Subsumed Label Elimination for Maximum Satisfiability

Jeremias Berg and **Paul Saikko** and **Matti Järvisalo**[1]

Abstract. We propose *subsumed label elimination* (SLE), a so-called *label-based* preprocessing technique for the Boolean optimization paradigm of maximum satisfiability (MaxSAT). We formally show that SLE is orthogonal to previously proposed SAT-based preprocessing techniques for MaxSAT in that it can simplify the underlying minimal unsatisfiable core structure of MaxSAT instances. We also formally show that SLE can considerably reduce the number of internal SAT solver calls within modern core-guided MaxSAT solvers. Empirically, we show that combining SLE with SAT-based preprocessing improves the performance of various state-of-the-art MaxSAT solvers on standard industrial weighted partial MaxSAT benchmarks.

1 INTRODUCTION

Maximum satisfiability (MaxSAT), the optimization counterpart of Boolean satisfiability (SAT), is becoming a competitive approach to solving hard optimization problems due to recent advances in MaxSAT solving [2, 38]. As MaxSAT is finding an increasing number of applications in solving real-world optimization problems—ranging from, e.g., inconsistency analysis, diagnosis, design debugging, and fault localization [15, 14, 4, 32, 44, 30, 39, 27, 35] to further applications in AI, combinatorics, data analysis, and bioinformatics [41, 23, 43, 3, 10, 21, 8, 9, 42]—there is a high demand for new techniques for speeding up MaxSAT solving further.

This paper focuses on improving the efficiency of solving real-world MaxSAT instances via preprocessing the instances before calling a state-of-the-art MaxSAT solver. In particular, effective preprocessing techniques for MaxSAT have the promise of providing *solver-independent* speeds-up to overall solving times, similarly to SAT where preprocessing is today an integral part of the solving process [20, 29]. This motivates work on MaxSAT-level preprocessing, in hope of bridging the gap between highly successful SAT preprocessing and the currently less studied and understood role of preprocessing for MaxSAT [7, 11, 31, 5, 13].

One approach to MaxSAT preprocessing is to lift commonly applied SAT preprocessing techniques, such as bounded variable elimination [20], self-subsuming resolution, and forms of clause elimination [26], to MaxSAT. Direct applications of such SAT preprocessing techniques are not correct w.r.t. preserving the optimal solutions of MaxSAT instances [7]. However, correct liftings to MaxSAT are enabled by the so-called *labelled conjunctive normal form* (LCNF) representation [7, 6].

A natural next goal for MaxSAT preprocessing is to go beyond lifting well-known SAT preprocessing techniques, by developing novel MaxSAT-specific LCNF-level preprocessing techniques that can be applied in conjunction with SAT-based preprocessing techniques, ideally with orthogonal simplification properties. In this paper, we address this challenge by proposing *label-based preprocessing* as a form of native LCNF-level MaxSAT preprocessing. In particular, we propose the preprocessing technique of *subsumed label elimination* (SLE). The main aim of SLE is, working in conjunction with SAT-based preprocessing on labelled MaxSAT instances, to detect and eliminate *redundant labels*, i.e., auxiliary variables that are first added to maintain correctness under SAT-based preprocessing, but which can be inferred to be redundant by a simple polynomial-time deduction rule that SLE implements. Arising from deduction rules proposed in the nineties for the so-called binate covering problem [17, 16], a key insight of SLE is that redundant labels can be eliminated by comparing the label-sets L of clauses C^L on the LCNF level, i.e., *regardless of the contents of* C. While SLE is based on a relatively simple observation, it significantly differs from the earlier proposed SAT-based preprocessing techniques for MaxSAT. In practice it also tends to provide further speed-ups to the MaxSAT solving process for several state-of-the-art MaxSAT solvers.

In more detail, we analyze how known LCNF-lifted SAT preprocessing techniques and SLE modify key properties of MaxSAT instances: the (labelled) minimal unsatisfiable cores (LMUSes) and (labelled) minimal correction sets (LMCSes). We show that SLE is fundamentally different from LCNF-lifted SAT preprocessing. In contrast to SAT preprocessing which is unable to simplify LMUSes and LMCSes, SLE can effectively remove labels from LMUSes. Via a straightforward translation of LCNFs to standard MaxSAT, this implies that SLE can reduce the number of standard MUSes in the resulting MaxSAT instance. This can improve the performance of so-called *core-guided MaxSAT solvers*, such as [22, 25, 40, 36, 37], as well as those based on the implicit hitting set approach [18, 19, 11]. Giving a concrete witnessing family of LCNF-MaxSAT instances, we show that SLE has the potential to drastically decrease the number of iterations performed by various core-guided MaxSAT solvers. Complementing the theoretical analysis, we show empirically that by combining SLE with LCNF-lifted SAT preprocessing, noticeably more labels (i.e. redundancies) are eliminated than without SLE on weighted partial MaxSAT instances of the industrial track of MaxSAT Evaluation 2015. Further, we show that the additional simplifications translate into runtime improvements for various state-of-the-art MaxSAT solvers on industrial weighted partial instances.

This paper is organized as follows. After preliminaries on labelled CNFs and SAT-based preprocessing for MaxSAT (Section 2), we detail subsumed label elimination (Section 3), and provide a theoretical analysis of SLE both in terms of its effects on the core structure of MaxSAT instances (Section 4) and its potential to speed-up MaxSAT solving (Section 5). Empirical results on simplifications provided by SLE and the impact of SLE on the performance of MaxSAT solvers are provided in Section 6.

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

2 PRELIMINARIES

Throughout this paper, we work with *labelled CNFs* (LCNFs) [7, 6] which allow for generalizing MaxSAT and provide a convenient formalism for describing correct liftings of SAT preprocessing techniques to MaxSAT. For an intuitive reading, in LCNF a set of *labels* is associated with each clause. An empty label-set denotes that the corresponding clause is hard, while a non-empty label-set implies that the corresponding clause is soft. Furthermore, key concepts such as maximum satisfiability, minimal unsatisfiable subsets and minimal correction sets, are defined over the *label-sets* L of LCNF clauses C^L instead of the clauses C.

Before the formal definitions, consider the MaxSAT instance with three unweighted soft clauses shown in Figure 1 (1). As argued in [7], in order to apply e.g. bounded variable elimination (VE) [20] and still maintain the set of optimal solutions, each soft clause C_i needs to be attached an auxiliary fresh variable $\mathbf{l_i}$, resulting in the instance (Figure 1, 2a). On the level of LCNFs [6], the resulting instance is shown in Figure 1 (2b). Restricting VE from eliminating any of the added variables allows for sound application of most SAT preprocessing techniques in terms of MaxSAT. As an example, first eliminating the variable x and then y gives (possibly among others; here $\bowtie_x$ denotes resolving on x) the clause shown in Figure 1 (3a). Notice how the original one-to-one mapping between the clauses and labels vanishes, as after VE a clause may contain multiple labels. To solve the MaxSAT instance after preprocessing, the clauses obtained by preprocessing are then considered hard, and for each $\mathbf{l_i}$ the unit soft clause $(\neg \mathbf{l_i})$ with weight inherited from C_i is added into the instance. On the LCNF level, *labelled* VE [7] results equivalently in the LCNF instance (3b), explicitly separating original variables and the labels in each of the clauses.

(1) MaxSAT instance:
$C_1 = (x \vee y \vee z)$, $C_2 = (\neg x \vee \neg a \vee y)$, $C_3 = (\neg y \vee \neg a \vee \neg b)$

(2a) After adding labels: $C_1 = (x \vee y \vee z \vee \mathbf{l_1})$ $C_2 = (\neg x \vee \neg a \vee y \vee \mathbf{l_2})$ $C_3 = (\neg y \vee \neg a \vee \neg b \vee \mathbf{l_3})$ $\dots$	(3a) After variable eliminating x and y: $((C_1 \bowtie_x C_2) \bowtie_y C_3)$ $= (\neg a \vee \neg b \vee z \vee \mathbf{l_1} \vee \mathbf{l_2} \vee \mathbf{l_3})$ $\dots$
(2b) LCNF representation: $C_1^{\{\mathbf{l_1}\}}$ $C_2^{\{\mathbf{l_2}\}}$ $C_3^{\{\mathbf{l_3}\}}$ $\dots$	(3b) After labelled variable elimination on x and y: $((C_1^{\{\mathbf{l_1}\}} \bowtie_x C_2^{\{\mathbf{l_2}\}}) \bowtie_y C_3^{\{\mathbf{l_3}\}})$ $= (\neg a \vee \neg b \vee z)^{\{\mathbf{l_1},\mathbf{l_2},\mathbf{l_3}\}}$ $\dots$

Figure 1: Example of SAT-based preprocessing on the CNF and LCNF level.

