ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-603

Efficient Autarkies

J. Marques-Silva [1, 2] and **A. Ignatiev** [1, 3] and **A. Morgado** [1] and **V. Manquinho** [1] and **I. Lynce** [1]

Abstract. Autarkies are partial truth assignments that satisfy all clauses having literals in the assigned variables. Autarkies provide important information in the analysis of unsatisfiable formulas. Indeed, clauses satisfied by autarkies cannot be included in minimal explanations or in minimal corrections of unsatisfiability. Computing the maximum autarky allows identifying all such clauses. In recent years, a number of alternative approaches have been proposed for computing a maximum autarky. This paper develops new models for representing autarkies, and proposes new algorithms for computing the maximum autarky. Experimental results, obtained on a large number of problem instances, show orders of magnitude performance improvements over existing approaches, and solving instances that could not otherwise be solved.

1 INTRODUCTION

The analysis of over-constrained sets of constraints finds a wide range of practical applications (e.g. [11, 26]). In the context of propositional formulas in conjunctive normal form (CNF), recent work addressed unsatisfiable formulas, either for finding minimal explanations of inconsistency (Minimal Unsatisfiable Subsets, MUSes, e.g. [3, 33]), or minimal relaxations for achieving satisfiability (Minimal Correction Subsets, MCSes, e.g. [8, 28, 25]).

Autarkies (or autark assignments) are partial truth assignments that satisfy all clauses having literals in the assigned variables [27]. These assigned variables are the autark variables. Autarkies were first studied in an approach for improving the worst-case complexity of solving propositional satisfiability (SAT) [27]. Nevertheless, later work showed that autarkies play a key role in the analysis of unsatisfiable formulas [15, 16, 18, 13, 17]. Indeed, autark variables denote variables that cannot be included in any MUS (and so, by hitting set duality [30, 5, 2, 22], cannot be included in any MCS). As a result, the identification of autark variables finds application in the analysis of unsatisfiable subformulas. For example, autarkies can be used for computing the lean kernel of a formula, i.e. the clauses that can be used in resolution refutations. In recent years, the identification of autark variables has been applied in MUS enumeration [23] and MUS extraction [3, 6, 33]. Autark assignments have also been used in developing fixed-parameter tractable MUS finding solutions [32] and in the study of homomorphisms of CNF formulas [31]. Moreover, autarkies have also been studied in a number of SAT solving approaches [9, 10], in addition to the original work [27].

Besides known uses of autarkies, other possible uses of autarkies can be envisioned. For example, since autark variables are not included in minimal explanations nor in minimal corrections of unsatisfiability, the identification of autark variables can be used for investigating possible inefficiencies when encoding problems into CNF. However, a key obstacle to the widespread use of approaches for finding autark variables is that existing approaches are usually inefficient for large formulas.

This paper develops a number of optimizations for computing autark variables. These include three new models for representing autark assignments, and several new algorithms for computing the maximum autark assignment. Compared to earlier work, the new algorithms are shown to allow efficiently finding the maximum autark assignment for large problem instances, including those used for validating MUS extraction algorithms. From a practical perspective the main goal of this work is to find autarkies efficiently (e.g. within a few seconds) for relevant instances of SAT, e.g. those related with computing MUSes or MCSes.

The paper is organized as follows. Section 2 introduces the notation and definitions used in the remainder of the paper. Section 3 develops the models to represent autark assignments. Section 4 develops algorithms for computing maximum autark assignments, highlighting the relationship with maximum satisfiability (MaxSAT) and minimal correction subsets (MCSes). Section 5 conducts a comprehensive experimental evaluation based on problem instances from the MUS track of the 2011 SAT competition [4], which represent problem instances commonly used for evaluating MUS extraction algorithms. Finally, Section 6 concludes the paper.

2 PRELIMINARIES

Standard SAT definitions are assumed (e.g. [4]). A CNF formula $\mathcal{F}$, with $|\mathcal{F}| = m$, is a conjunction of clauses, interpreted as a set of clauses. A clause is a disjunction of literals, interpreted as a set of literals. A literal is a variable or its negation. Clauses in $\mathcal{F}$ are denoted by c_i, $1 \leq i \leq m$. The set of variables of $\mathcal{F}$ is denoted by $\mathsf{var}(\mathcal{F})$, also represented as X, with $|X| = n$. The total number of literals in the formula is L. The variables occurring in a clause c_i are denoted by $\mathsf{var}(c_i) \subseteq X$. The variables occurring as a positive literal in clause c_i are denoted by $P(c_i) \subseteq \mathsf{var}(\mathcal{F})$. The variables occurring as a negative literal in clause c_i are denoted by $N(c_i) \subseteq \mathsf{var}(\mathcal{F})$. It is assumed clauses are non-tautologous, i.e. $P(c_i) \cap N(c_i) = \emptyset$. A truth assignment is a partial mapping from a set of variables to $\{0, 1\}$. The standard definition of interpretation is assumed.

