Clause Redundancy and Preprocessing in Maximum Satisfiability*

Hannes Ihalainen, Jeremias Berg[0000−0001−7660−8061], and Matti Järvisalo[0000−0003−2572−063X]

HIIT, Department of Computer Science, University of Helsinki, Finland
{firstname.lastname}@helsinki.fi

Abstract. The study of clause redundancy in Boolean satisfiability (SAT) has proven significant in various terms, from fundamental insights into preprocessing and inprocessing to the development of practical proof checkers and new types of strong proof systems. We study liftings of the recently-proposed notion of propagation redundancy—based on a semantic implication relationship between formulas—in the context of maximum satisfiability (MaxSAT), where of interest are reasoning techniques that preserve optimal cost (in contrast to preserving satisfiability in the realm of SAT). We establish that the strongest MaxSAT-lifting of propagation redundancy allows for changing in a controlled way the set of minimal correction sets in MaxSAT. This ability is key in succinctly expressing MaxSAT reasoning techniques and allows for obtaining correctness proofs in a uniform way for MaxSAT reasoning techniques very generally. Bridging theory to practice, we also provide a new MaxSAT preprocessor incorporating such extended techniques, and show through experiments its wide applicability in improving the performance of modern MaxSAT solvers.

Keywords: maximum satisfiability · clause redundancy · propagation redundancy · preprocessing

1 Introduction

Building heavily on the success of Boolean satisfiability (SAT) solving [13], maximum satisfiability (MaxSAT) as the optimization extension of SAT constitutes a viable approach to solving real-world NP-hard optimization problems [35, 6]. In the context of SAT, the study of fundamental aspects of clause redundancy [29, 28, 20, 31, 21, 32, 23] has proven central for developing novel types of preprocessing and inprocessing-style solving techniques [29, 24] as well as in enabling efficient proof checkers [19, 18, 15, 16, 41, 42, 7] via succinct representation of most practical SAT solving techniques. Furthermore, clause redundancy notions have

* Work financially supported by Academy of Finland under grants 322869, 328718 and 342145. The authors wish to thank the Finnish Computing Competence Infrastructure (FCCI) for supporting this project with computational and data storage resources.

been shown to give rise to very powerful proof systems, going far beyond resolution [22, 23, 30]. In contrast to viewing clause redundancy through the lens of logical entailment, the redundancy criteria developed in this line of work are based on a semantic implication relationship between formulas, making them desirably efficient to decide and at the same time are guaranteed to merely preserve satisfiability rather than logical equivalence.

The focus of this work is the study of clause redundancy in the context of MaxSAT through lifting recently-proposed variants of the notion of *propagation redundancy* [23] based on a semantic implication relationship between formulas from the realm of SAT. The study of such liftings is motivated from several perspectives. Firstly, earlier it has been shown that a natural MaxSAT-lifting called SRAT [10] of the redundancy notion of the notion of *resolution asymmetric tautologies* (RAT) [29] allows for establishing the general correctness of MaxSAT-liftings of typical preprocessing techniques in SAT solving [14], alleviating the need for correctness proofs for individual preprocessing techniques [8]. However, the need for preserving the *optimal cost* in MaxSAT—as a natural counterpart for preserving satisfiability in SAT—allows for developing MaxSAT-centric preprocessing and solving techniques which cannot be expressed through SRAT [11, 2]. Capturing more generally such cost-aware techniques requires developing more expressive notions of clause redundancy. Secondly, due to the fundamental connections between solutions and so-called minimal corrections sets (MCSes) of MaxSAT instances [25, 8], analyzing the effect of clauses that are redundant in terms of expressive notions of redundancy on the MCSes of MaxSAT instances can provide further understanding on the relationship between the different notions and their fundamental impact on the solutions of MaxSAT instances. Furthermore, in analogy with SAT, more expressive redundancy notions may prove fruitful for developing further practical preprocessing and solving techniques for MaxSAT.

Our main contributions are the following. We propose natural liftings of the three recently-proposed variants PR, LPR and SPR of propagation redundancy in the context of SAT to MaxSAT. We provide a complete characterization of the relative expressiveness of the lifted notions CPR, CLPR and CSPR (C standing for cost for short) and of their impact on the set of MCSes in MaxSAT instances. In particular, while removing or adding clauses redundant in terms of CSPR and CLPR (the latter shown to be equivalent with SRAT) do not influence the set of MCSes underlying MaxSAT instances, CPR can in fact have an influence on MCSes. In terms of solutions, this result implies that CSPR or CLPR clauses can not remove minimal (in terms of sum-of-weights of falsified soft clauses) solutions of MaxSAT instances, while CPR clauses can.

The—theoretically greater—effect that CPR clauses have on the solutions of MaxSAT instances is key for succinctly expressing further MaxSAT reasoning techniques via CPR and allows for obtaining correctness proofs in a uniform way for MaxSAT reasoning techniques very generally; we give concrete examples of how CPR captures techniques not in the reach of SRAT. Bridging to practical preprocessing in MaxSAT, we also provide a new MaxSAT preprocessor ex-

tended with such techniques. Finally, we provide large-scale empirical evidence on the positive impact of the preprocessor on the runtimes of various modern MaxSAT solvers, covering both complete and incomplete approaches, suggesting that extensive preprocessing going beyond the scope of SRAT appears beneficial to integrate for speeding up modern MaxSAT solvers.

An extended version of this paper, with formal proofs missing from this version, is available via the authors' homepages.

2 Preliminaries

SAT. For a Boolean variable x there are two literals, the positive x and the negative $\neg x$, with $\neg\neg l = l$ for a literal l. A clause C is a set (disjunction) of literals and a CNF formula F a set (conjunction) of clauses. We assume that all clauses are non-tautological, i.e., do not contain both a literal and its negation. The set $\texttt{var}(C) = \{x \mid x \in C \text{ or } \neg x \in C\}$ consists of the variables of the literals in C. The set of variables and literals, respectively, of a formula are $\texttt{var}(F) = \bigcup_{C \in F} \texttt{var}(C)$ and $\texttt{lit}(F) = \bigcup_{C \in F} C$, respectively. For a set L of literals, the set $\neg L = \{\neg l \mid l \in L\}$ consists of the negations of the literals in L.

A *(truth) assignment* τ is a set of literals for which $x \notin \tau$ or $\neg x \notin \tau$ for any variable x. For a literal l we denote $l \in \tau$ by $\tau(l) = 1$ and $\neg l \in \tau$ by $\tau(l) = 0$ or $\tau(\neg l) = 1$ as convenient, and say that τ assigns l the value 1 and 0, respectively. The set $\texttt{var}(\tau) = \{x \mid x \in \tau \text{ or } \neg x \in \tau\}$ is the range of τ, i.e., it consists of the variables τ assigns a value for. For a set L of literals and an assignment τ, the assignment $\tau_L = (\tau \setminus \neg L) \cup L$ is obtained from τ by setting $\tau_L(l) = 1$ for all $l \in L$ and $\tau_L(l) = \tau(l)$ for all $l \notin L$ assigned by τ. For a literal l, τ_l stands for $\tau_{\{l\}}$. An assignment τ satisfies a clause C ($\tau(C) = 1$) if $\tau \cap C \neq \emptyset$ or equivalently if $\tau(l) = 1$ for some $l \in C$, and a CNF formula F ($\tau(F) = 1$) if it satisfies each clause $C \in F$. A CNF formula is satisfiable if there is an assignment that satisfies it, and otherwise unsatisfiable. The empty formula $\top$ is satisfied by any truth assignment and the empty clause $\bot$ is unsatisfiable. The Boolean satisfiability problem (SAT) asks to decide whether a given CNF formula F is satisfiable.

Given two CNF formulas F_1 and F_2, F_1 entails F_2 ($F_1 \models F_2$) if any assignment τ that satisfies F_1 and only assigns variables of F_1 (i.e. for which $\texttt{var}(\tau) \subset \texttt{var}(F_1)$) can be extended into an assignment $\tau^2 \supset \tau$ that satisfies F_2. The formulas are equisatisfiable if F_1 is satisfiable iff F_2 is. An assignment τ is complete for a CNF formula F if $\texttt{var}(F) \subset \texttt{var}(\tau)$, and otherwise partial for F. The restriction $F\big|_\tau$ of F wrt a partial assignment τ is a CNF formula obtained by (i) removing from F all clauses that are satisfied by τ and (ii) removing from the remaining clauses of F literals l for which $\tau(l) = 0$. Applying unit propagation on F refers to iteratively restricting F by $\tau = \{l\}$ for a unit clause (clause with a single literal) $(l) \in F$ until the resulting (unique) formula, denoted by $\texttt{UP}(F)$, contains no unit clauses or some clause in F becomes empty. We say that unit propagation on F derives a conflict if $\texttt{UP}(F)$ contains the empty clause. The formula F_1 implies F_2 under unit propagation ($F_1 \vdash_1 F_2$) if, for each $C \in F_2$,

unit propagation derives a conflict in $F_1 \wedge \{(\neg l) \mid l \in C\}$. Note that $F_1 \vdash_1 F_2$ implies $F_1 \models F_2$, but not vice versa in general.

