Improving the Effectiveness of SAT-Based Preprocessing for MaxSAT

Jeremias Berg and **Paul Saikko** and **Matti Järvisalo**
HIIT & Department of Computer Science, University of Helsinki, Finland

Abstract

Solvers for the Maximum satisfiability (MaxSAT) problem find an increasing number of applications today. We focus on improving MaxHS—one of the most successful recent MaxSAT algorithms—via SAT-based preprocessing. We show that employing SAT-based preprocessing via the so-called labelled CNF (LCNF) framework before calling MaxHS can in some cases greatly degrade the performance of the solver. As a remedy, we propose a lifting of MaxHS that works directly on LCNFs, allowing for a tighter integration of SAT-based preprocessing and MaxHS. Our empirical results on standard crafted and industrial weighted partial MaxSAT Evaluation benchmarks show overall improvements over the original MaxHS algorithm both with and without SAT-based preprocessing.

1 Introduction

Boolean satisfiability (SAT) solving is a modern success story of computer science, providing means of solving various types of hard computational problems, based on both direct applications of SAT solvers, as well as on using SAT solvers as core NP procedures within more complex decision and optimization procedures. This success is based on several breakthroughs in practical solver techniques, central to which is preprocessing [Eén and Biere, 2005; Heule *et al.*, 2010; Järvisalo *et al.*, 2012]. However, applying SAT-level preprocessing in more complex applications of SAT solvers, such as minimal unsatisfiable core extraction [Belov *et al.*, 2013a], maximum satisfiability [Belov *et al.*, 2013b], and model counting [Lagniez and Marquis, 2014], becomes more difficult, as many of the central SAT preprocessing techniques can no longer be applied directly without losing correctness.

In this work, we focus on the Maximum satisfiability (MaxSAT) problem [Li and Manyà, 2009; Morgado *et al.*, 2013; Ansótegui *et al.*, 2013], a well-known optimization variant of SAT. Due to recent progress in MaxSAT solving [Heras *et al.*, 2011; Koshimura *et al.*, 2012; Davies and Bacchus, 2013a; 2013b; Morgado *et al.*, 2013; Ansótegui and Gabàs, 2013; Ansótegui *et al.*, 2013; Morgado *et al.*, 2014; Martins *et al.*, 2014], MaxSAT finds an increasing number of applications today [Jose and Majumdar, 2011; Zhu *et al.*, 2011; Guerra and Lynce, 2012; Berg and Järvisalo, 2013; Berg *et al.*, 2014; Bunte *et al.*, 2014]. While some of the most important SAT preprocessing techniques, such as bounded variable elimination [Eén and Biere, 2005], cannot be directly applied in the context of MaxSAT [Belov *et al.*, 2013b], a workaround is provided by applying the so-called labelled CNF (LCNF) framework [Belov and Marques-Silva, 2012].

We focus on improving the performance of the MaxHS approach [Davies and Bacchus, 2011; 2013a; 2013b] to MaxSAT solving via SAT-based preprocessing. MaxHS implements a hybrid approach to MaxSAT based on alternating between SAT-based unsatisfiable core extraction and integer programming (IP) based optimal hitting set computation over the unsatisfiable cores. The solver was one of the best in the 2014 MaxSAT Evaluation in the crafted weighted partial MaxSAT category. Motivated by this, we develop a lifting of MaxHS that works directly on LCNFs for solving MaxSAT instances, which allows for a tight integration of SAT-based preprocessing and MaxHS, and specifically, allows for *directly re-using* assumption variables from the SAT-based preprocessing step within the MaxHS solver loop. MaxHS computation heavily relies on assumption variables (both in the SAT solver and the IP solver), enabling more re-use of assumption variables from the preprocessing phase during the whole execution of the solver compared to e.g. earlier work on integrating preprocessing with MaxSAT algorithms [Belov *et al.*, 2013b]. The re-use is beneficial in terms of both having to introduce less clauses to the solver, and, as we explain, enabling more inference within MaxHS. We formally prove the correctness of the proposed LCNF-level lifting of MaxHS, and present details on how the lifting can be realized by minor modifications to the original MaxHS implementation. We present empirical results using our own competitive re-implementation of MaxHS, with additional features for implementing the LCNF-level lifting of MaxHS. The results show the benefits of the tighter integration of preprocessing and MaxHS, with overall improvements over the original MaxHS algorithm both with and without SAT-based preprocessing, on standard crafted and industrial weighted partial MaxSAT Evaluation benchmarks.

2 SAT, Preprocessing, and MaxSAT

SAT. For a Boolean variable x, there are two literals, x and $\neg x$. A clause is a disjunction ($\vee$) of literals. A truth assign-

ment is a function from Boolean variables to $\{0,1\}$. A clause C is satisfied by a truth assignment τ ($\tau(C)=1$) if $\tau(x)=1$ for a literal x in C, or $\tau(x)=0$ for a literal $\neg x$ in C. A set $F=\{C_1,\ldots,C_m\}$ of clauses, or equivalently, the conjunctive normal form (CNF) formula $\bigwedge_{i=1}^{m} C_i$, is satisfiable ($F \in$ SAT) if there is an assignment τ satisfying all clauses in F ($\tau(F)=1$), and unsatisfiable ($\tau(F)=0$ for any assignment τ; $F \in$ UNSAT) otherwise. The Boolean satisfiability problem (SAT) is to decide whether a given CNF formula is satisfiable.

