* Γ ($\sigma \vDash \Gamma$) if for every $C \in \Gamma$, $\sigma \vDash C$, i.e., $\Gamma\restriction_\sigma = \emptyset$. We say that $\Gamma \vDash C$ if for every substitution σ, if $\sigma \vDash \Gamma$ then $\sigma \vDash C$.

We identify a literal ℓ with the substitution that assigns ℓ to 1 and leaves all other variables unassigned. Hence we use notations like $\Gamma\restriction_\ell$. Likewise, given a clauses C we denote as $\overline{C}$ the assignment that maps all literals in C to false, and we use the notation $\Gamma\restriction_{\overline{C}}$.

Unit propagation

A *unit clause* is a clause of just one literal. Unit propagation works as follows. Start with a CNF Γ: if Γ has no unit clauses, the process ends, otherwise pick some unit clause ℓ in Γ arbitrarily, remove from Γ all clauses containing ℓ and remove literal $\bar{\ell}$ from all clauses containing it. Keep repeating until Γ has no more unit clauses. Regardless of the choice of the unit clause, the process always ends with the same formula.

We say that $\Gamma \vdash_1 C$ when the application of unit propagation to the formula $\Gamma\restriction_{\overline{C}}$ produces the empty clause. For two CNF formulas Γ, Δ we say that $\Gamma \vdash_1 \Delta$ if for every $D \in \Delta$, $\Gamma \vdash_1 D$. Clearly, if $\Gamma \supseteq \Delta$ then $\Gamma \vdash_1 \Delta$, and if $\Gamma \vdash_1 \Delta$, then $\Gamma \vDash \Delta$. It is important to stress that the $\vdash_1$ relation is *efficiently checkable*.

▶ **Observation 2.1** ([9, Fact 1.3])**.** *Let σ be a substitution and Γ, Δ be CNF formulas, if $\Gamma \vdash_1 \Delta$, then $\Gamma\restriction_\sigma \vdash_1 \Delta\restriction_\sigma$.*

Resolution

Resolution is a well-studied propositional deduction system with two inference rules: (i) from a clause A we can deduce any B s.t. $A \subseteq B$; (ii) from clauses $A \vee x$ and $B \vee \overline{x}$ we can deduce $A \vee B$. A *resolution proof* from a set of clauses Γ is a sequence of clauses $D_1, D_2, \ldots, D_t$ where each D_i is either already in Γ or is deduced from earlier clauses in the sequence using one of the two inference rules. Resolution is complete, thus deciding whether a clause C can be deduced from Γ is the same as deciding whether $\Gamma \vDash C$.

MaxSAT

Given a CNF formula F, MaxSAT asks to find the maximum number of clauses in F which can be simultaneously satisfied. In applications, it is useful to consider a generalization for which we divide the clauses into *hard* or *soft* (*partial* MaxSAT). Hard clauses must be satisfied, while soft clauses can be falsified with a cost. Consider $F = H \wedge S$ where H is the multiset of hard clauses and S is the multiset of soft ones. In this model, MaxSAT asks to find the maximum number of clauses in S that can be simultaneously satisfied by an assignment that satisfies all clauses in H. Observe that the optimization problem is not well defined if H is not satisfiable.[1] It is not relevant whether H is a set or a multiset. In S, on the other hand, the multiplicity of soft clauses must be accounted for.

Proof systems for MaxSAT aim to show lower bounds on the cost of (partial) MaxSAT instances, one such system is MaxSAT resolution.

MaxSAT with blocking variables

Without loss of generality we can assume that all soft clauses in a MaxSAT instance are unit clauses; indeed, using a new variable b, a soft clause C can be replaced with a hard clause $C \vee b$ and a soft clause $\overline{b}$, without affecting the cost. The variable b is usually called a *blocking variable*. This appears in [13], but it might have been used even earlier.

[1] An even more general version is *weighted* MaxSAT, where we would consider *weighted* set of clauses (F, w), i.e., each clauses $C \in F$ has an associated weight $w(C)$ where $w : F \to \mathbb{N} \cup \{\infty\}$. In this model the goal is to minimize the weight of the falsified clauses. The role of the weight ∞ is to model *hard* clauses. In this paper we do not focus on this model.

▶ **Definition 2.2.** *Let $F = H \wedge S$ with soft clauses $S = C_1 \wedge \cdots \wedge C_m$. The* blocking variables formulation *of F is $F' = H' \wedge S'$ where*

- $H' = H \wedge (C_1 \vee b_1) \wedge \cdots \wedge (C_m \vee b_m)$,
- $S' = \overline{b_1} \wedge \cdots \wedge \overline{b_m}$,

and $b_1, \ldots, b_m$ are new variables (blocking variables) *not appearing in F. We say that Γ is a MaxSAT instance* encoded with blocking variables*, when it is given as a set of hard clauses of the form as in H' above. The soft clauses, then, are implicit.*

▶ **Observation 2.3.** *Let $F = H \wedge S$ be a MaxSAT instance and $F' = H' \wedge S'$ be the blocking variables formulation of F. Any assignment that satisfies H and falsifies k clauses in S can be extended to an assignment that satisfies H' and sets k blocking variables to true. Vice versa, any assignment that satisfies H' and sets k blocking variables to true satisfies H too and falsifies at most k clauses in S.*

Because of Observation 2.3, for the rest of this work we consider Γ to be a MaxSAT instance encoded with blocking variables usually named $\{b_1, \ldots, b_m\}$. The goal is to satisfy Γ while setting to true the least number of blocking variables. More formally, given a total assignment α for Γ, we define

$$\mathrm{cost}(\alpha) = \sum_{i=1}^{m} \alpha(b_i) \quad \text{and} \quad \mathrm{cost}(\Gamma) = \min_{\alpha \,:\, \alpha \vDash \Gamma} \mathrm{cost}(\alpha)$$

and the goal is to find the value of $\mathrm{cost}(\Gamma)$. Notice that, the notation $\mathrm{cost}(\alpha)$ is defined even for assignments not satisfying Γ.

3 Redundancy rules for MaxSAT

In the context of SAT, a clause C is redundant w.r.t. a CNF instance Γ if Γ and $\Gamma \cup \{C\}$ are equisatisfiable, that is either they both are satisfiable or both unsatisfiable [25]. The natural adaptation of this notion to MaxSAT is a clause C that does not affect the cost of Γ.

▶ **Definition 3.1** (redundant clause, [21])**.** *A clause C is* redundant *w.r.t. a MaxSAT instance Γ when*

$$\mathrm{cost}(\Gamma) = \mathrm{cost}(\Gamma \cup \{C\}) \,. \tag{1}$$

Clauses that logically follow from Γ are obviously redundant, but there may be other useful clauses that do not follow logically, and yet do not increase the cost if added.

The condition in eq. (1) is not polynomially checkable (unless, say P = NP). Therefore, we consider efficiently certifiable notions of redundancy, i.e., ways to add redundant clauses (in the sense of eq. (1)) while certifying efficiently their redundancy. This is done showing how to extend in a systematic way the notions of efficiently certifiable redundancy already studied in the context of SAT (BC, RAT, LPR, SPR, PR, SR) [19, 9] to the context of MaxSAT. This is an alternative to the approach of [21]. This definition could also be seen as a very special case of veriPB [5] (see Section 3.1 for more details on the connections with veriPB).

