Boosting MCSes Enumeration

Éric Grégoire, Yacine Izza, Jean-Marie Lagniez
CRIL - Université d'Artois & CNRS, Lens, France
{gregoire,izza,lagniez}@cril.fr

Abstract

The enumeration of all Maximal Satisfiable Subsets (MSSes) or all Minimal Correction Subsets (MCSes) of an unsatisfiable CNF Boolean formula is a useful and sometimes necessary step for solving a variety of important A.I. issues. Although the number of different MCSes of a CNF Boolean formula is exponential in the worst case, it remains low in many practical situations; this makes the tentative enumeration possibly successful in these latter cases. In the paper, a technique is introduced that boosts the currently most efficient practical approaches to enumerate MCSes. It implements a model rotation paradigm that allows the set of MCSes to be computed in an heuristically efficient way.

1 Introduction

Computing one *maximal*[1] *satisfiable subset* of clauses, noted MSS, within an unsatisfiable CNF Boolean formula is a cornerstone task in various A.I. domains ranging from model-based diagnosis (see e.g., the seminal work in [Reiter, 1987], a readings book [Hamscher *et al.*, 1992] or some more recent research work in [Felfernig *et al.*, 2012; Marques-Silva *et al.*, 2015]) to various paradigms of belief change (see e.g., [Fermé and Hansson, 2011] for a survey of the field). In the Boolean setting, reasoning in a credulous [Reiter, 1980] way about contradictory information represented by an unsatisfiable CNF Σ can amount to reasoning about one MSS of Σ. Interpreted as a set of constraints, an unsatisfiable CNF formula Σ represents an over-constrained problem [Meseguer *et al.*, 2003] for which no solution exists. When Φ is one maximal satisfiable subset of Σ and when Ψ is defined as $\Sigma \setminus \Phi$, Ψ is one minimal subset of Σ such that dropping Ψ from the problem makes this one become feasible. Accordingly, Ψ is sometimes called a *minimal correction subset* of Σ (hence the interchangeable notations MCS and Co-MSS for this concept). In the worst case, the computation of an MSS or an MCS is a hard computational task since the basic problem of checking whether a set of clauses forms one MCS is DP-complete [Chen and Toda, 1995], which is also the complexity of the SAT-UNSAT problem. However, in many practical situations, computing one MSS is routinely performed by SAT-based tools (see e.g., [Marques-Silva *et al.*, 2013; Grégoire *et al.*, 2014; Bacchus *et al.*, 2014; Mencía *et al.*, 2015; 2016] among others).

The enumeration of all MCSes or all MSSes of Σ is even more computationally challenging. This task is a useful and sometimes necessary step in order to implement some forms of skeptical reasoning in abstract argumentation [Lagniez *et al.*, 2015] or in the presence of contradictions, which is a general issue that can be traced back to early seminal works about nonmonotonic logics [Bobrow, 1980]. It plays also a role in infeasibility analysis of a set of clauses, as the computation of all *minimal unsatisfiable subsets*, in short MUSes, can rely on the success of this enumeration (see e.g., [Liffiton and Sakallah, 2008; Grégoire *et al.*, 2007; Nadel *et al.*, 2014; Bacchus and Katsirelos, 2015; 2016; Previti and Marques-Silva, 2013; Liffiton *et al.*, 2016] among others). Although the number of different MCSes of the same formula is exponential in the worst case, it remains low in many practical situations; this makes the tentative enumeration of all MCSes possibly successful in these latter cases.

In the paper, a technique is introduced that boosts the currently most efficient practical techniques to enumerate the MCSes of an unsatisfiable Boolean formula Σ. It is based on a form of so-called model rotation paradigm [Belov and Marques-Silva, 2011a; Nadel *et al.*, 2014; Bacchus and Katsirelos, 2015]. We show that it allows the set of MCSes of Σ to be computed in an heuristically efficient way.

The paper is organized as follows. The technical background and the well-known MSS and MCS concepts are briefly reviewed in the preliminaries. In section 3, the key paradigms of transition clauses and clause selectors are recalled; their roles are illustrated through the basic linear search for one MCS. Section 4 presents a basic MCSes enumeration algorithm before an original advanced one is step-by-step described in section 5. In section 6, the use of model rotation in this latter algorithm is explained. Then, we present our extensive experimental study. Promising paths for further research are briefly introduced in the conclusion.

2 Technical Background

We consider a standard language of formulas $\mathcal{L}$ of Boolean logic. $\neg$, $\vee$, $\wedge$ and $\Rightarrow$ represent the negation, disjunction, conjunction and material implication connectives, respectively.

[1] We always consider set-inclusion maximality in this paper.

A literal is either a Boolean variable or its negation. A clause is a formula that consists of a disjunction of literals. A CNF (clausal normal form) formula is a conjunction (also represented as a set) of clauses. $\alpha, \beta, \ldots$ and $\Sigma, \Delta, \Phi, \Psi, \ldots$ denote formulas and sets of formulas, respectively. An interpretation μ assigns values from $\{0, 1\}$ to every Boolean variable, and, following usual compositional rules, to all formulas of $\mathcal{L}$. A formula α is satisfiable iff there exists at least one interpretation μ that satisfies α; μ is then called a model of α. Accordingly, μ satisfies a non-empty clause α when there exists at least one literal of α that is assigned 1 by μ; μ satisfies a CNF Σ iff μ satisfies all clauses of Σ. Any formula of $\mathcal{L}$ can be rewritten under an equisatisfiable CNF formula.

The core, MUS, MSS and MCS cross-related concepts are defined as follows. Let Σ be a CNF formula.

Definition 1. (Core) A CNF Σ' is a *core* of Σ iff $\Sigma' \subseteq \Sigma$ and Σ' is unsatisfiable.

Definition 2. (MUS) A *minimal unsatisfiable subset* (in short, MUS) Υ of Σ is a core of Σ such that $\forall \alpha \in \Upsilon$, $\Upsilon \setminus \{\alpha\}$ is satisfiable.

When a CNF Σ is unsatisfiable, it can be split into one MSS, i.e., one *maximal satisfiable subset* of Σ, and its set-theoretical complement in Σ, namely, one *minimal correction subset*, for short one MCS, of Σ.

