Exploiting Cycle Structures in Max-SAT

Chu Min Li[1], Felip Manyà[2], Nouredine Mohamedou[1], and Jordi Planes[3,*]

[1] MIS, Université de Picardie Jules Verne, 5 Rue du Moulin Neuf 80000 Amiens, France
[2] IIIA-CSIC, Campus UAB, 08193 Bellaterra Spain
[3] Computer Science Department, Universitat de Lleida, Jaume II, 69, 25001 Lleida, Spain

Abstract. We investigate the role of cycles structures (i.e., subsets of clauses of the form $\bar{l}_1 \vee l_2, \bar{l}_1 \vee l_3, \bar{l}_2 \vee \bar{l}_3$) in the quality of the lower bound (LB) of modern MaxSAT solvers. Given a cycle structure, we have two options: (i) use the cycle structure just to detect inconsistent subformulas in the underestimation component, and (ii) replace the cycle structure with $\bar{l}_1, l_1 \vee \bar{l}_2 \vee \bar{l}_3, \bar{l}_1 \vee l_2 \vee l_3$ by applying MaxSAT resolution and, at the same time, change the behaviour of the underestimation component. We first show that it is better to apply MaxSAT resolution to cycle structures occurring in inconsistent subformulas detected using unit propagation or failed literal detection. We then propose a heuristic that guides the application of MaxSAT resolution to cycle structures during failed literal detection, and evaluate this heuristic by implementing it in MaxSatz, obtaining a new solver called MaxSatz$_c$. Our experiments on weighted MaxSAT and Partial MaxSAT instances indicate that MaxSatz$_c$ substantially improves MaxSatz on many hard random, crafted and industrial instances.

1 Introduction

The lower bound (LB) computation method implemented in branch and bound MaxSAT solvers (e.g. [4,6,10,12,13]) is decisive for obtaining a competitive solver. The LB of MaxSatz [10] and MiniMaxSat [4] —two of the best performing solvers in the 2008 MaxSAT Evaluation— has two components: (i) the *underestimation component*, which detects disjoint inconsistent subformulas and takes the number of detected subformulas as an underestimation of the LB, and (ii) the *inference component*, which applies inference rules and, in the best case, makes explicit a contradiction by deriving an empty clause which allows to increment the LB. Both components are applied at each node of the search space, and cooperate rather than work independently.

A MaxSAT instance may contain different structures that influence the behavior of the two components of the LB. In this paper we investigate the role of the so-called cycles structures (i.e., subsets of clauses of the form $\bar{l}_1 \vee l_2, \bar{l}_1 \vee l_3, \bar{l}_2 \vee \bar{l}_3$)[1] in the quality of the LB. Given a cycle structure, we have two options: (i) use the cycle structure just to detect inconsistent subformulas in the underestimation component

---

[1] $\bar{l}_1 \vee l_2, \bar{l}_1 \vee l_3, \bar{l}_2 \vee \bar{l}_3$ is equivalent to $l_1 \rightarrow l_2, l_1 \rightarrow l_3, \bar{l}_2 \vee \bar{l}_3$.

(note that a cycle structure implies a failed literal l_1), and (ii) replace the cycle structure with $\bar{l}_1, l_1 \vee \bar{l}_2 \vee \bar{l}_3, \bar{l}_1 \vee l_2 \vee l_3$ by applying MaxSAT resolution [2,5], which amounts to activate the inference component and, at the same time, change the behaviour of the underestimation component. We first show that it is better to apply MaxSAT resolution to cycle structures occurring in inconsistent subformulas detected using unit propagation or failed literal detection. We then propose a heuristic that guides the application of MaxSAT resolution to cycle structures during failed literal detection, and evaluate this heuristic by implementing it in MaxSatz, obtaining a new solver called MaxSatz$_c$. Our experimental investigation on weighted MaxSAT and Partial MaxSAT instances shows that MaxSatz$_c$ substantially improves MaxSatz on many hard random, crafted and industrial instances.

This paper extends the results of [7] to Weighted MaxSAT and Partial MaxSAT, and includes experiments with random, crafted and industrial instances of the last MaxSAT Evaluation (in [7], the results are only for unweighted MaxSAT, and experiments are limited to Max-2SAT instances). The implementations in [7] were performed on top of an optimized version of MaxSatz for unweighed MaxSAT that was first used in the 2007 MaxSAT Evaluation, but the implementations of this paper were performed on top of an optimized version of MaxSatz for Weighted Partial MaxSAT that was used in the 2008 MaxSAT Evaluation. This paper also contains two new lemmas (Lemma 1 and Lemma 2), a formal proof of Proposition 1, an example (Example 2) that shows that applying MaxSAT resolution to cycle structures not contained in an inconsistent subformula may lead to worse LBs, and a deeper analysis of the experimental results. For the sake of clarity, we explain our work for unweighted MaxSAT, but the implementation and experiments include Weighted MaxSAT and Partial MaxSAT.

2   Preliminaries

We define CNF formulas as multisets of clauses because, in Max-SAT, duplicated clauses cannot be collapsed into one clause, and define weighted CNF formulas as multisets of weighted clauses. A weighted clause is a pair (C_i, w_i), where C_i is a disjunction of literals and w_i, its weight, is a positive number. The weighted clauses $(C, w_i), (C, w_j)$ can be replaced with $(C, w_i + w_j)$. A literal l in a (weighted) CNF formula ϕ is *failed* if unit propagation derives a contradiction from $\phi \wedge l$ but not from ϕ. An empty clause cannot be satisfied and is denoted by $\square$.

The *(Unweighted) MaxSAT problem* for a CNF formula ϕ is the problem of finding a truth assignment that maximizes (minimizes) the number of satisfied (unsatisfied) clauses.[2] MaxSAT instances ϕ_1 and ϕ_2 are equivalent if ϕ_1 and ϕ_2 have the same number of unsatisfied clauses for every complete assignment of ϕ_1 and ϕ_2. A MaxSAT inference rule is sound if it transforms an instance into an equivalent instance.

The *Weighted MaxSAT problem* for a weighted CNF formula ϕ is the problem of finding an assignment that minimizes the sum of weights of unsatisfied clauses. A *Partial MaxSAT* instance is a CNF formula in which some clauses are *relaxable* or *soft* and the rest are *non-relaxable* or *hard*. Solving a Partial MaxSAT instance amounts to find an assignment that satisfies all the hard clauses and the maximum number of soft clauses.

