Impact of SAT-Based Preprocessing on Core-Guided MaxSAT Solving*

Jeremias Berg and Matti Järvisalo

Helsinki Institute for Information Technology HIIT, Department of Computer Science,
University of Helsinki, Finland

Abstract. We present a formal analysis of the impact of Boolean satisfiability (SAT) based preprocessing techniques on core-guided solvers for the constraint optimization paradigm of maximum satisfiability (MaxSAT). We analyze the behavior of two solver abstractions of the core-guided approaches. We show that SAT-based preprocessing has no effect on the best-case number of iterations required by the solvers. This implies that, with respect to best-case performance, the potential benefits of applying SAT-based preprocessing in conjunction with core-guided MaxSAT solvers are in principle solely a result of speeding up the individual SAT solver calls made during MaxSAT search. We also show that, in contrast to best-case performance, SAT-based preprocessing can improve the worst-case performance of core-guided approaches to MaxSAT.

1 Introduction

Real-world applications [1–18] of maximum satisfiability (MaxSAT) [19–21], the optimization counterpart of the famous Boolean satisfiability problem (SAT) [22, 23], are increasing in numbers as recent breakthroughs in MaxSAT solvers [24–32] are making MaxSAT more and more competitive as a constraint optimization paradigm.

A great majority of state-of-the-art MaxSAT solvers for solving optimization problems from the real world are *core-guided* [20, 21], heavily relying on the power of SAT solvers as very effective means of proving unsatisfiability of subsets of soft constraints, or *unsat cores*, in an iterative fashion towards an optimal solution. Thus new breakthroughs in techniques for speeding up SAT solvers also have the potential of directly speeding up MaxSAT solvers further. One particularly fruitful line of research on speeding up SAT solvers has been the development of effective preprocessing techniques [33–35], applied most typically before search, as well as most recently also as inprocessing [34], i.e., during SAT search. Compared to SAT, preprocessing for MaxSAT has seen some but arguably less progress so far [26, 30, 36–39]. Recently, ways of employing preprocessing techniques developed for pure SAT in the context of MaxSAT have been explored [26, 30, 40]. However, the impact of SAT-based preprocessing for MaxSAT solving seems to often be somewhat more modest than in the context of SAT solving [26, 30, 40]. The exact reasons for this difference are currently unclear; specifically, we are not aware of studies towards fundamental understanding on the potential of SAT-based preprocessing in the context of MaxSAT.

* Work supported by Academy of Finland, grants 251170 COIN, 276412, 284591; and DoCS Doctoral Scool in Computer Science at the University of Helsinki.

In this paper, we aim at providing further understanding on the potential of SAT-based preprocessing techniques in speeding up modern MaxSAT solvers. More specifically, we formally analyze the impact of SAT-based preprocessing techniques on the best-case and worst-case behavior of core-guided MaxSAT solvers [41–43]. As the basis of our analysis, we focus on two abstractions MaxSAT solvers which together cover a number of modern core-guided MaxSAT solvers [25, 30, 42]. As the formal metric, we focus on the impact of SAT-based preprocessing on the best-case and worst-case number of iterations, which—although not the only possible metric—is a natural choice of metric applied in the literature for analyzing iterative SAT-based approaches in various problem settings [41–45] and which has also been subjected to some extent to empirical analysis for understanding specific MaxSAT solving approaches [46].

As the main contributions, considering *best-case performance* of the abstract core-guided solvers, we show that SAT-based preprocessing *has no effect* on the number of iterations required by the solvers. In fact, this is true regardless of assumptions on the type of cores (guaranteed-minimal or not) the underlying SAT solver (unsat core extractor) provides to the MaxSAT solvers; thus our analysis also sheds light on the impact of core minimization on the performance of the abstract core-guided solvers. Essentially, our results imply that, in terms of best-case performance—assuming optimal search heuristics—the potential benefits of applying SAT-based preprocessing in conjunction with core-guided MaxSAT solvers are solely a result of speeding up the individual SAT solver calls made during MaxSAT search. Furthermore, contrasting the results for best-case behavior, we also show that SAT-based preprocessing does, in cases, improve *worst-case* performance of core-guided MaxSAT solvers (without ever having a negative effect on the worst-case number of iterations).

This paper is organized as follows. After preliminaries on MaxSAT and SAT-based preprocessing for MaxSAT (Sect. 2), we detail abstractions of core-guided MaxSAT solvers we focus on (Sect. 3). Before detailed proofs of our results (provided in Sects. 5–6), we present a detailed overview of the main contributions (Sect. 4).

2 Preliminaries

Maximum satisfiability. For every Boolean variable x there are two literals: the positive literal x and the negative literal $\neg x$. A clause C is a disjunction of literals, and a CNF formula F is a conjunction of clauses. When convenient, we treat a clause as a set of literals and a CNF formula as a set of clauses. We denote by $\textsc{Var}(F)$ the set of variables appearing in F. A truth assignment is a function $\tau\colon \textsc{Var}(F) \to \{0,1\}$. A clause C is satisfied by τ if $\tau(l) = 1$ for a positive literal or $\tau(l) = 0$ for a negative literal $l \in C$. A CNF formula F is satisfied by τ if τ satisfies all clauses $C \in F$. A formula F is satisfiable if there is a truth assignment that satisfies it, otherwise it is unsatisfiable.

A (weighted partial) MaxSAT instance $F = (F_h, F_s, w)$ consists of two CNF formulas, F_h (hard clauses) and F_s (soft clauses), together with a function $w\colon F_s \to \mathbb{N}$ assigning a positive weight $w(C)$ to each $C \in F_s$. If $w(C) = 1$ for all $C \in F_s$, the instance is unweighted. An (unsatisfiable) *core* of a MaxSAT instance F is a subset $\kappa \subseteq F_s$ such that $\kappa \wedge F_h$ is unsatisfiable. A core is minimal (a MUS) if no $\kappa_s \subset \kappa$ is a core of F. We denote the set of all MUSes of F by $\mathsf{mus}(F)$. For a subset $S \subseteq F_s$ and

clause $C \in S$, C is *necessary* for S if $F_h \wedge S$ is unsatisfiable and $F_h \wedge (S \setminus \{C\})$ is satisfiable.

An assignment τ that satisfies F_h is a *solution* to a MaxSAT instance F. For a solution τ, let $\text{COST}(F, \tau) = \sum_{C \in F_s} w(C) \cdot (1 - \tau'(C))$, i.e., the sum of the weights of soft clauses in F not satisfied by τ. A solution τ is optimal if $\text{COST}(F, \tau) \leq \text{COST}(F, \tau')$ for every solution τ'; we denote the cost of F, i.e., the value $\text{COST}(F, \tau)$ for optimal solutions τ, by $\text{COST}(F)$. Given a MaxSAT instance F, the MaxSAT problem asks to find an optimal solution to F.

SAT-Based Preprocessing for MaxSAT. Preprocessing is today an integral part of SAT solving [33, 34]. Consisting of applying a combination of satisfiability-preserving simplification (or rewriting) rules on the input CNF formula F to obtain a preprocessed CNF formula $\mathsf{pre}(F)$, a central aim of preprocessing is to speed up the runtime of a SAT solver so that the combined preprocessing time and solving time on $\mathsf{pre}(F)$ is shorter than the runtime of the solver on F. Several preprocessing techniques for SAT have been proposed. In this work we will focus on bounded variable elimination, subsumption elimination, self-subsuming resolution, and blocked clause elimination, as perhaps the most common preprocessing techniques in modern SAT solving.

Resolution. Given two clauses $C = C_1 \vee l$ and $D = D_1 \vee \neg l$ of F, the *resolution rule* states that the clause $C \bowtie_l D = C_1 \vee D_1$, called the *resolvent*, can be inferred by *resolving* on the literal l. This is lifted to two sets $S_l \subseteq F$ and $S_{\neg l} \subseteq F$ of clauses that contain the literal l and $\neg l$, respectively, by $S_l \bowtie_l S_{\neg l} = \{C \bowtie_l D \mid C \in S_l, D \in S_{\neg l}, \text{ and } C \bowtie_l D \text{ is not a tautology}\}$.

Bounded Variable Elimination (BVE) [33]. For a variable $x \in \text{VAR}(F)$, denote by F_x ($F_{\neg x}$) the clauses of F containing the literal x ($\neg x$). If $|F_x \bowtie_x F_{\neg x}| \leq |F_x \cup F_{\neg x}|$, the BVE rule allows converting the formula F to $(F \setminus (F_x \cup F_{\neg x})) \cup (F_x \bowtie_x F_{\neg x})$.

Subsumption Elimination (SE). A clause $C \in F$ subsumes another clause $D \in F$ if $C \subseteq D$. The SE rule allows for removing subsumed clauses from F.

Self-Subsuming Resolution (SSR). Given two clauses $C, D \in F$ s.t. $C = C_1 \vee l$, $D = D_1 \vee \neg l$ for a literal l and $D_1 \subseteq C_1$, the SSR rule allows for replacing C by C_1.

Blocked Clause Elimination (BCE) [47]. A clause $C \in F$ is blocked if it contains a literal $l \in C$ s.t $C \bowtie_l D$ is a tautology for all $D \in F_{\neg l}$. BCE allows removing blocked clauses from F.