Definition 3. (MSS) A *maximal satisfiable subset* (in short, MSS) Φ of Σ is a subset $\Phi \subseteq \Sigma$ that is satisfiable and such that $\forall \alpha \in \Sigma \setminus \Phi$, $\Phi \cup \{\alpha\}$ is unsatisfiable.

Definition 4. (MCS) A *minimal correction subset* (in short MCS, also called Co-MSS) Ψ of Σ is a set $\Psi \subseteq \Sigma$ whose complement in Σ, i.e., $\Sigma \setminus \Psi$, is an MSS of Σ.

Example 1. *Let Σ be an unsatisfiable* CNF *formed by a set of clauses $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6\}$, where $\alpha_1 = a \vee b$, $\alpha_2 = \neg a \vee b$, $\alpha_3 = a \vee \neg b$, $\alpha_4 = \neg a \vee \neg b$, $\alpha_5 = \neg b$, $\alpha_6 = b$. The* MCS*es of Σ are $\{\alpha_1, \alpha_6\}$, $\{\alpha_2, \alpha_6\}$, $\{\alpha_3, \alpha_5\}$ and $\{\alpha_4, \alpha_5\}$. The* MUS*es of Σ are $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$, $\{\alpha_1, \alpha_2, \alpha_5\}$, $\{\alpha_3, \alpha_4, \alpha_6\}$ and $\{\alpha_5, \alpha_6\}$.*

Notice that one MSS of Σ can be interpreted as one approximate solution of Max-SAT on Σ, since maximality in MSS is about set-inclusion vs. cardinality. The currently most efficient approaches and tools to compute one MSS are described in [Grégoire *et al.*, 2014] and [Mencía *et al.*, 2015; 2016].

The MSS and MCS paradigms can be extended into the so-called Partial-MSS and Partial-MCS concepts in order to handle the situation where $\Sigma = \langle \Sigma_1, \Sigma_2 \rangle$, where Σ_1 and Σ_2 are sets of hard and soft clauses, respectively: hard clauses are required to belong to any Partial-MSS of Σ; they thus do not belong to any Partial-MCS of Σ. In the sequel, Σ_1 and Σ_2 always denote sets of hard and soft clauses, respectively. Notice that when Σ_1 is unsatisfiable, Partial-MSSes of Σ do not exist. In all the other cases, there always exists at least one Partial-MCS of Σ that is included in Σ_2. Moreover, every Partial-MCS of $\langle \Sigma_1, \Sigma_2 \rangle$ is an MCS of $\Sigma_1 \cup \Sigma_2$. Conversely, every MCS of $\Sigma_1 \cup \Sigma_2$ that does not satisfy Σ_1 is not a Partial-MCS of $\langle \Sigma_1, \Sigma_2 \rangle$.

3 Basic Linear Search for one MCS

State-of-the-art MCS extractors often make use of so-called *clause selectors* as follows. Each clause α of Σ is augmented with its own (negated) selector, namely a fresh new literal $\neg s_\alpha$. This yields a new (relaxed) CNF formula Σ^S. A clause $\alpha \vee \neg s_\alpha$ in Σ^S is thus activated (resp., deactivated) when the literal s_α (resp., $\neg s_\alpha$) is set to 1. Selectors play the role of assumptions that can be activated/deactivated during the same search while useful information can be recorded at each step. Especially, when Σ' is shown unsatisfiable under some such assumptions, modern SAT solvers can often extract a subset of the assumptions that causes Σ' to be unsatisfiable [Eén and Sörensson, 2003; Lagniez and Biere, 2013; Audemard *et al.*, 2013]. We will exploit this feature in the MCSes enumeration algorithm. In the rest of the paper, we often implicitly refer to Σ^S and make no ontological difference between the selectors and the other literals in Σ^S.

The concept of transition clause (in short, TC) is also often a key paradigm for the currently most efficient approaches to compute one MUS [Grégoire *et al.*, 2007; Previti and Marques-Silva, 2013], one MSS or one MCS [Grégoire *et al.*, 2014].

Definition 5. (Transition Clause) A clause $\alpha \in \Sigma$ is a transition clause (in short, TC) of Σ when, at the same time, Σ is unsatisfiable and $\Sigma \setminus \{\alpha\}$ is satisfiable.

Thus, when a clause α is a TC of Σ, α (resp., $\Sigma \setminus \{\alpha\}$) is an MCS (resp., MSS) of Σ. Moreover, if α is a TC of Σ then α belongs to every MUS of Σ.

Although Σ might not contain any TC, state-of-the-art MCS-finding tools often take advantage of TCes in the following way. Starting from the empty set, they iteratively and greedily construct a set Σ': at each step, clauses from Σ that have not been considered so far are inserted one by one in Σ' until Σ' becomes unsatisfiable. The last considered clause is a TC for Σ' and is introduced in the MCS under construction. The corresponding linear search algorithm for computing one MCS is called `BLS`. State-of-the-art MCS-finding tools [Grégoire *et al.*, 2014; Bacchus *et al.*, 2014; Marques-Silva *et al.*, 2013] have grafted various improvements to this algorithm skeleton; they take advantage of disjoint cores, computed models, the exploitation of the disjunction of the clauses of the MCS under construction and backbone literals, mainly. In the rest of the paper, we refer to improved versions of `BLS` that include subsets of these additional features: they are noted `ELS` for Enhanced Linear Search.

`ELS` is easily adapted in such a way that it computes one Partial-MCS: the hard clauses are initially inserted in the MSS without selectors. An initial call to a SAT solver is necessary to check whether the set of the hard clauses is unsatisfiable: in the positive case, an empty set needs to be returned. In the following, when we refer to a Partial-MCS-finding algorithm, we consider a procedure with a set of hard clauses Σ_1 and a set of soft clauses Σ_2 as input parameters: namely, `ExtractPartialMCS`(Σ_1, Σ_2).

