On Computing Minimal Equivalent Subformulas

Anton Belov[1], Mikoláš Janota[2], Ines Lynce[2], and Joao Marques-Silva[1,2]

[1] CASL, University College Dublin, Ireland
[2] IST/INESC-ID, Technical University of Lisbon, Portugal

Abstract. A propositional formula in Conjunctive Normal Form (CNF) may contain redundant clauses — clauses whose removal from the formula does not affect the set of its models. Identification of redundant clauses is important because redundancy often leads to unnecessary computation, wasted storage, and may obscure the structure of the problem. A formula obtained by the removal of all redundant clauses from a given CNF formula $\mathcal{F}$ is called a Minimal Equivalent Subformula (MES) of $\mathcal{F}$. This paper proposes a number of efficient algorithms and optimization techniques for the computation of MESes. Previous work on MES computation proposes a simple algorithm based on iterative application of the definition of a redundant clause, similar to the well-known deletion-based approach for the computation of Minimal Unsatisfiable Subformulas (MUSes). This paper observes that, in fact, most of the existing algorithms for the computation of MUSes can be adapted to the computation of MESes. However, some of the optimization techniques that are crucial for the performance of the state-of-the-art MUS extractors cannot be applied in the context of MES computation, and thus the resulting algorithms are often not efficient in practice. To address the problem of efficient computation of MESes, the paper develops a new class of algorithms that are based on the iterative analysis of subsets of clauses. The experimental results, obtained on representative problem instances, confirm the effectiveness of the proposed algorithms. The experimental results also reveal that many CNF instances obtained from the practical applications of SAT exhibit a large degree of redundancy.

1 Introduction

A propositional formula in Conjunctive Normal Form (CNF) is redundant if some of its clauses can be removed without changing the set of models of the formula. Formula redundancy is often desirable. For example, modern Conflict-Driven Clause Learning (CDCL) Boolean Satisfiability (SAT) solvers learn redundant clauses [33,1], which are often essential for solving practical instances of SAT. However, redundancy can also be undesirable. For example, in knowledge bases formula redundancy leads to the use of unnecessary storage and computational resources [25]. Another example is the undesirable redundant clauses in the CNF representation of belief states in a conformant planner [37,36]. In the context of probabilistic reasoning systems, concise representation of conditional independence information can be computed by removing redundant clauses from certain propositional encodings [30]. More generally, given the wide range of applications of SAT, one can pose the question: does a given problem domain encoder introduce redundancy, and if so, how significant is the percentage of redundant clauses? Removal of redundancies can also find application in solving problems from different complexity classes, for example Quantified Boolean Formulas (QBF). Besides propositional logic formulas, the problem of the identification of redundant constraints is relevant in other domains. Concrete examples include Constraint Satisfaction Problems (CSP) [12,10,9], Satisfiability Modulo Theories (SMT) [35], and Ontologies [19].

This paper addresses the problem of computing an irredundant subformula $\mathcal{E}$ of a redundant CNF formula $\mathcal{F}$, such that $\mathcal{F}$ and $\mathcal{E}$ have the same set of models. Such subformula $\mathcal{E}$ will be referred to as a *Minimal Equivalent Subformula (MES)* of $\mathcal{F}$. Previous work on removing redundant clauses from CNF formulas proposes a direct approach [7], which iteratively checks the definition of redundant clause and removes the clauses that are found to be redundant. While the direct approach is similar to the well-known deletion-based approach for the computation of Minimal Unsatisfiable Subformulas (MUSes), most of the techniques developed for the extraction of MUSes (e.g. see [18,14,27]) have not been extended to the computation of MESes.

The paper has five main contributions, summarized as follows. First, the paper shows that many of the existing MUS extraction algorithms can be extended to the computation of MESes. Since efficient MUS extraction uses a number of key techniques for reducing the total number of SAT solver calls — namely, clause set refinement [13,29,28] and model rotation [28,4] — this paper analyzes these techniques in the context of MES extraction. The second contribution of the paper is, then, to show that model rotation can be integrated, and, in fact, improved, in MES extraction, however clause set refinement *cannot* be used in MES algorithms derived from the existing MUS algorithms. Third, the paper proposes a reduction from MES computation problem to group-MUS computation problem [26,29]; this reduction enables the use of *both* model rotation and clause set refinement for MES extraction. Fourth, given that the approach of reduction to group-MUS can result in hard instances of SAT, the paper proposes an incremental reduction of MES to group-MUS extraction, that involves the separate analysis of subsets of clauses. Fifth, and finally, the paper develops solutions for checking that computed MESes are correct. These solutions find application in settings where independent certification is required.

Experimental results, obtained on representative satisfiable instances from past SAT competitions, show that the new algorithms for MES computation achieve significant performance gains over the basic algorithms, and allow targeting redundancy removal for reasonably sized formulas. In addition, the experimental results show that *many* CNF formulas, from a wide range of application domains, contain a significant percentage of redundant clauses, in some cases exceeding 90% of the original clauses.

2 Preliminaries

Standard definitions for propositional logic are assumed. Propositional formulas are defined over a set of propositional variables $X = \{x_1, \ldots, x_n\}$. A CNF formula $\mathcal{F}$ is a conjunction of disjunctions of literals (*clauses*), where a literal is a variable or its complement. Unless otherwise stated, a CNF formula will be referred to as a formula. Formulas are represented by letters with calligraphic fonts, e.g. $\mathcal{F}$, $\mathcal{E}$, $\mathcal{S}$, $\mathcal{W}$, etc. When necessary, subscripts are used. Clauses are represented by c or c_i, $i = 1, \ldots, m$. A CNF formula can also be viewed as a multiset of non-tautologous clauses where a clause is a (multi)set of literals. The two representations are used interchangeably, and are clear from the context. A *truth assignment* μ is a mapping from X to $\{0,1\}$, $\mu : X \to \{0,1\}$. The truth value of a clause c or formula $\mathcal{F}$, given a truth assignment μ, is represented as $c[\mu]$ and $\mathcal{F}[\mu]$, respectively. A truth assignment is a *model* if it satisfies all clauses in $\mathcal{F}$, i.e. $\mathcal{F}[\mu] = 1$. Two formulas $\mathcal{F}_1$ and $\mathcal{F}_2$ are *equivalent*, $\mathcal{F}_1 \equiv \mathcal{F}_2$, if they have the same set of models. The negation of a clause c, denoted by $\neg c$ represents a conjunction of unit clauses, one for each literal in c. It will also be necessary to negate CNF formulas, e.g. $\neg\mathcal{F}$. The following CNF encoding for $\neg\mathcal{F}$ is used [34]. An auxiliary variable u_i is associated with each $c_i \in \mathcal{F}$, defining a set of variables U. For each $l_{i,j} \in c_i$, create binary clauses $(\neg l_{i,j} \vee \neg u_i)$. Finally, create a clause $(\vee_{u_i \in U} u_i)$. This CNF encoding is represented as CNF$(\cdot)$. For example, CNF$(\neg c)$ and CNF$(\neg\mathcal{F})$ denote, respectively, the CNF encoding of $\neg c$ and $\neg\mathcal{F}$ described above; both CNF$(\neg c)$ and CNF$(\neg\mathcal{F})$ can be used in set context to denote sets of clauses. Calls to a SAT solver are represented with SAT$(\cdot)$.

