Re-using Auxiliary Variables for MaxSAT Preprocessing

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—a well-known optimization variant of Boolean satisfiability (SAT)—are finding an increasing number of applications. Preprocessing has proven an integral part of the SAT-based approach to efficiently solving various types of real-world problem instances. It was recently shown that SAT preprocessing for MaxSAT becomes more effective by re-using the auxiliary variables introduced in the preprocessing phase directly in the SAT solver within a core-based hybrid MaxSAT solver. We take this idea of re-using auxiliary variables further by identifying them among variables already present in the input MaxSAT instance. Such variables can be re-used already in the preprocessing step, avoiding the introduction of multiple layers of new auxiliary variables in the process. Empirical results show that by detecting auxiliary variables in the input MaxSAT instances can lead to modest additional runtime improvements when applied before preprocessing. Furthermore, we show that by re-using auxiliary variables not only within preprocessing but also as assumptions within the SAT solver of the MaxHS MaxSAT algorithm can alone lead to performance improvements similar to those observed by applying SAT-based preprocessing.**

I. Introduction

Maximum satisfiability (MaxSAT) [1], [2], [3] is a well-known optimization variant of the archetypical NP-complete problem of Boolean satisfiability (SAT). Building on the extraordinary success of SAT solvers, exact solvers for MaxSAT—and, especially, its weighted partial generalization—are finding an increasing number of applications, ranging e.g. from hardware design debugging and model-based diagnosis to bioinformatics and data analysis [4], [5], [6], [7], [8], [9], [10], [11]. This is brought on by recent improvements in MaxSAT solving techniques [12], [13], [14], [15], [2], [16], [3], [17], [18].

Recently it was shown [19] that SAT-based preprocessing [20], [21] for MaxSAT [22] can be made more effective by explicitly re-using the auxiliary variables introduced in the preprocessing phase directly as the assumption variables the SAT solver within a core-based hybrid MaxSAT solver [23], [14], [15]. In this work, we take this idea of re-using auxiliary variables further. Our idea is to automatically detect (*group detect*) auxiliary variables among the variables already present in the input MaxSAT instance. Such detected auxiliary variables can be used already within the preprocessing step. This avoids introducing layers of new auxiliary variables in the preprocessing and the solving steps. A key motivation for group detection comes from the fact that such auxiliary variables arise naturally when encoding more complex finite-domain soft constraints into MaxSAT via the so-called *Group MaxSAT* framework [24], [25].

We observe that group detection can be achieved by simple pattern matching, and show that this often identifies re-usable auxiliary variables in the weighted partial MaxSAT benchmark sets of the most recent 2014 MaxSAT Evaluation. We also detail why variable re-use via group detection can be beneficial in particular in conjunction with the MaxHS solver. We show that group detection applied before SAT-based preprocessing can bring modest runtime improvements to state-of-the-art MaxSAT solvers.

An additional benefit of group detection is that the detected auxiliary variables can be explicitly re-used as assumptions throughout the whole MaxSAT solving process—not only within SAT-based preprocessing for MaxSAT, but also in the SAT solver within SAT-based MaxSAT algorithms—by explicitly informing the MaxSAT solver of these variables. Using a recently proposed generalization of MaxHS for this purpose [19], we show that this results in overall improvements over MaxHS on weighted partial 2014 MaxSAT Evaluation benchmarks. Furthermore, surprisingly, explicitly re-using group detected variables alone results in similar overall improvements as applying SAT-based preprocessing together with variable re-use of the auxiliary variables necessary for the preprocessing phase.

The rest of the paper is organized as follows. We start with necessary preliminaries on MaxSAT and the group and labelled extensions of MaxSAT (Section II) and SAT-based preprocessing for labelled MaxSAT (Section III). We then detail the proposed approach to re-using variables present in the input MaxSAT instances via what we call group detection (Section IV). After this, we explain why group detection could be beneficial to apply in conjunction with a recently proposed labelled lifting of the MaxHS approach (Section V). Before concluding, we present results of an empirical evaluation on the effects of integrating group detection into the MaxSAT solving process (Section VI).

II. MaxSAT, Groups, and Labels

For a Boolean variable x, there are two literals, x and $\neg x$. A clause is a disjunction ($\vee$) of literals. A truth assignment 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 if there is an assignment τ satisfying all clauses in F ($\tau(F) = 1$), and unsatisfiable ($\tau(F) = 0$ for any assignment τ) otherwise. The Boolean satisfiability problem (SAT) is to decide whether a given CNF formula is satisfiable.

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. 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. A MaxSAT instance with $c(C) = 1$ for all soft clauses C is often called *unweighted*.

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

Group MaxSAT [24], [25] extends MaxSAT by allowing weights on (soft) *groups* of clauses. An instance $F = (F_h, \mathcal{G}_s, c)$ of weighted group MaxSAT consists of a set F_h of hard clauses, a set $\mathcal{G}_s$ of soft groups of clauses, and function $c : \mathcal{G}_s \to \mathbb{N}$ that associates a a non-negative cost with each group in $\mathcal{G}_s$. Each group $G \in \mathcal{G}_s$ is a set of clauses. A truth assignment τ satisfies G iff τ satisfies every clause in G. The Group MaxSAT problem asks to find an assignment τ that satisfies F_h and maximizes the sum of the costs of the groups satisfied by τ.

The framework of *labelled CNFs* (LCNFs) [26], [22] allows for generalizing MaxSAT and Group MaxSAT into maximum satisfiability of LCNF, as well as for lifting SAT preprocessing techniques to 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. An 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. An LCNF formula is satisfiable iff $Cl(\Phi)$ (which is a CNF formula) is satisfiable.

Given an LCNF formula Φ and a subset of its labels $M \subseteq Lbls(\Phi)$, the subformula $\Phi|_M$ of Φ induced by M is the LCNF formula $\{C^L \in \Phi : L \subseteq M\}$, 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 $L \subseteq Lbls(\Phi)$ such that the formula $\Phi|_L$ is unsatisfiable. An unsatisfiable core L is a LMUS iff the formula $\Phi|_{L'}$ is satisfiable for all $L' \subset L$. A *minimal correction subset* (LMCS) for Φ is a label-set $R \subseteq Lbls(\Phi)$ 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 $R' \subset R$.