2.1 Unsatisfiable Formulas & Autarkies

The paper assumes standard Maximum Satisfiability (MaxSAT) definitions [21]. Moreover, related definitions of Maximal Satisfiable Subsets (MSSes), Minimal Correction Subsets (MCSes) and Minimal Unsatisfiable Subsets (MUSes) are assumed [5, 2, 22], including the well-known relationship with minimal diagnoses [30]. A

[1] Instituto Superior Técnico/INESC-ID, Universidade de Lisboa, Portugal
[2] CSI/CASL, University College Dublin, Ireland
[3] ISDCT SB RAS, Irkutsk, Russia

[4] http://www.satcompetition.org/2011/.

MaxAutark_ProofBased ($\mathcal{F}$)
Input: $\mathcal{F}$: Formula
Output: A: Autark variables

```
repeat
    (st, π) ← SAT(F)
    if not st then
        Z ← ComputeVars(π)
        F ← RemoveClauses(F, Z)
until st
return var(F)
```

Algorithm 1: Computing the maximum autark assignment

number of algorithms for computing MCSes/MSSes have recently been proposed [8, 28, 25], that build and improve upon earlier work [12, 2, 29].

A set of variables A' is *autark* if there exists a (partial) truth assignment to the variables in A' such that any clause containing literals in the variables of A' is satisfied [27]. This partial truth assignment is referred to as an *autarky* (or an *autark (truth) assignment*). The clauses satisfied by an autarky are referred to as an *autark clause-set* [13]. The *largest autark clause-set* is the largest set of clauses satisfied by any autarky. The set of autark variables defining the largest autark clause-set is referred to as the *largest autark (variable) set*, and is denoted by A. This paper addresses the computation of the largest autark set, from which the largest autark clause-set is readily obtained. The truth assignment identifying the largest autark set is referred to as the *maximum autarky* or *maximum autark assignment*. Autark variables have a number of relevant properties. First, autark variables cannot be included in any MUS of a CNF formula $\mathcal{F}$ (e.g. [15, 13, 17]). Similarly, by hitting set duality (e.g. [30, 5, 2, 22]), autark variables cannot be included in any MCS of $\mathcal{F}$. Furthermore, it is well-known that the maximum autarky is *unique* (e.g. [15, 13]). Kleine Büning and Kullmann [13] provide a recent account of approaches for computing autarkies.

2.2 Previous Work

A number of algorithms have been proposed for computing the largest autark set (and associated largest autark clause-set). To our best knowledge, the earliest approach is based on the iterative identification of the set of clauses used in a resolution proof, their removal and also removal of resulting empty clauses [16]. Algorithm 1 shows the main steps of this algorithm. While the formula is unsatisfiable, a resolution proof (π) is used to identify which variables are used in the proof. These variables are removed from any clause, and any empty clause is also removed. This process is repeated until the formula becomes satisfiable, at which step the algorithm terminates and reports the remaining variables as autark.

More recently, approaches based on optimizing a cost function were proposed [23, 13]. The first of these approaches [23] is based on constructing a modified formula $\mathcal{F}_{01}^{\text{Aut}}$, representing valid autark assignments to $\mathcal{F}$, and then solving an optimization problem subject to $\mathcal{F}_{01}^{\text{Aut}}$. The set of clauses associated with this approach for computing the maximum autark assignment will be referred to as model Γ_{01}, and is described next.

The motivation for model Γ_{01} (and other related models) is to select variables to be included in the autark set, such that the condition for autark assignment is satisfied. Given the set of variables $X \triangleq \mathsf{var}(\mathcal{F})$, the following sets of variables are used: (i) the original set of variables X; (ii) the set of selected (or active) variables X^+; (iii) the set of variables associated with positive literals X^1; (iv) the set of variables associated with negative literals X^0; and (v) a set of variables Y associated with the clauses. The semantics of the new sets of variables is as follows. $y_i \in Y$ is 1 if and only if clause c_i is to be satisfied by the autark assignment. $x_j^+ \in X^+$ is 1 if and only if variable x_j is selected to be included in the autark set, i.e. variable x_j is *active*. $x_j^1 \in X^1$ is 1 if and only if x_j is in the autark set (i.e. active) and the value of x_j is 1. $x_j^0 \in X^0$ is 1 if and only if x_j is in the autark set (i.e. active) and the value of x_j is 0. The set of clauses $\mathcal{F}_{01}^{\text{Aut}}$ of model Γ_{01} is defined as follows:

1. For each clause $c_i \in \mathcal{F}$:
$$y_i \rightarrow \bigvee_{x_k \in P(c_i)} x_k^1 \vee \bigvee_{x_k \in N(c_i)} x_k^0 \tag{1}$$
2. For each clause $c_i \in \mathcal{F}$, and for each variable $x_j \in \mathsf{var}(c_i)$:
$$x_j^+ \rightarrow y_i \tag{2}$$
3. For each variable $x_j \in \mathsf{var}(\mathcal{F})$, add the CNF-encoding of the following constraints:
$$\begin{array}{l} x_j^0 \leftrightarrow x_j^+ \wedge \neg x_j \\ x_j^1 \leftrightarrow x_j^+ \wedge x_j \end{array} \tag{3}$$