4 MCSes Enumeration

Alg. 1 describes the typical skeleton of usual current approaches [Liffiton and Sakallah, 2008; Liffiton *et al.*, 2016; Marques-Silva *et al.*, 2013] to enumerate all MCSes: each time one MCS is found, an additional clause is created. Its role is to prevent the same MCS from being computed again. It is made of the disjunction of all the selectors of the clauses in the discovered MCS. In Alg. 1, the blocking clauses are inserted in a set Δ. Partial-MCSes of $(\Sigma^S \cup \Delta, S)$ are computed while MCSes still exist, or, equivalently, while $\Sigma^S \cup \Delta$ is satisfiable.

Clearly, the time-consuming part of this algorithm lies in the multiple calls to a SAT oracle in the routine extracting one Partial-MCS. In this respect, [Previti *et al.*, 2017] has proposed to enhance the algorithm by recording, for caching purpose, the cores discovered during the successive computations of the different MCSes: this yields the currently most efficient MCSes enumeration algorithm, noted `mcscache-els` in this paper. In the next section, we propose another approach to decrease the number of calls to a SAT oracle; it is compatible with the caching technique. In the experimental section, we show that the proposed approach outperforms `mcscache-els`. Moreover, when combined with the caching technique, it yields an even more efficient algorithm. It is based on properties of transition clauses and on the so-called recursive model rotation paradigm; both are described in the next two sections.

Algorithm 1: `Enum-ELS` (Enumerate All MCSes Computed with the `ExtractPartialMCS` procedure);

Input : an unsatisfiable CNF formula Σ
Output : all MCSes of Σ

$\Sigma^S \leftarrow \{\alpha \vee \neg s_\alpha | \alpha \in \Sigma\}$; // with s_α fresh variables
$S \leftarrow \{s_\alpha | \alpha \in \Sigma\}$; // a set of selectors
$\Delta \leftarrow \emptyset$;
while $\Sigma^S \cup \Delta$ *is satisfiable* **do**
    $M^- \leftarrow$ `ExtractPartialMCS`$(\Sigma^S \cup \Delta, S)$;
    $output(M^-)$;
    $\Delta \leftarrow \Delta \cup (\bigvee_{s_\alpha \in M^-} s_\alpha)$; // blocking clauses

5 More MCSes Thanks to Transition Clauses

First, the next property shows that any transition clause (in short, TC) α of an unsatisfiable subset $\Sigma' \subseteq \Sigma$ can be the starting point of a family of MCSes of Σ, where each member of this family contains α. This family is shown to capture the MCSes of Σ that are made of α together with any Partial-MCS of $\langle \Sigma' \setminus \{\alpha\}, \Sigma \setminus \Sigma' \rangle$,

Property 1. *Let Σ be an unsatisfiable* CNF *formula and let $\Sigma' \subseteq \Sigma$ such that Σ' contains at least one TC α.*
*For all Partial-*MCSes *Γ that can be built from $\langle \Sigma' \setminus \{\alpha\}, \Sigma \setminus \Sigma' \rangle$, $\Gamma \cup \{\alpha\}$ is an* MCS *of Σ.*

The previous property is easily adapted to address the case where Partial-MCSes are targeted, as the following corollary shows.

Corollary 1. *Let $\langle \Sigma_1, \Sigma_2 \rangle$ be a couple of* CNF *formulas such that Σ_1 is satisfiable and $\Sigma_1 \cup \Sigma_2$ is unsatisfiable. Assume that $\Sigma'_2 \subseteq \Sigma_2$ is such that $\Sigma_1 \cup \Sigma'_2$ contains at least one TC α of Σ'_2. For all Partial-*MCSes *Γ that can be built from $\langle \Sigma_1 \cup \Sigma'_2 \setminus \{\alpha\}, \Sigma_2 \setminus \Sigma'_2 \rangle$, we have that $\Gamma \cup \{\alpha\}$ is a Partial-*MCS *of $\langle \Sigma_1, \Sigma_2 \rangle$.*

Now assume that we compute several TCes for a given subformula CNF $\Sigma' \subseteq \Sigma$. For each TC, it is possible to recursively apply the previous corollary in order to compute several Partial-MCSes. Moreover, the following property ensures that all these Partial-MCSes are different.

Property 2. *Let $\langle \Sigma_1, \Sigma_2 \rangle$ be a couple of* CNF *formulas such that Σ_1 is satisfiable and $\Sigma_1 \cup \Sigma_2$ is unsatisfiable. Assume that $\Sigma'_2 \subseteq \Sigma_2$ is such that $\Sigma_1 \cup \Sigma'_2$ contains at least two different TCes α_1 and α_2 that occur in Σ'_2. Then, for all Partial-*MCSes *Γ_1 and Γ_2 that can be built from $\langle \Sigma_1 \cup \Sigma'_2 \setminus \{\alpha_1\}, \Sigma_2 \setminus \Sigma'_2 \rangle$ and $\langle \Sigma_1 \cup \Sigma'_2 \setminus \{\alpha_2\}, \Sigma_2 \setminus \Sigma'_2 \rangle$, respectively, we have that $\Gamma_1 \cup \{\alpha_1\}$ and $\Gamma_2 \cup \{\alpha_2\}$ are different Partial-*MCSes *of $\langle \Sigma_1, \Sigma_2 \rangle$.*

The latter property allows us to derive and justify the original recursive Algorithm 2, called `TC-MCS`, for *Transition-Clauses-Based Enumeration of* MCSes. This algorithm extends `ELS` by computing not just one but several MCSes in a recursive way by means of a model rotation method (lines 7-12, Alg. 2). Its input is a couple of CNF formulas $\langle \Sigma_1 \cup \Sigma_2^S, U \rangle$; the output is a set of Partial-MCSes for this couple. Remember that U is the set of selectors corresponding to the clauses that have not been assigned so far to either an MCS or an MSS under construction. We assume that Σ_1 is satisfiable and thus, that $\Sigma_1 \cup \Sigma_2^S$ is satisfiable. Σ_2^S is actually built from Σ_2 using selectors as explained earlier. Notice that hard clauses do not need selectors as they must always be satisfied and thus be activated. The Partial-MCSes of $\langle \Sigma_1, \Sigma_2 \rangle$ are derived directly from the Partial-MCSes of $\langle \Sigma_1 \cup \Sigma_2^S, S \rangle$, where S is the set of unit clauses corresponding to all selectors.

