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 maximal¹ 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 Σ \ Φ, 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 Λ of Boolean logic. ¬, ∨, ∧ and ⇒ represent the negation, disjunction, conjunction and material implication connectives, respectively.

¹We 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 \( \mu \) assigns values from \( \{0, 1\} \) to every Boolean variable, and, following usual compositional rules, to all formulas of \( \mathcal{L} \). A formula \( \alpha \) is satisfiable if there exists at least one interpretation \( \mu \) that satisfies \( \alpha \); \( \mu \) is then called a model of \( \alpha \). Accordingly, \( \mu \) satisfies a non-empty clause \( \alpha \) when there exists at least one literal of \( \alpha \) that is assigned 1 by \( \mu \); \( \mu \) satisfies a CNF \( \Sigma \) iff \( \mu \) satisfies all clauses of \( \Sigma \). 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 \( \Sigma \) be a CNF formula.

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

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

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

**Definition 3.** (MSS) A maximal satisfiable subset (in short, MSS) \( \Phi \) of \( \Sigma \) 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) \( \Psi \) of \( \Sigma \) is a set \( \Psi \subseteq \Sigma \) whose complement in \( \Sigma \), i.e., \( \Sigma \setminus \Psi \), is an MSS of \( \Sigma \).

**Example 1.** Let \( \Sigma \) be an unsatisfiable CNF formed by a set of clauses \( \{a_1, a_2, a_3, \alpha_4, \alpha_5, \alpha_6\} \), where \( a_1 = a \lor b \), \( a_2 = \neg a \lor b \), \( a_3 = a \lor \neg b \), \( \alpha_4 = \neg a \lor \neg b \), \( \alpha_5 = \neg b \), \( \alpha_6 = b \). The MCSes of \( \Sigma \) are \( \{a_1, a_6\}, \{a_2, a_6\}, \{a_3, a_5\} \) and \( \{a_4, a_5\} \). The MUSes of \( \Sigma \) are \( \{a_1, a_2, a_3, a_4\}, \{a_1, a_2, a_5\}, \{a_3, a_4, a_6\} \) and \( \{a_5, a_6\} \).

Notice that one MSS of \( \Sigma \) can be interpreted as one approximate solution of Max-SAT on \( \Sigma \), 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 \( \Sigma_1 \) and \( \Sigma_2 \) are sets of hard and soft clauses, respectively: hard clauses are required to belong to any Partial-MSS of \( \Sigma \); they thus do not belong to any Partial-MCS of \( \Sigma \). In the sequel, \( \Sigma_1 \) and \( \Sigma_2 \) always denote sets of hard and soft clauses, respectively. Notice that when \( \Sigma_1 \) is unsatisfiable, Partial-MSSes of \( \Sigma \) do not exist. In all the other cases, there always exists at least one Partial-MCS of \( \Sigma \) that is included in \( \Sigma_2 \). 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 \( \Sigma_1 \) 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 \( \alpha \) of \( \Sigma \) is augmented with its own (negated) selector, namely a fresh new literal \( \neg \alpha \). This yields a new (relaxed) CNF formula \( \Sigma' \). A clause \( \alpha \lor \neg \alpha \) in \( \Sigma' \) is thus activated (resp., deactivated) when the literal \( \alpha \) (resp., \( \neg \alpha \)) 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 \( \Sigma' \) is shown unsatisfiable under some such assumptions, modern SAT solvers can often extract a subset of the assumptions that causes \( \Sigma' \) 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 \( \Sigma' \) and make no ontological difference between the selectors and the other literals in \( \Sigma' \).

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 \( \Sigma \) when, at the same time, \( \Sigma \) is unsatisfiable and \( \Sigma \setminus \{\alpha\} \) is satisfiable.

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

Although \( \Sigma \) might not contain any TC, state-of-the-art MCS-finding tools often take advantage of TCs in the following way. Starting from the empty set, they iteratively and greedily construct a set \( \Sigma' \) at each step, clauses from \( \Sigma \) that have not been considered so far are inserted one by one in \( \Sigma' \) until \( \Sigma' \) becomes unsatisfiable. The last considered clause is a TC for \( \Sigma' \) 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 backbones 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 \( \Sigma_1 \) and a set of soft clauses \( \Sigma_2 \) as input parameters: namely, ExtractPartialMCS(\( \Sigma_1, \Sigma_2 \)).
4 MCSes Enumeration

Alg. 1 describes the typical skeleton of usual current approaches [Liffton and Sakallah, 2008; Liffton 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 S. Partial-MCSes of (s0, S, S) are computed while MCSes still exist, or, equivalently, while S0 ∪ S 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 \( \Sigma \)
Output : all MCSes of \( \Sigma \)