Recall that the activation variables (x_j^+) indicate whether a variable is included in the autark set of variables. If a variable is included in the autark set of variables, then all the clauses with a literal in x_j must also be active (and satisfied by a literal of an active variable). This constraint is captured by (2). The variables representing the positive and negative literals can be different from 0 only if the associated activation variable is 1. In this case, these variables take the value of the corresponding literals. This constraint is captured by (3). Moreover, if a clause is active, one of its literals must be assigned value 1, i.e. one of the active variables must have a literal assigned value 1 that satisfies the clause. This constraint is captured by (1). Thus, given any satisfying truth assignment to the variables of the above formula, the set of active variables (i.e. x_j variables such that $x_j^+ = 1$) denote autark variables. Finally, the cost function is captured with (unit) soft clauses, one for each of the Y variables (i.e. the target set).

An alternative model (referred to as model Γ_{02} in this paper) has been proposed more recently [13], even though no experimental results are reported. This model uses three variables for each original variable (which can be viewed as X^0, X^1 and X^+), but also proposes the use of the Y variables to enable the computation of the maximum autarky. The relationship between the three new variables for each original variable x_j is captured by the following constraints:

$$\begin{array}{l} x_j^0 + x_j^1 + \neg x_j^+ \leq 1 \\ x_j^0 + x_j^1 + \neg x_j^+ \geq 1 \end{array} \tag{4}$$

These constraints can be encoded to CNF using 7 clauses. Additional constraints used in model Γ_{02} are described in the next section (see (6), (7)) and are summarized in Table 1.

3 MODELING AUTARKIES

This section develops three alternative models for computing autark sets, which are more compact than the models described in Section 2.2, and which allow a number of different algorithms to be used as described in Section 4. All proposed models can be viewed as simplified versions of model Γ_{01} [23] (which is summarized in Section 2.2). The models start from a CNF formula $\mathcal{F}$ and create a formula $\mathcal{F}^{\text{Aut}}$. The sets of variables used as well as the resulting CNF

Table 1: Models for computing autarkies

Model N	Sets of variables	Target set	Set of clauses $\mathcal{F}_N^{\text{Aut}}$	# Variables	# Clauses
Γ_{01}	X, X^0, X^1, X^+, Y	Y	(1), (2), (3)	$4n + m$	$m + L + 6n$
Γ_{02}	X^0, X^1, X^+, Y	Y	(6), (7), (2), (4)	$3n + m$	$2L + 7n$
Γ_1	X, X^0, X^1, X^+	X^+	(5), (3)	$4n$	$L + 6n$
Γ_2	X^0, X^1, X^+	X^+	(6), (7), (8), (9)	$3n$	$L + 4n$
Γ_3	X^0, X^1	$X^0 \cup X^1$	(6), (7), (8)	$2n$	$L + n$

formula differ in each model.

The first new model (Γ_1) is a simplified version of model Γ_{01} in that the Y variables are eliminated. The clauses $\mathcal{F}_1^{\text{Aut}}$ of model Γ_1 are defined as follows:

1. For each clause $c_i \in \mathcal{F}$:
   (a) For each variable $x_j \in \mathsf{var}(c_i)$, add clauses:

$$x_j^+ \rightarrow \bigvee_{x_k \in P(c_i)} x_k^1 \vee \bigvee_{x_k \in N(c_i)} x_k^0 \tag{5}$$

2. For each variable $x_j \in \mathsf{var}(\mathcal{F})$, add the CNF-encoding of the constraints in (3).

Observe that model Γ_1 can be obtained from Γ_{01} by resolving away the Y variables. Nevertheless, this simplification also results in further insights, that enables developing more compact models.

The second model proposed in this paper, referred to as model Γ_2, reduces the set of variables used to X^0, X^1 and X^+. This can be achieved by noticing the semantics of the variables X^0 and X^1. $x_j^0 \in X^0$ is 1 if and only if x_j is in the autark set and the value used is 0. Similarly, $x_j^1 \in X^1$ is 1 if and only if x_j is in the autark set and the value used is 1. Thus, since for each variable x_j, the variables x_j^0 and x_j^1 already encode the value of the original variable x_j, then the original variables can be discarded. Moreover, X^+ is defined as above: $x_j \in X^+$ is 1 if and only if x_j is selected to be included in the autark set. Consequently, the set of clauses $\mathcal{F}_2^{\text{Aut}}$ of model Γ_2 is defined as follows:

1. For each clause $c_i \in \mathcal{F}$:
   (a) For each variable $x_j \in P(c_i)$, add clauses:

$$x_j^0 \rightarrow \bigvee_{x_k \in P(c_i)\setminus\{x_j\}} x_k^1 \vee \bigvee_{x_k \in N(c_i)} x_k^0 \tag{6}$$

   (b) For each variable $x_j \in N(c_i)$, add clauses:

$$x_j^1 \rightarrow \bigvee_{x_k \in P(c_i)} x_k^1 \vee \bigvee_{x_k \in N(c_i)\setminus\{x_j\}} x_k^0 \tag{7}$$