Maximum Satisfiability. An instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{F}_S, w)$ of (weighted partial) maximum satisfiability (MaxSAT for short) consists of two CNF formulas, the hard clauses $\mathcal{F}_H$ and the soft clauses $\mathcal{F}_S$, and a weight function $w\colon \mathcal{F}_S \to \mathbb{N}$ that assigns a positive weight to each soft clause.

Without loss of generality, we assume that every soft clause $C \in \mathcal{F}_S$ is unit[1]. The set of *blocking* literals $\mathcal{B}(\mathcal{F}) = \{l \mid (\neg l) \in \mathcal{F}_S\}$ consists of the literals l the negation of which occurs in $\mathcal{F}_S$. The weight function w is extended to blocking literals by $w(l) = w((\neg l))$. Without loss of generality, we also assume that $l \in \mathtt{lit}(\mathcal{F}_H)$ for all $l \in \mathcal{B}(\mathcal{F})$[2]. Instead of using the definition of MaxSAT in terms of hard and soft clauses, we will from now on view a MaxSAT instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ as a set $\mathcal{F}_H$ of hard clauses, a set $\mathcal{B}(\mathcal{F})$ of blocking literals and a weight function $w\colon \mathcal{B}(\mathcal{F}) \to \mathbb{N}$.

Any complete assignment τ over $\mathtt{var}(\mathcal{F}_H)$ that satisfies $\mathcal{F}_H$ is a solution to $\mathcal{F}$. The cost $\mathtt{COST}(\mathcal{F}, \tau) = \sum_{l \in \mathcal{B}(\mathcal{F})} \tau(l) w(l)$ of a solution τ is the sum of weights of blocking literals it assigns to 1[3]. The cost of a complete assignment τ that does not satisfy $\mathcal{F}_H$ is defined as ∞. The cost of a partial assignment τ over $\mathtt{var}(\mathcal{F}_H)$ is defined as the cost of smallest-cost assignments that are extensions of τ. A solution τ^o is optimal if $\mathtt{COST}(\mathcal{F}, \tau^o) \leq \mathtt{COST}(\mathcal{F}, \tau)$ holds for all solutions τ of $\mathcal{F}$. The cost of the optimal solutions of a MaxSAT instance is denoted by $\mathtt{COST}(\mathcal{F})$, with $\mathtt{COST}(\mathcal{F}) = \infty$ iff $\mathcal{F}_H$ is unsatisfiable. In MaxSAT the task is to find an optimal solution to a given MaxSAT instance.

Example 1. Let $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ be a MaxSAT instance with $\mathcal{F}_H = \{(x \vee b_1), (\neg x \vee b_2), (y \vee b_3 \vee b_4), (z \vee \neg y \vee b_4), (\neg z)\}$, $\mathcal{B}(\mathcal{F}) = \{b_1, b_2, b_3, b_4\}$ having $w(b_1) = w(b_4) = 1$, $w(b_2) = 2$ and $w(b_3) = 8$. The assignment $\tau = \{b_1, b_4, \neg b_2, \neg b_3, \neg x, \neg z, y\}$ is an example of an optimal solution of $\mathcal{F}$ and has $\mathtt{COST}(\mathcal{F}, \tau) = \mathtt{COST}(\mathcal{F}) = 2$.

With a slight abuse of notation, we denote by $\mathcal{F} \wedge C = (\mathcal{F}_H \cup \{C\}, \mathcal{B}(\mathcal{F} \wedge C), w)$ the MaxSAT instance obtained by adding a clause C to an instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$. Adding clauses may introduce new blocking literals but not change the weights of already existing ones, i.e., $\mathcal{B}(\mathcal{F}) \subset \mathcal{B}(\mathcal{F} \wedge C)$ and $w^{\mathcal{F}}(l) = w^{\mathcal{F} \wedge C}(l)$ for all $l \in \mathcal{B}(\mathcal{F})$.

Correction Sets. For a MaxSAT instance $\mathcal{F}$, a subset $\mathrm{cs} \subset \mathcal{B}(\mathcal{F})$ is a minimal correction set (MCS) of $\mathcal{F}$ if (i) $\mathcal{F}_H \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F}) \setminus \mathrm{cs}} (\neg l)$ is satisfiable and (ii) $\mathcal{F}_H \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F}) \setminus \mathrm{cs}_s} (\neg l)$ is unsatisfiable for every $\mathrm{cs}_s \subsetneq \mathrm{cs}$. In words, cs is an MCS if it

[1] A soft clause C can be replaced by the hard clause $C \vee x$ and soft clause $(\neg x)$, where x is a variable not in $\mathtt{var}(\mathcal{F}_H \wedge \mathcal{F}_S)$, without affecting the costs of solutions.

[2] Otherwise the instance can be simplified by unit propagating $\neg l$ without changing the costs of solutions. As a consequence, any complete assignment for $\mathcal{F}_H$ will be complete for $\mathcal{F}_H \wedge \mathcal{F}_S$ as well.

[3] This is equivalent to the sum of weights of soft clauses not satisfied by τ.

is a subset-minimal set of blocking literals that is included in some solution τ of $\mathcal{F}$.[4] We denote the set of MCSes of $\mathcal{F}$ by $\texttt{mcs}(\mathcal{F})$.

There is a tight connection between the MCSes and solutions of MaxSAT instances. Given an optimal solution τ^o of a MaxSAT instance $\mathcal{F}$, the set $\tau^o \cap \mathcal{B}(\mathcal{F})$ is an MCS of $\mathcal{F}$. In the other direction, for any $\text{cs} \in \texttt{mcs}(\mathcal{F})$, there is a (not necessary optimal) solution τ^{cs} such that $\text{cs} = \mathcal{B}(\mathcal{F}) \cap \tau^{\text{cs}}$ and $\texttt{COST}(\mathcal{F}, \tau^{\text{cs}}) = \sum_{l \in \text{cs}} w(l)$.

Example 2. Consider the instance $\mathcal{F}$ from Example 1. The set $\{b_1, b_4\} \in \texttt{mcs}(\mathcal{F})$ is an MCS of $\mathcal{F}$ that corresponds to the optimal solution τ described in Example 1. The set $\{b_2, b_3\} \in \texttt{mcs}(\mathcal{F})$ is another example of an MCS that instead corresponds to the solution $\tau_2 = \{b_2, b_3, \neg b_1, \neg b_4, x, \neg z, \neg y\}$ for which $\texttt{COST}(\mathcal{F}, \tau) = 10$.

3 Propagation Redundancy in MaxSAT

We extend recent work [23] on characterizing redundant clauses using semantic implication in the context of SAT to MaxSAT. In particular, we provide natural counterparts for several recently-proposed strong notions of redundancy in SAT to the context of MaxSAT and analyze the relationships between them.

In the context of SAT, the most general notion of clause redundancy is seemingly simple: a clause C is redundant for a formula F if it does not affect its satisfiability, i.e., clause C is redundant wrt a CNF formula F if F and $F \wedge \{C\}$ are equisatisfiable [29, 20]. This allows for the set of satisfying assignments to change, and does not require preserving logical equivalence; we are only interested in satisfiability.

A natural counterpart for this general view in MaxSAT is that the *cost* of optimal solutions (rather than the set of optimal solutions) should be preserved.

Definition 1. *A clause C is redundant wrt a MaxSAT instance $\mathcal{F}$ if* $\texttt{COST}(\mathcal{F}) = \texttt{COST}(\mathcal{F} \wedge C)$.

This coincides with the counterpart in SAT whenever $\mathcal{B}(\mathcal{F}) = \emptyset$, since then the cost of a MaxSAT instance $\mathcal{F}$ is either 0 (if $\mathcal{F}_H$ is satisfiable) or ∞ (if $\mathcal{F}_H$ is unsatisfiable). Unless explicitly specified, we will use the term "redundant" to refer to Definition 1.

Following [23], we say that a clause C *blocks* the assignment $\neg C$ (and all assignments τ for which $\neg C \subset \tau$). As shown in the context of SAT [23], a clause C is redundant (in the equisatisfiability sense) for a CNF formula F if C does not block all of its satisfying assignments. The counterpart that arises in the context of MaxSAT from Definition 1 is that the cost of at least one of the solutions not blocked by C is no greater than the cost of $\neg C$.

Proposition 1. *A clause C is redundant wrt a MaxSAT instance $\mathcal{F}$ if and only if there is an assignment τ for which* $\texttt{COST}(\mathcal{F} \wedge C, \tau) = \texttt{COST}(\mathcal{F}, \tau) \leq \texttt{COST}(\mathcal{F}, \neg C)$.

[4] This is equivalent to a subset-minimal set of soft clauses falsified by τ.

The equality $\texttt{COST}(\mathcal{F} \wedge C, \tau) = \texttt{COST}(\mathcal{F}, \tau)$ of Proposition 1 is necessary, as witnessed by the following example.