2.1 Labelled CNFs and MaxSAT

Assume a countable set Lbl of labels. A labelled clause C^L consists of a clause C and a (possibly empty) set $L \subseteq Lbl$ of labels. A LCNF formula Φ is a set of labelled clauses. $Cl(\Phi)$ and $Lbls(\Phi)$ denote the set of clauses and labels of Φ, respectively, and $LCl(\Phi, l) = \{C^L \mid C^L \in \Phi, l \in L\}$ the set of labelled clauses in Φ that have l in their label-set. A LCNF formula is satisfiable iff $Cl(\Phi)$ (a CNF formula) is satisfiable.

Given a LCNF formula Φ and a subset $M \subseteq Lbls(\Phi)$ of its labels, the subformula $\Phi|_M$ of Φ induced by M is $\{C^L \in \Phi \mid L \subseteq M\}$, i.e., the LCNF formula obtained by removing from Φ all labelled clauses with at least one label not in M; notice that $\Phi|_{Lbls(\Phi)\setminus M} = \{C^L \in \Phi \mid L \cap M = \emptyset\}$. The *removal* $\textsc{Remove}(\Phi, K)$ of the label-set $K \subseteq Lbls(\Phi)$ from Φ gives $\{C^{L\setminus K} \mid C^L \in \Phi\}$, i.e, the LCNF formula obtained by removing all labels from Φ that are in K (note that removal does not remove clauses).

A (labelled) *unsatisfiable core* of an unsatisfiable LCNF formula Φ is a label-set $L \subseteq Lbls(\Phi)$ such that $\Phi|_L$ is unsatisfiable. An unsatisfiable core L is minimal (a LMUS) iff $\Phi|_{L'}$ is satisfiable for all $L' \subset L$. We denote the set of minimal unsatisfiable cores of Φ by $\mathrm{LMUS}(\Phi)$. A (labelled) *minimal correction subset* (LMCS) of Φ is a label-set $R \subseteq Lbls(\Phi)$ such that (i) $\Phi|_{Lbls(\Phi)\setminus R}$ is satisfiable, and (ii) $\Phi|_{Lbls(\Phi)\setminus R'}$ is unsatisfiable for all $R' \subset R$. We denote the set of LMCSes of Φ by $\mathrm{LMCS}(\Phi)$. Hitting set duality, formalizing a connection between LMUSes and LMCSes, is useful in this work.

Theorem 1 (Hitting set duality [6]) *A label-set $R \subseteq Lbls(\Phi)$ of a LCNF formula Φ is a LMCS of Φ iff R is an irreducible hitting set over $\mathrm{LMUS}(\Phi)$, i.e., iff R is a hitting set over $\mathrm{LMUS}(\Phi)$ and no $R' \subset R$ is a hitting set of $\mathrm{LMUS}(\Phi)$.*

A LCNF-MaxSAT instance consists of a LCNF formula Φ, and a weight function $w\colon Lbls(\Phi) \to \mathbb{N}$ assigning a positive weight $w(l)$ to each label $l \in Lbls(\Phi)$. The cost of a label-set $L \subseteq Lbls(\Phi)$ is the sum of the weights of the labels in L. Given a LCNF-MaxSAT instance Φ such that $\Phi|_\emptyset$ is satisfiable, any assignment τ that satisfies $\Phi|_\emptyset$ is a solution to the LCNF-MaxSAT instance. A solution τ is optimal if it satisfies $\Phi|_{Lbls(\Phi)\setminus R}$ for some minimum-cost LMCS R of Φ. The cost of τ is the cost of R. We treat the MaxSAT problem for LCNFs as the problem of computing R. In the rest of the text we will always assume that solutions to (Φ, w) exist, i.e., that $\Phi|_\emptyset$ is satisfiable.

A (standard/non-labelled) MaxSAT instance $F = (F_h, F_s, w)$ consists of a set F_h of hard and a set F_s of soft clauses, together with a function $w\colon F_s \to \mathbb{N}$ assigning a positive weight $w(C)$ to each soft clause $C \in F_s$. A (standard) minimal correction set (MCS) of F is a subset-minimal subset of F_s whose removal from F_s makes the instance satisfiable. Similarly, a (standard) minimal unsatisfiable core (MUS) of F is a subset-minimal subset F'_s for which $F_h \cup F'_s$ is an unsatisfiable set of clauses. Given a non-labelled MaxSAT instance F, any truth assignment τ satisfying all hard clauses is a solution to the instance. A solution τ is optimal if the sum of the weights of the soft clauses τ satisfies is the maximum over all solutions. Notice that the soft clauses falsified by an optimal solution form a minimum-cost MCS of F.

A MaxSAT instance $F = (F_h, F_s, w)$ can be viewed as a LCNF-MaxSAT instance (Φ_F, w) by introducing (i) for each hard clause $C \in F_h$ the labelled clause $C^\emptyset$, and (ii) for each soft clause $C \in F_s$ the labelled clause $C^{\{l_C\}}$, where l_C is a distinct label for C with weight $w(l_C) = w(C)$. It is easy to see that any optimal solution to Φ_F is an optimal solution to F, and vice versa. An essential intuition is that LMCSes of (Φ_F, w) correspond exactly to the MCSes of (F_h, F_s, w) in that for any MCS $\{C_1, \dots, C_k\}$ there is a corresponding LMCS $\{l_{C_1}, \dots, l_{C_k}\}$ (and vice versa). Similarly, LMUSes of (Φ_F, w) correspond to MUSes of (F_h, F_s, w).

To the other direction, a LCNF-MaxSAT instance (Φ, w) can be viewed as a MaxSAT instance F_Φ by associating with each label $l_i \in Lbls(\Phi)$ a distinct variable a_i, and introducing (i) for each labelled clause $C^L \in \Phi$ a hard clause $C \vee \bigvee_{l_i \in L} a_i$, and (ii) for each $l_i \in Lbls(\Phi)$, a soft clause $(\neg a_i)$ with weight $w((\neg a_i)) = w(l_i)$, where $w(l_i)$ is the weight of the label l_i. Again, using this reduction, LMUSes and LMCSes of (Φ, w) correspond exactly to the MUSes

and MCSes of F_Φ. Importantly for this work, especially the discussion in Section 5, this reduction allows one to treat any standard MaxSAT solver as a LCNF-MaxSAT solver.

Example 2 *Consider the MaxSAT instance $F_{\text{ex}} = (F_h, F_s, w)$ with $w(C) = 1$ for all $C \in F_s$, $F_h = \{(x \vee y), (\neg t \vee \neg z), (\neg z \vee y), (\neg y \vee z), (z \vee t)\}$, and $F_s = \{(\neg x), (x), (y \vee t), (z \vee t \vee x)\}$. The assignment τ for which $\tau(t) = \tau(x) = 0$ and $\tau(y) = \tau(z) = 1$ is an optimal solution to F_{ex} with cost 1. The LCNF-MaxSAT instance $\Phi_{F_{\text{ex}}}$ corresponding to F_{ex} is*

$$\Phi_{F_{\text{ex}}} = \{(x \vee y)^{\emptyset}, (\neg t \vee \neg z)^{\emptyset}, (\neg z \vee y)^{\emptyset}, (\neg y \vee z)^{\emptyset}, (z \vee t)^{\emptyset}, (\neg x)^{\{l_1\}}, (x)^{\{l_2\}}, (y \vee t)^{\{l_3\}}, (z \vee t \vee x)^{\{l_4\}}\}$$

with $w(l_i) = 1$ for $i = 1..4$. Now $Cl(\Phi_{F_{\text{ex}}}) = F_h \cup F_s$ and $Lbls(\Phi_{F_{\text{ex}}}) = \{l_1, l_2, l_3, l_4\}$. The label-set $L = \{l_1, l_2\}$ is an LMUS of $\Phi_{F_{\text{ex}}}$ as

$$\Phi_{F_{\text{ex}}}|_L = \{(x \vee y)^{\emptyset}, (\neg t \vee \neg z)^{\emptyset}, (\neg z \vee y)^{\emptyset}, (\neg y \vee z)^{\emptyset}, (z \vee t)^{\emptyset}, (\neg x)^{\{l_1\}}, (x)^{\{l_2\}}\}$$

is unsatisfiable. The sets $R_1 = \{l_1\}$ and $R_2 = \{l_2\}$ are examples of (minimum-cost) LMCSes of $\Phi_{F_{\text{ex}}}$. The fact that τ is an optimal solution to the LCNF-MaxSAT instance $\Phi_{F_{\text{ex}}}$ can be verified by checking that τ satisfies $\Phi_{F_{\text{ex}}}|_{Lbls(\Phi_{F_{\text{ex}}}) \setminus R_2}$. Converting $\Phi_{F_{\text{ex}}}$ back to MaxSAT results in the instance $F' = (F'_h, F'_s, w)$ with

$$F'_h = \{(x \vee y), (\neg t \vee \neg z), (\neg z \vee y), (\neg y \vee z), (z \vee t), (\neg x \vee a_1), (x \vee a_2), (y \vee t \vee a_3), (z \vee t \vee x \vee a_4)\}$$
$$\text{and } F'_s = \{(\neg a_1), (\neg a_2), (\neg a_3), (\neg a_4)\}.$$

2.2 SAT-based Preprocessing for LCNFs

A motivation for viewing MaxSAT instances as LCNF in [7] was to develop sound applications of SAT preprocessing techniques for MaxSAT. Many important SAT preprocessing techniques, including bounded variable elimination (VE) [20], self-subsuming resolution (SSR), and subsumption elimination (SE), cannot be used directly on MaxSAT instances [7]. However, the techniques can be applied on LCNFs by taking into account the natural restrictions implied by the SAT-level techniques on the label-sets of labelled clauses. With this intuition, the following LCNF-liftings of VE, SSR, and SE were proposed [7].