2.1 MUSes and MESes

The following definition of Minimal Unsatisfiable Subformulas (MUSes) is used [6].

Definition 1 (MUS). *$\mathcal{M} \subseteq \mathcal{F}$ is a* Minimal Unsatisfiable Subformula *(MUS) iff $\mathcal{M}$ is unsatisfiable and $\forall_{\mathcal{S} \subsetneq \mathcal{M}}, \mathcal{S}$ is satisfiable.*

MUS extraction algorithms can be broadly characterized as deletion-based [8,3] or insertion-based. Moreover, insertion-based algorithms can be characterized as linear search [11] or dichotomic search [23,21]. Recent work proposed insertion-based MUS extraction with relaxation variables and AtMost1 constraints [28]. In practice, the most efficient MUS extraction algorithms are organized as deletion-based but operate as insertion-based to allow the integration of essential pruning techniques. These algorithms are referred to as *hybrid* [28]. Examples of pruning techniques used to reduce the number of SAT solver calls include *clause set trimming* (during preprocessing), *clause set refinement* and *model rotation* [13,29,28,4]. Clause set refinement [13,28] exploits the SAT solver *false* (or *unsatisfiable*) outcomes to reduce the set of clauses that need to be analyzed. This consists of removing clauses that are *not* included in the MUS being constructed, e.g. by restricting the target set of clauses to the unsatisfiable subset computed by the SAT solver. In contrast, model rotation exploits the SAT solver *true* (or *satisfiable*) outcomes to also reduce the set of clauses that need to be analyzed. In this case, models are used to identify clauses that *must* be included in the MUS being constructed. Recent experimental data [28] indicates that these two techniques are *essential* for MUS extraction on large application problem instances.

Motivated by several applications, MUSes and related concepts have been extended to CNF formulas where clauses are partitioned into disjoint sets called *groups* [26,29].

Definition 2 (Group-Oriented MUS). *Given an explicitly partitioned unsatisfiable CNF formula $\mathcal{F} = \mathcal{D} \cup \mathcal{G}_1 \cdots \cup \mathcal{G}_n$, a* group oriented MUS *(or,* group-MUS*) of $\mathcal{F}$ is a subset $\mathcal{F}' = \mathcal{D} \cup \mathcal{G}_{i_1} \cup \cdots \cup \mathcal{G}_{i_k}$ of $\mathcal{F}$ such that $\mathcal{F}'$ is unsatisfiable and, for every $1 \leq j \leq k$, $\mathcal{F}' \setminus \mathcal{G}_{i_j}$ is satisfiable.*

The group $\mathcal{D}$ in the above definition is called a *don't care* group, and the explicitly partitioned CNF formulas as above are referred to as *group-CNF* formulas.

MUSes are a special case (for unsatisfiable formulas) of irredundant subformulas [25]. The following definitions will be used throughout.

Definition 3 (Redundant/Irredundant Clause/Formula). *A clause $c \in \mathcal{F}$ is said to be* redundant *in $\mathcal{F}$ if $\mathcal{F} \setminus \{c\} \vDash c$ or, equivalently, $\mathcal{F} \setminus \{c\} \cup \mathrm{CNF}(\neg c) \vDash \bot$. Otherwise, c is said to be* irredundant *in $\mathcal{F}$. A formula $\mathcal{F}$ is redundant if it has at least one redundant clause; otherwise it is irredundant.*

Irredundant subformulas of (redundant) formulas are referred to as *irredundant equivalent subsets* [25] and as *irredundant cores* [24]. In this paper, irredundant subformulas are referred to as *Minimal Equivalent Subformulas* (MESes), by analogy with MUSes.

Definition 4 (MES). *$\mathcal{E} \subseteq \mathcal{F}$ is a* Minimal Equivalent Subformula *(MES) iff $\mathcal{E} \equiv \mathcal{F}$ and $\forall_{\mathcal{Q} \subsetneq \mathcal{E}}, \mathcal{Q} \not\equiv \mathcal{F}$.*

Clearly, an MES is irredundant. Moreover, deciding whether a CNF formula is an MES is D^P-complete [25]. In the case of group-CNF formulas, the concept of *group-oriented MES (group-MES)* can be defined analogously to Definition 2.

2.2 Related Work

MUSes find a wide range of practical applications, and have been extensively studied (see [18,14,27] for recent overviews, and [23,21,16,17] for connections with CSP). The problem of computing minimal (or irredundant) representations of CNF formulas (and related problems) has been the subject of extensive research (e.g. [2,20,7,25,24]). Complexity characterizations of redundancy problems in logic can be found in [25]. An algorithm for computing an MES based on the direct application of the definition of clause redundancy is studied in [7]. More recently, properties of MESes are studied in [24]. Approximate solutions for redundancy removal based on unit propagation are proposed in [15,31]. Applications (of restricted forms) of redundancy removal can be found in [22,37,36,15,25,30]. The importance of redundant clauses in CDCL SAT solvers is addressed in [33,1]. Redundancy problems have been studied in many other settings, e.g. [12,10,9,35,19].

3 MES Extraction Algorithms

This section develops several new approaches for computing one MES of a CNF formula $\mathcal{F}$. The first solution consists of adapting *any* MUS extraction algorithm based on identification of so-called transition clauses, for MES extraction. Afterwards, key techniques used in MUS extraction are studied. Model

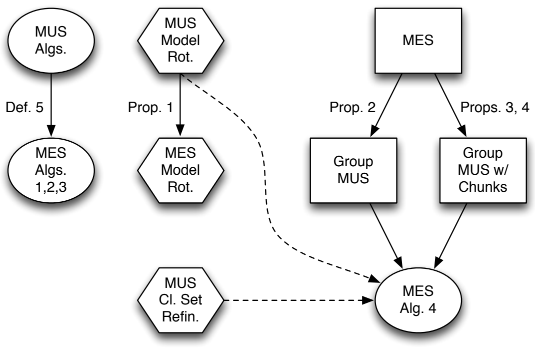

Fig. 1. Approaches to MES extraction.

rotation [28,4] is applied to MES extraction, and it is argued that clause set refinement [13,29,28] cannot be directly applied to MES extraction algorithms resulting from adapting existing MUS extraction algorithms. Next, a reduction from MES to group-MUS formulation [26,29] is developed, which enables the use of both model rotation and clause set refinement. Although the reduction of MES to group-MUS extraction enables the integration of key techniques, it is also the case that the resulting instances of SAT are hard to solve. This section concludes by developing an incremental reduction from MES to group-MUS extraction, which produces much easier instances of SAT. Figure 1 summarizes the approaches to MES extraction described in the remainder of this section.

3.1 From MUS Extraction to MES Extraction

A key definition in MUS extraction algorithms is that of *transition clause* [18]. A transition clause c is such that, if added to a satisfiable subformula $\mathcal{R}$, the resulting subformula is unsatisfiable. This definition can be generalized for MES extraction as follows. Let $\mathcal{R} \subsetneq \mathcal{F}$ denote a (reference) subformula of $\mathcal{F}$ which is to be extended to be equivalent to $\mathcal{F}$. Observe that $\mathcal{F} \vDash \mathcal{R}$, and so our goal is to extend $\mathcal{R}$ with a clause c, such that $\mathcal{R} \cup \{c\} \vDash \mathcal{F}$, and so $\mathcal{R} \cup \{c\} \equiv \mathcal{F}$.