An instance of the *weighted LCNF-MaxSAT* problem consists of an LCNF formula Φ, with a positive weight w_i associated with each label in $Lbls(\Phi)$. The cost of a label-set $L \subseteq Lbls(\Phi)$ is the sum of the weights of labels in L. Given a weighted LCNF-MaxSAT instance 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 LMCS R of Φ. The cost of τ is the cost of R. Similarly to MaxSAT, we will from here on refer to *weighted* LCNF-MaxSAT instances as LCNF-MaxSAT instances.

A MaxSAT instance $F = (F_h, F_s, c)$ can be viewed as a 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.

An LCNF-MaxSAT instance Φ can be viewed as a MaxSAT instance F_Φ [22] 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 cost $c(a_i) = w_i$, where w_i is the weight of the label l_i. We call this the *direct encoding*. Notice that converting a MaxSAT instance to LCNF and then back to MaxSAT using the direct encoding introduces new variables and clauses to the formula, as exemplified next.

Example 1: Consider the (unweighted) MaxSAT instance $F_{\text{ex}} = (F_h, F_s)$ with $F_h = \{(x \vee z), (\neg z), (y \vee z)\}$ and $F_s = \{(\neg x), (\neg y, \vee \neg z), (z \vee y), (\neg z \vee y)\}$. We will use F_{ex} as a running example in this paper. The assignment τ for which $\tau(x) = \tau(y) = 1$ and $\tau(z) = 0$ is an optimal solution to F_{ex} of cost 1. The set $\{(\neg x)\}$ is an example of a minimal unsatisfiable core of F_{ex}. The LCNF-MaxSAT instance $\Phi_{F_{\text{ex}}}$ corresponding to F_{ex} is

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

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 $L = \{l_1\}$ is a minimal unsatisfiable core of $\Phi_{F_{\text{ex}}}$ as

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

is unsatisfiable. L is also a minimal correction subset to $\Phi_{F_{\text{ex}}}$ as

$$\Phi_{F_{\text{ex}}}|_{Lbls(\Phi_{F_{\text{ex}}})\setminus L} = \{(x \vee z)^\emptyset, (\neg z)^\emptyset, (y \vee z)^\emptyset, (\neg y, \vee \neg z)^{\{l_2\}}, (z \vee y)^{\{l_3\}}, (\neg z \vee y)^{\{l_4\}}\},$$

is a LCNF formula satisfied by τ. As such τ is also an optimal solution to the LCNF-MaxSAT instance $\Phi_{F_{\text{ex}}}$. Converting $\Phi_{F_{\text{ex}}}$ back to a MaxSAT instance using the direct encoding results in the instance $F' = (F'_h, F'_s)$, where

$$F'_h = \{(x \vee z), (\neg z), (y \vee z), (\neg x \vee a_1), (\neg y, \vee \neg z \vee a_2), (z \vee y \vee a_3), (\neg z \vee y \vee a_4)\} \text{ and}$$
$$F'_s = \{(\neg a_1), (\neg a_2), (\neg a_3), (\neg a_4)\}.$$

III. SAT Preprocessing for MaxSAT via LCNFs

Preprocessing has proven an integral part of the SAT-based approach to efficiently solving various types of real-world problem instances. However, to date there is only little work on the effects of preprocessing for MaxSAT, and the benefits of applying SAT-based preprocessing for MaxSAT still remain somewhat unclear. In fact, as shown in [22], many of the commonly used SAT preprocessing techniques, including bounded variable elimination (BVE) [20], self-subsuming resolution (SSR), or even subsumption elimination (SE), cannot be used directly on MaxSAT instances. As a remedy to this problem, in [22] 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.

- *LCNF-lifting of the resolution rule:* The resolvent of two labelled clauses $(x \vee A)^{L_1}$ and $(\neg x \vee B)^{L_2}$ 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 BVE:* Let Φ_x and $\Phi_{\neg x}$, resp., denote the sets of labelled clauses C^L that contain the literal x and the literal $\neg x$, resp. The LCNF-level BVE allows for replacing $\Phi_x \cup \Phi_{\neg x}$ with $\Phi_x \bowtie_x \Phi_{\neg x} = \{A^{L_1} \bowtie_x B^{L_2} \mid A \in \Phi_x,\, B \in \Phi_{\neg x}, A \vee B \text{ not tautological}\}$ as long as the resulting formula does not contain more clauses than the original formula.
- *LCNF-lifting of SE:* A labelled clause A^{L_1} subsumes B^{L_2} if $A \subseteq B$ *and* $L_1 \subseteq L_2$. LCNF-level SE removes subsumed clauses until fixpoint.
- *LCNF-lifting of SSR:* Given labelled clauses $(l \vee A)^{L_1}$ and $(\neg l \vee B)^{L_2}$, if A^{L_1} subsumes B^{L_2}, replace $(\neg l \vee B)^{L_2}$ with B^{L_2}

Example 2: Eliminating z from $\Phi_{F_{\text{ex}}}$ gives the formula

$$\Phi^1_{F_{\text{ex}}} = \{(x)^{\emptyset}, (y)^{\emptyset}, (y)^{\{l_3\}}, (x \vee y)^{\{l_3\}}, (\neg y \vee x)^{\{l_2\}}, (x \vee y)^{\{l_4\}}, (y)^{\{l_4\}}, (y)^{\{l_3,l_4\}}, (\neg x)^{\{l_1\}}\}.$$

Applying labelled SE on $\Phi^1_{F_{\text{ex}}}$ results in the formula

$$\Phi^2_{F_{\text{ex}}} = \{(x)^{\emptyset}, (y)^{\emptyset}, (\neg x)^{\{l_1\}}\}.$$

Eliminating x from $\Phi^2_{F_{\text{ex}}}$ results in the formula

$$pre(\Phi_{F_{\text{ex}}}) = \{()^{\{l_1\}}, (y)^{\emptyset}\}.$$

The label-set $\{l_1\}$ is an LMCS of both $\Phi_{F_{\text{ex}}}$ and $pre(\Phi_{F_{\text{ex}}})$. Notice that the optimal costs of both formulas are the same.

