Electronic Colloquium on Computational Complexity, Report No. 45 (2024)

Redundancy rules for MaxSAT

Ilario Bonacina
UPC Universitat Politècnica de Catalunya, Spain

Maria Luisa Bonet
UPC Universitat Politècnica de Catalunya, Spain

Sam Buss
University of California, San Diego

Massimo Lauria
Sapienza Università di Roma, Italy

Abstract

The concept of redundancy in SAT lead to more expressive and powerful proof search techniques, e.g. able to express various inprocessing techniques, and to interesting hierarchies of proof systems [Heule *et.al*'20, Buss-Thapen'19].

We propose a general way to integrate redundancy rules in MaxSAT, that is we define MaxSAT variants of proof systems such as SPR, PR, SR, and others. The main difference compared to the recent alternative approach in [Ihalainen *et.al*'22] is that our redundancy rules are polynomially checkable. We discuss the strength of the systems introduced 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; Theory of computation → Complexity theory and logic

Keywords and phrases MaxSAT, Redundancy Rules, Pigeonhole Principles

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) [19] and RAT inferences [17, 11]. Other work on redundancy reasoning includes [10, 12, 15, 18, 9]; and, of particular relevance to the present paper, are the work of Heule, Kiesl, and Biere [14], and the work of Buss and Thapen [6]. Redundancy inference formalize "*without loss of generality*" reasoning [21] and can substantially strengthen resolution and, in some cases, the effectiveness of SAT solvers for hard problems such as the pigeonhole principle (PHP) [15]. 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 [3]. There are several approaches to MaxSAT solvers, including MaxSAT resolution, core-guided MaxSAT, and maximum-hitting-set MaxSAT; the present paper discusses only MaxSAT resolution. The MaxSAT resolution proof system was first defined by [20] and proved completed by [5].

Recent work of Ihalainen, Berg and Järvisalo [16], building on [2], introduced new versions of redundancy inferences that work with MaxSAT. These redundancy inferences introduce only clauses which are cost-preserving. Their new inferences included CPR, CLPR and CSPR, which are cost-preserving versions of propagation redundancy (PR), subset propagation redundancy (SPR), and literal propagation redundancy (LPR).

ISSN 1433-8092

The present paper provides an alternative approach to defining cost-perserving redundancy inferences. We define inferences 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) [6]. In contrast to the system CPR of [16], all of our "cost-" inferences are polynomial-checkable for validity, and thus all give traditional Cook-Reckhow proof systems for MaxSAT. 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 (Theorem 4.4). 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^m_n for arbitrary numbers $m > n$ of pigeons and holes (Theorem 5.3).

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 also used by [16]). Section 3 introduces the redundancy rules for MaxSAT, proves their basic propertie, and defines calculi based on those rules. Section 4 shows their soundness, and their completeness or (for cost-BC and cost-LPR) their incompleteness. Section 5 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 6 describes proof systems augmenting MaxSAT resolution and the systems defined in Section 3. Section 7 gives some concluding remarks.

2 Preliminaries

For a natural number n, let $[n]$ be the set $\{1, \ldots, n\}$. 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 $\perp$. 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 $\mathrm{Var}(\Gamma)$ 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 $C|_\sigma = \bigvee_i \sigma(\ell_i)$, simplified using the usual logic rules, *i.e.*, $D \vee 0 = D$, $D \vee 1 = 1$, and $D \vee \ell \vee \ell = D \vee \ell$. 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 $\Gamma|_\sigma$, 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 ($\sigma \vDash \Gamma$) if for every $C \in \Gamma$, $\sigma \vDash C$, *i.e.*, $\Gamma|_\sigma = \emptyset$. 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 assign ℓ to 1 and leaves all other variables unassigned. Hence we use notations like $\Gamma|_\ell$. 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 $\Gamma|_{\overline{C}}$.

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, and continue the process using restricted the formula $\Gamma|_\ell$. 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 $\Gamma|_{\overline{C}}$ 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** ([6, Fact 1.3]). *Let σ be a substitution and Γ, Δ be CNF formulas, if $\Gamma \vdash_1 \Delta$, then $\Gamma|_\sigma \vdash_1 \Delta|_\sigma$.* (For a proof of this fact see Appendix A.)