Definition 5 (Witness of Equivalence). *Let $\mathcal{S}$ denote a subformula of $\mathcal{F}$, $\mathcal{S} \subsetneq \mathcal{F}$, with $\mathcal{S} \nvDash \mathcal{F} \setminus \mathcal{S}$, and let $c \in \mathcal{F} \setminus \mathcal{S}$. If $\mathcal{S} \cup \{c\} \vDash \mathcal{F} \setminus (\mathcal{S} \cup \{c\})$, then c is a* witness *of $\mathcal{S} \cup \{c\} \equiv \mathcal{F}$.*

The above definition can now be used to adapt *any* MUS extraction algorithm for MES extraction. The remainder of this section illustrates how this can be achieved. Let $\mathcal{F}$ be a CNF formula partitioned as follows, $\mathcal{F} = \mathcal{E} \cup \mathcal{R} \cup \mathcal{S}$. A subformula $\mathcal{R}$ is redundant in $\mathcal{F}$ iff $\mathcal{E} \cup \mathcal{S} \vDash \mathcal{R}$. Given an under-approximation $\mathcal{E}$ of an MES of $\mathcal{F}$, and a working subformula $\mathcal{S} \subsetneq \mathcal{F}$, the objective is to decide whether $\mathcal{E}$ and $\mathcal{S}$ entail all the other clauses of $\mathcal{F}$. MUS extraction algorithms can be organized as (see Section 2.1): (i) deletion-based [8,3], (ii) insertion-based [11,38], (iii) insertion-based with dichotomic search [23,21], and (iv) insertion-based with relaxation variables [28].

Algorithm 1: Plain deletion-based MES extraction

```
   Input  : Formula $\mathcal{F}$
   Output: MES $\mathcal{E}$
 1 begin
 2     $\mathcal{S} \leftarrow \mathcal{F}$
 3     $\mathcal{E} \leftarrow \emptyset$                                   // MES under-approximation
 4     while $\mathcal{S} \neq \emptyset$ do
 5         $c \leftarrow$ SelectRemoveClause($\mathcal{S}$)           // Get a clause from $\mathcal{S}$
 6         $\mathcal{W} \leftarrow \{c\}$
 7         if SAT($\mathcal{E} \cup \mathcal{S} \cup$ CNF($\neg\mathcal{W}$)) then
 8             $\mathcal{E} \leftarrow \mathcal{E} \cup \{c\}$              // Add clause $c$ to $\mathcal{E}$ if $\mathcal{E} \cup \mathcal{S} \nvDash c$
 9     return $\mathcal{E}$                                          // Final $\mathcal{E}$ is MES
10 end
```

Algorithm 2: Plain insertion-based MES extraction

```
   Input  : Formula $\mathcal{F} = \{c_1, \ldots, c_m\}$
   Output: MES $\mathcal{E}$
 1 begin
 2     $\mathcal{W} \leftarrow \mathcal{F}$
 3     $\mathcal{E} \leftarrow \emptyset$                                   // MES under-approximation
 4     while $\mathcal{W} \neq \emptyset$ do
 5         $(\mathcal{S}, c_r) \leftarrow (\emptyset, \emptyset)$
 6         while SAT($\mathcal{E} \cup \mathcal{S} \cup$ CNF($\neg\mathcal{W}$)) do
 7             $c_r \leftarrow$ SelectRemoveClause($\mathcal{W}$)      // Extend $\mathcal{S}$ while $\mathcal{E} \cup \mathcal{S} \nvDash \mathcal{W}$
 8             $\mathcal{S} \leftarrow \mathcal{S} \cup \{c_r\}$
 9         $\mathcal{E} \leftarrow \mathcal{E} \cup \{c_r\}$                              // $c_r$ is in MES
10         $\mathcal{W} \leftarrow \mathcal{S} \setminus \{c_r\}$
11     return $\mathcal{E}$                                       // Final $\mathcal{E}$ is an MES
12 end
```

The pseudo-code for deletion-based MES extraction is shown in Algorithm 1. At each step, the algorithm uses an MES under-approximation $\mathcal{E}$, a set of (remaining) clauses $\mathcal{S}$, and a target clause c, to check whether $\mathcal{E} \cup \mathcal{S} \vDash c$. If this is not the case, i.e. if $\mathcal{E} \cup \mathcal{S} \nvDash c$, then c is witness of $\mathcal{E} \cup \mathcal{S} \cup \{c\} \equiv \mathcal{F}$, and so c is added to $\mathcal{E}$; otherwise, c is discarded. Observe that the deletion-based MES extraction algorithm corresponds to the direct implementation of the definition of redundant clause (e.g. see [7]).

The pseudo-code for insertion-based linear and dichotomic search MES extraction are shown in Algorithms 2 and 3, respectively. Both algorithms iteratively add clauses to set $\mathcal{S}$ while $\mathcal{E} \cup \mathcal{S} \nvDash \mathcal{W}$. The last clause included in $\mathcal{S}$ such that $\mathcal{E} \cup \mathcal{S} \vDash \mathcal{W}$ is the witness of equivalence. The main difference between algorithms 2 and 3 is how the witness of equivalence is searched for.

Recent experimental data indicates that the most efficient MUS extraction algorithms are deletion-based (or variants) [28]. Since insertion-based MES algorithms require SAT solver calls with (possibly large) complemented formu-

Algorithm 3: MES extraction with dichotomic search

```
Input  : Formula $\mathcal{F} = \{c_1, \ldots, c_m\}$
Output: MES $\mathcal{E}$
1  begin
2      $\mathcal{W} \leftarrow \mathcal{F}$
3      $\mathcal{E} \leftarrow \emptyset$                                   // MES under-approximation
4      while $\mathcal{W} \neq \emptyset$ do
5          $(\mathsf{min}, \mathsf{mid}, \mathsf{max}) \leftarrow (0, 0, |\mathcal{W}|)$
6          repeat
7              $\mathcal{S} \leftarrow \{c_1, \ldots, c_{\mathsf{mid}}\}$          // Extract sub-sequence of $\mathcal{W}$
8              if SAT$(\mathcal{E} \cup \mathcal{S} \cup$ CNF$(\neg(\mathcal{W} \setminus \mathcal{S})))$ then
9                  $\mathsf{min} \leftarrow \mathsf{mid} + 1$                  // Extend $\mathcal{S}$ if $\mathcal{E} \cup \mathcal{S} \nvDash \mathcal{W} \setminus \mathcal{S}$
10             else
11                 $\mathsf{max} \leftarrow \mathsf{mid}$                      // Reduce $\mathcal{S}$ if $\mathcal{E} \cup \mathcal{S} \vDash \mathcal{W} \setminus \mathcal{S}$
12             $\mathsf{mid} \leftarrow \lfloor (\mathsf{min} + \mathsf{max})/2 \rfloor$
13         until $\mathsf{min} = \mathsf{max}$
14         if $\mathsf{min} > 0$ then
15             $\mathcal{E} \leftarrow \mathcal{E} \cup \{c_{\mathsf{min}}\}$
16         $\mathcal{W} \leftarrow \{c_i \mid i < \mathsf{min}\}$
17     return $\mathcal{E}$                                        // Final $\mathcal{E}$ is an MES
18 end
```

las, deletion-based MES extraction algorithms are expected to outperform the insertion-based ones. This is confirmed by the results in Section 5.