- *LCNF-lifting of the resolution rule:* The resolvent of two labelled clauses $(x \vee A)^{L_1}$ and $(\neg x \vee B)^{L_2}$ w.r.t. x is $(x \vee A)^{L_1} \bowtie_x (\neg x \vee B)^{L_2} = (A \vee B)^{L_1 \cup L_2}$.
- *LCNF-lifting of VE (LVE):* Let Φ_x and $\Phi_{\neg x}$, resp., denote the sets of labelled clauses that contain the literal x and the literal $\neg x$, resp. LVE allows for replacing $\Phi_x \cup \Phi_{\neg x}$ with $\Phi_x \bowtie_x \Phi_{\neg x} = \{A^{L_1} \bowtie_x B^{L_2} \mid A^{L_1} \in \Phi_x, B^{L_2} \in \Phi_{\neg x}, A \vee B \text{ non-tautological}\}$ given that $|\Phi_x \bowtie_x \Phi_{\neg x}| \leq |\Phi_x \cup \Phi_{\neg x}|$.
- *LCNF-lifting of SE (LSE):* A labelled clause A^{L_1} subsumes B^{L_2} if $A \subseteq B$ *and* $L_1 \subseteq L_2$. LSE allows for removing subsumed clauses.
- *LCNF-lifting of SSR (LSSR):* Given labelled clauses $(l \vee A)^{L_1}$ and $(\neg l \vee B)^{L_2}$, if A^{L_1} subsumes B^{L_2}, LSSR allows for replacing $(\neg l \vee B)^{L_2}$ with B^{L_2}.

Blocked clause elimination (BCE) [28] is sound for MaxSAT [7], and could as such be directly applied on MaxSAT instances. However, for a uniform presentation, it makes sense to consider a straightforward lifting of BCE.

- *LCNF-lifting of BCE (LBCE):* A labelled clause C^L is *blocked* in Φ if C is blocked in $Cl(\Phi)$. LBCE allows for removing blocked clauses.

Example 3 *Consider the LCNF-MaxSAT instance $\Phi_{F_{\text{ex}}}$ from Example 2. Applying LSE to remove $(z \vee t \vee x)^{\{l_4\}}$ and LVE to eliminate x and t results in the formula*

$$\{(y)^{\{l_1\}}, (\neg z \vee y)^{\emptyset}, (\neg y \vee z)^{\emptyset}, ()^{\{l_1, l_2\}}, (y \vee \neg z)^{\{l_3\}}\}.$$

Removing $(y \vee \neg z)^{\{l_3\}}$ by LSE and eliminating z by LVE results in the preprocessed formula $\Phi^{pre}_{F_{\text{ex}}} = \{(y)^{\{l_1\}}, ()^{\{l_1, l_2\}}\}$.

LVE, LSSR, LSE, and LBCE are correct due to the following.

Proposition 4 ([7]) *Let Φ be a LCNF-MaxSAT instance and Φ^{pre} the LCNF-MaxSAT instance resulting from an application of LVE, LSSR, LSE, and LBCE on Φ. Then $\text{LMUS}(\Phi) = \text{LMUS}(\Phi^{pre})$ and, by Theorem 1, $\text{LMCS}(\Phi) = \text{LMCS}(\Phi^{pre})$.*

3 SUBSUMED LABEL ELIMINATION

We propose and analyze *subsumed label elimination* (SLE), a *label-based preprocessing* technique for MaxSAT. The primary goal of SLE is to provide further simplifications when applied in conjunction with SAT-based preprocessing; SLE focuses on removing labels from *non-singleton* label-sets (produced starting from non-labelled MaxSAT instances mainly by LVE). Before a formal definition of SLE, we begin with an example to illustrate some of the shortcomings of SAT-based preprocessing for MaxSAT that SLE seeks to address.

Example 5 *Consider the MaxSAT instance $F = (F_h, F_s, w)$ with $w(C) = 1$ for all $C \in F_s$ and*

$$F_h = \{(x \vee y)\} \text{ and } F_s = \{(\neg x), (\neg y)\}.$$

Converting F to LCNF gives the instance $\Phi_F = \{(x \vee y)^{\emptyset}, (\neg x)^{\{l_1\}}, (\neg y)^{\{l_2\}}\}$. Applying LVE to eliminate both x and y results in the LCNF-MaxSAT instance $pre(\Phi_F) = \{()^{\{l_1, l_2\}}\}$. Finally, converting $pre(\Phi_F)$ back to MaxSAT gives the MaxSAT instance $F' = (F'_h, F'_s, w)$ with

$$F'_h = \{(a_1 \vee a_2)\} \text{ and } F'_s = \{(\neg a_1), (\neg a_2)\},$$

i.e., the exact same instance as F modulo variable naming. In other words, LVE (or LSSR, LSE, and LBCE) is unable to simplify F. Furthermore, notice that F contains exactly one MUS: $\{(\neg x), (\neg y)\}$. As the clauses $(\neg x)$ and $(\neg y)$ occur in exactly the same MUSes, no optimal solution to F falsifies both of them. As an alternative view, no MCS of F contains both $(\neg x)$ and $(\neg y)$, which means that either clause could be hardened*, i.e., changed to a hard clause, without removing all of the optimal solutions of the instance. As we will see, SLE captures this simplification on the LCNF-level.*

More concretely, consider a LCNF-MaxSAT instance Φ. SLE is based on the following observation. Consider two labels $l_1, l_2 \in Lbls(\Phi)$ such that $w(l_1) \leq w(l_2)$, and l_1 appears in at least the same LMUSes of Φ as l_2. Then l_2 is redundant in that l_2 can be replaced by l_1 in any LMCS R of Φ without increasing the cost of R. Hence l_2 can be removed from Φ while maintaining at least one minimum-cost LMCS. This is more formally stated as Theorem 6.

Theorem 6 *Let $l_1, l_2 \in Lbls(\Phi)$ and $\Phi^{pre} = \text{REMOVE}(\Phi, \{l_2\})$. Assume that, for all $L \in \text{LMUS}(\Phi)$, $l_2 \in L$ implies $l_1 \in L$. Then $\emptyset \neq \text{LMCS}(\Phi^{pre}) \subseteq \text{LMCS}(\Phi)$.*

Proof. $\Phi^{pre}|_{Lbls(\Phi^{pre})\setminus R} = \Phi|_{Lbls(\Phi)\setminus R}$ for any label-set $R \subseteq Lbls(\Phi^{pre})$. Hence it suffices to show that there is an $R \in \text{LMCS}(\Phi)$ s.t. $R \subseteq Lbls(\Phi^{pre})$. This can be verified by viewing R as an irreducible hitting set of $\text{LMUS}(\Phi)$. If $R \not\subseteq Lbls(\Phi^{pre})$, then $l_2 \in R$. By assumption, $R' = (R \setminus \{l_2\}) \cup \{l_1\}$, a subset of $Lbls(\Phi^{pre})$, is also an irreducible hitting set of $\text{LMUS}(\Phi)$ and hence a LMCS of Φ. □

While the assumption in Theorem 6 is likely not checkable in polynomial time, a stricter, easier-to-check version of the assumption, formalized in Proposition 7, gives the basis for SLE. In words, let L be any label-set and $C^{L'}$ any labelled clause of Φ. If L' contains labels l_1 and l_2 such that $l_2 \in L$ but $l_1 \notin L$, then $C^{L'}$ is not a member of the formula $\Phi|_L$. This is specifically true for any LMUS of Φ.

Proposition 7 *Let $l_1, l_2 \in Lbls(\Phi)$ and $LCl(\Phi, l_2) \subseteq LCl(\Phi, l_1)$. Then, for all $L \in \text{LMUS}(\Phi)$, $l_2 \in L$ implies $l_1 \in L$.*

Proof. Let L be a label-set such that $l_2 \in L$ and $l_1 \notin L$. We show that L is not a LMUS of Φ. From the assumption $LCl(\Phi, l_2) \subseteq LCl(\Phi, l_1)$ it follows that, if $C^{L'}$ is a labelled clause for which $l_2 \in L'$, then $l_1 \in L'$. Thus $C^{L'} \notin \Phi|_L$, and hence $\Phi|_L = \Phi|_{L\setminus\{l_2\}}$. As such $L \notin \text{LMUS}(\Phi)$ as either $\Phi|_L$ is satisfiable or $\Phi|_{L_1}$ is unsatisfiable for $L_1 = L \setminus \{l_2\} \subset L$. □

The final part in the formalization of SLE ensures that the removal of l_2 preserves at least one *minimum-cost* LMCS of the instance. This follows by adding an assumption on the weights of l_1 and l_2.