SAT Preprocessing. The resolution rule states that, given two clauses $C_1=(x \vee A)$ and $C_2=(\neg x \vee B)$, the clause $C=(A \vee B)$, the *resolvent* of C_1 and C_2, can be inferred by *resolving* on the variable x. We write $C=C_1 \bowtie_x C_2$. This is lifted to two sets S_x and $S_{\neg x}$ of clauses that all contain the literal x and $\neg x$, resp., by $S_x \bowtie_x S_{\neg x}=\{C_1 \bowtie_x C_2 \mid C_1 \in S_x, C_2 \in S_{\neg x}, \text{ and } C_1 \bowtie_x C_2 \text{ is not a tautology}\}$.

Bounded variable elimination (VE) [Eén and Biere, 2005], currently the most important SAT preprocessing technique, follows the Davis-Putnam procedure (DP). The elimination of a variable x in a CNF formula is computed by resolving pairwise each clause in S_x with every clause in $S_{\neg x}$. Replacing the original clauses in $S_x \cup S_{\neg x}$ with the non-tautological resolvents $S=S_x \bowtie_x S_{\neg x}$ gives the CNF $(F \setminus (S_x \cup S_{\neg x})) \cup S$ that is equisatisfiable with F. To avoid exponential space complexity, VE is bounded typically by requiring that a variable x can be eliminated only if the resulting CNF formula $(F \setminus (S_x \cup S_{\neg x})) \cup S$ will not contain more than Δ more clauses than the original formula F [Eén and Biere, 2005].

A clause C in a CNF formula F is subsumed if there is a clause $C' \subset C$ in F. *Subsumption elimination* (SE) removes subsumed clauses. The *self-subsuming resolution* rule states that, given two clauses $C, D \in F$ such that (i) $l \in C$ and $\neg l \in D$ for some literal l, and (ii) D is subsumed by $C \bowtie_l D$, D can be *replaced* with $C \bowtie_l D$ in F (or, informally, $\neg l$ can be removed from D). A step of self-subsuming resolution (SSR), resolving C and D on l, gives the formula $(F \setminus D) \cup \{C \bowtie_l D\}$.

A clause C of a CNF formula F is *blocked* [Kullmann, 1999] if there is a literal $l \in C$ such that for every clause $C' \in F$ with $\neg l \in C'$, the resolvent $(C \setminus \{l\}) \cup (C' \setminus \{\neg l\})$ obtained from resolving C and C' on l is a tautology. *Blocked clause elimination* (BCE) [Järvisalo *et al.*, 2010] removes blocked clauses.

Maximum Satisfiability. An instance $F=(F_h,F_s,c)$ of the *weighted partial MaxSAT* problem consists of a set F_h of *hard* clauses, a set F_s of *soft* clauses, and a function $c: F_s \to \mathbb{N}$ that associates a non-negative cost (weight) with each of the soft clauses. Any truth assignment τ that satisfies F_h is a *solution* to F. The *cost* of a solution τ to F is

$$\text{COST}(F,\tau)=\sum_{C \in F_s} (1-\tau(C)) \cdot c(C),$$

i.e., the sum of the costs of the soft clauses not satisfied by τ. A solution τ is (globally) *optimal* for F if $\text{COST}(F,\tau) \leq \text{COST}(F,\tau')$ holds for any solution τ' to F. The cost of the optimal solutions of F is denoted by $\text{OPT}(F)$. Given a weighted partial MaxSAT instance F, the weighted partial MaxSAT problem asks to find an optimal solution to F. From here on, we refer to weighted partial MaxSAT instances simply as MaxSAT instances.

An *unsatisfiable core* of a MaxSAT instance $F=(F_h,F_s,c)$ is a subset $F_s' \subseteq F_s$ such that $F_h \cup F_s' \in$ UNSAT. An unsatisfiable core F_s' is *minimal* (MUS) if $F_h \cup F_s'' \in$ SAT for all $F_s'' \subset F_s'$.

3 Labelled CNFs and MaxSAT

The framework of *labelled CNFs* (LCNFs) [Belov and Marques-Silva, 2012; Belov *et al.*, 2013b] allows for generalizing MaxSAT into maximum satisfiability of LCNF, as well as for lifting SAT preprocessing techniques to MaxSAT. Assume a countable set of labels Lbl. A labelled clause C^L consists of a clause C and a (possibly empty) set of labels $L \subseteq Lbl$. A LCNF formula Φ is a set of labelled clauses. We use $Cl(\Phi)$ and $Lbls(\Phi)$ to denote the set of clauses and labels of Φ, respectively. A LCNF formula is satisfiable iff $Cl(\Phi)$ (which is a CNF formula) is satisfiable.

Given a LCNF formula Φ and a subset of its labels $M \subset Lbls(\Phi)$, the subformula $\Phi|_M$ of Φ induced by M is the LCNF formula $\{C^L \in \Phi : L \subset M\}$, i.e., the subformula obtained by removing from Φ all labelled clauses with at least one label not in M. An *unsatisfiable core* of an unsatisfiable LCNF formula Φ is a label-set $L \subset Lbls(\Phi)$ such that (i) the formula $\Phi|_L$ is unsatisfiable, and (ii) if the formula $\Phi|_{L'}$ is satisfiable for all $L' \subset L$, then L is an LMUS. We denote the set of minimal unsatisfiable cores (LMUSes) of Φ by $\text{LMUS}(\Phi)$ A *minimal correction subset* (MCS) for Φ is a label-set $R \subset Lbls(\Phi)$ such that (i) the formula $\Phi|_{Lbls(\Phi)\setminus R}$ is satisfiable, and (ii) the formula $\Phi|_{Lbls(\Phi)\setminus R'}$ is unsatisfiable for all $R' \subset R$.