Example 1. Consider the CNF formula
$F = \{(x \vee y), (\neg t \vee \neg z), (\neg z \vee y), (\neg y \vee z), (z \vee t), (x), (y \vee t), (z \vee t \vee x)\}$. Due to the clause (x), SE allows for removing $(x \vee y)$ and $(z \vee t \vee x)$. After this, using BVE to eliminate z, results in the formula $\mathsf{pre}(F) = \{(\neg t \vee \neg y), (t \vee y), (x)\}$.

As shown in [26], many important SAT preprocessing techniques, including BVE, SE, and SSR, cannot be used directly on MaxSAT instances. However, a correct lifting on these techniques for MaxSAT is enabled by the so-called *labelled CNF* (LCNF) framework [26, 48]. The LCNF framework enables correct applications of SAT-based preprocessing techniques on a MaxSAT instance $F = (F_h, F_s, w)$ using the procedure

1. $F_s^a = \{(C \vee l_C) \mid C \in F_s,\ l_C \text{ is a fresh variable}\}$.
2. Run VE, SSR, SE, and BCE on $F_h \cup F_s^a$ until fixpoint to obtain $\mathsf{pre}(F)_h$.
3. $\mathsf{pre}(F)_s = \{(\neg l_C) \mid \exists C' \in \mathsf{pre}(F)_h, l_C \in C'\}$.
4. $w^P((\neg l_C)) = w(C)$ for all $(\neg l_C) \in \mathsf{pre}(F)_s$.
5. Return $\mathsf{pre}(F) = (\mathsf{pre}(F)_h, \mathsf{pre}(F)_s, w^P)$.

Fig. 1. Applying SAT-based preprocessing to MaxSAT instance $F = (F_h, F_s, w)$.

outlined in Figure 1. Each soft clause $C \in F_s$ is augmented with a fresh *label variable* l_C (Step 1). Then SAT preprocessing is applied on the CNF formula $F_h \cup F_s^a$ (Step 2). To ensure correctness in terms of MaxSAT, the preprocessor needs to be restricted from resolving on any of the label variables. The hard clauses of $\mathsf{pre}(F)$ are the clauses output by the SAT preprocessor on $F_h \cup F_s^a$ (Step 3). The soft clauses of $\mathsf{pre}(F)$ contain a unit negation of each label variable that has not been eliminated by preprocessing; the weight function w^P assigns to each $(\neg l_C)$ the same weight as was assigned to C by w (Step 4). Finally, the procedure returns the preprocessed instance $\mathsf{pre}(F) = (\mathsf{pre}(F)_h, \mathsf{pre}(F)_s, w^P)$ (Step 5). The soft clauses of $\mathsf{pre}(F)$ are all unit soft clauses $(\neg l_C)$ where the variable l_C was added to some soft clause $C \in F_s$ of the original instance F in Step 1. Due to BVE, the variable l_C might appear in more than one hard clause of $\mathsf{pre}(F)$ and there might be literals that have been eliminated entirely from the formula during preprocessing.

Example 2. Let $F = (F_h, F_s)$ be an unweighted MaxSAT instance with $F_h = \{(x \vee y), (z), (z \vee t)\}$ and $F_s = \{(\neg x), (\neg y), (\neg t)\}$. Augmenting the soft clauses with the label variables l_1, l_2, and l_3 to form $F_s^a = \{(\neg x \vee l_1), (\neg y \vee l_2), (\neg t \vee l_3)\}$, and applying SAT-based preprocessing (BVE and SE) results in the instance $\mathsf{pre}(F)$ with $\mathsf{pre}(F)_h = \{(l_1 \vee l_2), (z)\}$ and $\mathsf{pre}(F)_s = \{(\neg l_1), (\neg l_2)\}$. Notice that preprocessing eliminates the label l_3.

Correctness of SAT-based preprocessing for MaxSAT is summarized as follows [26].

Theorem 1 ([26]). *Let F be a MaxSAT instance and $\mathsf{pre}(F)$ the instance resulting from preprocessing F according to the procedure in Figure 1. The following hold: (i) $\text{COST}(F) = \text{COST}(\mathsf{pre}(F))$; (ii) any optimal solution to $\mathsf{pre}(F)$ restricted to $\text{VAR}(F)$ is an optimal solution to F; and (iii) $\{C_1, \ldots, C_n\} \in \mathsf{mus}(F)$ iff $\{(\neg l_{C_1}), \ldots, (\neg l_{C_n})\} \in \mathsf{mus}(\mathsf{pre}(F))$.*

3 Core-guided MaxSAT Algorithms

In this section we detail the two abstractions of MaxSAT algorithms we analyze in this work: *CG* and *HS*. Both are examples of so-called *core-guided MaxSAT solvers*, one of the most successful current MaxSAT solving approaches with several variants, e.g. [49–51, 42, 28, 52, 31]. *CG* (Figure 2 left) is the same abstraction as studied in [53]. CG works by iteratively calling a SAT solver to extract unsatisfiable cores and ruling out each of the found cores by exploiting *cardinality constraints*. HS (Figure 2 right)

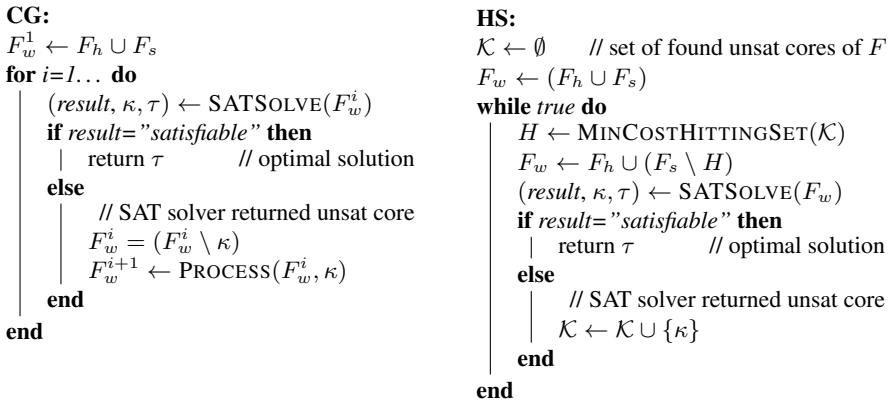

Fig. 2. Abstractions of MaxSAT solvers: CG (left) and HS (right), given a MaxSAT instance $F = (F_h, F_s, w)$ as input.

follows the implicit hitting set approach to MaxSAT [54, 55], iteratively using a SAT solver to extract unsatisfiable cores, and an exact minimum-cost hitting set algorithm to compute hitting sets over the found cores.

In more detail, at each iteration i, CG checks the satisfiability of a working formula F_w^i, which initially contains all clauses in the input formula, using a SAT solver. If F_w^i is satisfiable, CG returns the satisfying assignment τ returned by the SAT solver restricted onto the variables of F. Otherwise, the SAT solver returns a core κ^i of F_w^i. Finally, CG forms the next working formula F_w^{i+1} by processing the core κ^i. The exact method in which CG processes κ^i is left abstract. Following [53], we consider algorithms that extend soft clauses with blocking variables and impose hard linear (in)equalities over the blocking variables. More precisely, CG is allowed to modify the soft clauses $C \in F_s^i$ by two operations: $\mathbf{Relax}(C)$ and $\mathbf{Clone}(C, w)$.

- $\mathbf{Relax}(C)$ allows replacing C by $C \vee b$ where b is a new *blocking variable* not appearing anywhere else in the formula.
- $\mathbf{Clone}(C, w)$ allows adding a soft duplicate C' of C to the formula and relaxing C' by calling $\mathbf{Relax}(C')$. The (relaxed) *clone* C' is assigned weight w, and w is subtracted from the weight of C (C is discarded once it has weight 0).

In addition to these operations, CG is also allowed to add hard linear (in)equalities (cardinality, or more precisely, pseudo-Boolean, constraints) over the blocking variables. Given a cardinality constraint $\sum w_i \cdot x_i \circ K$ over variables x_i, constants w_i, and $\circ \in \{=, <, \leq\}$, we denote by $\text{CNF}(\sum w_i \cdot x_i \circ K)$ a CNF encoding of such a constraint. Following most core-guided MaxSAT algorithm implementations, we place two important restrictions on how CG can process the cores it encounters. First, the cardinality constraints are not allowed to mention any of the variables in the initial formula F. Second, if the algorithm extracts n cores during solving an instance F, and w_m^i is the smallest weight over all clauses in the ith core extracted, the optimum cost of F is $\text{COST}(F) = \sum_{i=1}^{n} w_m^i$. A concrete example of an algorithm fitting the CG model is the

WPM1 algorithm [50], concurrently proposed as WMSU1 [51], as an extension of the classical Fu-Malik algorithm [49] to weighted MaxSAT. Given a core κ^i, WPM1 first computes w_m^i. Then it calls $\mathbf{Clone}(C^i, w_m^i)$ for each $C^i \in \kappa^i$ and adds an *exactly-one* constraint $\sum b_i = 1$ over the blocking variables b_i added during the cloning operation.

HS is a hybrid algorithm, instantiated in [25, 55], that uses a SAT solver for core extraction from a working formula F_w, initially all clauses of the working formula. Given a collection $\mathcal{K}$ of extracted cores, HS uses an exact algorithm (an integer programming solver in practice) to find a minimum-cost hitting set hs over $\mathcal{K}$. The working formula is then updated to contain all clauses of F except for the soft clauses in hs, and the SAT solver invoked again. If the working formula is satisfiable, the satisfying assignment obtained is an optimal solution to F. Otherwise another core is obtained and the search continues with hitting set computation.