2. For each variable $x_j \in \mathsf{var}(\mathcal{F})$, add clauses (representing an AtMost1 constraint):

$$(\neg x_j^1 \vee \neg x_j^0) \tag{8}$$

3. For each variable $x_j \in \mathsf{var}(\mathcal{F})$, add clauses:

$$x_j^+ \leftrightarrow (x_j^1 \vee x_j^0) \tag{9}$$

Although Γ_2 eliminates the set X of original variables, there are also important differences to the clauses used. The information encoded in the variables in X^0 and X^1 is used to define the X^+ in terms of these variables. This constraint is captured by (9). A key observation is that at most one of x_j^0 and x_j^1 can be assigned value 1. This constraint is captured by (8). Finally, if a variable is active, any clause where it assigns a literal to 0 must be satisfied by some other literal. This constraint is captured by (6) and (7).

Regarding model Γ_2 a few (optional) modifications can be considered. First, given the optimization algorithms described in the next section, the equivalence (9) can be replaced with two clauses:

$$(\neg x_j^1 \vee x_j^+) \wedge (\neg x_j^0 \vee x_j^+) \tag{10}$$

Moreover, observe that equations (8) and (9) could be merged into:

$$x_j^+ \leftrightarrow \neg(x_j^1 \leftrightarrow x_j^0) \tag{11}$$

This modification introduces a (possibly large) number of XOR constraints, which are usually problematic for clause learning SAT solvers. As a result, model Γ_2 uses equations (8) and (9) instead. Nevertheless, it would be possible to consider the alternative formulation, for example by using a SAT solver with dedicated techniques for reasoning with XOR constraints [20, 19].

The third and final model, referred to as Γ_3, is a simplification of model Γ_2. Observe that variables in X^+ are only relevant for identifying which variables are active, and so included in the set of autark variables. However, as long as either x_j^0 or x_j^1 is assigned value 1 (and, due to constraint (8), these variables *cannot* both be assigned value 1), then we know that variable x_j is active and so included in the set of autark variables. Given the above, model Γ_3 uses solely the sets of variables X^0 and X^1. As a result, the clauses given by (9) are not included in the set of clauses $\mathcal{F}_3^{\text{Aut}}$ of model Γ_3. Everything else mimics model Γ_2.

Example 1. *Consider the unsatisfiable formula:*

$$\mathcal{F} = \{(x_1 \vee x_2), (\neg x_1 \vee x_2), (\neg x_2), (\neg x_1 \vee x_3)\} \tag{12}$$

representing clauses c_1, c_2, c_3, c_4, respectively, and let $X = \{x_1, x_2, x_3\}$. The new sets of variables are: $X^0 = \{x_1^0, x_2^0, x_3^0\}$ and $X^1 = \{x_1^1, x_2^1, x_3^1\}$. From (8), the set of clauses $\mathcal{F}_3^{\text{Aut}}$ created given model Γ_3 is as follows:

$$\{(\neg x_1^0 \vee \neg x_1^1), (\neg x_2^0 \vee \neg x_2^1), (\neg x_3^0 \vee \neg x_3^1)\} \tag{13}$$

Moreover, from (6) and (7), the following clauses are created:

$$\begin{aligned}\{&(\neg x_1^0 \vee x_2^1), (\neg x_2^0 \vee x_1^1), (\neg x_1^1 \vee x_2^1), (\neg x_2^0 \vee x_1^0),\\ &(\neg x_2^1), (\neg x_1^1 \vee x_3^1), (\neg x_3^0 \vee x_1^0)\}\end{aligned} \tag{14}$$

Table 1 summarizes the proposed models and compares them with the reference models, Γ_{01} [23] and Γ_{02} [13], both of which are summarized in Section 2.2. The first column shows the model number. The second column lists the sets of variables used. The third column indicates the *target set*, i.e. the set of variables which is to be used for computing the maximum autark assignment (e.g. by optimizing with respect to the sum of variables in the target set). The fourth column lists the sets of constraints associated with each model. Finally, the fifth and sixth columns show the total number of variables and clauses, respectively. The total numbers of variables and clauses are obtained directly from the sets of clauses associated with each model. Observe that, for model Γ_2 it would be possible to reduce the total number of clauses to $L + 3n$, by using (10) instead of (9). Moreover, it should be noted that, when compared to models Γ_{01} and Γ_{02}, model Γ_3 reduces significantly the number of variables (to less than half) but also the number of clauses (under the assumption L is not much larger than m).

4 COMPUTING AUTARKIES

This section describes algorithms for computing the maximum autark set. These algorithms exploit the models proposed in the previous section, as well as earlier models [23, 13]. Another alternative approach is Algorithm 1, described in Section 2.2. For the models described in Section 3 and in Section 2.2 (see Table 1), a simple algorithm consists of iteratively checking whether each variable in the target set can take value 1. For models Γ_1, Γ_2 and Γ_3 each such variable indicates an original variable that is part of the autark set. For model Γ_{01} [23], each variable denotes a clause and, if the clause is active, then some of its variables are also active and are added to the set of autark variables. Observe that for model Γ_{01}, although the proposed target set is Y [23], set X^+ could also be used as the target set. Clearly, this algorithm requires $\mathcal{O}(|T|)$ calls to a SAT solver, where T is the target set. The purpose of this section is to develop alternative approaches that require fewer calls to a SAT solver.