In a *weighted* LCNF formula Φ, a positive weight w_i is associated with each label in $Lbls(\Phi)$. The cost of a label-set $L \subset Lbls(\Phi)$ is the sum of the weights of labels in L. Given a weighted LCNF formula Φ such that $\Phi|_\emptyset$ is satisfiable, any assignment τ that satisfies $\Phi|_\emptyset$ is a solution to the MaxSAT problem of LCNF formulas. A solution τ is optimal if it satisfies $\Phi|_{Lbls(\Phi)\setminus R}$ for some minimum-cost MCS R of Φ. The cost of τ is the cost of R.

From MaxSAT to Weighted LCNF MaxSAT. A MaxSAT instance $F=(F_h,F_s,c)$ can viewed as a weighted LCNF MaxSAT instance Φ_F 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 $c(C)$. It is easy to see that any optimal solution to Φ_F is an optimal solution to F, and vice versa.

From Weighted LCNF MaxSAT to MaxSAT. A *direct encoding* [Belov *et al.*, 2013b] of a weighted LCNF MaxSAT instance Φ as a MaxSAT instance F_Φ is as follows. Associate with each label $l_i \in Lbls(\Phi)$ a distinct variable a_i, and introduce (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 cost $c(a_i)=w_i$, where w_i is the weight of the label l_i. The resulting instance can then be input to any MaxSAT solver.

3.1 SAT Preprocessing for MaxSAT via LCNFs

Assume that we apply a SAT preprocessing technique P directly on a MaxSAT instance F, not making a distinction between the hard and soft clauses, and perhaps adjusting the weights of the clauses in the resulting MaxSAT instance in some way (weight ∞ implying a hard clause). Following [Belov *et al.*, 2013b], a SAT preprocessing technique P is *sound* for MaxSAT if there is a poly-time computable function α_P such that for any MaxSAT instance F and any optimal solution τ of $P(F)$, $\alpha_P(\tau)$ is an optimal solution of F. As argued in [Belov *et al.*, 2013b], based on the fact that blocked clause elimination does not affect the set of MUSes of any CNF formula [Belov *et al.*, 2013a], it can be shown that BCE is sound for MaxSAT. On the other hand, as shown in [Belov *et al.*, 2013b], directly applying bounded variable elimination, self-subsuming resolution, or even subsumption elimination is not sound.

As a remedy to this problem, in [Belov *et al.*, 2013b] liftings of VE, SSR, and SE to LCNF formulas were proposed. Essentially, 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.

- The resolution rule is lifted to labelled clauses by defining the resolvent $(x \vee A)^{L_1} \bowtie_x (\neg x \vee B)^{L_2}$ of two labelled clauses $(x \vee A)^{L_1}$ and $(\neg x \vee B)^{L_2}$ as $(A \vee B)^{L_1 \cup L_2}$. The rule is lifted to two sets Φ_1 and Φ_2 of labelled clauses analogously to the CNF case.
- Eliminating a variable x then gives the LCNF $(\Phi \setminus (\Phi_x \cup \Phi_{\neg x})) \cup \Phi_x \bowtie_x \Phi_{\neg x}$, resulting a natural lifting of *bounded variable elimination for LCNFs*.
- The *self-subsuming resolution rule for LCNFs*, given two labelled clauses $C_1^{L_1} = (x \vee A)^{L_1}$ and $C_2^{L_2} = (\neg x \vee B)^{L_2}$ such that $A \subset B$ *and* $L_1 \subseteq L_2$ results in the formula $(\Phi \setminus \{C_2^{L_2}\}) \cup B^{L_2}$.
- A labelled clause $C_1^{L_1}$ subsumes $C_2^{L_2}$ if both $C_1 \subset C_2$ *and* $L_1 \subseteq L_2$, which gives the redundancy property used for *subsumption elimination for LCNFs*.

Here it is important to notice that, due to the resolution rule for LCNFs, bounded variable elimination and self-subsuming resolution can cause an increase in the size of the label-sets of the resulting labelled clauses. In particular, consider the encoding of MaxSAT as weighted LCNF MaxSAT. Even though each labelled clause corresponding to a soft clause in the original MaxSAT instance will have a singleton label-set, after LCNF-level preprocessing some of the clauses can have label-sets with more than one label. Direct encoding of the preprocessed weighted LCNF MaxSAT instance as a MaxSAT instance will then add multiple new variables, corresponding to the labels of the labelled clauses, to the resulting soft clauses. Furthermore, we note that in some cases LCNF-level preprocessing may result in a labelled clause $\emptyset^L$, i.e., a labelled clause the actual clause of which is empty.

We further note that, when applying BCE together with VE, SSE, and SE, it makes sense to consider a straightforward lifting of BCE to LCNF formulas to simplify the preprocessing pipeline: a labelled clause C^L is blocked in a LCNF formula Φ if C is blocked in $Cl(\Phi)$. As BCE is sound for MaxSAT, it clear that this lifting is sound for LCNF MaxSAT.