---

[2] In the sequel, we always refer to the minimization version of MaxSAT, also called MinUNSAT.

3   Related Work

3.1   Underestimation Component

The underestimation in LB UP [8] is the number of disjoint inconsistent subformulas that can be detected with unit propagation. UP works as follows: it applies unit propagation until a contradiction is derived. Then, UP identifies, by inspecting the implication graph created by unit propagation, a subset of clauses from which a unit refutation can be constructed, and tries to derive new contradictions from the remaining clauses. The order in which unit clauses are propagated has a clear impact on the quality of the LB [9]. Recently, Darras et al. [3] and Han et al. [11] have developed two versions of UP in which the computation of the LB is made more incremental.

UP can be enhanced with failed literal detection (UP_{FL}) [9] : Given a MaxSAT instance ϕ *to which we have already applied UP*, and a variable x, UP_{FL} applies UP to $\phi \wedge x$ and $\phi \wedge \bar{x}$. If UP derives a contradiction from both $\phi \wedge x$ and $\phi \wedge \bar{x}$, then the union of the two inconsistent subformulas identified by UP respectively in $\phi \wedge x$ and $\phi \wedge \bar{x}$ is an inconsistent subformula of ϕ, after excluding x and $\bar{x}$. Since applying failed literal detection to every variable is time consuming, it is only applied to the variables which do not occur in unit clauses, and have at least two positive and two negative occurrences in binary clauses. Once an inconsistent subformula γ is detected, γ is removed from ϕ, the underestimation is increased by one, and UP_{FL} continues in the modified ϕ.

In this paper, when we say an inconsistent subformula, we mean an inconsistent subformula detected using unit propagation or failed literal detection.

Another approach for computing underestimations is based on first reducing the MaxSAT instance one wants to solve to an instance of another problem, and then solving a relaxation of the obtained instance. For example, Clone [12] and SR(w) [13] solve the minimum cardinality problem of a deterministic decomposable negation normal form (d-DNNF) compilation of a relaxation of the current MaxSAT instance.

3.2   Inference Component

An alternative to improve the quality of the LB consists in applying MaxSAT resolution. In practice, competitive solvers apply some refinements of the rule for efficiency reasons. In contrast to SAT resolution, a MaxSAT inference rule replaces the clauses in the premises with the clauses in the conclusion in order to preserve the number of unsatisfied clauses. If the conclusion would be added to the premises as in SAT resolution, the number of unsatisfied clauses might increase.

MaxSatz [10] incorporates the following rules (also called Rule 1, Rule 2, Rule 3, Rule 4 in this paper) capturing special structures in a MaxSAT instance:

$$l_1, \ \bar{l}_1 \vee \bar{l}_2, \ l_2 \implies \Box, \ l_1 \vee l_2 \tag{1}$$

$$l_1, \ \bar{l}_1 \vee l_2, \ \bar{l}_2 \vee l_3, \cdots, \ \bar{l}_k \vee l_{k+1}, \ \bar{l}_{k+1} \implies \Box, \ l_1 \vee \bar{l}_2, \ l_2 \vee \bar{l}_3, \cdots, \ l_k \vee \bar{l}_{k+1} \tag{2}$$

$$l_1, \ \bar{l}_1 \vee l_2, \ \bar{l}_1 \vee l_3, \ \bar{l}_2 \vee \bar{l}_3 \implies \Box, \ l_1 \vee \bar{l}_2 \vee \bar{l}_3, \ \bar{l}_1 \vee l_2 \vee l_3 \tag{3}$$

$$l_1, \; \bar{l}_1 \vee l_2, \; \bar{l}_2 \vee l_3, \cdots, \; \bar{l}_k \vee l_{k+1}, \qquad \Longrightarrow \qquad \square, \; l_1 \vee \bar{l}_2, \; l_2 \vee \bar{l}_3, \cdots, \; l_k \vee \bar{l}_{k+1},$$
$$\bar{l}_{k+1} \vee l_{k+2}, \; \bar{l}_{k+1} \vee l_{k+3}, \; \bar{l}_{k+2} \vee \bar{l}_{k+3} \qquad\qquad l_{k+1} \vee \bar{l}_{k+2} \vee \bar{l}_{k+3}, \; \bar{l}_{k+1} \vee l_{k+2} \vee l_{k+3}$$

$$\tag{4}$$

Max-DPLL [6] incorporates several rules for weighted MaxSAT, including chain resolution (which is equivalent to Rule 2 in the unweighted case) and cycle resolution. Cycle resolution, which captures the cycle structure, is implemented for 3 variables:

$$\bar{l}_1 \vee l_2, \; \bar{l}_1 \vee l_3, \; \bar{l}_2 \vee \bar{l}_3 \implies \bar{l}_1, \; l_1 \vee \bar{l}_2 \vee \bar{l}_3, \; \bar{l}_1 \vee l_2 \vee l_3 \tag{5}$$

MiniMaxSat incorporates LB UP and, once a contradiction is found, it applies MaxSAT resolution to the detected inconsistent subformula if the largest resolvent in the refutation has arity less than 4; otherwise, it just increments the underestimation.

Max-DPLL applies MaxSAT resolution, via the cycle resolution inference rule, to all the cycle structures occurring in a MaxSAT instance, and does not combine its application with the underestimation component. MaxSatz and MiniMaxSat both select cycle structures to which MaxSAT resolution can be applied. MaxSatz applies MaxSAT resolution, via Rule 3 and Rule 4, just when unit propagation detects a contradiction containing the cycle structure. MiniMaxSat applies MaxSAT resolution to cycles structures which are contained in an inconsistent subformula detected by UP provided that the largest resolvent in the refutation of the subformula has arity less than 4.

4   Cycle Structures and Lower Bounds

Exploiting cycle structures has proved very useful in Max-DPLL, MaxSatz, and Mini-MaxSat. In this section, we study why and when exploiting cycle structures is useful.