4 Overview of Results

In this section we give an overview of the main contributions of this paper. The algorithm-dependent formal proofs are provided after this overview in Sections 5–6.

We start by first defining the metric with respect to which we perform the formal analysis. The definition, intuitively matching with the number of iterations made by the abstract MaxSAT solvers considered, relies on the concept of *core traces*. Informally, a core trace T is a finite sequence of MaxSAT cores matching a possible execution of a core-guided MaxSAT solver. More formally, given a MaxSAT instance F and $\mathcal{A} \in \{\text{CG}, \text{HS}\}$, a sequence $(\kappa^1, \ldots, \kappa^n)$ of cores is an $\mathcal{A}$ *core trace* on F if there exists an execution of $\mathcal{A}$ on F such that (i) the core extracted by $\mathcal{A}$ at iteration i is κ^i; and (ii) $\mathcal{A}$ terminates after having encountered all cores in the sequence (i.e., the $(n+1)$th SAT solver call is satisfiable). For a core trace T, we denote by $|T|$ the number of cores in T, i.e., the *length* of T. Whenever appropriate, we refer to $\mathcal{A}$ core traces on F simply as $\mathcal{A}$ traces on F.

As the metric under analysis, we consider both the *minimum* and *maximum length* over all possible $\mathcal{A}$ traces for different choices of $\mathcal{A}$. More specifically, for $\mathcal{A} \in \{\text{CG}, \text{HS}\}$, we analyze the relative minimum and maximum lengths of core traces for the following variants of $\mathcal{A}$.

- $\mathcal{A}_{\mathsf{pre}}$: $\mathcal{A}$ applied after SAT-based preprocessing (recall Fig. 1).
- $\mathcal{A}^{\mathsf{mus}}$: $\mathcal{A}$ using a SAT solver that is guaranteed to return a MUS when invoked on an unsatisfiable formula (notice that an $\mathcal{A}^{\mathsf{mus}}$ trace contains only MUSes).
- $\mathcal{A}^{\mathsf{mus}}_{\mathsf{pre}}$: $\mathcal{A}^{\mathsf{mus}}$ applied after SAT-based preprocessing.

For a MaxSAT instance F, we denote by $\mathsf{minlen}(\mathcal{A}, F)$ and $\mathsf{maxlen}(\mathcal{A}, F)$ the minimum and maximum length $\mathcal{A}$ traces on F, respectively, or in other words, the best-case and worst-case number of iterations required by $\mathcal{A}$ for solving F.

Results. We provide a full characterization of the effect of preprocessing on the maximum and minimum length of core traces on F. The results on the best-case performance (minimum lengths of core traces) are summarized in Figure 3 for $\mathcal{A} \in \{\text{CG}, \text{HS}\}$. In the figure, an edge $X \to Y$ indicates that, for any MaxSAT instance F, the shortest

X core trace on F is at most as long as the shortest Y core trace on F. Analogously, our results for the worst-case performance (maximum lengths of core traces) are summarized in Figure 4. Here the edge $X \to Y$ indicates that, for any MaxSAT instance F, the longest X core trace on F is at most as long as the longest Y core trace on F; $X \nrightarrow Y$ indicates that $X \to Y$ does not hold. In words, we will provide in the following sections detailed proofs for the fact that SAT-based preprocessing cannot lower the minimum number of iterations required by CG or HS. For some intuition, we will show that for $\mathcal{A} \in \{\text{CG}, \text{HS}\}$, one of the shortest $\mathcal{A}$ core traces on any MaxSAT instance F is also a $\mathcal{A}^{\mathsf{mus}}$ trace, and that preprocessing cannot alter the MUS structure nor the $\mathcal{A}^{\mathsf{mus}}$ traces on F. In contrast, we will also show that preprocessing can improve the worst-case performance of both of the algorithms. Intuitively, this is due to the fact that preprocessing can remove soft clauses that are not members of any MUSes of F and hence do not contribute to the unsatisfiability of F, but still might force either algorithm to iterate unnecessarily many times.

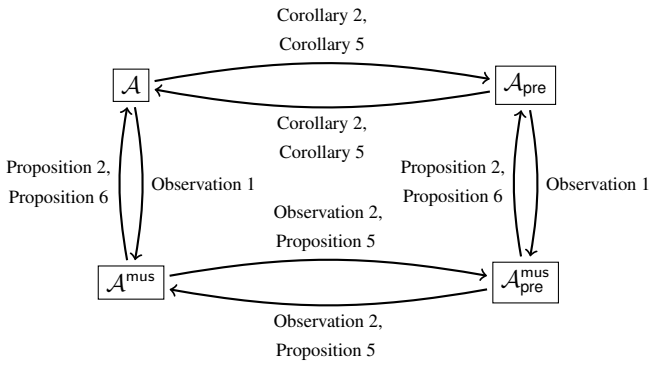

Fig. 3. Best-case performance in the number of iterations of $\mathcal{A} \in \{\text{CG}, \text{HS}\}$. Here $X \to Y$ iff $\mathsf{minlen}(X, F) \leq \mathsf{minlen}(Y, F)$ for all instances F.

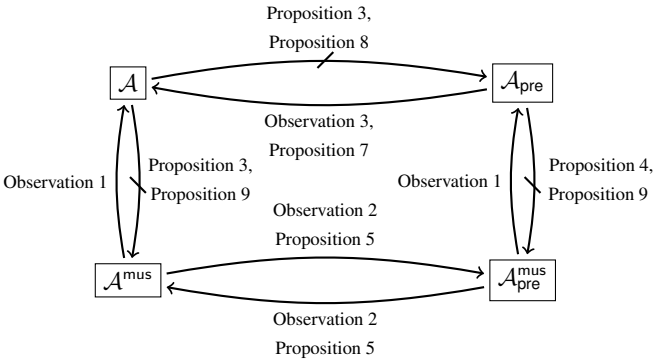

Fig. 4. Worst-case performance in the number of iterations of $\mathcal{A} \in \{\text{CG}, \text{HS}\}$. Here $X \to Y$ iff $\mathsf{maxlen}(X, F) \leq \mathsf{maxlen}(Y, F)$ for all F, and $X \nrightarrow Y$ indicates that $X \to Y$ does not hold.

We proceed now throughout Sects. 5–6 by providing formal proofs for all of the results summarized in Figures 3 and 4. Before the more involved proofs, we start with an algorithm-independent observation and an auxiliary result that makes the remaining proofs simpler by allowing us to assume MaxSAT instances to have a specific form without loss of generality.

Observation 1 *For $\mathcal{A} \in \{CG, HS\}$ and any MaxSAT instance F, any $\mathcal{A}^{\mathsf{mus}}$ trace on F is also an $\mathcal{A}$ trace on F. Hence $\mathsf{maxlen}(\mathcal{A}^{\mathsf{mus}}, F) \leq \mathsf{maxlen}(\mathcal{A}, F)$ and $\mathsf{minlen}(\mathcal{A}^{\mathsf{mus}}, F) \geq \mathsf{minlen}(\mathcal{A}, F)$.*

Finally, in the remaining proofs, we will use the fact that Theorem 1 guarantees that SAT-based preprocessing does not affect the set of MUSes of F in terms of of the mapping $(\neg l_C) \rightarrow C$ between the soft clauses of $\mathsf{pre}(F)$ and F. In order to avoid explicitly referring to this mapping in every proof, we will employ a technical observation from [40]. More specifically, *we will assume for the remaining part of this paper* that the soft clauses $C \in F_s$ of each MaxSAT instance F have already been augmented with label variables l_C to form the hard clause $C \vee l_C$ and the soft clause $(\neg l_C)$. In other words, we will assume that all soft clauses of F are unit negative literals $(\neg l_C)$ with the variable l_C not appearing negatively in any other clause and only appearing positively among the hard clauses. Under this assumption, the literals appearing in the soft clauses of F can be reused as label variables while preprocessing [40], thus removing the need of adding any new variables. Hence $\mathsf{pre}(F)_s \subseteq F_s$, and Theorem 1 can be simplified.

Corollary 1 (of Theorem 1) *Let F be a MaxSAT instance and $\mathsf{pre}(F)$ the instance resulting after preprocessing F. Then $\mathsf{mus}(F) = \mathsf{mus}(\mathsf{pre}(F))$.*

Most importantly, our assumption on the form of MaxSAT instances *does not affect core traces*. A proof for this auxiliary result is provided in Appendix A.

Proposition 1. *Let $F = (F_h, F_s, w)$ be a MaxSAT instance, and $F^P = (F_h \cup F_s^a, F_s^P, w^P)$ the MaxSAT instance with $F_s^a = \{C \vee l_C \mid C \in F_s,\ l_C \text{ is a fresh variable}\}$, $F_s^P = \{(\neg l_C) \mid C \in F_s\}$, and $w^P((\neg l_C)) = w(C)$. The following observations hold.*

1. *$\textsc{cost}(F) = \textsc{cost}(F^P)$, and the optimal solutions of F are the same as the optimal solutions of F^P restricted to $\textsc{var}(F)$.*
2. *For $\mathcal{A} \in \{HS, CG\}$, there is a one-to-one mapping between the $\mathcal{A}$ core traces on F and F^P of equal length.*

5 Impact of Preprocessing on HS

We continue with formal proofs of our main results for HS. An essential intuition for these proofs is that HS only extracts cores of the original instance. In other words, an HS core trace on any F only contains cores of the original instance F.