4 Lifting MaxHS to Weighted LCNFs

As discussed in [Belov *et al.*, 2013b], SAT-based preprocessing can be applied in the context of MaxSAT solving using the following observations. (i) By viewing MaxSAT instances as weighted LCNF MaxSAT instances, the LCNF-liftings of VE, SSR, and SE can be soundly applied on MaxSAT instances; and (ii) using the reduction from weighted LCNF MaxSAT to MaxSAT, one can directly employ any MaxSAT solver to obtain solutions to preprocessed weighted LCNF MaxSAT instances, and hence also to the original MaxSAT instances. However, part (ii) in this flow can in cases be non-optimal, especially when applying one of the many SAT-based MaxSAT solvers which use *assumptions* for switching on and off soft clauses from one SAT solver call to another.[1]

An alternative, as done in this work, is to develop liftings of the SAT-based MaxSAT solvers for weighted LCNF MaxSAT. A benefit of doing so is that such liftings can *use the labels of the labelled clauses directly as assumption variables for the SAT solver calls*. By doing this, one avoids both adding the additional soft unit clauses introduced by the direct encoding from weighted LCNF MaxSAT to MaxSAT, as well as the *additional layer of assumption variables* added afterwards by the MaxSAT solver to the soft clauses.

The special nature of preprocessed MaxSAT instances—assumption variables being distributed to multiple clauses, and individual clauses having multiple assumption variables—requires care in how the assumptions are used, which depends on the SAT-based MaxSAT algorithm being lifted to weighted LCNF MaxSAT. In this work we lift the previously proposed MaxHS [Davies and Bacchus, 2011; 2013a; 2013b] algorithm into the LCNF framework. Our motivation for this is that MaxHS—one of the best-performing solvers in the 2014 MaxSAT Evaluation crafted weighted partial category—heavily relies on assumption variables. This makes it a prime candidate for integrating the idea of re-using assumption variables from the preprocessing phase.

MaxHS. An overview of MaxHS is presented as Algorithm 1. MaxHS is a core-guided algorithm that exploits the fact that, when invoked on an unsatisfiable set of clauses, most CDCL SAT solvers can output an unsatisfiable core over the assumption variables used in the solver calls. During execution, MaxHS maintains a collection $\mathcal{C}$ of cores (over the soft clauses) of the input MaxSAT instance $F = (F_h, F_s, c)$. At each iteration, a minimum-cost hitting set H over $\mathcal{C}$ is computed. This hitting set problem is stated over the assumption variables and solved using an IP solver. A SAT solver is then invoked on $F_h \wedge F_s$ with the assumption variables in H set to 1 (and the other assumption variables to 0). If the solver reports *satisfiable*, the algorithm terminates and returns the truth assignment produced by the SAT solver, which is guaranteed to be an optimal solution to F. If the solver reports

[1] Assumptions refer to adding a distinct fresh variable a_i to each of the soft clauses C_i in the input formula. Calling the SAT solver under the assumption $a_i = 1$ is equivalent to *removing* C_i from the instance. Similarly, the assumption $a_i = 0$ switches the clause on.

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

Algorithm 1: The MaxHS algorithm

unsatisfiable, the algorithm obtains a new core C from the SAT solver and reiterates. The intuition behind the algorithm is that when the reduced formula is satisfiable, the found hitting set is also a minimum-cost hitting set over *all* MUSes of F. Hence removing the clauses in the hitting set removes all sources of unsatisfiability from the formula in a minimum-cost manner [Davies and Bacchus, 2011].

MaxHS for Weighted LCNFs. While our lifting of the MaxHS algorithm to LCNFs, LCNF-MaxHS (Alg. 2), closely follows the original MaxHS algorithm, it also makes a critical shift from the clause-centric view (with a single distinct assumption variable for each soft clause) to a label-centric view in which overlapping label-sets with more than one label are allowed. This generalizes MaxHS to LCNFs, while still maintaining correctness (as proven in the following). The unsatisfiable cores on the LCNF-level are explicitly maintained as sets of labels. On each iteration, LCNF-MaxHS checks the satisfiability of the subformula *now induced by* $Lbls(\Phi) \setminus R$ for some minimum-cost hitting set over the collection of identified cores $\mathcal{L}$ of Φ. Notice that inducing a subformula by $Lbls(\Phi) \setminus R$ is analogous to removing all clauses present in the hitting set of the original MaxHS algorithm.

4.1 Correctness

We proceed by a formal correctness proof for LCNF-MaxHS, which relies on the hitting set duality theorem for LCNFs [Belov and Marques-Silva, 2012]. Recall that a hitting set H over an arbitrary collection of sets $\mathcal{S}$ is *irreducible* if no $H' \subset H$ is a hitting set over $\mathcal{S}$.

Theorem 1 *A label-set $M \subset Lbls(\Phi)$ of a LCNF formula Φ is an MCS of Φ iff it is an irreducible hitting set over* $\mathrm{LMUS}(\Phi)$.

The correctness follows from the following.

Proposition 1 *Let Φ be a LCNF formula, $\mathcal{L} \subset \mathcal{P}(Lbls(\Phi))$ a set of its cores, and R a minimum-cost hitting set over $\mathcal{L}$. Assume τ is an assignment satisfying $\Phi|_{Lbls(\Phi)\setminus R}$. Then R is a minimum-cost irreducible hitting set over* $\mathrm{LMUS}(\Phi)$.

Proof. 1) R is a hitting set over $\mathrm{LMUS}(\Phi)$. Otherwise there would be a LMUS M of Φ such that $M \subset Lbls(\Phi) \setminus R$, contradicting the assumption that $\Phi|_{Lbls(\Phi)\setminus R}$ is satisfiable.