Based on the fact that blocked clause elimination [27], [21] (BCE) does not affect the set of MUSes of any CNF formula [28], BCE is sound for MaxSAT. However, we note that in combination with LCNF-level variable elimination, self-subsuming resolution, and subsumption elimination, it is simpler to consider a straightforward lifting of blocked clause elimination[1] to LCNFs: a labelled clause C^L is *blocked* in an LCNF formula Φ if C is blocked in the CNF formula $Cl(\Phi)$. The soundness of BCE for LCNFs follows from the soundness of BCE for MaxSAT.

Here it is important to notice that, due to the resolution rule for LCNFs, bounded variable elimination can cause an increase in the size of the label-sets of the resulting labelled clauses. In particular, consider the encoding of MaxSAT as 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.

Integration of SAT-based preprocessing into the MaxSAT solving process is outlined in Figure 1, given a weighted partial MaxSAT instance $F = (F_h, F_s, c)$ as input.

For the solving (Step 3), any MaxSAT solver can be used by converting the LCNF back to standard MaxSAT.

[1]More generally, any *monotone* clause elimination procedure [28].

IV. Re-using Auxiliary Variables

Both the assumptions used in MaxSAT solving and the labels which enable SAT-based preprocessing require us to add a layer of new auxiliary variables to the working formula. In this work, we build on the idea of re-using the labels introduced in the SAT-based preprocessing step. We propose to identify variables in the input MaxSAT instance which could be re-used as labels *in the preprocessing step*.

A. Group Detecting Auxiliary Variables

Assume that we are given a weighted partial MaxSAT instance $F = (F_h, F_s, c)$ as input. We observe that MaxSAT instances may already contain variables that can be *directly re-used* as labels during preprocessing and as assumptions by a SAT solver. These variables can be easily identified from the instances by pattern matching. Especially, when viewing F as an LCNF-MaxSAT instance, this allows us to avoid introducing distinct labels for those clauses which contain variables that can be re-used as labels. In this work, we use the following simple scheme to identify such variables. Assume that F contains a literal[2] l that fulfills the following conditions.

(i) $(\neg l) \in F_s$.

(ii) $\neg l \notin C$ for any clause $C \in (F_h \cup F_s) \setminus \{(\neg l)\}$.

(iii) $l \notin C$ for any soft clause $C \in F_s$.

In words, we know that such a literal l only appears in the hard clauses of F and the literal $\neg l$ appears in a single soft unit clause in F. We note that such literals can be easily detected, and call this identification task *group detection*.

Example 3: Consider again the MaxSAT instance F_{ex} from Example 1. In this instance, the literal x satisfies the conditions of group detection and as such can be re-used as a label when converting F_{ex} to an LCNF-MaxSAT instance. This results in the formula

$$\Phi^g_{F_{\text{ex}}} = \{(z)^{\{x\}}, (\neg z)^{\emptyset}, (y \vee z)^{\emptyset}, (\neg y, \vee \neg z)^{\{l_1\}}, (z \vee y)^{\{l_2\}}, (\neg z \vee y)^{\{l_3\}}\}.$$

For concrete motivation for group detection, consider an arbitrary finite-domain constraint $\mathcal{C}$, and let $\mathsf{cnf}(\mathcal{C}) = \bigwedge_{i=1}^{k} C_i$

[2]Note here that a literal can be either a variable or its negation.

1) View $F = (F_h, F_s, c)$ as the LCNF-MaxSAT instance Φ_F as follows.
   - For each $C \in F_h$, introduce a labelled clause $C^{\emptyset}$.
   - For each $C \in F_s$, introduce a labelled clause $C^{\{l_C\}}$ where l_C is a distinct label for C with weight $c(C)$.
2) Apply the LCNF-liftings of BCE, VE, SSR, and SE on Φ_F to obtain the preprocessed LCNF $pre(\Phi_F)$.
3) Solve $pre(\Phi_F)$.

Fig. 1. Integrating SAT preprocessing into MaxSAT solving

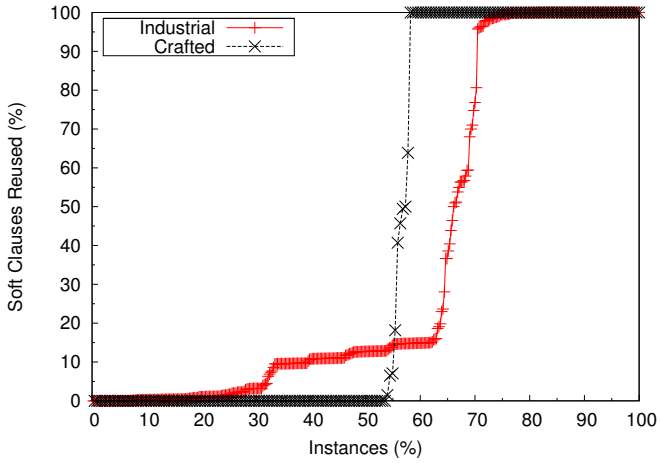

Fig. 2. Labels detected in the weighted partial benchmarks from MaxSAT Evaluation 2014

be a conjunctive normal form encoding of $\mathcal{C}$ (i.e., a representation of $\mathcal{C}$ as a set $\{C_1, \ldots, C_k\}$ of clauses). Now assume that $\mathcal{C}$ is a *soft* constraint, with an associated weight $W_{\mathcal{C}}$ defining the cost of not satisfying $\mathcal{C}$. On the level of Group MaxSAT, the soft constraint $\mathcal{C}$ with weight $W_{\mathcal{C}}$ can be represented as the soft group $\{C_1, \ldots, C_k\}$ with weight $W_{\mathcal{C}}$. For employing a standard MaxSAT solver, a natural way of encoding such a group-level MaxSAT representation [25] is to introduce an auxiliary variable $a_{\mathcal{C}}$, and to consider the set $\{(C_1 \vee a_{\mathcal{C}}), \ldots, (C_k \vee a_{\mathcal{C}})\}$ of *hard clauses* together with the soft clause $(\neg a_{\mathcal{C}})$ with weight $W_{\mathcal{C}}$.