▶ **Definition 3.2.** *A clause C is* cost substitution redundant *(*cost-SR*) w.r.t. to Γ if there exists a substitution σ such that*

1. $\Gamma \restriction_{\overline{C}} \vdash_1 (\Gamma \cup \{C\})|_\sigma$ (redundancy)
2. *for all total assignments $\tau \supseteq \overline{C}$,* $\mathrm{cost}(\tau \circ \sigma) \leq \mathrm{cost}(\tau)$ (cost).

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

Cost propagation redundant (cost-PR) *if σ is a partial assigment.*

Cost subset propagation redundant (cost-SPR) *if σ is a partial assigment with the same domain as $\overline{C}$. In other words, σ flips some variables in $\overline{C}$.*

Cost literal propagation redundant (cost-LPR) *if σ is a partial assignment with the same domain as $\overline{C}$, but differs from $\overline{C}$ on exactly one variable.*

Cost blocked clause (cost-BC) *if σ is a partial assignment with the same domain as $\overline{C}$, which differs from $\overline{C}$ on exactly one variable v, and moreover, for every clause $D \in \Gamma$ containing the variable v, $\sigma \vDash D$.*[2]

Item 1 in Definition 3.2 claims that adding C does not make Γ unsatisfiable, unless it was already the case. Together with Item 2, it ensures that any assignment that falsifies the new clause C can be patched with a substitution σ so that C is satisfied without increasing the minimum cost (see Lemma 3.4).

Indeed, Item 1 in Definition 3.2 and its variants correspond to the redundancy rules of proof systems SR, PR, SPR, and LPR from [9], adapted here to consider cost. Since LPR is the same as RAT (see [9, Theorem 1.10]), the notion of cost literal propagation redundancy could as well be called *cost*-RAT *redundancy*.

▶ Remark 3.3. It is important to compare Definition 3.2 with [21, Definition 2]. Redundancy conditions are very similar and the main differences are in the cost conditions. Let us compare their `CPR` rule with our cost-PR . In cost-PR, the cost condition requires the witness σ to be at least as good as all possible extensions of $\overline{C}$, while in `CPR` the requirement is enforced only on those extensions of $\overline{C}$ that satisfy Γ. The latter more permissive condition allows to derive more clauses, but it is unlikely to be polynomially checkable, while the condition in cost-PR is polynomially checkable (see Lemma 3.5). In [21], the authors also define two polynomially checkable rules where the cost condition is not present, but implicitly enforced via restrictions on the type of assignments used. Those rules are special cases of cost-LPR and cost-SPR respectively.

▶ **Lemma 3.4.** *If C is* cost-SR *w.r.t. to Γ, then C is redundant w.r.t. Γ.*

Proof. It is enough to show that $\text{cost}(\Gamma) \geq \text{cost}(\Gamma \cup \{C\})$. Let $\text{cost}(\Gamma) = k$. To show that adding C to Γ does not increase the cost, consider an optimal total assignment α that satisfies Γ and sets to true exactly k blocking variables. If $\alpha \vDash C$ we already have that $\alpha \vDash \Gamma \cup \{C\}$ and $\text{cost}(\alpha) = k$. Otherwise, α extends $\overline{C}$ and, by assumption, there is a substitution σ such that $\text{cost}(\alpha \circ \sigma) \leq k$. To show that $\text{cost}(\Gamma \cup \{C\}) \leq k$, it remains to show that indeed $\alpha \circ \sigma \vDash \Gamma \cup \{C\}$. By assumption,

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

and, since $\alpha \vDash \Gamma$ and extends $\overline{C}$, then $\alpha \vDash (\Gamma \cup \{C\})|_\sigma$ too. Equivalently, $\alpha \circ \sigma \vDash \Gamma \cup \{C\}$. ◀

Both Item 1 and Item 2 of Definition 3.2 are stronger than what is actually needed for Lemma 3.4 to hold. Indeed, for Item 1, it would be enough that $\Gamma \restriction_{\overline{C}} \vDash (\Gamma \cup \{C\})|_\sigma$, and, for Item 2, it would be sufficient to check it for any $\tau \supseteq \overline{C}$ such that $\tau \vDash \Gamma$. Unfortunately, these latter versions of Item 1 and Item 2 are in general not polynomially checkable. Instead, our conditions are checkable in polynomial time.

[2] The definition of *blocked clause* is written to match the previous definitions. In this case, the redundancy condition is always satisfied.

▶ **Lemma 3.5.** *Let Γ be a MaxSAT instance, C a clause and σ a substitution. There is a polynomial time algorithm to decide whether C is* cost-SR *w.r.t. Γ, given the substitution σ.*

Proof (sketch). The redundancy condition in Definition 3.2 is polynomially checkable, since it is a unit propagation. To check the cost condition we need to check whether

$$\max_{\tau \supseteq \overline{C}} (\text{cost}(\tau \circ \sigma) - \text{cost}(\tau)) \tag{2}$$

is at most 0. This amounts to maximizing the function

$$\Big(\sum_{i=1}^{n} \sigma(b_i) - b_i \Big) \Big|_{\overline{C}} \tag{3}$$

over all assignments on variables in $\text{Var}(\Gamma) \setminus \text{Var}(\overline{C})$. Observe that eq. (3) is a function of the form $c + \sum_{v \in \text{Var}(\Gamma) \setminus \text{Var}(\overline{C})} c_v \cdot v$, for suitable constants c and c_vs. Therefore its maximum can be easily determined in polynomial time and corresponds to assigning each v to 1 when $c_v > 0$ and to 0 otherwise. ◀

Lemma 3.5 allows the definition of proof systems that extend resolution using cost redundancy rules in the sense of Cook-Reckhow [10].

▶ **Definition 3.6** (cost-SR calculus)**.** *The* cost-SR calculus *is a proof system for MaxSAT. A derivation of a clause C from a MaxSAT instance Γ (encoded with blocking variables) is a sequence of clauses $D_1, D_2, \ldots, D_t$ where, $C \in \Gamma \cup \{D_i\}_{i \in [t]}$, each D_i is either already in Γ or is deduced from earlier clauses in the sequence using a resolution rule, or D_i is* cost-SR *w.r.t. to $\Gamma \cup \{D_1, \ldots, D_{i-1}\}$ with $\text{Var}(D_i) \subseteq \text{Var}(\Gamma)$.*[3] *The length of such derivation is t, i.e., the number of derived clauses. To check the validity of derivations in polynomial time, any application of the* cost-SR *rule comes accompanied by the corresponding substitution that witnesses its soundness.*

If the goal is to certify that $\text{cost}(\Gamma) \geq s$, we can accomplish this deriving s distinct unit clauses of the form $\{b_{i_1}, \ldots, b_{i_s}\}$ (see Theorem 4.2). If the goal is to certify that $\text{cost}(\Gamma) = s$, we can accomplish this deriving s distinct unit clauses of the form $\{b_{i_1}, \ldots, b_{i_s}\}$ together with the unit clauses $\{\overline{b}_j : j \notin \{i_1, \ldots, i_s\}\}$ (see Theorem 4.2).