Efficient MUS extraction algorithms [13,29,28] *must* use a number of additional techniques for reducing the number of SAT solver calls. These techniques include *clause set refinement* [13,28] and *model rotation* [28,4]. The next section shows how model rotation can be used in MES extraction. In contrast, clause set refinement *cannot* be applied in the algorithms described above.

Example 1. Let $\mathcal{F} = (x_1) \wedge (x_1 \vee x_3) \wedge (x_2)$, and consider the execution of Algorithm 1. First, clause (x_1) is removed and the resulting formula is satisfiable with $x_1 = 0$, $x_2 = x_3 = 1$; hence (x_1) is irredundant. Second, clause $(x_1 \vee x_3)$ is removed and the resulting formula is unsatisfiable; hence $(x_1 \vee x_3)$ is redundant. Moreover, the computed unsatisfiable core is $\{(x_1), (\neg x_1)\}$, where $(\neg x_1)$ is taken from $\neg(x_1 \vee x_3)$. However, (x_2) is not in the unsatisfiable core, but it cannot be removed.

As illustrated by the previous example, since MES extraction algorithms require adding the negation of a subformula of $\mathcal{F}$, the computed unsatisfiable cores depend on this negation, which changes as the algorithm executes. Thus, a computed unsatisfiable core provides no information about which clauses need not be considered further. Despite this negative result, Sections 3.3 and 3.4 develop solutions for MES extraction that enable clause set refinement.

3.2 Using Model Rotation in MES Extraction

The definition of irredundant clause (see Definition 3) can be associated with specific truth assignments, that can serve as *witnesses* of irredundancy.

Proposition 1. *A clause $c \in \mathcal{F}$ is irredundant in $\mathcal{F}$ if and only if there exists a truth assignment μ, such that $(\mathcal{F} \setminus \{c\})[\mu] = 1$ and $c[\mu] = 0$.*

Proof. Follows directly from the definition of an irredundant clause and the semantics of $\mathcal{F} \setminus \{c\} \not\models c$. □

In the context of MUS extraction, Proposition 1 is used as a basis of *model rotation* [28,4] — a powerful optimization technique that in practice allows to significantly reduce the number of calls to SAT solver, and results in multiple orders of magnitude reduction in run-times of hybrid MUS extraction algorithms on industrial problem instances (cf. [4]). Proposition 1 has also been used in the context of local search for MUS extraction [32].

In the context of MES extraction, model rotation can be instrumented as follows. If a SAT solver call returns satisfiable (see, for example, line 7 in Algorithm 1), then we can use the model computed by the solver to look for other irredundant clauses using Proposition 1. Note that the clauses of the negation of the working formula (e.g. $\text{CNF}(\neg\mathcal{W})$ in Algorithm 1) can be disregarded, and the objective is to modify the returned model such that *exactly* one other clause of the working formula becomes falsified. Iteratively, each literal from the currently falsified clause is flipped, and one checks whether the formula has another *single* falsified clause, which can then be declared irredundant according to Proposition 1, *without* a SAT solver call. This process is continued recursively from the newly detected irredundant clause.

Model rotation for MES extraction can be improved further. Since the formula is satisfiable, model rotation may reach assignments where *all* clauses are satisfied (note that this situation is not possible in MUS extraction). In this case, rather than terminating the process, clauses with the smallest number of satisfied literals are selected, the satisfied literals are flipped, and again one checks whether the formula has a single unsatisfied clause. As demonstrated in Section 5, this *improved model rotation* is very effective for redundancy removal.

3.3 A Reduction of MES to Group-MUS

As argued in Section 3.1, although MUS extraction algorithms can be modified for MES extraction, clause set refinement [13,29,28] cannot be used. In the context of MUS computation, this technique is *paramount* for reducing the number of SAT solver calls in instances with many redundant clauses. This section develops a reduction of MES computation problem to group-MUS computation problem [26,29] — recall Definition 2. A key advantage of this reduction is that it enables clause set refinement.

Proposition 2 (MES to Group-MUS Reduction). *Given a CNF formula $\mathcal{F}$, and any $\mathcal{E} \subseteq \mathcal{F}$, let $R_{\mathcal{F}}(\mathcal{E})$ be the group-CNF formula $\mathcal{D} \cup \bigcup_{c \in \mathcal{E}} \mathcal{G}_c$, where $\mathcal{D} = \text{CNF}(\neg\mathcal{F})$ is the don't care group, and $\mathcal{G}_c = \{c\}$. Then, $\mathcal{E}$ is an MES of $\mathcal{F}$ if and only if $R_{\mathcal{F}}(\mathcal{E})$ is a group-MUS of $R_{\mathcal{F}}(\mathcal{F})$.*

Proof. We prove both directions simultaneously. Since $R_{\mathcal{F}}(\mathcal{E}) = \text{CNF}(\neg\mathcal{F}) \cup \mathcal{E}$, we have that $\mathcal{E} \vDash \mathcal{F}$ (and so $\mathcal{E} \equiv \mathcal{F}$), if and only if $R_{\mathcal{F}}(\mathcal{E})$ is unsatisfiable. Let c be any clause of $\mathcal{E}$. Then, by Proposition 1, c is irredundant in $\mathcal{E}$ if and only if there exists an assignment μ, such that $(\mathcal{E} \setminus \{c\})[\mu] = 1$ and $c[\mu] = 0$. Equivalently, there is an extension μ' of μ such that $\text{CNF}(\neg\mathcal{F})[\mu'] = 1$ and $(\mathcal{E} \setminus \{c\})[\mu'] = 1$, i.e. $R_{\mathcal{F}}(\mathcal{E} \setminus \{c\})$ is satisfiable. □

Expressing the MES computation problem as a group-MUS computation problem enables the use of optimization techniques for (group-)MUS extraction in computing irredundant CNF formulas. Most importantly, the clause-set refinement becomes usable and effective again. We demonstrate this with the following example.

Example 2. Consider the CNF formula $\mathcal{F} = (x_1) \wedge (x_1 \vee y_i \vee y_j)$, with $1 \leq i < j \leq k$, and $k \geq 2$. All clauses with a literal in the y variables are redundant, for a total of $k(k-1)/2$ redundant clauses. The reduction in Proposition 2 produces the group-CNF formula $R_{\mathcal{F}}(\mathcal{F})$ with the don't care group $\mathcal{D} = \text{CNF}(\neg\mathcal{F})$, and a singleton group for each clause in $\mathcal{F}$, i.e. $\mathcal{G}_{11} = \{(x_1)\}$, $\mathcal{G}_{ij} = \{(x_1 \vee y_i \vee y_j)\}$, with $1 \leq i < j \leq k$. For this example, let us assume that the group-MUS of $R_{\mathcal{F}}(\mathcal{F})$ is computed using a deletion-based algorithm. Let $\mathcal{G}_{11}$ be the first group to be analyzed. This is done by removing the clause in $\mathcal{G}_{11}$ from the formula. The resulting formula has a model μ, with $\mu(x_1) = 0$ and $\mu(y_i) = \mu(y_j) = 1$, with $1 \leq i < j \leq k$. As a result, (x_1) is declared irredundant, and added back to the formula. Afterwards, pick one of the other groups, e.g. $\mathcal{G}_{12}$. If the corresponding clause is removed from the formula, the resulting formula is unsatisfiable, and so the clause is declared redundant. More importantly, a CDCL SAT solver will produce an unsatisfiable core $\mathcal{U} \subseteq \{x_1\} \cup \text{CNF}(\neg\mathcal{F})$. Hence, clause set refinement serves to eliminate *all* of the remaining groups of clauses, and so the MES is computed with *two* SAT solver calls. As noted earlier, MES algorithms based on adapting existing MUS algorithms are unable to implement clause set refinement, and so cannot drop $k(k-1)/2$ clauses after the second SAT solver call.