We first analyze best-case performance. The first observation shows that preprocessing does not affect the lengths of HS MUS traces in a significant way.

Observation 2 *For any MaxSAT instance F, $\mathsf{minlen}(HS^{\mathsf{mus}}, F) = \mathsf{minlen}(HS^{\mathsf{mus}}_{\mathsf{pre}}, F)$.*

Proof. (Sketch) By Corollary 1 we obtain $\kappa \in \mathsf{mus}(F)$ iff $\kappa \in \mathsf{mus}(\mathsf{pre}(F))$. The fact that an HS^{mus} trace on F only contains MUSes of F implies that T is an HS^{mus} trace on F iff it is an $\text{HS}^{\mathsf{mus}}_{\mathsf{pre}}$ trace on F. □

Next we show that executions of HS^{mus} are always shortest executions of HS.

Proposition 2. *For any MaxSAT instance F, $\mathit{minlen}(\mathit{HS}, F) \geq \mathit{minlen}(\mathit{HS}^{\mathsf{mus}}, F)$ and $\mathit{minlen}(\mathit{HS}_{\mathit{pre}}, F) \geq \mathit{minlen}(\mathit{HS}^{\mathsf{mus}}_{\mathit{pre}}, F)$.*

Proof. We will show that $\mathsf{minlen}(\text{HS}, F) \geq \mathsf{minlen}(\text{HS}^{\mathsf{mus}}, F)$ for any F, and thus $\mathsf{minlen}(\text{HS}_{\mathsf{pre}}, F) \geq \mathsf{minlen}(\text{HS}^{\mathsf{mus}}_{\mathsf{pre}}, F)$ as well. Let $T = (\kappa^1, \ldots, \kappa^n)$ be an arbitrary HS core trace on F. Let hs^* be a minimum-cost hitting set over $\{\kappa^1, \ldots, \kappa^n\}$ for which $F \setminus hs^*$ is satisfiable. The statement follows by constructing an HS^{mus} trace T_m on F s.t. $|T_m| \leq |T|$. As each $\kappa^i \in T$ is a core of F, all contain at least one MUS $m \subseteq \kappa^i$. Consider the set $\mathcal{M}$ of at most n MUSes of F constructed as follows. (1) $\mathcal{M}^1 = \{m^1\}$, where m^1 is any MUS contained in κ^1; (2) let $\mathcal{M}^i = \mathcal{M}^{i-1} \cup \{m^i\}$, where $m^i \subseteq \kappa^i$ is a MUS such that $m^i \notin \mathcal{M}^{i-1}$ if any exist, else let $\mathcal{M}^i = \mathcal{M}^{i-1}$. We obtain $\mathcal{M}^n = \mathcal{M}$ of size $|\mathcal{M}| = k \leq n$ such that each $m \in \mathcal{M}$ is a subset of some $\kappa^i \in T$.

We show that $\mathcal{M}$ can be ordered to form an HS^{mus} trace on F of length at most k, since if a minimum-cost hitting set hs over any proper subset $\mathcal{M}_s \subset \mathcal{M}$ hits all $m \in \mathcal{M}$, then hs^* is also a minimum-cost hitting set over $\mathcal{M}_s$, and HS^{mus} can terminate. As $F \setminus hs^*$ is satisfiable, hs^* is also a hitting set over $\mathcal{M}$ and over $\mathcal{M}_s$. Furthermore, as each $m \in \mathcal{M}$ is a subset of some $\kappa^i \in T$ and each $\kappa^i \in T$ contains a MUS in $\mathcal{M}$, hs^* is a minimum-cost hitting set of $\mathcal{M}$. Finally, as hs is a hitting set over $\mathcal{M}$ the cost of hs is not less than the cost of hs^*. Hence hs^* is a minimum-cost hitting set of $\mathcal{M}_s$, so the hitting set computation could have returned hs^*, thus allowing HS^{mus} to terminate. □

A simple corollary is that shortest executions of HS and HS_{pre} are of equal length.

Corollary 2 *For any MaxSAT instance F, $\mathit{minlen}(\mathit{HS}, F) = \mathit{minlen}(\mathit{HS}_{\mathit{pre}}, F)$.*

Proof. Observation 1 and Proposition 2 establish $\mathsf{minlen}(\text{HS}, F) = \mathsf{minlen}(\text{HS}^{\mathsf{mus}}, F)$ and $\mathsf{minlen}(\text{HS}_{\mathsf{pre}}, F) = \mathsf{minlen}(\text{HS}^{\mathsf{mus}}_{\mathsf{pre}}, F)$. Together with Observation 2 this implies $\mathsf{minlen}(\text{HS}, F) = \mathsf{minlen}(\text{HS}^{\mathsf{mus}}, F) = \mathsf{minlen}(\text{HS}^{\mathsf{mus}}_{\mathsf{pre}}, F) = \mathsf{minlen}(\text{HS}_{\mathsf{pre}}, F)$. □

We move on to the worst-case results. Corollary 1 can be used to show that valid executions of HS_{pre} are also valid executions of HS on any MaxSAT instance.

Observation 3 *For any MaxSAT instance F, $\mathit{maxlen}(\mathit{HS}_{\mathit{pre}}, F) \leq \mathit{maxlen}(\mathit{HS}, F)$.*

Proof. As $\mathsf{pre}(F)_s \subseteq F_s$ and any MUS of $\mathsf{pre}(F)$ is a MUS of F, any core of $\mathsf{pre}(F)$ is a core of F. □

Finally for this section, we prove the three $X \nrightarrow Y$ edges in Figure 4 for HS. For this, we need as a witness a family of MaxSAT instances $F(n)$ and a X core trace T on $F(n)$ s.t. $|T| > \mathsf{maxlen}(Y, F(n))$.

Proposition 3. *There is a family of MaxSAT instances $F(n)$ with $\mathcal{O}(n)$ soft clauses s.t. $\mathit{maxlen}(\mathit{HS}, F(n)) \geq n$ and $\mathit{maxlen}(\mathit{HS}^{\mathsf{mus}}, F(n)) = \mathit{maxlen}(\mathit{HS}_{\mathit{pre}}, F(n)) = 1$.*

Proof. Fix n and let $F(n)_h = \{(x \vee y)\} \cup \{(x \vee y \vee z_i) \mid i = 1, \ldots, n\}$ and $F(n)_s = \{(\neg x), (\neg y)\} \cup \{(\neg z_i) \mid i = 1, \ldots, n\}$ with $w((\neg x)) = w((\neg y)) = n$ and $w((\neg z_i)) = 1$ for all i. Now $\text{COST}(F(n)) = n$ and $\mathsf{mus}(F(n)) = \{\{(\neg x), (\neg y)\}\}$, explaining why $\mathsf{maxlen}(\text{HS}^{\mathsf{mus}}, F(n)) = 1$. A linear-length HS core trace on $F(n)$ is $(\kappa^1, \ldots, \kappa^n)$, where $\kappa^i = \{(\neg x), (\neg y), (\neg z_i)\}$. HS cannot terminate before extracting all n cores. To see this, consider an earlier iteration $i < n$. The weight of the hitting set $\{(\neg z_j) \mid j = 1, \ldots, i\}$ over $\mathcal{K}^i = \{\kappa^1, \ldots, \kappa^i\}$ is $i < n = w((\neg x)) = w((\neg y))$ and as such any minimum-cost hitting set over $\mathcal{K}^i$ can not contain $(\neg x)$ or $(\neg y)$, preventing HS from terminating. Hence $\mathsf{maxlen}(\text{HS}, F(n)) \geq n$.

However, due to the clause $(x \vee y)$, SE allows the removal of the clause $(x \vee y \vee z_i)$ for all i. Hence $\mathsf{pre}(F(n))$ has $\mathsf{pre}(F(n))_h = \{(x \vee y)\}$ and $\mathsf{pre}(F(n))_s = \{(\neg x), (\neg y)\}$. The only core of $\mathsf{pre}(F(n))$ is $\{(\neg x), (\neg y)\}$, and thus $\mathsf{maxlen}(\text{HS}_{\mathsf{pre}}, F(n)) = 1$. □

Proposition 4. *For any n, there is a family of MaxSAT instances $F(n)$ with $\mathcal{O}(n)$ soft clauses s.t. $\mathsf{maxlen}(HS_{\mathsf{pre}}, F(n)) \geq n$ and $\mathsf{maxlen}(HS^{\mathsf{mus}}_{\mathsf{pre}}, F) = 1$.*