Proposition 8 *Let $l_1, l_2 \in Lbls(\Phi)$ and $\Phi^{pre} = \text{REMOVE}(\Phi, \{l_2\})$. Assume that, for all $L \in \text{LMUS}(\Phi)$, $l_2 \in L$ implies $l_1 \in L$, and $w(l_1) \leq w(l_2)$. Then all minimum-cost LMCSes of Φ^{pre} are also minimum-cost LMCSes of Φ.*

Proof. Following the proof of Theorem 6 let $R' = (R \setminus \{l_2\}) \cup \{l_1\}$ be the LMCS of Φ constructed in order to replace the LMCS $R \not\subseteq Lbls(\Phi^{pre})$. The extra assumption on the weights guarantees that the cost of R' is not higher than the cost of R. □

Putting these results together gives SLE. Informally, SLE removes subsumed labels l_2, or, more formally, converts Φ into $\text{REMOVE}(\Phi, \{l_2\})$.

Definition 9 (Subsumed Label Elimination (SLE)) *Let Φ be a LCNF-MaxSAT instance and $l_1, l_2 \in Lbls(\Phi)$. We say that l_1 subsumes l_2 if (i) $LCl(\Phi, l_2) \subseteq LCl(\Phi, l_1)$, and (ii) $w(l_1) \leq w(l_2)$. SLE allows for removing subsumed labels from LCNF-MaxSAT instances.*

Example 10 *Consider the LCNF-MaxSAT instance*

$$\begin{aligned}\Phi =& \{(x_i \vee y_j)^{\emptyset} \mid i,j = 1..4\} \cup \\ & \{(\neg x_i \vee \neg x_3)^{\emptyset}, (\neg x_i \vee \neg x_4)^{\emptyset} \mid i = 1,2\} \cup \\ & \{(\neg y_i)^{\{l,l_i\}}, (\neg y_i)^{\{l,t_i\}} \mid i = 1..4\}\end{aligned}$$

with $w(l) = 1$ and $w(l_i) = w(t_i) = 2$ for all i. First note that LVE, LSSR, LSE, and LBCE cannot simplify Φ. Specifically, as every variable appears both negatively and positively at least twice and no produced resolvents are tautologies, LVE cannot eliminate any variables. However, l subsumes all of the other labels, and hence applying SLE gives

$$\begin{aligned}& \{(x_i \vee y_j)^{\emptyset} \mid i,j = 1..4\} \cup \\ & \{(\neg x_i \vee \neg x_3)^{\emptyset}, (\neg x_i \vee \neg x_4)^{\emptyset} \mid i = 1,2\} \cup \\ & \{(\neg y_i)^{\{l\}} \mid i = 1..4\}.\end{aligned}$$

Each y_i appears negatively only in a single clause and can hence be eliminated by LVE, resulting in

$$\{(x_i)^{\{l\}} \mid i = 1..4\} \cup \{(\neg x_i \vee \neg x_3)^{\emptyset}, (\neg x_i \vee \neg x_4)^{\emptyset} \mid i = 1,2\}.$$

Now each x_i only appears positively in a single clause. LVE then gives $\Phi^{pre} = \{()^{\{l\}}\}$.

Remark 1 *While the main focus of this work is on understanding the effect of SLE on the core structure of MaxSAT instances and the potential of SLE to speed up state-of-the-art MaxSAT solvers, we note that SLE (for MaxSAT) can be viewed as the counterpart of the so-called* dominance rule *proposed in the early 90s in conjunction with branch-and-bound approaches for the so-called* binate covering problem *[17, 16] with applications in logic synthesis. More details on this connection are provided in Appendix A. To the best of our knowledge, however, SLE has not been previously proposed, analyzed, or empirically evaluated in the context of MaxSAT.*

4 EFFECTS OF SLE

We continue by analyzing SLE in terms of how it simplifies LCNFs. We show that SLE is orthogonal to the LCNF-lifted SAT-based preprocessing techniques in terms of the LMUSes and LMCSes—and hence MaxSAT solutions—preserved under simplification.

We start with relatively simple corollaries of the definition. First, we observe that subsumed labels remain subsumed after applications of SAT-based preprocessing.

Proposition 11 *Let $l \in Lbls(\Phi)$ and assume that SLE can eliminate l from Φ. Let Φ^{pre} be Φ after applying LVE, LSSR, LSE, or LBCE. Then SLE can eliminate l from Φ^{pre}.*

Proof. Let l_1 be a label that subsumes l in Φ. It suffices to show that the preconditions of SLE are satisfied in Φ^{pre}. First, the precondition $w(l_1) \leq w(l)$ is trivially satisfied as none of the techniques alter the weights of labels. For the second precondition, $LCl(\Phi^{pre}, l) \subseteq LCl(\Phi^{pre}, l_1)$, the non-trivial case is $LCl(\Phi^{pre}, l) \neq \emptyset$. As $LCl(\Phi, l) \subseteq LCl(\Phi, l_1)$, it is enough to verify that none of the SAT-based preprocessing techniques introduce a labelled clause $C^L \in \Phi^{pre}$ with $l \in L$ and $l_1 \notin L$. This is trivially true for LSE and LBCE as they only remove clauses. This is also true for LSSR as it only removes literals, not labels. Finally, LVE cannot produce resolvents which contain l but not l_1, since there are no labelled clauses $C^{L'}$ in Φ with $l \in L'$ and $l_1 \notin L'$. Thus the label-set of any resolvent produced by LVE, which is a union of label-sets in Φ, contains either both or neither of l_1 and l. □

Thus it makes sense to incorporate SLE into the preprocessing loop together with LVE, LSSR, LSE, and LBCE.

In analogy with Proposition 11, subsumed labels remain subsumed also under SLE steps quite generally. An exception comes from cases in which two labels l_1 and l_2 subsume each other, i.e., when l_1 and l_2 occur in exactly the same label-sets *and* $w(l_2) = w(l_1)$. Note also that, generally, if l_1 subsumes l_2, and l_2 subsumes l_3, then l_1 subsumes l_3.

Turning to comparing SLE and SAT-based preprocessing, Propositions 4 and 12 together illustrate fundamental differences between SLE and LVE, LSSR, LSE, and LBCE. By Proposition 4, LVE, LSSR, LSE, and LBCE preserve the LMUSes of LCNF-MaxSAT instances. This is not true for SLE. Instead, SLE guarantees (only) that at least one minimum-cost (optimal) LMCS and, as such, that at least one optimal solution of the instance is preserved.

Proposition 12 *SLE does not in general preserve LMUSes (or LMCSes) of LCNF-MaxSAT instances.*

Proof. Consider the instances Φ and Φ^{pre} from Example 10. The sets $\{l, l_i\}$ and $\{l, t_i\}$ are LMUSes of Φ for all i but not of Φ^{pre}. □

An alternative way of stating Proposition 12 is that applying SLE does not in general preserve all optimal solutions to LCNF-MaxSAT instances. For a simple example, consider the LCNF-MaxSAT instance $\Phi = \{(x)^{\{l_1\}}, (\neg x)^{\{l_2\}}\}$ with unit-weighted labels. There are two optimal solutions to Φ: $\tau_1(x) = 1$ satisfying $\Phi|_{Lbls(\Phi)\setminus\{l_2\}}$, and $\tau_2(x) = 0$ satisfying $\Phi|_{Lbls(\Phi)\setminus\{l_1\}}$. However, by LVE we can simplify Φ to $\{()^{\{l_1,l_2\}}\}$ and by SLE further to $\{()^{\{l_1\}}\}$. The only LMCS of the simplified instance is $\{l_1\}$, corresponding to the solution τ_2.

Instead of preserving LMUSes, SLE could be seen as a form of LMUS minimization in the sense that all LMUSes remaining after SLE are projections of LMUSes of the original LCNF onto the remaining set of labels.

Theorem 13 *Let Φ be a LCNF-MaxSAT instance and $l \in Lbls(\Phi)$ a subsumed label. Let $\Phi^{pre} = \text{REMOVE}(\Phi, \{l\})$, i.e., the formula after eliminating l by SLE from Φ. Then all LMUSes L^p of Φ^{pre} are of the form $L^p = L \cap Lbls(\Phi^{pre})$ for some LMUS L of Φ.*

Proof. First notice that $\Phi|_{L^p} \subseteq \Phi^{\text{pre}}|_{L^p}$ as the restriction operator only removes labels from label-sets, not clauses. If $\Phi|_{L^p} = \Phi^{\text{pre}}|_{L^p}$, then the same will be true for any $L^p_s \subseteq L^p$, so L^p itself is an LMUS of Φ. Otherwise, the reason for a labelled clause C^L to be in $\Phi^{\text{pre}}|_{L^p}$ but not in $\Phi|_{L^p}$ is that the eliminated label l was in L, i.e., $C^L \notin \Phi$ but $C^{L\cup\{l\}} \in \Phi$. Hence $\Phi|_{L^p\cup\{l\}} = \Phi^{\text{pre}}|_{L^p}$, and $L^p \cup \{l\}$ is a LMUS of Φ. □

For further differences between SLE and LVE, LSSR, LSE, and LBCE, consider a MaxSAT instance F and a soft clause $C \in F_s$. Let Φ_F be the LCNF-MaxSAT instance corresponding to F and l_C the label for which $C^{\{l_c\}} \in \Phi_F$. A simple application of Theorem 4 gives that if l_C is removed from Φ_F by LVE, LSSR, LSE, or LBCE, then *any* optimal solution to Φ_F, which is also an optimal solution to F, will satisfy C.