Observe that the *group-MUS approach* to the MES extraction problem, i.e. using the reduction in Proposition 2, is *independent* of the actual group-MUS extraction algorithm used. Hence, any existing group-MUS extraction algorithm can be used for the MES extraction problem. In practice, given the significant performance difference between deletion-based and insertion-based MUS extraction algorithms [28], our implementation is based on deletion-based group-MUS extraction and its most recent instantiation, i.e. the *hybrid* approach in [28].

3.4 Incremental Reduction of MES to Group-MUS

A major drawback of the group-MUS approach is that for large input formulas the resulting instances of SAT can be hard. This is due to the CNF encoding of $\neg\mathcal{F}$ that produces a large disjunction of auxiliary variables. A solution to this issue is based on an *incremental* reduction of MES extraction to group-MUS extraction. Let $\mathcal{T}$ be any subset of clauses of $\mathcal{F}$ — we refer to clauses of $\mathcal{T}$ as *target* clauses, and to the set $\mathcal{T}$ itself as a *chunk* of $\mathcal{F}$. The incremental reduction is based on the observation that in group-MUS approach the redundancy of any target clause $c \in \mathcal{T}$ can be established by analysing c with respect to $\mathrm{CNF}(\neg\mathcal{T})$ rather than $\mathrm{CNF}(\neg\mathcal{F})$. This observation is stated precisely below.

Proposition 3. *Let $\mathcal{T} \subseteq \mathcal{F}$ be a set of target clauses. For any $\mathcal{E} \subseteq \mathcal{T}$, let $R_{\mathcal{T}}(\mathcal{E})$ be the group-CNF formula $\mathcal{D} \cup \bigcup_{c\in\mathcal{E}} \mathcal{G}_c$, where $\mathcal{D} = \mathcal{F} \setminus \mathcal{T} \cup \mathrm{CNF}(\neg\mathcal{T})$ is the don't care group, and $\mathcal{G}_c = \{c\}$. Then, $\mathcal{E}$ is irredundant in $\mathcal{F}$ and $\mathcal{F} \setminus \mathcal{T} \cup \mathcal{E} \equiv \mathcal{F}$ if and only if $R_{\mathcal{T}}(\mathcal{E})$ is a group-MUS of $R_{\mathcal{T}}(\mathcal{T})$.*

Note that $R_{\mathcal{T}}(\mathcal{T})$ is unsatisfiable. Also, in the case when the chunk $\mathcal{T}$ is taken to be the whole formula $\mathcal{F}$, the group-CNF formula $R_{\mathcal{T}}(\mathcal{E})$ is exactly the formula $R_{\mathcal{F}}(\mathcal{E})$ from Proposition 2, and so the claim of Proposition 2 is a special case of the claim of Proposition 3. We omit the proof of Proposition 3 as it essentially repeats the steps of the proof of Proposition 2, using the definition of $R_{\mathcal{T}}(\mathcal{E})$ instead of $R_{\mathcal{F}}(\mathcal{E})$.

For the general case, consider a partition $\mathcal{F}_1, \dots, \mathcal{F}_k$ of $\mathcal{F}$ into chunks. Then, an MES of $\mathcal{F}$ can be computed by applying the group-MUS approach of Proposition 3 to each chunk. Proposition 3 is applied in order, with already computed irredundant subformulas replacing the original (redundant) subformulas. Explicitly, for the iteration j, $1 \le j \le k$, the input group-CNF formula in Proposition 3 is defined as follows:

$$\mathcal{D} \cup \bigcup_{c\in\mathcal{F}_j} \mathcal{G}_c, \text{ with } \mathcal{D} = \mathcal{E}_1 \cup \ldots \cup \mathcal{E}_{j-1} \cup \mathcal{F}_{j+1} \cup \ldots \mathcal{F}_k \cup \mathrm{CNF}(\neg\mathcal{F}_j) \qquad (1)$$

as a don't care group, where $\mathcal{E}_i$, $1 \le i < j$, is the computed irredundant set of clauses in $\mathcal{F}_i$, and, as before, $\mathcal{G}_c = \{c\}$.

Proposition 4. *Let $\mathcal{F}_1, \dots, \mathcal{F}_k$ be a partition of $\mathcal{F}$ into chunks. Let $\mathcal{E}_j$ be the set of clauses obtained from applying group-MUS extraction to formula in (1), and by considering each $\mathcal{F}_j$ as the set of target clauses. Let $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_k$. Then $\mathcal{E}$ is an MES of $\mathcal{F}$* [3].

Proof sketch. The proof uses induction on the sets $\mathcal{F}_j$ and Proposition 3 to prove the following inductive invariant: for $1 \le j \le k$, $(\bigcup_{r\le j} \mathcal{E}_r \cup \bigcup_{r>j} \mathcal{F}_r) \equiv \mathcal{F}$, and the formula $\bigcup_{r\le j} \mathcal{E}_r$ is irredundant in $\mathcal{F}$.
Base case: Take $\mathcal{T} = \mathcal{F}_1$, and apply Proposition 3 to formula $\mathcal{F}$.
Inductive step: For $1 < j \le k$, take $\mathcal{T} = \mathcal{F}_j$, and apply Proposition 3 to the formula $\bigcup_{r<j} \mathcal{E}_r \cup \bigcup_{r\ge j} \mathcal{F}_r$. Then, using the inductive hypothesis, establish the required invariant. □

While Proposition 4 can be used to compute an MES of an input formula $\mathcal{F}$ by iteratively calling a group-MUS extractor, it can also be integrated into a unified algorithm to enable certain optimizations (e.g. incremental SAT solving). Algorithm 4 shows the pseudo-code for deletion-based group-MUS approach to MES extraction using chunks. Different chunk sizes can be considered. Concrete examples include chunks of size 1 or a single chunk aggregating all clauses (i.e. the reduction to group-MUS defined in Proposition 2). For chunks of size

[3] The proposition is stated slightly informally as to avoid additional notation.