4.1 Using Maximum Satisfiability

The computation of the largest autark (variable or clause) set can be modeled with partial maximum satisfiability. For each variable t in the target set T create a soft clause (t), denoting a preference to include the element (clause or variable) associated with variable t in the autark set. (Observe that the target set depends on which model is considered, as shown in Table 1.) The hard clauses are given by the clauses associated with each model. Thus, one can compute the largest autark set by solving partial MaxSAT. Moreover, observe that MaxSAT was used in earlier approaches [23], namely with model Γ_{01}. However, whereas earlier work used a linear search algorithm [23], we can in fact use *any* MaxSAT algorithm. Therefore, the number of calls to a SAT solver can range between $\mathcal{O}(\log|T|)$ and $\mathcal{O}(|T|)$ (or possibly $\mathcal{O}(|A|)$), where T is the target set and A is the set of autark variables. The main drawback of using partial MaxSAT is that most MaxSAT solvers must encode cardinality constraints [4] to CNF, and this can become an issue if the number of autark variables is not negligible, resulting in large right-hand sides for cardinality constraints. However, as shown in the next section, the use of MaxSAT is actually unnecessary.

4.2 Using Minimal Correction Subsets

An alternative to MaxSAT is to compute an MCS. The key insight is that the MaxSAT formula associated with computing the largest autark set has a unique MSS/MCS. Thus, it suffices to compute an MSS/MCS instead of solving MaxSAT.

Proposition 1. *For MaxSAT instances representing the computation of the largest autark set, there is a unique MSS/MCS.*

Proof. (Sketch) A well-known result in the theory of autark assignments is that the largest autark set A is unique and any autark set is contained in A [15, 13]. Any satisfying assignment computed for any of the models (i.e. $\Gamma_{01}, \ldots, \Gamma_3$) represents an autark assignment, and so the associated set of variables is contained in A. An MSS for the MaxSAT problems associated with any of the models (i.e. $\Gamma_{01}, \ldots, \Gamma_3$) *must* represent an autark set of variables (since the hard clauses are satisfied), and so it is contained in A. Since an MSS is subset maximal, then it must be A. □

MAXAUTARK_MCSBASED ($\mathcal{F}^{\text{Aut}}, T$)
Input: $\mathcal{F}^{\text{Aut}}$: Model; T: Target set
Output: A: Autark variables
$A \leftarrow \emptyset$
repeat
  $D \leftarrow (\vee_{t \in T}\, t)$
  $(st, \mu) \leftarrow$ SAT$(\mathcal{F}^{\text{Aut}} \cup \{D\})$
  if *st* **then**
    $A \leftarrow A \cup$ RemoveSatElems(μ, T)
    $T \leftarrow T \setminus A$
until not *st*
return A

Algorithm 2: Computing the largest autark set

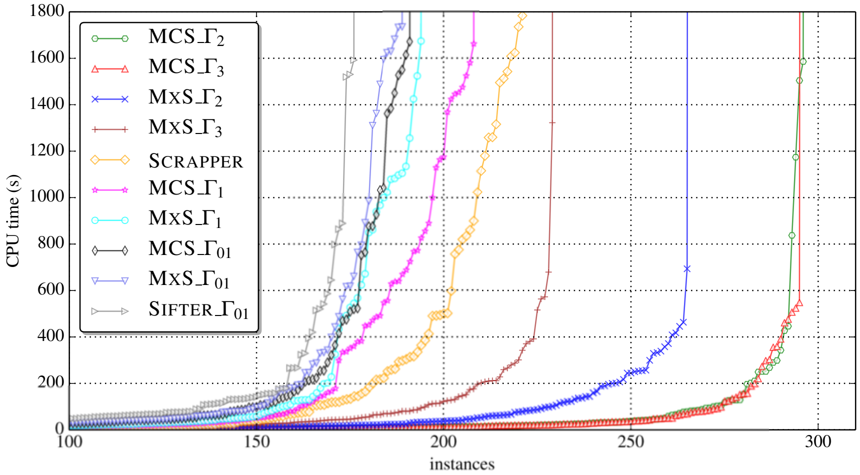
As a result, instead of using MaxSAT, one can simply compute an MCS for *any* of the models described in the previous sections. Several MCS extraction algorithms have been proposed in recent years, e.g. [2, 29, 8, 28, 25]. Thus, any of these algorithms can be used for computing the largest autark set. Similar to the MaxSAT case, the number of calls to a SAT solver can range between $\mathcal{O}(\log|T|)$ and $\mathcal{O}(|T|)$ (or even $\mathcal{O}(|A|)$), where T is the target set and A is the set of autark variables. However, since in practice the percentage of autark variables is usually small (and so $|A| \ll |T|$), our goal is to develop solutions that require a number of calls to the SAT solver that grows with $|A|$.