While the proposed group detection procedure is simple, it can relatively often detect variables of interest in real MaxSAT benchmarks. Figure 2 shows the percentages of detected labels out of the number of soft clauses for each instance in the industrial and crafted weighted partial benchmarks from MaxSAT Evaluation 2014. The instances within each of the two categories are sorted by the percentage of detected labels. On a significant percentage of the instances, group detection was able to re-use all of the soft clauses in the input instance. Notice that this is only possible if all of the soft clauses in the input instances are unit, i.e., only contain a single literal.

B. Group Detected Variables as Labels

Group detection allows for re-using the detected "labelling" literals as labels when viewing F as an LCNF-MaxSAT instance. Concretely, we propose the computation steps outlined in Figure 3 as a refinement of the steps outlined in Figure 1. The essential difference here is that, instead of introducing a new label for each soft clause in order to apply SAT preprocessing (as in Figure 1), a new label is only introduced for soft clauses for which Step 0 identified no re-usable variables.

The soundness of group detection is formalized as follows.

Proposition 1: Let $F = (F_h, F_s, c)$ be a MaxSAT instance, Φ_F the LCNF-MaxSAT instance obtained from F following Step 1 in Figure 1, and Φ_F^g the LCNF-MaxSAT instance obtained from F following Steps 0-1 in Figure 3. The cost of the optimal solutions of Φ_F and Φ_F^g are the same.

Proof: (Sketch) We sketch the conversion of a solution τ of Φ_F^g to a solution of Φ_F. Let R be an LMCS such that τ satisfies $\Phi_F^g|_{Lbls(\Phi_F^g)\setminus R}$. Now construct an LMCS R' of Φ_F by including (i) each of the labels $l \in R \cap Lbls(\Phi_F)$, and (ii) for each label in $R \cap (Lbls(\Phi_F^g) \setminus Lbls(\Phi_F))$ (i.e., the group-detected labels), the label of the corresponding soft clause in Φ_F. It is easy to see that R and R' have the same cost. The fact that R' is an LMCS of Φ_F follows by considering the cases (i) and (ii) separately. The less obvious case (ii) follows from each such label in R being a pure literal in $\Phi_F|_{Lbls(\Phi_F)\setminus R'}$. ■

We end this section by noting that there is a connection between group detection and the preprocessing technique of labelled BVE. Consider again the MaxSAT instance F_{ex} from the previous examples. Only eliminating the variable x from $\Phi_{F_{\text{ex}}}$, the direct encoding of F_{ex} in LCNF results in the instance

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

The same LCNF-MaxSAT instance $\Phi_{F_{\text{ex}}}^g$ (modulo label renaming) can also be obtained by encoding F_{ex} as LCNF with group detection.

V. MaxHS for Weighted LCNFs

In order to more thoroughly evaluate group detection, we make use of the LCNF-MaxHS algorithm developed in [19]. This allows us to test the impact of group detection without preprocessing, by re-using detected variables directly within a MaxSAT solver.

MaxHS ([23], [14], [15]) is a recent algorithm for weighted partial MaxSAT. It is a hybrid approach that alternates between

0. Apply group detection on $F = (F_h, F_s, c)$. Assume that group detection is able to identify a set $\mathcal{L}$ of labels associated with a subset $F_h' \subseteq F_h$ of hard clauses.

1'. Convert F into an LCNF-MaxSAT instance Φ_F^g as follows.
- For each $C \in F_h'$, where $L \subseteq C$ for some subset $L \subseteq \mathcal{L}$, introduce the labelled clause $(C \setminus L)^L$. For each label $l \in L$, associate the weight $c((\neg l))$ with l.
- For each $C \in F_h \setminus F_h'$, introduce the labelled clause $C^{\emptyset}$.
- For each $C \in F_s$ that does not contain any literal in $\mathcal{L}$, introduce the labelled clause $C^{\{l_C\}}$, where l_C is a distinct label for C with weight $c(C)$.

2. Apply the labelled liftings of BCE, VE, SSR, and SE on Φ_F^g to obtain the preprocessed LCNF $pre(\Phi_F^g)$.

3. Solve $pre(\Phi_F^g)$.

Fig. 3. Combining group detection, SAT preprocessing, and LCNF-level MaxSAT solving

a SAT solver to compute unsatisfiable cores, and an integer programming (IP) solver to compute minimum-cost hitting sets (MCHS) over the found cores. In short, given a set of cores K for a formula F, MaxHS will invoke the IP solver to find a minimum-cost hitting set hs for K, and the SAT solver to solve the formula $F_h \cup (F_s \setminus hs)$. If the formula is unsatisfiable, a new core κ is derived and added to K and the process is repeated. Otherwise, hs implicitly hits *every* core of F with minimum cost, and the satisfying assignment to $F_h \cup (F_s \setminus hs)$ represents an optimal MaxSAT solution to F.

A. Lifting MaxHS

A lifting of the MaxHS algorithm to LCNFs, LCNF-MaxHS (Algorithm 1), was proposed in [19]. While LCNF-MaxHS closely follows the original MaxHS algorithm, it also makes a critical shift from the clause-centric view (with a single distinct auxiliary variable for each soft clause) to a label-centric view in which overlapping label-sets with more than one label are allowed. In more detail, LCNF-MaxHS maintains a set $\mathcal{L}$ of already identified cores (explicitly maintained on the LCNF-level as subsets of labels from $Lbls(\Phi)$) and a MCHS R for $\mathcal{L}$. During each iteration, an IP solver is used to find a MCHS R of $\mathcal{L}$. A SAT solver is then used to determine the satisfiability of $\Phi|_{Lbls(\Phi)\setminus R}$, the subformula of Φ induced by $Lbls(\Phi)\setminus R$.[3] If satisfiable, an optimal model was produced and the algorithm terminates. Otherwise, a new core L is obtained from the SAT solver, L is added to $\mathcal{L}$, and then the next iteration starts. This generalizes MaxHS to LCNFs while still maintaining correctness [19].