Algorithm 4: Deletion-based group-MUS extraction of MES with chunks

```
Input  : Formula $\mathcal{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_k\}$ with $k$ chunks
Output: MES $\mathcal{E}$
1  begin
2      for $j \leftarrow 1$ to $k$ do                          // Analyze each of the $k$ chunks
3          $\mathcal{D} = \mathcal{E}_1 \cup \ldots \cup \mathcal{E}_{j-1} \cup \mathcal{F}_{j+1} \cup \ldots \cup \mathcal{F}_k \cup \texttt{CNF}(\neg\mathcal{F}_j)$ // Don't care group
4          $\mathcal{W} = \mathcal{F}_j$                               // Target group of clauses $\mathcal{F}_j$
5          $\mathcal{E}_j \leftarrow \emptyset$                              // Irredundant clauses in chunk $j$
6          while $\mathcal{W} \neq \emptyset$ do
7              $c \leftarrow \texttt{SelectRemoveClause}(\mathcal{W})$
8              $(\text{st}, \nu, \mathcal{U}) = \texttt{SAT}(\mathcal{D} \cup \mathcal{E}_j \cup \mathcal{W})$
9              if st = true then                        // If SAT, $c$ is irredundant in $\mathcal{F}$
10                 $\mathcal{E}_j \leftarrow \mathcal{E}_j \cup \{c\}$
11                 $(\mathcal{W}, \mathcal{E}_j) \leftarrow \texttt{Rotate}(\mathcal{W}, \mathcal{E}_j, \nu)$        // Apply model rotation
12             else
13                 $\mathcal{W} \leftarrow \mathcal{U} \cap \mathcal{W}$                          // Clause-set refinement
14     $\mathcal{E} = \mathcal{E}_1 \cup \ldots \cup \mathcal{E}_k$
15     return $\mathcal{E}$                                           // $\mathcal{E}$ is an MES
16 end
```

great than 1, the group-MUS approach has the ability to prove several clauses redundant by using clause-set refinement. Generally, the chunk size in the group-MUS approach controls the *trade-off* between the potential power of clause-set refinement and the difficulty of the instances of SAT given to the SAT solver.

Nevertheless, an essential issue with Algorithm 4 is the selection of the size of the chunks. The current implementation of the algorithm uses chunks of fixed size, independently of the size of the formula. Alternative solutions include using chunk sizes dependent on the size of the formula, and also *adaptive* chunk sizes.

4 Certification of Correctness

In some applications, it is paramount to guarantee that the computed subformula is indeed irredundant and, possibly more importantly, that each computed irredundant subformula $\mathcal{E}$ is equivalent to the original CNF formula $\mathcal{F}$. Given a computed CNF formula $\mathcal{E}$, it is simple to validate whether it is irredundant. Essentially, one can run one of the algorithms outlined in earlier sections. The problem of validating whether $\mathcal{E} \equiv \mathcal{F}$ looks more challenging.

To check whether $\mathcal{E} \equiv \mathcal{F}$, it suffices to exhibit a truth assignment that is a model of one formula and not of the other. This condition corresponds to testing the satisfiability of the following CNF formula:

$$(\mathcal{E} \wedge \text{CNF}(\neg\mathcal{F})) \vee (\mathcal{F} \wedge \text{CNF}(\neg\mathcal{E})) \tag{2}$$

Clearly, the resulting instances of SAT are expected to be hard to solve, given the disjunction and the negation of formulas in (2). Nevertheless, it is also the case that $\mathcal{E} \subseteq \mathcal{F}$ and so, $\mathcal{F} \vDash \mathcal{E}$. Hence it is only necessary to check whether

$\mathcal{E} \vDash \mathcal{F}$. Since $\mathcal{F} = \mathcal{E} \wedge \mathcal{R}$, $\mathcal{E} \vDash \mathcal{F}$ can be represented as $\mathcal{E} \vDash \mathcal{E} \wedge \mathcal{R}$, or equivalently, $\mathcal{E} \wedge (\neg\mathcal{E} \vee \neg\mathcal{R}) \vDash \bot$, which simplifies to $\mathcal{E} \wedge \neg\mathcal{R} \vDash \bot$. This condition corresponds to testing the satisfiability of the following CNF formula:

$$\mathcal{E} \cup \mathrm{CNF}(\neg\mathcal{R}) \tag{3}$$

If $\mathcal{E} \cup \mathrm{CNF}(\neg\mathcal{R})$ has a model, then one can satisfy $\mathcal{E}$ while unsatisfying one or more clauses in $\mathcal{R}$. Hence, the two formulas would not be equivalent.

Although easier than (2), (3) can still result in hard instances of SAT (similarly to what happens with the reduction of MES to group-MUS extraction). A technique to reduce the complexity of the resulting instances of SAT is to partition $\mathcal{R}$ into chunks, and check each chunk separately for equivalence. The objective is to check whether $\mathcal{E} \vDash \mathcal{R}$. If this condition holds, then $\mathcal{E} \equiv \mathcal{F}$; otherwise $\mathcal{E} \not\equiv \mathcal{F}$. If $\mathcal{R}$ is partitioned into a number of subformulas (or chunks), we get $\mathcal{R} = \mathcal{R}_1 \wedge \ldots \wedge \mathcal{R}_k$, and so the condition becomes, $\mathcal{E} \vDash \mathcal{R}_1 \wedge \ldots \wedge \mathcal{R}_k$, that can also be represented as $\mathcal{E} \wedge (\neg\mathcal{R}_1 \vee \ldots \vee \neg\mathcal{R}_k) \vDash \bot$. This condition holds if and only if, $\forall_{1 \leq j \leq k}$, $\mathcal{E} \wedge \neg\mathcal{R}_j \vDash \bot$. Hence, the use of chunks allows splitting a potentially hard (and believed unsatisfiable) instance of SAT, into k (likely) easier (and also believed unsatisfiable) instances of SAT.

5 Experimental Results

The algorithms described in the previous sections were implemented within the MUS extraction framework of a state-of-the-art MUS extractor `MUSer2` [4]; the framework was configured to use SAT solver `picosat-935` [5] in the incremental mode. The experiments were performed on an HPC cluster, where each node is dual quad-core Intel Xeon E5450 3 GHz with 32 GB of memory. Each algorithm was run with a timeout of 1800 seconds and a memory limit of 4 GB per input instance. To evaluate the algorithms, we selected 300 problem instances from practical application domains of SAT used in past SAT competitions [5]. The instances were selected using the following criteria: the instance is solvable within 1 second by `picosat-935`, and the number of clauses in the instance is less than 100,000. These criteria were derived from the worst-case analysis of deletion-based MES algorithms (whereby the number of SAT calls is linear in the size of the input formula) and our previous experience with the effects various optimization techniques in the context of MUS extraction.

The cactus plot[6] in Fig. 2 provides an overview of the results of our experimental study. The legend in this, and the subsequent plots, is as follows: DEL represents the implementation of the deletion-based algorithm (Algorithm 1); INS (resp. DICH) is the implementation of insertion-based (resp. dichotomic) algorithms from Section 3.1; +MR indicates the addition of model rotation, while +IMR indicates the addition of the improved version of model rotation (cf. Section 3.2); GRP-MUS is the implementation of the reduction of MES to

[4] `http://logos.ucd.ie/wiki/doku.php?id=muser`.

[5] `http://www.satcompetition.org/`.

[6] Cactus plots show the sorted run times of algorithms over all instances and are commonly used in SAT competitions to compare performance of multiple solvers.

