

Redundancy rules for MaxSAT*

Ilario Bonacina
Universitat Politècnica de Catalunya
Barcelona, Spain
ilario.bonacina@upc.edu

Maria Luisa Bonet
Universitat Politècnica de Catalunya
Barcelona, Spain
bonet@cs.upc.edu

Sam Buss
University of California
San Diego, USA
sbuss@ucsd.edu

Massimo Lauria
Sapienza Università di Roma
Rome, Italy
massimo.lauria@uniroma1.it

November 19, 2025

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_n^m falsifies at least $m - n$ clauses.

We conclude by discussing the integration of our rules with the MaxSAT resolution proof system, which is a commonly studied proof system for MaxSAT.

*An extended abstract of this article has appeared in proceedings of SAT'25 [BBBL25]. Ilario Bonacina and Maria Luisa Bonet have been supported by grants PID2022-138506NB-C21 and PID2022-138506NB-C22 (PROOFS BEYOND) funded by AEI. Massimo Lauria has been supported by the project PRIN 2022 “Logical Methods in Combinatorics” N. 2022BXH4R5 of the Italian Ministry of University and Research (MIUR).

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) [Kul99] and Resolution Asymmetric Tautology (RAT) [JHB12, HJW13b]. Other work on redundancy reasoning includes [HJW13a, HJW15, HKSB17, KRH18, HB18]; and, of particular relevance to the present paper, are the work of Heule, Kiesl, and Biere [HKB19], and the work of Buss and Thapen [BT21]. Redundancy inferences formalize “*without loss of generality*” reasoning [RS18] and can substantially strengthen resolution and, in some cases, the effectiveness of SAT solvers for hard problems such as the pigeonhole principle (PHP) [HKSB17]. 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 [BBI⁺18]. There are several approaches to MaxSAT solvers, including MaxSAT resolution [BLM07, LH05], core-guided MaxSAT [FM06, ABL13, MHL⁺13, MDM14, NB14], and maximum-hitting-set MaxSAT [BJM21, DB11, SBJ16]; the present paper discusses only MaxSAT resolution. The MaxSAT resolution proof system was first defined by Larrosa and Heras [LH05] and proved completed by Bonet, Levy and Manyà [BLM07].

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) [BT21]. 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 [IBJ22], building on [BJ19], introduced versions of redundancy inferences that work with MaxSAT. In contrast to the system CPR of [IBJ22], 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 [BGMN23]. 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 [BBN⁺23]. The propositional fragment of veriPB is as strong as Extended Resolution and DRAT, and plausibly even stronger [KT24]. 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 [BT21] 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 [BBL24]. 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 [BT21]). Likewise the possibility of introducing new variables in redundant clauses makes all such systems as powerful as extended resolution [Kul99]. With the goal of having systems that are simple and amenable to proof complexity analysis, in this paper we 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 [IBJ22]). 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. To deal with the incompleteness of cost-BC and cost-LPR, Section 7 describes proof systems augmenting MaxSAT resolution and the systems defined in Section 3. Section 8 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 $\bar{x}$. A *clause* is a finite disjunction of literals, *i.e.*, $C = \bigvee_i \ell_i$. The empty clause is $\perp$. 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 $\text{Var}(\Gamma)$ 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 $\sigma(\bar{x}) = \overline{\sigma(x)}$ 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 $\text{dom}(\sigma) = \sigma^{-1}(\{0, 1\})$, 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 $C|_\sigma = \bigvee_i \sigma(\ell_i)$, 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 $C|_\sigma$ is $\sigma(C)$. If $\sigma(C) = 1$ or $\sigma(C)$ is tautological we say that $\sigma \models C$, i.e., σ *satisfies* C .

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

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

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 $\Gamma|_{\bar{C}}$ 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 $\Gamma \models \Delta$. It is important to stress that the $\vdash_1$ relation is *efficiently checkable*.

Observation 2.1 ([BT21, Fact 1.3]). *Let σ be a substitution and Γ, Δ be CNF formulas, if $\Gamma \vdash_1 \Delta$, then $\Gamma|_\sigma \vdash_1 \Delta|_\sigma$.*

Proof. By assumption, for every clause $D \in \Delta$, $\Gamma|_{\bar{D}} \vdash_1 \perp$. Let $D = \bigvee_{i \in I} \ell_i$. Saying that $\Gamma|_{\bar{D}} \vdash_1 \perp$ is equivalent to saying $\Gamma \wedge \bigwedge_{i \in I} \bar{\ell}_i \vdash_1 \perp$. This in turn implies

$$(\Gamma \wedge \bigwedge_{i \in I} \bar{\ell}_i)|_\sigma \vdash_1 \perp, \quad (1)$$

since σ maps unit literals into unit literals (or 1, 0). By basic properties of substitutions we have $(\Gamma \wedge \bigwedge_{i \in I} \bar{\ell}_i)|_\sigma = \Gamma|_\sigma \wedge \bigwedge_{i \in I} (\bar{\ell}_i)|_\sigma$ and $(\bar{\ell}_i)|_\sigma = \overline{(\ell_i|_\sigma)}$, that is (1) implies

$$\Gamma|_\sigma|_{\overline{D|_\sigma}} \vdash_1 \perp$$

and $\Gamma|_\sigma \vdash_1 D|_\sigma$. □

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 $B \vee \bar{x}$ we can deduce $A \vee B$. A *resolution proof* from a set of clauses Γ is a sequence of clauses $D_1, D_2, \dots, D_t$ 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 $\Gamma \models C$.