Example 3. Consider the MaxSAT instance $\mathcal{F}$ detailed in Example 1, the clause $C = (b_5)$ with $b_5 \in \mathcal{B}(\mathcal{F} \wedge C)$ and the assignment $\tau = \{b_5\}$. Then $2 = \texttt{COST}(\mathcal{F}, \tau) \leq \texttt{COST}(\mathcal{F}, \neg C) = 2$ but C is not redundant since $\texttt{COST}(\mathcal{F} \wedge C) = 2 + w^{\mathcal{F} \wedge C}(b_5) > 2 = \texttt{COST}(\mathcal{F})$.

Proposition 1 provides a sufficient condition for a clause C being redundant. Further requirements on the assignment τ can be imposed without loss of generality.

Theorem 1. *A non-empty clause C is redundant wrt a MaxSAT instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ if and only if there is an assignment τ such that*
(i) $\tau(C) = 1$, (ii) $\mathcal{F}_H\big|_{\neg C} \models \mathcal{F}_H\big|_\tau$ and
(iii) $\texttt{COST}(\mathcal{F} \wedge C, \tau) = \texttt{COST}(\mathcal{F}, \tau) \leq \texttt{COST}(\mathcal{F}, \neg C)$.

As we will see later, a reason for including two additional conditions in Theorem 1 is to allow defining different restrictions of redundancy notions, some of which allow for efficiently identifying redundant clauses.

Example 4. Consider the instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ detailed in Example 1, a clause $C = (\neg x \vee b_5)$ for a $b_5 \in \mathcal{B}(\mathcal{F} \wedge C)$ and an assignment $\tau = \{\neg x, b_1\}$. Then: $\tau(C) = 1$, $\{(b_2), (y \vee b_3 \vee b_4), (z \vee \neg y \vee b_4), (\neg z)\} = \mathcal{F}_H\big|_{\neg C} \models \mathcal{F}_H\big|_\tau = \{(y \vee b_3 \vee b_4), (z \vee \neg y \vee b_4), (\neg z)\}$, and $2 = \texttt{COST}(\mathcal{F} \wedge C, \tau) = \texttt{COST}(\mathcal{F}, \tau) \leq \texttt{COST}(\mathcal{F}, \neg C) = 3$. We conclude that C is redundant.

In the context of SAT, imposing restrictions on the entailment operator and the set of assignments has been shown to give rise to several interesting redundancy notions which hold promise of practical applicability. These include three variants (LPR, SPR, and PR) of so-called (literal/set) propagation redundancy [23]. For completness we restate the definitions of these three notions. A clause C is LPR wrt a CNF formula F if there is a literal $l \in C$ for which $F\big|_{\neg C} \vdash_1 F\big|_{(\neg C)_l}$, SPR if the same holds for a subset $L \subset C$, and PR if there exists an assignment τ that satisfies C and for which $F\big|_{\neg C} \vdash_1 F\big|_\tau$. With the help of Theorem 1, we obtain counterparts for these notions in the context of MaxSAT.

Definition 2. *With respect to an instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$, a clause C is*

- ***cost literal propagation redundant (CLPR)*** *(on l) there is a literal $l \in C$ for which either (i) $\bot \in \texttt{UP}(\mathcal{F}_H\big|_{\neg C})$ or (ii) $l \notin \mathcal{B}(\mathcal{F} \wedge C)$ and $\mathcal{F}_H\big|_{\neg C} \vdash_1 \mathcal{F}_H\big|_{(\neg C)_l}$;*
- ***cost set propagation redundant (CSPR)*** *(on L) if there is a set $L \subset C \setminus \mathcal{B}(\mathcal{F} \wedge C)$ of literals for which $\mathcal{F}_H\big|_{\neg C} \vdash_1 \mathcal{F}_H\big|_{(\neg C)_L}$; and*
- ***cost propagation redundant (CPR)*** *if there is an assignment τ such that*
  (i) $\tau(C) = 1$, (ii) $\mathcal{F}_H\big|_{\neg C} \vdash_1 \mathcal{F}_H\big|_\tau$ and
  (iii) $\texttt{COST}(\mathcal{F} \wedge C, \tau) = \texttt{COST}(\mathcal{F}, \tau) \leq \texttt{COST}(\mathcal{F}, \neg C)$.

Example 5. Consider again $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ from Example 1. The clause $D = (b_1 \vee b_2)$ is CLPR wrt $\mathcal{F}$ since $\bot \in \mathtt{UP}(\mathcal{F}_H|_{\neg D})$ as $\{(x), (\neg x)\} \subset \mathcal{F}_H|_{\neg D}$. As for the redundant clause C and assignment τ detailed in Example 3, we have that C is CPR, since $\mathcal{F}_H|_\tau \subset \mathcal{F}_H|_{\neg C}$ which implies $\mathcal{F}_H|_{\neg C} \vdash_1 \mathcal{F}_H|_\tau$.

We begin the analysis of the relationship between these redundancy notions by showing that CSPR (and by extension CLPR) clauses also satisfy the MaxSAT-centric condition (iii) of Theorem 1. Assume that C is CSPR wrt a instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ on the set L.

Lemma 1. *Let $\tau \supset \neg C$ be a solution of $\mathcal{F}$. Then, $\mathtt{COST}(\mathcal{F}, \tau) \geq \mathtt{COST}(\mathcal{F}, \tau_L)$.*

The following corollary of Lemma 1 establishes that CSPR and CLPR clauses are redundant according to Definition 1.

Corollary 1. $\mathtt{COST}(\mathcal{F} \wedge C, (\neg C)_L) = \mathtt{COST}(\mathcal{F}, (\neg C)_L) \leq \mathtt{COST}(\mathcal{F}, \neg C)$.

The fact that CPR clauses are redundant follows trivially from the fact that $\mathcal{F}_H|_{\neg C} \vdash_1 \mathcal{F}_H|_\tau$ implies $\mathcal{F}_H|_{\neg C} \models \mathcal{F}_H|_\tau$. However, given a solution ω that does not satisfy a CPR clause C, the next example demonstrates that the assignment ω_τ need not have a cost lower than ω. Stated in another way, the example demonstrates that an observation similar to Lemma 1 does not hold for CPR clauses in general.

Example 6. Consider a MaxSAT instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ having $\mathcal{F}_H = \{(x \vee b_1), (\neg x, b_2)\}$, $\mathcal{B}(\mathcal{F}) = \{b_1, b_2\}$ and $w(b_1) = w(b_2) = 1$. The clause $C = (x)$ is CPR wrt $\mathcal{F}$, the assignment $\tau = \{x, b_2\}$ satisfies the three conditions of Definition 2. Now $\delta = \{\neg x, b_1\}$ is a solution of $\mathcal{F}$ that does not satisfy C for which $\delta_\tau = \{x, b_1, b_2\}$ and $1 = \mathtt{COST}(\mathcal{F}, \delta) < 2 = \mathtt{COST}(\mathcal{F}, \delta_\tau)$.

Similarly as in the context of SAT, verifying that a clause is CSPR (and by extension CLPR) can be done efficiently. However, in contrast to SAT, we conjecture that verifying that a clause is CPR can not in the general case be done efficiently, *even if the assignment τ is given.* While we will not go into detail on the complexity of identifying CPR clauses, the following proposition gives some support for our conjecture.

Proposition 2. *Let $\mathcal{F}$ be an instance and $k \in \mathbb{N}$. There is another instance $\mathcal{F}^M$, a clause C, and an assignment τ such that C is CPR wrt $\mathcal{F}^M$ if and only if $\mathtt{COST}(\mathcal{F}) \geq k$.*

As deciding if $\mathtt{COST}(\mathcal{F}) \geq k$ is NP-complete in the general case, Proposition 2 suggests that it may not be possible to decide in polynomial time if an assignment τ satisfies the three conditions of Definition 2 unless P=NP. This is in contrast to SAT, where verifying propagation redundancy can be done in polynomial time if the assignment τ is given, but is NP-complete if not [24].

The following observations establish a more precise relationship between the redundancy notions. For the following, let $\mathrm{RED}(\mathcal{F})$ denote the set of clauses that are redundant wrt a MaxSAT instance $\mathcal{F}$ according to Definition 1. Analogously, the sets $\mathrm{CPR}(\mathcal{F})$, $\mathrm{CSPR}(\mathcal{F})$ and $\mathrm{CLPR}(\mathcal{F})$ consist of the clauses that are CPR, CSPR and CLPR wrt $\mathcal{F}$, respectively.

Observation 1 $\mathrm{CLPR}(\mathcal{F}) \subset \mathrm{CSPR}(\mathcal{F}) \subset \mathrm{CPR}(\mathcal{F}) \subset \mathrm{RED}(\mathcal{F})$ *holds for any MaxSAT instance* $\mathcal{F}$*.*

Observation 2 *There are MaxSAT instances* $\mathcal{F}_1, \mathcal{F}_2$ *and* $\mathcal{F}_3$ *for which* $\mathrm{CLPR}(\mathcal{F}_1) \subsetneq \mathrm{CSPR}(\mathcal{F}_1)$*,* $\mathrm{CSPR}(\mathcal{F}_2) \subsetneq \mathrm{CPR}(\mathcal{F}_2)$ *and* $\mathrm{CPR}(\mathcal{F}_3) \subsetneq \mathrm{RED}(\mathcal{F}_3)$*.*

The proofs of Observations 1 and 2 follow directly from known results in the context of SAT [23] by noting that any CNF formula can be viewed as an instance of MaxSAT without blocking literals.

