Unifying SAT-Based Approaches to Maximum Satisfiability Solving

Hannes Ihalainen HANNES.IHALAINEN@HELSINKI.FI
Jeremias Berg JEREMIAS.BERG@HELSINKI.FI
Matti Järvisalo MATTI.JARVISALO@HELSINKI.FI
Department of Computer Science,
University of Helsinki,
Finland

Abstract

Maximum satisfiability (MaxSAT), employing propositional logic as the declarative language of choice, has turned into a viable approach to solving NP-hard optimization problems arising from artificial intelligence and other real-world settings. A key contributing factor to the success of MaxSAT is the rise of increasingly effective exact solvers that are based on iterative calls to a Boolean satisfiability (SAT) solver. The three types of SAT-based MaxSAT solving approaches, each with its distinguishing features, implemented in current state-of-the-art MaxSAT solvers are the core-guided, the implicit hitting set (IHS), and the objective-bounding approaches. The objective-bounding approach is based on directly searching over the objective function range by iteratively querying a SAT solver if the MaxSAT instance at hand has a solution under different bounds on the objective. In contrast, both core-guided and IHS are so-called unsatisfiability-based approaches that employ a SAT solver as an unsatisfiable core extractor to determine sources of inconsistencies, but critically differ in how the found unsatisfiable cores are made use of towards finding a provably optimal solution. Furthermore, a variety of different algorithmic variants of the core-guided approach in particular have been proposed and implemented in solvers. It is well-acknowledged that each of the three approaches has its advantages and disadvantages, which is also witnessed by instance and problem-domain specific runtime performance differences (and at times similarities) of MaxSAT solvers implementing variants of the approaches. However, the questions of to what extent the approaches are fundamentally different and how the benefits of the individual methods could be combined in a single algorithmic approach are currently not fully understood. In this work, we approach these questions by developing UniMaxSAT, a general unifying algorithmic framework. Based on the recent notion of abstract cores, UniMaxSAT captures in general core-guided, IHS and objective-bounding computations. The framework offers a unified way of establishing quite generally the correctness of the current approaches. We illustrate this by formally showing that UniMaxSAT can simulate the computations of various algorithmic instantiations of the three types of MaxSAT solving approaches. Furthermore, UniMaxSAT can be instantiated in novel ways giving rise to new algorithmic variants of the approaches. We illustrate this aspect by developing a prototype implementation of an algorithmic variant for MaxSAT based on the framework.

1. Introduction

The declarative paradigm of maximum satisfiability (MaxSAT) (Li & Manyà, 2021; Bacchus et al., 2021) is today a viable approach to solving NP-hard optimization problems arising from AI and other real-world settings. Much of the success of MaxSAT is due to advances in practical algorithms for MaxSAT and their fine-grained implementations as MaxSAT solvers.

MaxSAT solvers can be categorized into exact (or "complete") and inexact (or "incomplete") solvers. Inexact solvers are typically based on stochastic local search (Jiang et al., 1995; Cai et al., 2014; Lei & Cai, 2018; Chu et al., 2023) and/or combinations of techniques from exact solvers (Joshi et al., 2019; Berg et al., 2019; Nadel, 2020), and are in general geared towards finding "good" solutions in relatively short time instead of providing guarantees on finding provable optimal solutions. In contrast, exact solvers are guaranteed to find optimal solutions, given enough runtime resources. A majority of research on developing increasingly effective MaxSAT solvers has to date focused on exact solvers which are also the focus of this work.

Exact MaxSAT solvers are based on one or more types of algorithmic approaches. As the main focus of this work, a great majority of modern exact MaxSAT solvers are "SAT-based" (Bacchus et al., 2021), implementing Boolean satisfiability (SAT) based MaxSAT algorithms making use of SAT solvers as "real-world NP-oracles" in an iterative fashion (Morgado et al., 2013; Ansótegui et al., 2013). Apart from the mainstream SAT-based MaxSAT solvers, we note that branch-and-bound based MaxSAT solvers—most recently integrating specific search techniques from SAT solving, such as clause learning—have also been developed (Planes, 2003; Li et al., 2005, 2006; Heras et al., 2008; Abramé & Habet, 2014, 2016; Li et al., 2021, 2022; Li & Manyà, 2021).

The underlying algorithmic approaches implemented in the various SAT-based MaxSAT solvers today can be categorized into three types: the so-called core-guided approach (Fu & Malik, 2006; Marques-Silva & Planes, 2007; Heras et al., 2011; Ansótegui et al., 2013; Morgado et al., 2013, 2014; Narodytska & Bacchus, 2014; Alviano et al., 2015; Ansótegui et al., 2016; Ansótegui & Gabàs, 2017), the implicit hitting set (IHS) approach (Davies & Bacchus, 2011, 2013; Saikko et al., 2016), and the objective-bounding approach (Fu & Malik, 2006; Eén & Sörensson, 2006; Berre & Parrain, 2010; Koshimura et al., 2012; Heras et al., 2011; Ignatiev et al., 2014). The objective-bounding approach is based on directly searching over the objective function range by iteratively querying a SAT solver if the MaxSAT instance at hand has a solution under different bounds on the objective using different strategies such as model-improving search (Eén & Sörensson, 2006; Berre & Parrain, 2010; Koshimura et al., 2012), binary search (Fu & Malik, 2006; Heras et al., 2011; Piotrów, 2020) or progression-based search (Ignatiev et al., 2014). In contrast, both core-guided and IHS are unsatisfiability-based approaches, relying on iteratively extracting sources of inconsistencies in terms of unsatisfiable cores using a SAT solver (Eén & Sörensson, 2003) as a core-extracting decision oracle. However, core-guided and IHS solvers deal with cores extracted during search differently. Core-guided algorithms reformulate the current working instance—starting with the input MaxSAT instance—to take into account the so-far extracted cores in subsequent search iterations towards an optimal solution. The various different core-guided algorithms differ in the way in which the reformulation steps change the working instance. In contrast, in each iteration of IHS search, the SAT solver is invoked on a subset of clauses of the input instance, without reformulation-style modifications to the instance. The choice of the subset of constraints to consider in each iteration is dictated by computing a (minimum-cost) hitting set of the so-far accumulated set of cores.

In practice, state-of-the-art core-guided, IHS and objective-bounding (especially model-improving) MaxSAT solvers are all competitive in terms of runtime performance. However, the relative performance on distinct problem domains can vary noticeably between solvers implementing a specific approach (Bacchus et al., 2019). The fundamental reasons behind this are not well understood, despite recent advances showing that for a specific classic variant of core-guided search, the cores extracted from the reformulated working formulas during core-guided search are tightly related to cores extracted in IHS search on the original instance (Bacchus & Narodytska, 2014; Narodytska & Bjørner, 2022). Furthermore, fundamental insights into how to combine the best of each of the three types of algorithmic approaches are currently lacking.

In this work, we develop a *general algorithmic framework* that captures core-guided, IHS, and objective-bounding computations in a unifying way. The framework is based on the recently-proposed notion of abstract cores originally presented as a performance-improving variant of IHS for MaxSAT (Berg et al., 2020) that brings a flavor of core-guided reformulation into the representation of the hitting set problems solved during IHS search. While the correctness of the objective-bounding approach is relatively straightforward to establish directly, this is not generally the case for fine-grained variants of the core-guided approach—which would also translate to non-trivial individual correctness proofs for any non-trivial combinations of, e.g., the core-guided and IHS approaches. Our framework provides a unified way of establishing the correctness of variants of core-guided and IHS approaches. The framework also has the potential of being instantiated in novel ways, thereby giving rise to new variants of provably-correct MaxSAT algorithms. While the main focus of this work is evidently on the general formal algorithmic framework, as an illustration of its potential for obtaining novel types of unsatisfiability-based algorithms we more shortly outline and provide a prototype implementation of a core-guided variant for MaxSAT obtained through the framework.

In terms of related work, the motivations underlying the UniMaxSAT framework are in part similar to generic frameworks developed for capturing reasoning performed by modern SAT solvers and closely related solver technologies. These include the inprocessing rules framework (Järvisalo et al., 2012; Fazekas et al., 2019) for capturing the various types of reasoning steps applied by inprocessing SAT solvers as well as the DPLL(T) framework (Nieuwenhuis et al., 2006) and its extensions which have been developed for fine-grained formalization of satisfiability modulo theories (SMT) solvers (Cimatti et al., 2010; Barrett et al., 2021; Bjørner & Fazekas, 2023), specific types of optimization approaches in SMT (Fazekas et al., 2018), as well as, e.g., reasoning performed by answer set (ASP) solvers (Baselice et al., 2005; Gebser et al., 2009). Both the inprocessing and the DPLL(T) framework take the view of formalizing solving steps as transition systems, describing the possible next-state transitions from a current solver state. The inprocessing framework is instantiated by specific redundancy notions—which have more recently been generalized to the realm of MaxSAT (Ihalainen et al., 2022)—which themselves can generally cover the various reasoning techniques applied in SAT solvers. However, such redundancy notions on the MaxSAT level do not allow for directly capturing the multitude of SAT-based MaxSAT algorithms. In contrast, a key motivation behind the UniMaxSAT framework is to cover all of the three main SAT-based MaxSAT solving approaches, also with the aim of providing a unifying view towards novel types of MaxSAT algorithms that would combine aspects of the different approaches in correct ways.

A shorter preliminary version of this work was presented at the IJCAI 2023 conference (Ihalainen et al., 2023). The present article considerably revises and extends on the work reported at IJCAI 2023. Most notably, we here thoroughly revise the details of the UniMaxSAT framework. As a result, the framework now allows for capturing not only core-guided and implicit hitting set approaches to MaxSAT, but more generally SAT-based MaxSAT solving, including what we will refer to as objective-bounding MaxSAT algorithms, instantiations of which include the model-improving approach as well as binary and progression-based search for MaxSAT. The revised framework is also arguably cleaner in terms of being more directly connected with how the various SAT-based MaxSAT algorithms perform search. As a result, formal proofs have been thoroughly revised and further details included, including formal explanations of how the general framework captures several further algorithmic instantiations of SAT-based MaxSAT approaches.