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 $\bar{b}$, without affecting the cost. The variable b is usually called a *blocking variable*. This appears in [GW94], but it might have been used even earlier.

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

- $H' = H \wedge (C_1 \vee b_1) \wedge \dots \wedge (C_m \vee b_m)$,
- $S' = \bar{b}_1 \wedge \dots \wedge \bar{b}_m$,

and $b_1, \dots, b_m$ 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 .

¹An even more general version is *weighted* MaxSAT, where we would consider *weighted* set of clauses (F, w) , i.e., each clause $C \in F$ has an associated weight $w(C)$ where $w : F \rightarrow \mathbb{N} \cup \{\infty\}$. 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.

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 $\{b_1, \dots, b_m\}$. 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

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

and the goal is to find the value of $\text{cost}(\Gamma)$. Notice that, the notation $\text{cost}(\alpha)$ 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 [Kul99]. 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, [IBJ22]). A clause C is *redundant* w.r.t. a MaxSAT instance Γ when

$$\text{cost}(\Gamma) = \text{cost}(\Gamma \cup \{C\}) . \quad (2)$$

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. (2) 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. (2)) 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) [HKB19, BT21] to the context of MaxSAT. This is an alternative to the approach of [IBJ22]. This definition could also be seen as a very special case of veriPB [BGMN23] (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. $\Gamma|_{\overline{C}} \vdash_1 (\Gamma \cup \{C\})|_{\sigma}$ (*redundancy*)
2. for all total assignments $\tau \supseteq \overline{C}$, $\text{cost}(\tau \circ \sigma) \leq \text{cost}(\tau)$ (*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 assignment.

Cost subset propagation redundant (cost-SPR) if σ is a partial assignment 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.

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 3.7).

Indeed, Item 1 in Definition 3.2 and its variants correspond to the redundancy rules of proof systems SR, PR, SPR, and LPR from [BT21], adapted here to consider cost. Since LPR is the same as RAT (see [BT21, 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 [IBJ22, 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 3.8). In [IBJ22], 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.*

The very first notion of adding systematically redundant clauses is the one of *blocked clause* [Kul99]. Since this notion does not match exactly the scheme of Definition 3.2 Item 1, we define cost-BC directly as follows.

Definition 3.4 (cost-BC). A clause $C \vee \ell$ is a cost-BC w.r.t. a set of clauses Γ and a literal ℓ if for every clause $D \vee \overline{\ell} \in \Gamma$, $C \vee D$ is a tautology, and ℓ is not a blocking variable with positive polarity.

Proposition 3.5. *If $C \vee \ell$ is a cost-BC with respect to Γ and the literal ℓ , then $\text{cost}(\Gamma) = \text{cost}(\Gamma \cup \{C \vee \ell\})$.*

Proof. It is enough to show that $\text{cost}(\Gamma) \geq \text{cost}(\Gamma \cup \{C \vee \ell\})$. Let $\text{cost}(\Gamma) = k$. Consider an optimal total assignment α that satisfies Γ and sets to true exactly k blocking variables. If $\alpha \models C \vee \ell$ we already have that $\alpha \models \Gamma \cup \{C \vee \ell\}$ and $\text{cost}(\alpha) = k$. Otherwise, α extends $\overline{C \vee \ell}$. Consider the assignment β that is equal to α except in ℓ where the value is flipped.

By construction $\beta \models C \vee \ell$, moreover $\beta \models \Gamma$ since every clause which was satisfied by α and now might not be, has the form $D \vee \overline{\ell}$, and hence by the blocked clause definition is satisfied by $\overline{C \vee \ell}$ and therefore β . Moreover, $\text{cost}(\beta) \leq k$ since the flipped variable does not increase the cost. $\square$

Observation 3.6. *Any cost-BC clause $C \vee \ell$ w.r.t. to Γ and ℓ is also a cost-LPR clause w.r.t. Γ .*

Proof. Let σ be defined as $\overline{C} \cup \{\ell\}$. By definition, setting $\ell = 1$ does not increase the cost, so the cost-condition is satisfied, and furthermore it satisfies $C \vee \ell$. We need to show that $\Gamma|_{\overline{C \vee \ell}} \vdash_1 \Gamma|_{\sigma}$. All clauses not containing the variable of ℓ are in the sets on both sides. The one with literal ℓ are satisfied on the right side. The only clauses we need to check are the ones of the form $D \vee \ell$. In that case $D \vee C$ is a tautology, and the assignment $\overline{C}$, satisfies D . This means that these clause are all satisfied and do not occur in $\Gamma|_{\sigma}$. $\square$

Lemma 3.7. *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 $\alpha \models C$ we already have that $\alpha \models \Gamma \cup \{C\}$ 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 $\alpha \circ \sigma \models \Gamma \cup \{C\}$. By assumption,

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

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

Both Item 1 and Item 2 of Definiton 3.2 are stronger than what is actually needed for Lemma 3.7 to hold. Indeed, for Item 1, it would be enough that $\Gamma|_{\overline{C}} \vdash_1 (\Gamma \cup \{C\})|_{\sigma}$, and, for Item 2, it would be sufficient to check it for any $\tau \supseteq \overline{C}$ such that $\tau \models \Gamma$. Unfortunately, these latter versions of Item 1 and Item 2 are in general not polynomially checkable. Instead, our conditions are checkable in polynomial time.