Proof. Fix n and let

$$F(n)_h = \{(x_{1,2} \vee x_{1,3} \vee \neg x_{2,3}), (E \vee x_{2,3})\} \cup \tag{1}$$

$$\bigcup_{i=4}^{n+3} \{(x_{1,2} \vee x_{2,i} \vee \neg x_{1,i}), (x_{1,i} \vee x_{1,3} \vee \neg x_{3,i}), (x_{3,i} \vee x_{2,i} \vee \neg x_{2,3})\} \cup \tag{2}$$

$$\{(x_{T,x} \vee x_{x,y} \vee \neg x_{T,y}), (x_{T,x} \vee x_{T,y} \vee \neg x_{x,y}) \mid 1 \leq x, y \leq n+3\} \tag{3}$$

and $F(n)_s = \{(\neg x_{1,2}), (\neg x_{1,3}), (\neg E)\} \cup \{(\neg x_{2,i}) \mid i = 4, \ldots, n{+}3\}$ with $w((\neg x_{1,2})) = w((\neg x_{1,3})) = w((\neg E)) = n$ and $w((\neg x_{2,i})) = 1$ for all i. The hard clauses on row 3 are included in order to prevent preprocessing from simplifying $F(n)$ in any way. Intuitively, $F(n)$ encodes hard transitivity constraints over an undirected graph with each node having degree at least 4. Hence $\mathsf{pre}(F(n)) = F(n)$ at it suffices to show $\mathsf{maxlen}(\text{HS}, F(n)) \geq n$ and $\mathsf{maxlen}(\text{HS}^{\mathsf{mus}}, F) = 1$. Both arguments are similar to Proposition 3. As $\mathsf{mus}(F(n)) = \{\{(\neg x_{1,2}), (\neg x_{1,3}), (\neg E)\}\}$, it follows that $\mathsf{maxlen}(\text{HS}^{\mathsf{mus}}, F) = 1$. A linear-length HS core trace on $F(n)$ is $(\kappa^1, \ldots, \kappa^n)$, where $\kappa^i = \{(\neg x_{1,2}), (\neg x_{1,3}), (\neg E), (\neg x_{2,i+3})\}$. □

6 Impact of Preprocessing on CG

We start the analysis for CG by linking CG core traces with optimum cost.

Observation 4 *Let $T = (\kappa^1, \ldots, \kappa^n)$ be a CG or CG^{mus} core trace on a MaxSAT instance F, and $w^i = \min\{w(C^i) \mid C^i \in \kappa^i\}$. The cost of F is $\text{COST}(F) = \sum_{i=1}^{n} w^i$.*

An important corollary of Observation 4 is that no proper subsequence of a CG or CG^{mus} core trace on F can in itself be a CG or CG^{mus} trace on F.

The proofs on CG, in contrast to HS, need to consider the fact that the ith core κ^i in a CG core trace on F is not a core of F, but rather, of the working formula F^i instead. Following this, a relationship between the cores of F^i and the cores of F was derived in [53]. After necessary definitions and restatement of the result of [53], we will prove an analogous result regarding the relationship between the MUSes of F^i and F, which proves useful for obtaining our main results for CG.

6.1 Cores and MUSes of Working Formulas of CG

We follow here definitions from [53]. Let F be a MaxSAT instance and F^i the working formula of CG on iteration i when invoked on F. Let $\mathbf{card}^i$ be the set of all cardinality constraints added to F by CG during iterations $1, \ldots, i$. Thus the hard clauses of F^i are $F^i_h = F_h \cup \mathbf{card}^i$. We denote by $\mathbf{soln}(\mathbf{card}^i)$ the set of truth assignments satisfying $\mathbf{card}^i$ and not assigning any of the variables in F. Given any $\tau\colon \textsc{Var}(F) \to \{0,1\}$ and $\alpha \in \mathbf{soln}(\mathbf{card}^i)$, $(\tau{:}\alpha)$ is the truth assignment over the variables of F^i that assigns all variables of F according to τ and the rest according to α; $(\tau{:}\alpha)$ is well-defined as the auxiliary cardinality constraints are not allowed to mention variables in F. For any $\beta \in \mathbf{soln}(\mathbf{card}^i)$ and $S^i \subseteq F^i_s$, the *reduction* of S^i wrt β, $S^i|_\beta$ is obtained by (1) removing from S^i all clauses satisfied by β; (2) removing from each remaining clause $C^i \in S^i$ all blocking variables, i.e., all literals falsified by β; and (3) setting the weights of each $C^i \in S^i$ back to their original weights in F (removing duplicates). The restriction $\mathrm{R}(C^i) \in F_s$ of a soft clause $C^i \in F^i_s$ is obtained by (1) removing all added blocking variables from C^i; (2) removing all clones of C^i from the instance; and (3) setting the weight of C^i back to its original weight in F. Restriction is lifted to a set $S^i \subseteq F^i_s$ by $\mathrm{R}(S^i) = \{\mathrm{R}(C^i) \mid C^i \in S^i\}$. Notice that $S^i|_\beta \subseteq \mathrm{R}(S^i) \subseteq F_s$. With these definitions we can now restate a central result from [53].

Theorem 2 (Adapted from [53]).
A set $\kappa^i \subseteq F^i_s$ is a core of F^i iff $\kappa^i|_\beta$ is a core of F for all $\beta \in \mathbf{soln}(\mathbf{card}^i)$.

We will now prove an analogous characterization of the MUSes of F^i.

Theorem 3. *A set $M^i \subseteq F^i_s$ is a MUS of F^i iff there is a collection $\Upsilon \subseteq \mathsf{mus}(F)$ s.t.*

1. $\mathrm{R}(M^i) = \bigcup_{M \in \Upsilon} M$;
2. *for each $M \in \Upsilon$, there is an $\alpha \in \mathbf{soln}(\mathbf{card}^i)$ s.t. $M \subseteq M^i|_\alpha$ and $M' \not\subseteq M^i|_\alpha$ for all other $M' \in \Upsilon$; and*
3. *for each $\alpha \in \mathbf{soln}(\mathbf{card}^i)$, there is an $M \in \Upsilon$ s.t. $M \subseteq M^i|_\alpha$.*

Note that condition 3 is equivalent to the requirement of Theorem 2 for the set M^i being a core of F^i, since $M^i|_\alpha \subseteq \mathrm{R}(M^i)$ and $M^i|_\alpha$ should be unsatisfiable for all α.

Before proving Theorem 3, consider the following example for more intuition.

Example 3. Consider the unweighted MaxSAT instance $F = (F_h, F_s)$ with $F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7), (x_8)\}$ and $F_s = \cup_{i=1}^{8}\{(\neg x_i)\}$. Invoke WPM1 [50] on F and assume that it first processes the core $\{(\neg x_3), (\neg x_4), (\neg x_5)\}$. Afterwards the working formula F^2 is $F^2_h = F_h \cup \{\mathrm{CNF}(r_1 + r_2 + r_3 = 1)\}$ and $F^2_s = \{(\neg x_1), (\neg x_2), (\neg x_3 \vee r_1), (\neg x_4 \vee r_2), (\neg x_5 \vee r_3), (\neg x_6), (\neg x_7)(\neg x_8)\}$. Now $\mathbf{card}^2 = \{\mathrm{CNF}(r_1 + r_2 + r_3 = 1)\}$ and the set $\mathbf{soln}(\mathbf{card}^2)$ contains three assignments α^i, $i = 1, \ldots, 3$, assigning r_i to 1 and the others to 0. By Theorem 2, the set $\kappa^2 = \{(\neg x_1), (\neg x_2), (\neg x_3 \vee r_1), (\neg x_5 \vee r_3), (\neg x_6), (\neg x_7)\}$ is a core of F^2 as each $\kappa^2|_{\alpha^i}$ is a core of F. For example, $\kappa^2|_{\alpha^1} = \{(\neg x_1), (\neg x_2), (\neg x_5), (\neg x_6), (\neg x_7)\}$. In order to use Theorem 3 to show that κ^2 is also a MUS of F^2, note that $\mathrm{R}(\kappa^2) = \{(\neg x_1), (\neg x_2), (\neg x_3), (\neg x_5), (\neg x_6), (\neg x_7)\} = \{(\neg x_1), (\neg x_2), (\neg x_3)\} \cup \{(\neg x_5), (\neg x_6), (\neg x_7)\}$, where $\{(\neg x_1), (\neg x_2), (\neg x_3)\}$ and $\{(\neg x_5), (\neg x_6), (\neg x_7)\}$ are MUSes of F. Condition 2 of Theorem 3 follows since the only MUS in $\kappa^2|_{\alpha^3}$ is $\{(\neg x_1), (\neg x_2), (\neg x_3)\}$ and the only MUS in $\kappa^2|_{\alpha^1}$ is $\{(\neg x_5), (\neg x_6), (\neg x_7)\}$.

Next we prove Theorem 3. We begin by some lemmas. Assume for each of them that CG is invoked on an instance F and that F^i is the working formula on iteration i.