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 (see Section 6).

[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.

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 [8], but it might have been used even earlier.

▶ **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

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

and the goal is to find the value of cost(Γ). Notice that, the notation cost(α) 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 [19]. 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, [16])**.** 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) [14, 6] to the context of MaxSAT. This is an alternative to the approach of [16].

▶ **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 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).

Item 1 in Definition 3.2 is the same as the one in [6]. Indeed this notion and the other three special cases correspond to the rules of proof systems SR, PR, SPR, and LPR from [6], adapted here to consider cost. Since LPR is the same as RAT (see [6, 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 [16, 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. This latter condition is more expressive but unlikely to be polynomially checkable, while the condition in cost-PR is polynomially checkable (see Lemma 3.5). In [16], 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 $\text{cost}(\Gamma \cup \{C\}) \leq k$, is then enough to show that $\alpha \circ \sigma \vDash \Gamma \cup \{C\}$. By assumption,

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

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

[2] The definition of *blocked clause* is written to match the previous definitions. For the equivalence with the usual notion of blocked clause see Appendix B. In this case, the redundancy condition is always satisfied.

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

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

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

and decide whether this value is at most zero. The value of $\mathrm{cost}(\tau)$ 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 $\mathrm{cost}(\tau \circ \sigma)$ 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 expresions 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.

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

And with this notation we can write

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

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

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

It is crucial to observe that to achieve the maximum value of the objective (2), the variables not in $\mathrm{dom}(\overline{C})$ can be set independently from each other. Hence

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

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

Lemma 3.5 allows the definition of proof systems that extend resolution using cost redundancy rules in the sense of Cook-Reckhow [7].

▶ **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 $D_1, D_2, \dots, D_t$ 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 $\mathrm{Var}(D_i) \subseteq \mathrm{Var}(\Gamma)$.[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 $\mathrm{cost}(\Gamma) \geq s$, 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 $\mathrm{cost}(\Gamma) = s$, we can accomplish this deriving s distinct unit clauses of the form $\{b_{i_1}, \dots, b_{i_s}\}$ together with the unit clauses $\{\overline{b}_j : j \notin \{i_1, \dots, i_s\}\}$ (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.

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

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

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

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_js is mapped to 1 under σ. The cost condition is true because for every $\tau \supseteq \overline{C_j}$, $\mathrm{cost}(\tau) \geq k + 1$ and $\mathrm{cost}(\tau \circ \sigma) = \mathrm{cost}(\sigma) = k$.

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

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

[3] We only consider the case where no new variables are added via cost-SR rules. To be coherent with [6], 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.

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 $\mathrm{cost}(\Gamma) \geq k$. That is Γ' also contains contains k distinct unit clauses $b_{i_1}, \dots, b_{i_k}$, hence $\mathrm{cost}(\Gamma') \geq k$. By Lemma 3.4, the cost is preserved along proof steps, therefore $\mathrm{cost}(\Gamma) = \mathrm{cost}(\Gamma') \geq k$. In the case where we have all the $\overline{b}_j$s then $\mathrm{cost}(\Gamma') = k$ and therefore $\mathrm{cost}(\Gamma) = k$. ◀

As an immediate consequence of Theorem 4.2, also all the cost-PR, cost-SPR, cost-LPR calculi are sound.

Even if there are multiple optimal solutions, we show that in cost-SPR calculus is sufficient to identify a specific set of blocking variables and prove the optimal lower bound.

▶ **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 $b_{i_1}, \dots, b_{i_k}$ for some distinct k blocking literals and all the $\overline{b}_j$ for $j \notin \{i_1, \dots, i_k\}$.*

Proof. Let $b_1, \dots, b_m$ be the blocking variables of Γ. Take α_{opt} a 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 $b_1, \dots, b_k$ to 1 and the remaining b_js to 0.

Given any assignment γ, let $\bar{\gamma}$ 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 and sets $\gamma(b_i) = 0$ for some $1 \leq i \leq k$. We want to derive Σ from Γ, essentially forbidding any satisfying assignment except for α_{opt}.