Proposition 14 *Let Φ_F^{pre} be the instance resulting after an application of LVE, LSSR, LSE, or LBCE on Φ_F. If $l_C \notin Lbls(\Phi_F^{pre})$, then any optimal solution τ to Φ_F, which is also an optimal solution to F, will satisfy C.*

Proof. Since τ is optimal, it satisfies $\Phi_F|_{Lbls(\Phi_F)\setminus R}$ for some minimum-cost LMCS R of Φ_F. By Theorem 4, $l_C \notin R$, and thus $C \in Cl(\Phi_F|_{Lbls(\Phi_F)\setminus R})$. □

Informally, it could be said that SAT-based preprocessing can only remove labels that are "uninteresting" in terms of LMCS computation. In contrast, elimination of l_C by SLE means that *some* (but not necessarily all) optimal solutions of F satisfy C, as shown next.

Proposition 15 *Let Φ_F^{pre} be the instance resulting from an application of SLE on Φ_F. If $l_C \notin Lbls(\Phi^{pre})$, then there is an optimal solution τ to Φ_F and F that satisfies C. Furthermore, there may exist optimal solutions to Φ_F that do not satisfy C.*

Proof. By the assumption that l_C is subsumed, it follows from Theorem 6 and Proposition 8 that there is a minimum-cost LMCS R of Φ_F for which $l_C \notin R$. The first part of the claim follows by observing that $\Phi_F|_{Lbls(\Phi_F)\setminus R}$ is satisfiable and $C \in Cl(\Phi_F|_{Lbls(\Phi_F)\setminus R})$. For the second part of the claim, consider the discussion following Proposition 12. □

5 SLE AND CORE-GUIDED SOLVERS

We now show that SLE has the potential to considerably lower the number of iterations made by so-called *core-guided MaxSAT solvers*, one of the most successful current MaxSAT solving approaches. The core-guided approach has several variants, e.g. [2, 38, 22, 25, 40, 36, 37, 18, 19]. In this work, we study the effect of SLE on two different types of core-guided solvers through generic abstractions. The first one, *CG-MaxSAT* (Algorithm 1), iteratively employs a SAT solver to extract unsatisfiable cores and rules out each of the found cores from the formula by a *clause replication and relaxation* step. Several algorithms that fit the CG-MaxSAT abstraction have been proposed [22, 25, 40, 36, 37]. The second one, MaxHS (Algorithm 2), is an abstraction of the implicit hitting set approach to MaxSAT [18, 19], 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-MaxSAT invokes a SAT solver on the clauses of a working formula F^i_w (initialized as all clauses of the MaxSAT instance viewed as hard). If the working formula is satisfiable, CG-MaxSAT terminates and returns the satisfying assignment returned by the SAT solver. Otherwise, the SAT solver returns an unsatisfiable core κ of F^i_w. CG-MaxSAT then duplicates the clauses in κ to create two sets κ^r and $\kappa^{\bar{r}}$. Both sets contain exactly the same clauses as κ; each clause $C \in \kappa$ is duplicated into two: $C^r \in \kappa^r$ and $C^{\bar{r}} \in \kappa^{\bar{r}}$. The weight of C^r is set to w_m, the minimum weight over the clauses in the core, and the weight of $C^{\bar{r}}$ to $w(C) - w_m$. The clauses of $\kappa^{\bar{r}}$ are added to the working formula unaltered. Finally, the working formula is updated by *relaxing* the clauses in κ^r. The method of relaxation varies between core-guided solvers. For our analysis, the important consequences of relaxation are that the (possibly altered) clauses of κ^r do not appear as a core in future iterations, and that the optimal cost of F^{i+1}_w (when viewed as a MaxSAT instance) is exactly w_m lower than the optimal cost of F^i_w. Termination of CG-MaxSAT is guaranteed by the fact that $w_m > 0$ on all iterations and that a MaxSAT instance of cost 0 is satisfiable as a SAT instance. For a concrete example of a relaxation step, consider the classical Fu-Malik algorithm [22] and its extensions to the weighted case [33, 1]. These algorithms augments each $C_i \in \kappa^r$ with a fresh relaxation variable r_i, creating the clause $C_i \vee r_i$, and additionally adds a hard *exactly-one* constraint $\sum r_i = 1$ over the relaxation variables. The intuition behind this step is that assigning a relaxation variable to 1 effectively removes the corresponding clause from the formula, hence removing the core κ^r. Additionally,

Input: MaxSAT instance $F = (F_h, F_s, w)$.
Output: An optimal solution τ for F.
$F^0_w \leftarrow F_h \cup F_s$
for *i=0,...* **do**
  $(\textit{result}, \kappa, \tau) \leftarrow \text{SATSOLVE}(F^i_w)$
  if *result="satisfiable"* **then**
    return τ // optimal solution
  else
    $F^i_w = (F^i_w \setminus \kappa)$ // SAT solver returned unsat core
    $w_m \leftarrow \min\{w(C) \mid C \in \kappa\}$
    $(\kappa^r, \kappa^{\bar{r}}) \leftarrow \text{CLAUSEREPLICATE}(\kappa, w_m)$
    $F^i_w \leftarrow F^i_w \cup \kappa^{\bar{r}}$
    $F^{i+1}_w \leftarrow \text{RELAX}(F^i_w, \kappa^r)$
  end
end

Algorithm 1: CG-MaxSAT

Input: MaxSAT instance $F = (F_h, F_s, w)$.
Output: An optimal solution τ for F.
$\mathcal{K} \leftarrow \emptyset$ // set of found unsat cores of F
$F_w \leftarrow (F_h \cup F_s)$
while *true* **do**
  $H \leftarrow$ MINCOSTHITTINGSET($\mathcal{K}$)
  $F_w \leftarrow F_h \cup (F_s \setminus H)$
  (*result*, κ, τ) $\leftarrow$ SATSOLVE(F_w)
  if *result="satisfiable"* **then**
    return τ // optimal solution
  else
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\kappa\}$ // SAT solver returned unsat core
  end
end

Algorithm 2: MaxHS

the exactly-one constraint ensures that the cost is lowered exactly by w_m.

MaxHS is a hybrid algorithm that uses a SAT solver for core extraction over a working formula F_w (initialized as all clauses of the input instance viewed as hard). Given a collection $\mathcal{K}$ of extracted cores, MaxHS uses an exact algorithm (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 is 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 again with hitting set computation.

The main result of this section is that there are families of LCNF-MaxSAT instances on which SLE can significantly decrease the number of SAT solver calls and clause replication when subsequently solving the instances with CG-MaxSAT or MaxHS.

Proposition 16 *For $\mathcal{A} \in \{$CG-MaxSAT, MaxHS$\}$, there is a family of LCNF-MaxSAT instances Φ_N, with $\Theta(N)$ different labels, on which*

(i) $\mathcal{A}$ requires $\Theta(N)$ calls to its SAT solver, and, for $\mathcal{A} =$ CG-MaxSAT, $\mathcal{A}$ requires $\Theta(N)$ clause replication steps, on $\Theta(N!)$ different executions; while
(ii) $\mathcal{A}$ is guaranteed to require only two (one unsatisfiable and one satisfiable) SAT solver calls if SLE is applied on Φ_N before $\mathcal{A}$

under the assumption that the internal SAT solver is guaranteed to return minimal unsatisfiable cores.

Proof. The family of LCNF-MaxSAT instances witnessing the claim is the same for CG-MaxSAT and MaxHS. Let N be sufficiently large and define

$$\Phi_N := \bigcup_{i=1}^{2N-2} \mathbf{P}_i \cup \bigcup_{i=1}^{N-1} \mathbf{H}_i, \text{ where}$$

$$\mathbf{P}_i = \bigcup_{j=1}^{N} \{(\neg p_i^j \vee \neg p_k^j)^{\emptyset} \mid k = (i+1)..(2N-1)\} \text{ and}$$

$$\mathbf{H}_i = \left\{ \left(\bigvee_{j=1}^{N} p_k^j \right)^{\{l, l_i\}} \;\middle|\; k = i..(N+i) \right\},$$

with $w(l) = w(l_{N-1}) = N$ and $w(l_i) = 1$ for all other labels l_i. Notice that Φ_N contains $N-1$ LMUSes of the form $\{l, l_i\}$ for all $1 \leq i \leq N-1$. Hence, the only minimum-cost LMCS of Φ_N is $\{l\}$. Furthermore, refuting any of the LMUSes requires proving the unsatisfiability of the formula $\Phi_N|_{\{l,l_i\}}$, which corresponds to an instance of the pigeonhole principle; meaning that the extraction any of the LMUSes of Φ_N requires an exponentially long SAT solver call [24]. Next we sketch the executions of both CG-MaxSAT and MaxHS that require $\Theta(N)$ SAT-solver calls when solving Φ_N.