The algorithm starts by checking if U is empty. In the positive case, the procedure returns $\emptyset$ since the set of Partial-MCSes of $\langle \Sigma_1 \cup \Sigma_2^S, U \rangle$ is empty as $\Sigma_1 \cup \Sigma_2^S$ is always satisfiable by construction. This is also the non-recursive step. Otherwise, the set of computed Partial-MCSes Θ and the set of selectors M^+ are initialized to the empty set. Remember that M^+ records the selectors corresponding to the clauses that can be activated when $\Sigma_1 \cup \Sigma_2^S \cup M^+$ is satisfiable. s_α is initialized to $\top$, i.e., the tautology.

Next, the loop (lines 3–6) incrementally augments M^+ with a sequence of s_α, which are removed from U. This process is iterated while U is not empty and, at the same time, while s_α is not a TC of $\Sigma_1 \cup \Sigma_2^S \cup M^+ \cup \{s_\alpha\}$. s_α is a TC when $\Sigma_1 \cup \Sigma_2^S \cup M^+ \cup \{s_\alpha\}$ becomes unsatisfiable. Indeed, all the selectors inserted so far in M^+ have guaranteed that $\Sigma_1 \cup \Sigma_2^S \cup M^+$ remained satisfiable. Thus, when an incoming s_α makes the formula become unsatisfiable, this selector s_α is a TC. At the end of the loop, two cases can occur: (1) $s_\alpha \in M^+$. In this case, M^+ has captured all the selectors from U and the input formula was actually satisfiable. Accordingly, the empty set is returned; (2) $s_\alpha \notin M^+$ and thus s_α is a TC. In this case, we look for a core Γ of the current unsatisfiable formula $\Sigma_1 \cup \Sigma_2^S \cup M^+ \cup \{s_\alpha\}$ (line 8). It is easy to see that s_α belongs to Γ and it is a TC of Γ. As already mentioned, modern SAT solvers return as a byproduct a core that is often smaller than the formula that is shown unsatisfiable. We choose to use this core instead of the latter formula since the core often contains more TCes. It is easy to prove by contradiction that every TC is present in the core. Once a core is extracted, a method is called in order to identify TCes belonging to the core. In the next section, we present an approach to achieve this process, but for the moment, we just assume that TCes are extracted. Notice that we have no guarantee that this approach is efficient and identifies s_α as a TC. For this reason, we directly add s_α into T, i.e., the set of transitions clauses, in order to ensure the termination of the algorithm. Then, for each identified TC $s_\beta \in (T \cap M^+)$ we recursively call the function with $\langle \Sigma_1 \cup \Sigma_2^S \cup (\Gamma \setminus \{\neg s_\beta\}), U \cup (M^+ \setminus \Gamma)\rangle$ as input. It is easy to show that these parameters match the conditions stated in Corollary 1. Accordingly, all the Partial-MCSes that we can compute from $\langle \Sigma_1 \cup \Sigma_2^S \cup (\Gamma \setminus \{\neg s_\beta\}), U \cup (M^+ \setminus \Gamma)\rangle$ can be augmented with s_β to yield a Partial-MCS of the input formula.

Algorithm 2: TC-MCS ($\langle \Sigma_1 \cup \Sigma_2^S, U\rangle$);

Input : $\langle \Sigma_1 \cup \Sigma_2^S, U\rangle$ a couple of CNF formulas
Output : Θ a set of Partial-MCSes of $\langle \Sigma_1 \cup \Sigma_2^S, U\rangle$

```
1  if U = ∅ then return ∅;
2  Θ ← ∅; M⁺ ← ∅; s_α ← ⊤;
3  while U ≠ ∅ and Σ_1 ∪ Σ_2^S ∪ M⁺ ∪ {s_α} is satisfiable do
4      M⁺ ← M⁺ ∪ {s_α};
5      s_α ← choose s_α ∈ U;
6      U ← U \ {s_α};
7  if s_α ∉ M⁺ then
8      Γ ← Core(Σ_1 ∪ Σ_2^S ∪ M⁺ ∪ {s_α});
9      T ← {s_α} ∪ Find-TC(Γ);
10     foreach s_β ∈ T ∩ M⁺ do
11         Ω ← TC-MCS(⟨Σ_1 ∪ Σ_2^S ∪ (Γ \ {¬s_β}), U ∪ (M⁺ \ Γ)⟩);
12         Θ ← Θ ∪ (β × Ω);
13 return Θ;
```

TC-MCS ($\langle \Sigma_1 \cup \Sigma_2^S, U\rangle$) can be inserted in Alg. 1 to yield a new enumeration algorithm for MCSes, depicted in Alg. 3.

Algorithm 3: Enum-ELS-RMR;

Input : an unsatisfiable CNF formula Σ
Output : all MCSes of Σ

```
Σ^S ← {α ∨ ¬s_α | α ∈ Σ} ;          // with s_α fresh variables
S ← {s_α | α ∈ Σ } ;                 // a set of selectors
Δ ← ∅;
while Σ^S ∪ Δ is satisfiable do
    Θ ← TC-MCS (⟨Σ^S ∪ Δ, S⟩);
    foreach M⁻ ∈ Θ do
        output(M⁻);
        Δ ← Δ ∪ (⋁_{s_α ∈ M⁻} s_α) ;   // blocking clauses
```

6 Using Model Rotation

Let us now explain how the Find-TC procedure (line 9 of Alg. 2) computes additional TCes. A naive method would consist in computing all the MCSes of $\Sigma_1 \cup \Sigma_2^S \cup M^+ \cup \{s_\alpha\}$ that are singletons contained in M^+. As this complete direct method can be too time-consuming, we propose an incomplete process to compute additional TCes, based on the so-called recursive model rotation paradigm [Belov and Marques-Silva, 2011b] noted rmr. This latter paradigm has been initially defined in the context of computing one or several Minimal Unsatisfiable Subsets (MUSes) of an unsatisfiable CNF formula, where computing additional TCes in a fast way can also play a key role. As for [Bacchus and Katsirelos, 2015] where MUSes enumeration is targeted, we take advantage of the duality between MUSes and MCSes. However, we use the rmr paradigm for a very different task, which consists in enumerating MCSes. rmr is based on a model-theoretical perspective of TC: a TC of an unsatisfiable CNF formula Σ is any clause $\alpha \in \Sigma$ such that there exists a complete interpretation μ of Σ that satisfies $\Sigma \setminus \{\alpha\}$ (and falsifies α, otherwise the formula Σ would be satisfiable). Starting from a model μ of $\Sigma \setminus \alpha$, rmr consists in flipping variables from α and checking if this new interpretation μ' satisfies all the clauses of Σ except some α' of Σ. In the positive case, α' is marked as being a TC of Σ provided that it was not already marked as such, and the process is recursively repeated with μ' and α'.