The rest of this article is organized as follows. We start with a background on maximum satisfiability (Section 2) and an overview of the objective-bounding, core-guided, and implicit hitting set approaches to MaxSAT solving (Section 3). Then, as the main contribution of this work, we detail the UniMaxSAT framework (Section 4) and argue that it can simulate the behavior of the objective-bounding and implicit hitting set approaches (Section 5) as well as the various variants of the core-guided approaches proposed in the literature so far (Section 6). Before conclusions, we further detail a novel variant of the core-guided approach to illustrate the use of the UniMaxSAT framework to formulate new algorithmic variants (Section 7).

2. Maximum Satisfiability

For a Boolean variable x there are two literals, x and $\bar{x}$. A clause $C = l_1 \vee \ldots \vee l_n$ is a disjunction of literals, and a conjunctive normal form (CNF) formula is a set of clauses $F = \{C_1, \ldots, C_m\}$. For a clause C, the set $\texttt{var}(C)$ consists of variables x for which either $x \in C$ or $\bar{x} \in C$. An assignment τ maps variables to 1 (true) or 0 (false). Assignments extend to a literal l, clause C and formula F standardly by $\tau(\bar{l}) = 1 - \tau(l)$, $\tau(C) = \max\{\tau(l) \mid l \in C\}$ and $\tau(F) = \min\{\tau(C) \mid C \in F\}$. An assignment τ satisfies (or is a solution to) F if $\tau(F) = 1$. Interchangeably, we may treat τ as the set of literals τ assigns to 1. Then $l \in \tau$ denotes $\tau(l) = 1$ and $\bar{l} \in \tau$ denotes $\tau(l) = 0$. The set of variables assigned by τ is $\texttt{var}(\tau) = \{x \mid x \in \tau \text{ or } \bar{x} \in \tau\}$; τ is complete for F if it assigns each variable in F a value, and otherwise partial. An assignment τ that assigns each literal in a clause C to 0 falsifies C, denoted by $\tau \supseteq \neg C$.

Pseudo-Boolean constraints are linear inequalities of form $\sum_i c_i x_i \geq k$, where each x_i is a Boolean variable, each c_i a positive coefficient, and k a positive constant. The constraint $\sum_i c_i x_i \geq k$ is satisfied by an assignment τ if $\sum_i c_i \tau(x_i) \geq k$. When we do not make assumptions about how exactly pseudo-Boolean constraints are represented as CNF formulas[1], we abstractly use $\textsc{asCNF}(\sum_i c_i x_i \geq k)$ to denote a CNF formula that is satisfied by an assignment τ iff $\sum_i c_i \tau(x_i) \geq k$. Taking a name o_k to indicate whether a

1. Various CNF encodings of pseudo-Boolean constraint have been proposed (Warners, 1998; Bailleux & Boufkhad, 2003; Sinz, 2005; Eén & Sörensson, 2006; Bailleux et al., 2009; Codish & Zazon-Ivry, 2010;

pseudo-Boolean constraint is satisfied, we also use $\textsc{asCNF}(\sum_i c_i x_i \geq k \leftrightarrow o_k)$ to denote a (CNF-representation of) a reified pseudo-Boolean constraint, i.e., a CNF formula that is satisfied by any assignment τ that sets $\tau(o_k) = 1$ iff $\sum_i c_i \tau(x_i) \geq k$. An important special case of pseudo-Boolean constraints is the so-called cardinality constraints $\sum_i x_i \geq k$, which are pseudo-Boolean constraints where each coefficient is 1. Notice how the o_k variable of a reified cardinality constraint $\textsc{asCNF}(\sum_i x_i \geq k \leftrightarrow o_k)$ essentially counts whether the number of x_i variables assigned to 1 is more or less than k.

An instance $\mathcal{F} = (F, O)$ of (weighted partial) maximum satisfiability (MaxSAT for short) consists of a CNF formula F and an objective function $O = \sum_i w_i b_i + W^{\text{LB}}$ under minimization, where w_i are positive integers and b_i are variables of F. Notice that we here include for convenience the constant term W^{LB}. This allows for explicitly representing lower bounds on costs of solutions as computed by core-guided MaxSAT algorithms (as detailed in Section 3.2).

Remark 1. *The definition of MaxSAT in terms of a CNF formula and an objective we use in this work is equivalent to the arguably more classical (clausal) definition of MaxSAT in terms of hard and weighted soft clauses in the following sense. Going from the objective function representation to the clausal representation, the clauses remain hard clauses, and each term $w_i b_i$ in the objective function O is equivalently represented as a soft clause $\langle (\bar{b_i}), w_i \rangle$, i.e., a unit soft clause $(\bar{b_i})$ with weight w_i. To the other direction, any clausal instance of MaxSAT can be converted to an instance where each soft clause is a unit clause by the blocking variable transformation (Bacchus et al., 2021) standardly employed in SAT-based MaxSAT solvers before search: introduce a fresh variable b_i for each non-unit soft clause C_i with weight w_i and replace C_i with the hard clause $C_i \vee b_i$ and the soft clause $\langle (\bar{b}_i), w_i \rangle$. After this transformation, the introduced soft unit clauses are evidently equivalent to the objective function $\sum_i w_i b_i$. For example, consider the following (clausal) MaxSAT instance (F_H, F_S) consisting of the hard clauses $F_H = \{(\bar{x} \vee b_1), (y \vee z \vee b_2)\}$ and the soft clauses $F_S = \{\langle (\bar{y} \vee x), 1 \rangle, \langle (\bar{b}_1), 2 \rangle, \langle (\bar{b}_2), 5 \rangle, \langle (\bar{z} \vee x \vee b_1), 3 \rangle\}$. Applying the blocking variable transformation for each non-unit soft clause results the set of hard clauses $F_H^b = \{(\bar{x} \vee b_1), (y \vee z \vee b_2), (\bar{y} \vee x \vee b_3), (\bar{z} \vee x \vee b_1 \vee b_4)\}$ and the set of soft clauses $F_S^b = \{\langle (\bar{b}_1), 2 \rangle, \langle (\bar{b}_2), 5 \rangle, \langle (\bar{b}_3), 1 \rangle, \langle (\bar{b}_4), 3 \rangle\}$. This instance can be equivalently represented using the objective function representation as the instance $\mathcal{F} = (F_H^b, O)$ with $O = 2b_1 + 5b_2 + b_3 + 3b_4$.*

The set $\texttt{var}(O)$ consists of variables that occur in O. A complete satisfying assignment τ to F is a solution to $\mathcal{F}$ and has cost $O(\tau) = \sum_i w_i \tau(b_i) + W^{\text{LB}}$. A solution is optimal if there are no solutions with lower costs. The cost of optimal solutions to a MaxSAT instance $\mathcal{F}$ is denoted by $\textsc{opt}(\mathcal{F})$.

Example 1. *Consider the MaxSAT instance $\mathcal{F} = (F, O)$ with $F = \{(b_1 \vee b_2 \vee x), (\bar{x} \vee b_3), (b_3 \vee b_4 \vee b_5)\}$ and $O = b_1 + b_2 + 3b_3 + b_4 + 2b_5$. An optimal solution to $\mathcal{F}$ is $\tau = \{\bar{b}_1, b_2, \bar{x}, \bar{b}_3, b_4, \bar{b}_5\}$ to $\mathcal{F}$, assigning all variables except b_2, b_4 to 0. The cost of τ is $O(\tau) = \tau(b_1) + \tau(b_2) + 3\tau(b_3) + \tau(b_4) + 2\tau(b_5) = 2$.*

Asín et al., 2011; Hölldobler et al., 2012; Abío et al., 2013; Ogawa et al., 2013; Manthey et al., 2014; Joshi et al., 2015; Paxian et al., 2018; Karpinski & Piotrów, 2019).

Algorithm 1 The objective-bounding search approach to MaxSAT

```
Input: A MaxSAT instance $\mathcal{F} = (F, O)$ where $O = \sum_i w_i b_i$.
Output: An optimal solution $\tau$ to $\mathcal{F}$.
1: $\tau^* = \emptyset$
2: while true do
3:     $w$ = Next-value-to-test()
4:     $(\text{res}, \_, \tau)$ = Extract-Core($F \cup$ asCNF($\sum_i w_i b_i \leq k \leftrightarrow o_w$), $\{o_w\}$)
5:     if res ='true' and $O(\tau) < O(\tau^*)$ then $\tau^* = \tau$
6:     if res ='false' and $w = O(\tau^*) - 1$ then return $\tau^*$
```

A clause C is *a(n unsatisfiable) core* of a MaxSAT instance $\mathcal{F} = (F, O)$ if all literals in C are objective variables (i.e., $\texttt{var}(C) \subseteq \texttt{var}(O)$) and every solution to $\mathcal{F}$ satisfies C (i.e., F logically entails C).

Example 2. *The clauses* $(b_1 \lor b_2 \lor b_3)$ *and* $(b_3 \lor b_4 \lor b_5)$ *are two of the cores of the MaxSAT instance detailed in Example 1.*

3. SAT-Based Approaches to MaxSAT

We develop a unifying algorithmic framework for modern SAT-based algorithms, capturing forms of objective-bounding search, core-guided algorithms, and algorithms based on the implicit hitting set (IHS) approach. As necessary background, we describe each of these approaches in general terms; practical solver implementations employ various heuristics and optimizations that do not affect our main contributions and, as such, are not detailed here.

Common to the three types of modern SAT-based MaxSAT algorithms is the use of an incremental SAT solver that can determine the satisfiability of CNF formulas under different sets of assumptions (Eén & Sörensson, 2003). Given a CNF formula F and a partial assignment γ^A (constituting a set of assumptions, represented as a set of literals), we abstract the SAT solver into the subroutine Extract-Core that returns a triplet (res, C, τ). Here res='true' if there is a solution $\tau \supseteq \gamma^A$ to F. If there is no such solution, res='false' and C is a clause over a subset of the variables in γ^A that is entailed by F. Invoked on F under a set of assumptions γ^A for which $F \land \gamma^A$ is unsatisfiable, modern SAT solvers provide such a clause C at termination without computational overhead. In the context of SAT-based MaxSAT algorithms, such a C found during MaxSAT search will be unsatisfiable core of the current working instance.