In a similar fashion we define cost-PR, cost-SPR, cost-LPR, and cost-BC calculi. A remarkable aspect of these calculi is that a proof must somehow identify the blocking variables to be set to true. When there are multiple optimal solutions, it is quite possible that none of the b_i follows logically from Γ. Nevertheless, the redundancy rules, often used to model "without loss of generality" reasoning, can reduce the solution space.

3.1 Simulation by veriPB

We show that cost-SR and its subsystems are p-simulated by veriPB. This is actually not surprising since veriPB proof steps are based on cutting planes, which in turn easily simulates resolution. Furthermore veriPB includes the SAT redundancy rules equipped with some criteria of cost preservation. veriPB can argue within the proof whether some substitution is cost preserving. If veriPB can do that for the substitutions that respect the second condition in Definition 3.2, then it p-simulates cost-SR.

[3] We only consider the case where no new variables are added via cost-SR rules. To be coherent with [9], cost-SR should be cost-SR$^-$, following the notational convention of adding an exponent with "$-$" to denote SR, PR and SPR when the systems are not allowed to introduce new variables. We ignore that convention to ease an already rich notation.

Since veriPB is a complex system, we describe the minimal fragment needed to simulate cost-SR. Consider a CNF formula F and some objective linear function f to be minimized. A veriPB derivation starts with F encoded as linear inequalities. At each step a linear inequality C is derived from the set Γ of previously derived inequalities in one of the following ways:

1. C is derived from Γ using some cutting planes rule;
2. there exists a substitution σ so that $\Gamma \cup \overline{C} \vdash (\Gamma \cup \{C\}) \restriction_\sigma \cup \{f \restriction_\sigma \leq f\}$,where the symbol $\vdash$ means that there is a cutting planes proof that witnesses the implication.

In Item 2 the cutting plane proof is included in the veriPB proof to make it polynomially verifiable, unless it just consists in unit propagation steps or certain uses of Boolean axioms. In the latter cases the verifier can efficiently recover such derivation steps on its own.

The resolution inference rule of cost-SR can be simulated using 1. The redundancy inference rule of cost-SR is simulated by 2. The redundancy condition is syntactically the same, while the cost condition is argued via a cutting planes proof. The latter is the only non-trivial part. This argument gives a proof sketch of the following proposition.

▶ **Proposition 3.7.** *Let F be a MaxSAT instance over n variables with blocking variables $b_1, b_2, \ldots, b_k$. If there is a t step* cost-SR *derivation of a set of clauses Γ from it, then there is a* veriPB *proof from F and objective $f = \sum_{i=1}^{k} b_i$ of some $\Gamma' \supseteq \Gamma$ in* poly(nt) *steps.*

4 Soundness and completeness

The calculi cost-SR/cost-PR/cost-SPR are *sound* and *complete*. Before proving the soundness we show an auxiliary lemma, that shows that when the calculus certifies the lower bound for the optimal values, it can also certify the optimality.

▶ **Lemma 4.1.** *Let Γ be a MaxSAT instance encoded with blocking variables $b_1, \ldots, b_m$, of* cost$(\Gamma) = k$*, and suppose Γ contains the unit clauses $b_{i_1}, \ldots, b_{i_k}$. Then* cost-PR *can prove $\overline{b}_j$ for each $j \notin \{i_1, \ldots, i_k\}$ in $\mathcal{O}(km)$ steps.*

Proof. Let σ be a total assignment satisfying Γ with cost$(\sigma) = k$, that is σ maps all the b_{i_ℓ}s to 1 and the other blocking variables to 0. We derive all the clauses

$$C_j = \overline{b}_{i_1} \vee \cdots \vee \overline{b}_{i_k} \vee \overline{b}_j$$

with $j \notin \{i_1, \ldots, i_k\}$ using the cost-PR rule. For all clauses C_j, the substitution witnessing the validity of the cost-PR rule is always σ. The redundancy condition from Definition 3.2 is trivially true since Γ union an arbitrary set of C_js is mapped to 1 under σ. The cost condition is true because for every $\tau \supseteq \overline{C_j}$, cost$(\tau) \geq k + 1$ and cost$(\tau \circ \sigma)$ = cost$(\sigma) = k$.

To conclude, by resolution, derive $\overline{b}_j$ from C_j and the unit clauses $b_{i_1}, \ldots, b_{i_k}$. ◀

▶ **Theorem 4.2** (soundness of cost-SR)**.** *Let Γ be a MaxSAT instance encoded with blocking variables. If there is a* cost-SR *proof of k distinct blocking variables, then* cost$(\Gamma) \geq k$*. If there is a* cost-SR *proof of k distinct blocking variables $\{b_{i_1}, \ldots, b_{i_k}\}$ and all the unit clauses $\overline{b}_j$ for $j \notin \{i_1, \ldots, i_k\}$, then* cost$(\Gamma) = k$*.*

Proof. Let $b_1, \ldots, b_m$ be the blocking variables of Γ. Let Γ' the set of clauses in Γ plus all the clauses derived in the proof of cost$(\Gamma) \geq k$. That is Γ' also contains contains k distinct unit clauses $b_{i_1}, \ldots, b_{i_k}$, hence cost$(\Gamma') \geq k$. By Lemma 3.4, the cost is preserved along proof steps, therefore cost(Γ) = cost$(\Gamma') \geq k$. In the case where we have all the $\overline{b}_j$s then cost$(\Gamma') = k$ and therefore cost$(\Gamma) = k$. ◀

As an immediate consequence of Theorem 4.2, also all the cost-PR, cost-SPR, cost-LPR calculi are sound. Moreover, we can always prove the optimal lower bound in cost-SPR calculus.

▶ **Theorem 4.3** (completeness of cost-SPR). *Let Γ be a MaxSAT instance encoded with blocking variables, of* $\mathrm{cost}(\Gamma) = k$. *There is a* cost-SPR *derivation of the unit clauses $b_{i_1}, \ldots, b_{i_k}$ for some distinct k blocking literals and all the $\bar{b}_j$ for $j \notin \{i_1, \ldots, i_k\}$.*

Proof. Let $b_1, \ldots, b_m$ be the blocking variables of Γ. Take α_{opt} to be an optimal assignment, that is $\alpha_{\mathrm{opt}} \vDash \Gamma$, $\mathrm{cost}(\alpha_{\mathrm{opt}}) = k$, and for every total assignment β that satisfies Γ, $\mathrm{cost}(\beta) \geq k$. Without loss of generality we can assume α_{opt} sets variables $b_1, \ldots, b_k$ to 1 and the remaining b_js to 0.

Given any assignment γ, let $\bar{\gamma}$ be the largest clause falsified by γ. Let Σ be the set of all clauses $\bar{\gamma}$ where γ is a total assignment that satisfies Γ is different from α_{opt}. We want to derive Σ from Γ, essentially forbidding any satisfying assignment except for α_{opt}.