rmr can thus compute several TCes when a first one is identified as such. In the search of one Partial-MCS, this situation occurs when we have shown that there exists a model μ of $\Sigma_1 \cup \Sigma_2^S \cup M^+$ and proven that $\Sigma_1 \cup \Sigma_2^S \cup M^+ \cup \{s_\alpha\}$ is unsatisfiable. μ then plays the role of the initial interpretation and we keep the detected TCes belonging to $\{\beta \in \Sigma_2 \mid s_\beta \in M^+\}$, only. Finally, the set of selectors associated with the discovered TCes can be returned. It is important to note that we make sure that the clause selectors are never flipped by the process.

Although the rmr process is a polynomial one, it turns out that it can be time consuming in practice; it can actually reduce the practical efficiency of the enumeration algorithm. Such a situation occurs when the number of clauses contained in the set Δ of blocking clauses becomes too large. To avoid such a drawback, we point out some sufficient conditions to deprive clauses of Δ of the rmr process.

First, let us show the possible critical role of Δ in the search for additional TCes. Consider $\langle \Sigma^S, S\rangle$ with $\Sigma^S = \{\neg s_1 \vee a \vee b, \neg s_2 \vee \neg a, \neg s_3 \vee \neg b\}$. Assume that one MCS, namely the singleton $\{s_1\}$, has been computed, already. Thus, at this step $\Delta = \{s_1\}$. Let us iterate the process and compute one more MCS and call TC-MCS on $\langle \Sigma^S \cup \Delta, S\rangle$. Let us suppose that we stop the main loop of TC-MCS when $M^+ = \{s_1, s_2\}$ and $s_\alpha = s_3$. The condition in line 7 is satisfied; one core that is the complete formula is computed. Then, we search more TCes of $\Sigma^S \cup S$ instead of $\Sigma^S \cup S \cup \Delta$. In this case, it is easy to show that s_1, s_2, s_3 are TCes and twice the same MCS will be computed. Obviously, a posteriori checking whether an MCS has already been computed by consulting Δ can be too time-consuming.

However, a useful feature is that Σ^S and Δ do not share literals. Indeed, Δ is a set of positive clauses composed of selectors whereas these selectors occur negatively in Σ^S. Accordingly, the satisfiabilty of $\Sigma^S \cup \Delta$ can be split into two independent sub-problems following a partition $\{P, N\}$ of the set of selectors, where P and N denote the set of selectors that are positive and the set of negative ones, respectively. More precisely:

Property 3. *Let $\{P, N\}$ be a partition of S, Σ^S a* CNF *formula augmented with the set of selectors S and Δ a* CNF *formula built on selector variables and formed of positive clauses, only. $\Sigma^S \cup \Delta \cup \bigwedge_{s\in P} s \cup \bigwedge_{s\in N} \neg s$ is satisfiable if and only if $\Sigma^S \cup \bigwedge_{s\in P} s$ and $\Delta \cup \bigwedge_{s\in N} \neg s$ are satisfiable.*

When an MCS is under construction we are implicitly computing a bi-partition $\{M^+, M^-\}$ of S such that $\Sigma^S \cup \bigwedge_{s\in M^+} s$ and $\Delta \cup \bigwedge_{s\in M^-} \neg s$ are satisfiable. The aim is then to move as many as possible selectors from M^- to M^+ while keeping both $\Sigma^S \cup \bigwedge_{s\in M^+} s$ and $\Delta \cup \bigwedge_{s\in M^-} \neg s$ satisfiable. It is easy to prove that if the bi-partition $\{M^+, M^-\}$ of S is such that $\Sigma^S \cup \bigwedge_{s\in M^+} s$ and $\Delta \cup \bigwedge_{s\in M^-} \neg s$ are satisfiable, then if we move one element s' from M^- to M^+ such that $\Sigma^S \cup \bigwedge_{s\in M^+\cup\{s'\}} s$ is satisfiable then $\Delta \cup \bigwedge_{s\in M^-\setminus\{s'\}} \neg s$ is satisfiable (as all the clauses of Δ are positive, assigning one more positive selector cannot make the formula become unsatisfiable). Let us stress that symmetric results can be obtained when one element is moved from M^+ to M^-.

In order to avoid the need to consider Δ in the `rmr` procedure, we propose the following process. First, compute a partition $\{M^+, M^-\}$ of S that satisfies $\Sigma^S \cup \bigwedge_{s\in M^+} s$ and $\Delta \cup \bigwedge_{s\in M^-} \neg s$. M^+ is then used as starting point to be evolved into an MSS and all the clauses of M^- are marked as being candidates for being a TC. Then, each time we augment M^+ with a new selector, we have that $\Sigma^S \cup \bigwedge_{s\in M^+} s$ is satisfiable. Let us notice that, when we add an element to M^+, in some sense we "move" this element from M^- to M^+. Consequently, both $\Sigma^S \cup \bigwedge_{s\in M^+} s$ and $\Delta \cup \bigwedge_{s\in M^-} \neg s$ are satisfiable. In some sense, the marked selectors are responsible for the satisfiability of Δ. Thus, since M^+ is constructed such that $\Sigma^S \cup \bigwedge_{s\in M^+} s$ is satisfiable, it becomes useless to check the satisfiability of Δ during the `rmr` process when we forbid the marked clauses to be selected as TCes.