Lemma 1. *Let M^i be a MUS of F^i and $C^i \in M^i$. There is an $\alpha \in \mathbf{soln}(\mathbf{card}^i)$ s.t. $\mathrm{R}(C^i)$ is necessary for $M^i|_\alpha$.*

Proof. By Theorem 2, $M^i|_{\alpha'}$ is a core of F for all $\alpha' \in \mathbf{soln}(\mathbf{card}^i)$. Hence it suffices to show that $M^i|_\alpha \setminus \mathrm{R}(C^i)$ is not a core for some α. Consider the assignment $(\tau{:}\alpha)$ satisfying $F_h^i \wedge (M^i \setminus \{C^i\})$, guaranteed to exist as M^i is a MUS of F^i. Now τ satisfies $F_h \wedge (M^i \setminus \{C^i\})|_\alpha = F_h \wedge (M^i|_\alpha \setminus \mathrm{R}(C^i))$ as required. □

Corollary 3 *For any MUS M^i of F^i, $\mathrm{R}(M^i) \subseteq \bigcup \mathsf{mus}(F)$.*

Corollary 4 *For any MUS M^i of F^i, there is an irreducible $\Upsilon \subseteq \mathsf{mus}(F)$ s.t. $\mathrm{R}(M^i) = \bigcup_{M \in \Upsilon} M$.*

Proof. Take Υ as the smallest collection of MUSes of F for which $\mathrm{R}(M^i) \subseteq \bigcup_{M \in \Upsilon} M$; by Corollary 3 such a collection exists. We claim that $\bigcup_{M \in \Upsilon} M \subseteq \mathrm{R}(M^i)$, from which irreducibility follows directly by minimality of Υ. Fix an arbitrary $C_e \in M$ in some $M \in \Upsilon$. By minimality of Υ, there is a clause $C^i \in M^i$ for which the only MUS of Υ containing $\mathrm{R}(C^i)$ is M. By Lemma 1, there exists a β for which $\mathrm{R}(C^i)$ is necessary for $M^i|_\beta$. As $M^i|_\beta \subseteq \mathrm{R}(M^i) \subseteq \bigcup_{M \in \Upsilon} M$ and the only MUS in Υ containing $\mathrm{R}(C^i)$ is M, we have $C_e \in M \subseteq M^i|_\beta \subseteq \mathrm{R}(M^i)$, establishing $C_e \in \mathrm{R}(M^i)$ and $\bigcup_{M \in \Upsilon} M \subseteq \mathrm{R}(M^i)$. □

We are now ready to prove Theorem 3.

Proof (of Theorem 3). A collection $\Upsilon \subseteq \mathsf{mus}(F)$ satisfying condition 1 exists by Corollary 4. For condition 2, we use the fact that the set Υ is irreducible. Let $M \in \Upsilon$ be arbitrary. Similarly to the proof of Corollary 4, we can find a $C^i \in M^i \in \Upsilon$ and $\alpha \in \mathbf{soln}(\mathbf{card}^i)$ s.t $\mathrm{R}(C^i) \notin M'$ for any other $M' \in \Upsilon$ and $\mathrm{R}(C^i)$ is necessary for $M^i|_\alpha$, implying that the only MUS in $M^i|_\alpha$ is M. Finally, condition 3 follows from M^i being a core of F^i and Theorem 2.

What remains is to show that subset $M^i \subseteq F_s^i$ satisfying conditions 1-3 is a MUS of F^i. By condition 3 and Theorem 2, M^i is a core of F^i. Hence we only need to show that it is minimally unsatisfiable, i.e., $F_h^i \wedge (M^i \setminus \{C^i\})$ is satisfiable for all $C^i \in M^i$. Fix $C^i \in M^i$ and let Υ be the collection of MUSes of F for which $\mathrm{R}(M^i) = \bigcup_{M \in \Upsilon} M$. Consider any MUS $M_C \in \Upsilon$ s.t. $\mathrm{R}(C^i) \in M_C$. By condition 2, there is an $\alpha \in \mathbf{soln}(\mathbf{card}^i)$ for which the only MUS (of F) in $M^i|_\alpha \subseteq \mathrm{R}(M^i)$ is M_C. For such α, $F_h \wedge M^i|_\alpha \setminus \{\mathrm{R}(C_i)\}$ is satisfied by some τ. Hence $(\tau{:}\alpha)$ satisfies $F_h \wedge \mathbf{card}^i \wedge (M^i \setminus \{C^i\}) = F_h^i \wedge (M^i \setminus \{C^i\})$. □

Finally, we note that each condition in Theorem 3 is necessary.

Example 4. Consider again the MaxSAT instance F from Example 3. The set $\{(\neg x_1), (\neg x_2), (\neg x_3 \vee r_1)\}$ is an example of a non-MUS of F^1 satisfying conditions 1-2 and the set $\{(\neg x_1), (\neg x_2), (\neg x_3 \vee r_1), (\neg x_5 \vee r_3), (\neg x_6), (\neg x_7), (\neg x_8)\}$ is an example of a non-MUS of F^1 satisfying conditions 1 and 3.

6.2 Results on Core Trace Lengths

We proceed with proofs on the number of iterations for CG. With respect to best-case, preprocessing does not affect the lengths of CG$^{\mathsf{mus}}$ traces significantly.

Proposition 5. *For any MaxSAT instance F,* $\mathit{minlen}(\mathit{CG}^{\mathsf{mus}}, F) = \mathit{minlen}(\mathit{CG}^{\mathsf{mus}}_{\mathit{pre}}, F)$.

Proof. We show that a $T_m = (m^1, \ldots, m^n)$ is a CG$^{\mathsf{mus}}$ trace on F iff it is a CG$^{\mathsf{mus}}_{\mathsf{pre}}$ trace on F. We prove the left-to-right direction, the other is similar. We will show that there is an execution of CG$^{\mathsf{mus}}_{\mathsf{pre}}$ on F for which the ith MUS extracted is m^i and which terminates only after extracting all MUSes of T_m. The termination follows from no proper subset of a CG$^{\mathsf{mus}}$ trace being a core trace in itself.

We show that each m^i is a MUS of $\mathsf{pre}(F)^i$ by induction. By Corollary 1, m^1 is a MUS of $\mathsf{pre}(F)$. Assume that CG$^{\mathsf{mus}}$ has extracted and processed the MUSes $(m^1, \ldots, m^{i-1})$ from $\mathsf{pre}(F)$ and consider the ith iteration. As m^i is a MUS of F^i, by Theorem 3 there is an $\Upsilon \subseteq \mathsf{mus}(F)$ s.t. $\mathrm{R}(m^i) = \cup_{m \in \Upsilon} m$. For $m^i \in \mathsf{mus}(\mathsf{pre}(F)^i)$, we show that Υ satisfies the conditions of Theorem 3 in $\mathsf{pre}(F)$ as well. By Corollary 1, each $m \in \Upsilon$ is a MUS of $\mathsf{pre}(F)$. For the other two conditions, note that by induction, the set of cardinality constraints $\mathbf{card}^i_p$ added to $\mathsf{pre}(F)$ after processing the MUSes $m^1, \ldots, m^{i-1}$ is the same as the set $\mathbf{card}^i$ added to F after processing the same sequence of MUSes. Hence $\alpha \in \mathbf{soln}(\mathbf{card}^i_p)$ iff $\alpha \in \mathbf{soln}(\mathbf{card}^i)$, which implies the two other conditions of Theorem 3. □

Next we show that some shortest execution of CG is also an execution of CG$^{\mathsf{mus}}$.

Proposition 6. *For any MaxSAT instance F,* $\mathit{minlen}(\mathit{CG}^{\mathsf{mus}}, F) \leq \mathit{minlen}(\mathit{CG}, F)$ *and* $\mathit{minlen}(\mathit{CG}^{\mathsf{mus}}_{\mathit{pre}}, F) \leq \mathit{minlen}(\mathit{CG}_{\mathit{pre}}, F)$.

Proof. (Sketch) We prove $\mathsf{minlen}(\mathrm{CG}^{\mathsf{mus}}, F) \leq \mathsf{minlen}(\mathrm{CG}, F)$; the same proof works for $\mathsf{minlen}(\mathrm{CG}^{\mathsf{mus}}_{\mathsf{pre}}, F) \leq \mathsf{minlen}(\mathrm{CG}_{\mathsf{pre}}, F)$ as well. Let $T = (\kappa^1, \ldots, \kappa^n)$ be a CG trace on F. We construct a CG$^{\mathsf{mus}}$ trace $T_m = (m^1, \ldots, m^k)$ on F of at most the same length recursively. For intuition, on each iteration i CG$^{\mathsf{mus}}$ processes a subset of the clauses CG would have processed on the ith iteration of the execution corresponding to T. Hence, if $\mathbf{card}^i_m$ and $\mathbf{card}^i$ are the set of cardinality constraints added to F by the ith iteration on the execution corresponding to T_m and T, respectively, then any $\alpha \in \mathbf{soln}(\mathbf{card}^i_m)$ can be extended to a solution to $\mathbf{card}^i$ by assigning the remaining variables to 0.

Let m^1 be an MUS of F contained in κ^1. Assume that CG$^{\mathsf{mus}}$ has extracted the MUSes m^j for $j = 1, \ldots, i-1$ s.t each $m^j \subseteq \kappa^j$. Consider the ith iteration and the current working formula F^i_m. As κ^i is a core of F^i, the ith working formula on the execution corresponding to T, by Theorem 2 $\kappa^i|_\beta$ is a core of F for all $\beta \in \mathbf{soln}(\mathbf{card}^i)$. Hence $\kappa^i|_\beta$ is also a core of F for all $\beta \in \mathbf{card}^i_m$. Applying Theorem 2 gives that κ^i is a core of F^i_m. Hence it also contains a MUS m^i of F^i_m. For termination of CG$^{\mathsf{mus}}$, note that $\min_{C^i \in \kappa^i}\{w(C^i)\} \leq \min_{C^i \in m^i}\{w(C^i)\}$ for every i. Since $\sum_{i=1}^{n} \min_{C^i \in \kappa^i}\{w(C^i)\} = \textsc{cost}(F)$, termination of CG$^{\mathsf{mus}}$ occurs at the latest after n iterations on the execution corresponding T_m. □

Finally, we show that the shortest executions of CG and CG$_{\mathsf{pre}}$ are of the same length.