We can add all clauses in Σ one by one by the cost-SPR rule. Indeed, for any clause $\bar{\gamma} \in \Sigma$ and any $\Sigma' \subseteq \Sigma$, the clause $\bar{\gamma}$ is cost-SPR w.r.t. $\Gamma \cup \Sigma'$, with α_{opt} as the witnessing assignment. The redundancy condition

$$(\Gamma \cup \Sigma') \restriction_{\gamma} \vdash_1 (\Gamma \cup \Sigma' \cup \bar{\gamma}) \restriction_{\alpha_{\mathrm{opt}}}$$

holds because $\alpha_{\mathrm{opt}} \vDash \Gamma \cup \Sigma' \cup \bar{\gamma}$ so the RHS is just true. The cost condition holds by optimality of α_{opt}. In the end, the only assignment that satisfies $\Gamma \cup \Sigma$ is α_{opt}. By the completeness resolution we can prove all its literals, in particular literals b_i for $1 \leq i \leq k$ and literals $\bar{b}_i$ for $i > k$. ◀

5 Incompleteness and Width Lower bounds

Theorem 4.3 shows the completeness of cost-SPR, hence for cost-PR and cost-SR. It is not a coincidence that the proof does not apply to cost-LPR, indeed the latter is not complete. We will see that for some redundant clause derived according to Definition 3.2, the number of values that partial assignment σ flips with respect to any $\tau \supseteq \overline{C}$ may have to be large. In a cost-LPR proof this number is always at most one. For a redundant clause C derived in some subsystem of cost-SR via a witness substitution σ we define

$$\mathrm{flip}(C, \sigma) = \max_{\tau \supseteq \overline{C}} \mathrm{HammingDistance}(\tau, \tau \circ \sigma) \ ,$$

where HammingDistance is the number of different bits between two total assignments. The following result shows sufficient conditions for $\mathrm{flip}(C, \sigma)$ to be large.

▶ **Theorem 5.1.** *Let Γ be a MaxSAT instance encoded with blocking variables, of* $\mathrm{cost}(\Gamma) = k$, *and let A be the set of optimal total assignments for Γ, i.e., $\alpha \in A$ when $\alpha \vDash \Gamma$ and* $\mathrm{cost}(\alpha) = k$. *If A is such that*

1. *all pairs of assignments in A have Hamming distance at least d, and*
2. *for every blocking variable b there are $\alpha, \beta \in A$ s.t. $\alpha(b) = 0$ and $\beta(b) = 1$,*

then to derive any blocking literal b, cost-SR *must derive a redundant clause with* $\mathrm{flip}(C, \sigma) \geq d$, *where σ is the witnessing substitution for C.*

Proof. Consider a cost-SR derivation from Γ as a sequence $\Gamma_0, \Gamma_1, \ldots, \Gamma_s$ where each $\Gamma_{i+1} := \Gamma_i \cup \{C\}$ with C either derived by resolution from clauses in Γ_i, or C is cost-SR w.r.t. Γ_i. For $0 \leq j \leq s$, let $\mu(j)$ be the number of the optimal assignments for Γ_j.

At the beginning $\mu(0) = |A|$ by construction. If at some point Γ_j contains some clause b_i, then the value $\mu(j)$ must be strictly smaller than $|A|$ because A contains at least some assignment with $\{b_i \mapsto 0\}$ (by assumption 2).

Let j be the first step where $\mu(j)$ drops below $|A|$. The clause C introduced at that moment must be cost-SR w.r.t. Γ_{j-1}, because the resolution steps do not change the set of optimal assignments. Let then σ be the witnessing substitution used to derive C, we have

$$\Gamma_{j-1}\restriction_{\overline{C}} \vdash_1 (\Gamma_{j-1} \cup \{C\})\restriction_{\sigma} \ . \tag{4}$$

Since $\mu(j)$ dropped below $|A|$, clause C must be incompatible with some $\tau \in A$, that is $\tau \supseteq \overline{C}$. Therefore, by cost preservation,

$$\text{cost}(\tau \circ \sigma) \leq \text{cost}(\tau) = k \ . \tag{5}$$

By Observation 2.1, eq. (4) implies that

$$\Gamma_{j-1}\restriction_{\tau} \vdash_1 (\Gamma_{j-1} \cup \{C\})\restriction_{\tau\circ\sigma} \ .$$

Since j was the first moment when $\mu(j) < |A|$ we have that $\tau \vDash \Gamma_{j-1}$ and therefore $\tau \circ \sigma \vDash \Gamma_{j-1} \cup \{C\}$. In particular, $\tau \circ \sigma \vDash \Gamma$. By eq. (5) then it must be $\tau \circ \sigma \in A$. But then, by assumption 1, τ and $\tau \circ \sigma$ have Hamming distance at least d, which implies $\text{flip}(C, \sigma) \geq d$. ◀

We see an example application of this result to cost-LPR, where the number of values allowed to be flipped by the rule is at most 1.

The formula $F = \{x \vee y \vee b_1, \overline{x} \vee b_2, \overline{y} \vee b_3\}$ has cost 1 and its optimal assignments to variables x, y, b_1, b_2, b_3 are $\{00100, 10010, 01001\}$. These assignments fulfil the premises of Theorem 5.1, with Hamming distance > 1. Therefore cost-LPR cannot prove the cost of F to be 1, and hence cost-LPR is incomplete.

▶ **Corollary 5.2.** *Proof systems* cost-LPR*,* cost-BC *are incomplete.*

▶ **Corollary 5.3.** *There is a formula family F_n with $O(n)$ variables, $O(n)$ clauses and $\text{cost}(F_n) = \Omega(n)$ where, in order to prove $\text{cost}(F_n) \geq 1$, any* cost-SR *proof derives a redundant clause with $\text{flip}(C, \sigma) = \Omega(n)$, where σ is the witnessing substitution for C.*

Proof. We define F_n on variables $x_0, x_1, \ldots, x_n$, and variables $y_0, y_1, \ldots, y_n$. The formula contains hard clauses to encode the constraint $x_0 \neq y_0$, and constraints $x_0 = x_i$, $y_0 = y_i$ for $i \in [n]$. Furthermore F_n has the soft clauses, encoded as hard clauses with blocking variables, $\neg x_i \vee b_i$ and $\neg y_i \vee b_{i+n}$ for $i \in [n]$.

To satisfy the formula an assignment must either set all x's to true and y's to false, or vice versa. Both such assignments set to true n of the blocking variables, and no blocking variable is fixed to a constant value. Therefore the claim follows from Theorem 5.1. ◀

▶ **Corollary 5.4.** *Any* cost-SPR *proof for a formula respecting the hypothesis of Theorem 5.1 requires some redundant clauses of width at least d.*

Proof. Any redundant clause C derived in cost-SPR must have $|C| \geq \text{flip}(C, \sigma)$, and by Theorem 5.1, $\text{flip}(C, \sigma) \geq d$. ◀

For instance, the formula F_n from Corollary 5.3 requires cost-SPR refutations of width $\Omega(n)$.

6 Short proofs using redundancy rules

We show applications demonstarting the power of the redundancy rules on notable families of CNF formulas. In Section 6.1 we consider minimally unsatisfiable formulas, while in Section 6.2 we consider the *weak Pigeonhole Principle*.