The motivation for LCNF-MaxHS in [19] was to allow for clean integration of SAT-based preprocessing for MaxSAT via LCNFs, and, importantly, to re-use the labels introduced in the preprocessing step directly auxiliary variables which can be used as assumptions within MaxHS. This avoids introducing an additional layer of variables in the SAT solver calls within MaxHS. This is implemented by the steps outlined in Figure 1. For Step 3, LCNF-MaxHS can be applied. In turn, LCNF-MaxHS can be realized by extending an implementation of the MaxHS algorithm to take as input the MaxSAT instance created by the *direct encoding* of $pre(\Phi_F)$. As proposed in [19], the SAT solver within MaxHS can be altered to work directly on the a_i variables as the assumptions, without having to introduce a new layer of auxiliary variables and without having to explicitly add the soft clauses $(\neg a_i)$ to the solver. Step 2 can be implemented using a SAT preprocessor on the direct encoding of Φ_F by restricting the preprocessor from removing any of the a_i variables corresponding to labels.

[3]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.

Input: An 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 1: LCNF-MaxHS, lifting of MaxHS to LCNFs

B. Understanding Group Detection with MaxHS

Eq-seeding was suggested in [14] to improve the effectiveness of the IP solver used in MaxHS. On the LCNF level, eq-seeding takes advantage of the fact that LCNF-MaxHS represents a unit labelled clause $(l)^{\{a_i\}}$ as the unit clause (l) augmented with an auxiliary variable (a_i), i.e., the clause $(l \vee a_i)$. In doing so, an implicit logical equivalence $l \leftrightarrow \neg a_i$ is created [14]. These equivalences need not be added to the SAT solver, but they can sometimes be used to derive linear constraints which can be added to the hitting set IP formulation of MaxHS. Let $C = (l_1 \vee \cdots \vee l_n)$ be a clause in F. A suitable linear constraint can be derived if for every $l_j \in C$, either l_j or $\neg l_j$ is equivalent to an auxiliary variable, or l_j is itself an auxiliary variable. Replacing each l_i with an equivalent auxiliary literal gives a linear at-least-one constraint equivalent to C, which can be added to the hitting set IP used in solving the minimum-cost hitting set problems encountered during search.

The next example demonstrates how group detection can improve the effectiveness of eq-seeding within LCNF-MaxHS. Encoding a MaxSAT instance F as an LCNF-MaxSAT instance Φ_F^g with group detection (via Steps 0–1 of Figure 3) can enable LCNF-MaxHS to derive more linear constraints during solving compared to the direct encoding (Step 1 in Figure 1) of F to Φ_F.

Example 4: Consider the (unweighted) MaxSAT instance $F' = (F'_h, F'_s)$ with

$$\begin{aligned} F'_h =& \{(g_1 \vee x_1), (g_1 \vee x_2), (g_2 \vee x_3), \\ & (g_2 \vee x_4), (\neg x_1 \vee \neg x_3)\} \text{ and} \\ F'_s =& \{(\neg g_1), (\neg g_2)\}. \end{aligned}$$

Converting F into an LCNF-MaxSAT instance $\Phi_{F'}^g$ using group detection results in the LCNF-MaxSAT instance

$$\Phi_{F'}^g = \{(x_1)^{\{g_1\}}, (x_2)^{\{g_1\}}, (x_3)^{\{g_2\}}, (x_4)^{\{g_2\}}, (\neg x_1 \vee \neg x_3)^{\emptyset}\}.$$

During solving, LCNF-MaxHS will treat these labelled clauses as the formula

$$\{(g_1 \vee x_1), (g_1 \vee x_2), (g_2 \vee x_3), (g_2 \vee x_4), (\neg x_1 \vee \neg x_3)\}$$

with auxiliary variables g_1 and g_2. Eq-seeding is able to infer the constraint $g_1 + g_2 \geq 1$ from the clause $(\neg x_1 \vee \neg x_3)$ and the equivalences $x_1 \leftrightarrow \neg g_1$ and $x_3 \leftrightarrow \neg g_2$. On the other hand, a direct encoding of F' in LCNF results in the instance

$$\begin{aligned} &\{(g_1 \vee x_1)^{\emptyset}, (g_1 \vee x_2)^{\emptyset}, (g_2 \vee x_3)^{\emptyset}, (g_2 \vee x_4)^{\emptyset}, \\ &(\neg x_1 \vee \neg x_3)^{\emptyset}, (\neg g_1)^{\{l_1\}}, (\neg g_2)^{\{l_2\}}\} \end{aligned}$$

which will be treated by LCNF-MaxHS as the formula

$$\begin{aligned} &\{(g_1 \vee x_1), (g_1 \vee x_2), (g_2 \vee x_3), (g_2 \vee x_4), \\ &(\neg x_1 \vee \neg x_3), (\neg g_1 \vee l_1), (\neg g_2 \vee l_2)\} \end{aligned}$$

with auxiliary variables l_1 and l_2. Here, eq-seeding will identify the equivalences $g_1 \leftrightarrow l_1$ and $g_2 \leftrightarrow l_2$ but cannot derive any linear constraints suitable for the hitting set IP.

Furthermore, the possibility of deriving more linear constraints can in some instances decrease the number of SAT and IP solver calls required by LCNF-MaxHS.

Example 5: Consider the LCNF-MaxSAT instances $\Phi_{F_{\text{ex}}}$ and $\Phi^g_{F_{\text{ex}}}$ from Examples 1 and 3, respectively. Notice that eq-seeding cannot derive any constraints from $\Phi_{F_{\text{ex}}}$. First we illustrate a possible (worst-case) execution of LCNF-MaxHS on $\Phi_{F_{\text{ex}}}$. Initially, LCNF-MaxHS invokes its SAT solver on the clauses of $\Phi_{F_{\text{ex}}}$. Assume that the SAT solver returns the core $L_1 = \{l_1, l_2\}$. At this point, the set of identified cores only contains L. Assume that the IP solver returns the minimum-cost hitting set $R = \{l_2\}$. Next, LCNF-MaxHS reiterates and invokes the SAT solver on the clauses of $\Phi_{F_{\text{ex}}}|_{Lbls(\Phi)\setminus\{l_2\}}$. The formula is still unsatisfiable. Assume that the SAT solver then returns the core $L_2 = \{l_1, l_3\}$. This time, the only minimum-cost hitting set over the set of all identified cores, $\{L_1, L_2\}$, is $\{l_1\}$. Finally, LCNF-MaxHS invokes the SAT solver on the clauses of $\Phi_F|_{Lbls(\Phi)\setminus\{l_1\}}$. This formula is satisfiable so the algorithm terminates and returns the satisfying assignment returned by the SAT solver. In total, two SAT- and IP-solver calls were needed. In contrast, as the constraint $x = 1$ can be derived from $\Phi^g_{F_{\text{ex}}}$ using eq-seeding, and every unsatisfiable core of $\Phi^g_{F_{\text{ex}}}$ has to include x, LCNF-MaxHS is guaranteed to require only a single SAT and IP solver call when solving $\Phi^g_{F_{\text{ex}}}$.