Lemma 3.8. *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. The redundancy condition in Definiton 3.2 is polynomially checkable, since it is a unit propagation. To check the cost condition we need to compute

$$\max_{\tau \supseteq \overline{C}} (\text{cost}(\tau \circ \sigma) - \text{cost}(\tau)), \quad (3)$$

and decide whether this value is at most zero. The value of $\text{cost}(\tau)$ is by definition the number of variables or constants in the sequence $R = \langle b_1, b_2, \dots, b_m \rangle$ that evaluate to 1 after applying the assignment τ . The value of $\text{cost}(\tau \circ \sigma)$ is the same, but for the sequence of literals (or constants) $L = \langle \sigma(b_1), \sigma(b_2), \dots, \sigma(b_m) \rangle$. All the expressions in the sequences L and R are either constant values or literals, hence each of their evaluations is either independent from τ or is completely determined by the value assigned to the occurring variable. It is useful to highlight how much a single variable assignment increases the number of 1's in L and R .

For any sequence E of expressions that can either be 0, 1 or some literal, we use the following notation to indicate how many new 1's occur in E as a consequence of assigning some variable v to a value.

$$\begin{aligned} \text{Value}_E(v \mapsto 0) &= \text{number of occurrences of } \overline{v} \text{ in } E \\ \text{Value}_E(v \mapsto 1) &= \text{number of occurrences of } v \text{ in } E \end{aligned}$$

And with this notation we can write

$$\begin{aligned}
\text{cost}(\tau \circ \sigma) &= |\{i : \sigma(b_i) = 1\}| + |\{i : \sigma(b_i) \notin \{0, 1\} \text{ and } \tau \circ \sigma(b_i) = 1\}| \\
&= |\{i : \sigma(b_i) = 1\}| + \sum_v \text{Value}_L(v \mapsto \tau(v)) \\
\text{cost}(\tau) &= \sum_v \text{Value}_R(v \mapsto \tau(v))
\end{aligned}$$

Let us rewrite our objective function and separate the part that is fixed on all $\tau \supseteq \overline{C}$,

$$\begin{aligned}
\text{cost}(\tau \circ \sigma) - \text{cost}(\tau) &= \overbrace{|\{i : \sigma(b_i) = 1\}| + \sum_{v \in \text{dom}(\overline{C})} (\text{Value}_L(v \mapsto \tau(v)) - \text{Value}_R(v \mapsto \tau(v)))}^{\text{Fixed part } V} \\
&\quad + \sum_{v \notin \text{dom}(\overline{C})} (\text{Value}_L(v \mapsto \tau(v)) - \text{Value}_R(v \mapsto \tau(v))) .
\end{aligned}$$

It is crucial to observe that to achieve the maximum value of the objective (3), the variables not in $\text{dom}(\overline{C})$ can be set independently from each other. Hence

$$\max_{\tau \supseteq \overline{C}} (\text{cost}(\tau \circ \sigma) - \text{cost}(\tau)) = V + \sum_{v \notin \text{dom}(\overline{C})} \max_{k \in \{0, 1\}} (\text{Value}_L(v \mapsto k) - \text{Value}_R(v \mapsto k)) .$$

In the last equation V and the summands are easily computable in polynomial time, given C and σ . $\square$

Lemma 3.8 allows the definition of proof systems that extend resolution using cost redundancy rules in the sense of Cook-Reckhow [CR79].

Definition 3.9 (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 $D_1, D_2, \dots, D_t$ 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 $\Gamma \cup \{D_1, \dots, D_{i-1}\}$ with $\text{Var}(D_i) \subseteq \text{Var}(\Gamma)$.² 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 $\{b_{i_1}, \dots, b_{i_s}\}$ (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 $\{b_{i_1}, \dots, b_{i_s}\}$ together with the unit clauses $\{\bar{b}_j : j \notin \{i_1, \dots, i_s\}\}$ (see Theorem 4.2).

²We only consider the case where no new variables are added via cost-SR rules. To be coherent with [BT21], 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.

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, and when dealing with optimization problems, augment them with some criteria of cost preservation. Indeed, veriPB can argue within the proof itself whether some substitution is cost preserving. If veriPB can do that for the type of substitutions that respect the second condition in Definition 3.2, then it p-simulates cost-SR.

Since veriPB is a rich system, we only describe the 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 $\Gamma \cup \overline{C} \vdash (\Gamma \cup \{C\})|_{\sigma} \cup \{f|_{\sigma} \leq f\}$, where the symbol $\vdash$ means that there is a cutting planes proof that witnesses the implication.