The operation of Rule 3 and Rule 4 of MaxSatz can be analyzed as follows. Given an inconsistent subformula $\{l_1, \; \bar{l}_1 \vee l_2, \; \bar{l}_1 \vee l_3, \; \bar{l}_2 \vee \bar{l}_3\}$ detected using unit propagation, Rule 3 applies MaxSAT resolution to transform the subformula into $\{l_1, \bar{l}_1, \; l_1 \vee \bar{l}_2 \vee \bar{l}_3, \; \bar{l}_1 \vee l_2 \vee l_3\}$, and then into $\{\square, \; l_1 \vee \bar{l}_2 \vee \bar{l}_3, \; \bar{l}_1 \vee l_2 \vee l_3\}$ (since $\{l_1, \bar{l}_1\}$ is equivalent to $\square$). The benefit of the transformation is twofold: (i) the empty clause does not need to be re-detected in the subtree rooted at the current node because it remains in the transformed formula, and (ii) the transformed subformula includes two new ternary clauses, and such *liberated* clauses may be used to detect further inconsistent subformulas, allowing to compute better LBs. The case of Rule 4 is similar.

The next example illustrates the usefulness of applying MaxSAT resolution to cycle structures in scenarios where there is no unit clause.

Example 1. Assume that a MaxSAT instance ϕ contains

$$x_1 \vee x_2, \; \bar{x}_2 \vee x_3, \; \bar{x}_2 \vee x_4, \; \bar{x}_3 \vee \bar{x}_4, \; \bar{x}_1 \vee x_5, \; \bar{x}_5 \vee x_6, \; \bar{x}_1 \vee x_7, \; \bar{x}_6 \vee \bar{x}_7$$
$$x_8 \vee \bar{x}_2, \; x_8 \vee x_3, \; x_8 \vee x_4, \; \bar{x}_8 \vee x_9, \; \bar{x}_8 \vee x_{10}, \; \bar{x}_8 \vee x_{11}, \; \bar{x}_9 \vee \bar{x}_{10} \vee \bar{x}_{11}$$

Rule 3 and Rule 4 are not applied since there is no unit clause. Failed literal detection on the variable x_1 finds the inconsistent subformula in the first line. After removing this

subformula, it cannot detect further inconsistent subformulas. The underestimation is only incremented by 1. However, if MaxSAT resolution is applied to

$$\bar{x}_2 \vee x_3, \ \bar{x}_2 \vee x_4, \ \bar{x}_3 \vee \bar{x}_4$$

in the first line, these clauses are replaced with

$$\bar{x}_2, \ x_2 \vee \bar{x}_3 \vee \bar{x}_4, \ \bar{x}_2 \vee x_3 \vee x_4$$

and then the underestimation component detects 2 inconsistent subformulas instead of 1. The first with failed literal detection on the variable x_1:

$$x_1 \vee x_2, \ \bar{x}_2, \ \bar{x}_1 \vee x_5, \ \bar{x}_5 \vee x_6, \ \bar{x}_1 \vee x_7, \ \bar{x}_6 \vee \bar{x}_7$$

and the second with failed literal detection on the variable x_8:

$$x_8 \vee \bar{x}_2, \ x_8 \vee x_3, \ x_8 \vee x_4, \ \bar{x}_8 \vee x_9, \ \bar{x}_8 \vee x_{10}, \ \bar{x}_8 \vee x_{11}, \ \bar{x}_9 \vee \bar{x}_{10} \vee \bar{x}_{11}, \ x_2 \vee \bar{x}_3 \vee \bar{x}_4$$

Example 1, together with the analysis of Rule 3 and Rule 4, suggests that one should apply MaxSAT resolution to cycle structures contained in an inconsistent subformula to improve the quality of LBs. In fact, generally speaking, let ϕ be a MaxSAT instance and l a literal of ϕ, we have

Lemma 1. *Let l be a failed literal in ϕ (i.e., $UP(\phi \wedge l)$ derives an empty clause), and let S_l be the set of clauses used to derive the contradiction in $UP(\phi \wedge l)$. If S_l contains the cycle structure $\bar{l}_1 \vee l_2, \ \bar{l}_1 \vee l_3, \ \bar{l}_2 \vee \bar{l}_3$, then l_1 was set to true in the unit propagation.*

Proof. Except the empty clause, every clause in S_l becomes unit when it is used for propagation, meaning that every clause in S_l is satisfied by at most one literal in the unit propagation. If l_1 was set to false in the propagation, then at least one of the three clauses $\bar{l}_1 \vee l_2, \ \bar{l}_1 \vee l_3, \ \bar{l}_2 \vee \bar{l}_3$ would be satisfied by two literals and could not belong to S_l. So, l_1 was set to true in the unit propagation. ∎

Lemma 2. *If l is a failed literal, and S_l contains the cycle structure $\bar{l}_1 \vee l_2, \ \bar{l}_1 \vee l_3, \ \bar{l}_2 \vee \bar{l}_3$, then l_1 was assigned a truth value before l_2 and l_3 in $UP(\phi \wedge l)$.*

Proof. Except the empty clause, every clause in S_l was unit when it was satisfied in the unit propagation. We assume that l_2 was assigned a truth value before l_1 and show that this is impossible. If l_2 was assigned true, clause $\bar{l}_1 \vee l_2$ would be satisfied without being unit; if l_2 was assigned false, then l_3 would be assigned true before l_2 was assigned false, since otherwise clause $\bar{l}_2 \vee \bar{l}_3$ would be satisfied before being unit. But in the latter case, clause $\bar{l}_1 \vee l_3$ would be satisfied without being unit. ∎

Lemma 2 also means that if S_l contains a cycle structure, then the cycle structure must be the last three binary clauses in the implication graph detecting S_l, which makes the identification of the cycle structure in S_l fast and easy.