Group detection allows LCNF-MaxHS to derive more constraints using eq-seeding also in practice. Figure 4 shows a comparison between the number of constraints derivable by eq-seeding with and without group detection from the weighted partial benchmarks of the MaxSAT Evaluation 2014. While no further eq-seeding is obtained on the crafted instances, group detection improves eq-seeding on the industrial instances. A hypothetical explanation for the behavior on crafted instances is offered by the number of labels detected with group detection. As seen from Figure 2, on most crafted instances group detection detects either all soft clauses or no soft clauses. In fact, we observed a clear correlation between no soft clauses detected and no extra constraints derived by eq-seeding: whenever no soft clauses are detected, the LCNF-MaxSAT instances created with and without group detection are the same. For some intuition of the correlation between all soft clauses detected and no extra constraints derived, assume a MaxSAT instance F in which all soft clauses can be group detected. Let Φ^g_F and Φ_F the LCNF-MaxSAT instances created from F with and without group detection, respectively, and $\mathcal{C}$ a linear constraint derivable by eq-seeding from Φ^g_F but not from Φ_F. As discussed earlier, the fact that every soft clause was detected in F means that all soft clauses of F are unit soft clauses of form $(\neg l_i)$, where l_i is a label of Φ^g_F and $(\neg l_i)^{\{a_i\}}$ is a labelled clause of Φ_F. The fact that $\mathcal{C}$ was derivable from Φ^g_F but not from Φ_F suggests that there exists some set of labelled clauses of form $(x_i)^{\{l_i\}}$ and $(\bigvee_i x_i)^{\emptyset}$ in Φ^g_F. Then, the clauses $(x_i)^{\{l_i\}} \in \Phi^g_F$ correspond to clauses $(x_i \vee l_i)^{\emptyset} \in \Phi_F$, explaining why $\mathcal{C}$ cannot be derived from Φ. Such clauses, even though theoretically possible, would seem to only add unnecessary complexity to MaxSAT encodings arising from the real world. In such cases one could in the encoding substitute the (l_i)'s with the x_i variables in F.

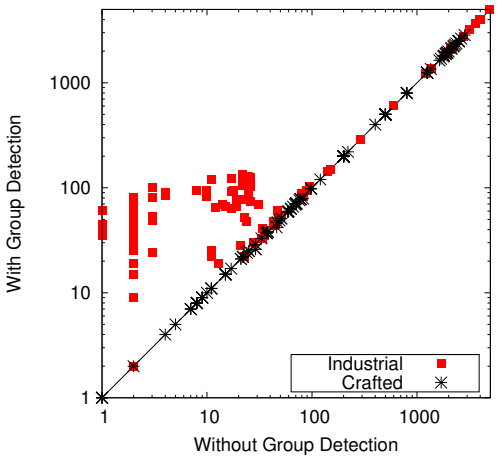

Fig. 4. Comparison of the number of linear constraints derivable by equivalence seeding with and without group detection on the weighted partial benchmarks from MaxSAT Evaluation 2014

VI. Experiments

We overview results from an empirical evaluation, with the aim of understanding the possible benefits of applying group detection in the MaxSAT solving process.

In the evaluation, we used the 410 instances from the weighted partial industrial category of MaxSAT Evaluation 2014 (http://www.maxsat.udl.cat/14/). The experiments were run on 2.53-GHz Intel Xeon quad-core machines with 32-GB RAM and Ubuntu Linux 12.04. A per-instance timeout of 1 h and memory limit of 30 GB were enforced. As the MaxSAT solvers, we used Eva [29], an award-winning core-based MaxSAT solver from the industrial weighted partial track of the 2014 MaxSAT Evaluation; and our own re-implementation of MaxHS that also enables the lifting of MaxHS to weighted LCNFs (Algorithm 1). This re-implementation includes the SAT solver tweaks and disjoint phase of [23], the non-optimal hitting set computations of [15], as well as the core minimization and eq-seeding techniques of [14]. MiniSAT 2.2.0 [30] is used as the underlying SAT solver, and IBM CPLEX 12.6.0 [31] is used to solve the minimum-cost hitting set IPs. For realizing LCNF-MaxHS, we extended our MaxHS implementation to take as input a list of variables suitable for use as assumptions. As the SAT preprocessor, we used Coprocessor 2.0 [32]. Its file parser was modified to automatically detect literals that can be re-used as labels, to add new auxiliary variables (labels) to the other soft clauses, and then using its whitelisting feature to forbid removal of all occurrences of every variable that represents a label during preprocessing.

We report on using the following preprocessing+solver combinations.

Eva: the Eva solver.

Eva-pre: Eva solver with preprocessing, using the direct encoding after preprocessing.

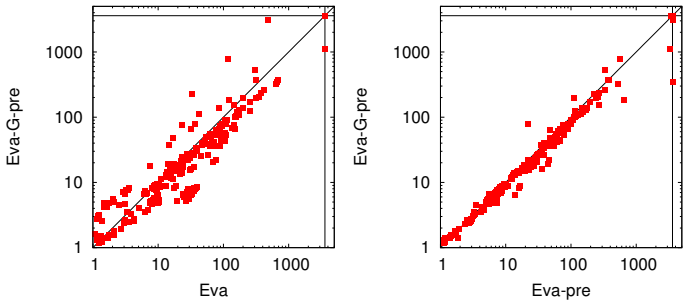

Fig. 5. Running times: Eva v Eva-G-pre (left); Eva-pre v Eva-G-pre (right).