To guarantee that the number of calls grows with $|A|$, our approach is similar to the recently proposed CLD algorithm for MCS extraction [25]. Algorithm 2 shows the proposed algorithm for computing a largest autark set. At each iteration of the algorithm, clause D contains all elements from the target set not yet included in A. A SAT solver checks the satisfiability of $\mathcal{F}_N^{\text{Aut}}$ (for some model N) union clause D, and returns true if the formula is satisfiable or false if the formula is unsatisfiable, i.e. *st* set to true or false, respectively. The SAT solver also returns a truth assignment μ in case the formula is satisfiable. In this case, the truth assignment is used to update both the autark set of variables A and the target set T. While elements from the target set T can be satisfied, these are removed from set T and added to set A. The process terminates when no more elements from the target set can be satisfied, i.e. all autark variables are already included in set A. In the worst-case, the number of times the loop is executed, and so the number of calls to the SAT solver, is $|A| + 1$.

In practice, it is preferable to encode the problem of computing the largest autark set as partial MaxSAT, and use an off-the-shelf state-of-the-art MaxSAT solver or MCS extractor, thus allowing the computation of the largest autark set to exploit techniques developed for either MaxSAT solving or MCS extraction. For example, the MCS extractor MCSls [25] configured with algorithm CLD will implement Algorithm 2. Moreover, MCSls implements a few additional techniques aiming to reduce the number of calls to the SAT solver [25]. The next section evaluates a number of alternative approaches for computing the largest autark set, that build on the ideas described in this and the previous section.

5 RESULTS

The models proposed in Section 3 as well as model Γ_{01} have been implemented and evaluated both with MaxSAT solvers and MCS



Configuration	# Solved
MCS_Γ_2	297
MCS_Γ_3	296
MxS_Γ_2	266
MxS_Γ_3	230
SCRAPER	222
MCS_Γ_1	209
MxS_Γ_1	195
MCS_Γ_{01}	192
MxS_Γ_{01}	190
SIFTER_Γ_{01}	177

Figure 1: Cactus plot and statistics for the different configurations

extractors. Moreover, for comparison purposes, updated versions of SIFTER [23] as well as SCRAPER [16, 18, 23] (see Algorithm 1 in Section 2.2) have been implemented. (Observe that the original versions of these tools are not publicly available). These updated versions use recent SAT solvers as well as new SAT solving techniques. The problem instances considered were taken from the MUS track of the 2011 SAT competition. This represents a suite of 300 problem instances, obtained from several practical applications, with instances ranging from a few thousand variables and clauses to instances with millions of variables and clauses. Thus the problem instances considered are significantly more challenging than the ones considered in earlier work [23]. Moreover, the number of autark variables ranges from 0 to a few thousand for some of the larger instances. All experiments were run on an HPC cluster, each node having two processors E5-2620 @2GHz, with each processor having 6 cores, and with a total of 128 GByte of physical memory. Each process was limited to 4GByte of RAM and to a time limit of 1800 seconds.

The following solvers were used in the experiments. The MaxSAT solver used is QMaxSAT[5] 0.21 [14], since it is the best (non-portfolio) partial MaxSAT solver in the 2013 MaxSAT Evaluation[6]. QMaxSAT 0.21 uses Glucose 2.0 [1] as the backend SAT solver. The MCS extractor used is MCSls[7] [25]. MCSls uses MiniSat [7], version 2.2 (Nov 2012), as the backend SAT solver. For the computation of autarkies by iterated proof extraction, i.e. a re-implementation of SCRAPER [16, 18, 23], the SAT solver used is MiniSat [7], version 2.2 (Sep 2013). Observe that MiniSat is used in incremental mode, the set of assumptions in the final learned clause represents the unsatisfiable core (obtained with resolution operations) and so no explicit proof trace is recorded. This is in general significantly more efficient than the original proof-tracing algorithm [18].

The experiments consider both QMaxSAT (MxS) and MCSls (MCS) with models Γ_{01}, Γ_1, Γ_2 and Γ_3. For example, QMaxSAT running Γ_{01} can be viewed as a state-of-the-art implementation of SIFTER [23], even though QMaxSAT also implements a number of additional MaxSAT solving features [14]. As a result, we also implemented our own version of SIFTER, aiming to mimic the original implementation [23]. SIFTER is based on MiniSat 2.2, it is run in non-incremental mode, it does not exploit upper bound information, but uses the same cardinality encoding as QMaxSAT.