For a MaxSAT-centric observation on the relationship between the redundancy notions, we note that the concept of redundancy and CPR coincide for any MaxSAT instance that has solutions.

Observation 3 $\mathrm{CPR}(\mathcal{F}) = \mathrm{RED}(\mathcal{F})$ *holds for any MaxSAT instance* $\mathcal{F}$ *with* $\mathtt{COST}(\mathcal{F}) < \infty$*.*

We note that a result similar to Observation 3 could be formulated in the context of SAT. The SAT-counterpart would state that the concept of redundancy (in the equisatisfiability sense) coincides with the concept of propagation redundancy for SAT solving (defined e.g. in [23]) for *satisfiable* CNF formulas. However, assuming that a CNF formula is satisfiable is very restrictive in the context of SAT. In contrast, it is natural to assume that a MaxSAT instance admits solutions.

We end this section with a simple observation: adding a redundant clause C to a MaxSAT instance $\mathcal{F}$ preserves not only optimal cost, but optimal solutions of $\mathcal{F} \wedge C$ are also optimal solutions of $\mathcal{F}$. However, the converse need not hold; an instance $\mathcal{F}$ might have optimal solutions that do not satisfy C.

Example 7. Consider an instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ with $\mathcal{F}_H = \{(b_1 \vee b_2)\}$, $\mathcal{B}(\mathcal{F}) = \{b_1, b_2\}$ and $w(b_1) = w(b_2) = 1$. The clause $C = (\neg b_1)$ is CPR wrt $\mathcal{F}$. In order to see this, let $\tau = \{\neg b_1, b_2\}$. Then τ satisfies C (condition (i) of Definition 2). Furthermore, τ satisfies $\mathcal{F}_H$, implying $\mathcal{F}_H\big|_{\neg C} \vdash_1 \mathcal{F}_H\big|_{\tau}$ (condition (ii)). Finally, we have that $1 = \mathtt{COST}(\mathcal{F}, \tau) = \mathtt{COST}(\mathcal{F} \wedge C, \tau) \leq \mathtt{COST}(\mathcal{F}, \neg C) = 1$ (condition (iii)). The assignment $\delta = \{b_1, \neg b_2\}$ is an example of an optimal solution of $\mathcal{F}$ that is not a solution of $\mathcal{F} \wedge C$.

4 Propagation Redundancy and MCSes

In this section, we analyze the effect of adding redundant clauses on the MCSes of MaxSAT instances. As the main result, we show that adding CSPR (and by extension CLPR) clauses to a MaxSAT instance $\mathcal{F}$ preserves all MCSes while adding CPR clauses does not in general. Stated in terms of solutions, this means that adding CSPR clauses to $\mathcal{F}$ preserves not only all optimal solutions, but all solutions τ for which $(\tau \cap \mathcal{B}(\mathcal{F})) \in \mathtt{mcs}(\mathcal{F})$, while adding CPR clauses only preserves at least one optimal solution.

Effect of CLPR Clauses on MCSes. MaxSAT-liftings of four specific SAT solving techniques (including bounded variable elimination and self-subsuming

resolution) were earlier proposed in [8]. Notably, the correctness of the liftings was shown individually for each of the techniques by arguing individually that applying one of the liftings does not change the set of MCSes of any MaxSAT instance. Towards a more generic understanding of optimal cost preserving MaxSAT preprocessing, in [10] the notion of solution resolution asymmetric tautologies (SRAT) was proposed as a MaxSAT-lifting of the concept of resolution asymmetric tautologies (RAT). In short, a clause C is a SRAT clause for a MaxSAT instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ if there is a literal $l \in C \setminus \mathcal{B}(\mathcal{F} \wedge C)$ such that $\mathcal{F}_H \vdash_1 ((C \vee D) \setminus \{\neg l\})$ for every $D \in \mathcal{F}_H$ for which $\neg l \in D$.

In analogy with RAT [29], SRAT was shown in [10] to allow for a general proof of correctness for natural MaxSAT-liftings of a wide range of SAT preprocessing techniques, covering among other the four techniques for which individual correctness proofs were provided in [8]. The generality follows essentially from the fact that the addition and removal of SRAT clauses preserves MCSes. The same observations apply to CLPR, as CLPR and SRAT are equivalent.

Proposition 3. *A clause C is CLPR wrt $\mathcal{F}$ iff it is SRAT wrt $\mathcal{F}$.*

The proof of Proposition 3 follows directly from corresponding results in the context of SAT [23]. Informally speaking, a clause C is SRAT on a literal l iff it is RAT [29] on l and $l \notin \mathcal{B}(\mathcal{F})$. Similarly, a clause C is CLPR on a literal l iff it is LPR as defined in [23] on l and $l \notin \mathcal{B}(\mathcal{F})$. Proposition 3 together with previous results from [10] implies that the MCSes of MaxSAT instances are preserved under removing and adding CLPR clauses.

Corollary 2. *If C is CLPR wrt $\mathcal{F}$, then $\texttt{mcs}(\mathcal{F}) = \texttt{mcs}(\mathcal{F} \wedge C)$.*

Effect of CPR Clauses on MCSes. We turn our attention to the effect of CPR clauses on the MCSes of MaxSAT instances. Our analysis makes use of the previously-proposed MaxSAT-centric preprocessing rule known as *subsumed label elimination* (SLE) [11, 33][5].

Definition 3. *(Subsumed Label Elimination [11, 33]) Consider a MaxSAT instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ and a blocking literal $l \in \mathcal{B}(\mathcal{F})$ for which $\neg l \notin \texttt{lit}(\mathcal{F}_H)$. Assume that there is another blocking literal $l_s \in \mathcal{B}(\mathcal{F})$ for which (1) $\neg l_s \notin \texttt{lit}(\mathcal{F}_H)$, (2) $\{C \in \mathcal{F}_H \mid l \in C\} \subset \{C \in \mathcal{F}_H \mid l_s \in C\}$ and (3) $w(l) \geq w(l_s)$. The subsumed label elimination (SLE) rule allows adding $(\neg l)$ to $\mathcal{F}_H$.*

A specific proof of correctness of SLE was given in [11]. The following proposition provides an alternative proof based on CPR.

Proposition 4 (Proof of correctness for SLE). *Let $\mathcal{F}$ be a MaxSAT instance and assume that the blocking literals $l, l_s \in \mathcal{B}(\mathcal{F})$ satisfy the three conditions of Definition 3. Then, the clause $C = (\neg l)$ is CPR wrt $\mathcal{F}$.*

[5] Rephrased here using our notation.

Proof. We show that $\tau = \{\neg l, l_s\}$ satisfies the three conditions of Definition 2. First τ satisfies C (condition (i)). Conditions (1) and (2) of Definition 3 imply $\mathcal{F}_H\big|_\tau \subset \mathcal{F}_H\big|_{\neg C}$ which in turn implies $\mathcal{F}_H\big|_{\neg C} \vdash_1 \mathcal{F}_H\big|_\tau$ (condition (ii)).

As for condition (iii), the requirement $\texttt{COST}(\mathcal{F} \wedge C, \tau) = \texttt{COST}(\mathcal{F}, \tau)$ follows from $\mathcal{B}(\mathcal{F} \wedge C) = \mathcal{B}(\mathcal{F})$. Let $\delta \supset \neg C$ be a complete assignment of $\mathcal{F}_H$ for which $\texttt{COST}(\mathcal{F}, \delta) = \texttt{COST}(\mathcal{F}, \neg C)$. If $\texttt{COST}(\mathcal{F}, \delta) = \infty$ then $\texttt{COST}(\mathcal{F}, \tau) \leq \texttt{COST}(\mathcal{F}, \neg C)$ follows trivially. Otherwise $\delta \setminus \neg C$ satisfies $\mathcal{F}_H\big|_{\neg C}$ so by $\mathcal{F}_H\big|_{\neg C} \vdash_1 \mathcal{F}_H\big|_\tau$ it satisfies $\mathcal{F}_H\big|_\tau$ as well. Thus $\delta^R = ((\delta \setminus \neg C) \setminus \{\neg l \mid l \in \tau\}) \cup \tau = (\delta \setminus \{l, \neg l, \neg l_s\}) \cup \{\neg l, l_s\}$ is an extension of τ that satisfies $\mathcal{F}_H$ and for which $\texttt{COST}(\mathcal{F}, \tau) \leq \texttt{COST}(\mathcal{F}, \delta^R) \leq \texttt{COST}(\mathcal{F}, \delta)$ by condition (3) of Definition 3. Thereby τ satisfies the conditions of Definition 2 so C is CPR wrt $\mathcal{F}$. □

Example 8. The blocking literals $b_3, b_4 \in \mathcal{B}(\mathcal{F})$ of the instance $\mathcal{F}$ detailed in Example 1 satisfy the conditions of Definition 3. By Proposition 4 the clause $(\neg b_3)$ is CPR wrt $\mathcal{F}$.