Proposition 1. *Let l be a failed literal in ϕ (i.e., $UP(\phi \wedge l)$ derives an empty clause), and let S_l be the set of clauses used to derive the contradiction in $UP(\phi \wedge l)$. If S_l contains the cycle structure $\bar{l}_1 \vee l_2, \ \bar{l}_1 \vee l_3, \ \bar{l}_2 \vee \bar{l}_3$, and S'_l is S_l after applying MaxSAT resolution to the cycle structure, then $S'_l - \{l_1 \vee \bar{l}_2 \vee \bar{l}_3, \ \bar{l}_1 \vee l_2 \vee l_3\}$ is inconsistent.*

Proof. By Lemma 1 and Lemma 2, l_1 was assigned true in UP($\phi \wedge l$) independently of the three clauses $\bar{l}_1 \vee l_2$, $\bar{l}_1 \vee l_3$, $\bar{l}_2 \vee \bar{l}_3$, which are replaced with $\{\bar{l}_1, l_1 \vee \bar{l}_2 \vee \bar{l}_3, \bar{l}_1 \vee l_2 \vee l_3\}$ in S'_l. So, unit propagation in $S'_l - \{l_1 \vee \bar{l}_2 \vee \bar{l}_3, \bar{l}_1 \vee l_2 \vee l_3\}$ derives an empty clause from the unit clause $\bar{l}_1$. ∎

Proposition 1 means that, if both l and $\bar{l}$ are failed literals, and S_l contains the cycle structure $\bar{l}_1 \vee l_2$, $\bar{l}_1 \vee l_3$, $\bar{l}_2 \vee \bar{l}_3$, we can apply MaxSAT resolution in ϕ (replacing these three binary clauses with one unit clause $\bar{l}_1$ and two ternary clauses $l_1 \vee \bar{l}_2 \vee \bar{l}_3$ and $\bar{l}_1 \vee l_2 \vee l_3$) to obtain S'_l, and then transform the inconsistent subformula $S'_l \cup S_{\bar{l}} - \{l, \bar{l}\}$ into a smaller inconsistent subformula $S'_l \cup S_{\bar{l}} - \{l, \bar{l}, l_1 \vee \bar{l}_2 \vee \bar{l}_3, \bar{l}_1 \vee l_2 \vee l_3\}$ of ϕ. So, apart from incrementing the underestimation by 1, this transformation liberates two ternary clauses from $S'_l \cup S_{\bar{l}} - \{l, \bar{l}\}$ that can be used to derive other disjoint inconsistent subformulas, allowing to obtain better LBs.

In Example 1, $S_{\bar{x}_1} \cup S_{x_1} - \{\bar{x}_1, x_1\}$ is equal to

$$\{x_1 \vee x_2, \bar{x}_2 \vee x_3, \bar{x}_2 \vee x_4, \bar{x}_3 \vee \bar{x}_4, \bar{x}_1 \vee x_5, \bar{x}_5 \vee x_6, \bar{x}_1 \vee x_7, \bar{x}_6 \vee \bar{x}_7\}$$

which is transformed by applying MaxSAT resolution to the cycle structure $\bar{x}_2 \vee x_3$, $\bar{x}_2 \vee x_4$, $\bar{x}_3 \vee \bar{x}_4$ into a smaller inconsistent subformula

$$\{x_1 \vee x_2, \bar{x}_2, \bar{x}_1 \vee x_5, \bar{x}_5 \vee x_6, \bar{x}_1 \vee x_7, \bar{x}_6 \vee \bar{x}_7\}$$

So, after incrementing the underestimation by 1, we have two additional ternary clauses ($x_2 \vee \bar{x}_3 \vee \bar{x}_4$ and $\bar{x}_2 \vee x_3 \vee x_4$) liberated from the cycle structure which can be used to detect further inconsistent subformulas.

The benefit of applying MaxSAT resolution to a cycle structure not contained in an inconsistent subformula is not so clear. The next example suggests that this application may lead to a worse LB.

Example 2. Assume that a MaxSAT instance ϕ contains

$$x_1 \vee \bar{x}_2, \ x_1 \vee \bar{x}_3, \ x_2 \vee x_3, \ x_2 \vee x_6, \ x_3 \vee \bar{x}_6,$$
$$\bar{x}_1 \vee x_7, \ \bar{x}_1 \vee x_8, \ \bar{x}_7 \vee x_9, \ \bar{x}_8 \vee \bar{x}_9,$$
$$\bar{x}_4, \ x_4 \vee \bar{x}_2, \ \bar{x}_3 \vee x_5, \ \bar{x}_5$$

Without activating the inference component, unit propagation detects an inconsistent subformula

$$\{x_2 \vee x_3, \bar{x}_4, x_4 \vee \bar{x}_2, \bar{x}_3 \vee x_5, \bar{x}_5\}$$

Then, after removing this subformula from ϕ, failed literal detection on x_1 (i.e., unit propagation in $\phi \wedge \bar{x}_1$ and in $\phi \wedge x_1$ respectively) finds the second inconsistent subformula

$$\{x_1 \vee \bar{x}_2, x_1 \vee \bar{x}_3, x_2 \vee x_6, x_3 \vee \bar{x}_6, \bar{x}_1 \vee x_7, \bar{x}_1 \vee x_8, \bar{x}_7 \vee x_9, \bar{x}_8 \vee \bar{x}_9\}$$

Note that the first three clauses of ϕ form a cycle structure but do not belong to a same inconsistent subformula detected using unit propagation or failed literal detection. If MaxSAT resolution is applied to the cycle structure, ϕ becomes

$$x_1, \ \bar{x}_1 \vee x_2 \vee x_3, \ x_1 \vee \bar{x}_2 \vee \bar{x}_3, \ x_2 \vee x_6, \ x_3 \vee \bar{x}_6,$$
$$\bar{x}_1 \vee x_7, \ \bar{x}_1 \vee x_8, \ \bar{x}_7 \vee x_9, \ \bar{x}_8 \vee \bar{x}_9,$$
$$\bar{x}_4, \ x_4 \vee \bar{x}_2, \ \bar{x}_3 \vee x_5, \ \bar{x}_5.$$

Once unit propagation detects the inconsistent subformula

$$\{x_1, \ \bar{x}_1 \vee x_2 \vee x_3, \ \bar{x}_4, \ x_4 \vee \bar{x}_2, \ \bar{x}_3 \vee x_5, \ \bar{x}_5\}$$

ϕ becomes (after removing the inconsistent subformula)

$$\{x_1 \vee \bar{x}_2 \vee \bar{x}_3, \ x_2 \vee x_6, \ x_3 \vee \bar{x}_6, \ \bar{x}_1 \vee x_7, \ \bar{x}_1 \vee x_8, \ \bar{x}_7 \vee x_9, \ \bar{x}_8 \vee \bar{x}_9\}$$

and is consistent. So, only one inconsistent subformula is detected when MaxSAT resolution is applied to the cycle structure, making the LB worse.