We show that 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. $\Gamma \cup \Sigma'$. The redundancy condition

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

holds because $\alpha_{\mathrm{opt}} \vDash \Gamma \cup \Sigma' \cup \bar{\gamma}$ and the RHS is just true. Indeed by assumption $\alpha_{\mathrm{opt}} \vDash \Gamma$, and all the clauses in Σ contain some variable from $b_1, \dots, b_k$ appearing positively that by construction α_{opt} sets to 1. The cost condition holds by optimality of α_{opt}. We show that for every $i \in [k]$,

$$(\Gamma \cup \Sigma)|_{b_i=0} \tag{3}$$

is unsatisfiable. To see this, assume by contradiction that for some b_i with $1 \leq i \leq k$ the corresponding CNF formula (3) was satisfiable by a total assignment β. Observe that b_i is not assigned by b. Then $\gamma = \beta \cup \{b_i = 0\}$ would satisfy $\Gamma \cup \Sigma$, and therefore $\bar{\gamma} \in \Sigma$. That is we would have γ satisfying $\bar{\gamma}$ which is not possible.

Therefore, for each i, the CNF formula in (3) being unsatisfiable implies $\Gamma \cup \Sigma \vDash b_i$, and by the completeness of resolution then from $\Gamma \cup \Sigma$ we can derive b_i for each $i \in [k]$. By Lemma 4.1, we can also derive all the $\overline{b}_j$s for $j \notin [k]$. ◀

Theorem 4.3 gives the completeness of the cost-SPR calculus (and the stronger systems cost-PR and cost-SR calculi as well), but does not give the completeness of cost-LPR calculus, which, indeed, is not complete.

▶ **Theorem 4.4.** *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* $\mathrm{cost}(\alpha) = k$*. We claim that if A is such that*

1. *all pairs of assignments in A have hamming distance at least 2, and*

2. *for every blocking variable b there are $\alpha, \beta \in A$ s.t. $\alpha(b) = 0$ and $\beta(b) = 1$,*

then cost-LPR *cannot derive any blocking literal b.*
An example is $\Gamma = \{x \vee y \vee b_1, \overline{x} \vee b_2, \overline{y} \vee b_3\}$ that has cost 1 and the corresponding set of optimal assignments for the variables x, y, b_1, b_2, b_3 is $A = \{10001, 01010, 00100\}$.

Proof. Consider a cost-LPR 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-LPR 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 $\{b_i \mapsto 0\}$. We show that $\mu(j) = |A|$ for the whole proof, therefore no clauses b_i can be ever derived.

Suppose, towards a contradiction, that μ drops below $|A|$ for the first time at step j. The clause C introduced at that moment must be cost-LPR w.r.t. Γ_{j-1}, because the resolution steps do not change the set of optimal assignments. Since $\mu(j)$ dropped below $|A|$, clause C must be incompatible with some $\alpha \in A$, that is $\alpha \subseteq \overline{C}$.

By definition of cost-LPR we have that there exists some assignment σ such that

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

and

$$\mathrm{cost}(\alpha \circ \sigma) \leq \mathrm{cost}(\alpha) = k\ . \tag{5}$$

By Observation 2.1, eq. (4) implies that

$$\Gamma_{j-1}|_\alpha \vdash_1 (\Gamma_{j-1} \cup \{C\})|_{\alpha \circ \sigma}\ .$$

Since j was the first moment when $\mu(j) < |A|$ we have that $\alpha \vDash \Gamma_{j-1}$ and therefore $\alpha \circ \sigma \vDash \Gamma_{j-1} \cup \{C\}$. In particular, $\alpha \circ \sigma \vDash \Gamma$. By eq. (5) then it must be $\alpha \circ \sigma \in A$. But then, by assumption α and $\alpha \circ \sigma$ have hamming distance at least 2, which is incompatible with the cost-LPR condition that σ must flip at most one variable in $\overline{C}$. ◀

The previous result also implies that cost-BC is also incomplete. In Section 6 we show a way to generalize Definition 3.6 to have calculi complete also for the cost-LPR and cost-BC redundancy rules.