3.1 Objective-Bounding Search

The so-called objective-bounding search algorithms—captured by Algorithm 1—compute an optimal solution to a given MaxSAT instance (F, O) by iteratively selecting a value w (Line 3) and querying Extract-Core for a solution τ to F satisfying $O(\tau) \leq w$ (Line 4). In practice, the query is formed by encoding a pseudo-Boolean constraint enforcing the bound w on the objective O. The solution of lowest cost found so far is stored in τ^* and updated whenever the Extract-Core returns res='true' and a new solution (Line 5).

The search terminates when the algorithm establishes that there is no solution τ for which $O(\tau) \leq O(\tau^*) - 1$ (Line 6).

The different objective-bounding search algorithms differ mainly in the order in which different value choices for w are selected, which also is reflected in the termination criterion. The main approaches proposed in the literature are the so-called solution-improving (sometimes called SAT-UNSAT) search (Eén & Sörensson, 2006; Berre & Parrain, 2010; Koshimura et al., 2012; Paxian et al., 2018), UNSAT-SAT (lower-bounding) search, binary search (Fu & Malik, 2006; Heras et al., 2011; Piotrów, 2020) and progression-based search (Ignatiev et al., 2014).

Solution-improving search is an upper-bounding approach that—starting from some upper bound such as the sum of all coefficients of the objective—in each iteration sets w to be one lower than the cost of the best currently known solution τ^*. This strategy guarantees that the algorithm terminates when Extract-Core returns res=‘false’. Solution-improving search is today the most widely employed objective-bounding search algorithm. In contrast, UNSAT-SAT search is a lower-bounding approach that starts from $k = 0$ and increments w each time Extract-Core returns res=‘false’. The search terminates when Extract-Core returns res=‘true’.[2] Binary search algorithms perform binary search over the range of the objective function, maintaining both an upper and a lower bound on optimal cost. The upper bound is updated whenever Extract-Core returns res=‘true’, and the lower bound whenever Extract-Core returns res=‘false’.[3] Finally, progression-based search can be seen as a combination of UNSAT-SAT and binary search. Progression-based search initially tests the values $w = (2^0 - 1, 2^1 - 1, 2^2 - 1, \dots)$ until Extract-Core reports res=‘true’ and a solution on a particular ith iteration. At this stage, the algorithm has determined that the optimal cost is between $2^{i-1} - 1$ and $2^i - 1$ and switches to binary search using these as the initial lower and upper bounds.

3.2 Core-Guided Search

Turning to core-guided search, we outline as Algorithm 2 a general abstraction of the core-guided approach to computing an optimal solution to a given MaxSAT instance $\mathcal{F} = (F, O)$. The algorithm first initializes a set Constraints of cardinality constraints as the empty set and a reformulated objective function O^R as the objective function O (Lines 1–2). In each iteration of the main loop (Lines 3–9) a SAT solver is queried for a solution τ that (i) satisfies all clauses in F and all of the cardinality constraints in Constraints and (ii) falsifies all objective variables of the current reformulated objective O^R, i.e., $O^R(\tau) = W^{\text{LB}}$ (Lines 4–5). If there is such a τ, it is returned as an optimal solution to the input MaxSAT instance (Line 6). Otherwise, a core C of $(F \cup \text{Constraints}, O^R)$ is obtained. The core is then *relaxed* (Lines 7–9) by transforming the current working instance in a way

2. Today, UNSAT-SAT search is not commonly used. This is mainly because core-guided algorithms can be seen as refined versions of UNSAT-SAT search and generally outperform UNSAT-SAT search in practice.
3. In theory, binary search has the desirable property of guaranteed termination within a logarithmic number of calls to Extract-Core in terms of the sum of objective coefficients. In practice, however, it is commonly acknowledged by MaxSAT solver developers that implementations of binary search are often outperformed by implementations of solution-improving search. This is due to the fact that the intermediate calls to Extract-Core that report res=‘false’ can often be challenging when solving real-world instances.

Algorithm 2 The core-guided approach to MaxSAT

Input: A MaxSAT instance $\mathcal{F} = (F, O)$.
Output: An optimal solution τ to $\mathcal{F}$.

```
1: $\textsc{Constraints} = \emptyset$
2: $O^R = O$
3: while true do
4:     $\gamma^A = \{\bar{x} \mid x \in \mathtt{var}(O^R)\}$
5:     $(\text{res}, C, \tau) = \textsc{Extract-Core}(F \cup \textsc{Constraints}, \gamma^A)$
6:     if res =‘true’ then return $\tau$
7:     $(D, out) = \textsc{Generate-Cardinality-Constraints}(C)$
8:     $\textsc{Constraints} = \textsc{Constraints} \cup D$
9:     $O^R = \textsc{Refine-Objective}(C, out, O^R)$
```

that enables (at most) one variable in core C to incur cost in subsequent iterations. This is achieved by adding a cardinality constraint over the core to Constraints (Lines 7–8) and updating the current working objective (Line 9).

Conceptually, modern core-guided algorithms differ mainly in the specifics of the core-relaxation step. We detail the relaxation of the core-guided OLL algorithm (Andres et al., 2012; Morgado et al., 2014) as arguably one of the most successful core-guided approaches. In OLL, an invocation of Generate-Cardinality-Constraints(C) returns a set of cardinality constraints $D = \{\text{asCNF}(\sum_{x \in C} x \geq j \leftrightarrow o_j^C) \mid 2 \leq j \leq |C|\}$ and a set $out = \{o_2^C \dots o_{|C|}^C\}$ of *output* variables. Intuitively, as enforcing o_k^C to 0 limits the number of literals in C assigned to 1 to at most k, the new cardinality constraints define output variables that count the number of literals in C assigned to 1 in subsequent iterations. The output variable with index 1 is not introduced, since the fact that C is a core implies that every solution to the instance assigns at least one literal to 1. In the objective reformulation step (Refine-Objective procedure in Algorithm 2) OLL adds the newly-introduced outputs to the objective in a way that preserves the set of optimal solutions. The coefficient of each $x \in C$ is decreased by $w^C = \min_{x \in C_i}\{O^R(x)\}$, removing from O^R every literal whose coefficient decreases to 0. The coefficient of each output variable in *out* is set to w^C and the constant term of O^R is increased by w^C. During the reformulation step, the coefficient of at least one variable in C decreases to 0. Thus, at least one more literal may incur cost in subsequent iterations.

Example 3. *Invoke OLL on $\mathcal{F} = (F, O)$ from Example 1. The first call to* Extract-Core *is under the assumptions $\gamma^A = \{\bar{b_1}, \bar{b_2}, \bar{b_3}, \bar{b_4}, \bar{b_5}\}$. Let the first core obtained be $C_1 = (b_1 \vee b_2 \vee b_3 \vee b_4 \vee b_5)$. Relaxing C_1 introduces the cardinality constraint* asCNF$(\sum_{x \in C_1} x \geq i \leftrightarrow o_i^1)$ *for $i = 2, 3, 4, 5$. The new objective O^R is formed based on the following observations: (i) as C_1 is a core, any solution to $\mathcal{F}$ assigns one literal in C_1 to 1 and as such incurs 1 cost in O, (ii) each additional literal of C_1 assigned to 1 should incur precisely 1 more cost. The new objective is $O^R = 2b_3 + b_5 + o_2^1 + o_3^1 + o_4^1 + o_5^1 + 1$. Notice how observation (i) results in the addition of a constant term 1 and observation (ii) in the addition of the outputs of the new cardinality constraint to the objective.*

Algorithm 3 The implicit hitting set approach to MaxSAT

```
Input: A MaxSAT instance $\mathcal{F} = (F, O)$.
Output: An optimal solution $\tau$ to $\mathcal{F}$.
1: $\mathcal{K} = \emptyset$
2: while true do
3:     $\gamma^A = \{\bar{x} \mid x \in \texttt{var}(O) \setminus \textsc{Mincost-HS}(\mathcal{K})\}$
4:     $(\text{res}, C, \tau) = \textsc{Extract-Core}(F, \gamma^A)$
5:     if res = 'true' then return $\tau$
6:     else $\mathcal{K} = \mathcal{K} \cup C$
```

The next call to Extract-Core *is under the assumptions* $\gamma^A = \{\bar{b_3}, \bar{b_5}, \bar{o}_2^1, \bar{o}_3^1, \bar{o}_4^1, \bar{o}_5^1\}$*. Let the next core obtained be* $C_2 = (o_2^1 \vee b_3)$*. Relaxing* C_2 *introduces the cardinality constraint* asCNF$(\sum_{x \in C_2} x \geq 2 \leftrightarrow o_2^2)$ *and the new objective* $O^R = b_3 + b_5 + o_3^1 + o_4^1 + o_5^1 + o_2^2 + 2$*. The set of assumptions in the third call to* Extract-Core *is* $\gamma^A = \{\bar{b_3}, \bar{b_5}, \bar{o}_3^1, \bar{o}_4^1, \bar{o}_5^1, \bar{o}_2^2\}$*. One potentially obtained assignment is now* $\tau = \{b_1, \bar{b_2}, \bar{x}, \bar{b_3}, b_4, \bar{b_5}\}$ *that also assigns* $\{o_2^1, \bar{o}_3^1, \bar{o}_4^1, \bar{o}_5^1, \bar{o}_2^2\}$*. The assignment is returned as an optimal solution to the input instance* $\mathcal{F}$*.*

3.3 Implicit Hitting Set Approach

A generic abstraction of the implicit hitting set (IHS) approach to MaxSAT is outlined as Algorithm 3. IHS iteratively extracts cores of a given MaxSAT instance $\mathcal{F} = (F, O)$ and stores them in the set $\mathcal{K}$. In contrast to core-guided algorithms, instead of reformulating the objective after each core-extraction step, IHS invokes the Mincost-HS$(\mathcal{K})$ procedure that computes a minimum-cost hitting set (MCHS) over $\mathcal{K}$ under O. Here an MCHS is a minimum-cost (in terms of O) subset hs of the objective variables such that by assigning the variables in hs to 1 all cores in $\mathcal{K}$ are satisfied. In each iteration of the main loop (Lines 2–6), Extract-Core is queried for a solution that falsifies all objective variables that are not contained in the hs computed over the current set of cores (Lines 3–4). (Note that the assumptions γ^A set up on Line 3 constitute a partial assignment over the objective variables that can be extended to a solution to $\mathcal{K}$ in a unique way.) If there is such a τ, it is an optimal solution to the input instance (Line 5). Otherwise, a new core is obtained and added to $\mathcal{K}$ (Line 6). The MCHS computed in each iteration represents a way of satisfying all cores found so far in an optimal way under O, this giving a lower bound on optimal cost of $\mathcal{F}$. IHS iterates until the most recent MCHS can be extended to a solution to F. At which point, the solution satisfies ("hits") *all* cores—not only those currently accumulated in $\mathcal{K}$—of the instance, and is thereby an optimal solution to $\mathcal{F}$.