Eva-G-pre: Eva solver, with group detection. Detected variables are re-used in preprocessing as labels. The direct encoding is used after preprocessing.

MHS2.5: MaxHS version 2.5 by the original MaxHS authors (http://www.maxhs.org/).

MHS: our re-implementation of MaxHS.

MHS-pre: MHS with preprocessing, using the direct encoding after preprocessing.

MHS-G-pre: MHS-pre, with group detection. Detected variables are re-used in preprocessing as labels. The direct encoding is used after preprocessing.

LMHS-G: MHS with group detection (i.e., LCNF-MaxHS with group detection). Detected variables are re-used as assumptions in the solver.

LMHS-pre: MHS with preprocessing, re-using auxiliary variables from preprocessing as assumptions in the solver.

LMHS-G-pre: MHS with group detection and preprocessing. Group detected variables are re-used as labels in preprocessing. Auxiliary variables from preprocessing are re-used as assumptions in the solver.

Result for Eva are shown in Figure 5. While no substantial differences in overall performance are observed, Eva-G-pre solves a majority of the instances faster than Eva, and timeouts on one less instance. Comparing Eva-G-pre to Eva-pre, i.e., looking at the additional effect of group detection in conjunction with preprocessing, we observe again that Eva-G-pre solves most instances slightly faster than Eva-pre, and Eva-pre timeouts on two more instances.

Results for variants of MHS are presented in Figures 6 and 7. We note that our MaxHS re-implementation is competitive with MaxHS 2.5. Second, we observe that employing SAT preprocessing in combination with plain MHS without re-using labels from the preprocessing step (MHS-pre) improves performance. Adding group detection (MHS-G-pre) improves further on MHS-pre. Perhaps the most interesting observation is that LMHS-G, i.e., applying group detection *solely* (i.e., without preprocessing) in conjunction with LCNF-MaxHS, is surprisingly effective; see also Figure 7. In more detail, first note that the MHS solver in Figure 6 could equivalently be viewed as LMHS without preprocessing or group detection, that is, Steps 1 and 3 of the computation outlined in Figure 1. As such, comparing the performance of MHS and LMHS-G gives an indication of the effect group detection alone has on MaxSAT solving. We see that group detection (LMHS-G) performs noticeably better than MHS, solving 16 instances more. One possible explanation for this performance—as discussed earlier—is eq-seeding; eq-seeding within LMHS-G was indeed able to derive more linear constraints than MHS on every single one of the instances that were solvable by LMHS-G but not MHS. Furthermore, on 92% of all benchmarks we observed that if LMHS-G solved the instance quicker than MHS, then it also needed fewer UNSAT cores. This further suggests that the reason for the difference in performance between MHS and LMHS-G is linked to the quality of the hitting sets returned by the IP solver. The connection between group detection and preprocessing is less clear. Comparing LMHS-G and LMHS-pre we surprisingly observe a similar level of performance, suggesting that group detection and preprocessing have similar effects on MaxSAT solving—although, a slight further performance increase is observed when combining the two (LMHS-G-pre).

VII. Conclusions

We proposed automatically detecting auxiliary variables from input MaxSAT instances that can be re-used in the SAT-based preprocessing step before MaxSAT solving. This further avoids adding unnecessary layers of auxiliary variables throughout the MaxSAT solving process both in preprocessing and also within the SAT-solver used in MaxSAT solvers. We empirically showed that such auxiliary variables can indeed be detected in real MaxSAT benchmarks, and that re-using these variables as assumptions gives—somewhat surprisingly—similar improvements on its own as applying SAT preprocessing on industrial weighted partial MaxSAT instances in terms of solving efficiency. As future work, we aim at studying ways of detecting and soundly exploiting auxiliary variables arising from clausal encodings of more complex soft constraints.

Acknowledgements

Work presented in this paper was funded by Academy of Finland, grants 251170 Centre of Excellence in Computational Inference Research, 276412, and 284591; Research Funds of the University of Helsinki; and the Emil Aaltonen Foundation.

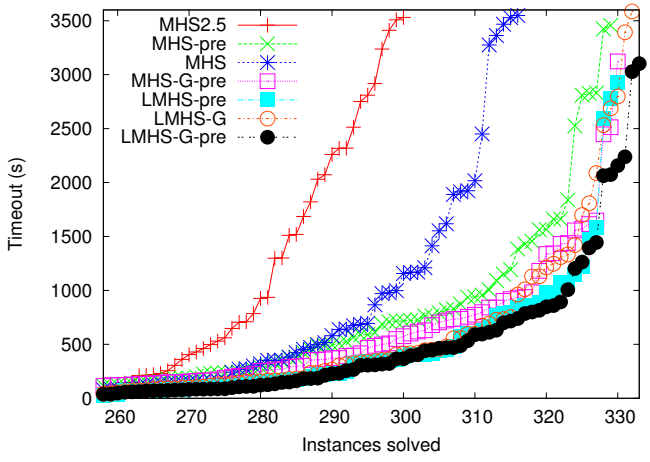

Fig. 6. Comparison of the different MaxHS variants

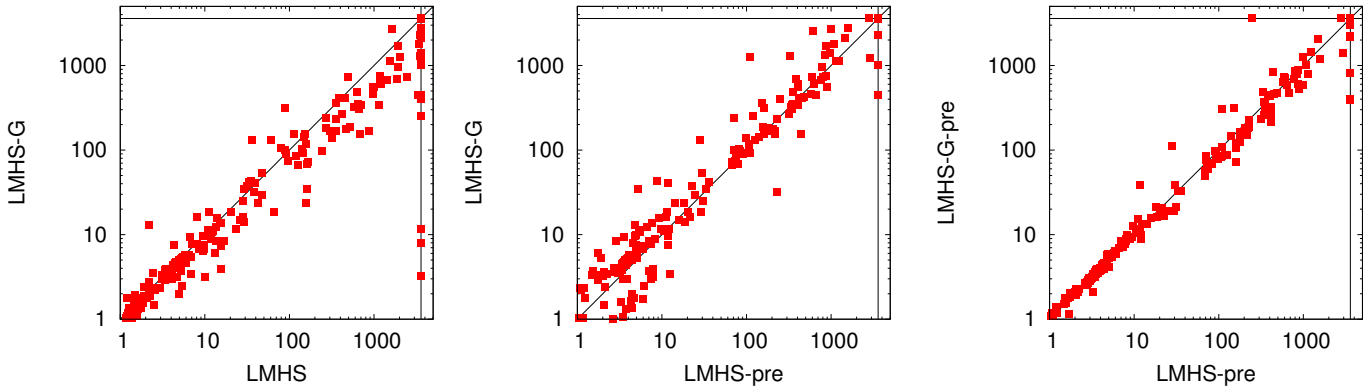

Fig. 7. Effect of group detection on runtimes: LMHS v LMHS-G (left); LMHS-G v LMHS-pre (middle); LMHS-pre v LMHS-G-Pre (right).