Conversion of Φ_N to MaxSAT results in the formula $F = (F_h, F_s, w)$, where

$$F_h = \bigcup_{i=1}^{2N-2} \bigcup_{j=1}^{N} \{(\neg p_i^j \vee \neg p_k^j) \mid k = i..(2N-1)\}$$
$$\cup \bigcup_{i=1}^{N-1} \left\{ \left(a_l \vee a_i \vee \bigvee_{j=1}^{N} p_k^j \right) \;\middle|\; k = i..(N+i) \right\}$$

and $F_s = \{(\neg a_l), (\neg a_1), \ldots, (\neg a_{N-1})\}$
with $w((\neg a_l)) = w((\neg a_{N-1})) = N$ and $w(C) = 1$ for all other $C \in F_s$. The MUSes of F correspond exactly to the LMUSes of Φ_N and are of the form $\{(\neg a_l), (\neg a_i)\}$ for all $i = 1..N-1$. For an intuition on the executions requiring a linear number of SAT solver calls of both algorithms, notice that both can terminate immediately and only after encountering and processing the MUS $\{(\neg a_l), (\neg a_{N-1})\}$ corresponding to the the LMUS $\{l, l_{N-1}\}$.

For $\mathcal{A} =$ MaxHS, assume that the internal SAT solver returns the MUSes of in any order with $\{(\neg a_l), (\neg a_{N-1})\}$ last. Then the hitting set hs computed by MaxHS will not contain the clause $(\neg a_l)$ before the $(N-1)$th iteration and as such MaxHS can not terminate as $F \setminus hs$ will always contain the MUS $\{(\neg a_l), (\neg a_{N-1})\}$. There are a total of $(N-2)!$ executions in which the MUS $\{(\neg a_l), (\neg a_{N-1})\}$ is returned last.

For $\mathcal{A} =$ CG-MaxSAT, the long executions are similar. Assume that the first MUS returned by the SAT-solver in CG-MaxSAT is $\{(\neg a_l), (\neg a_1)\}$. The smallest weight w_m of the clauses in the core is 1, so CG-MaxSAT proceeds by replicating the clause $(\neg a_l)$ into two clauses $C^r = (\neg a_l)$ and $C_2 = (\neg a_l)$, setting $w(C^r) = 1$ and $w(C_2) = N-1$, adding C_2 back into the working formula, relaxing the core $\{C^r, (\neg a_1)\}$, and reiterating. Assume that CG-MaxSAT proceeds similarly by processing the cores $\{(\neg a_l)^i, (\neg a_i)\}$ for $i = 1..N-2$ during the first $N-2$ iterations where $(\neg a_l)^i$ is the copy of the clause $(\neg a_l)$ produced in the previous iteration. Finally on the $(N-1)$th iteration CG-MaxSAT encounters the core $\{(\neg a_l)^{N-2}, (\neg a_{N-1})\}$. At this point $w((\neg a_l)^{N-2}) = 2$ and $w((\neg a_{N-1})) = N$, so CG-MaxSAT replicates $(\neg a_{N-1})$ and relaxes the core before invoking its SAT solver one final time in order to find the current working formula satisfiable. In total, CG-MaxSAT performs N SAT solver calls and $N-1$ clause replications. A similar argument can be made for any ordering of the MUSes with $\{(\neg a_l), (\neg a_{N-1})\}$ last.

Part (ii) of the proposition follows by noting that SLE can remove l_{N-1} due to l, resulting in the formula

$$pre(\Phi_N) := \bigcup_{i=1}^{2N-2} \mathbf{P}_i \cup \bigcup_{i=1}^{N-2} \mathbf{H}_i \cup \left\{ \left(\bigvee_{j=1}^{N} p_k^j \right)^{\{l\}} \;\middle|\; k = (N-1)..(2N-1) \right\}.$$

The only LMUS of the preprocessed formula is $\{l\}$, which is why both algorithms are guaranteed to need only a single unsatisfiable and a single satisfiable SAT-solver call, and furthermore, why CG-MaxSAT needs no clause replication steps, during solving. □

6 EXPERIMENTS

Complementing the theoretical analysis, we evaluate the practical effects of SLE on the 2015 MaxSAT Evaluation benchmarks (http://www.maxsat.udl.cat/15/). We observe that SLE is beneficial especially on *industrial* weighted partial benchmark instances. When applying SLE in conjunction with the LCNF-lifted SAT-based preprocessing techniques (LVE, LSSR, LSE, LBCE), noticeably more labels can be removed than without applying SLE. Furthermore, SLE improves the overall performance of various state-of-the-art MaxSAT solvers on industrial weighted partial benchmarks.

All reported solving times include the time spent in preprocessing as well as in the actual MaxSAT solving. The experiments were run on 2.53-GHz Intel Xeon quad-core machines with 32-GB RAM under Linux. A per-instance timeout of 1800 seconds and a memory limit of 30 GB were enforced.

We implemented SLE by extending the Coprocessor 2.0 SAT preprocessor [34] in the following way. Given a MaxSAT instance as input, we convert the instance to LCNF, apply Coprocessor to preprocess the LCNF, and then convert the preprocessed LCNF back to a MaxSAT instance. LVE, LSSR LSE, LSSR, and LBCE are realized by representing a labelled clause C^L as $C \vee \bigvee_{l_i \in L} a_i$ in Coprocessor, applying the existing implementations of VE, SSR, SE and BCE, while forbidding the elimination of any of the a_i variables corresponding to the labels.

A simple way of implementing SLE consists of explicitly checking for each label l whether or not l is subsumed. A potentially more efficient way of implementing SLE would be to track the resolvents produced by LVE and only check labels that have appeared in resolvents produced. However, as shown in Figure 3, even the simple implementation appears to be sufficient; we did not observe any significant increase in total preprocessing time (w/pre+SLE) compared to not using SLE (w/pre). We also note that SLE does not increase overall memory consumption wrt SAT-based preprocessing.

The fraction of labels (i.e. soft clauses) remaining after preprocessing with and without SLE (applying in both cases LVE, LSSR, LSE, and LBCE) is shown in Figure 2 for both unweighted and weighted partial industrial and crafted instances. SLE is effective in removing additional labels in particular on the industrial weighted partial instances. For example, for one third of the instances ($x = 0.3$), with SLE close to 80% of the labels are eliminated ($y \approx 0.2$, i.e., some 20% of the labels remain afterwards); in comparison, without SLE only $\approx 45\%$ are eliminated. As a side-note, when examining the instance families in more detail, we found that out of the 172 industrial benchmarks in which no labels were removable by preprocessing, 151 were new instances in the 2015 evaluation. In fact, when preprocessing the 2014 evaluation instances—which are a subset of the 2015 evaluation instances—using SLE, at least 80% of the labels are eliminated from over 50% of the instances. This suggests that, in terms of SLE, the instances added for 2015 are structurally different from the ones from 2014.

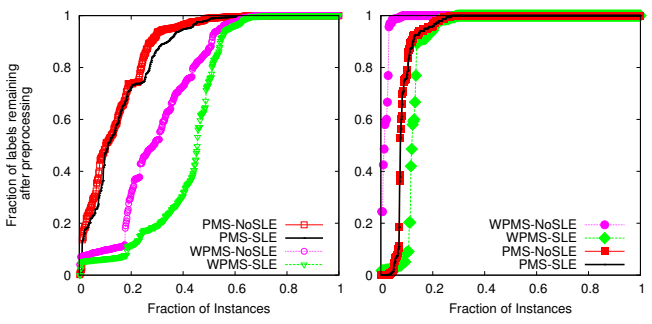

Figure 2: Fraction of labels remaining in industrial (left) and crafted (right) unweighted (PMS) and weighted (WPMS) benchmarks after preprocessing with and without SLE.

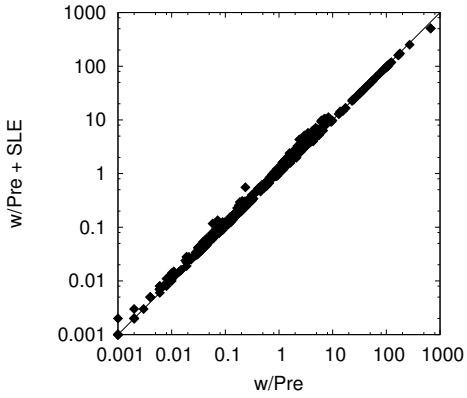

Figure 3: Influence of SLE on preprocessing time

Table 1: Number of solved industrial weighted partial benchmarks and total time spent on solved instances without preprocessing (default), with SAT-based preprocessing (w/pre), and with both SAT-based preprocessing and SLE (w/pre+SLE).