Example 4. *Invoke Algorithm 3 on the MaxSAT instance* $\mathcal{F} = (F, O)$ *from Example 1. In the first iteration there are no cores and hence* Mincost-HS$(\mathcal{K}) = \emptyset$*. The first call to* Extract-Core *is under the assumptions* $\gamma^A = \{\bar{b_1}, \ldots, \bar{b_5}\}$*. There are a number of cores that could be returned; let the first core obtained be* $C_1 = (b_1 \vee b_2 \vee b_3 \vee b_4 \vee b_5)$*. In the second iteration, there are three different MCHSes over* $\mathcal{K} = \{C_1\}$*. Assume that* Mincost-HS *returns* $\{b_1\}$*. The assumptions for the next call to* Extract-Core *are* $\gamma^A = \{\bar{b_2}, \bar{b_3}, \bar{b_4}, \bar{b_5}\}$*. Assume that the next core obtained is* $C_2 = (b_3 \vee b_4 \vee b_5)$*. In this third iteration, the only MCHS over* $\mathcal{K} = \{C_1, C_2\}$ *is* $\{b_4\}$*. The assumptions for the next call to* Extract-Core *are $\gamma^A = \{\bar{b_1}, \bar{b_2}, \bar{b_3}, \bar{b_5}\}$. Assume that the next core is $C_3 = (b_1 \vee b_2 \vee b_3)$. In this fourth iteration, there are two possible MCHSs over $\mathcal{K} = \{C_1, C_2, C_3\}$. Assume that* Mincost-HS *returns $\{b_2, b_4\}$. This leads to the assumptions $\gamma^A = \{\bar{b_1}, \bar{b_3}, \bar{b_5}\}$. Given these assumptions,* Extract-Core *returns the solution $\tau = \{\bar{b_1}, b_2, \bar{x}, \bar{b_3}, b_4, \bar{b_5}\}$ as an optimal solution to $\mathcal{F}$.*

4. UniMaxSAT: A General Framework for SAT-Based MaxSAT Algorithms

As the main contribution of this article, we present UniMaxSAT as a general algorithmic framework unifying SAT-based MaxSAT algorithms. The framework makes use of the notion of abstract cores originally proposed as a basis for a refinement of IHS (Berg et al., 2020). Here, going considerably beyond their original intended purpose, we build on abstract cores to obtain a framework that captures SAT-based MaxSAT algorithms in general terms.

4.1 Abstraction Sets and Abstract Cores

We start by defining abstraction sets and abstract cores. On a high level, abstraction sets and abstract cores of a MaxSAT instance capture generic properties of the instance compactly in the sense that a large number of "standard" cores would be needed to express the same properties (Berg et al., 2020).

Informally speaking, an abstraction set models a relationship between a set of *input literals in* and a set *out* of output literals via a CNF formula D. In typical practical instantiations, the formula D corresponds to a cardinality constraint that essentially counts the number of input literals assigned to 1 by satisfying assignments of D, assigning the kth output literal to 1 if and only if k input literals are assigned to 1. In the following, we give a more general definition that is sufficient for proving the correctness of UniMaxSAT.

Definition 1. *An abstraction set* $\textsc{ab} = (in, D, out)$ *consists of a set in of input literals, a set out of output literals, and a satisfiable CNF formula D over a superset of the set of literals $in \cup out$, i.e., it holds that $\textbf{var}(in \cup out) \subseteq \textbf{var}(D)$. Solutions to D are uniquely defined by assignments to the inputs: for any assignment τ over in there is exactly one extension $\tau^E \supseteq \tau$ that satisfies D.*

For a given abstraction set $\textsc{ab} = (in, D, out)$, we refer to D as the definitions of the outputs *out*. For a collection $\mathcal{AB} = \{(in_i, D_i, out_i) \mid i = 1, \ldots, n\}$ of abstraction sets, the CNF formula $\text{DEF}(\mathcal{AB}) = \bigcup_{i=1}^{n} D_i$ is the conjunction of the definitions in $\mathcal{AB}$, $\text{OUTS}(\mathcal{AB}) = \bigcup_{i=1}^{n} out_i$ is the set of outputs occurring in $\mathcal{AB}$, and $\text{INPUTS}(\mathcal{AB}) = \bigcup_{i=1}^{n} in_i$ is the set of inputs occurring in $\mathcal{AB}$. We say that $\mathcal{AB}$ is *feasible* for a MaxSAT instance $\mathcal{F} = (F, O)$ if $\text{DEF}(\mathcal{AB})$ does not change the set of solutions to $\mathcal{F}$, i.e., if every solution τ to F can be extended to a solution $\tau^E \supseteq \tau$ to $F \cup \text{DEF}(\mathcal{AB})$. We will only consider collections of abstraction sets that are feasible for MaxSAT instances at hand.

An abstract core of a MaxSAT instance $\mathcal{F} = (F, O)$ is a clause that is logically entailed by F and the definitions of some feasible collection of abstraction sets. Importantly, an abstract core can contain both objective variables and outputs of abstraction sets.

Definition 2. *For a MaxSAT instance $\mathcal{F} = (F, O)$ and collection $\mathcal{AB}$ of feasible abstraction sets, a clause C is an* abstract core *of $\mathcal{F}$ wrt $\mathcal{AB}$ if*

(i) $\textbf{var}(C) \subseteq (\textbf{var}(O) \cup \textbf{var}(\text{OUTS}(\mathcal{AB}))$ *and*

(ii) $\tau(C) = 1$ *for each solution* τ *to* $F \cup \text{DEF}(\mathcal{AB})$.

Every (standard) core of a MaxSAT instance $\mathcal{F}$ is also an abstract core of $\mathcal{F}$ with respect to any collection of feasible abstraction sets.

Example 5. *Consider the MaxSAT instance* $\mathcal{F} = (F, O)$ *from Example 1 and the abstraction set*

$$\text{AB} = (\{b_1, b_2, b_3, b_4, b_5\}, \{\text{ASCNF}(\sum_{1 \le i \le 5} b_i \ge j \leftrightarrow o_j) \mid j = 2, 3, 4, 5\}, \{o_2, o_3, o_4, o_5\}).$$

We have that $C = (o_2 \vee b_3)$ *is an abstract core of* $\mathcal{F}$ *as any assignment that satisfies* $F \cup \text{DEF}(\text{AB})$ *must assign either* $b_3 = 1$ *or at least two variables of* $\{b_1, b_2, b_3, b_4, b_5\}$ *to* 1*, thereby forcing* $o_2 = 1$.

Note how the abstract core C in Example 5 corresponds to the core C_2 in Example 3. This demonstrates how cores of the reformulated instance extracted by OLL can be viewed as abstract cores of the original instance.[4]

The UniMaxSAT framework is based on computing minimum-cost solutions to abstract cores and extending them to a solution to the MaxSAT instance at hand. To differentiate solutions to an input MaxSAT instance from solutions to cores, we call solutions to a set of abstract cores *candidate solutions* (or *candidates* for short). More precisely, for a MaxSAT instance $\mathcal{F} = (F, O)$, a collection $\mathcal{AB}$ of feasible abstraction sets and a set $\mathcal{K}$ of abstract cores, an assignment δ that satisfies $\mathcal{K} \cup \text{DEF}(\mathcal{AB})$ and assigns each variable in $\texttt{var}(O)$ is a $(\mathcal{K}, \mathcal{AB})$–candidate of $\mathcal{F}$ and has cost $O(\delta)$. A $(\mathcal{K}, \mathcal{AB})$–candidate δ is minimum-cost if $O(\delta) \le O(\delta^*)$ for all $(\mathcal{K}, \mathcal{AB})$–candidates δ^*.

Abstraction sets and abstract cores are employed in the UniMaxSAT framework for computing lower bounds (which are in turn used to prove the optimality of solutions) based on the following proposition.

Proposition 1. *Let* $\mathcal{F} = (F, O)$ *be a MaxSAT instance,* $\mathcal{AB}$ *a set of feasible abstraction sets,* $\mathcal{K}$ *a set of abstract cores of* $\mathcal{F}$ *wrt* $\mathcal{AB}$*, and* δ *a minimum-cost* $(\mathcal{K}, \mathcal{AB})$*–candidate. Then* $O(\delta) \le \text{OPT}(\mathcal{F})$.

Proof. Consider an arbitrary solution τ to $\mathcal{F}$. By feasibility of $\mathcal{AB}$ there is an extension $\delta^\tau \supseteq \tau$ which is a solution to $F \cup \text{DEF}(\mathcal{AB})$ and for which $O(\delta^\tau) = O(\tau)$. By the definition of abstract cores, δ^τ is a $(\mathcal{K}, \mathcal{AB})$–candidate. Since δ is minimum-cost, we have that $O(\tau) = O(\delta^\tau) \ge O(\delta)$. As τ is an arbitrary solution to $\mathcal{F}$, we conclude that $\text{OPT}(\mathcal{F}) \ge O(\delta)$. $\square$

A simple corollary to Proposition 1 is that any assignment τ that extends a minimum-cost $(\mathcal{K}, \mathcal{AB})$–candidate δ and satisfies F is an optimal solution to $\mathcal{F}$. The UniMaxSAT framework we develop works intuitively by iteratively computing minimum-cost $(\mathcal{K}, \mathcal{AB})$–candidates for an increasing set $\mathcal{AB}$ and $\mathcal{K}$ of feasible abstraction sets, and abstract cores, respectively, and checking whether they can be extended to solutions to the whole instance. Each check either determines that an optimal solution has been found or provides a new

4. Viewing cores of the reformulated instance during OLL search as abstract cores bears resemblance to previous work on analyzing core-guided solvers in which cores of the original instance are separated from cores of the reformulated instance ("metas") (Narodytska & Bjørner, 2022; Katsirelos, 2023).

abstract core that is falsified by the current $(\mathcal{K}, \mathcal{AB})$–candidate. In the latter case, the new abstract core is an explanation for why the current $(\mathcal{K}, \mathcal{AB})$–candidate can not be extended to an optimal solution of the instance at hand. While this is similar to a correctness proof for basic IHS search (see, e.g., (Davies & Bacchus, 2011)), the generality of UniMaxSAT allows for capturing also other types of SAT-based MaxSAT algorithms. Intuitively, the ability of UniMaxSAT to simulate core-guided algorithms follows from (i) the use of abstract cores and (ii) the fact that UniMaxSAT rules out not only complete $(\mathcal{K}, \mathcal{AB})$–candidates but also partial assignments that extend solely to minimum-cost $(\mathcal{K}, \mathcal{AB})$–candidates.