5 Short proofs using redundancy rules

We show applications of the redundancy rules on notable families of CNF formulas. In Section 5.1 we consider minimally unsatisfiable formulas, while in Section 5.2 we consider the *weak Pigeonhole Principle.*

▶ Remark 5.1. Due to Theorem 4.2 and Theorem 4.3, we refer to a cost-SPR (resp. cost-PR, cost-SR) derivation from Γ of $b_{i_1}, \ldots, b_{i_k}$ for some distinct k blocking literals and all the $\overline{b}_j$ for $j \notin \{i_1, \ldots, i_k\}$, as a *proof of* $\mathrm{cost}(\Gamma) = k$ in cost-SPR (resp. cost-PR, cost-SR).

5.1 Short proofs of minimally unsatisfiable formulas

Recall the definition of PR from [6, Definition 1.16]. A PR calculus refutation of a CNF formula Γ is a sequence of clauses $D_1, \ldots, D_t$ 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. $\Gamma_i = \Gamma \cup \{D_1, \ldots, D_i\}$, that is D_{i+1} satisfies

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

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.

▶ **Theorem 5.2.** *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* $\mathrm{cost}(\{C_1 \vee b_1, \dots, C_m \vee b_m\}) = 1$ *of at most $\mathcal{O}(s+m)$ many clauses.*

Proof. Let $F = \{C_1, \dots, C_m\}$, and $\Gamma = \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ the corresponding MaxSAT instance. Let $\pi = (D_1, \dots, D_s)$ be a PR refutation of F, that is, in particular $D_s = \bot$. 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, just observe that, by construction, $\Gamma_i|_{\overline{D_{i+1}} \wedge \overline{B}} = F_i|_{\overline{D_{i+1}}}$, $(F_i \cup \{D_{i+1}\})|_\alpha = (\Gamma_i \cup \{D_{i+1} \vee B\})|_{\alpha'}$, and $F_i|_{\overline{D_{i+1}}} \vdash_1 (F_i \cup \{D_{i+1}\})|_\alpha$.

The last clause of π_B is B. Let α_{opt} be an optimal assignment of Γ. Since F is minimally unsatisfiable, $\mathrm{cost}(\alpha_{opt}) = 1$. 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 $\mathrm{cost}(\tau) \geq 1$ for any $\tau \supseteq \overline{E_i}$ and $\mathrm{cost}(\tau \circ \alpha_{opt}) = \mathrm{cost}(\alpha_{opt}) = 1$.

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 5.2 shows that the upper bounds for minimally unsatisfiable formulas in [6] translate immediately to the setting of this article. In particular, as a corollary of Theorem 5.2, we have that cost-PR proves in polynomial size that

- the *Pigeonhole Principle* with $n+1$ pigeons and n holes [6, Theorem 4.3] and [14, Section 5],
- the *Bit-Pigeonhole Principle* [6, Theorem 4.4],
- the *Parity Principle* [6, Theorem 4.6],
- the *Tseitin Principle* on a connected graph [6, Theorem 4.10],

have all cost 1, since they are all minimally unsatisfiable.

5.2 Short proofs of the minimum cost of PHP^m_n

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^m_n

$$\begin{aligned} &\textstyle\bigvee_{j \in [n]} p_{i,j} \vee b_i && \text{for } i \in [m], && \text{(totality)} \\ &\overline{p}_{i,j} \vee \overline{p}_{k,j} \vee b_{i,k,j} && \text{for } 1 \leq i < k \leq m \text{ and } j \in [n]. && \text{(injectivity)} \end{aligned}$$

We use $b_{k,i,j}$ as an alias of the variable $b_{i,k,j}$, given that $i < k$.

▶ **Theorem 5.3.** cost-SR *proves* $\mathrm{cost}(\mathsf{bPHP}^m_n) = m - n$ *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 [6].