	Solved instances (total running time over solved in seconds)			
config.	**Eva**	**LMHS**	**Open-WBO**	**Primal-Dual**
default	379 (22,543)	354 (50,981)	331 (15,762)	390 (18,423)
w/pre	384 (20,613)	368 (46,525)	369 (12,345)	391 (15,267)
w/pre+SLE	**386 (19,138)**	**389** (48,277)	**369 (11,739)**	**392 (13,925)**

The additional simplifications obtained via SLE are also reflected in the total number of solved instances and solver runtimes on industrial weighted partial instances. Results are shown in Table 1 for the state-of-the-art MaxSAT solvers Eva [40], core-guided, best industrial weighted partial solver in 2014; LMHS [11], one of the best crafted and industrial weighted partial solvers in 2015, a labelled lifting of the SAT-IP hybrid MaxSAT solver MaxHS [18]; Open-WBO [36], one of the best industrial unweighted solvers in 2015; and Primal-Dual [12], a new core-guided solver from 2015. SAT-based preprocessing together with SLE results in the highest number of solved instances for each of the solvers. The increase in the number of solved instances is especially noticeable for LMHS. SLE also decreases the total runtime over all solved instances for each of the solvers. For example, for both Eva and Primal-Dual, using SLE improves further on applying only SAT-based preprocessing by decreasing the total runtime by approximately 10%, at the same time enabling Primal-Dual and Eva to solve one and two more instances, respectively. Finally, Figure 4 shows a comparison the running times of the individual instances with the solvers are presented in the order LMHS (first column), Eva (second), Open-WBO (third), and Primal-Dual (fourth column). For each solver, we compare runtimes on logscale when applying SLE together with LVE, LSSR, LSE, and LBCE ('w/pre+SLE') to (i) without preprocessing (left), and (ii) preprocessing only with LVE, LSSR, LSE, and LBCE ('w/pre", right). For a majority of the instances, SLE improves the total solving time of each of the solvers both compared to using no preprocessing, and only using LVE, LSSR, LSE and LBCE.

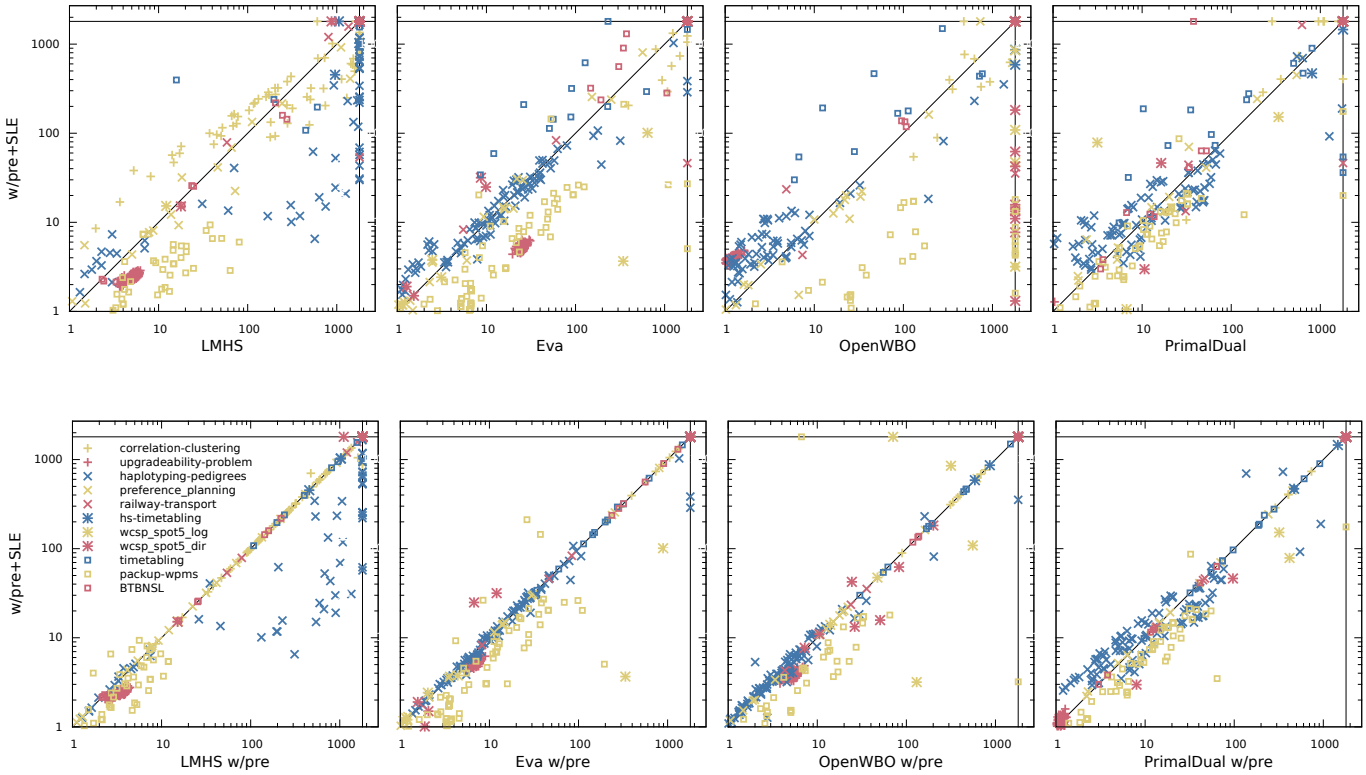

Figure 4: Effect of SLE on runtimes without (top) and with (bottom) other preprocessing on industrial weighted partial instances.

7 CONCLUSIONS

We proposed *subsumed label elimination* (SLE) as a MaxSAT preprocessing technique that is beneficial to apply in conjunction with SAT-based preprocessing techniques before MaxSAT solving. SLE is orthogonal to SAT-based preprocessing in that SLE can eliminate redundant auxiliary variables (labels) from clauses irrespective of the original variables occurring in clauses. On the level of labelled CNFs, this accounts to removing redundant labels from LMUSes, thereby resulting in cases in a decrease in the number and sizes of MUSes of MaxSAT instances. Furthermore, SLE has the potential to drastically reduce the number of iterations performed by core-guided MaxSAT solvers, currently one of the important classes of MaxSAT solvers. Applying SLE further improves the running times of various state-of-the-art MaxSAT solvers on standard industrial weighted partial benchmarks. For future work, we aim to study more general notions of redundancies over labels in LCNFs to obtain further label-based preprocessing techniques for MaxSAT, as well as to study potential applications in MUS extraction.

ACKNOWLEDGEMENTS

This work has been funded by Academy of Finland, grants 251170 COIN, 276412, and 284591; and Doctoral School in Computer Science DoCS and Research Funds of the University of Helsinki.

A SLE and Dominance in Binate Covering

SLE (for MaxSAT) can be viewed as the counterpart of the so-called *dominance rule* proposed in the early 90s in conjunction with branch-and-bound approaches for the so-called *binate covering problem* [17, 16] with applications in logic synthesis. In short, in the binate covering problem, we are given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ over the variables $x_1, \ldots, x_n$, and a function $cost\colon \{1..n\} \to \mathbb{N}$ assigning a non-negative cost $cost(i)$ to each variable x_i. The task is to find a truth assignment τ over $x_1, \ldots, x_n$ that minimizes $\sum_{i=1}^{n} \tau(x_i) \cdot cost(i)$ subject to $f(\tau(x_1), \ldots, \tau(x_n)) = 1$. The dominance rule for binate covering is described in [17] for the so-called modified covering matrix representation of binate covering for Boolean functions in CNF. We interpret the rule directly on the definition as follows: variable x_i dominates x_j if (i) the literal x_i occurs in a clause C whenever the literal x_j occurs in C; (ii) $\neg x_j$ occurs in a clause C whenever $\neg x_i$ occurs in C; and (iii) $cost(x_i) \leq cost(x_j)$. A dominated variable can be assigned to 0.

A LCNF-MaxSAT instance (Φ, w) can be viewed as an instance of binate covering by viewing each labelled clause $C^L \in \Phi$ as the clause $C \vee L$, and letting $cost(l) = w(l)$ for each $l \in Lbls(\Phi)$ and $cost(x) = 0$ for each variable in $\bigcup Cl(\Phi)$. After this reduction, one can observe that, for any label $l \in Lbls(\Phi)$, it holds that l is dominated in the resulting binate covering instance if and only if SLE can eliminate l from (Φ, w).

REFERENCES

[1] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy, 'Solving (weighted) partial MaxSAT through satisfiability testing', in *Proc. SAT*, volume 5584 of *Lecture Notes in Computer Science*, pp. 427–440. Springer, (2009).

[2] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy, 'SAT-based MaxSAT algorithms', *Artificial Intelligence*, **196**, 77–105, (2013).