▶ Remark 6.1. Due to Theorem 4.2 and Theorem 4.3, we refer to a cost-SPR (resp. cost-PR, cost-SR) derivation from Γ of $b_{i_1}, \dots, b_{i_k}$ for some distinct k blocking literals and all the $\overline{b}_j$ for $j \notin \{i_1, \dots, i_k\}$, as a *proof of* $\text{cost}(\Gamma) = k$ in cost-SPR (resp. cost-PR, cost-SR).

6.1 Short proofs of minimally unsatisfiable formulas

Recall the definition of PR from [18] (see also [9, Definition 1.16]). A PR calculus refutation of a CNF formula Γ is a sequence of clauses $D_1, \dots, D_t$ where $D_t = \bot$, and each D_{i+1} is either a clause in Γ, or derived by resolution, or is PR w.r.t. $\Gamma_i = \Gamma \cup \{D_1, \dots, D_i\}$, that is D_{i+1} satisfies

$$\Gamma_i \restriction_{\overline{D_{i+1}}} \vdash_1 (\Gamma_i \cup \{D_{i+1}\})|_\sigma \ ,$$

that is, Item 1 of Definition 3.2 for a σ which is a partial assignment. A PR refutation is a PR derivation of $\bot$. The *size* of a refutation is the number of clauses in it.

An unsatisfiable set Γ of clauses is *minimally unsatisfiable* if no proper subset of Γ is unsatisfiable.

▶ **Theorem 6.2.** *If a minimally unsatisfiable CNF formula* $\{C_1, \dots, C_m\}$ *has a* PR *refutation of size* s*, then there is a* cost-PR *proof of* $\text{cost}(\{C_1 \vee b_1, \dots, C_m \vee b_m\}) = 1$ *of at most* $\mathcal{O}(s+m)$ *many clauses.*

Proof. Let $F = \{C_1, \dots, C_m\}$ and $\Gamma = \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ be the corresponding MaxSAT instance. Let $\pi = (D_1, \dots, D_s)$ be a PR refutation of F, so $D_s = \bot$. First we show that

$$\pi_B = (D_1 \vee B, \dots, D_s \vee B) \ ,$$

with $B = \bigvee_{i \in [m]} b_i$, is a valid cost-PR derivation of B from Γ. In particular, assuming we already derived the first i steps of π_B, we show how to derive $D_{i+1} \vee B$.

When $D_{i+i} \in F$, the clause $D_{i+i} \vee B$ is the weakening of some clause in Γ. If D_{i+1} was derived using a resolution rule on some premises in π, then $D_{i+i} \vee B$ can be derived in the same way from the corresponding premises in π_B. The remaining case is when D_{i+1} is PR w.r.t. $F_i = F \cup \{D_1, \dots, D_i\}$. Let α be the assignment that witnesses it. This assignment only maps variables from the original formula F, so we extend it to $\alpha' = \alpha \cup \{b_1 \mapsto 0, \dots, b_m \mapsto 0\}$, and then use α' to witness that indeed $D_{i+1} \vee B$ is cost-PR w.r.t. $\Gamma_i = \Gamma \cup \{D_1 \vee B, \dots D_i \vee B\}$. For the cost condition in Definition 3.2, just observe that any extension of α' has cost 0. For the redundancy condition, observe that, by construction, $\Gamma_i \restriction_{\overline{D_{i+1}} \wedge \overline{B}} = F_i \restriction_{\overline{D_{i+1}}}$, $(F_i \cup \{D_{i+1}\}) \restriction_\alpha = (\Gamma_i \cup \{D_{i+1} \vee B\}) \restriction_{\alpha'}$, and $F_i \restriction_{\overline{D_{i+1}}} \vdash_1 (F_i \cup \{D_{i+1}\}) \restriction_\alpha$.

The last clause of π_B is B. Let α_{opt} be an optimal assignment of Γ. Since F is minimally unsatisfiable, $\text{cost}(\alpha_{opt}) = 1$. W.l.o.g. assume α_{opt} sets $b_m = 1$ and all $b_i = 0$ for $i < m$.

Now, for each $i < m$, the clause $E_i = \overline{b}_i \vee b_m$ is cost-PR w.r.t. $\pi_B \cup \{E_j : j < i\}$, using α_{opt} itself as the witnessing assignment: redundancy holds since α_{opt} satisfies every clause in π_B and all clauses E_j. The cost condition follow since $\text{cost}(\tau) \geq 1$ for any $\tau \supseteq \overline{E_i}$ and $\text{cost}(\tau \circ \alpha_{opt}) = \text{cost}(\alpha_{opt}) = 1$.

In the end we use $O(m)$ steps to derive b_m from B and $E_1, \dots, E_{m-1}$, and to derive in cost-PR calculus all the units $\overline{b}_1, \dots, \overline{b}_{m-1}$ via Lemma 4.1. ◀

Theorem 6.2 shows that the propositional refutations for the minimally unsatisfiable formulas in [9] translate immediately to certificates in the MaxSAT. In particular, as a corollary of Theorem 6.2, we have that cost-PR proves in polynomial size that

- the *Pigeonhole Principle* with $n+1$ pigeons and n holes [9, Theorem 4.3] and [19, Section 5],
- the *Bit-Pigeonhole Principle* [9, Theorem 4.4],
- the *Parity Principle* [9, Theorem 4.6],
- the *Tseitin Principle* on a connected graph [9, Theorem 4.10],

have all cost 1, since they are all minimally unsatisfiable. In MaxSAT resolution that would require exponentially long derivations.

6.2 Short proofs of the minimum cost of PHP_n^m

Let $m > n \geq 1$. The pigeonhole principle from m pigeons to n holes, with blocking variables, has the following formulation, that we call bPHP_n^m: the *totality* clauses $\bigvee_{j\in[n]} p_{i,j} \vee b_i$ for $i \in [m]$, and the *injectivity* clauses $\overline{p}_{i,j} \vee \overline{p}_{k,j} \vee b_{i,k,j}$ for $1 \leq i < k \leq m$ and $j \in [n]$. We use $b_{k,i,j}$ as an alias of the variable $b_{i,k,j}$, given that $i < k$.

▶ **Theorem 6.3.** cost-SR *proves* $\mathrm{cost}(\mathsf{bPHP}_n^m) = m - n$ *in polynomial size.*

This is the main result of the section. Before proving it we show two useful lemmas. The first lemma is used to "clean up" the set of clauses during a derivation. For each new step in a cost-SR calculus derivation the redundancy condition must be checked against an ever increasing set of clauses. It turns out that some already derived clauses can be completely ignored for the rest of the derivation under some technical conditions. This makes up for the lack of a deletion rule, that we do not have, and in the context of SAT seems to give more power to the systems [9].

▶ **Lemma 6.4.** *Let* Γ *and* Σ *be two sets of clauses. Any* cost-SR *derivation* $D_1 \dots, D_t$ *from* Γ *is also a valid derivation from* $\Gamma \cup \Sigma$ *if either of the two cases applies*

1. *Variables in* Σ *do not occur in* $\Gamma \cup \{D_1, \dots, D_t\}$.
2. *For every clause* $C \in \Sigma$ *there is a clause* $C' \in \Gamma$ *so that* $C' \subseteq C$.

Proof. The cost condition does not depend on the set of clauses, therefore we only need to check the validity of the redundancy condition. In the first case, the redundancy condition applies because the clauses of Σ are unaffected by the substitutions involved.