In [11] it was shown that SLE does not preserve MCSes in general. By Corollary 2, this implies that SLE can not be viewed as the addition of CLPR clauses. Furthermore, by Proposition 4 we obtain the following.

Corollary 3. *There is a MaxSAT instance $\mathcal{F}$ and a clause C that is CPR wrt $\mathcal{F}$ for which $\texttt{mcs}(\mathcal{F}) \neq \texttt{mcs}(\mathcal{F} \wedge C)$.*

Effect of CSPR Clauses on MCSes. Having established that CLPR clauses preserve MCSes while CPR clauses do not, we complete the analysis by demonstrating that CSPR clauses preserve MCSes.

Theorem 2. *Let $\mathcal{F}$ be a MaxSAT instance and C a CSPR clause of $\mathcal{F}$. Then $\texttt{mcs}(\mathcal{F}) = \texttt{mcs}(\mathcal{F} \wedge C)$.*

Theorem 2 follows from the following lemmas and propositions. In the following, let C be a clause that is CSPR wrt a MaxSAT instance $\mathcal{F}$ on a set $L \subset C \setminus \mathcal{B}(\mathcal{F} \wedge C)$.

Lemma 2. *Let $cs \subset \mathcal{B}(\mathcal{F})$. If $\mathcal{F}_H \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F}) \setminus cs}(\neg l)$ is satisfiable, then $(\mathcal{F}_H \wedge C) \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F} \wedge C) \setminus cs}(\neg l)$ is satisfiable.*

Lemma 2 helps in establishing one direction of Theorem 2.

Proposition 5. $\texttt{mcs}(\mathcal{F}) \subset \texttt{mcs}(\mathcal{F} \wedge C)$.

Proof. Let $\text{cs} \in \texttt{mcs}(\mathcal{F})$. Then $\mathcal{F}_H \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F}) \setminus \text{cs}}(\neg l)$ is satisfiable, which by Lemma 2 implies that $(\mathcal{F}_H \wedge C) \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F} \wedge C) \setminus \text{cs}}(\neg l)$ is satisfiable.

To show that $(\mathcal{F}_H \wedge C) \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F} \wedge C) \setminus \text{cs}_s}(\neg l)$ is unsatisfiable for any $\text{cs}_s \subsetneq \text{cs} \subset \mathcal{B}(\mathcal{F})$, we note that any assignment satisfying $(\mathcal{F}_H \wedge C) \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F} \wedge C) \setminus \text{cs}_s}(\neg l)$ would also satisfy $\mathcal{F}_H \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F}) \setminus \text{cs}_s}(\neg l)$, contradicting $\text{cs} \in \texttt{mcs}(\mathcal{F})$. □

The following lemma is useful for showing inclusion in the other direction.

Lemma 3. *Let* $cs \in \texttt{mcs}(\mathcal{F} \wedge C)$. *Then* $cs \subset \mathcal{B}(\mathcal{F})$.

Lemma 3 allows for completing the proof of Theorem 2.

Proposition 6. $\texttt{mcs}(\mathcal{F} \wedge C) \subset \texttt{mcs}(\mathcal{F})$.

Proof. Let $\text{cs} \in \texttt{mcs}(\mathcal{F} \wedge C)$, which by Lemma 3 implies $\text{cs} \subset \mathcal{B}(\mathcal{F})$. Let τ be a solution that satisfies $(\mathcal{F}_H \wedge C) \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F} \wedge C) \setminus \text{cs}} (\neg l)$. Then τ satisfies $\mathcal{F}_H \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F}) \setminus \text{cs}} (\neg l)$. For contradiction, assume that $\mathcal{F}_H \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F}) \setminus \text{cs}_s} (\neg l)$ is satisfiable for some $\text{cs}_s \subsetneq \text{cs}$. Then by Lemma 2, $(\mathcal{F}_H \wedge C) \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F} \wedge C) \setminus \text{cs}_s} (\neg l)$ is satisfiable as well, contradicting $\text{cs} \in \texttt{mcs}(\mathcal{F} \wedge C)$. Thereby $\text{cs} \in \texttt{mcs}(\mathcal{F})$. □

Theorem 2 implies that SLE can not be viewed as the addition of CSPR clauses. In light of this, an interesting remark is that—in contrast to CPR clauses in general (recall Example 6)—the assignment τ used in the proof of Proposition 4 can be used to convert any assignment that does not satisfy the CPR clause detailed in Definition 3 into one that does, without increasing its cost.

Observation 4 *Let* $\mathcal{F}$ *be a MaxSAT instance and assume that the blocking literals* $l, l_s \in \mathcal{B}(\mathcal{F})$ *satisfy the three conditions of Definition 3. Let* $\tau = \{\neg l, l_s\}$ *and consider any solution* $\delta \supset \neg C$ *of* $\mathcal{F}$ *that does not satisfy the CPR clause* $C = (\neg l)$. *Then* δ_τ *is a solution of* $\mathcal{F} \wedge C$ *for which* $\texttt{COST}(\mathcal{F}, \delta_\tau) \leq \texttt{COST}(\mathcal{F}, \delta)$.

5 CPR-based Preprocessing for MaxSAT

Mapping the theoretical observations into practical preprocessing, in this section we discuss through examples how CPR clauses can be used as a unified theoretical basis for capturing a wide variety of known MaxSAT reasoning rules, and how they could potentially help in the development of novel MaxSAT reasoning techniques.

Our first example is the so-called *hardening rule* [2, 17, 26, 8]. In terms of our notation, given a solution τ to a MaxSAT instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ and a blocking literal $l \in \mathcal{B}(\mathcal{F})$ for which $w(l) > \texttt{COST}(\mathcal{F}, \tau)$, the hardening rule allows adding the clause $C = (\neg l)$ to $\mathcal{F}_H$.

The correctness of the hardening rule can be established with CPR clauses. More specifically, as $\texttt{COST}(\mathcal{F}, \tau) < w(l)$ it follows that $\tau(C) = 1$ (condition (i) of Definition 2). Since τ satisfies $\mathcal{F}$, we have that $\mathcal{F}_H\big|_\tau = \top$ so $\mathcal{F}_H\big|_{\neg C} \vdash_1 \mathcal{F}_H\big|_\tau$ (condition (ii)). Finally, as $\texttt{COST}(\mathcal{F}, \delta) \geq w(l) > \texttt{COST}(\mathcal{F}, \tau)$ holds for all $\delta \supset \neg C$ it follows that $\texttt{COST}(\mathcal{F}, \neg C) > \texttt{COST}(\mathcal{F}, \tau) = \texttt{COST}(\mathcal{F} \wedge C, \tau)$. As such, $(\neg l)$ is CPR clause wrt $\mathcal{F}$. If fact, instead of assuming $w(l) > \texttt{COST}(\mathcal{F}, \tau)$ it suffices to assume $w(l) \geq \texttt{COST}(\mathcal{F}, \tau)$ and $\tau(l) = 0$.

The hardening rule can not be viewed as the addition of CSPR or CLPR clauses because it does not in general preserve MCSes.

Example 9. Consider the MaxSAT instance $\mathcal{F}$ from Example 1 and a solution $\tau = \{b_1, b_2, b_4, \neg b_3, \neg z, x, y\}$. Since $\texttt{COST}(\mathcal{F}, \tau) = 3 < 8 = w(b_3)$, the clause $(\neg b_3)$ is CPR. However, $\texttt{mcs}(\mathcal{F}) \neq \texttt{mcs}(\mathcal{F} \wedge C)$ since the set $\{b_2, b_3\} \in \texttt{mcs}(\mathcal{F})$ is not an MCS of $\mathcal{F} \wedge C$ as $(\mathcal{F}_H \wedge C) \wedge \bigwedge_{l \in \mathcal{B}(\mathcal{F}) \setminus \text{cs}} (\neg l) = (\mathcal{F}_H \wedge (\neg b_3)) \wedge (\neg b_1) \wedge (\neg b_4)$ is not satisfiable.

Viewing the hardening rule through the lens of CPR clauses demonstrates novel aspects of the MaxSAT-liftings of propagation redundancy. In particular, instantiated in the context of SAT, an argument similar to the one we made for hardening shows that given a CNF formula F, an assignment τ satisfying F, and a literal l for which $\tau(l) = 0$, the clause $(\neg l)$ is redundant (wrt equisatisfiability). While formally correct, such a rule is not very useful for SAT solving. In contrast, in the context of MaxSAT the hardening rule is employed in various modern MaxSAT solvers and leads to non-trivial performance-improvements [4, 5].

As another example of capturing MaxSAT-centric reasoning with CPR, consider the so-called TrimMaxSAT rule [39]. Given a MaxSAT instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ and a literal $l \in \mathcal{B}(\mathcal{F})$ for which $\tau(l) = 1$ for all solutions of $\mathcal{F}$, the TrimMaxSAT rule allows adding the clause $C = (l)$ to $\mathcal{F}_H$. In this case the assumptions imply that all solutions of $\mathcal{F}$ also satisfy C, i.e., that $\mathcal{F}_H\big|_{\neg C}$ is unsatisfiable. As such, any assignment τ that satisfies C and $\mathcal{F}_H$ will also satisfy the three conditions of Definition 2 which demonstrates that C is CPR. It is, however, not CSPR since the only literal in C is blocking.