7 Experimental Study

We have implemented all our algorithms in C++ and used `Minisat` `http://minisat.se/` as backend SAT solver. We have selected the 866 benchmarks used in [Previti *et al.*, 2017; Marques-Silva *et al.*, 2013]: 269 instances are plain Max-SAT ones and the remaining 597 are Partial-Max-SAT ones. We have enriched this experimental setting by also considering a second series of plain Max-SAT benchmarks made of the instances from the MUS competition `http://www.satcompetition.org/2011`. Some of these instances were already present in the benchmarks proposed by [Previti *et al.*, 2017]. As we only kept the new ones, 1090 benchmarks were considered in total: 493 of them are plain Max-SAT instances and 597 are Partial-Max-SAT ones.

All experimentations have been conducted on Intel Xeon E52643 (3.30GHz) processors with 64Gb memory on Linux CentOS. Time-out was set to 1800 seconds for each run of an algorithm on an instance; memory-out was set to 8 Gb for each such run. All data, results and software used in the experimentations are available from `http://www.cril.fr/enumcs`.

First, we compared our own implementation of `Enum-ELS-RMR` (Alg. 4) with the same algorithm deprived of `rmr`. Our version of `ELS` included the exploitation of computed models [Grégoire *et al.*, 2014], as well as of backbone literals [Marques-Silva *et al.*, 2013]. The comparison was made in terms of the total number of computed MCSes for each benchmark instance. As shown by Fig. 1 the `rmr` paradigm allowed us to compute more (or the same number) of MCSes for every plain Max-SAT benchmark instance. The same result was obtained for most Partial-Max-SAT benchmarks, too.

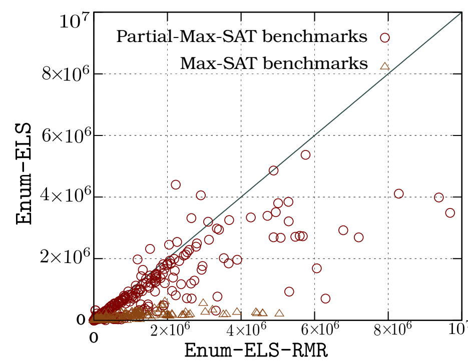

Figure 1: `Enum-ELS-RMR` vs. `Enum-ELS`

Then, we combined the caching technique with `Enum-ELS-RMR`, giving rise to `Enum-ELS-RMR-Cache`. Following the recommendation of [Previti *et al.*, 2017], we did not include the backbone feature in `ELS` since it might slow down the caching technique. `Enum-ELS-RMR-Cache` allowed a largest number of MCSes to be computed, most often (Fig. 2). Noticeably, it appeared that the instances for which `Enum-ELS-RMR-Cache` delivered a smaller number of MCSes where such that `Enum-ELS-Cache` already produced a smaller number of MCSes than `Enum-ELS-RMR`. Also, introducing the caching method leads to more memory-out conclusions. These two last points can be explained by the fact that, as already pointed out in [Previti *et al.*, 2017], the caching memory can become too large to handle.

Finally, we have compared the `mcscache-els` tool from [Previti *et al.*, 2017], which implements the caching technique on a state-of-the-art version of `Enum-ELS`, with `Enum-ELS-RMR-Cache`. Fig. 3 clearly shows that `Enum-ELS-RMR-Cache` outperforms `mcscache-els` for almost all benchmarks and the difference between both approaches is generally even more significant for the instances with the largest numbers of MCSes.

8 Conclusion and Perspectives

In this paper, we have enhanced the most efficient tool to enumerate all the minimal correction subsets (MCSes) of a Boolean CNF formula. Although the number of MCSes can

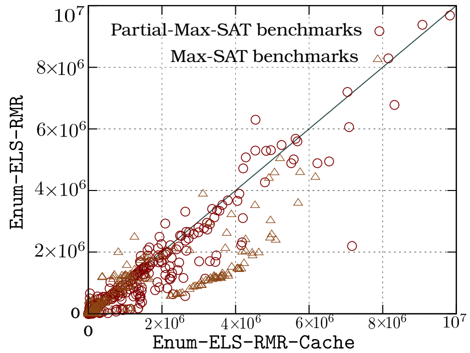

Figure 2: `Enum-ELS-RMR-Cache` vs. `Enum-ELS-RMR`

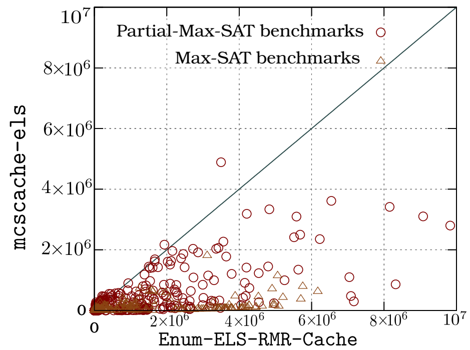

Figure 3: `Enum-ELS-RMR-Cache` vs. `mcscache-els`

be exponential in the worst case, it remains low in many real-life problems, especially in problems for which the number and the cardinality of the different minimal sources of unsatisfiability remain low. Indeed, there exists a hitting set correspondence between the set of MCSes and the set of MUSes, namely of minimal unsatisfiable subsets. Accordingly, computational progress made in enumerating all MCSes of CNF formulas when their number of MCSes is large opens new perspectives for enumerating all MUSes for the same formula. Indeed, the approches for listing all MUSes that first compute all MCSes before they compute MUSes from them can clearly benefit from these improvements obtained in the MCSes enumeration task. We believe that this study opens other various paths for further research, too. Specifically, Property 3 opens the way for some parallelization of the enumeration task. It could be interesting to extend `rmr` using forms of local search to detect additional TCes. Also, we plan to adapt this study to the enumeration of all preferred MCSes when the clauses of Σ obey some preference pre-orderings. Finally, as skeptical reasoning in the presence of conflicting information can amount to computing the intersection of all maximal satisfiable subsets (MSSes), new progress in enumerating all MSSes, and thus all MCSes, can prove valuable for implementing such forms of reasoning.

9 Proofs

Proof of Property 1. By contradiction. Let us assume that $\Gamma \cup \{\alpha\}$ is not an MCS of Σ. That means that either (1) $\Sigma \setminus (\Gamma \cup \{\alpha\})$ is unsatisfiable or (2) $\exists \beta \in \Gamma \cup \{\alpha\}$ such that $\Sigma \setminus ((\Gamma \cup \{\alpha\}) \setminus \{\beta\})$ is satisfiable.