2) R is irreducible as any $R' \subset R$ that is a hitting set over $\mathrm{LMUS}(\Phi)$ is also a hitting set over $\mathcal{L}$. As R' contains fewer labels than R, it has to be of lower cost, contradicting the assumed minimum cost (over the hitting sets of $\mathcal{L}$) of R.

3) R has minimum cost over all hitting sets of $\mathrm{LMUS}(\Phi)$ which follows, similarly to case 2, from the fact that any hitting set R'' over $\mathrm{LMUS}(\Phi)$ is also a hitting set over $\mathcal{L}$. Hence R'' has to have at least the same cost as R. □

Input: A weighted LCNF MaxSAT instance Φ
Output: An optimal solution τ for Φ
$\mathcal{L} \leftarrow \emptyset$ // set of found unsat cores of $Lbls(\Phi)$
while *true* **do**
  $R \leftarrow$ MINCOSTHITTINGSET($\mathcal{L}$)
  (*result*, L, τ) $\leftarrow$ SATSOLVE($\Phi|_{Lbls(\Phi)\setminus R}$)
  if *result="satisfiable"* **then**
    return τ // solver returned SAT
  else
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{L\}$ // solver returned unsat core of $Lbls(\Phi)$
  end
end

Algorithm 2: LCNF-MaxHS, lifting of MaxHS to LCNFs

Theorem 2 *The assignment τ returned by the LCNF-MaxHS algorithm is an optimal solution to the weighted MaxSAT problem for LCNFs.*

Proof. By Proposition 1, τ satisfies $\Phi|_{Lbls(\Phi)\setminus R}$ for a minimum-cost irreducible hitting set R over $\mathrm{LMUS}(\Phi)$. By Theorem 1, R is also a minimum-cost MCS of Φ. □

4.2 Integrating SAT-Based Preprocessing and LCNF-MaxHS

Given a MaxSAT instance $F = (F_h, F_s, c)$ as input, the discussed SAT-based preprocessing can be integrated with the lifting of MaxHS to the weighted LCNF setting as follows.

1. Apply the labelled liftings of BCE, VE, SSR, and SE on Φ_F (i.e., F as a weighted LCNF MaxSAT instance), to obtain the preprocessed LCNF Φ'_F.
2. Solve Φ'_F using LCNF-MaxHS.

In practice, the steps above can be implemented, based on the correctness of the LCNF-MaxHS algorithm, by extending the MaxHS algorithm to take as part of the input an explicit listing of assumption variables and modifying the implementation to directly use these assumption variables instead of instrumenting the input soft clauses with new assumption variables. More precisely, we do the following:

1'. Extend each $C_i \in F_s$ with a distinct new assumption variable and apply BCE, VE, SSR, and SE on $F_h \wedge \bigwedge_{C_i \in F_s}(C_i \vee a_i)$, forbiding the removal of any a_i variables during preproccessing. Divide the resulting set of clauses into (i) "hard" clauses F'_h which do not include any of the assumption variables a_i and (ii) "soft" clauses F'_s which each contain *at least one* of the assumption variables.

2'. Apply MaxHS on the MaxSAT instance (F'_h, F'_s, c'), where $c'(a_i) = c(C_i)$ for each $C_i \in F_s$, and explicitly guide MaxHS to work on the a_i variables as the assumption variables.

Notice especially that step 2' avoids adding the soft unit clauses over the assumption variables—produced by the earlier mentioned *direct encoding*—that encode the weights of the clauses to which the assumption variables are added in the direct encoding. This makes a difference when applying the MaxHS algorithm, as explained in the following.

Eq-seeding was proposed in [Davies and Bacchus, 2013a] for improving the efficiency of solving the minimum-cost hitting set problems with IP. In short, eq-seeding uses the fact that each binary clause $(l \vee a_i)$, where a_i is an assumption variable, can actually be viewed as the logical equivalence $l \leftrightarrow \neg a_i$ [Davies and Bacchus, 2013a]. While these logical equivalences are *not* added to the SAT solver, they can be used for deriving additional linear constraints that are added to the hitting set IPs as follows: if for each literal l_j of a clause $C = (l_1 \vee \cdots \vee l_m \vee l_{m+1} \vee \cdots \vee l_n)$ in the MaxSAT instance, either (i) l_j or $\neg l_j$ is equivalent to an assumption variable a_i (i.e., $(l_j \leftrightarrow a_i)$ or $(\neg l_j \leftrightarrow a_i)$); or (ii) l_j is an assumption variable itself (in which case we implicitly have $(l_j \leftrightarrow l_j)$), then replacing l_j by its equivalent assumption variable for each of the variables in C gives a linear at-least-one constraint purely over the assumption variables. These derived linear constraints are added to the IP solver.

An interesting observation here is that eq-seeding within our LCNF-MaxHS implementation can in some cases derive more linear constraints than when invoking the original MaxHS algorithm on the direct encoding after preprocessing.