For the second case, consider the derivation of a clause D_i witnessed by σ_i. The clauses in $\Sigma\restriction_{\overline{D_i}}$ and $\Sigma\restriction_{\sigma_i}$ are subsumed by clauses in $\Gamma\restriction_{\overline{D_i}}$ and $\Gamma\restriction_{\sigma_i}$, respectively. Hence

$$(\Gamma \cup \{D_1, \dots, D_{i-1}\})\restriction_{\overline{D_i}} \vdash_1 (\Gamma \cup \Sigma \cup \{D_1, \dots, D_i\})\restriction_{\sigma_i}$$

which implies the validity of the redundancy condition. ◀

The second lemma is used as a general condition to enforce clauses to be cost-SR.

▶ **Lemma 6.5.** *Let* C *be a clause and* Γ *a set of clauses. If there exists a permutation* π *such that*

1. π *maps the set of blocking variables to itself,*
2. *the substitution* $\overline{C} \circ \pi$ *satisfies* C*, and* $\Gamma\restriction_{\overline{C}} \supseteq \Gamma\restriction_{\overline{C}\circ\pi}$,

then C *is* cost-SR *w.r.t.* Γ*. Notice that the second condition in item* (2) *is automatically satisfied if* π *is a symmetry of* Γ*, i.e.* $\Gamma = \Gamma\restriction_{\pi}$.

Proof. The cost condition follows from Item 1. The redundancy condition is immediate by Item 2 using as σ the substitution $\overline{C} \circ \pi$. ◀

Proof of Theorem 6.3. The proof is by induction. The goal is to reduce the formula to $m-1$ pigeons and $n-1$ holes. First we do some preprocessing: from bPHP_n^m we derive a slightly more structured formula F_n^m. Then we show how to derive F_{n-1}^{m-1} in a polynomial number of steps. The results follows because after n such derivations we obtain the formula F_0^{m-n} that contains the clauses $b_1, \ldots, b_{m-n}$. Moreover, we also derive $\overline{b_{m-n+1}}, \ldots, \overline{b_m}$ along the way.

We derive F_{n-1}^{m-1} from F_n^m using the rules of cost-SR calculus. We divide the argument into several steps, but first we show how to derive F_n^m from bPHP_n^m.

Preprocessing 1. *"Make b_i full-fledged extension variables".* [4] Turn all the variables b_i into full-fledged extension variables that satisfy $b_i \leftrightarrow \neg(p_{i,1} \vee \cdots \vee p_{i,n})$, by adding the clauses

$$\mathsf{Ext} = \{\overline{p_{i,j}} \vee \overline{b_i} : i \in [m], j \in [n]\}$$

one by one in cost-LPR.

We need to derive clause $D_j = \overline{p_{1,j}} \vee \overline{b_1}$ for every $j \in [n]$. Assume that we already got $D_1, \ldots, D_{j-1}$, we derive D_j as a cost-LPR clause w.r.t. $\mathsf{bPHP}_n^m \cup \{D_1, \ldots, D_{j-1}\}$. The witnessing assignment is $\sigma_j := \{p_{1,j} = 1, b_1 = 0\}$. Since $\overline{D_j} = \{p_{1,j} = 1, b_1 = 1\}$, the cost condition is satisfied. The redundancy condition follows from

$$\mathsf{bPHP}_n^m \restriction_{\overline{D_j}} \supseteq (\mathsf{bPHP}_n^m \cup \{D_1, \ldots, D_j\}) \restriction_{\sigma_j} \ .$$

Indeed, on clauses of bPHP_n^m that do not contain the variable b_j, the assignments $\overline{D_j}$ and σ_j behave identically, while all the clauses containing b_j are satisfied by σ_j. Repeat the previous argument to get all the clauses $\overline{p_{i,j}} \vee \overline{b_i}$. The current database of clauses is $\mathsf{bPHP}_n^m \cup \mathsf{Ext}$.

Preprocessing 2. *"Enforce injectivity".* Optimal assignments for bPHP_n^m can have unassigned pigeons or have collisions between pigeons. It is more convenient to avoid collisions and just focus on assignments that are partial matching. A moment's thought suffices to realize that such restriction does not change the optimal cost but simplifies the solution space. We enforce collisions to never occur by deriving all the unit clauses $\overline{b_{i,k,j}}$ by cost-PR.

These clauses can be derived in any particular order: to show that $\overline{b_{i,k,j}}$ is cost-SR w.r.t. Γ_0 and the previously derived $\overline{b_{i',k',j'}}$ we pick one of the two pigeons involved (say k) and use $\sigma = \{b_{i,k,j} = 0, b_k = 1, p_{k,1} = \cdots = p_{k,n} = 0\}$ as the witnessing assignment. The cost is not increased, and to check the redundancy condition observe that σ satisfies all the clauses that touches, so on the right side of the redundancy condition has a subset of $\mathsf{bPHP}_n^m \cup \mathsf{Ext}$ with no occurrences $b_{i,k,j}$, while the left side has the same set of clauses, but restricted with $b_{i,k,j} = 0$.

Now that we have all clauses $\overline{b_{i,k,j}}$ we resolve them with the corresponding clauses $\overline{p_{i,j}} \vee \overline{p_{k,j}} \vee b_{i,k,j}$ to get the set of clauses

$$\mathsf{Inj} = \{\overline{p_{i,j}} \vee \overline{p_{k,j}} : 1 \leq i < k \leq m \text{ and } j \in [n]\} \ ,$$

for all holes j and pair of pigeons i and k.

[4] This is true in general: if a MaxSAT instance contains a clause $C \vee b$ then it is possible to make the blocking variable b a full-fledged extension variable ($b \leftrightarrow \overline{C}$) by cost-LPR.

We do not need variables $b_{i,k,j}$ anymore. By one application of Lemma 6.4, from now on we can ignore all clauses $\overline{p_{i,j}} \vee \overline{p_{k,j}} \vee b_{i,k,j}$. By another application, we can also ignore the clauses $\overline{b_{i,k,j}}$. We will do induction on the current database of clauses.

For clarity we list all its clauses again.

$$\text{Formula } \mathsf{F}_n^m \quad \begin{array}{lll} \bigvee_{j\in[n]} p_{i,j} \vee b_i & \text{for } i \in [m] & (\textit{totality 1}), \\ \overline{p_{i,j}} \vee \overline{b_i} & \text{for } i \in [m] \text{ and } j \in [n] & (\textit{totality 2}), \\ \overline{p_{i,j}} \vee \overline{p_{k,j}} & \text{for } 1 \le i < k \le m \text{ and } j \in [n] & (\textit{injectivity}). \end{array}$$

The core idea of the induction is that if a pigeon flies to a hole, we can assume without loss of generality that it is pigeon m that flies into hole n.

Step 1. *"If some pigeon i flies, we can assume it is pigeon m who flies".* We want to derive, in this order, the set of clauses

$$\Delta_1 = \{\overline{b_m} \vee b_1, \overline{b_m} \vee b_2, \ldots, \overline{b_m} \vee b_{(m-1)}\}$$

from F_n^m, to claim that if some pigeon is mapped, then pigeon m is mapped too. For each $C_i = \overline{b_m} \vee b_i$ we apply Lemma 6.5 using as the witnessing permutation π_i, the permutation that swaps pigeons m and i.