The cutting plane proof in Item 2 is included in the veriPB proof to make it polynomially verifiable. This is not needed if such proof 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.

We claim that this fragment of veriPB indeed p-simulates cost-SR. Before proving that we need few additional notations and facts about veriPB. Every time we talk of clauses in veriPB we implicitly assume that they are represented via the standard encoding as linear inequalities. For instance, $x \vee y \vee \neg z$ is represented as $x + y + \bar{z} \geq 1$. Furthermore the system includes the linear inequalities $0 \leq x \leq 1$, $1 \leq x + \bar{x} \leq 1$ for each variable x in the formula (the *Boolean axioms*). The encoding uses formal representation of negative literal and variables, but the intended meaning of $\text{Var}(C)$ is the set of all the original variables, e.g., $\text{Var}(x \vee y \vee \bar{z}) = \text{Var}(x + y + \bar{z} \geq 1) = \{x, y, z\}$.

For a clause C encoded as $\ell \geq 1$, its formal negation $\overline{C}$ is the inequality $\ell \leq 0$. Using the Boolean axioms cutting planes easily proves that all the literals in C must be less or equal than 0.

Proposition 3.10. *Let F be a MaxSAT instance over n variables with blocking variables $b_1, b_2, \dots, b_k$. 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 $\text{poly}(nt)$ steps.*

Proof. We proceed by induction on t . If $t = 0$ then Γ is F itself plus the Boolean axioms, which is the initial configuration of the veriPB proof.

For $t > 0$ we can assume that the cost-SR derivation has obtained Γ up to step $t - 1$ and now at step t derives a clause C using either resolution, weakening or the redundancy rule, according to Definition 3.2. For the first two cases, observe that $\Gamma \cup \overline{C}$ is unsatisfiable and immediately implies contradiction by unit propagation, and therefore by a short cutting planes proof. Hence, in both cases rule 2 of veriPB proves C , with σ being the empty substitution.

The last case is when C is a cost preserving redundant clause with respect to Γ , witnessed by substitution σ . That is

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

We show that C can be derived by the redundancy inference rule of veriPB from Γ using the same substitution σ , that is

$$\Gamma \cup \overline{C} \vdash (\Gamma \cup \{C\})|_{\sigma} \cup \{f|_{\sigma} \leq f\}, \quad (4)$$

and the $\vdash$ is witnessed by a short cutting planes proof. One part of claim in (4), namely the implication $\Gamma \cup \overline{C} \vdash (\Gamma \cup \{C\})|_{\sigma}$, follows from the unit propagation $\Gamma \cup \overline{C} \vdash \Gamma|_{\overline{C}}$ and from the assumption $\Gamma|_{\overline{C}} \vdash_1 (\Gamma \cup \{C\})|_{\sigma}$.

The last part of claim (4) is the implication $\Gamma \cup \overline{C} \vdash \{f|_{\sigma} \leq f\}$. The goal inequality $f|_{\sigma} \leq f$ is indeed

$$\sum_{i=1}^k \sigma(b_i) - \sum_{i=1}^k b_i \leq 0.$$

Modulo the assumption $\overline{C}$ this is provably equivalent to

$$\left(\sum_{i=1}^k \sigma(b_i) - \sum_{i=1}^k b_i \right) \Big|_{\overline{C}} \leq 0. \quad (5)$$

The quantity $\max_{\tau \supseteq \overline{C}} (\text{cost}(\tau \circ \sigma) - \text{cost}(\tau))$ is the maximum achievable by the LHS of the inequality in (5) over all assignments on variables in $\text{Var}(\Gamma) \setminus \text{Var}(\overline{C})$. Using the Boolean axioms $x + \bar{x} = 1$ we remove negative literals from the LHS of (5) obtaining an equivalent representation of the form

$$c + \sum_{v \in \text{Var}(\Gamma) \setminus \text{Var}(\overline{C})} c_v \cdot v, \quad (6)$$

where c and c_v s are suitable integer constants. The maximum is achieved exactly when the contribution of each variable $v \notin \text{Var}(\overline{C})$ is maximized. Hence

$$\max_{\tau \supseteq \overline{C}} (\text{cost}(\tau \circ \sigma) - \text{cost}(\tau)) = c + \sum_{v \in \text{Var}(\Gamma) \setminus \text{Var}(\overline{C})} \max\{c_v, 0\}. \quad (7)$$

Simple applications of the Boolean axioms give cutting planes proof of $c_v \cdot v \leq \max\{c_v, 0\}$ for each $v \notin \text{Var}(\overline{C})$. The latter inequalities, summed together, prove that the LHS of (5) is at most (7). Observe that formula (7) has a constant value at most zero, by the cost condition assumption. Therefore we have produced a cutting planes proof of the inequality in (5). $\square$

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 $b_1, \dots, b_m$, of $\text{cost}(\Gamma) = k$, and suppose Γ contains the unit clauses $b_{i_1}, \dots, b_{i_k}$. Then cost-PR can prove $\bar{b}_j$ for each $j \notin \{i_1, \dots, i_k\}$ in $\mathcal{O}(km)$ steps.*

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

$$C_j = \bar{b}_{i_1} \vee \dots \vee \bar{b}_{i_k} \vee \bar{b}_j$$

with $j \notin \{i_1, \dots, i_k\}$ 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_j s is mapped to 1 under σ . The cost condition is true because for every $\tau \supseteq \overline{C_j}$, $\text{cost}(\tau) \geq k + 1$ and $\text{cost}(\tau \circ \sigma) = \text{cost}(\sigma) = k$.