5 Heuristics for Applying MaxSAT Resolution in Cycle Structures

From the previous analysis, we observe that it is better to apply MaxSAT resolution to cycle structures contained in an inconsistent subformula in order to transform the inconsistent subformula and liberate two ternary clauses from the subformula. In practice, when we identify a cycle structure at a node of the search tree, we distinguish three cases:

1. The cycle structure is contained in an inconsistent subformula.
2. The cycle structure is not contained in an inconsistent subformula at the current node, but probably belongs to an inconsistent subformula in the subtree below the current node.
3. The cycle structure is not contained in an inconsistent subformula at the current node and probably will not belong to an inconsistent subformula in the subtree.

We define a heuristic that applies MaxSAT resolution in the first two cases. As we will see, the benefit of applying MaxSAT resolution in the second case is twofold: two ternary clauses are liberated in advance, and the probable inconsistent subformula containing the cycle structure will be easier and faster to detect in the subtree with the application of MaxSAT resolution. This heuristic is implemented in Algorithm 1., where occ2(l) is the number of occurrences of literal l in the binary clauses of ϕ.

Between the two literals of a variable x that have reasonable probability to be failed (since their satisfaction results in at least two new unit clauses), Algorithm 1. detects first the literal l with more occurrences in binary clauses. Note that l has a smaller probability of being failed than $\bar{l}$ since its satisfaction produces fewer new unit clauses than the satisfaction of $\bar{l}$.

If l is a failed literal and S_l contains a cycle structure, the cycle structure is replaced to obtain S'_l before detecting $\bar{l}$. If $\bar{l}$ also is a failed literal, the inconsistent subformula $S'_l \cup S_{\bar{l}} - \{l, \bar{l}\}$ is transformed into a smaller inconsistent subformula to liberate two ternary clauses. If $\bar{l}$ is not a failed literal in the current node, failed literal detection does not detect an inconsistent subformula containing the cycle structure of S_l, but S_l is now smaller thanks to MaxSAT resolution because it becomes now $S'_l - \{l_1 \vee \bar{l}_2 \vee \bar{l}_3, \bar{l}_1 \vee l_2 \vee l_3\}$ by Proposition 1, and it will be easier to re-detect in the subtree. Note that $\bar{l}$ has reasonable probability to be a failed literal in the subtree, i.e., the cycle structure in the original S_l has reasonable probability to be contained in an inconsistent subformula in the subtree. As soon as $\bar{l}$ fails in the subtree, Algorithm 1. will detect the smaller inconsistent subformula $S_l \cup S_{\bar{l}} - \{l, \bar{l}\}$ with smaller cost (recall S_l is now smaller).

Algorithm 1. *flAndCycle*(ϕ, x), combining MaxSAT resolution to cycle structures and failed literal detection

---

Input: A MaxSAT instance ϕ, and a variable x such that occ2(x)$\geq$ 2 and occ2($\bar{x}$)$\geq$ 2
Output: ϕ in which MaxSAT resolution is possibly applied to a cycle structure, and an underestimation

1 **begin**
2   **if** occ2(x)>occ2($\bar{x}$) **then** $l \longleftarrow x$; **else** $l \longleftarrow \bar{x}$;
    underestimation $\longleftarrow$ 0;
    if *UP($\phi \wedge l$) derives a contradiction* **then**
3   if S_l contains a cycle structure, replace the cycle structure with one unit clause and two ternary clauses;
        if *UP($\phi \wedge \bar{l}$) derives a contradiction* **then**
4   if $S_{\bar{l}}$ contains a cycle structure, replace the cycle structure with one unit clause and two ternary clauses;
            underestimation $\longleftarrow$ 1;
5   **return** new ϕ and *underestimation*
6 **end**

On the contrary, if l is not a failed literal, Algorithm 1. does not detect an inconsistent subformula, $\bar{l}$ is not detected and no inference is applied to $S_{\bar{l}}$ even if $\bar{l}$ is a failed literal, avoiding the application of MaxSAT resolution to a cycle structure not contained in an inconsistent subformula.

With the aim of evaluating the impact of Algorithm 1. in the performance of MaxSatz, we define the following variants of solvers:

- MaxSatz: It is a Weighted Partial MaxSAT solver developed by J. Argelich, C.M. Li and F. Manyà [1]. MaxSatz participated in the 2008 MaxSAT Evaluation, and incorporates all the MaxSatz inference rules, and failed literal detection, besides UP, in the underestimation component. MaxSatz applies MaxSAT resolution to cycles structures in a limited way using Rule 3 and Rule 4.
- MaxSatz$_c$: It is a variant of MaxSatz in which failed literal detection is combined with the heuristic application of MaxSAT resolution to cycle structures. For every variable x such that occ2(x)$\geq$ 2 and occ2($\bar{x}$)$\geq$ 2, failed literal detection is replaced with Algorithm 1.. For the rest of variables, it is applied as in MaxSatz.
- MaxSatz$_c^p$: It is a variant of MaxSatz$_c$ in which MaxSAT resolution is applied to all the cycle structures appearing at the root node, and is applied as in MaxSatz$_c$ to the cycle structures appearing in the rest of nodes. In other words, MaxSAT resolution is exhaustively applied to cycle structures as a preprocessing. Notice that this preprocessing has no effect on problems not containing cycle structures in the input formula (e.g., Max-3SAT). In this case, MaxSatz$_c^p$ is just MaxSatz$_c$. Although all cycle structures at the root node are replaced, new cycle structures can be created in the rest of nodes. Cycle structures may appear because (i) non-binary clauses may become binary clauses during the search, and (ii) Rule 1, Rule 2, Rule 3, and Rule 4 applied in UP, before failed literal detection, may transform binary clauses and add ternary clauses.

- MaxSatz^p: It is a variant of MaxSatz in which MaxSAT resolution is applied to all the cycle structures appearing at the root node, and in the rest of nodes, it is just MaxSatz.
- MaxSatz$_{c*}$: It is a variant of MaxSatz in which MaxSAT resolution is applied exhaustively to all cycle structures at each node after applying UP and inference rules (Rule 1, Rule 2, Rule 3, and Rule 4), and before applying failed literal detection. The application is exhaustive because no subset of binary clauses matching a cycle structure remains in the current instance.