Namely, $\pi_i(p_{m,j}) = p_{i,j}$, $\pi_i(p_{i,j}) = p_{m,j}$, $\pi_i(b_m) = b_i$, $\pi_i(b_i) = b_m$, and π_i is the identity on all other variables, therefore π_i satisfies the first requirement for the lemma. Likewise $\overline{C_i} \circ \pi_i \vDash C_i$, and we need to check that

$$(\mathsf{F}_n^m \cup \{C_1, \ldots, C_{(i-1)}\}) \restriction_{\overline{C_i}} \supseteq (\mathsf{F}_n^m \cup \{C_1, \ldots, C_{(i-1)}\}) \restriction_{\overline{C_i} \circ \pi_i} .$$

By symmetry $\mathsf{F}_n^m \restriction_{\overline{C_i}} = \mathsf{F}_n^m \restriction_{\overline{C_i} \circ \pi_i}$, and for $1 \le i' < i$, $C_{i'} \restriction_{\overline{C_i} \circ \pi_i} = 1$, hence the inclusion is true. The current database of clauses is $\Gamma_1 = \mathsf{F}_n^m \cup \Delta_1$.

Step 2. *"If pigeon m flies to some hole, we can assume it flies to hole n".* Using cost-SR inferences, we derive from Γ_1, in this order, the clauses

$$\Delta_2 = \{\overline{p_{m,1}} \vee p_{m,n}, \overline{p_{m,2}} \vee p_{m,n}, \ldots, \overline{p_{m,(n-1)}} \vee p_{m,n}\}$$

expressing that if pigeon m flies to some hole, this hole is the last one.

For each $C_j = \overline{p_{m,j}} \vee p_{m,n}$ we apply Lemma 6.5 with the witnessing permutation π_j swapping holes n and j.

Namely $\pi_j(p_{i,n}) = p_{i,j}$ and $\pi_j(p_{i,j}) = p_{i,n}$, and π_j is the identity on all other variables. By construction π_j satisfies the first requirement for the lemma, and likewise $\overline{C_j} \circ \pi_j \vDash C_j$, and, again, we need to check

$$(\Gamma_1 \cup \{C_1, \ldots, C_{(j-1)}\}) \restriction_{\overline{C_j}} \supseteq (\Gamma_1 \cup \{C_1, \ldots, C_{(j-1)}\}) \restriction_{\overline{C_j} \circ \pi_j} .$$

By symmetry $\Gamma_1 \restriction_{\overline{C_j}} = \Gamma_1 \restriction_{\overline{C_j} \circ \pi_j}$, and for $1 \le j' < j$, $C_{j'} \restriction_{\overline{C_j} \circ \pi_j} = 1$, hence the inclusion is true. The current database of clauses is $\Gamma_2 = \Gamma_1 \cup \Delta_2 = \mathsf{F}_n^m \cup \Delta_1 \cup \Delta_2$.

Step 3. *"Obtain $\overline{p_{k,n}}$ for every $1 \le k < m$ via resolution".* Resolve the clause $(p_{m,1} \vee p_{m,2} \vee \cdots \vee p_{m,n} \vee b_m)$ (totality 1) with $\overline{p_{m,n}} \vee \overline{p_{k,n}}$, the resulting clause with all clauses $\overline{p_{m,j}} \vee p_{m,n}$ from step 2, to get $b_m \vee p_{m,n} \vee \overline{p_{k,n}}$. Then resolve $b_m \vee p_{m,n} \vee \overline{p_{k,n}}$ again with the injectivity clause $\overline{p_{m,n}} \vee \overline{p_{k,n}}$, then the result with clause $\overline{b_m} \vee b_k$ (from step 1), and again this latter result with clause $\overline{b_k} \vee \overline{p_{k,n}}$ (totality 2). The final result is $\overline{p_{k,n}}$.

The clauses $\overline{p_{k,n}}$ subsume the clauses in Inj of the form $\overline{p_{m,n}} \vee \overline{p_{k,n}}$ and all the intermediate clauses from the previous resolution steps. Therefore we use Lemma 6.4 to be able to ignore the subsumed clauses.

The current database of clauses is Γ_3 is equal to

$$\mathsf{F}_n^m \cup \Delta_1 \cup \Delta_2 \cup \{\overline{p_{k,n}} : 1 \le k < m\} \setminus \{\overline{p_{m,n}} \vee \overline{p_{k,n}} : 1 \le k < m\}.$$

Step 4. *"Assign pigeon m to hole n: derive unit clauses $p_{m,n}$ and $\overline{b_m}$".* The goal is to enforce pigeon m to be mapped to hole n, by deriving the clause $p_{m,n}$ using the cost-PR rule. Then we get $\overline{b_m}$ immediately by resolving $p_{m,n}$ with $\overline{p_{m,n}} \vee \overline{b_m}$ (totality 2).

The unit clause $p_{m,n}$ is cost-PR w.r.t. Γ_3, using partial assignment $\sigma = \{p_{m,n} = 1, b_m = 0\}$ as witness.

Clearly σ satisfies the cost condition. To see that the redundancy condition holds as well, we need to show that $\Gamma_3|_{\overline{C}} \vdash_1 D|_\sigma$ for all D in Γ_3 that contain $\overline{p_{m,n}}$, but the only such clause that remains in Γ_3 is $\overline{p_{m,n}} \vee \overline{b_m}$, which is satisfied by σ. The current database of clauses is $\Gamma_4 = \Gamma_3 \cup \{p_{m,n}, \overline{b_m}\}$.

Step 5. *"Derive $\overline{p_{m,1}}, \dots, \overline{p_{m,(n-1)}}$ by* cost-SR*".* We can derive them in any order using as witnessing substitution of the cost-SR rule the assignment σ setting $p_{m,n} = 1$, $p_{m,1} = \cdots = p_{m,(n-1)} = 0$, and $b_m = 0$.

The cost condition is immediate, and the redundancy condition follows from the fact that $\Gamma_4 \restriction_\sigma \subseteq \Gamma_4$.

Step 6. *"Reduction to $m-1$ pigeons and $n-1$ holes".* First we derive by unit propagation all the the totality clauses of F_{n-1}^{m-1}. That is, we remove the hole n from the totality axioms of the pigeons $1, \dots, m-1$ in the current database. Now, the current database is F_{n-1}^{m-1}, the unit clauses $\overline{b_m}$, $p_{m,n}$, $\overline{p_{k,n}}$ for $k \neq m$ and $\overline{p_{m,j}}$ for $j \neq n$, and clauses that are subsumed by one of these unit clauses. Therefore by Lemma 6.4 we can ignore all the unit clauses and all the clauses subsumed by them. That is we can carry on the derivation using only F_{n-1}^{m-1}.

Thus steps (1)–(6) are repeated $n-1$ times, up to derive F_0^{m-n}.

The unit clauses derived in the whole process include

- $b_1, \dots, b_{(m-n)}$ (totality clauses in F_0^{m-n}).
- $\overline{b_{n+1}}, \dots, \overline{b_m}$ (derived at each step of the induction),
- $\overline{b_{i,k,j}}$ for all $i < k$ and j (derived at the preprocessing).