Figure 1 shows a cactus plot comparing the 10 configurations described above. The table next to the plot shows the number of solved instances. The MCS-based approach using either Γ_2 or Γ_3 solve the most instances. The performance improvement over the MaxSAT approach, using model Γ_2 (and with larger differences for Γ_3), is illustrated by the rightmost scatter plot in Figure 2. These results confirm the importance of using MCS extraction instead of MaxSAT solving. Notice that the observed outliers can be explained by instances with fewer autarkies and a more efficient SAT solver (Glucose 2.0) being used by QMaxSAT. The two other scatter plots (and the cactus plot) also confirm the performance improvement over what can be considered existing solutions for computing maximum autarkies, namely SIFTER, SCRAPER and QMaxSAT using model Γ_{01}. Indeed, the configuration that solves more instances (i.e. MCS with model Γ_2), is able to solve approximately 34% more instances (i.e. from 222 to 297) than the best among existing solutions (i.e. our implementation of SCRAPER). There is a small performance gap between SIFTER and Qmaxsat with Γ_{01}, due the different SAT solvers and the non-incremental interface in SIFTER. Moreover, the performance gap between MCSls with model Γ_{01} and with model Γ_3 (or Γ_2) demonstrates the improvements achieved by using the new models. Finally, although MCSls with Γ_2 solves one more instance than with Γ_3, it is also the case that Γ_3 in general performs better than Γ_2. By considering the 296 problem instances solved by both models, Γ_3 solves these instances in 9640s, whereas Γ_2 solves the same instances in 11063s, i.e. an overall reduction of 12.8%. For MaxSAT-based approaches, the larger number of soft clauses required by Γ_3 can be an issue, as illustrated by the results of QMaxSAT with Γ_2 and Γ_3.

Of the 300 instances, MCSls with model Γ_3 (Γ_2) solves 219 (226) instances in less than 20 seconds, and 25 (27) instances require more than 100 seconds. In contrast, QMaxSAT with model Γ_{01} solves 76 instances in less than 20 seconds, and for 148 instances it requires more than 100 seconds. These differences highlight the performance gains introduced by both the new models (Γ_2 and Γ_3) and the use of MCS extraction.

6 CONCLUSIONS

The identification of maximum autark assignments represents another tool for the analysis of unsatisfiable formulas. This paper develops three new optimization models for computing the largest autark set, all of which are shown to be more compact than existing models [23, 13]. Moreover, this paper shows that, instead of computing the maximum autark assignment with MaxSAT (or iterative identi-

[5] https://sites.google.com/site/qmaxsat/.
[6] http://maxsat.ia.udl.cat/.
[7] http://logos.ucd.ie/wiki/doku.php?id=mcsls.

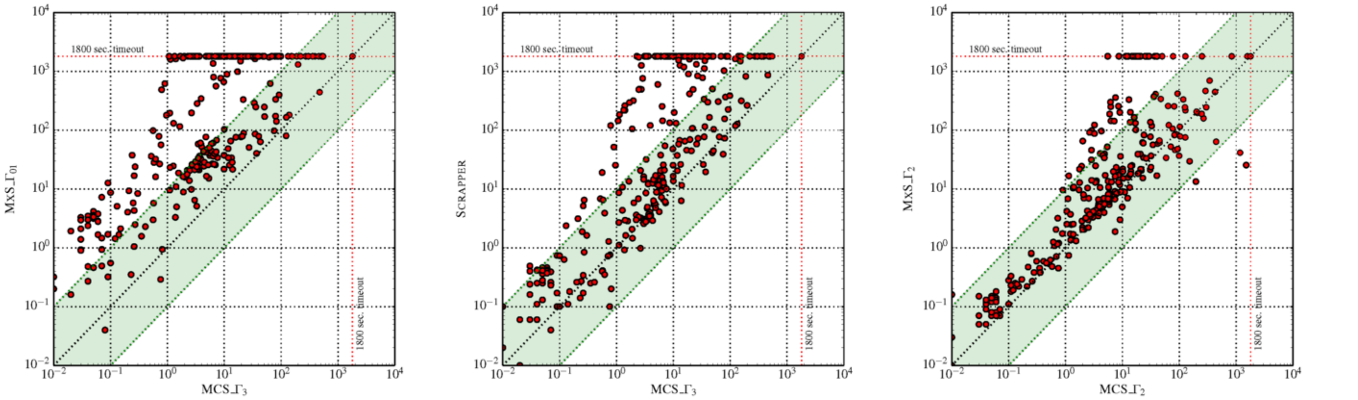


Figure 2: Scatter plots comparing selected configurations: MxS_Γ_{01} vs. MCS_Γ_3, SCRAPER vs. MCS_Γ_3, and MxS_Γ_2 vs. MCS_Γ_2

fication of resolution proofs), it suffices to compute an MCS of a partial MaxSAT formula. The consequences of this insight are significant since, in practice, the computation of an MCS is expected to be simpler than solving MaxSAT. Experimental results demonstrate that the new models and the use of MCS extraction provide consistent performance improvements, often with gains exceeding an order of magnitude. Moreover, existing approaches often do not scale for large problem instances, whereas the new algorithms are shown to scale significantly better.

Despite the observed performance improvements, it is also the case that a few instances either cannot be solved or require long run times. For these more challenging problem instances, additional techniques should be investigated. These could include algorithm configuration and algorithm portfolios [34, 24], for selecting a likely effective algorithm, as well as formula preprocessing techniques. In addition, the connection with MCSes allows tapping on any additional improvement made to MCS extraction algorithms.