As a third example of capturing (new) reasoning techniques with CPR, consider an extension of the central variable elimination rule that allows (to some extent) for eliminating blocking literals.

Definition 4. *Consider a MaxSAT instance $\mathcal{F}$ and a blocking literal $l \in \mathcal{B}(\mathcal{F})$. Let* $\mathrm{BBVE}(\mathcal{F})$ *be the instance obtained by (i) adding the clause $C \vee D$ to $\mathcal{F}$ for every pair $(C \vee l), (D \vee \neg l) \in \mathcal{F}_H$ and (ii) removing all clauses $(D \vee \neg l) \in \mathcal{F}_H$. Then* $\mathtt{COST}(\mathcal{F}) = \mathtt{COST}(\mathrm{BBVE}(\mathcal{F}))$ *and* $\mathtt{mcs}(\mathcal{F}) = \mathtt{mcs}(\mathrm{BBVE}(\mathcal{F}))$.

On the Limitations of CPR. Finally, we note that while CPR clauses significantly generalize existing theory on reasoning and preprocessing rules for MaxSAT, there are known reasoning techniques that can not (at least straightforwardly) be viewed through the lens of propagation redundancy. For a concrete example, consider the so-called intrinsic atmost1 technique [26].

Definition 5. *Consider a MaxSAT instance $\mathcal{F}$ and a set $L \subset \mathcal{B}(\mathcal{F})$ of blocking literals. Assume that (i) $|\tau \cap \{\neg l \mid l \in L\}| \leq 1$ holds for any solution τ of $\mathcal{F}$ and (ii) $w(l) = 1$ for each $l \in L$. Now form the instance* $\text{AT-MOST-ONE}(\mathcal{F}, L)$ *by (i) removing each literal $l \in L$ from $\mathcal{B}(\mathcal{F})$, and (ii) adding the clause $\{(\neg l) \mid l \in L\} \cup \{l_L\}$ to $\mathcal{F}$, where l_L is a fresh blocking literal with $w(l_L) = 1$.*

It has been established that any optimal solution of $\text{AT-MOST-ONE}(\mathcal{F}, L)$ is an optimal solution of $\mathcal{F}$ [26]. However, as the next example demonstrates, the preservation of optimal solutions is in general not due to the clauses added being redundant, as applying the technique can affect optimal cost.

Example 10. Consider the MaxSAT instance $\mathcal{F} = (\mathcal{F}_H, \mathcal{B}(\mathcal{F}), w)$ with $\mathcal{F}_H = \{(l_i) \mid i = 1 \dots n\}$, $\mathcal{B}(\mathcal{F}) = \{l_1 \dots l_n\}$ and $w(l) = 1$ for all $l \in \mathcal{B}(\mathcal{F})$. Then $|\tau \cap \neg\mathcal{B}(\mathcal{F})| = 0 \leq 1$ holds for all solutions τ of $\mathcal{F}$ so the intrinsic-at-most-one technique can be used to obtain the instance $\mathcal{F}^2 = \text{AT-MOST-ONE}(\mathcal{F}, \mathcal{B}(\mathcal{F})) = (\mathcal{F}^2_H, \mathcal{B}(\mathcal{F}^2), w^2)$ with $\mathcal{F}^2_H = \mathcal{F}_H \cup \{(\neg l_1 \vee \dots \vee \neg l_n \vee l_L)\}$, $\mathcal{B}(\mathcal{F}^2) = \{l_L\}$ and

$w^2(l_L) = 1$. Now $\delta = \{l \mid l \in \mathcal{B}(\mathcal{F})\} \cup \{l_L\}$ is an optimal solution to both $\mathcal{F}^2$ and $\mathcal{F}$ for which $1 = \texttt{COST}(\mathcal{F}^2, \delta) < \texttt{COST}(\mathcal{F}, \delta) = n$.

Example 10 implies that the intrinsic atmost1 technique can not be viewed as the addition or removal of redundant clauses. Generalizing CPR to cover weight changes could lead to further insights especially due to potential connections with core-guided MaxSAT solving [1, 36–38].

6 MaxPre 2: More General Preprocesssing in Practice

Connecting to practice, we extended the MaxSAT preprocessor MaxPre [33] version 1 with support for techniques captured by propagation redundancy. The resulting MaxPre version 2, as outlined in the following, hence includes techniques which have previously only been implemented in specific solver implementations rather than in general-purpose MaxSAT preprocessors.

First, let us mention that the earlier MaxPre [33] version 1 assumes that any blocking literals only appear in a single polarity among the hard clauses. Removing this assumption—supported by theory developed in Sections 3-4—decreases the number of auxiliary variables that need to be introduced when a MaxSAT instance is rewritten to only include unit soft clauses. For example, consider a MaxSAT instance $\mathcal{F}$ with $\mathcal{F}_H = \{(\neg x \vee y), (\neg y \vee x)\}$ and $\mathcal{F}_S = \{(x), (\neg y)\}$. For preprocessing the instance, MaxPre 1 extends both soft clauses with a new, auxiliary variable and runs preprocessing on the instance $\mathcal{F} = \{(\neg x \vee y), (\neg y \vee x), (x \vee b_1), (\neg y \vee b_2)\}$ with $\mathcal{B}(\mathcal{F}) = \{b_1, b_2\}$. In contrast, MaxPre 2 detects that the clauses in $\mathcal{F}_S$ are unit and reuses them as blocking literals, invoking preprocessing on $\mathcal{F} = \{(\neg x \vee y), (\neg y \vee x)\}$ with $\mathcal{B}(\mathcal{F}) = \{\neg x, y\}$.

In addition to the techniques already implemented in MaxPre 1, MaxPre 2 includes the following additional techniques: hardening [2], a variant TrimMaxSAT [39] that works on all literals of a MaxSAT instance, the intrinsic atmost1 technique [26] and a MaxSAT-lifting of failed literal elimination [12]. In short, failed literal elimination adds the clause $(\neg l)$ to the hard clauses $\mathcal{F}_H$ of an instance in case unit-propagation derives a conflict in $\mathcal{F}_H \wedge \{(l)\}$. Additionally, the implementation of failed literal elimination attempts to identify implied equivalences between literals that can lead to further simplification.

For computing the solutions required by TrimMaxSAT and detecting the cardinality constraints required by intrinsic-at-most-one constraints, MaxPre 2 uses the Glucose 3.0 SAT-solver [3]. For computing solutions required by hardening, MaxPre 2 additionally uses the SatLike incomplete MaxSAT solver [34] within preprocessing. MaxPre 2 is available in open source at `https://bitbucket.org/coreo-group/maxpre2/`.

We emphasize that, while the additional techniques implemented by MaxPre 2 have been previously implemented as heuristics in specific solver implementations, MaxPre 2 is—to the best of our understanding—the first stand-alone implementation supporting techniques whose correctness cannot be established with previously-proposed MaxSAT redundancy notions (i.e., SRAT). The goal

of our empirical evaluation presented in the next section is to demonstrate the potential of viewing expressive reasoning techniques not only as solver heuristics, but as a separate step in the MaxSAT solving process whose correctness can be established via propagation redundancy.

7 Empirical Evaluation

We report on results from an experimental evaluation of the potential of incorporating more general reasoning in MaxSAT preprocessing. In particular, we evaluated both complete solvers (geared towards finding provably-optimal solutions) and incomplete solvers (geared towards finding relatively good solutions fast) on standard heterogenous benchmarks from recent MaxSAT Evaluations. All experiments were run on 2.60-GHz Intel Xeon E5-2670 8-core machines with 64 GB memory and CentOS 7. All reported runtimes include the time used in preprocessing (when applicable).

7.1 Impact of Preprocessing on Complete Solvers

We start by considering recent representative complete solvers covering three central MaxSAT solving paradigms: the core-guided solver CGSS [27] (as a recent improvement to the successful RC2 solver [26]), and the MaxSAT Evaluation 2021 versions of the implicit hitting set based solver MaxHS [17] and the solution-improving solver Pacose [40]. For each solver S we consider the following variants.

- S: S in its default configuration.
- S `no preprocess`: S with the solver's own internal preprocessing turned off (when applicable).
- S+`maxpre1`: S after applying MaxPre 1 using its default configuration.
- S+`maxpre2 no new`: S after applying MaxPre 2 using the default configuration of MaxPre 1.
- S+`maxpre2-w:<TECH>`: S after applying MaxPre 2 using the standard configuration of MaxPre 1 and additional techniques integrated into MaxPre 2 (as detailed in Section 6) as specified by `<TECH>`.

More precisely, `<TECH>` specifies which of the techniques HTVGR are applied: H for hardening, T and V for TrimMaxSAT on blocking and non-blocking literals, respectively, G for intrinsic-at-most-one-constraints and R for failed literal elimination. It should be noted that an exhaustive evaluation of all subsets and application orders of these techniques is infeasible in practice. Based on preliminary experiments, we observed that the following choices were promising: HRT for CGSS and MaxHS, and HTVGR for Pacose; we report results using these individual configurations.