Corollary 5 *For any MaxSAT instance F,* $\mathsf{minlen}(CG, F) = \mathsf{minlen}(CG_{pre}, F)$.

Proof. Proposition 6 and Observation 1 imply $\mathsf{minlen}(\text{CG}, F) = \mathsf{minlen}(\text{CG}^{\mathsf{mus}}, F)$ and $\mathsf{minlen}(\text{CG}^{\mathsf{mus}}_{\mathsf{pre}}, F) = \mathsf{minlen}(\text{CG}_{\mathsf{pre}}, F)$. Together with Proposition 5 we obtain $\mathsf{minlen}(\text{CG}, F) = \mathsf{minlen}(\text{CG}^{\mathsf{mus}}, F) = \mathsf{minlen}(\text{CG}^{\mathsf{mus}}_{\mathsf{pre}}, F) = \mathsf{minlen}(\text{CG}_{\mathsf{pre}}, F)$. □

We move on to worst-case results for CG. We begin by showing that valid executions of CG_{pre} are also valid executions of CG.

Proposition 7. *For any MaxSAT instance F,* $\mathsf{maxlen}(CG, F) \geq \mathsf{maxlen}(CG_{pre}, F)$.

Proof. We show that a CG_{pre} trace $T = (\kappa^1, \ldots, \kappa^n)$ on F is also a CG trace on F. The termination of CG only after n iterations follows from the cost-preserving properties of preprocessing and Observation 4. We show that each κ^i is a valid core of F^i by induction. The case $i = 1$ follows from $\mathsf{pre}(F)_s \subseteq F_s$ and Corollary 1. Assume next that all κ^j for $j < i$ have been cores of F^j and consider κ^i. By Theorem 2, $\kappa^i|_\beta$ is a core of $\mathsf{pre}(F)$ for all $\beta \in \mathbf{soln}(\mathbf{card}^i_p)$, where $\mathbf{card}^i_p$ is the set of cardinality constraints added to $\mathsf{pre}(F)$ after processing cores $\kappa^1, \ldots, \kappa^{i-1}$. By induction, this set is exactly the same as set of cardinality constraints $\mathbf{card}^i$ added to F after processing the same cores. As any core of $\mathsf{pre}(F)$ is a core of F, it follows that $\kappa^i|_\beta$ is a core of F for all $\beta \in \mathbf{soln}(\mathbf{card}^i)$. We conclude that κ^i is a core of F^i. □

Finally, two families of instances witness the $\nrightarrow$ edges in Figure 4 for CG.

Proposition 8. *There is a family of MaxSAT instances $F(n)$ with $\mathcal{O}(n)$ soft clauses s.t.* $\mathsf{maxlen}(CG, F(n)) \geq n$ *and* $\mathsf{maxlen}(CG^{\mathsf{mus}}, F(n)) = \mathsf{maxlen}(CG_{pre}, F(n)) = 1$.

Proof. (Sketch) Consider again the instance $F(n)$ constructed in the proof of Proposition 3. We showed that $\mathsf{maxlen}(\text{HS}^{\mathsf{mus}}, F) = \mathsf{maxlen}(\text{HS}_{\mathsf{pre}}, F) = 1$. This also holds for CG. A linear-length CG core trace $(\kappa^1, \ldots, \kappa^n)$, on F can be constructed iteratively as follows: $\kappa^1 = \{(\neg x), (\neg y), (\neg z_1)\}$ and $\kappa^i = \{(\neg x)^c_{i-1}, (\neg y)^c_{i-1}, (\neg z_i)\}$ where $(\neg x)^c_{i-1}$ and $(\neg y)^c_{i-1}$ are duplicates of the original clauses added on iteration $i-1$. The existence of such duplicates for all n iterations follows from $w((\neg x)) = w((\neg y)) = n$ and $w((\neg z_i)) = 1$. The termination of CG after the nth iteration follows from Observation 4 as the smallest weight among the clauses in each κ^i is 1. □

Proposition 9. *There is a family of MaxSAT instances $F(n)$ with $\mathcal{O}(n)$ soft clauses s.t.* $\mathsf{maxlen}(CG_{pre}, F(n)) \geq n$ *and* $\mathsf{maxlen}(CG^{\mathsf{mus}}_{pre}, F) = 1$.

Proof. (Sketch) $F(n)$ is the same as for HS and the proof follows Proposition 4. A linear-length CG core trace can be constructed similarly to Proposition 8 by replacing clauses in the linear-length HS trace from Proposition 4 with duplicates of original clauses where required. □

7 Conclusions

We formally analyzed the effect of SAT-based preprocessing, as well as core minimization, on the performance of core-guided MaxSAT solvers. As a main result, we showed that SAT-based preprocessing has no effect on the best-case number of iterations required by the solvers but can improve on the worst-case. In terms of best-case performance, the potential benefits of applying SAT-based preprocessing in conjunction with core-guided MaxSAT solvers are thus in principle—assuming optimal search heuristics—solely in speeding up individual SAT solver calls made during MaxSAT search. Simultaneously, our analysis also revealed an analogous result on the impact of core minimization in core-guided MaxSAT solvers. Our results motivate further work on developing MaxSAT-specific preprocessing techniques capable of affecting the MaxSAT algorithms on a more general level. In contrast, SAT-based preprocessing does in cases have a positive effect on the worst-case number of iterations. Of independent interest, we established a formal characterization of how the underlying MUS structure is altered by iterative revisions performed by CG solvers on MaxSAT instances (Theorem 3), thus sharpening the main results of [53].

A Proof of Proposition 1

(1) If an optimal solution τ to F assigns $\tau(C) = 0$, then an optimal solution τ^P to F_P has to assign $F_P(l_C) = 1$. Similarly, if $\tau(C) = 1$, then τ^P can assign $\tau^P(l_C) = 0$.
(2) We sketch the conversion of an $\mathcal{A}$ core trace $T_P = (\kappa_P^1, \ldots, \kappa_P^n)$ on F_P into a core trace $T = (\kappa^1, \ldots, \kappa^n)$ on F, the other direction is similar. For $\mathcal{A} =$ HS, every κ_P^i is a core of F_P. The corresponding core trace of F is obtained by exchanging each $\kappa_P^i = \{(\neg l_{C_i}) \mid i = 1, \ldots, n\}$ with $\kappa^i = \{C_i \mid i = 1, \ldots, n\}$. Now κ_P^i is a core of F_P iff κ^i is a core of F. To see this, note that if κ^i is not a core of F, then it can be satisfied by some assignment τ. The same τ extended by setting all l_{C_i} variables to 0 to satisfies both κ_P^i and the hard clauses $\{C_1 \vee l_{C_1}, \ldots, C_n \vee l_{C_n}\}$. Hence κ_P^i is not a core of F_P either. A similar argument shows the other direction. Finally the termination of HS after n iterations follows by a similar argument showing that $F \setminus hs$ is satisfiable for some $hs = \{C_1, \ldots, C_i\}$ iff $F^P \setminus hs^P$ is satisfiable for $hs^P = \{(\neg l_{C_1}), \ldots, (\neg l_{C_i})\}$. Hence the trace $T = (\kappa^1, \ldots, \kappa^n)$ is a HS trace on F of the same length as T_P.

For $\mathcal{A} =$ CG the argument is similar but inductive. To form a CG trace T on F, every occurrence of a $(\neg l_{C_i})$ in a clause $C^i \in \kappa_P^i$ is replaced by C_i to form a core κ^i of F^i. For $i > 0$, each such C^i may have been augmented with blocking variables, i.e., $C^i = (\neg l_{C_i} \vee \bigvee b)$ for some set of blocking variables. However, the substitution $(\neg l_{C_i} \vee \bigvee b) \rightarrow C_i \vee \bigvee b$ is still valid as, by induction, if CG adds $\bigvee b$ to $(\neg l_{C_i})$ on the execution corresponding to T_P, then it also adds $\bigvee b$ to C_i on the execution corresponding to T. □