▶ **Lemma 5.4.** *Let Γ and Σ be two sets of clauses. Any* cost-SR *derivation $D_1 \ldots, D_t$ 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 $\Gamma \cup \{D_1, \ldots, D_t\}$.*
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 $\Sigma|_{\overline{D_i}}$ and $\Sigma|_{\sigma_i}$ are subsumed by clauses in $\Gamma|_{\overline{D_i}}$ and $\Gamma|_{\sigma_i}$, respectively. Hence

$$(\Gamma \cup \{D_1, \ldots, D_{i-1}\})|_{\overline{D_i}} \vdash_1 (\Gamma \cup \Sigma \cup \{D_1, \ldots, D_{i-1}, D_i\})|_{\sigma_i}$$

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 5.5.** *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 $\Gamma|_{\overline{C}} \supseteq \Gamma|_{\overline{C} \circ \pi}$,*

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

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

Now we prove Theorem 5.3.

Proof of Theorem 5.3. 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^m_n we derive a slightly more structured formula F^m_n. Then we show how to derive F^{m-1}_{n-1} in a polynomial number of steps. The results follows because after n such derivations we obtain the formula F^{m-n}_0 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^{m-1}_{n-1} from F^m_n using the rules of cost-SR calculus. We divide the argument into several steps, but first we show how to derive F^m_n from bPHP^m_n.

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. $\mathsf{bPHP}^m_n \cup \{D_1, \ldots, D_{j-1}\}$. 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

$$\mathsf{bPHP}^m_n|_{\overline{D_j}} \supseteq (\mathsf{bPHP}^m_n \cup \{D_1, \ldots, D_j\})|_{\sigma_j} \ .$$

Indeed, on clauses of bPHP^m_n 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 $\mathsf{bPHP}^m_n \cup \mathsf{Ext}$.

Preprocessing 2. *"Enforce injectivity".* Optimal assignments for bPHP^m_n can have unassigned pigeons or have collisions between pigeons. We enforce the latter 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 $\mathsf{bPHP}^m_n \cup \mathsf{Ext}$ 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.

We do not need variables $b_{i,k,j}$ anymore. By one application of Lemma 5.4, 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.

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

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^m_n, 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 5.5 using as the witnessing permutation π_i, the permutation that

[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, see Appendix C.

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

$$(\mathsf{F}^m_n \cup \{C_1, \ldots, C_{(i-1)}\})|_{\overline{C_i}} \supseteq (\mathsf{F}^m_n \cup \{C_1, \ldots, C_{(i-1)}\})|_{\overline{C_i} \circ \pi_i} .$$

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

Step 2. *"If pigeon m flies to some hole, we can assume it flies to hole n".* We want to derive, in this order, the clauses

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

from Γ_1, to claim 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 5.5 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

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

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

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 5.4 to be able to ignore the subsumed clauses. The current database of clauses is Γ_3 is equal to

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

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 $\overline{p_{m,1}}, \ldots, \overline{p_{m,(n-1)}}$ 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 $\Gamma_4|_\sigma \subseteq \Gamma_4$.

Step 6. *"Reduction to $m-1$ pigeons and $n-1$ holes".* First we derive by unit propagation all the the totality clauses of F^{m-1}_{n-1}. That is, we remove the hole n from the totality axioms of the pigeons $1, \ldots, m-1$ in the current database. Now, the current database is F^{m-1}_{n-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 5.4 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^{m-1}_{n-1}.

Thus steps (1)–(6) are repeated $n-1$ times, up to derive F^{m-n}_0.

The unit clauses derived in the whole process include

- $b_1, \dots, b_{(m-n)}$ (totality clauses in F^{m-n}_0).
- $\overline{b_{n+1}}, \dots, \overline{b_m}$ (derived at each step of the induction),
- $\overline{b_{i,k,j}}$ for all $i < k$ and j (derived at the preprocessing).

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

6 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* [5] a proof starts from a set H_0 of hard clauses and a multiset of soft clauses S_0. At each step i a 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 $\bot$ 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 [1], first used in the context of MaxSAT in [4]. 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 $\{\overline{b}_1, \dots, \overline{b}_m\}$.

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

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

If the derivation obtains k copies of $\bot$ 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 $\bot$. Soundness and completeness of MaxSAT resolution are proved in [5].

As a deduction system on the soft clauses, MaxSAT resolution is simulated by resolution [5, Theorem 17]. Therefore, to build inference systems stronger than MaxSAT resolution, we introduce redundancy rules (Definition 3.2).