MaxSatz$_{c*}$ is related to Max-DPLL when replacing every cycle structure with one unit clause and two ternary clauses. MaxSatz$_c$ extends MaxSatz and MiniMaxSat in that MaxSatz$_c$ additionally applies MaxSAT resolution to cycle structures contained in an inconsistent subformula detected using failed literal detection. Moreover, differently from MiniMaxSat, MaxSatz$_c$ replaces these cycle structures no matter if the refutation has arity less than 4 or not.

Recall that a weighted clause (C, w_1+w_2) is equivalent to two weighted clauses (C, w_1) and (C, w_2). The difference between MaxSatz$_c$ and MaxSatz$_{c*}$ should be bigger for Weighted MaxSAT than for unweighted MaxSAT. For example, if a weighted formula contains a cycle structure $(\bar{l}_1 \vee l_2, 3), (\bar{l}_1 \vee l_3, 4), (\bar{l}_2 \vee \bar{l}_3, 5)$, MaxSatz$_{c*}$ replaces entirely this cycle structure with $(\bar{l}_1, 3), (l_1 \vee \bar{l}_2 \vee \bar{l}_3, 3), (\bar{l}_1 \vee l_2 \vee l_3, 3)$, and leaves two clauses $(\bar{l}_1 \vee l_3, 1), (\bar{l}_2 \vee \bar{l}_3, 2)$ in the formula. On the contrary, MaxSatz$_c$ only replaces the part of this cycle structure contained in an inconsistent subformula. If the minimum clause weight in the inconsistent subformula is 2, MaxSatz$_c$ replaces this cycle structure with $(\bar{l}_1, 2), (l_1 \vee \bar{l}_2 \vee \bar{l}_3, 2), (\bar{l}_1 \vee l_2 \vee l_3, 2)$, and leaves the cycle structure $(\bar{l}_1 \vee l_2, 1), (\bar{l}_1 \vee l_3, 2), (\bar{l}_2 \vee \bar{l}_3, 3)$ in the formula, which is different from the unweighted case where MaxSatz$_c$ *never partly* replaces a cycle structure.

6 Experimental Results and Analysis

We conducted experiments to compare the performance of the different versions of MaxSatz described in the previous section (MaxSatz, MaxSatz$_c$, MaxSatz$_c^p$, MaxSatz^p, and MaxSatz$_{c*}$).

As benchmarks for Weighted MaxSAT, we considered random weighted Max-2SAT, random weighted Max-3SAT, and random weighted Max-CUT instances, and all the crafted instances of the 2008 MaxSAT Evaluation (the evaluation did not include industrial instances for Weighted MaxSAT). As benchmarks for Partial MaxSAT, we considered random partial Max-2SAT and random partial Max-3SAT instances, and all the industrial and crafted instances of the 2008 MaxSAT Evaluation. We did not solve the random instances of the Weighted MaxSAT and Partial MaxSAT categories of the 2008 MaxSAT Evaluation because they are easily solved with the different versions of MaxSatz. We selected instances which are harder and allow to analyze the scaling behavior of the solvers.

We do not include the experimental results with other solvers in this section for three reasons: (i) the purpose of the experiments is to show the effectiveness of the heuristic replacement of cycles structures given in Algorithm 1., while keeping other things equal

in a solver; (ii) for randomly generated (weighted or partial) instances, other solvers are too slow to be displayed in the figures; (iii) for crafted and industrial instances of the 2008 MaxSAT Evaluation, the performance of other solvers can be checked in the web page of the evaluation (http//www.maxsat.udl.cat/08/).

Experiments were performed on a MacPro with two 2.8GHz Quad-Core Intel Xeon processors and 4Gb of RAM. For every instance, besides the run time, we compute the total number k of cycle structures replaced by applying MaxSAT resolution (in addition to Rule 3 and Rule 4), and divide k by the search tree size t. The ratio k/t roughly indicates the average number of cycle structures replaced (in addition to the applications of Rule 3 and Rule 4) at a search tree node. For the 2008 MaxSAT Evaluation instances, we set a cutoff of 30 minutes. These instances generally include no or very few cycle structures in their initial formulas, so that the preprocessing is not significant. We do not include $MaxSatz_c^p$ and $MaxSatz^p$ in the comparison for them for the sake of clarity.

The experimental results for Weighted MaxSAT are shown in Figure 1, Figure 2, Figure 3, and Table 1. Figure 1 shows the mean time (left plot) and the mean ratio k/t (right plot) to solve sets of 100 randomly generated Weighted Max-2SAT instances with 100 variables and an increasing number of clauses. MaxSatz is not included in the right plot, because cycle structures replaced by Rule 3 and Rule 4 are not counted in k, so that k is always 0 for MaxSatz. We observe a clear advantage of $MaxSatz_c$ and $MaxSatz_c^p$, which are up to one order of magnitude better than MaxSatz and $MaxSatz_{c^*}$. The search tree size (not shown for lack of space) follows the same ordering. It can be observed that, it suffices to replace three or four cycle structures contained in an inconsistent subformula at a search tree node to obtain an important gain, and the gain grows with the number of cycle structures replaced. However, if the cycle structures not contained in an inconsistent subformula are also replaced as in $MaxSatz_{c^*}$, the LB becomes substantially worse and the search tree larger, and the more replacements there are, the worse the LB.
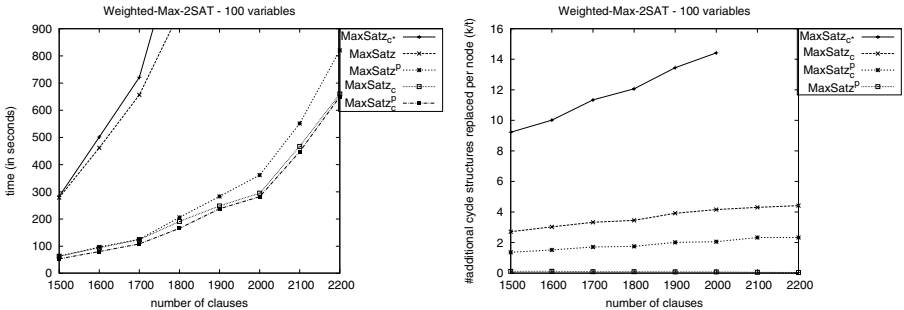

Fig. 1. Weighted Max-2SAT instances

Figure 2 shows the mean time (left plot) and the mean ratio k/t (right plot) to solve sets of 100 randomly generated Weighted Max-3SAT instance with 60 variables and an increasing number of clauses. Since weighted Max-3SAT instances do not include cycle structures, the preprocessing has no effect. Therefore, Figure 2 does not include