To conclude, by resolution, derive $\bar{b}_j$ from C_j and the unit clauses $b_{i_1}, \dots, b_{i_k}$. $\square$

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 $\text{cost}(\Gamma) \geq k$. If there is a cost-SR proof of k distinct blocking variables $\{b_{i_1}, \dots, b_{i_k}\}$ and all the unit clauses $\bar{b}_j$ for $j \notin \{i_1, \dots, i_k\}$, then $\text{cost}(\Gamma) = k$.*

Proof. Let $b_1, \dots, b_m$ be the blocking variables of Γ . Let Γ' the set of clauses in Γ plus all the clauses derived in the proof of $\text{cost}(\Gamma) \geq k$. That is Γ' also contains k distinct unit clauses $b_{i_1}, \dots, b_{i_k}$, hence $\text{cost}(\Gamma') \geq k$. By Lemma 3.7, the cost is preserved along proof steps, therefore $\text{cost}(\Gamma) = \text{cost}(\Gamma') \geq k$. In the case where we have all the $\bar{b}_j$ s then $\text{cost}(\Gamma') = k$ and therefore $\text{cost}(\Gamma) = k$. $\square$

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 $\text{cost}(\Gamma) = k$. There is a cost-SPR derivation of the unit clauses $b_{i_1}, \dots, b_{i_k}$ for some distinct k blocking literals and all the $\bar{b}_j$ for $j \notin \{i_1, \dots, i_k\}$.*

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

$$(\Gamma \cup \Sigma') \upharpoonright_{\gamma} \vdash_1 (\Gamma \cup \Sigma' \cup \bar{\gamma}) \upharpoonright_{\alpha_{\text{opt}}}$$

holds because $\alpha_{\text{opt}} \models \Gamma \cup \Sigma' \cup \bar{\gamma}$ 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 $i > k$. $\square$

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 a partial assignment σ flips with respect to any $\tau \supseteq \bar{C}$ 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

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

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

Theorem 5.1. *Let Γ be a MaxSAT instance encoded with blocking variables, of $\text{cost}(\Gamma) = k$, and let A be the set of optimal total assignments for Γ , i.e., $\alpha \in A$ when $\alpha \models \Gamma$ and $\text{cost}(\alpha) = k$. 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 $\text{flip}(C, \sigma) \geq d$, where σ is the witnessing substitution for C .

Proof. Consider a cost-SR derivation from Γ as a sequence $\Gamma_0, \Gamma_1, \dots, \Gamma_s$ 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

$$\Gamma_{j-1} \upharpoonright_{\overline{C}} \vdash_1 (\Gamma_{j-1} \cup \{C\}) \upharpoonright_{\sigma} . \quad (8)$$

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,

$$\text{cost}(\tau \circ \sigma) \leq \text{cost}(\tau) = k . \quad (9)$$

By Observation 2.1, eq. (8) implies that

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

Since j was the first moment when $\mu(j) < |A|$ we have that $\tau \models \Gamma_{j-1}$ and therefore $\tau \circ \sigma \models \Gamma_{j-1} \cup \{C\}$. In particular, $\tau \circ \sigma \models \Gamma$. By eq. (9) 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 $\text{flip}(C, \sigma) \geq d$. $\square$

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 $x_0, x_1, \dots, x_n$, and variables $y_0, y_1, \dots, y_n$. 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. $\square$

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, $\text{flip}(C, \sigma) \geq d$. $\square$

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 demonstrating 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 $\bar{b}_j$ 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 [HKB17] (see also [BT21, Definition 1.16]). A PR calculus refutation of a CNF formula Γ is a sequence of clauses $D_1, \dots, D_t$ where $D_t = \perp$, 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

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

that is, Item 1 of Definition 3.2 for a σ which is a partial assignment. A PR refutation is a PR derivation of $\perp$. 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 $D_s = \perp$. 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 $D_{i+1} \in F$, the clause $D_{i+1} \vee B$ is the weakening of some clause in Γ . If D_{i+1} was derived using a resolution rule on some premises in π , then $D_{i+1} \vee B$ 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, $\Gamma_i \upharpoonright_{\overline{D_{i+1}} \wedge \overline{B}} = F_i \upharpoonright_{\overline{D_{i+1}}}$, $(F_i \cup \{D_{i+1}\}) \upharpoonright_{\alpha} = (\Gamma_i \cup \{D_{i+1} \vee B\}) \upharpoonright_{\alpha'}$, and $F_i \upharpoonright_{\overline{D_{i+1}}} \vdash_1 (F_i \cup \{D_{i+1}\}) \upharpoonright_{\alpha}$.