Assume (1). As Γ is defined as a Partial-MCS of $\langle \Sigma' \setminus \{\alpha\}, \Sigma \setminus \Sigma' \rangle$, we have that Γ is an MCS of $(\Sigma \setminus \{\alpha\})$. Thus, $(\Sigma \setminus \{\alpha\}) \setminus \Gamma$ is satisfiable. This entails that $((\Sigma \setminus \{\alpha\}) \cup \{\alpha\}) \setminus (\Gamma \cup \{\alpha\})$ is satisfiable. This latter formula can be simplified into $\Sigma \setminus (\Gamma \cup \{\alpha\})$, which is thus also satisfiable. This contradicts (1). Accordingly, (1) never occurs.

Assume (2). Let us suppose that $\beta \in \Gamma$ (then $\beta \neq \alpha$). Since Γ is a Partial-MCS of $\langle \Sigma' \setminus \{\alpha\}, \Sigma \setminus \Sigma' \rangle$, we have that Γ is an MCS of $\Sigma \setminus \{\alpha\}$. Thus, $(\Sigma \setminus \{\alpha\}) \setminus (\Gamma \setminus \{\beta\})$ is unsatisfiable. This entails that $((\Sigma \setminus \{\alpha\}) \cup \{\alpha\}) \setminus ((\Gamma \setminus \{\beta\}) \cup \{\alpha\})$ is unsatisfiable. Because we supposed $\beta \neq \alpha$, we have $(\Gamma \setminus \{\beta\}) \cup \{\alpha\} = (\Gamma \cup \{\alpha\}) \setminus \{\beta\}$. Consequently, $\Sigma \setminus ((\Gamma \cup \{\alpha\}) \setminus \{\beta\})$ is unsatisfiable. This contradicts the assumption and thus β and α must be the same clause.

Because Γ is a Partial-MCS of $\langle \Sigma' \setminus \{\alpha\}, \Sigma \setminus \Sigma' \rangle$, we have $(\Sigma' \setminus \{\alpha\}) \cap \Gamma = \emptyset$ and $\Gamma \subseteq \Sigma \setminus \Sigma'$. By hypothesis $\alpha \in \Sigma'$, this means $\alpha \notin \Gamma$ and thus $((\Sigma' \setminus \{\alpha\}) \cup \{\alpha\}) \cap \Gamma = \emptyset$. Hence $\Sigma' \cap \Gamma = \emptyset$. If $\beta = \alpha$, then we have $\Sigma \setminus ((\Gamma \cup \{\alpha\}) \setminus \{\alpha\}) = \Sigma \setminus \Gamma$ is satisfiable according to assumption (2). Since $\Sigma' \subseteq \Sigma$, we also have $\Sigma' \setminus \Gamma$ is satisfiable. Since $\Sigma' \cap \Gamma = \emptyset$ we have that Σ' is satisfiable. This contradicts the hypothesis that asserts that Σ' is an unsatisfiable subset of Σ.

Proof of Corollary 1. Property 1 allows us to conclude that $\Gamma \cup \{\alpha\}$ is an MCS of $\Sigma_1 \cup (\Sigma_2' \setminus \{\alpha\}) \cup (\Sigma_2 \setminus \Sigma_2') \cup \{\alpha\}$, and thus of $\Sigma_1 \cup \Sigma_2$. Since $\Gamma \cup \{\alpha\} \subseteq \Sigma_2$, we directly conclude that $\Gamma \cup \{\alpha\}$ is a Partial-MCS of $\langle \Sigma_1, \Sigma_2 \rangle$.

Proof of Property 2. From Corollary 1 it is easy to show that $\Gamma_1 \cup \{\alpha_1\}$ and $\Gamma_2 \cup \{\alpha_2\}$ are both Partial-MCSes of $\langle \Sigma_1, \Sigma_2 \rangle$. Now, let us show that these Partial-MCSes are different. It is sufficient to show that $\alpha_1 \notin \Gamma_2$. By definition of a Partial-MCS, we have $\Gamma_2 \subseteq \Sigma_2 \setminus \Sigma_2'$. As $\alpha_1 \in \Sigma_2'$, it is straightforward that $\alpha_1 \notin \Gamma_2$ and then $\Gamma_1 \cup \{\alpha_1\} \neq \Gamma_2 \cup \{\alpha_2\}$.

Proof of Property 3. Let us show that if $\Sigma^S \cup \Delta \cup \bigwedge_{s \in P} s \cup \bigwedge_{s \in N} \neg s$ then $\Sigma^S \cup \bigwedge_{s \in P} s$ and $\Delta \cup \bigwedge_{s \in N} \neg s$ are satisfiable and *vice versa*. ($\Rightarrow$) Straightforward. ($\Leftarrow$) If $\Sigma^S \cup \bigwedge_{s \in P} s$ and $\Delta \cup \bigwedge_{s \in N} \neg s$ are both satisfiable, then there exists a model μ that satisfies $\Sigma^S \cup \bigwedge_{s \in P} s$. By definition of Σ^S, whatever the satisfiable interpretation considered is, it is always possible to flip the truth value of selectors from 1 to 0 and keep the resulting interpretation μ_P such that μ_P is a model of Σ^S (this is possible because assigning a selector to 0 deactivates clauses of Σ^S and then weakens this formula). Thus, since $P \cap N = \emptyset$, we can construct the interpretation μ_P^N that is equivalent to μ_P except on the truth value of the selectors belonging to N where we force them to be assigned to 0. It is clear that μ_P^N is a model of $\Sigma^S \cup \bigwedge_{s \in P} s$. Now, let us prove that μ_P^N is a model of $\Delta \cup \bigwedge_{s \in N} \neg s$ too. By construction, Δ only contains positive clauses composed of selectors. Then, whatever the model of $\Delta \cup \bigwedge_{s \in N} \neg s$ considered is, it is always possible to flip the truth value of some selectors from 0 to 1 and keep this interpretation as a model of Δ. Thus, since all models of $\Delta \cup \bigwedge_{s \in N} \neg s$ must satisfy $\bigwedge_{s \in N} \neg s$, we can construct the interpretation $\bigwedge_{s \in N} \neg s \wedge \bigwedge_{s \in P} s$ that satisfies $\Delta \cup \bigwedge_{s \in N} \neg s$. Thus, since Δ is only constructed on selector variables, it is easy to show that μ_P^N also satisfies $\Delta \cup \bigwedge_{s \in N} \neg s$. Consequently, μ_P^N satisfies $\Sigma^S \cup \Delta \cup \bigwedge_{s \in P} s \cup \bigwedge_{s \in N} \neg s$.