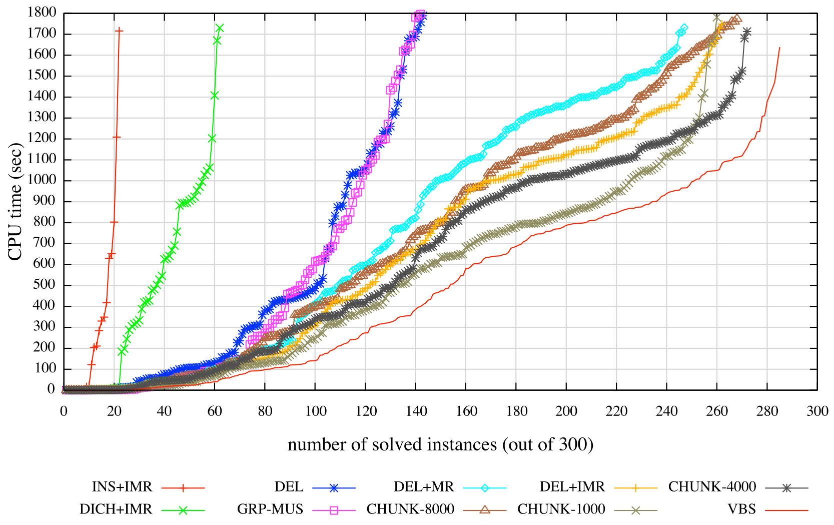

Fig. 2. Cactus plot with the run times of all algorithms.

group-MUS from Section 3.3; CHUNK-x is the implementation of the deletion-based chunked group-oriented MUS algorithm from Section 3.3 with chunk size x; VBS refers to a *virtual best solver* [7] — we elaborate on its composition shortly.

A number of conclusions can be drawn from the plot in Fig. 2. First, we note that the improvements to the plain deletion-based algorithm (DEL) suggested in Section 3.1, namely the addition of model rotation (DEL+MR) and the improved model rotation (DEL+IMR), have a very significant positive effect on the performance of the algorithm; also observe that the improved model rotation provides a notable boost over model rotation. Further, it is clear that the the insertion-based and the dichotomic algorithms, even with the addition of IMR, do not scale. As a side note, the gap between the performance of these and the deletion-based algorithm in the MES setting is *significantly* larger than that in the MUS setting (see for example [28]) — this is due to the addition of the negation of the working formula, which is unavoidable in the MES setting. We also observe the weak performance of the GRP-MUS approach — this is not surprising, since for large formulas, GRP-MUS can produce hard instances of SAT; this deficiency, in fact, was the motivation for the chunked approach. While DEL+IMR is among the best performing algorithms, on most of the instances it is significantly outperformed by the chunked group-oriented MUS algorithm with chunk size 1000. However, CHUNK-1000 loses to DEL+IMR on some of the hard instances — increasing the chunk size to 4000 pushes the performance of the algorithm ahead, however a further increase of chunk size to 8000 begins to affect the performance negatively. The VBS in Fig. 2 is constructed from DEL+IMR, GRP-MUS, CHUNK-1000 and CHUNK-4000. The former two are

[7] VBS represents a solver obtained from running a number of algorithms in parallel. It can be seen as a naive portfolio solver — we note that portfolio solvers are the current tour de force in SAT solving.

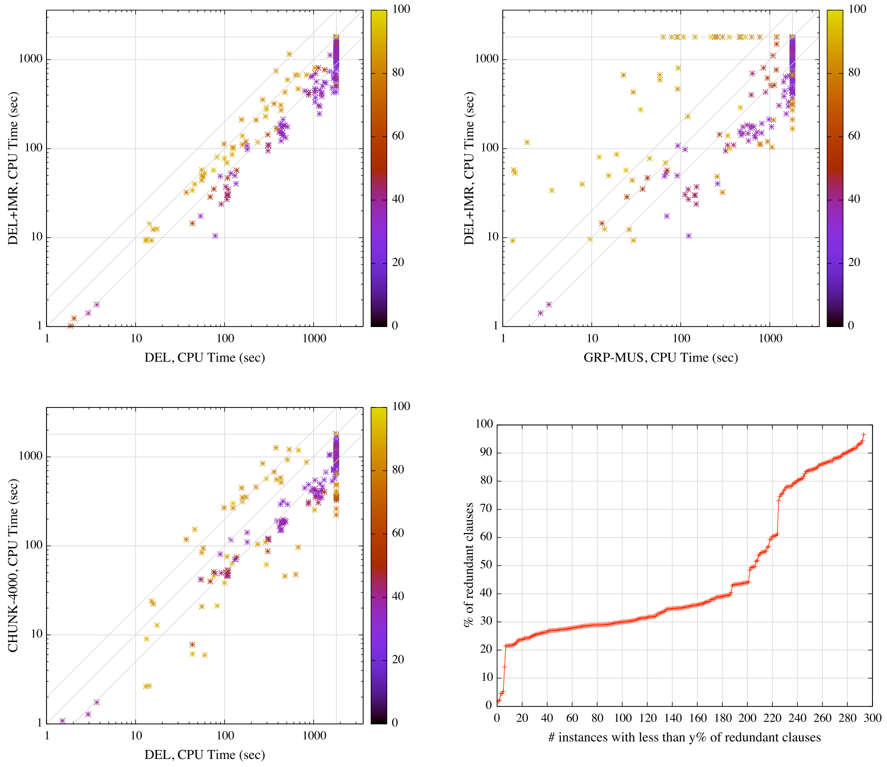

Fig. 3. Top and bottom-left: selected scatter plots (timeout 1800 sec.); color range represents % of redundant clauses in an instance. Bottom-right: cumulative histogram of the % of redundant clauses in the set of instances; note that the histogram is rotated 90° to be consistent with the cactus plot in Fig.2.

taken because they represent the extremes of the chunked approach — chunks of size 1 for DEL+IMR and a single chunk of the size of the input formula for GRP-MUS. The fact that the results for the VBS configuration (285 solved instances) are clearly superior to any of the individual algorithms indicates that the proposed algorithms are highly complementary, and are suitable for a multi-core/portfolio implementation. We emphasize that the plain deletion-based DEL is, to our knowledge, the current published state-of-the-art in MES computation, and so the algorithms proposed in this paper constitute a significant advancement, with the best algorithms solving more than twice the number of instances within the timeout.

The scatter plots in Fig. 3 provide additional insights into the performance of some of the algorithms. The color code indicates the amount of redundant clauses in the formulas: yellow (lighter) indicates close to 100% redundant clauses, whereas blue (darker) indicates less than 20% redundant clauses. The top-left plot (DEL vs. DEL+IMR) demonstrates the impact of improved model rotation (IMR) on the performance of the deletion-based approach. We note that while IMR allows to solve significantly more instances (263 vs 144 – see Fig. 2), the technique has little impact on many highly redundant instances (yellow). This behaviour is expected as IMR can only help to detect irredundant clauses, and it was the motivation for the development of the group-MUS approach. The top-right plot of Fig. 3 (GRP-MUS vs. DEL+IMR) demonstrates the effectiveness of the group-MUS approach on the highly redundant instances. The plot clearly shows that the direct and the group-MUS approaches are complementary – the former excels on mostly irredundant instances, while the latter is best on highly redundant ones. This observation provides additional, empirical, justification for the development of the chunked group-oriented MUS algorithm. The bottom-left plot (DEL vs. CHUNK-4000) confirms the effectiveness of the chunked approach — we observe significant performance improvements on instances with diverse degrees of redundancy.