1. \( \Sigma^{S} \leftarrow \{ \alpha \lor \neg s_{\alpha} | \alpha \in \Sigma \} \); // with \( s_{\alpha} \) fresh variables
2. \( S \leftarrow \{ s_{\alpha} | \alpha \in \Sigma \} \); // a set of selectors
3. \( \Delta \leftarrow \emptyset \);
4. while \( \Sigma^{S} \cup \Delta \) is satisfiable do
5. \( M^{f} \leftarrow \text{ExtractPartialMCS}(\Sigma^{S} \cup \Delta, S) \);
6. output(\( M^{f} \));
7. \( \Delta \leftarrow \Delta \cup (\bigvee s_{\alpha} \in M^{-} s_{\alpha}) \); // blocking clauses

5 More MCSes Thanks to Transition Clauses

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

Property 1. Let \( \Sigma \) be an unsatisfiable CNF formula and let \( \Sigma^{f} \subseteq \Sigma \) such that \( \Sigma^{f} \) contains at least one TC \( \alpha \). For all Partial-MCSes \( \Gamma \) that can be built from \( (\Sigma^{f} \setminus \{ \alpha \}, \Sigma \setminus \Sigma^{f}) \), \( \Gamma \cup \{ \alpha \} \) is an MCS of \( \Sigma \).

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 \( \Sigma_{1} \) is satisfiable and \( \Sigma_{1} \cup \Sigma_{2} \) is unsatisfiable. Assume that \( \Sigma_{2}^{f} \subseteq \Sigma_{2} \) is such that \( \Sigma_{1} \cup \Sigma_{2}^{f} \) contains at least one TC \( \alpha \). For all Partial-MCSes \( \Gamma \) that can be built from \( (\Sigma_{1} \cup \Sigma_{2}^{f} \setminus \{ \alpha \}, \Sigma_{2} \setminus \Sigma_{2}^{f}) \), we have that \( \Gamma \cup \{ \alpha \} \) is a Partial-MCS of \( (\Sigma_{1}, \Sigma_{2}) \).

Now assume that we compute several TCes for a given subformula CNF \( \Sigma^{f} \subseteq \Sigma \). 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 \( \Sigma_{1} \) is satisfiable and \( \Sigma_{1} \cup \Sigma_{2} \) is unsatisfiable. Assume that \( \Sigma_{2}^{f} \subseteq \Sigma_{2} \) is such that \( \Sigma_{1} \cup \Sigma_{2}^{f} \) contains at least two different TCes \( \alpha_{1} \) and \( \alpha_{2} \) that occur in \( \Sigma_{2}^{f} \). Then, for all Partial-MCSes \( \Gamma_{1} \) and \( \Gamma_{2} \) that can be built from \( (\Sigma_{1} \cup \Sigma_{2}^{f} \setminus \{ \alpha_{1} \}, \Sigma_{2} \setminus \Sigma_{2}^{f}) \) and \( (\Sigma_{1} \cup \Sigma_{2}^{f} \setminus \{ \alpha_{2} \}, \Sigma_{2} \setminus \Sigma_{2}^{f}) \), respectively, we have that \( \Gamma_{1} \cup \{ \alpha_{1} \} \) and \( \Gamma_{2} \cup \{ \alpha_{2} \} \) are different Partial-MCSes of \( (\Sigma_{1}, \Sigma_{2}) \).

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 \( (\Sigma_{1} \cup \Sigma_{2}^{f}, U) \); 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 \( \Sigma_{1} \) is satisfiable and thus, that \( \Sigma_{1} \cup \Sigma_{2}^{f} \) is satisfiable. \( \Sigma_{2}^{f} \) is actually built from \( \Sigma_{2} \) 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 \( (\Sigma_{1}, \Sigma_{2}) \) are derived directly from the Partial-MCSes of \( (\Sigma_{1} \cup \Sigma_{2}^{f}, S) \), 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 \( (\Sigma_{1} \cup \Sigma_{2}^{f}, U) \) is empty as \( \Sigma_{1} \cup \Sigma_{2}^{f} \) is always satisfiable by construction. This is also the non-recursive step. Otherwise, the set of computed Partial-MCSes \( \Theta \) 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 \( \Sigma_{1} \cup \Sigma_{2}^{f} \cup M^{+} \) is satisfiable. \( s_{\alpha} \) is initialized to \( \top \), i.e., the tautology.