Example 1 *Assume that, after preprocessing, we have the LCNF formula* $\Phi = \{(\emptyset)^{\{a_1,a_2\}}, (\neg x_1)^{\{a_3\}}, (\neg x_2)^{\{a_4\}}, (x_1 \vee x_2)^{\emptyset}\}$. *For LCNF-MaxHS, these labelled clauses are represented as the clauses* $F = \{(a_1 \vee a_2), (\neg x_1 \vee a_3), (\neg x_2 \vee a_4), (x_1 \vee x_2)\}$ *where the* a_i*s are used as assumption variables. From* $(a_1 \vee a_2)$*, eq-seeding infers the linear constraint* $a_1 + a_2 \geq 1$. *Furthermore, since* a_3 *can be considered equivalent to* x_1*, and* a_4 *to* x_2*, eq-seeding infers* $a_3 + a_4 \geq 1$. *In contrast, consider invoking the original MaxHS algorithm on the direct encoding of* Φ*, i.e., the MaxSAT instance* (F_h, F_s) *with* $F_h = F$ *and* $F_s = \{(\neg a_1), (\neg a_2), (\neg a_3), (\neg a_4)\}$. *Without any knowledge of the fact that the* a_i *variables could be used as assumptions, MaxHS will add to each unit soft clause* $(\neg a_i)$ *a new assumption variable* b_i*, giving the clause* $(\neg a_i \vee b_i)$*, and will hence consider* a_i *to be equivalent to* b_i *for each* i*. From this, eq-seeding can still infer* $b_1 + b_2 \geq 1$*, which is equivalent to* $a_1 + a_2 \geq 1$ *inferred by LCNF-MaxHS. However, eq-seeding in MaxHS will not be able to infer the second linear constraint inferred by eq-seeding within LCNF-MaxHS.*

Special Cases Arising from Preprocessing. Finally, we note an interesting special case arising purely from applying the labelled preprocessing techniques to MaxSAT instances. As already mentioned, in our experiments we often observed labelled clauses of the form $\emptyset^L$, which is equivalent to a MaxSAT clause $\bigvee_{l_i \in L} a_i$, i.e., a clause consisting purely of assumption variables. In fact, in the experiments we report on in the following, we observed that for some benchmarks, preprocessing resulted in instances purely consisting of such clauses. Such clauses can be directly added as the linear constraint $\sum_{l_i \in L} a_i \geq 1$ to the IP solver used for solving the hitting set problems. This is actually done automatically by the eq-seeding technique proposed in [Davies and Bacchus, 2013a] and implemented in our solver.

5 Implementation and Experiments

For implementing the lifting of MaxHS to weighted LCNFs (Alg. 2), we extended our own prototype re-implementation of the MaxHS algorithm for weighted LCNF MaxSAT. Refining Algorithm 1, this re-implementation includes the SAT solver tweaks and disjoint phase of [Davies and Bacchus, 2011], the non-optimal hitting set computations of [Davies and Bacchus, 2013b], as well as the core minimization and eq-seeding techniques of [Davies and Bacchus, 2013a]. MiniSAT 2.2.0 [Eén and Sörensson, 2003] is used as the underlying SAT solver and IBM CPLEX 12.6.0 [Cpl, 2015] is used to solve the minimum-cost hitting set IPs. We extended our MaxHS implementation to take a list of assumption variables as part of the solver input.

As the SAT preprocessor, we used Coprocessor 2.0 [Manthey, 2012] that we modified to add the required assumption variables to the soft clauses, and used its whitelisting feature to forbid removal of any occurrences of the assumption variables during preprocessing. For the experiments reported on, we did not yet integrate Coprocessor into our MaxHS re-implementation. Instead, although somewhat non-optimal in terms of time spent on preprocessing, we called Coprocessor sepaately from the solver, applying BCE, VE, SSR, and SE. The total preprocessing time is included in the running times reported.

We report on experiments using the following solvers.

MHS2.5: The most recent version of the original implementation of the MaxHS algorithm (reference purposes).
MHS: our re-implementation of MaxHS.
MHS+pre: MHS with preprocessing, using the direct encoding after preprocessing.
LMHS+pre: MHS with preprocessing, re-using the assumption variables from preprocessing (i.e., LCNF-MaxHS with preprocessing).
Eva: the best-performing solver in the weighted partial industrial category of MaxSAT Evaluation 2014 [Narodytska and Bacchus, 2014].
Eva+pre: Eva with preprocessing, using the direct encoding after preprocessing.

We used all 624 instances from the weighted partial crafted (214) and industrial (410) categories of MaxSAT Evaluation 2014 (http://www.maxsat.udl.cat/14/). Note that the weighted partial crafted benchmark set contains 310 instances; however, 96 of the instances do not contain hard clauses. The experiments were run on a cluster of 2.53-GHz Intel Xeon quad core machines with 32-GB memory and Ubuntu Linux 12.04. A timeout of 1 h was enforced for solving each benchmark instance.

Results are presented in Figures 1 and 2. Figure 2 shows the number of instances solved for different time limits over all the benchmarks. For example, to solve 500 instances, MHS needs a per-instance timeout of 1500 s, while less than