REFERENCES

[1] C. Li and F. Manyà, "MaxSAT, hard and soft constraints," in *Handbook of Satisfiability*. IOS Press, 2009, pp. 613–631.

[2] A. Morgado, F. Heras, M. Liffiton, J. Planes, and J. Marques-Silva, "Iterative and core-guided MaxSAT solving: A survey and assessment," *Constraints*, vol. 18, no. 4, pp. 478–534, 2013.

[3] C. Ansótegui, M. Bonet, and J. Levy, "SAT-based MaxSAT algorithms," *Artificial Intelligence*, vol. 196, pp. 77–105, 2013.

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

[5] C. Zhu, G. Weissenbacher, and S. Malik, "Post-silicon fault localisation using maximum satisfiability and backbones," in *Proc. FMCAD*. FMCAD Inc., 2011, pp. 63–66.

[6] J. Guerra and I. Lynce, "Reasoning over biological networks using maximum satisfiability," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. 7514. Springer, 2012, pp. 941–956.

[7] J. Berg, M. Järvisalo, and B. Malone, "Learning optimal bounded treewidth bayesian networks via maximum satisfiability," in *Proc. AISTATS*, vol. 33. JMLR, 2014, pp. 86–95.

[8] K. Bunte, M. Järvisalo, J. Berg, P. Myllymäki, J. Peltonen, and S. Kaski, "Optimal neighborhood preserving visualization by maximum satisfiability," in *Proc. AAAI*. AAAI Press, 2014, pp. 1694–1700.

[9] J. Marques-Silva, M. Janota, A. Ignatiev, and A. Morgado, "Efficient model based diagnosis with maximum satisfiability," in *Proc. IJCAI*. AAAI Press, 2015.

[10] P. Saikko, B. Malone, and M. Järvisalo, "MaxSAT-based cutting planes for learning graphical models," in *Proc. CPAIOR*, ser. Lecture Notes in Computer Science, vol. 9075. Springer, 2015, pp. 345–354.

[11] J. Berg and M. Järvisalo, "Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability," *Artificial Intelligence*, 2015, in press.

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

[13] M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa, "QMaxSAT: A partial Max-SAT solver," *Journal of Satisfiability, Boolean Modeling and Computation*, vol. 8, no. 1/2, pp. 95–100, 2012.

[14] J. Davies and F. Bacchus, "Exploiting the power of MIP solvers in MaxSAT," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 7962. Springer, 2013, pp. 166–181.

[15] ——, "Postponing optimization to speed up MAXSAT solving," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. 8124. Springer, 2013, pp. 247–262.

[16] C. Ansótegui and J. Gabàs, "Solving (weighted) partial MaxSAT with ILP," in *Proc. CPAIOR*, ser. Lecture Notes in Computer Science, vol. 7874. Springer, 2013, pp. 403–409.

[17] A. Morgado, C. Dodaro, and J. Marques-Silva, "Core-guided MaxSAT with soft cardinality constraints," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. 8656. Springer, 2014, pp. 564–573.

[18] R. Martins, S. Joshi, V. Manquinho, and I. Lynce, "Incremental cardinality constraints for MaxSAT," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. 8656. Springer, 2014, pp. 531–548.

[19] J. Berg, P. Saikko, and M. Järvisalo, "Improving the effectiveness of SAT-based preprocessing for MaxSAT," in *Proc. IJCAI*. AAAI Press, 2015.

[20] N. Eén and A. Biere, "Effective preprocessing in SAT through variable and clause elimination," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 3569. Springer, 2005, pp. 61–75.

[21] M. Järvisalo, A. Biere, and M. Heule, "Blocked clause elimination," in *Proc. TACAS*, ser. Lecture Notes in Computer Science, vol. 6015. Springer, 2010, pp. 129–144.

[22] A. Belov, A. Morgado, and J. Marques-Silva, "SAT-based preprocessing for MaxSAT," in *Proc. LPAR-19*, ser. Lecture Notes in Computer Science, vol. 8312. Springer, 2013, pp. 96–111.

[23] J. Davies and F. Bacchus, "Solving MAXSAT by solving a sequence of simpler SAT instances," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. 6876. Springer, 2011, pp. 225–239.

[24] J. Argelich and F. Manyà, "Exact Max-SAT solvers for over-constrained problems," *Journal of Heuristics*, vol. 12, no. 4-5, pp. 375–392, 2006.

[25] F. Heras, A. Morgado, and J. Marques-Silva, "MaxSAT-based encodings for Group MaxSAT," *AI Communications*, vol. 28, no. 2, pp. 195–214, 2015.

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

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

[28] A. Belov, M. Järvisalo, and J. Marques-Silva, "Formula preprocessing in MUS extraction," in *Proc. TACAS*, ser. Lecture Notes in Computer Science, vol. 7795. Springer, 2013, pp. 108–123.

[29] N. Narodytska and F. Bacchus, "Maximum satisfiability using core-guided MaxSAT resolution," in *Proc. AAAI*. AAAI Press, 2014, pp. 2717–2723.

[30] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 2919. Springer, 2003, pp. 502–518.

[31] IBM, "CPLEX Optimizer," 2015, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[32] N. Manthey, "Coprocessor 2.0 - A flexible CNF simplifier," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 7317. Springer, 2012, pp. 436–441.