Next, the loop (lines 3–6) incrementally augments \( M^{+} \) with a sequence of \( s_{\alpha} \), which are removed from \( U \). This process is iterated while \( U \) is not empty and, at the same time, while \( s_{\alpha} \) is not a TC of \( \Sigma_{1} \cup \Sigma_{2}^{f} \cup M^{+} \cup \{ s_{\alpha} \} \). \( s_{\alpha} \) is a TC when \( \Sigma_{1} \cup \Sigma_{2}^{f} \cup M^{+} \cup \{ s_{\alpha} \} \) becomes unsatisfiable. Indeed, all the selectors inserted so far in \( M^{+} \) have guaranteed that \( \Sigma_{1} \cup \Sigma_{2}^{f} \cup M^{+} \) remained satisfiable. Thus, when an incoming \( s_{\alpha} \) makes the formula become unsatisfiable, this selector \( s_{\alpha} \) 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 \( s_{\alpha} \) is a TC. In this case, we look for a core \( \Gamma \) of the current unsatisfiable formula \( \Sigma_{1} \cup \Sigma_{2}^{f} \cup M^{+} \cup \{ s_{\alpha} \} \) (line 8). It is easy to
see that \( s_\alpha \) belongs to \( \Gamma \) and it is a TC of \( \Gamma \). 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 TCs. 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 TCs belonging to the core. In the next section, we present an approach to achieve this process, but for the moment, we just assume that TCs are extracted. Notice that we have no guarantee that this approach is efficient and identifies \( s_\alpha \) as a TC. For this reason, we directly add \( s_\alpha \) into \( T \), i.e., the set of transitions clauses, in order to ensure the termination of the algorithm. Then, for each identified TC \( s_\beta \) \( \in (T \land M^+) \) we recursively call the function with \( (\Sigma_1 \cup \Sigma_2^\pm \cup (\Gamma \setminus \{\neg s_\beta\}), U \cup (M^+ \land \Gamma)) \) as input. It is easy to show that these parameters match the conditions stated in Corollary 1. Accordingly, all the Partial-MCSs that we can compute from \( (\Sigma_1 \cup \Sigma_2^\pm \cup (\Gamma \setminus \{\neg s_\beta\}), U \cup (M^+ \land \Gamma)) \) can be augmented with \( s_\beta \) to yield a Partial-MCS of the input formula.

### Algorithm 2: TC-MCS \((\Sigma_1 \cup \Sigma_2^\pm, U)\);

**Input**: \((\Sigma_1 \cup \Sigma_2^\pm, U)\) a couple of CNF formulas

**Output**: \(\Theta\) a set of Partial-MCSes of \((\Sigma_1 \cup \Sigma_2^\pm, U)\)

1. if \(U = \emptyset\) then return \(\emptyset\);
2. \(\Theta \leftarrow \emptyset; M^+ \leftarrow \emptyset; s_\alpha \leftarrow T;\)
3. while \(U \neq \emptyset\) and \(\Sigma_1 \cup \Sigma_2^\pm \cup M^+ \cup \{s_\alpha\}\) is satisfiable do
   4. \(M^+ \leftarrow M^+ \cup \{s_\alpha\};\)
   5. \(s_\alpha \leftarrow \text{choose } g_1 \in U;\)
   6. \(U \leftarrow U \setminus \{s_\alpha\};\)
   7. if \(s_\alpha \notin M^+\) then
      8. \(\Gamma \leftarrow \text{Core}(\Sigma_1 \cup \Sigma_2^\pm \cup M^+ \cup \{s_\alpha\});\)
      9. \(T \leftarrow \{s_\alpha\} \cup \text{Find-TC}(\Gamma);\)
   10. foreach \(s_\beta \in T \land M^+\) do
      11. \(\Omega \leftarrow \text{TC-MCS}(\Sigma_1 \cup \Sigma_2^\pm \cup (\Gamma \setminus \{\neg s_\beta\}), U \cup (M^+ \land \Gamma));\)
      12. \(\Theta \leftarrow \Theta \cup (\beta \times \Omega);\)
13. return \(\Theta;\)

TC-MCS \((\Sigma_1 \cup \Sigma_2^\pm, U)\) 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 \(\Sigma\)

**Output**: all MCSes of \(\Sigma\)

1. \(\Sigma^S \leftarrow \{\alpha \lor \neg s_\alpha | \alpha \in \Sigma\};\) // with \(s_\alpha\) fresh variables
2. \(S \leftarrow \{s_\alpha | \alpha \in \Sigma\};\) // a set of selectors
3. \(\Delta \leftarrow \emptyset;\)
4. while \(\Sigma^S \land \Delta\) is satisfiable do
   5. \(\Theta \leftarrow \text{TC-MCS}(\Sigma^S \cup \Delta, S);\)
6. foreach \(M^- \in \Theta\) do
   7. output(M);\)
   8. \(\Delta \leftarrow \Delta \cup \{s_\alpha \in M^- \land s_\alpha\};\) // blocking clauses