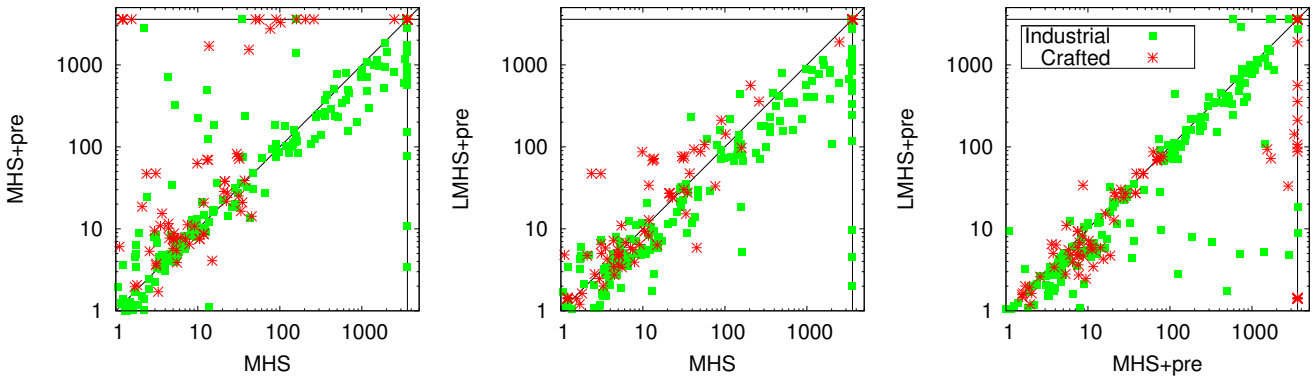

Figure 1: Comparison of MaxHS variants with and without preprocessing, runtimes in seconds. Left: MaxHS w/o preprocessing v MaxHS w/preprocessing using the direct encoding; middle: MaxHS w/o preprocessing v LCNF-MaxHS w/preprocessing; right: MaxHS w/preprocessing using the direct encoding v LCNF-MaxHS w/preprocessing.

500 s suffices for LMHS+pre. Using the direct encoding after preprocessing decreases the performance of MHS, especially on the crafted instances. LMHS+pre clearly improves over the direct encoding and over not using preprocessing at all. Also note that our re-implementation of MaxHS appears to be competitive when compared to the latest version of the original MaxHS solver (MHS2.5), as well as Eva when comparing over all weighted partial instances. Figure 2 also gives some insight into the effect of the individual preprocessing techniques on the performance of LMHS. We ran two sets of experiments, one only using VE, SSR, and SE ("NoBCE") and one only using BCE ("OnlyBCE"). The combination of all techniques resulted in 99 timeouts for LMHS, while using BCE only resulted in 98 and leaving out BCE in 95 timeouts. These differences are due to industrial instances.

Figure 1 gives a pairwise comparison of MHS, MHS+pre, and LMHS+pre. Figure 1 (right) shows that LMHS+pre (preprocessing and re-using assumptions) improves noticeably on MHS+pre (preprocessing and direct encoding). MHS+pre performs noticeably worse than MHS (no preprocessing) on the crafted instances (Figure 1 left). LMHS+pre improves on MHS+pre on these instances (Figure 1 right), with performance closer to that of MHS. On the industrial instances, the results between MHS+pre and MHS are inconclusive. LMHS+pre on the other hand improves somewhat on MHS+pre and more on MHS. LMHS+pre timed out on 99 instances (18 crafted, 81 industrial), MHS on 111 (18, 93). LMHS+pre timed out on only 1 (industrial) instance solved by MHS, while MHS on 13 instances solved by LMHS+pre (all industrial). We also conducted experiments on the unweighted partial crafted benchmarks from the 2014 MaxSAT Evaluation: MHS timed out on 212, (88 of 421 crafted and 124 of 568 industrial), MHS+pre on 280 (129, 151) and LMHS+pre on 204 (84, 120). As a comparison, MHS2.5 timed out on 200 instances (90, 110) and Open-WBO [Martins *et al.*, 2014], one of the best-performing solvers in the 2014 partial industrial track, on 206 (122, 84).

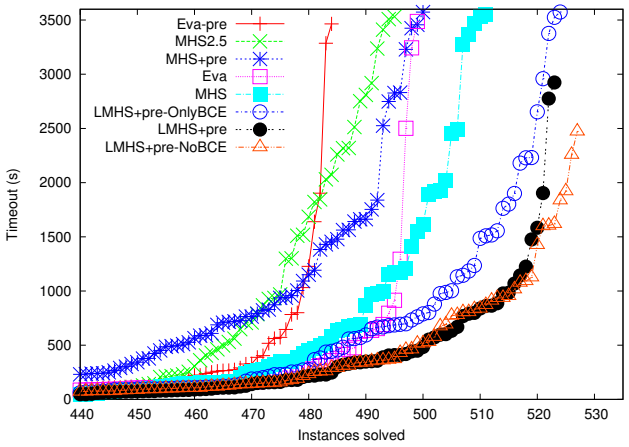

Figure 2: Cactus plot comparing the different MHS variants

6 Conclusions

We presented a lifting of the MaxHS algorithm to labelled LCNFs, enabling a tighter integration of preprocessing and MaxHS via re-using assumption variables from the preprocessing step. We explained how the lifting can be implemented via modifications to MaxHS, and pointed out concrete examples of why assumption re-use can be beneficial. Experiments showed that our LCNF lifting of MaxHS does improve the effectiveness of preprocessing especially on crafted weighted partial MaxSAT, and also improves on the overall performance of MaxHS both with and without direct preprocessing. For future work, an interesting question is if other MaxSAT solvers, such as Eva, could benefit from tighter integration of preprocessing.

Acknowledgments

Work funded by Academy of Finland, grants 251170 COIN Centre of Excellence in Computational Inference Research, 276412, and 284591; and Research Funds of the University of Helsinki.

References

[Ansótegui and Gabàs, 2013] C. Ansótegui and J. Gabàs. Solving (weighted) partial MaxSAT with ILP. In *Proc. CPAIOR*, volume 7874 of *LNCS*, pages 403–409. Springer, 2013.

[Ansótegui *et al.*, 2013] C. Ansótegui, M.L. Bonet, and J. Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196:77–105, 2013.

[Belov and Marques-Silva, 2012] A. Belov and J. Marques-Silva. Generalizing redundancy in propositional logic: Foundations and hitting sets duality. *CoRR*, abs/1207.1257, 2012.