Acknowledgments

This work has been supported in part by the CPER DATA project funded by the *Hauts-de-France* Region. We thank the reviewers for their useful comments.

References

[Audemard *et al.*, 2013] Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *Proc. of SAT 2013*, pages 309–317, 2013.

[Bacchus and Katsirelos, 2015] Fahiem Bacchus and George Katsirelos. Using minimal correction sets to more efficiently compute minimal unsatisfiable sets. In *Proc. of CAV 2015*, pages 70–86, 2015.

[Bacchus and Katsirelos, 2016] Fahiem Bacchus and George Katsirelos. Finding a collection of muses incrementally. In *Proc. of CPAIOR 2016*, pages 35–44, 2016.

[Bacchus *et al.*, 2014] Fahiem Bacchus, Jessica Davies, Maria Tsimpoukelli, and George Katsirelos. Relaxation search: A simple way of managing optional clauses. In *Proc. of AAAI 2014*, pages 835–841, 2014.

[Belov and Marques-Silva, 2011a] Anton Belov and João Marques-Silva. Accelerating MUS extraction with recursive model rotation. In *Proc. of FMCAD 2011*, pages 37–40, 2011.

[Belov and Marques-Silva, 2011b] Anton Belov and João Marques-Silva. Accelerating MUS extraction with recursive model rotation. In *Proc. of FMCAD 2011*, pages 37–40, 2011.

[Bobrow, 1980] Daniel G. Bobrow. Special issue non nonmonotonic logics. *Artificial Intelligence*, 13(1-2), 1980.

[Chen and Toda, 1995] Zhi-Zhong Chen and Seinosuke Toda. The complexity of selecting maximal solutions. *Information and Computation*, 119(2):231–239, 1995.

[Eén and Sörensson, 2003] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Selected Revised Papers of the 6th Int. Conf. of SAT 2003*, pages 502–518, 2003.

[Felfernig *et al.*, 2012] Alexander Felfernig, Monika Schubert, and Christoph Zehentner. An efficient diagnosis algorithm for inconsistent constraint sets. *AI EDAM*, 26(1):53–62, 2012.

[Fermé and Hansson, 2011] Eduardo L. Fermé and Sven Ove Hansson. AGM 25 years - twenty-five years of research in belief change. *Journal of Philosophical Logic*, 40(2):295–331, 2011.

[Grégoire *et al.*, 2007] Éric Grégoire, Bertrand Mazure, and Cédric Piette. Boosting a complete technique to find MSS and MUS thanks to a local search oracle. In *Proc. of IJCAI 2007*, pages 2300–2305, 2007.

[Grégoire *et al.*, 2014] Éric Grégoire, Jean-Marie Lagniez, and Bertrand Mazure. An experimentally efficient method for (MSS, CoMSS) partitioning. In *Proc. of (AAAI 2014)*, pages 2666–2673, 2014.

[Hamscher *et al.*, 1992] Walter Hamscher, Luca Console, and Johan de Kleer. *Readings in model-based diagnosis*. Morgan Kaufmann, 1992.

[Lagniez and Biere, 2013] Jean-Marie Lagniez and Armin Biere. Factoring out assumptions to speed up MUS extraction. In *Proc. of SAT 2013*, pages 276–292, 2013.

[Lagniez *et al.*, 2015] Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly. Coquiaas: A constraint-based quick abstract argumentation solver. In *Proc. of ICTAI 2015*, pages 928–935, 2015.

[Liffiton and Sakallah, 2008] Mark H. Liffiton and Karem A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *Journal of Automated Reasoning*, 40(1):1–33, 2008.

[Liffiton *et al.*, 2016] Mark H. Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva. Fast, flexible MUS enumeration. *Constraints*, 21(2):223–250, 2016.

[Marques-Silva *et al.*, 2013] João Marques-Silva, Federico Heras, Mikolás Janota, Alessandro Previti, and Anton Belov. On computing minimal correction subsets. In *Proc. of IJCAI 2013*, pages 615–622, 2013.

[Marques-Silva *et al.*, 2015] João Marques-Silva, Mikolás Janota, Alexey Ignatiev, and António Morgado. Efficient model based diagnosis with maximum satisfiability. In *Proc. of IJCAI 2015*, pages 1966–1972, 2015.

[Mencía *et al.*, 2015] Carlos Mencía, Alessandro Previti, and João Marques-Silva. Literal-based MCS extraction. In *Proc. of IJCAI 2015*, pages 1973–1979, 2015.

[Mencía *et al.*, 2016] Carlos Mencía, Alexey Ignatiev, Alessandro Previti, and Joao Marques-Silva. MCS extraction with sublinear oracle queries. In *Proc. of SAT 2016*, pages 342–360, 2016.

[Meseguer *et al.*, 2003] Pedro Meseguer, Noureddine Bouhmala, Taoufik Bouzoubaa, Morten Irgens, and Martí Sánchez-Fibla. Current approaches for solving over-constrained problems. *Constraints*, 8(1):9–39, 2003.

[Nadel *et al.*, 2014] Alexander Nadel, Vadim Ryvchin, and Ofer Strichman. Accelerated deletion-based extraction of minimal unsatisfiable cores. *Journal of Satisfiability JSAT*, 9:27–51, 2014.

[Previti and Marques-Silva, 2013] Alessandro Previti and João Marques-Silva. Partial MUS enumeration. In *Proc. of AAAI 2013*, 2013.

[Previti *et al.*, 2017] Alessandro Previti, Carlos Mencía, Matti Järvisalo, and Joao Marques-Silva. Improving MCS enumeration via caching. In *Proc. of SAT 2017*, pages 184–194, 2017.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.