The following notion of a (minimum-cost) $(\mathcal{K}, \mathcal{AB})$–abstract candidate is central for establishing the correctness and generality of UniMaxSAT.

Definition 3. *Let $\mathcal{F} = (F, O)$ be a MaxSAT instance, $\mathcal{AB}$ a collection of feasible abstraction sets and $\mathcal{K}$ a set of abstract cores wrt to $\mathcal{AB}$. A partial assignment γ^A over a subset of the variables in $\textbf{\textit{var}}(\mathcal{K}) \cup \textbf{\textit{var}}(O)$ is a $(\mathcal{K}, \mathcal{AB})$–abstract candidate if*

(i) there is at least one extension $\tau \supseteq \gamma^A$ which is a solution to $\text{DEF}(\mathcal{AB}) \cup \mathcal{K}$, i.e., a $(\mathcal{K}, \mathcal{AB})$–candidate of $\mathcal{F}$, and

(ii) all such extensions are minimum-cost $(\mathcal{K}, \mathcal{AB})$–candidates.

Example 6. *Consider the MaxSAT instance $\mathcal{F} = (F, O)$ from Example 1, the empty collection $\mathcal{AB} = \emptyset$ of abstraction sets and the set $\mathcal{K} = \{(b_1 \vee b_2 \vee b_3), (b_3 \vee b_4 \vee b_5)\}$ of abstract cores. One minimum-cost $(\mathcal{K}, \mathcal{AB})$–candidate is $\delta = \{\bar{b_1}, b_2, \bar{b_3}, b_4, \bar{b_5}\}$, and one $(\mathcal{K}, \mathcal{AB})$–abstract candidate is $\gamma^A = \{\bar{b_1}, \bar{b_3}, \bar{b_5}\}$ since the only extension of γ^A to a solution to $\mathcal{K}$ is δ. The set $\{\bar{b_1}, \bar{b_3}\}$ is not a $(\mathcal{K}, \mathcal{AB})$–abstract candidate since it extends to the $(\mathcal{K}, \mathcal{AB})$–candidate $\{\bar{b_1}, b_2, \bar{b_3}, \bar{b_4}, b_5\}$ to $\mathcal{K}$ which is not minimum-cost.*

An important insight is that the assumptions enforced during the iterations of a core-guided algorithm can be seen as $(\mathcal{K}, \mathcal{AB})$–abstract candidates of the set $\mathcal{AB}$ of abstraction sets that corresponds to the cardinality constraints added by the core-guided algorithm and the set $\mathcal{K}$ of cores of the reformulated instance extracted by the algorithm. The following example illustrates this for the OLL algorithm (we will detail other core-guided algorithms in Section 5).

Example 7. *Recall the MaxSAT instance $\mathcal{F} = (F, O)$ from Example 1. Consider the core $C = (b_1 \vee b_2 \vee b_3 \vee b_4 \vee b_5)$, and the abstraction set $\textsc{ab} = (in, D, out)$ with $in = \{b_1, b_2, b_3, b_4, b_5\}$, $out = \{o^C_2, o^C_3, o^C_4, o^C_5\}$ and $D = \{\textsc{as}\text{CNF}(\sum_{x \in C} x \geq i \leftrightarrow o^C_i) \mid i = 2, 3, 4, 5\}$. Let $\mathcal{K} = \{C\}$ and $\mathcal{AB} = \{\textsc{ab}\}$. The set $\gamma^A = \{\bar{b_3}, \bar{b_5}, \bar{o}^C_2, \bar{o}^C_3, \bar{o}^C_4, \bar{o}^C_5\}$ is a $(\mathcal{K}, \mathcal{AB})$–abstract candidate since it can be extended to a solution to $\mathcal{K} \cup \text{DEF}(\mathcal{AB})$ by assigning exactly one literal in $\{b_1, b_2, b_4\}$ to 1 and the rest to 0. Note that γ^A is exactly the set of assumptions that* Extract-Core *is queried under during the second iteration of the OLL invocation detailed in Example 3.*

4.2 UniMaxSAT in Detail

With the necessary preliminaries in place, we now detail the UniMaxSAT framework. A high-level view to the framework is shown in Figure 1, and the framework is detailed in

Algorithm 4 UniMaxSAT, a unifying framework for SAT-based MaxSAT algorithms

Input: A MaxSAT instance $\mathcal{F} = (F, O)$.

Output: An optimal solution τ^* to $\mathcal{F}$.

```
1: $\mathcal{AB}^1 = \emptyset$, $\mathcal{K}^1 = \emptyset$
2: for $i = 1 \ldots$ do
3:     $(\gamma_i^A, \texttt{lb}_i) = \textsc{Optimize}(O, \mathcal{AB}^i, \mathcal{K}^i)$
4:     $(\text{res}, C_i, \tau) = \textsc{Extract-AbstractCore}(F, \text{DEF}(\mathcal{AB}^i), \gamma_i^A)$
5:     if res = 'true' and $O(\tau) = \texttt{lb}_i$ then return $\tau$
6:     $\mathcal{K}^{i+1} = \{C_i\} \cup \mathcal{K}^i$
7:     $\mathcal{AB}^{i+1} = \mathcal{AB}^i \cup \textsc{Add-AbstractionSets}(\mathcal{K}^{i+1})$
```

pseudo-code as Algorithm 4. Given a MaxSAT instance $\mathcal{F} = (F, O)$ as input, UniMaxSAT outputs an optimal solution to $\mathcal{F}$.[5]

UniMaxSAT accumulates two sets, $\mathcal{AB}$ and $\mathcal{K}$, of abstraction sets and abstract cores, respectively. In each iteration, the Optimize subroutine computes an assignment γ^A over $\text{OUTS}(\mathcal{AB}) \cup \texttt{var}(O)$ and a lower bound $\texttt{lb}$ for the optimal cost of $\mathcal{F}$. The subroutine Extract-AbstractCore is invoked to check for an extension of γ^A to a solution to F. If such an extension τ exists (i.e., if res ='true'), UniMaxSAT checks if the cost of τ matches $\texttt{lb}$. If this is the case, UniMaxSAT terminates and returns τ as an optimal solution. Otherwise, a new abstract core falsified by γ^A is obtained and added to $\mathcal{K}$.

To ensure termination, we require that the γ^A returned by Optimize must correspond to a $(\mathcal{K}, \mathcal{AB})$–abstract candidate sufficiently often. Furthermore, when γ^A is a $(\mathcal{K}, \mathcal{AB})$–abstract candidate, the lower bound $\texttt{lb}$ returned by Optimize must equal to the costs of its

5. We note that the framework as presented here is a significant modification of the framework described in the preliminary version of this article (Ihalainen et al., 2023). In particular, the correctness of the present version does not require computing an abstract candidate in every iteration, only that each iteration is succeeded by another one on which an abstract candidate is computed. Compared to the preliminary version, this modification allows not only to further capture objective-bounding search algorithms but also more directly capture core-guided search algorithms that do not compute abstract candidates in every iteration.

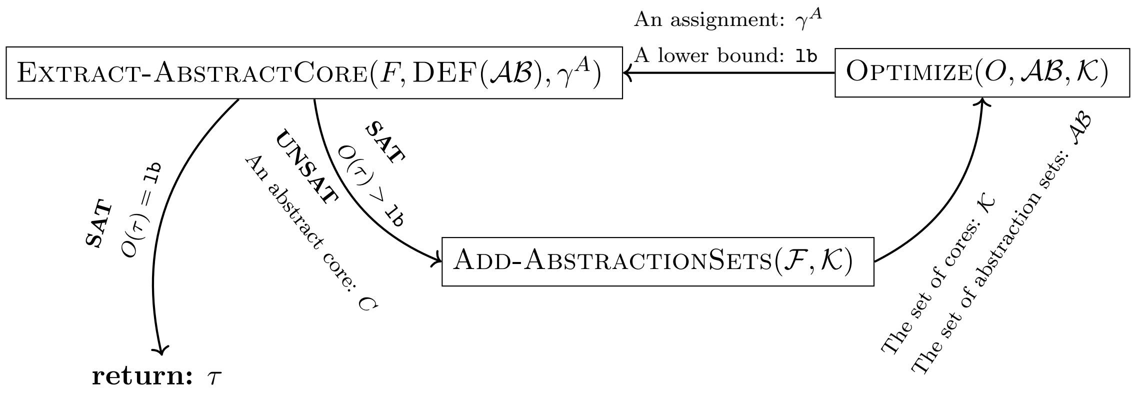

An assignment: γ^A
A lower bound: lb
$\textsc{Extract-AbstractCore}(F, \text{DEF}(\mathcal{AB}), \gamma^A)$
$\textsc{Optimize}(O, \mathcal{AB}, \mathcal{K})$
SAT
$O(\tau) = \texttt{lb}$
UNSAT
An abstract core: C
SAT
$O(\tau) > \texttt{lb}$
$\textsc{Add-AbstractionSets}(\mathcal{F}, \mathcal{K})$
The set of cores: $\mathcal{K}$
The set of abstraction sets: $\mathcal{AB}$
return: τ

Figure 1: A schematic overview of UniMaxSAT, invoked on a MaxSAT instance (F, O).

extensions. Since all such extensions are minimum-cost, it follows that, whenever OPTIMIZE computes a $(\mathcal{K}, \mathcal{AB})$–abstract candidate, the lower bound OPTIMIZE returns is as high as possible in terms of the currently accumulated set of cores. Note that OPTIMIZE does not need to identify that the assignment it computes is a $(\mathcal{K}, \mathcal{AB})$–abstract candidate. (The identification of $(\mathcal{K}, \mathcal{AB})$–abstract candidate could be computationally challenging for many practical instantiations and is in fact not required for the correctness of UNIMAXSAT.)