Therefore $\mathrm{cost}(\mathsf{bPHP}_n^m) = m - n$. ◀

7 Conclusions and open problems

We proposed a way to extend redundancy rules, originally introduced for SAT, into polynomially verifiable rules for MaxSAT. We defined sound and complete calculi based on those rules and we showed the strength of some of the calculi giving short derivations of notable principles and we showed the incompleteness of the weaker ones and width lower bounds for the stronger ones. We conclude this article with a list of open problems:

1. The cost constraint for the redundancy rules is very strict, for example compared to the rule `CPR` in [21]. Indeed, cost-PR enforces the check on the cost even on assignments falsifying the hard clauses of the formula. Is it possible to relax cost-PR without giving up on efficient verification as in [21]?
2. Does cost-SR simulate MaxSAT Resolution? That is, if we have a MaxSAT instance Γ with blocking variables and MaxSAT Resolution proves in size s that $\mathrm{cost}(\Gamma) = k$, is there a proof of $\mathrm{cost}(\Gamma) = k$ in cost-SR of size $\mathsf{poly}(s)$?
3. We proved a width lower bound for cost-SPR and an analogue of a width lower bound for cost-SR on formulas with optimal assignments far from each other in the Hamming distance. We reiterate the open problem of proving size lower bounds for cost-SPR and stronger systems.

References

1 Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artif. Intell.*, 196:77–105, 2013. doi:10.1016/J.ARTINT.2013.01.002.

2 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.

3 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)*, volume 14132, pages 1–22, 2023. doi:10.1007/978-3-031-38499-8_1.

4 Jeremias Berg and Matti Järvisalo. Unifying reasoning and core-guided search for maximum satisfiability. In *Proceedings of the 16th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 11468, pages 287–303, 2019. doi:10.1007/978-3-030-19570-0_19.

5 Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified dominance and symmetry breaking for combinatorial optimisation. *Journal of Artificial Intelligence Research*, 77:1539–1589, 2023. doi:10.1613/JAIR.1.14296.

6 Ilario Bonacina, Maria Luisa Bonet, and Massimo Lauria. MaxSAT Resolution with Inclusion Redundancy. In *Proceedings of the 27th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 305, pages 7:1–7:15, 2024. doi:10.4230/LIPIcs.SAT.2024.7.

7 Maria Luisa Bonet, Sam Buss, Alexey Ignatiev, João Marques-Silva, and António Morgado. MaxSAT resolution with the dual rail encoding. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI), the 30th innovative Applications of Artificial Intelligence (IAAI), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI)*, pages 6565–6572, 2018. doi:10.1609/AAAI.V32I1.12204.

8 Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8-9):606–618, 2007. doi:10.1016/J.ARTINT.2007.03.001.

9 Sam Buss and Neil Thapen. DRAT and propagation redundancy proofs without new variables. *Logical Methods in Computer Science*, 17(2), 2021. URL: https://lmcs.episciences.org/7400.

10 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979. doi:10.2307/2273702.

11 Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 225–239, 2011. doi:10.1007/978-3-642-23786-7_19.

12 Zhaohui Fu and Sharad Malik. On solving the partial MAX-SAT problem. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 252–265, 2006. doi:10.1007/11814948_25.

13 Michel X. Goemans and David P. Williamson. New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(4):656–666, 1994. doi:10.1137/S0895480192243516.

14 Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Trimming while checking clausal proofs. In *Proceedings of the conference on Formal Methods in Computer-Aided Design (FMCAD)*, pages 181–188, 2013. URL: https://ieeexplore.ieee.org/document/6679408/.

15 Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Verifying refutations with extended resolution. In *Proceedings of the 24th International Conference on Automated Deduction (CADE)*, volume 7898, pages 345–359, 2013. doi:10.1007/978-3-642-38574-2_24.

16 Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Expressing symmetry breaking in DRAT proofs. In *Proceedings of the 25th International Conference on Automated Deduction (CADE)*, volume 9195, pages 591–606, 2015. doi:10.1007/978-3-319-21401-6_40.

17 Marijn J. H. Heule and Armin Biere. What a difference a variable makes. In *Proceedings of the 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Held as Part of the European Joint Conferences on Theory and Practice of Software (ETAPS)*, volume 10806, pages 75–92, 2018. doi:10.1007/978-3-319-89963-3_5.

18 Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Short proofs without new variables. In *Proceedings of the 26th International Conference on Automated Deduction (CADE)*, volume 10395, pages 130–147, 2017. doi:10.1007/978-3-319-63046-5_9.

19 Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Strong extension-free proof systems. *Journal of Automated Reasoning*, 64(3):533–554, 2019. Extended version of [18]. doi:10.1007/s10817-019-09516-0.

20 Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere. Pruning through satisfaction. In *Proceedings of the Hardware and Software: Verification and Testing - 13th International Haifa Verification Conference (HVC)*, volume 10629, pages 179–194, 2017. doi:10.1007/978-3-319-70389-3_12.

21 Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo. Clause redundancy and preprocessing in maximum satisfiability. In *Proceedings of the 11th International Joint Conference on Automated Reasoning (IJCAR)*, volume 13385, pages 75–94, 2022. doi:10.1007/978-3-031-10769-6_6.

22 Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR)*, volume 7364, pages 355–370, 2012. doi:10.1007/978-3-642-31365-3_28.

23 Benjamin Kiesl, Adrián Rebola-Pardo, and Marijn J. H. Heule. Extended resolution simulates DRAT. In *Proceedings of the 9th International Joint Conference on Automated Reasoning (IJCAR), Held as Part of the Federated Logic Conference, FloC 2018*, volume 10900, pages 516–531, 2018. doi:10.1007/978-3-319-94205-6_34.

24 Leszek Aleksander Kołodziejczyk and Neil Thapen. The Strength of the Dominance Rule. In *Proceedings of the 27th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 305, pages 20:1–20:22, 2024. doi:10.4230/LIPIcs.SAT.2024.20.

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

26 Javier Larrosa and Federico Heras. Resolution in max-sat and its relation to local consistency in weighted csps. In *Proceedings of the 19h International Joint Conference on Artificial Intelligence (IJCAI)*, pages 193–198, 2005. URL: http://ijcai.org/Proceedings/05/Papers/0360.pdf.

27 António Morgado, Carmine Dodaro, and João Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 564–573, 2014. doi:10.1007/978-3-319-10428-7_41.

28 António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints An Int. J.*, 18(4):478–534, 2013. doi:10.1007/S10601-013-9146-2.

29 Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proceedings of the 28th Conference on Artificial Intelligence (AAAI)*, pages 2717–2723, 2014. doi:10.1609/AAAI.V28I1.9124.

30 Adrián Rebola-Pardo and Martin Suda. A theory of satisfiability-preserving proofs in SAT solving. In *Proceedings of the 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, volume 57, pages 583–603, 2018. doi:10.29007/TC7Q.

31 Paul Saikko, Jeremias Berg, and Matti Järvisalo. LMHS: a SAT-IP hybrid MaxSAT solver. In *Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 539–546, 2016. doi:10.1007/978-3-319-40970-2_34.