$MaxSatz_c^p$ and $MaxSatz^p$. We observe that $MaxSatz_c$ is clearly better than the rest of solvers. Note that all cycle structures are dynamically created during search.
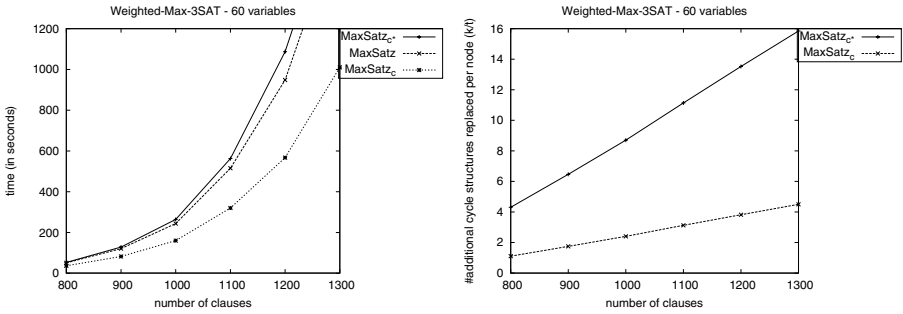

Fig. 2. Weighted Max-3SAT instances

Figure 3 shows the mean time (left plot) and the mean ratio k/t (right plot) to solve sets of 100 random Weighted Max-CUT instances generated from random graphs of 100 nodes and an increasing number of edges. $MaxSatz_{c*}$ is better than MaxSatz because a cycle structure easily belongs to an inconsistent subformula due to the special structure of the Max-CUT problem. Nevertheless the heuristic application of MaxSAT resolution to cycle structures contained in an inconsistent subformula makes $MaxSatz_c$ significantly better than $MaxSatz_{c*}$.
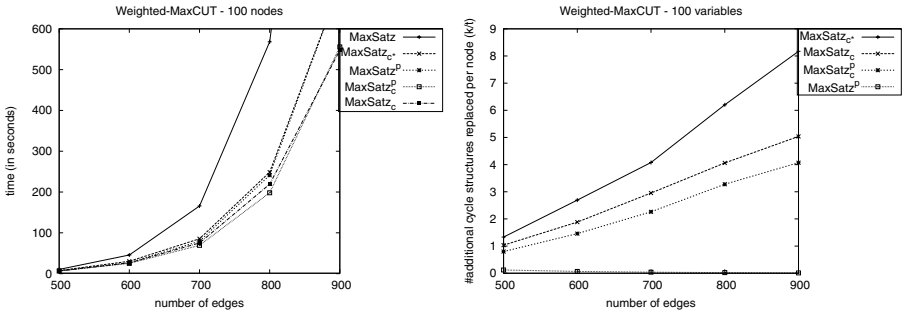

Fig. 3. Weighted Max-CUT instances

Table 1 contains the results for the crafted instances of the Weighted Max-SAT category of the 2008 MaxSAT Evaluation. For each group of instances, we display the number of instances **I** in the group, and for each solver, the number of instances solved within the cutoff of 30 minutes (in brackets) and the mean time in seconds to solve these solved instances. $MaxSatz_c$ is the best performing solver, which solves 4 instances more than MaxSatz and 2 instances more than $Maxsatz_{c*}$.

Table 1. Crafted instances of the Weighted Max-SAT category of the 2008 MaxSAT Evaluation

Instance set	I	MaxSatz	MaxSatz$_c$	MaxSatz$_{c*}$
KeXu	15	14.97(10)	**13.85(10)**	25.28(10)
RAMSEY	48	25.45(36)	**10.93(36)**	14.80(36)
WMAXCUT-DIMACS-MOD	62	54.22(55)	**90.95(57)**	89.34(55)
WMAXCUT-RANDOM	40	10.22(40)	**3.54(40)**	5.38(40)
WMAXCUT-SPINGLASS	5	301.83(2)	**31.23(4)**	35.60(4)
All instances	170	143	**147**	145

The experimental results for Partial MaxSAT are shown in Figure 4, Figure 5, Table 2, and Table 3. Figure 4 shows the mean time (left plot) and the mean ratio k/t (right plot) to solve sets of 100 randomly generated partial Max-2SAT instance with 150 variables, 150 hard clauses as in the 2008 MaxSAT evaluation, and an increasing number of soft clauses. For these large instances, MaxSatz$_c$ outperforms the rest of solvers, and MaxSatz$_{c*}$ is by far the worst.
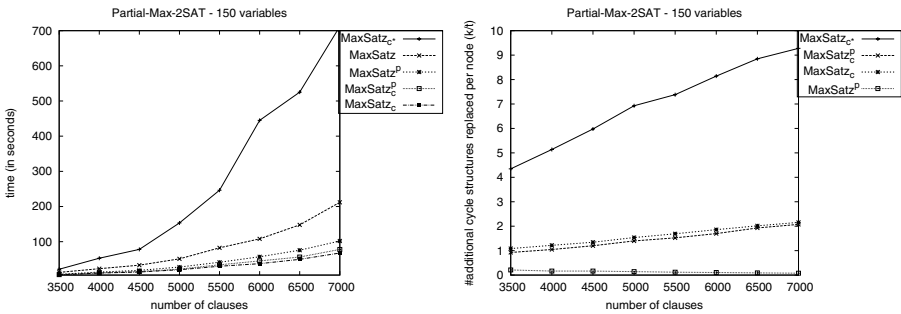

Fig. 4. Partial Max-2SAT instances

Figure 5 shows the mean time (left plot) and the mean ratio k/t (right plot) to solve sets of 100 randomly generated partial Max-3SAT instance with 80 variables, 80 hard clauses, and an increasing number of soft clauses. Note that there are very few cycle structures replaced at a search tree node, but the gain of MaxSatz$_c$ and the loss of MaxSatz$_{c*}$ are very significant.