As benchmarks, we used the combined set of weighted instances from the complete tracks of MaxSAT Evaluation 2020 and 2021. After removing duplicates, this gave a total of 1117 instances. We enforced a per-instance time limit of

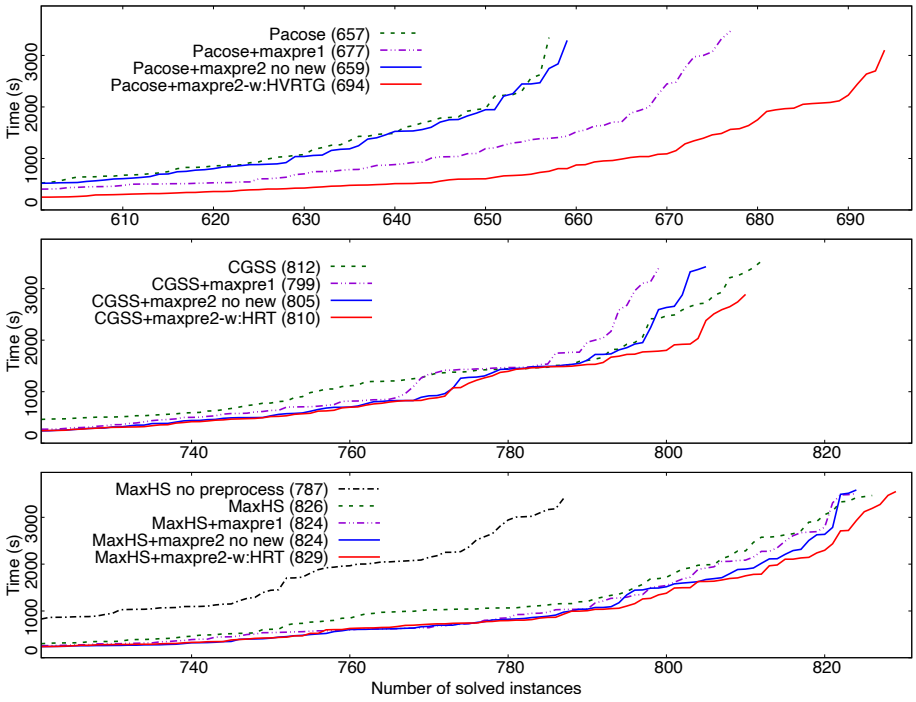

Fig. 1. Impact of preprocessing on complete solvers. For each solver, the number of instances solved within a 60-min per-instance time limit in parentheses.

60 minutes and memory limit of 32 GB. Furthermore, we enforced a per-instance 120-second time limit on preprocessing.

An overview of the results is shown in Figure 1, illustrating for each solver the number of instances solved (x-axis) under different per-instance time limits (y-axis). We observe that for both CGSS and MaxHS, S+`maxpre1` and S+`maxpre2-no-new` leads to less instances solved compared to S. In contrast, S+`maxpre2-w:HRT`, i.e., incorporating the stronger reasoning techniques of MaxPre 2, performs best of all preprocessing variants and improves on MaxHS also in terms of the number of instances solved. For Pacose, we observe that both Pacose+`maxpre1` and Pacose+`maxpre2 no new` (without the stronger reasoning techniques) already improve the performance of Pacose, leading to more instances solved. Incorporating the stronger reasoning rules further significantly improves performance, with Pacose+`maxpre2-w:HVRTG` performing the best among all of the Pacose variants.

7.2 Impact of Preprocessing on Incomplete MaxSAT Solving

As a representative incomplete MaxSAT solver we consider the MaxSAT Evaluation 2021 version of Loandra [9], as the best-performing solver in the incomplete

Table 1. Impact of preprocessing on the incomplete solver Loandra. The wins are organized column-wise, the cell on row X column Y contains the total number of instances that the solver on column Y wins over the solver on row X.

#Wins	`base` (`maxpre1`)	`no-prepro`	`maxpre2 no new`	`maxpre2-w:VG`
`base` (`maxpre1`)	—	154	135	152
`no-prepro`	208	—	216	218
`maxpre2 no new`	105	143	—	77
`maxpre2-w:VG`	110	140	80	—
Score (avg):	0.852	0.840	0.863	0.870

track of MaxSAT Evaluation under a 300s per-instance time limit on weighted instances. Loandra combines core-guided and solution-improving search towards finding good solutions fast. We consider the following variants of Loandra.

- `base` (`maxpre1`): Loandra in its default configuration which makes use of MaxPre 1.
- `no-prepro`: Loandra with its internal preprocessing turned off.
- `maxpre2 no new`: `base` with its internal preprocessor changed from MaxPre 1 to MaxPre 2 using the default configuration of MaxPre 1.
- `maxpre2-w:VG`: `maxpre2` incorporating the additional intrinsic-at-most-one constraints technique and the extension of TrimMaxSAT to non-blocking literals (cf. Section 6), found promising in preliminary experimentation.

As benchmarks, we used the combined set of weighted instances from the incomplete tracks of MaxSAT Evaluation 2020 and 2021. After removing duplicates, this gave a total of 451 instances. When reporting results, we consider for each instance and solver the cost of the best solution found by the solver within 300 seconds (including time spent preprocessing and solution reconstruction).

We compare the relative runtime performance of the solver variants using two metrics: *#wins* and the average *incomplete score*. Assume that τ_x and τ_y are the lowest-cost solutions computed by two solvers X and Y on a MaxSAT instance $\mathcal{F}$ and that $\texttt{best-cost}(\mathcal{F})$ is the lowest cost of a solution of $\mathcal{F}$ found either in our evaluation or in the MaxSAT Evaluations. Then X wins over Y if $\texttt{COST}(\mathcal{F}, \tau_x) < \texttt{COST}(\mathcal{F}, \tau_y)$. The incomplete score, $\texttt{score}(\mathcal{F}, X)$, obtained by solver X on $\mathcal{F}$ is the ratio between the cost of the solution found by X and $\texttt{best-cost}(\mathcal{F})$, i.e., $\texttt{score}(\mathcal{F}, X) = (\texttt{best-cost}(\mathcal{F}) + 1)/(\texttt{COST}(\mathcal{F}, \tau_x) + 1)$. The score of X on $\mathcal{F}$ is 0 if X is unable to find any solutions within 300 seconds.

An overview of the results is shown in Table 1. The upper part of the table shows a pairwise comparison on the number of wins over all benchmarks. The wins are organized column-wise, i.e., the cell on row X column Y contains the total number of instances that the solver on column Y wins over the solver on row X. The last row contains the average score obtained by each solver over all instances. We observe that any form of preprocessing improves the performance of Loandra, as witnessed by the fact that `no-prepro` is clearly the worst-

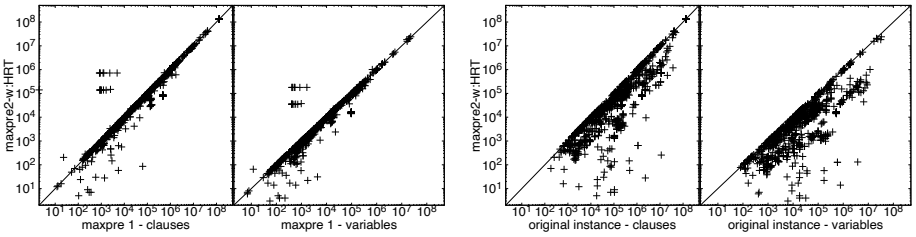

Fig. 2. Impact of preprocessing on instance size.

performing variant. The variants that make use of MaxPre 2 outperform the baseline under both metrics; both `maxpre2 no new` and `maxpre2-w:VG` obtain a higher average score and win on more instances over `base`. The comparison between `maxpre2 no new` and `maxpre2-w:VG` is not as clear. On one hand, the score obtained by `maxpre2-w:VG` is higher. On the other hand, `maxpre2 no new` wins on 80 instances over `maxpre2-w:VG` and looses on 77. This suggests that the quality of solutions computed by `maxpre2-w:VG` is on average higher, and that on the instances on which `maxpre2 no new` wins the difference is smaller.

7.3 Impact of Preprocessing on Instance Sizes

In addition to improved solver runtimes, we note that MaxPre 2 has a positive effect on the size of instances (both in terms of the number of variables and clauses remaining) when compared to preprocessing with MaxPre 1; see Figure 2 for a comparison, with `maxpre2-w:HRT` compared to `maxpre1` (left) and to original instance sizes (right).

8 Conclusions

We studied liftings of variants of propagation redundancy from SAT in the context of maximum satisfiability where—more fine-grained than in SAT—of interest are reasoning techniques that preserve optimal cost. We showed that CPR, the strongest MaxSAT-lifting, allows for changing minimal corrections sets in MaxSAT in a controlled way, thereby succinctly expressing MaxSAT reasoning techniques very generally. We also provided a practical MaxSAT preprocessor extended with techniques captured by CPR and showed empirically that extended preprocessing has a positive overall impact on a range of MaxSAT solvers. Interesting future work includes the development of new CPR-based preprocessing rules for MaxSAT capable of significantly affecting the MaxSAT solving pipeline both in theory and practice, as well as developing an understanding of the relationship between redundancy notions and the transformations performed by MaxSAT solving algorithms.