[Belov *et al.*, 2013a] A. Belov, M. Järvisalo, and J. Marques-Silva. Formula preprocessing in MUS extraction. In *Proc. TACAS*, volume 7795 of *LNCS*, pages 108–123. Springer, 2013.

[Belov *et al.*, 2013b] A. Belov, A. Morgado, and J. Marques-Silva. SAT-based preprocessing for MaxSAT. In *Proc. LPAR-19*, volume 8312 of *LNCS*, pages 96–111. Springer, 2013.

[Berg and Järvisalo, 2013] J. Berg and M. Järvisalo. Optimal correlation clustering via MaxSAT. In *Proc. 2013 IEEE ICDM Workshops*, pages 750–757. IEEE Press, 2013.

[Berg *et al.*, 2014] J. Berg, M. Järvisalo, and B. Malone. Learning optimal bounded treewidth bayesian networks via maximum satisfiability. In *Proc. AISTATS*, volume 33, pages 86–95. JMLR, 2014.

[Bunte *et al.*, 2014] Kerstin Bunte, Matti Järvisalo, Jeremias Berg, Petri Myllymäki, Jaakko Peltonen, and Samuel Kaski. Optimal neighborhood preserving visualization by maximum satisfiability. In *Proc. AAAI*, pages 1694–1700. AAAI Press, 2014.

[Cpl, 2015] CPLEX, 2015. `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`.

[Davies and Bacchus, 2011] J. Davies and F. Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *Proc. CP*, volume 6876 of *LNCS*, pages 225–239. Springer, 2011.

[Davies and Bacchus, 2013a] J. Davies and F. Bacchus. Exploiting the power of MIP solvers in MaxSAT. In *Proc. SAT*, volume 7962 of *LNCS*, pages 166–181. Springer, 2013.

[Davies and Bacchus, 2013b] J. Davies and F. Bacchus. Postponing optimization to speed up MAXSAT solving. In *Proc. CP*, volume 8124 of *LNCS*, pages 247–262. Springer, 2013.

[Eén and Biere, 2005] N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proc. SAT*, volume 3569 of *LNCS*, pages 61–75. Springer, 2005.

[Eén and Sörensson, 2003] N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. SAT*, volume 2919 of *LNCS*, pages 502–518. Springer, 2003.

[Guerra and Lynce, 2012] J. Guerra and I. Lynce. Reasoning over biological networks using maximum satisfiability. In *Proc. CP*, volume 7514 of *LNCS*, pages 941–956. Springer, 2012.

[Heras *et al.*, 2011] F. Heras, A. Morgado, and J. Marques-Silva. Core-guided binary search algorithms for maximum satisfiability. In *Proc. AAAI*. AAAI Press, 2011.

[Heule *et al.*, 2010] M. Heule, M. Järvisalo, and A. Biere. Clause elimination procedures for CNF formulas. In *Proc. LPAR-17*, volume 6397 of *LNCS*, pages 357–371. Springer, 2010.

[Järvisalo *et al.*, 2010] M. Järvisalo, A. Biere, and M. Heule. Blocked clause elimination. In *Proc. TACAS*, volume 6015 of *LNCS*, pages 129–144. Springer, 2010.

[Järvisalo *et al.*, 2012] Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In *Proc. IJCAR*, volume 7364 of *LNCS*, pages 355–370. Springer, 2012.

[Jose and Majumdar, 2011] M. Jose and R. Majumdar. Cause clue clauses: error localization using maximum satisfiability. In *Proc. PLDI*, pages 437–446. ACM, 2011.

[Koshimura *et al.*, 2012] M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa. QMaxSAT: A partial Max-SAT solver. *Journal of Satisfiability, Boolean Modeling and Computation*, 8(1/2):95–100, 2012.

[Kullmann, 1999] O. Kullmann. On a generalization of extended resolution. *Discrete Applied Mathematics*, 96-97:149–176, 1999.

[Lagniez and Marquis, 2014] J.-M. Lagniez and P. Marquis. Preprocessing for propositional model counting. In *Proc. AAAI*, pages 2688–2694. AAAI Press, 2014.

[Li and Manyà, 2009] C.M. Li and F. Manyà. MaxSAT, hard and soft constraints. In *Handbook of Satisfiability*, pages 613–631. IOS Press, 2009.

[Manthey, 2012] N. Manthey. Coprocessor 2.0 - A flexible CNF simplifier. In *Proc. SAT*, volume 7317 of *LNCS*, pages 436–441. Springer, 2012.

[Martins *et al.*, 2014] R. Martins, S. Joshi, V.M. Manquinho, and I. Lynce. Incremental cardinality constraints for MaxSAT. In *Proc. CP*, volume 8656 of *LNCS*, pages 531–548. Springer, 2014.

[Morgado *et al.*, 2013] A. Morgado, F. Heras, M.H. Liffiton, J. Planes, and J. Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.

[Morgado *et al.*, 2014] A. Morgado, C. Dodaro, and J. Marques-Silva. Core-guided maxsat with soft cardinality constraints. In *Proc. CP*, volume 8656 of *LNCS*, pages 564–573. Springer, 2014.

[Narodytska and Bacchus, 2014] N. Narodytska and F. Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proc. AAAI*, pages 2717–2723, 2014.

[Zhu *et al.*, 2011] C.S. Zhu, G. Weissenbacher, and S. Malik. Post-silicon fault localisation using maximum satisfiability and backbones. In *Proc. FMCAD*, pages 63–66, 2011.