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 $E_i = \bar{b}_i \vee b_m$ 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 follows 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 $\bar{b}_1, \dots, \bar{b}_{m-1}$ via Lemma 4.1. $\square$

Theorem 6.2 shows that the propositional refutations for the minimally unsatisfiable formulas in [BT21] 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 [BT21, Theorem 4.3] and [HKB19, Section 5],
- the *Bit-Pigeonhole Principle* [BT21, Theorem 4.4],
- the *Parity Principle* [BT21, Theorem 4.6],
- the *Tseitin Principle* on a connected graph [BT21, 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 $\bar{p}_{i,j} \vee \bar{p}_{k,j} \vee b_{i,k,j}$ 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 $\text{cost}(\text{bPHP}_n^m) = m - n$ 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 [BT21].

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 $\Sigma \upharpoonright_{\overline{D_i}}$ and $\Sigma \upharpoonright_{\sigma_i}$ are subsumed by clauses in $\Gamma \upharpoonright_{\overline{D_i}}$ and $\Gamma \upharpoonright_{\sigma_i}$, respectively. Hence

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

which implies the validity of the redundancy condition. $\square$

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 $\Gamma \upharpoonright_{\overline{C}} \supseteq \Gamma \upharpoonright_{\overline{C} \circ \pi}$,

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. $\Gamma = \Gamma \upharpoonright_{\pi}$.

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

To satisfy Γ with a clause $C \vee b$, where b is the corresponding blocking variable, b must be true whenever C is false. To minimize cost, though, it makes sense to set b to false whenever C is satisfied. Namely, to have $b \leftrightarrow \overline{C}$. This does not follow logically from Γ , but can be derived in cost-LPR.

Lemma 6.6. *Let Γ contain a clause $\{C \vee b\}$, so that b is a blocking variables and that clause is its unique occurrence in Γ . It is possible to introduce all the clauses of the form $\overline{\ell} \vee \overline{b}$, for every literal ℓ in C , using the cost-LPR rule. That is, we can turn b into a full-fledged extension variable such that $b \leftrightarrow \overline{C}$.*

Proof. Let Γ be as in the statement, let $C = \ell_1 \vee \dots \vee \ell_t$. We need to show we can derive clause $D_i = \overline{\ell_i} \vee \overline{b}$ for every $1 \leq i \leq t$. Assume that we derived $D_1, \dots, D_{i-1}$, we derive D_i in cost-LPR using witnessing assignment $\sigma_i := \{\ell_i \mapsto 1, b \mapsto 0\}$. Since $\overline{D_i} = \{\ell_i \mapsto 1, b \mapsto 1\}$, the cost condition (item 2 of Definition 3.2) is satisfied. To show the redundancy condition (item 1 of Definition 3.2) we prove something stronger, that is

$$\Gamma \upharpoonright_{\overline{D_i}} \supseteq (\Gamma \cup \{D_1, \dots, D_i\}) \upharpoonright_{\sigma_i}.$$

Indeed, on clauses of Γ that do not contain the variable b , the assignments $\overline{D_i}$ and σ_i behave identically, while all the clauses containing b are satisfied by σ_i since those clauses are $\{C \vee b\}$ and $D_1, \dots, D_i$. $\square$

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 $b_1, \dots, b_{m-n}$. Moreover, we also derive $\overline{b_{m-n+1}}, \dots, \overline{b_m}$ 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*”. Using Lemma 6.6, turn all the variables b_i into full-fledged extension variables that satisfy $b_i \leftrightarrow \neg(p_{i,1} \vee \dots \vee p_{i,n})$, by adding the clauses

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

one by one in cost-LPR. The current database of clauses is $\text{bPHP}_n^m \cup \text{Ext}$.

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 $\sigma = \{b_{i,k,j} = 0, b_k = 1, p_{k,1} = \dots = p_{k,n} = 0\}$ 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 $\text{bPHP}_n^m \cup \text{Ext}$ 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

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

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

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.

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

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

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

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 $\overline{C_i} \circ \pi_i \models C_i$, and we need to check that

$$(F_n^m \cup \{C_1, \dots, C_{(i-1)}\}) \upharpoonright_{\overline{C_i}} \supseteq (F_n^m \cup \{C_1, \dots, C_{(i-1)}\}) \upharpoonright_{\overline{C_i} \circ \pi_i}.$$

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

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

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

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 $\overline{C_j} \circ \pi_j \models C_j$, and, again, we need to check

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

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

Step 3. “Obtain $\overline{p_{k,n}}$ for every $1 \leq k < m$ via resolution”. Resolve the clause $(p_{m,1} \vee p_{m,2} \vee \dots \vee p_{m,n} \vee b_m)$ (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

$$F_n^m \cup \Delta_1 \cup \Delta_2 \cup \{\overline{p_{k,n}} : 1 \leq k < m\} \setminus \{\overline{p_{m,n}} \vee \overline{p_{k,n}} : 1 \leq k < m\}.$$

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 $p_{m,n} = 1$, $p_{m,1} = \dots = p_{m,(n-1)} = 0$, and $b_m = 0$.