We formalize the correctness of Algorithm 4 in the following terms. UNIMAXSAT terminates on any MaxSAT instance and outputs an optimal solution to the input MaxSAT instance at hand, subject to the generic properties of its three subroutines. Importantly, the correctness of the general framework allows for establishing the correctness of any of its instantiations—including variants of objective-bounding, core-guided and IHS algorithms for MaxSAT—by showing how each algorithm can be viewed as an instantiation of UNIMAXSAT.

First, we establish general conditions that instantiations of OPTIMIZE need to meet in order to guarantee that UNIMAXSAT correctly computes an optimal solutions to an arbitrary input MaxSAT instance.

Definition 4 (Correctness condition). *An instantiation of* OPTIMIZE *satisfies the correctness condition if the following conditions hold at every iteration i of* UNIMAXSAT when invoked on an arbitrary input MaxSAT instance $\mathcal{F} = (F, O)$.

1. γ_i^A assigns a subset of the variables in $\texttt{var}(O) \cup \text{OUTS}(\mathcal{AB}^i)$.
2. $\texttt{lb}_i$ is a lower bound on the optimal cost of $\mathcal{F}$, i.e., $\texttt{lb}_i \leq \text{OPT}(\mathcal{F})$.
3. If γ_i^A is a $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate, then $\texttt{lb}_i$ is equal to the cost of an extension of γ_i^A to a $(\mathcal{K}^i, \mathcal{AB}^i)$–candidate of $\mathcal{F}$.
4. There exists an $r \geq 0$ such that OPTIMIZE returns a $(\mathcal{K}^{i+r}, \mathcal{AB}^{i+r})$–abstract candidate in iteration $i + r$.

In words, condition 1 ensures that in the iterations in which EXTRACT-ABSTRACTCORE determines the instance to be unsatisfiable, an abstract core of the instance is obtained. Condition 2 ensures that UNIMAXSAT does not terminate before finding an optimal solution, while conditions 3 and 4 ensure that termination takes place eventually. The following main theorem formalizes this intuition and establishes generic conditions that the other subroutines of UNIMAXSAT need to satisfy to ensure the correctness of UNIMAXSAT.

Theorem 1. *Assume that the following three properties hold in every iteration i of* UNIMAXSAT *on an input MaxSAT instance $\mathcal{F} = (F, O)$ that has a solution.*

1. OPTIMIZE satisfies the correctness condition (Definition 4).
2. EXTRACT-ABSTRACTCORE$(F, \text{DEF}(\mathcal{AB}^i), \gamma_i^A)$ computes either a solution $\tau \supseteq \gamma_i^A$ to $F \cup \text{DEF}(\mathcal{AB}^i)$ or a(n abstract) core C_i that is satisfied by all solutions to $F \cup \text{DEF}(\mathcal{AB}^i)$ and falsified by γ_i^A.
3. $\mathcal{AB}^i$ is feasible for $\mathcal{F}$.

Then UniMaxSAT terminates and returns an optimal solution to $\mathcal{F}$.

The formal proof of Theorem 1 relies on the following lemma stating that, in each iteration i in which a $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate of $\mathcal{F}$ is computed for the current set $\mathcal{AB}^i$ and $\mathcal{K}^i$ of abstraction sets and abstract cores, respectively, the set of assignments from which Optimize will return an assignment shrinks.

Lemma 1. *Invoke* UniMaxSAT *on a MaxSAT instance* $\mathcal{F} = (F, O)$ *and consider an iteration* i*. Let* $\mathcal{K}^i$ *and* $\mathcal{AB}^i$ *be the set of abstract cores and abstraction sets obtained so far, respectively, and denote by* $\textsc{obj-sols}^i$ *the restrictions of all solutions to* $\mathcal{K}^i \cup \mathrm{DEF}(\mathcal{AB}^i)$ *onto* $\mathtt{var}(O)$*. Assume* UniMaxSAT *does not terminate on iteration* i *and* Optimize *computes an* $(\mathcal{K}^i, \mathcal{AB}^i)$*–abstract candidate of* $\mathcal{F}$*. Then* $\textsc{obj-sols}^{i+1} \subsetneq \textsc{obj-sols}^i$*.*

Proof. The fact that every element in $\textsc{obj-sols}^{i+1}$ is also an element of $\textsc{obj-sols}^i$ follows from the fact that the sets of abstract cores and abstraction sets monotonically increase during the execution of UniMaxSAT. To show that there is a $\tau^o \in \textsc{obj-sols}^i \setminus \textsc{obj-sols}^{i+1}$, consider the $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate γ_i^A and the abstract core C_i computed in iteration i. By definition, there is a minimum-cost $(\mathcal{K}^i, \mathcal{AB}^i)$–candidate $\delta \supseteq \gamma_i^A \supseteq \neg C_i$ that falsifies C_i. Let τ^o be the restriction of δ onto the objective variables $\mathtt{var}(O)$. The claim of the lemma is equivalent to the claim that there is no extension of τ^o to a solution to $\mathrm{DEF}(\mathcal{AB}^i)$ that satisfies C_i. As $\mathcal{AB}$ is feasible, there is exactly one way of extending τ^o to a solution to $\mathrm{DEF}(\mathcal{AB}^i)$. Since δ is such an extension and $\neg C_i \subseteq \delta$, we conclude that τ^o cannot be extended to a solution to $\mathrm{DEF}(\mathcal{AB}^i)$ that satisfies C_i. □

We are now ready to give a proof of Theorem 1.

Proof of Theorem 1. First note that by assumption 1 of the theorem and assumption 1 of the correctness condition, whenever Extract-AbstractCore reports that the current assignment γ_i^A is not extendable to a solution on Line 4 of Algorithm 4, the clause C_i returned by Extract-AbstractCore is an abstract core of the instance at hand with respect to the current set of abstraction sets. As the sets of abstraction sets monotonically increase, $\mathcal{AB}^i \subseteq \mathcal{AB}^{i+1}$ holds for all i during the execution of UniMaxSAT. We conclude that, in each iteration i of UniMaxSAT, all clauses in $\mathcal{K}^i$ are abstract cores of $\mathcal{F}$ wrt $\mathcal{AB}^i$.

Optimality of returned solutions. Assume that UniMaxSAT terminates in iteration i and returns a solution τ. As $O(\tau) = \mathtt{lb} \leq \textsc{opt}(\mathcal{F}) \leq O(\tau)$ it follows that $O(\tau) = \textsc{opt}(\mathcal{F})$.

Termination. Given that $\mathcal{F}$ has solutions and $\mathcal{AB}^i$ is feasible, $F \cup \mathrm{DEF}(\mathcal{AB}^i)$ has a solution for each i. By the definition of abstract cores, all solutions to $F \cup \mathrm{DEF}(\mathcal{AB}^i)$ are solutions to $\mathrm{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$. Thus termination follows by showing that Optimize will eventually return a $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate γ_i^A that can be extended to a solution τ to $F \cup \mathrm{DEF}(\mathcal{AB}^i)$. This in turn follows from the number of solutions to $\mathrm{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$, in particular, from the fact that each new core rules out at least one of the—finitely many—solutions that Optimize may return. More specifically, whenever Optimize returns a $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate and UniMaxSAT does not terminate, the number of assignments to $\mathtt{var}(O)$ that can be extended to solutions to the cores decreases by Lemma 1. As Optimize satisfies the correctness condition, the sequence of iterations in which it returns $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidates is infinite. This implies that eventually a $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate can be extended to a solution to $F \cup \mathrm{DEF}(\mathcal{AB}^i)$. □

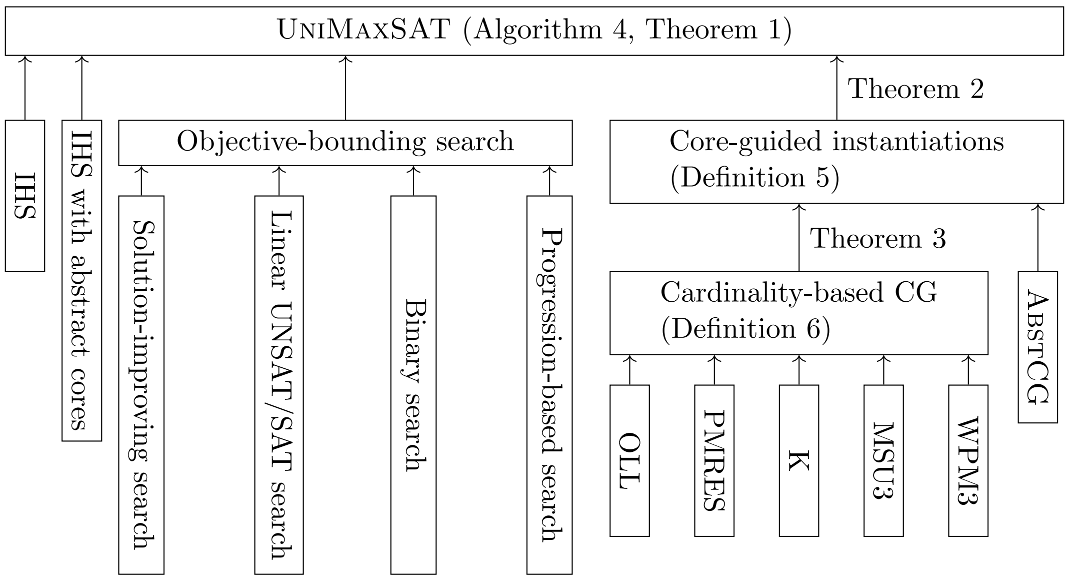

UniMaxSAT (Algorithm 4, Theorem 1)
Theorem 2
IHS
IHS with abstract cores
Objective-bounding search
Solution-improving search
Linear UNSAT/SAT search
Binary search
Progression-based search
Core-guided instantiations (Definition 5)
Theorem 3
Cardinality-based CG (Definition 6)
AbstCG
OLL
PMRES
K
MSU3
WPM3

Figure 2: The structure of results of Sections 5–7. In the figure, an arrow $A \to B$ from algorithm A to B indicates that A can be seen as a special case of B.