▶ **Definition 6.1** (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 $S = \{\overline{b}_1, \dots, \overline{b}_m\}$. 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') $H_i := H_{i-1} \cup \{C\}$, 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 $\bot$. Analogous systems can be defined strengthening MaxSAT resolution with any of cost-PR, cost-SPR, cost-LPR, cost-BC.

Definition 6.1 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 $\bot$ 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 $\overline{b_{i_1}}, \overline{b_{i_2}}, \dots, \overline{b_{i_k}}$, using rule (d), to get k empty clauses $\bot$. 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 [5].

▶ **Theorem 6.2** (completeness). *Let P denote any of* cost-SR*,* cost-PR*,* cost-SPR*,* cost-LPR*,* cost-BC*. The system MaxSAT resolution + P is complete.*

The soundness of the systems is proved by manipulating the proofs so that all the redundancy rules (a) and (a') are applied in a first block, followed by a second block of rules (b)–(d). Then the soundness of the two blocks follows from the soundness of cost-SPR and MaxSAT resolution respectively. See details in Appendix D.

▶ **Theorem 6.3** (soundness). *The system MaxSAT resolution +* cost-SR *is sound, i.e., if MaxSAT resolution +* cost-SR *can derive $\bot$ as a soft clause with multiplicity k from a formula $F = H_0 \land S_0$ with satisfiable hard clauses $H_0 = \{C_1 \lor b_1, \dots, C_m \lor b_m\}$ and soft clauses $S_0 = \{\overline{b}_1, \dots, \overline{b}_m\}$, then the number of clauses that need to be falsified in S_0 is at least k, or equivalently* $\mathrm{cost}(H_0) \geq k$*.*

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. We then showed how to integrate such calculi with MaxSAT resolution. 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 [16]. 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 [16]?
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 $\mathsf{poly}(s)$?

References

1 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`.

2 Jeremias Berg and Matti Järvisalo. Unifying reasoning and care-guided search for maximum satisfiability. In *16th European Conf. on Logics in Artificial Intelligence, JELIA*, Lecture Notes in Computer Science 11468, pages 287–303. Springer, 2019.

3 Maria Luisa Bonet, Sam Buss, Alexey Ignatiev, João Marques-Silva, and Antonio Morgado. MaxSAT resolution with the dual rail encoding. In *Proc. 32nd Intl. AAAI Conference on Artificial Intelligence (AAAI'18)*, 2018. Available at aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16782, date accessed: 07 Jul. 2019.

4 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.

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

6 Sam Buss and Neil Thapen. DRAT proofs, propagation redundancy, and extended resolution. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 71–89. Springer, 2019.

7 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. 44:36–50, 1979.

8 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.

9 Marijn J. H. Heule and Armin Biere. What a difference a variable makes. In *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference (TACAS 2018)*, Lecture Notes in Computer Science 10806, pages 75–92. Springer Verlag, 2018.

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

11 Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Verifying refutations with extended resolution. In *Automated Deduction - 24th International Conference (CADE)*, Lecture Notes in Computer Science 7898, pages 345–359. Springer Verlag, 2013.

12 Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Expressing symmetry breaking in DRAT proofs. In *Automated Deduction - 25th International Conference (CADE)*, Lecture Notes in Computer Science 9195, pages 591–606. Springer Verlag, 2015.

13 Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Short proofs without new variables. In *Automated Deduction - 26th International Conference (CADE)*, Lecture Notes in Computer Science 10395, pages 130–147. Springer Verlag, 2017.

14 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 [13]. doi:10.1007/s10817-019-09516-0.

15 Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere. PRuning through satisfaction. In *Hardware and Software: Verification and Testing - 13th International Haifa Verification Conference (HVC)*, Lecture Notes in Computer Science 10629, pages 179–194. Springer Verlag, 2017.

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

17 Matti Järvisalo, Marijn J. H. Heule, and Armin Biere. Inprocessing rules. In *Automated Reasoning - 6th International Joint Conference (IJCAR)*, Lecture Notes in Computer Science 7364, pages 355–270. Springer Verlag, 2012.