The bottom-right plot in Fig. 3 presents the cumulative histogram of the degree of redundancy in the problem instances used in our experiments. Note that the instances were selected from the SAT competition benchmark sets *prior* to the experiments, and so the selection was not biased by the degree of redundancy. Nevertheless, approximately 2/3 of the instances have between 20% and 50% redundant clauses, the remaining instances have over 50% redundant clauses, and close to 5% of the instances have in excess of 90% redundant clauses. We conclude that problem instances from practical applications may exhibit very significant levels of redundancy.

6 Conclusions

This paper proposes novel algorithms for computing MESes. The main contributions of the paper can be summarized as follows: (i) Adapting existing MUS extraction algorithms for MES extraction; (ii) Development of model rotation for MES extraction, and analysis of why clause set refinement cannot be applied; (iii) Reduction of MES to group-MUS extraction, which enables both model rotation and clause set refinement to be applied; (iv) Use of chunks for incremental reduction of MES to group-MUS extraction, aiming at reducing the hardness of instances of SAT when the formula includes the negation of a CNF formula; and (v) Development of a solution for the independent certification of the correctness of computed MESes. The experimental results indicate that the algorithms proposed in this paper improve the direct approach [7] significantly, more than doubling the number of instances that can be solved, and with significant performance gains, which can exceed one order of magnitude.

The experimental evaluation carried out in the paper demonstrates that the developed algorithms are relevant for the practical applications of SAT. Indeed, our results show that many real-world SAT instances have significant percentages of redundant clauses. These findings clearly motivate revisiting the CNF encoding techniques used for generating these instances.

Although in this paper we do not address group-MES computation problem explicitly, the algorithms developed in the paper can be extended without difficulty to this setting. This could enable the identification and the removal of redundant constraints in other domains, such as CSP [12,10,9], SMT [35], and Ontologies [19]. This is the subject of future work.

References

1. A. Atserias, J. K. Fichte, and M. Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artif. Intell. Res.*, 40:353–373, 2011.
2. G. Ausiello, A. D'Atri, and D. Saccà. Minimal representation of directed hypergraphs. *SIAM J. Comput.*, 15(2):418–431, 1986.
3. R. R. Bakker, F. Dikker, F. Tempelman, and P. M. Wognum. Diagnosing and solving over-determined constraint satisfaction problems. In *IJCAI*, pages 276–281, 1993.
4. A. Belov and J. Marques-Silva. Accelerating MUS extraction with recursive model rotation. In *FMCAD*, pages 37–40, 2011.
5. A. Biere. Picosat essentials. *Journal on Satisfiability, Boolean Modeling and Computation*, 4:75–97, 2008.
6. A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.
7. Y. Boufkhad and O. Roussel. Redundancy in random SAT formulas. In *AAAI*, pages 273–278, 2000.
8. J. W. Chinneck and E. W. Dravnieks. Locating minimal infeasible constraint sets in linear programs. *INFORMS Journal on Computing*, 3(2):157–168, 1991.
9. A. Chmeiss, V. Krawczyk, and L. Sais. Redundancy in CSPs. In *ECAI*, pages 907–908, 2008.
10. C. W. Choi, J. H.-M. Lee, and P. J. Stuckey. Removing propagation redundant constraints in redundant modeling. *ACM Trans. Comput. Log.*, 8(4), 2007.
11. J. L. de Siqueira N. and J.-F. Puget. Explanation-based generalisation of failures. In *ECAI*, pages 339–344, 1988.
12. A. Dechter and R. Dechter. Removing redundancies in constraint networks. In *AAAI*, pages 105–109, 1987.
13. N. Dershowitz, Z. Hanna, and A. Nadel. A scalable algorithm for minimal unsatisfiable core extraction. In *SAT*, pages 36–41, 2006.
14. C. Desrosiers, P. Galinier, A. Hertz, and S. Paroz. Using heuristics to find minimal unsatisfiable subformulas in satisfiability problems. *J. Comb. Optim.*, 18(2):124–150, 2009.
15. O. Fourdrinoy, É. Grégoire, B. Mazure, and L. Sais. Eliminating redundant clauses in SAT instances. In *CPAIOR*, pages 71–83, 2007.
16. S. M. González and P. Meseguer. Boosting MUS extraction. In *SARA*, pages 285–299, 2007.
17. É. Grégoire, B. Mazure, and C. Piette. MUST: Provide a finer-grained explanation of unsatisfiability. In *CP*, pages 317–331, 2007.
18. É. Grégoire, B. Mazure, and C. Piette. On approaches to explaining infeasibility of sets of Boolean clauses. In *ICTAI*, pages 74–83, November 2008.
19. S. Grimm and J. Wissmann. Elimination of redundancy in ontologies. In *ESWC*, pages 260–274, 2011.
20. P. L. Hammer and A. Kogan. Optimal compression of propositional horn knowledge bases: Complexity and approximation. *Artif. Intell.*, 64(1):131–145, 1993.
21. F. Hemery, C. Lecoutre, L. Sais, and F. Boussemart. Extracting MUCs from constraint networks. In *ECAI*, pages 113–117, 2006.
22. M. Jarvisalo, M. Heule, and A. Biere. Inprocessing rules. In *IJCAR*, 2012. In Press. Available from `http://fmv.jku.at/papers/JarvisaloHeuleBiere-IJCAR12.pdf`.
23. U. Junker. QUICKXPLAIN: Preferred explanations and relaxations for over-constrained problems. In *AAAI*, pages 167–172, 2004.

24. O. Kullmann. Constraint satisfaction problems in clausal form II: Minimal unsatisfiability and conflict structure. *Fundam. Inform.*, 109(1):83–119, 2011.
25. P. Liberatore. Redundancy in logic I: CNF propositional formulae. *Artif. Intell.*, 163(2):203–232, 2005.
26. M. H. Liffiton and K. A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40(1):1–33, 2008.
27. J. Marques-Silva. Minimal unsatisfiability: Models, algorithms and applications. In *ISMVL*, pages 9–14, 2010.
28. J. Marques-Silva and I. Lynce. On improving MUS extraction algorithms. In *SAT*, pages 159–173, 2011.
29. A. Nadel. Boosting minimal unsatisfiable core extraction. In *FMCAD*, pages 121–128, October 2010.
30. M. Niepert, D. V. Gucht, and M. Gyssens. Logical and algorithmic properties of stable conditional independence. *Int. J. Approx. Reasoning*, 51(5):531–543, 2010.
31. C. Piette. Let the solver deal with redundancy. In *ICTAI*, pages 67–73, 2008.
32. C. Piette, Y. Hamadi, and L. Sais. Efficient combination of decision procedures for MUS computation. In *FroCos*, pages 335–349, September 2009.
33. K. Pipatsrisawat and A. Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artif. Intell.*, 175(2):512–525, 2011.
34. D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2(3):293–304, September 1986.
35. C. Scholl, S. Disch, F. Pigorsch, and S. Kupferschmid. Computing optimized representations for non-convex polyhedra by detection and removal of redundant linear constraints. In *TACAS*, pages 383–397, 2009.
36. S. T. To, T. C. Son, and E. Pontelli. On the use of prime implicates in conformant planning. In *AAAI*, 2010.
37. S. T. To, T. C. Son, and E. Pontelli. Conjunctive representations in contingent planning: Prime implicates versus minimal CNF formula. In *AAAI*, 2011.
38. H. van Maaren and S. Wieringa. Finding guaranteed MUSes fast. In *SAT*, pages 291–304, 2008.