5. Capturing SAT-based MaxSAT Algorithms with UniMaxSAT

In this and the following section, we detail how existing SAT-based MaxSAT algorithms can be viewed as instantiations of UniMaxSAT. Specifically, for the existing objective-bounding search, IHS, and several variants of core-guided search, we explain how to instantiate the three subroutines of UniMaxSAT under the assumptions of Theorem 1 so that the resulting instantiation matches the previously proposed algorithm. By Theorem 1, this yields uniform proofs of correctness for IHS (Davies & Bacchus, 2013, 2011) (including its abstract-cores extension (Berg et al., 2020)), objective-bounding search algorithms (including solution-improving search (Eén & Sörensson, 2006; Berre & Parrain, 2010; Koshimura et al., 2012; Paxian et al., 2018)) and modern core-guided algorithms such as OLL (Andres et al., 2012; Morgado et al., 2014), MSU3 (Marques-Silva & Planes, 2007), WPM3 (Ansótegui & Gabàs, 2017), PMRES (Narodytska & Bacchus, 2014) and K (Alviano et al., 2015). In addition to proving the correctness of existing algorithms, we will furthermore outline in Section 7 a proof-of-concept novel core-guided algorithm and establish its correctness via UniMaxSAT.

Figure 2 provides a road map for Sections 5–7. We start with general observations on the extraction of abstract cores and feasibility of abstraction sets in Section 5.1, and then proceed in Sections 5.2–5.3 by detailing how IHS, IHS with abstract cores, and objective-bounding search algorithms can be viewed as direct instantiations of UniMaxSAT in a way that satisfies the assumptions of Theorem 1. Section 6 is dedicated to viewing core-guided algorithms via UniMaxSAT. For capturing the core-guided algorithms, we will define a generic *core-guided instantiation* of UniMaxSAT (Definition 5) that captures general properties of core-guided algorithms sufficient for obtaining correct instantiations of UniMaxSAT (as established by Theorem 2). We further introduce the notion of a *cardinality-based CG instantiation* of UniMaxSAT (Definition 6) that intuitively models core-guided algorithms that add a single cardinality constraint for each extracted core into the instance. We will show that all cardinality-based CG instantiations are also core-guided instantiations (Theorem 3), thereby establishing that cardinality-based CG instantiations are also correct instantiations of UniMaxSAT that satisfy Theorem 1. Turning to individual existing core-guided algorithms, we will show in Section 6.3 that the OLL, PMRES, K, WPM3 and MSU3 algorithms are cardinality-based CG instantiations of UniMaxSAT, thereby also establishing the correctness of each of the algorithms in terms of Theorem 1. Finally, to further demonstrate the usefulness of the hierarchy depicted in Figure 2 for defining new algorithms, we detail the novel AbstCG algorithm in Section 7 and show that it is a core-guided instantiation of UniMaxSAT but not a cardinality-based CG instantiation.

5.1 Extraction of Abstract Cores and Feasibility of Abstraction sets

For all specific instantiations of UniMaxSAT we discuss in Sections 5.2, 5.3, 6, and 7 the Extract-AbstractCore subroutine of UniMaxSAT is assumed to be core-extracting SAT solver. Given a MaxSAT instance $\mathcal{F} = (F, O)$, a feasible collection $\mathcal{AB}$ of abstraction sets and an assignment γ^A, Extract-AbstractCore invokes a SAT solver on $F \cup \text{DEF}(\mathcal{AB})$ under the assumptions γ^A. The search returns either a solution to $F \cup \text{DEF}(\mathcal{AB})$ that extends γ^A, or a clause entailed by $F \cup \text{DEF}(\mathcal{AB})$ that is falsified by γ^A. Assumption 2 of Theorem 1 on the Extract-AbstractCore subroutine follows directly from established properties of incremental SAT solvers (Eén & Sörensson, 2003; Audemard et al., 2013) instantiating the CDCL SAT solving paradigm (Silva & Sakallah, 1999; Zhang et al., 2001; Marques-Silva et al., 2021).

The feasibility of all abstraction sets computed—i.e., assumption 3 of Theorem 1—follows from the fact that the definitions of every new abstraction set computed only intersect with the MaxSAT instance and previous abstraction sets on the inputs. More precisely, the abstraction sets computed in every instantiation of Add-AbstractionSets we consider satisfy the assumptions of the following lemma.

Lemma 2. *Consider a MaxSAT instance $\mathcal{F} = (F, O)$ and a set of feasible abstraction sets $\mathcal{AB}$. Let (in, D, out) be an abstraction set and assume $\textbf{\textit{var}}(D) \cap \textbf{\textit{var}}(F \cup \text{DEF}(\mathcal{AB})) \subseteq in$. Then $\mathcal{AB} \cup \{(in, D, out)\}$ is feasible for $\mathcal{F}$.*

Proof. Let τ be a solution to F. By the feasibility of $\mathcal{AB}$, there is a solution $\tau^e \supseteq \tau$ to $F \cup \text{DEF}(\mathcal{AB})$. Since the only variables of D assigned by τ^e are in in, by Definition 1 there is a solution $\tau^E \supseteq \tau^e$ to D. Such a τ^E is a solution to $F \cup \text{DEF}(\mathcal{AB} \cup \{(in, D, out)\})$, establishing the feasibility of $\mathcal{AB} \cup \{(in, D, out)\}$. □

With these considerations, we will from now on assume all abstraction sets to be feasible and the Extract-AbstractCore to be instantiated with a SAT solver.

5.2 Capturing IHS with UniMaxSAT

UniMaxSAT gives the (basic) IHS (Algorithm 3) through the following instantiation. Instantiate Add-AbstractionSets to never add any abstraction sets, i.e., so that $\mathcal{AB}^i = \emptyset$ for all i. Further, instantiate Optimize as a procedure that, given a set $\mathcal{K}$ of cores, returns the tuple $(\gamma^A, \texttt{lb})$ where $\gamma^A = \{\bar{x} \mid x \in \texttt{var}(O) \setminus \text{Mincost-HS}(\mathcal{K})\}$ is an assignment that sets all literals in the objective to 0 except the ones in the most recent minimum-cost hitting set $\text{Mincost-HS}(\mathcal{K})$ over $\mathcal{K}$. The value of `lb` is the sum of the coefficients of the variables in the minimum-cost hitting set. The correctness of Algorithm 3 now follows by Theorem 1 by observing that the only extension of γ^A to a $(\mathcal{K}, \emptyset)$–candidate is $\delta = \{\bar{x} \mid x \in \texttt{var}(O) \setminus \text{Mincost-HS}(\mathcal{K})\} \cup \{x \mid x \in \text{Mincost-HS}(\mathcal{K})\}$; this candidate is minimum-cost and has $O(\delta) = \texttt{lb}$. Hence γ^A is a $(\mathcal{K}, \emptyset)$–abstract candidate and `lb` a lower bound on the optimal cost by Proposition 1.

Example 8. *Invoke the instantiation of* UniMaxSAT *that simulates the basic IHS algorithm on MaxSAT instance* $\mathcal{F} = (F, O)$ *from Example 1. In the first iteration* ($i = 1$), *we have* $\mathcal{K}^1 = \emptyset$ *and* $\mathcal{AB}^1 = \emptyset$. *As such the call* $\text{Optimize}(O, \mathcal{AB}^1, \mathcal{K}^1)$ *returns* $\gamma_1^A = \{\bar{b_1}, \ldots \bar{b_5}\}$ *and* $\texttt{lb}_1 = 0$. *The subsequent invocation of* $\text{Extract-AbstractCore}(F, \text{DEF}(\mathcal{AB}^1), \gamma_1^A)$ *returns* res =*'false' and (for example)* $C_1 = (b_1 \vee b_2 \vee b_3 \vee b_4 \vee b_5)$. *The first iteration ends with* UniMaxSAT *setting* $\mathcal{K}^2 = \{C_1\}$ *and* $\mathcal{AB}^2 = \emptyset$. *In the second iteration, the call* $\text{Optimize}(O, \mathcal{AB}^2, \mathcal{K}^2)$ *may return an* $(\mathcal{K}^2, \mathcal{AB}^2)$*-abstract candidate corresponding to any of the three MCHSes over* $\mathcal{K}^2$. *Suppose that it returns* $\gamma_2^A = \{\bar{b_2}, \ldots \bar{b_5}\}$ *and* $\texttt{lb}_2 = 1$. *Then the call* $\text{Extract-AbstractCore}(F, \text{DEF}(\mathcal{AB}^2), \gamma_2^A)$ *again returns* res =*'false' and, e.g.,* $C_2 = (b_3 \vee b_4 \vee b_5)$. *The sets* $\mathcal{K}^3$ *and* $\mathcal{AB}^3$ *are set to* $\{C_1, C_2\}$ *and* $\emptyset$, *respectively. In the third iteration,* Optimize *computes* $\{b_4\}$ *as the (only) MCHS over* $\mathcal{K}^3$ *and returns* $\gamma_3^A = \{\bar{b_1}, \bar{b_2}, \bar{b_3}, \bar{b_5}\}$ *and* $\texttt{lb}_3 = 1$. *Notice that* γ_3^A *is the only* $(\mathcal{K}^3, \mathcal{AB}^3)$*-abstract candidate at this stage. The call* $\text{Extract-AbstractCore}(F, \text{DEF}(\mathcal{AB}^3), \gamma_3^A)$ *returns* res =*'false' and, e.g.* $C_3 = (b_1 \vee b_2 \vee b_3)$. *In the fourth iteration there are two possible MCHSes over* $\mathcal{K}^4 = \{C_1, C_2, C_3\}$ *that both correspond to* $(\mathcal{K}^4, \mathcal{AB}^4)$*-abstract candidates. Assume that* $\text{Optimize}(O, \mathcal{AB}^4, \mathcal{K}^4)$ *returns* $\gamma_4^A = \{\bar{b_2}, \bar{b_3}, \bar{b_5}\}$ *and* $\texttt{lb}_4 = 2$. *Then the call* $\text{Extract-AbstractCore}(F, \text{DEF}(\mathcal{AB}^4), \gamma_4^A)$ *returns* res =*'true' and the solution* $\tau = \{b_1, \bar{b_2}, \bar{b_3}, b_4, \bar{b_5}, \bar{x}\}$. *As* $O(\tau) = 2 = \texttt{lb}_4$, UniMaxSAT *terminates and returns* τ *as an optimal solution of* $\mathcal{F}$.