References

1. Park, J.D.: Using weighted MAX-SAT engines to solve MPE. In: Proc. AAAI, AAAI Press / The MIT Press (2002) 682–687

2. Chen, Y., Safarpour, S., Veneris, A.G., Marques-Silva, J.P.: Spatial and temporal design debug using partial MaxSAT. In: Proc. 19th ACM Great Lakes Symposium on VLSI, ACM (2009) 345–350
3. Chen, Y., Safarpour, S., Marques-Silva, J., Veneris, A.G.: Automated design debugging with maximum satisfiability. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **29**(11) (2010) 1804–1817
4. Argelich, J., Berre, D.L., Lynce, I., Marques-Silva, J.P., Rapicault, P.: Solving linux upgradeability problems using boolean optimization. In: Proc. LoCoCo. Volume 29 of EPTCS. (2010) 11–22
5. Lynce, I., Marques-Silva, J.: Restoring CSP satisfiability with MaxSAT. Fundam. Inform. **107**(2-3) (2011) 249–266
6. Zhu, C., Weissenbacher, G., Malik, S.: Post-silicon fault localisation using maximum satisfiability and backbones. In: Proc. FMCAD, FMCAD Inc. (2011) 63–66
7. Jose, M., Majumdar, R.: Cause clue clauses: error localization using maximum satisfiability. In: Proc. PLDI, ACM (2011) 437–446
8. Morgado, A., Liffiton, M.H., Marques-Silva, J.: MaxSAT-based MCS enumeration. In: Revised Selected Papers of HVC 2012. Volume 7857 of Lecture Notes in Computer Science., Springer (2013) 86–101
9. Guerra, J., Lynce, I.: Reasoning over biological networks using maximum satisfiability. In: Proc. CP. Volume 7514 of Lecture Notes in Computer Science., Springer (2012) 941–956
10. Zhang, L., Bacchus, F.: MAXSAT heuristics for cost optimal planning. In: Proc. AAAI, AAAI Press (2012)
11. Ansótegui, C., Izquierdo, I., Manyà, F., Torres-Jiménez, J.: A Max-SAT-based approach to constructing optimal covering arrays. In: Proc. CCIA. Volume 256 of Frontiers in Artificial Intelligence and Applications., IOS Press (2013) 51–59
12. Ignatiev, A., Janota, M., Marques-Silva, J.: Towards efficient optimization in package management systems. In: Proc. ICSE, ACM (2014) 745–755
13. Berg, J., Järvisalo, M., Malone, B.: Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In: Proc. AISTATS. Volume 33 of JMLR Workshop and Conference Proceedings., JMLR.org (2014) 86–95
14. Fang, Z., Li, C., Qiao, K., Feng, X., Xu, K.: Solving maximum weight clique using maximum satisfiability reasoning. In: Proc. ECAI. Volume 263 of Frontiers in Artificial Intelligence and Applications., IOS Press (2014) 303–308
15. Berg, J., Järvisalo, M.: SAT-based approaches to treewidth computation: An evaluation. In: Proc. ICTAI, IEEE Computer Society (2014) 328–335
16. Marques-Silva, J., Janota, M., Ignatiev, A., Morgado, A.: Efficient model based diagnosis with maximum satisfiability. In: Proc. IJCAI, AAAI Press (2015) 1966–1972
17. Berg, J., Järvisalo, M.: Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. Artificial Intelligence (2015) in press.
18. Wallner, J.P., Niskanen, A., Järvisalo, M.: Complexity results and algorithms for extension enforcement in abstract argumentation. In: Proc. AAAI, AAAI Press (2016)
19. Li, C., Manyà, F.: MaxSAT, hard and soft constraints. In: Handbook of Satisfiability. IOS Press (2009) 613–631
20. Ansótegui, C., Bonet, M., Levy, J.: SAT-based MaxSAT algorithms. Artificial Intelligence **196** (2013) 77–105
21. Morgado, A., Heras, F., Liffiton, M., Planes, J., Marques-Silva, J.: Iterative and core-guided MaxSAT solving: A survey and assessment. Constraints **18**(4) (2013) 478–534
22. Cook, S.A.: The complexity of theorem-proving procedures. In: Proc. STOC, ACM (1971) 151–158

23. Biere, A., Biere, A., Heule, M., van Maaren, H., Walsh, T.: Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, The Netherlands, The Netherlands (2009)
24. Koshimura, M., Zhang, T., Fujita, H., Hasegawa, R.: QMaxSAT: A partial Max-SAT solver. Journal of Satisfiability, Boolean Modeling and Computation **8**(1/2) (2012) 95–100
25. Davies, J., Bacchus, F.: Exploiting the power of MIP solvers in MaxSAT. In: Proc. SAT. Volume 7962 of Lecture Notes in Computer Science., Springer (2013) 166–181
26. Belov, A., Morgado, A., Marques-Silva, J.: SAT-based preprocessing for MaxSAT. In: Proc. LPAR-19. Volume 8312 of Lecture Notes in Computer Science., Springer (2013) 96–111
27. Martins, R., Manquinho, V.M., Lynce, I.: Open-WBO: A modular MaxSAT solver. In: Proc. SAT. Volume 8561 of Lecture Notes in Computer Science., Springer (2014) 438–445
28. Narodytska, N., Bacchus, F.: Maximum satisfiability using core-guided MaxSAT resolution. In: Proc. AAAI, AAAI Press (2014) 2717–2723
29. Bjørner, N., Narodytska, N.: Maximum satisfiability using cores and correction sets. In: Proc. IJCAI, AAAI Press (2015) 246–252
30. Berg, J., Saikko, P., Järvisalo, M.: Improving the effectiveness of SAT-based preprocessing for MaxSAT. In: Proc. IJCAI, AAAI Press (2015) 239–245
31. Morgado, A., Dodaro, C., Marques-Silva, J.: Core-guided maxsat with soft cardinality constraints. In: Proc. CP. Volume 8656 of Lecture Notes in Computer Science., Springer (2014) 564–573
32. Ansótegui, C., Gabàs, J.: Solving (weighted) partial MaxSAT with ILP. In: Proc. CPAIOR. Volume 7874 of Lecture Notes in Computer Science., Springer (2013) 403–409
33. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Proc. SAT. Volume 3569 of Lecture Notes in Computer Science., Springer (2005) 61–75
34. Järvisalo, M., Heule, M., Biere, A.: Inprocessing rules. In: Proc. IJCAR. Volume 7364 of Lecture Notes in Computer Science., Springer (2012) 355–370
35. Lagniez, J.M., Marquis, P.: Preprocessing for propositional model counting. In: Proc. AAAI, AAAI Press (2014) 2688–2694
36. Li, C.M., Manyà, F., Mohamedou, N.O., Planes, J.: Exploiting cycle structures in Max-SAT. In: Proc. SAT. Volume 5584 of Lecture Notes in Computer Science., Springer (2009) 467–480
37. Argelich, J., Li, C.M., Manyà, F.: A preprocessor for Max-SAT solvers. In: Proc. SAT. Volume 4996 of Lecture Notes in Computer Science., Springer (2008) 15–20
38. Bonet, M.L., Levy, J., Manyà, F.: Resolution for Max-SAT. Artificial Intelligence **171**(8-9) (2007) 606–618
39. Heras, F., Marques-Silva, J.: Read-once resolution for unsatisfiability-based Max-SAT algorithms. In: Proc. IJCAI, AAAI Press (2011) 572–577
40. Berg, J., Saikko, P., Järvisalo, M.: Re-using auxiliary variables for maxsat preprocessing. In: Proc ICTAI, IEEE (2015) 813–820
41. Krentel, M.W.: The complexity of optimization problems. Journal of Computer and System Sciences **36**(3) (1988) 490 – 509
42. Heras, F., Morgado, A., Marques-Silva, J.: Core-guided binary search algorithms for maximum satisfiability. In: Proc. AAAI, AAAI Press (2011)
43. Ignatiev, A., Morgado, A., Manquinho, V.M., Lynce, I., Marques-Silva, J.: Progression in maximum satisfiability. In: ECAI 2014, IOS Press (2014) 453–458
44. Kullmann, O., Marques-Silva, J.: Computing maximal autarkies with few and simple oracle queries. CoRR **abs/1505.02371** (2015)
45. Janota, M., Marques-Silva, J.: On the query complexity of selecting minimal sets for monotone predicates. Artif. Intell. **233** (2016) 73–83

46. Ansótegui, C., Gabàs, J., Levy, J.: Exploiting subproblem optimization in SAT-based MaxSAT algorithms. J. Heuristics **22**(1) (2016) 1–53
47. Järvisalo, M., Biere, A., Heule, M.: Blocked clause elimination. In: Proc. TACAS. Volume 6015 of Lecture Notes in Computer Science., Springer (2010) 129–144
48. Belov, A., Järvisalo, M., Marques-Silva, J.: Formula preprocessing in MUS extraction. In: Proc. TACAS. Volume 7795 of Lecture Notes in Computer Science., Springer (2013) 108–123
49. Fu, Z., Malik, S.: On solving the partial MaxSAT problem. In: Proc. SAT. Volume 4121 of Lecture Notes in Computer Science., Springer (2006) 252–265
50. Manquinho, V.M., Marques-Silva, J.P., Planes, J.: Algorithms for weighted boolean optimization. In: Proc. SAT. Volume 5584 of Lecture Notes in Computer Science., Springer (2009) 495–508
51. Ansótegui, C., Bonet, M.L., Levy, J.: Solving (weighted) partial maxsat through satisfiability testing. In: Proc. SAT. Volume 5584 of Lecture Notes in Computer Science., Springer (2009) 427–440
52. Martins, R., Joshi, S., Manquinho, V., Lynce, I.: Incremental cardinality constraints for MaxSAT. In: Proc. CP. Volume 8656 of Lecture Notes in Computer Science., Springer (2014) 531–548
53. Bacchus, F., Narodytska, N.: Cores in core based MaxSat algorithms: An analysis. In: Proc. SAT. Volume 8561 of Lecture Notes in Computer Science., Springer (2014) 7–15
54. Davies, J., Bacchus, F.: Postponing optimization to speed up MAXSAT solving. In: Proc. CP. Volume 8124 of Lecture Notes in Computer Science., Springer (2013) 247–262
55. Saikko, P., Berg, J., Järvisalo, M.: LMHS: A SAT-IP hybrid MaxSAT solver. In Creignou, N., Berre, D.L., eds.: Proc. SAT. Volume 9710 of Lecture Notes in Computer Science., Springer (2016) 539–546