The cost condition is immediate, and the redundancy condition follows from the fact that $\Gamma_4|_{\sigma} \subseteq \Gamma_4$.

Step 6. “Reduction to $m - 1$ pigeons and $n - 1$ holes”. First we derive by unit propagation all 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 $\text{cost}(\text{bPHP}_n^m) = m - n$. □

7 MaxSAT Resolution + Redundancy

We conclude this article showing a natural strengthening of MaxSAT resolution, based on the redundancy rules and calculi from Section 3.

In *MaxSAT resolution* [BLM07] a proof starts from a set H_0 of hard clauses and a multiset of soft clauses S_0 . At each step i , new sets H_i, S_i are derived according to some rules that keep invariant the minimum number of clauses in S_i that must be falsified for any truth assignment satisfying H_0 .

If at some step there are k copies of $\perp$ among the soft clauses, it means that at least k clauses from S_0 will be falsified by any assignment satisfying H_0 . Resolution rules are not good enough, for example if $A \vee x$ and $B \vee \overline{x}$ are in S_i , just setting $S_{i+1} := S_i \cup \{A \vee B\}$ does not guarantee cost invariance. There are several equivalent ways to describe the rules of MaxSAT resolution. We use those from [AL23], first used in the context of MaxSAT

in [BL20]. From Observation 2.3 we can assume MaxSAT instances to be of the form $H_0 \wedge S_0$, where H_0 is a set of hard clauses and the multiset of soft clauses S_0 equals the set of *unit* clauses $\{\bar{b}_1, \dots, \bar{b}_m\}$.

A MaxSAT resolution derivation is then a sequence of pairs (H_i, S_i) such that

- (a) $S_i := S_{i-1}$ and $H_i := H_{i-1} \cup \{C\}$, where C is deduced by resolution from H_{i-1} ;
- (b) $S_i := S_{i-1} \cup \{C\}$ where $C \in H_{i-1}$ and $H_i := H_{i-1}$;
- (c) $S_i := S_{i-1} \setminus \{C\} \cup \{C \vee x, C \vee \bar{x}\}$ where $C \in S_{i-1}$ and $H_i := H_{i-1}$;
- (d) $S_i := S_{i-1} \setminus \{C \vee x, C \vee \bar{x}\} \cup \{C\}$ where $\{C \vee x, C \vee \bar{x}\} \subseteq S_{i-1}$ and $H_i := H_{i-1}$.

If the derivation obtains k copies of $\perp$ in some S_i , then this is a certificate of the fact that the original instance has cost at least k . This system is complete as well, in the sense that from an instance of minimum cost k there is always a derivation of k copies of $\perp$. Soundness and completeness of MaxSAT resolution are proved in [BLM07].

As a deduction system on the soft clauses, MaxSAT resolution is simulated by resolution [BLM07, Theorem 17]. Therefore, to build inference systems stronger than MaxSAT resolution, we introduce redundancy rules (Definiton 3.2).

Definition 7.1 (MaxSAT resolution + cost-SR). Let $F = H \wedge S$ be a MaxSAT instance with hard clauses $H = \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ and soft clauses $S = \{\bar{b}_1, \dots, \bar{b}_m\}$. A derivation in the system *MaxSAT resolution* + cost-SR is a sequence of pairs of multisets of clauses (H_i, S_i) , where H_i and S_i are obtained from H_{i-1} and S_{i-1} either using one of the MaxSAT resolution rules (a)–(d) above or

- (a') $S_i := S_{i-1}$ and $H_i := H_{i-1} \cup \{C\}$, where C is cost-SR w.r.t. H_{i-1} .

Assuming H_0 to be satisfiable, we say that the system proves that the instance has cost at least k if there is some step i where S_i contains k copies of $\perp$. Analogous systems can be defined strengthening MaxSAT resolution with any of cost-PR, cost-SPR, cost-LPR, cost-BC.

Definiton 7.1 extends the systems from Section 3, but those systems formally prove that cost is at least k by deriving k hard unit clauses $b_{i_1}, b_{i_2}, \dots, b_{i_k}$, while here we want to prove k copies of $\perp$ instead. This is not an issue since we can copy these $b_{i_1}, b_{i_2}, \dots, b_{i_k}$ in the database of soft clauses, using rule (b), and resolve them with the corresponding soft clauses $\bar{b}_{i_1}, \bar{b}_{i_2}, \dots, \bar{b}_{i_k}$, using rule (d), to get k empty clauses $\perp$. That is, the extension MaxSAT resolution + P p-simulates the original calculus P and therefore is also complete when P is either cost-SPR, cost-PR, or cost-SR (Theorem 4.3). The completeness of MaxSAT resolution + cost-BC and cost-LPR follows instead from the completeness of MaxSAT resolution itself [BLM07].

Theorem 7.2 (completeness). *Let P denote any of cost-SR, cost-PR, cost-SPR, cost-LPR, cost-BC. The system MaxSAT resolution + P is complete.*

It is easier to prove the soundness of MaxSAT resolution + cost-SR if we first put the derivations in a convenient normal form.