Table 2 contains the results for the industrial instances of the Partial Max-SAT category of the 2008 MaxSAT Evaluation. MaxSatz$_c$ solves 25 instances more than MaxSatz, and 40 instances more than MaxSatz$_{c*}$. Table 3 contains the results for the crafted instances of the Partial Max-SAT category of the 2008 MaxSAT Evaluation. MaxSatz$_c$ solves 7 instances more than MaxSatz$_{c*}$. There are very few cycle structures contained in an inconsistent subformula during search for these instances. Their exploitation still makes MaxSatz$_c$ the best performing solver in general.
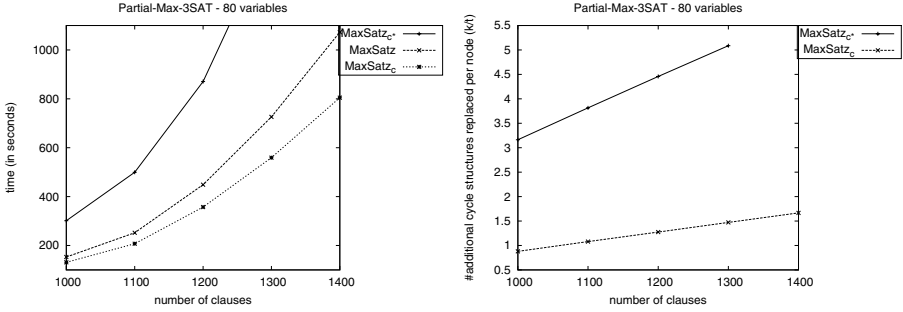
Fig. 5. Partial Max-3SAT instances

Table 2. Industrial instances of the Partial Max-SAT category of the 2008 MaxSAT Evaluation

Instance set	I	MaxSatz	MaxSatz$_c$	MaxSatz$_{c*}$
bcp-fir	59	8.91(7)	**5.03(7)**	8.16(7)
bcp-hipp-yRa1	1183	73.18(734)	**70.81(744)**	77.01(721)
bcp-msp	148	40.53(94)	**17.86(94)**	22.41(94)
bcp-mtg	215	33.07(144)	**96.25(157)**	95.41(154)
bcp-syn	74	**97.41(22)**	104.68(22)	82.67(21)
pbo-mqc-nencdr	128	**514.05(77)**	436.91(76)	475.17(64)
pbo-mqc-nlogencdr	128	323.57(104)	**270.38(107)**	333.04(106)
pbo-routing	15	61.43(5)	**3.13(5)**	5.22(5)
All instances	1950	1187	**1212**	1172

Table 3. Crafted instances of the Partial Max-SAT category of the 2008 MaxSAT Evaluation

Instance set	I	MaxSatz	MaxSatz$_c$	MaxSatz$_{c*}$
MAXCLIQUE-RANDOM	96	75.67(83)	**68.17(83)**	48.68(80)
MAXCLIQUE-STRUCTURED	62	164.70(25)	**138.90(25)**	149.25(22)
MAXONE-3SAT	80	**139.09(78)**	148.70(78)	204.47(77)
MAXONE-STRUCTURED	60	80.63(58)	**75.36(58)**	96.41(58)
All instances	298	244	**244**	237

7 Conclusions

We have studied why and when is useful to apply MaxSAT resolution to cycle structures in MaxSAT LB computation. We found that the exhaustive application of MaxSAT resolution is not so effective in general, and that MaxSAT resolution is effective if it is applied to cycle structures contained in an inconsistent subformula detected using unit propagation or failed literal detection. The benefit is twofold: (i) the inconsistent subformula can be transformed into a smaller one to liberate two ternary clauses for detecting other inconsistent subformulas, (ii) the smaller inconsistent subformula is easier and faster to detect or re-detect in subtrees. Experimental results suggest that the solver becomes

much slower when MaxSAT resolution is applied to cycle structures not contained in an inconsistent subformula.

We defined a heuristic that guides the applications of MaxSAT resolution to cycle structures. The implementation of this heuristic provides empirical evidence that it is very effective on many hard instances of Weighted MaxSAT and Partial MaxSAT, independently if they are random, crafted or industrial, as soon as few inconsistent subformulas contain a cycle structure.

In the future, we will study the exploitation of other structures in MaxSAT instances. It is remarkable that a relevant exploitation of few structures at a search tree node can result in a substantial speed-up in the solving of a MaxSAT problem.

References

1. Argelich, J., Li, C.M., Manyà, F.: An improved exact solver for partial Max-SAT. In: NCP 2007, pp. 230–231 (2007)
2. Bonet, M.L., Levy, J., Manyà, F.: Resolution for Max-SAT. Artificial Intelligence 171(8-9), 240–251 (2007)
3. Darras, S., Dequen, G., Devendeville, L., Li, C.M.: On inconsistent clause-subsets for max-SAT solving. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 225–240. Springer, Heidelberg (2007)
4. Heras, F., Larrosa, J., Oliveras, A.: MiniMaxSat: A new weighted Max-SAT solver. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 41–55. Springer, Heidelberg (2007)
5. Larrosa, J., Heras, F.: Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In: IJCAI 2005, pp. 193–198 (2005)
6. Larrosa, J., Heras, F., de Givry, S.: A logical approach to efficient Max-SAT solving. Artificial Intelligence 172(2-3), 204–233 (2008)
7. Li, C.M., Manyà, F., Mohamedou, N.O., Planes, J.: Transforming inconsistent subformulas in MaxSAT lower bound computation. In: Stuckey, P.J. (ed.) CP 2008. LNCS, vol. 5202, pp. 582–587. Springer, Heidelberg (2008)
8. Li, C.M., Manyà, F., Planes, J.: Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers. In: van Beek, P. (ed.) CP 2005. LNCS, vol. 3709, pp. 403–414. Springer, Heidelberg (2005)
9. Li, C.M., Manyà, F., Planes, J.: Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In: AAAI 2006, pp. 86–91 (2006)
10. Li, C.M., Manyà, F., Planes, J.: New inference rules for Max-SAT. Journal of Artificial Intelligence Research 30, 321–359 (2007)
11. Lin, H., Su, K., Li, C.M.: Within-problem learning for efficient lower bound computation in Max-SAT solving. In: AAAI 2008, pp. 351–356 (2008)
12. Pipatsrisawat, K., Darwiche, A.: Clone: Solving weighted Max-SAT in a reduced search space. In: AI 2007, pp. 223–233 (2007)
13. Ramírez, M., Geffner, H.: Structural relaxations by variable renaming and their compilation for solving MinCostSAT. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 605–619. Springer, Heidelberg (2007)