18 Benjamin Kiesl, Adrián Rebola-Pardo, and Marijn J. H. Heule. Extended resolution simulates DRAT. In *Automated Reasoning - 6th International Joint Conference (IJCAR)*, Lecture Notes in Computer Science 10900, pages 516–531. Springer Verlag, 2018.

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

20 Javier Larrosa and Federico Heras. Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In *Proc. Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 193–198. Professional Book Center, 2005.

21 Adrián Rebola-Pardo and Martin Suda. A theory of satisfiability-preserving proofs in SAT solving. In *Proc., 22nd Intl. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'22)*, EPiC Series in Computing 57, pages 583–603. EasyChair, 2018.

Appendix

A Extra material for Section 2

▶ **Observation 2.1** (restated from page 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 $D \in \Delta$, $\Gamma|_{\overline{D}} \vdash_1 \bot$. Let $D = \bigvee_{i \in I} \ell_i$. Saying that $\Gamma|_{\overline{D}} \vdash_1 \bot$ is equivalent to saying $\Gamma \wedge \bigwedge_{i \in I} \overline{\ell}_i \vdash_1 \bot$. This in turn implies

$$(\Gamma \wedge \bigwedge_{i \in I} \overline{\ell}_i)|_\sigma \vdash_1 \bot, \tag{6}$$

since σ maps unit literals into unit literals (or 1, 0). By basic properties of substitutions we have $(\Gamma \wedge \bigwedge_{i \in I} \overline{\ell}_i)|_\sigma = \Gamma|_\sigma \wedge \bigwedge_{i \in I} (\overline{\ell}_i)|_\sigma$ and $(\overline{\ell}_i)|_\sigma = \overline{(\ell_i|_\sigma)}$, that is (6) implies

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

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

B Extra material for Section 3

▶ **Definition B.1.** A clause $C \vee \ell$ is a *blocked clause* (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.

▶ **Observation B.2.** *A clause $C \vee \ell$ is BC w.r.t. Γ and literal ℓ if and only if the assignment $\overline{C} \cup \{\ell = 1\}$ is such that for every clause $D \in \Gamma$ containing the underlying variable of ℓ, $\overline{C} \cup \{\ell = 1\}$ satisfies D.*

Proof. Let $C \vee \ell$ be BC w.r.t. Γ and ℓ. Let D be a clause in Γ containing the underlying variable of ℓ, i.e. either $D = D' \vee \ell$ or $D = D' \vee \overline{\ell}$. In the first case it is trivial that $\overline{C} \cup \{\ell = 1\}$ satisfies D. In the second case, since $C \vee \ell$ is BC, we have that $C \vee D'$ is a tautology, i.e. again $\overline{C}$ satisfies D'.

Vice versa, assume that given a clause $C \vee \ell$, for every clause $D \in \Gamma$ containing the underlying variable of ℓ, the assignment $\overline{C} \cup \{\ell = 1\}$ satisfies D. That is, in particular $\overline{C} \cup \{\ell = 1\}$ satisfies the clauses of Γ of the form $D' \vee \overline{\ell}$. Which is only possible if $\overline{C}$ satisfies D', which is equivalent to saying $C \vee D'$ is a tautology. ◀

C Extra material for Section 5

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 C.1.** *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 $C = \ell_1 \vee \cdots \vee \ell_t$. 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 $D_1, \ldots, D_{i-1}$, 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

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

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 $D_1, \dots, D_i$. ◀

D Extra material for Section 6

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

▶ **Proposition D.1** (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. ◀

▶ **Theorem 6.3** (restated from page 15). *The system MaxSAT resolution +* cost-SR *is sound, i.e., if MaxSAT resolution +* cost-SR *can derive $\bot$ 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* $\mathrm{cost}(H_0) \geq k$.

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 $\bot$.

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

$$\min_{\alpha \,:\, \alpha \models H_i} c(\alpha, S_i) = \mathrm{cost}(H_0) \ . \tag{7}$$

Because of Proposition D.1 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 soundess of cost-SR (Lemma 3.4) we have that when $i \leq t_0$, H_i is satisfiable and $\mathrm{cost}(H_i) = \mathrm{cost}(H_0)$. Furthermore $S_i = S_0$. Hence

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

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

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

where the first equality is the soundness of MaxSAT resolution. ◀