Using Model Rotation

Let us now explain how the Find-TC procedure (line 9 of Alg. 2) computes additional TCs. A naive method would consist in computing all the MCSes of \(\Sigma_1 \cup \Sigma_2^\pm \cup M^+ \cup \{s_\alpha\}\) that are singletons contained in \(M^+\). As this complete direct method can be too time-consuming, we propose an incomplete process to compute additional TCs, based on the so-called recursive model rotation paradigm [Belov and Marques-Silva, 2011b] noted \(\text{rmr}\). 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 TCs 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. \(\text{rmr}\) is based on a model-theoretical perspective of TC: a TC of an unsatisfiable CNF formula \(\Sigma\) is any clause \(\alpha \in \Sigma\) such that there exists a complete interpretation \(\mu\) of \(\Sigma\) that satisfies \(\Sigma \setminus \{\alpha\}\) (and falsifies \(\alpha\), otherwise the formula \(\Sigma\) would be satisfiable). Starting from a model \(\mu\) of \(\Sigma \setminus \alpha\), \(\text{rmr}\) consists in flipping variables from \(\alpha\) and checking if this new interpretation \(\mu'\) satisfies all the clauses of \(\Sigma\) except some \(\alpha'\) of \(\Sigma\). In the positive case, \(\alpha'\) is marked as belonging to a TC of \(\Sigma\) provided that it was not already marked as such, and the process is recursively repeated with \(\mu'\) and \(\alpha'\).

\(\text{rmr}\) can thus compute several TCs 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 \(\mu\) of \(\Sigma_1 \cup \Sigma_2^\pm \cup M^+\) and proven that \(\Sigma_1 \cup \Sigma_2^\pm \cup M^+ \cup \{s_\alpha\}\) is unsatisfiable. \(\mu\) then plays the role of the initial interpretation and we keep the detected TCs belonging to \(\{\beta \in \Sigma_2 | s_\beta \in M^+\}\) only. Finally, the set of selectors associated with the discovered TCs can be returned. It is important to note that we make sure that the clause selectors are never flipped by the process.

Although the \(\text{rmr}\) 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 \(\Delta\) of blocking clauses becomes too large. To avoid such a drawback, we point out some sufficient conditions to deprive clauses of \(\Delta\) of the \(\text{rmr}\) process.

First, let us show the possible critical role of \(\Delta\) in the search for additional TCs. Consider \((\Sigma^S, S)\) with \(\Sigma^S = \{\neg s_1 \lor a \lor b, \neg s_2 \lor \neg a, \neg s_3 \lor \neg b\}\). 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 \((\Sigma^S \cup \Delta, S)\). Let us suppose that we stop the main loop of TC-MCS when \(M^+ = \{s_1, s_2\}\) and \(s_3 = s_3\). The condition in line 7 is satisfied; one core that is the complete formula is computed. Then, we search more TCs 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 TCs and twice the same MCS will be computed. Obviously, a posteriori checking whether an MCS has already been computed by consulting \(\Delta\) can be too time-consuming.

However, a useful feature is that \(\Sigma^S\) and \(\Delta\) do not share
literals. Indeed, $\Delta$ is a set of positive clauses composed of selectors whereas these selectors occur negatively in $\Sigma^S$. Accordingly, the satisfiability 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 $S$, $\Sigma^S$ a CNF formula augmented with the set of selectors $S$ and $\Delta$ 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 $\Sigma^S \cup \bigwedge_{s \in P} s \cup \bigwedge_{s \in N} \neg s$ 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 $\Delta$ 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 $\Delta$ in the $\text{rmr}$ 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 $\Delta$. 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 $\Delta$ during the $\text{rmr}$ 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 $\text{rmr}$. 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 $\text{rmr}$ 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](image)