References

1. Ansótegui, C., Bonet, M., Levy, J.: SAT-based MaxSAT algorithms. Artificial Intelligence **196**, 77–105 (2013)
2. Ansótegui, C., Bonet, M.L., Gabàs, J., Levy, J.: Improving SAT-based weighted MaxSAT solvers. In: Proc. CP. Lecture Notes in Computer Science, vol. 7514, pp. 86–101. Springer (2012)
3. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: Proc. IJCAI. pp. 399–404 (2009)
4. Bacchus, F., Berg, J., Järvisalo, M., Martins, R. (eds.): MaxSAT Evaluation 2020: Solver and Benchmark Descriptions, Department of Computer Science Report Series B, vol. B-2020-2. Department of Computer Science, University of Helsinki (2020)
5. Bacchus, F., Järvisalo, M., Martins, R. (eds.): MaxSAT Evaluation 2019: Solver and Benchmark Descriptions, Department of Computer Science Report Series B, vol. B-2019-2. Department of Computer Science, University of Helsinki (2019)
6. Bacchus, F., Järvisalo, M., Martins, R.: Maximum satisfiability. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, chap. 24, pp. 929–991. Frontiers in Artificial Intelligence and Applications, IOS Press (2021)
7. Baek, S., Carneiro, M., Heule, M.J.H.: A flexible proof format for SAT solver-elaborator communication. In: Proc. TACAS. Lecture Notes in Computer Science, vol. 12651, pp. 59–75. Springer (2021)
8. Belov, A., Morgado, A., Marques-Silva, J.: SAT-based preprocessing for MaxSAT. In: Proc. LPAR-19. Lecture Notes in Computer Science, vol. 8312, pp. 96–111. Springer (2013)
9. Berg, J., Demirovic, E., Stuckey, P.J.: Core-boosted linear search for incomplete MaxSAT. In: Proc. CPAIOR. Lecture Notes in Computer Science, vol. 11494, pp. 39–56. Springer (2019)
10. Berg, J., Järvisalo, M.: Unifying reasoning and core-guided search for maximum satisfiability. In: Proc. JELIA. Lecture Notes in Computer Science, vol. 11468, pp. 287–303. Springer (2019)
11. Berg, J., Saikko, P., Järvisalo, M.: Subsumed label elimination for maximum satisfiability. In: Proc. ECAI. Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 630–638. IOS Press (2016)
12. Bhalla, A., Lynce, I., de Sousa, J.T., Marques-Silva, J.P.: Heuristic-based backtracking for propositional satisfiability. In: Proc. EPIA. Lecture Notes in Computer Science, vol. 2902, pp. 116–130. Springer (2003)
13. Biere, A., Heule, M., van Maaren, H., Walsh, T.: Handbook of Satisfiability (Second Edition): Volume 336 Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, The Netherlands, The Netherlands (2021)
14. Biere, A., Järvisalo, M., Kiesl, B.: Preprocessing in SAT solving. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, chap. 9, pp. 391–435. Frontiers in Artificial Intelligence and Applications, IOS Press (2021)
15. Cruz-Filipe, L., Heule, M.J.H., Hunt Jr., W.A., Kaufmann, M., Schneider-Kamp, P.: Efficient certified RAT verification. In: Proc. CADE. Lecture Notes in Computer Science, vol. 10395, pp. 220–236. Springer (2017)
16. Cruz-Filipe, L., Marques-Silva, J., Schneider-Kamp, P.: Efficient certified resolution proof checking. In: Proc. TACAS. Lecture Notes in Computer Science, vol. 10205, pp. 118–135 (2017)

17. Davies, J., Bacchus, F.: Exploiting the power of MIP solvers in MaxSAT. In: Proc. SAT. Lecture Notes in Computer Science, vol. 7962, pp. 166–181. Springer (2013)
18. Heule, M., Hunt Jr., W.A., Kaufmann, M., Wetzler, N.: Efficient, verified checking of propositional proofs. In: Proc. ITP. Lecture Notes in Computer Science, vol. 10499, pp. 269–284. Springer (2017)
19. Heule, M., Hunt Jr., W.A., Wetzler, N.: Bridging the gap between easy generation and efficient verification of unsatisfiability proofs. Softw. Test. Verification Reliab. **24**(8), 593–607 (2014)
20. Heule, M., Järvisalo, M., Lonsing, F., Seidl, M., Biere, A.: Clause Elimination for SAT and QSAT. Journal of Artificial Intelligence Research **53**, 127–168 (2015)
21. Heule, M., Kiesl, B.: The potential of interference-based proof systems. In: Proc. ARCADE@CADE. EPiC Series in Computing, vol. 51, pp. 51–54. EasyChair (2017)
22. Heule, M.J.H., Kiesl, B., Biere, A.: Short proofs without new variables. In: Proc. CADE. Lecture Notes in Computer Science, vol. 10395, pp. 130–147. Springer (2017)
23. Heule, M.J.H., Kiesl, B., Biere, A.: Strong extension-free proof systems. J. Autom. Reasoning **64**(3), 533–554 (2020)
24. Heule, M.J.H., Kiesl, B., Seidl, M., Biere, A.: PRuning through satisfaction. In: Proc. HVC. Lecture Notes in Computer Science, vol. 10629, pp. 179–194. Springer (2017)
25. Hou, A.: A theory of measurement in diagnosis from first principles. Artif. Intell. **65**(2), 281–328 (1994)
26. Ignatiev, A., Morgado, A., Marques-Silva, J.: RC2: an efficient MaxSAT solver. J. Satisf. Boolean Model. Comput. **11**(1), 53–64 (2019)
27. Ihalainen, H., Berg, J., Järvisalo, M.: Refined core relaxation for core-guided maxsat solving. In: Proc. CP. LIPIcs, vol. 210, pp. 28:1–28:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
28. Järvisalo, M., Biere, A., Heule, M.: Simulating circuit-level simplifications on CNF. J. Autom. Reason. **49**(4), 583–619 (2012)
29. Järvisalo, M., Heule, M., Biere, A.: Inprocessing rules. In: Proc. IJCAR. Lecture Notes in Computer Science, vol. 7364, pp. 355–370. Springer (2012)
30. Kiesl, B., Rebola-Pardo, A., Heule, M.J.H., Biere, A.: Simulating strong practical proof systems with extended resolution. J. Autom. Reason. **64**(7), 1247–1267 (2020)
31. Kiesl, B., Seidl, M., Tompits, H., Biere, A.: Super-blocked clauses. In: Proc. IJCAR. Lecture Notes in Computer Science, vol. 9706, pp. 45–61. Springer (2016)
32. Kiesl, B., Seidl, M., Tompits, H., Biere, A.: Local redundancy in SAT: Generalizations of blocked clauses. Log. Methods Comput. Sci. **14**(4) (2018)
33. Korhonen, T., Berg, J., Saikko, P., Järvisalo, M.: MaxPre: An extended MaxSAT preprocessor. In: Proc. SAT. Lecture Notes in Computer Science, vol. 10491, pp. 449–456 (2017)
34. Lei, Z., Cai, S.: Solving (weighted) partial MaxSAT by dynamic local search for SAT. In: Proc. IJCAI. pp. 1346–1352. ijcai.org (2018)
35. Li, C., Manyà, F.: MaxSAT, Hard and Soft Constraints. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, pp. 613–631. IOS Press (2009)
36. Morgado, A., Dodaro, C., Marques-Silva, J.: Core-guided MaxSAT with soft cardinality constraints. In: Proc. CP. Lecture Notes in Computer Science, vol. 8656, pp. 564–573. Springer (2014)

37. Morgado, A., Heras, F., Liffiton, M., Planes, J., Marques-Silva, J.: Iterative and core-guided MaxSAT solving: A survey and assessment. Constraints **18**(4), 478–534 (2013)
38. Narodytska, N., Bacchus, F.: Maximum satisfiability using core-guided MaxSAT resolution. In: Proc. AAAI. pp. 2717–2723. AAAI Press (2014)
39. Paxian, T., Raiola, P., Becker, B.: On preprocessing for weighted MaxSAT. In: Proc. VMCAI. Lecture Notes in Computer Science, vol. 12597, pp. 556–577. Springer (2021)
40. Paxian, T., Reimer, S., Becker, B.: Dynamic polynomial watchdog encoding for solving weighted maxsat. In: SAT. Lecture Notes in Computer Science, vol. 10929, pp. 37–53. Springer (2018)
41. Rebola-Pardo, A., Cruz-Filipe, L.: Complete and efficient DRAT proof checking. In: Proc. FMCAD. pp. 1–9. IEEE (2018)
42. Yolcu, E., Wu, X., Heule, M.J.H.: Mycielski graphs and PR proofs. In: Proc. SAT. Lecture Notes in Computer Science, vol. 12178, pp. 201–217. Springer (2020)