ACKNOWLEDGEMENTS

This work is partially supported by SFI PI grant BEACON (09/IN.1/I2618), FCT grants POLARIS (PTDC/EIA-CCO/123051/2010) and ASPEN (PTDC/EIA-CCO/110921/2009), and INESC-ID multiannual PIDDAC funding PEst-OE/EEI/LA0021/2013.

REFERENCES

[1] G. Audemard and L. Simon. Predicting learnt clauses quality in modern SAT solvers. In *IJCAI*, pages 399–404, 2009.

[2] J. Bailey and P. J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *PADL*, pages 174–186, 2005.

[3] A. Belov, I. Lynce, and J. Marques-Silva. Towards efficient MUS extraction. *AI Commun.*, 25(2):97–116, 2012.

[4] 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.

[5] E. Birnbaum and E. L. Lozinskii. Consistent subsets of inconsistent systems: structure and behaviour. *J. Exp. Theor. Artif. Intell.*, 15(1):25–46, 2003.

[6] J. Dellert, C. Zielke, and M. Kaufmann. MUStICCa: MUS extraction with interactive choice of candidates. In *SAT*, pages 408–414, 2013.

[7] N. Eén and N. Sörensson. An extensible SAT-solver. In *SAT*, pages 502–518, 2003.

[8] A. Felfernig, M. Schubert, and C. Zehentner. An efficient diagnosis algorithm for inconsistent constraint sets. *AI EDAM*, 26(1):53–62, 2012.

[9] A. V. Gelder. Complexity analysis of propositional resolution with autarky pruning. *Discrete Applied Mathematics*, 96-97:195–221, 1999.

[10] É. Grégoire, B. Mazure, and L. Sais. Local autarkies searching for the dynamic partition of CNF formulae. In *ICTAI*, pages 107–114, 2009.

[11] M. Jampel, E. C. Freuder, and M. J. Maher, editors. *Over-Constrained Systems*, volume 1106 of *LNCS*. Springer, 1996.

[12] U. Junker. QuickXplain: Preferred explanations and relaxations for over-constrained problems. In *AAAI*, pages 167–172, 2004.

[13] H. Kleine-Büning and O. Kullmann. Minimal unsatisfiability and autarkies. In Biere et al. [4], pages 339–401.

[14] M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa. QMaxSAT: A partial Max-SAT solver. *JSAT*, 8(1/2):95–100, 2012.

[15] O. Kullmann. Investigations on autark assignments. *Discrete Applied Mathematics*, 107(1-3):99–137, 2000.

[16] O. Kullmann. On the use of autarkies for satisfiability decision. *Electronic Notes in Discrete Mathematics*, 9:231–253, 2001.

[17] O. Kullmann. Constraint satisfaction problems in clausal form I: Autarkies and deficiency. *Fundam. Inform.*, 109(1):27–81, 2011.

[18] O. Kullmann, I. Lynce, and J. Marques-Silva. Categorisation of clauses in conjunctive normal forms: Minimally unsatisfiable sub-clause-sets and the lean kernel. In *SAT*, pages 22–35, 2006.

[19] T. Laitinen, T. A. Junttila, and I. Niemelä. Extending clause learning DPLL with parity reasoning. In *ECAI*, pages 21–26, 2010.

[20] C. M. Li. Integrating equivalency reasoning into Davis-Putnam procedure. In *AAAI/IAAI*, pages 291–296, 2000.

[21] C. M. Li and F. Manyà. MaxSAT, hard and soft constraints. In Biere et al. [4], pages 613–631.

[22] M. H. Liffiton and K. A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40(1):1–33, 2008.

[23] M. H. Liffiton and K. A. Sakallah. Searching for autarkies to trim unsatisfiable clause sets. In *SAT*, pages 182–195, 2008.

[24] Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann. Algorithm portfolios based on cost-sensitive hierarchical clustering. In *IJCAI*, 2013.

[25] J. Marques-Silva, F. Heras, M. Janota, A. Previti, and A. Belov. On computing minimal correction subsets. In *IJCAI*, 2013.

[26] P. Meseguer, N. Bouhmala, T. Bouzoubaa, M. Irgens, and M. Sánchez. Current approaches for solving over-constrained problems. *Constraints*, 8(1):9–39, 2003.

[27] B. Monien and E. Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10(3):287–295, 1985.

[28] A. Nöhrer, A. Biere, and A. Egyed. Managing SAT inconsistencies with HUMUS. In *VaMoS*, pages 83–91, 2012.

[29] B. O'Callaghan, B. O'Sullivan, and E. C. Freuder. Generating corrective explanations for interactive constraint satisfaction. In *CP*, pages 445–459, 2005.

[30] R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[31] S. Szeider. Homomorphisms of conjunctive normal forms. *Discrete Applied Mathematics*, 130(2):351–365, 2003.

[32] S. Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. In *COCOON*, pages 548–558, 2003.

[33] S. Wieringa. *Incremental Satisfiability Solving and its Applications*. PhD thesis, Aalto University, 2014.

[34] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *J. Artif. Intell. Res. (JAIR)*, 32:565–606, 2008.