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
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 approaches 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 \textsc{rmr} using forms of local search to detect additional TCSes. Also, we plan to adapt this study to the enumeration of all preferred MCSes when the clauses of \( \Sigma \) 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 \( \Sigma \). 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 \( \Gamma \) is defined as a Partial-MCS of \( \{ \Sigma \setminus \{ \alpha \}, \Sigma \setminus \Sigma' \} \), we have that \( \Gamma \) is an MCS of \( \{ \Sigma \setminus \{ \alpha \} \} \). Thus, \( \{ \Sigma \setminus \{ \alpha \} \} \setminus \Gamma \) is satisfiable. This entails that \( \{ \{ \Sigma \setminus \{ \alpha \} \} \cup \{ \alpha \} \} \setminus \Gamma \) is satisfiable. This latter formula can be simplified into \( \{ \Sigma \setminus \Gamma \} \). This 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 \( \Gamma \) is a Partial-MCS of \( \{ \Sigma \setminus \{ \alpha \}, \Sigma \setminus \Sigma' \} \), we have that \( \Gamma \) is an MCS of \( \Sigma \setminus \{ \alpha \} \). Thus, \( \{ \Sigma \setminus \{ \alpha \} \} \setminus \{ \beta \} \) is unsatisfiable. This entails that \( \{ \{ \Sigma \setminus \{ \alpha \} \} \cup \{ \alpha \} \} \setminus \{ \beta \} \) is satisfiable. Because we supposed \( \beta \neq \alpha \), we have \( \{ \beta \} \cup \{ \alpha \} = \{ \Gamma \cup \{ \alpha \} \} \setminus \{ \beta \} \). Consequently, \( \Sigma \setminus ((\Gamma \cup \{ \alpha \}) \setminus \{ \beta \}) \) is unsatisfiable. This contradicts the assumption and thus \( \beta \) and \( \alpha \) must be the same clause.

Because \( \Gamma \) is a Partial-MCS of \( \{ \Sigma \setminus \{ \alpha \}, \Sigma \setminus \Sigma' \} \), 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 \( \Sigma' \cap \Gamma = \emptyset \) and we have that \( \Sigma' \) is satisfiable. This contradicts the hypothesis that asserts that \( \Sigma' \) is an unsatisfiable subset of \( \Sigma \).

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 \( \{ \Sigma_1, \Sigma_2 \} \).

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 \( \{ \Sigma_1, \Sigma_2 \} \). Now, let us show that these Partial-MCSes are different. It is sufficient to show that \( \alpha_1 \neq \Gamma_2 \). 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 \( \alpha_1 \neq \Gamma_2 \) and then \( \Gamma_1 \cup \{ \alpha_1 \} \neq \Gamma_2 \cup \{ \alpha_2 \} \).

Proof of Property 3. Let us show that if \( \Sigma^S \cup \Delta \cup \bigwedge_{\alpha \in P} s \cup \bigwedge_{\alpha \in N} \neg s \) and \( \bigwedge_{\alpha \in N} \neg s \) are satisfiable, then \( \Sigma^S \cup \bigwedge_{\alpha \in P} s \) and \( \bigwedge_{\alpha \in N} \neg s \) are both satisfiable, then there exists a model \( \mu \) that satisfies \( \Sigma^S \cup \bigwedge_{\alpha \in P} s \). By definition of \( \Sigma^S \), 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 \( \mu_p \) such that \( \mu_p^S \) is not a model of \( \Sigma^S \) (this is possible because assigning a selector to 0 deactivates clauses of \( \Sigma^S \) and then weakens this formula). Thus, since \( P \cap N = \emptyset \), we can construct the interpretation \( \mu_p^N \) that is equivalent to \( \mu_p^S \) except on the truth value of the selectors belonging to \( N \) where we force them to be assigned to 0. It is clear that \( \mu_p^N \) is a model of \( \Sigma^S \cup \bigwedge_{\alpha \in N} \neg s \). Now, let us prove that \( \mu_p^N \) is a model of \( \Sigma^S \cup \bigwedge_{\alpha \in N} \neg s \). By construction, \( \Delta \) only contains positive clauses composed of selectors. Then, whatever the model of \( \Delta \cup \bigwedge_{\alpha \in N} \neg s \) 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 \( \Delta \). Thus, since all models of \( \Delta \cup \bigwedge_{\alpha \in N} \neg s \) satisfy \( \bigwedge_{\alpha \in N} \neg s \), we can construct the interpretation \( \bigwedge_{\alpha \in N} \neg s \wedge \bigwedge_{\alpha \in P} s \) that satisfies \( \Delta \cup \bigwedge_{\alpha \in N} \neg s \). Thus, since \( \Delta \) is only constructed on selector variables, it is easy to show that \( \mu_p^N \) also satisfies \( \bigwedge_{\alpha \in N} \neg s \). Consequently, \( \mu_p^N \) satisfies \( \Sigma^S \cup \Delta \cup \bigwedge_{\alpha \in P} s \cup \bigwedge_{\alpha \in N} \neg s \).
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