[3] Carlos Ansótegui, Idelfonso Izquierdo, Felip Manyà, and José Torres-Jiménez, 'A Max-SAT-based approach to constructing optimal covering arrays', in *Proc. CCIA*, volume 256 of *Frontiers in Artificial Intelligence and Applications*, pp. 51–59. IOS Press, (2013).

[4] Josep Argelich, Daniel Le Berre, Inês Lynce, João P. Marques-Silva, and Pascal Rapicault, 'Solving linux upgradeability problems using boolean optimization', in *Proc. LoCoCo*, volume 29 of *EPTCS*, pp. 11–22, (2010).

[5] Josep Argelich, Chu Min Li, and Felip Manyà, 'A preprocessor for Max-SAT solvers', in *Proc. SAT*, volume 4996 of *Lecture Notes in Computer Science*, pp. 15–20. Springer, (2008).

[6] Anton Belov and Joao Marques-Silva, 'Generalizing redundancy in propositional logic: Foundations and hitting sets duality', *CoRR*, **abs/1207.1257**, (2012).

[7] Anton Belov, Antonio Morgado, and Joao Marques-Silva, 'SAT-based preprocessing for MaxSAT', in *Proc. LPAR-19*, volume 8312 of *Lecture Notes in Computer Science*, pp. 96–111. Springer, (2013).

[8] Jeremias Berg and Matti Järvisalo, 'SAT-based approaches to treewidth computation: An evaluation', in *Proc. ICTAI*, pp. 328–335. IEEE Computer Society, (2014).

[9] Jeremias Berg and Matti Järvisalo, 'Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability', *Artificial Intelligence*, (2015). in press.

[10] Jeremias Berg, Matti Järvisalo, and Brandon Malone, 'Learning optimal bounded treewidth Bayesian networks via maximum satisfiability', in *Proc. AISTATS*, volume 33, pp. 86–95. JMLR, (2014).

[11] Jeremias Berg, Paul Saikko, and Matti Järvisalo, 'Improving the effectiveness of SAT-based preprocessing for MaxSAT', in *Proc. IJCAI*, pp. 239–245. AAAI Press, (2015).

[12] Nikolaj Bjørner and Nina Narodytska, 'Maximum satisfiability using cores and correction sets', in *Proc. IJCAI*, pp. 246–252. AAAI Press, (2015).

[13] Maria Luisa Bonet, Jordi Levy, and Felip Manyà, 'Resolution for Max-SAT', *Artificial Intelligence*, **171**(8-9), 606–618, (2007).

[14] Yibin Chen, Sean Safarpour, João Marques-Silva, and Andreas G. Veneris, 'Automated design debugging with maximum satisfiability', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **29**(11), 1804–1817, (2010).

[15] Yibin Chen, Sean Safarpour, Andreas G. Veneris, and João P. Marques-Silva, 'Spatial and temporal design debug using partial MaxSAT', in *Proc. 19th ACM Great Lakes Symposium on VLSI*, pp. 345–350. ACM, (2009).

[16] Olivier Coudert, 'On solving covering problems', in *Proc. DAC*, pp. 197–202. ACM Press, (1996).

[17] Olivier Coudert and Jean Christophe Madre, 'New ideas for solving covering problems', in *Proc. DAC*, pp. 641–646. ACM Press, (1995).

[18] Jessica Davies and Fahiem Bacchus, 'Exploiting the power of MIP solvers in MaxSAT', in *Proc. SAT*, volume 7962 of *Lecture Notes in Computer Science*, pp. 166–181. Springer, (2013).

[19] Jessica Davies and Fahiem Bacchus, 'Postponing optimization to speed up MAXSAT solving', in *Proc. CP*, volume 8124 of *Lecture Notes in Computer Science*, pp. 247–262. Springer, (2013).

[20] Niklas Eén and Armin Biere, 'Effective preprocessing in SAT through variable and clause elimination', in *Proc. SAT*, volume 3569 of *Lecture Notes in Computer Science*, pp. 61–75. Springer, (2005).

[21] Zhiwen Fang, Chu-Min Li, Kan Qiao, Xu Feng, and Ke Xu, 'Solving maximum weight clique using maximum satisfiability reasoning', in *Proc. ECAI*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 303–308. IOS Press, (2014).

[22] Zhaohui Fu and Sharad Malik, 'On solving the partial MaxSAT problem', in *Proc. SAT*, volume 4121 of *Lecture Notes in Computer Science*, pp. 252–265. Springer, (2006).

[23] Joao Guerra and Ines Lynce, 'Reasoning over biological networks using maximum satisfiability', in *Proc. CP*, volume 7514 of *Lecture Notes in Computer Science*, pp. 941–956. Springer, (2012).

[24] Armin Haken, 'The intractability of resolution', *Theoretical Computer Science*, **39**, 297–308, (1985).

[25] Federico Heras, Antonio Morgado, and Joao Marques-Silva, 'Core-guided binary search algorithms for maximum satisfiability', in *Proc. AAAI*. AAAI Press, (2011).

[26] Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere, 'Clause elimination for SAT and QSAT', *Journal of Artificial Intelligence Research*, **53**, 127–168, (2015).

[27] Alexey Ignatiev, Mikolás Janota, and João Marques-Silva, 'Towards efficient optimization in package management systems', in *Proc. ICSE*, pp. 745–755. ACM, (2014).

[28] Matti Järvisalo, Armin Biere, and Marijn Heule, 'Blocked clause elimination', in *Proc. TACAS*, volume 6015 of *Lecture Notes in Computer Science*, pp. 129–144. Springer, (2010).

[29] Matti Järvisalo, Marijn Heule, and Armin Biere, 'Inprocessing rules', in *Proc. IJCAR*, volume 7364 of *Lecture Notes in Computer Science*, pp. 355–370. Springer, (2012).

[30] Manu Jose and Rupak Majumdar, 'Cause clue clauses: error localization using maximum satisfiability', in *Proc. PLDI*, pp. 437–446. ACM, (2011).

[31] Chu Min Li, Felip Manyà, Nouredine Ould Mohamedou, and Jordi Planes, 'Exploiting cycle structures in Max-SAT', in *Proc. SAT*, volume 5584 of *Lecture Notes in Computer Science*, pp. 467–480. Springer, (2009).

[32] Inês Lynce and João Marques-Silva, 'Restoring CSP satisfiability with MaxSAT', *Fundam. Inform.*, **107**(2-3), 249–266, (2011).

[33] Vasco M. Manquinho, João P. Marques-Silva, and Jordi Planes, 'Algorithms for weighted boolean optimization', in *Proc. SAT*, volume 5584 of *Lecture Notes in Computer Science*, pp. 495–508. Springer, (2009).

[34] Norbert Manthey, 'Coprocessor 2.0 - A flexible CNF simplifier', in *Proc. SAT*, volume 7317 of *Lecture Notes in Computer Science*, pp. 436–441. Springer, (2012).

[35] Joao Marques-Silva, Mikolas Janota, Alexey Ignatiev, and Antonio Morgado, 'Efficient model based diagnosis with maximum satisfiability', in *Proc. IJCAI*, pp. 1966–1972. AAAI Press, (2015).

[36] Ruben Martins, Saurabh Joshi, Vasco M. Manquinho, and Ines Lynce, 'Incremental cardinality constraints for MaxSAT', in *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, pp. 531–548. Springer, (2014).

[37] Antonio Morgado, Carmine Dodaro, and Joao Marques-Silva, 'Core-guided MaxSAT with soft cardinality constraints', in *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, pp. 564–573. Springer, (2014).

[38] Antonio Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and Joao Marques-Silva, 'Iterative and core-guided MaxSAT solving: A survey and assessment', *Constraints*, **18**(4), 478–534, (2013).

[39] António Morgado, Mark H. Liffiton, and João Marques-Silva, 'MaxSAT-based MCS enumeration', in *Revised Selected Papers of HVC 2012*, volume 7857 of *Lecture Notes in Computer Science*, pp. 86–101. Springer, (2013).

[40] Nina Narodytska and Fahiem Bacchus, 'Maximum satisfiability using core-guided MaxSAT resolution', in *Proc. AAAI*, pp. 2717–2723. AAAI Press, (2014).

[41] James D. Park, 'Using weighted MAX-SAT engines to solve MPE', in *Proc. AAAI*, pp. 682–687. AAAI Press / The MIT Press, (2002).

[42] Johannes Peter Wallner, Andreas Niskanen, and Matti Järvisalo, 'Complexity results and algorithms for extension enforcement in abstract argumentation', in *Proc. AAAI*, pp. 1088–1094. AAAI Press, (2016).

[43] Lei Zhang and Fahiem Bacchus, 'MAXSAT heuristics for cost optimal planning', in *Proc. AAAI*. AAAI Press, (2012).

[44] Charlie S. Zhu, Georg Weissenbacher, and Sharad Malik, 'Post-silicon fault localisation using maximum satisfiability and backbones', in *Proc. FMCAD*, pp. 63–66. FMCAD Inc., (2011).