Proposition 7.3 (normal form). *In a MaxSAT resolution + cost-SR derivation we can always assume, at the cost of a polynomial increase in size of the derivations, that the rules (a) and (a') are applied before the rules (b)–(d).*

Proof. Having more hard clauses in the database never prevents applications of the rules (b)–(d), and furthermore rules (a) and (a') do not depend in any way on the soft clauses in it. Therefore we can assume all the hard clauses to be derived before any manipulation of the soft clauses. $\square$

Theorem 7.4 (soundness). *The system MaxSAT resolution + cost-SR is sound, i.e., if MaxSAT resolution + cost-SR can derive $\perp$ as a soft clause with multiplicity k from a formula $F = H_0 \wedge S_0$ with satisfiable hard clauses $H_0 = \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ and soft clauses $S_0 = \{\bar{b}_1, \dots, \bar{b}_m\}$, then the number of clauses that need to be falsified in S_0 is at least k , or equivalently $\text{cost}(H_0) \geq k$.*

Proof. Consider a MaxSAT instance resolution + cost-SR derivation $(H_0, S_0), \dots, (H_t, S_t)$ where S_t contains k copies of $\perp$.

Given a total assignment α , let $c(\alpha, S_i)$ be the number (counted with multiplicity) of clauses in S_i falsified by α . That is, in particular, for any total assignment α , $c(\alpha, S_t) \geq k$. By induction on i , we will show that

$$\min_{\alpha : \alpha \models H_i} c(\alpha, S_i) = \text{cost}(H_0) . \quad (10)$$

Because of Proposition 7.3 we can assume that up to some step t_0 , included, the proof only uses rules (a) or (a'), and from step $t_0 + 1$ to t , the proof only uses rules (b)–(d).

By the soundness of cost-SR (Lemma 3.7) we have that when $i \leq t_0$, H_i is satisfiable and $\text{cost}(H_i) = \text{cost}(H_0)$. Furthermore $S_i = S_0$. Hence

$$\min_{\alpha : \alpha \models H_i} c(\alpha, S_i) = \min_{\alpha : \alpha \models H_i} c(\alpha, S_0) = \text{cost}(H_i) = \text{cost}(H_0) .$$

for $i \leq t_0$. For $i > t_0$ we applied one of rules (b)–(d) above, and

$$\min_{\alpha : \alpha \models H_i} c(\alpha, S_i) = \min_{\alpha : \alpha \models H_{i-1}} c(\alpha, S_{i-1}) \stackrel{IH}{=} \text{cost}(H_0) ,$$

where the first equality is the soundness of MaxSAT resolution. $\square$

8 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 [IBJ22]. 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 [IBJ22]?
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 $\text{cost}(\Gamma) = k$, is there a proof of $\text{cost}(\Gamma) = k$ in cost-SR of size $\text{poly}(s)$? Does cost-SR simulate MaxSAT resolution + cost-SR?
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.

Acknowledgments

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* and the *Meta-Complexity* programs (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.

References

- [ABL13] 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.
- [AL23] Albert Atserias and Massimo Lauria. Circular (yet sound) proofs in propositional logic. *ACM Trans. Comput. Log.*, 24(3):20:1–20:26, 2023. doi:10.1145/3579997.
- [BBBL25] Ilario Bonacina, Maria Luisa Bonet, Sam Buss, and Massimo Lauria. Redundancy Rules for MaxSAT. In *28th International Conference on Theory and Applications of Satisfiability Testing (SAT 2025)*, volume 341, pages 7:1–7:17, 2025. doi:10.4230/LIPIcs.SAT.2025.7.
- [BBI⁺18] 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.

- [BBL24] 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.
- [BBN⁺23] 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.
- [BGMN23] 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.
- [BJ19] 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.
- [BJM21] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maximum satisfiability. 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.
- [BL20] Maria Luisa Bonet and Jordi Levy. Equivalence between systems stronger than resolution. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing – SAT 2020*, pages 166–181, Cham, 2020. Springer International Publishing.
- [BLM07] 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.
- [BT21] 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>.
- [CR79] 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.
- [DB11] 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.

- [FM06] 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.
- [GW94] 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.
- [HB18] 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.
- [HJW13a] 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/>.
- [HJW13b] 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.
- [HJW15] 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.
- [HKB17] 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.
- [HKB19] 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 [HKB17]. doi:10.1007/s10817-019-09516-0.
- [HKS17] 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.
- [IBJ22] 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.
- [JHB12] 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.
- [KRH18] 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.
- [KT24] 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.
- [Kul99] Oliver Kullmann. On a generalization of extended resolution. *Discrete Applied Mathematics*, 96-97:149–176, 1999. doi:10.1016/S0166-218X(99)00037-2.
- [LH05] Javier Larrosa and Federico Heras. Resolution in max-sat and its relation to local consistency in weighted csp. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 193–198, 2005. URL: <http://ijcai.org/Proceedings/05/Papers/0360.pdf>.
- [MDM14] 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.
- [MHL⁺13] 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.
- [NB14] 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.
- [RS18] 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.

- [SBJ16] 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.