Abstract cores. UniMaxSAT gives IHS enhanced with abstract cores by instantiating Add-AbstractionSets to (heuristically) compute abstraction sets (in, D, out), where the inputs $in = \{x_1, \ldots, x_n\}$ are a subset of n objective variables that all have the same coefficient in O, $out = \{o_1, \ldots, o_n\}$ is a set of n new variables, and $D = \{\text{asCNF}(\sum_{x \in in} x \geq i \leftrightarrow o_i) \mid k = 1 \ldots n\}$. Informally speaking, the outputs of the added abstraction sets count the number of inputs assigned to 1 in all satisfying assignments. The instantiation of the Optimize subroutine first computes a minimum-cost solution γ to $\mathcal{K} \cup \text{DEF}(\mathcal{AB})$ that assigns all variables in $\texttt{var}(O) \cup \text{OUTS}(\mathcal{AB})$, and then returns either γ_1^A as the restriction of γ onto the objective variables $\texttt{var}(O)$ assigned to 0, or γ_2^A as the restriction of γ onto the outputs of the current abstraction sets and objective variables that are not inputs to any abstraction sets assigned to 0. More precisely, if $\text{INPUTS}(\mathcal{AB})$ is the set of all variables that are inputs to some abstraction set, then γ_2^A assigns to 0 all variables $\text{OUTS}(\mathcal{AB}) \cup (\texttt{var}(O) \setminus \text{INPUTS}(\mathcal{AB}))$ that are assigned to 0 by γ and does not assign any other variables. In both cases, Optimize also returns the cost of γ as `lb`.

The correctness of IHS enhanced with abstract cores now follows by Theorem 1 by showing that Optimize satisfies the correctness condition. This in turn follows directly from showing that both γ_1^A and γ_2^A are $(\mathcal{K}, \mathcal{AB})$–abstract candidates since then γ is a minimum-cost $(\mathcal{K}, \mathcal{AB})$–candidate, which by Proposition 1 implies that $\texttt{lb} = O(\gamma)$ is a lower bound on the optimal cost.

The fact that γ_1^A is a $(\mathcal{K}, \mathcal{AB})$–abstract candidate follows since the only extension of γ_1^A to a solution to $\mathcal{K} \cup \text{DEF}(\mathcal{AB})$ is γ. Further, γ_2^A is a $(\mathcal{K}, \mathcal{AB})$–abstract candidate since (i) γ is an extension of γ_2^A to a solution to $\mathcal{K} \cup \text{DEF}(\mathcal{AB})$ and (ii) every such extension sets equally many inputs of each abstraction set to 1, thus incurring exactly the same cost (since the inputs to each abstraction set have the same coefficient in O).

5.3 Capturing Objective-Bounding Search with UniMaxSAT

To capture objective-bounding search by UniMaxSAT, Add-AbstractionSets is instantiated to compute a single abstraction set $\text{AB} = (\texttt{var}(O), D, out)$, where all of the variables in the objective occur as inputs, there is an output o_w for every $w = 1, \ldots, W$, where W is the sum of coefficients of $O = \sum_i w_i b_i$, and where definitions are

$$D = \{\text{ASCNF}(\sum w_i b_i \geq w \leftrightarrow o_w) \mid w = 1 \ldots W\}$$

that—informally speaking—count the sum of coefficients of objective variables set to 1 by satisfying assignments, i.e., the costs of satisfying assignments.

The instantiations of Optimize for capturing different objective-bounding search algorithms follow by noting that, for any solution τ to $F \cup \text{DEF}(\{\text{AB}\})$, the cost of τ can be read from the outputs of AB, $\tau(o_k) = 0$ if and only if $O(\tau) < k$ holds for all solutions τ to $F \cup \text{DEF}(\{\text{AB}\})$. Similarly, abstract cores of this instance map to lower bounds on the optimal cost, the unit clause (o_k) is an abstract core if and only if $k < \text{OPT}(\mathcal{F})$. Thus solution-improving, UNSAT-SAT, binary, and progression-based search are all obtained by an instantiation of Optimize that returns assignments that set a single output variable to 0 and a lower bound equal to the largest index w for which (o_w) has been determined to be an abstract core. Termination occurs once the Extract-AbstractCore subroutine has determined $(o_{\text{OPT}(\mathcal{F})})$ to be an abstract core and $\{\bar{o}_{\text{OPT}(\mathcal{F})+1}\}$ a $(\mathcal{K}, \{\text{AB}\})$–abstract candidate for the set $\mathcal{K}$ of cores obtained so far.

As a concrete example, to capture solution-improving search, Optimize returns in the first iteration the assignment $(\bar{o}_W)$ and $\texttt{lb} = 0$. In subsequent iterations Optimize returns $(\bar{o}_{O(\tau)-1})$ and $\texttt{lb} = 0$ where $O(\tau)$ is the cost of the latest solution computed by Extract-AbstractCore. When Extract-AbstractCore reports unsatisfiability, an abstract core (o_w) is obtained. Correctness of solution-improving search in terms of Theorem 1 follows from the fact that the assignment $\gamma^A = (\bar{o}_{w+1})$ is a $(\{(o_w)\}, \{\text{AB}\})$–abstract candidate the extensions of which have cost $\texttt{lb} = w = \text{OPT}(\mathcal{F})$.

6. Capturing Core-Guided Search with UniMaxSAT

We turn to detailing how various modern core-guided algorithms can be viewed as instantiations of UniMaxSAT. Key to viewing any core-guided algorithm through UniMaxSAT is to view the cardinality constraints a core-guided algorithm introduces as abstraction sets, and the reformulation of the objective as an implicit computation of a $(\mathcal{K}, \mathcal{AB})$–abstract candidate for the instance, where the set $\mathcal{K}$ and $\mathcal{AB}$ are the abstract cores and abstraction sets computed so far, respectively.

6.1 Capturing Generic Properties of Core-Guided Algorithms

Compared to the IHS and objective-bounding search algorithms, significantly more variants of core-guided algorithms have been proposed, differing in the specifics of how the core-relaxation steps are performed. We will in the following identify properties of core-relaxation steps shared by all core-guided algorithms we consider. These general properties allow for a more generic proof of correctness for core-guided algorithms via viewing the algorithms as instantiations of UniMaxSAT.

Definition 5. *Consider the ith iteration of* UniMaxSAT when invoked on a MaxSAT instance $\mathcal{F} = (F, O)$. Let $\mathcal{AB}$ and $\mathcal{K}$ be the accumulated set of abstraction sets and abstract cores, respectively. We say that an instantiation of UniMaxSAT is a *core-guided instantiation* if the following properties hold.

- Extract-AbstractCore is a core-extracting SAT-solver and Add-AbstractionSets introduces feasible abstraction sets.
- Optimize maintains a reformulated objective function O^R and in each iteration returns as assumptions the assignment $\{\bar{x} \mid x \in \texttt{var}(O^R)\}$ that sets all of the variables in O^R to 0. It also returns the constant term of O^R as the lower bound on the optimal cost.
- In each iteration we have $O^R(\tau) = O(\tau)$ for any solution τ to $\text{DEF}(\mathcal{AB}) \cup \mathcal{K}$.
- Given a core, Optimize reformulates O^R in a way that increases the constant term of O^R.

We will now show that any core-guided instantiation of UniMaxSAT correctly computes optimal solutions to MaxSAT instances.

Theorem 2. *For any input MaxSAT instance $\mathcal{F}$ that has a solution, any core-guided instantiation of* UniMaxSAT *terminates and returns an optimal solution to $\mathcal{F}$.*

We prove Theorem 2 by showing that a core-guided instantiation of UniMaxSAT satisfies the assumptions of Theorem 1. The non-trivial part of the proof deals with arguing that Optimize satisfies the correctness condition (Definition 4).

In the following lemmas, we consider the ith iteration of a core-guided instantiation of UniMaxSAT when invoked on a MaxSAT instance $\mathcal{F} = (F, O)$. Let $\mathcal{AB}^i$ and $\mathcal{K}^i$ be the set of abstraction sets and abstract cores collected, respectively; O^R_i the reformulated objective maintained by Optimize; and W^{LB}_i the constant term of O^R_i. The following two observations follow directly from the definition of core-guided instantiations.

Observation 1. *W^{LB}_i is a lower bound on $\text{OPT}(\mathcal{F})$.*

Observation 2. *For any solution $\gamma \supseteq \{\bar{x} \mid x \in \texttt{var}(O^R_i)\}$ of $\text{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$, we have $O(\gamma) = W^{\text{LB}}_i$.*

What remains to be shown is that the assignment $\gamma^A_i = \{\bar{x} \mid x \in \texttt{var}(O^R_i)\}$ computed by Optimize will be a $(\mathcal{K}^{i+r}, \mathcal{AB}^{i+r})$–abstract candidate in iteration $i + r$ for some $r \geq 0$. We first establish that whenever γ^A_i can be extended to a solution to $\text{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$, any extension of γ^A_i will be minimum-cost.

Lemma 3. *Assume that the assignment $\gamma_i^A = \{\bar{x} \mid x \in \mathit{var}(O_i^R)\}$ can be extended to a solution to* $\text{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$. *Then γ_i^A is a $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate.*

Proof. Consider an extension τ of γ_i^A to a solution to $\text{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$. We show that τ is minimum-cost. Since $\tau(x) = 0$ for all $x \in \mathtt{var}(O_i^R)$, we have that $O_i^R(\tau) = W_i^{\text{LB}}$. By the definition of core-guided instantiations, this implies $O(\tau) = W_i^{\text{LB}}$. The fact that τ is minimum-cost follows from observing that $O(\delta) = O_i^R(\delta) \geq W_i^{\text{LB}}$ holds for all solutions δ to $\text{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$. □

Next we show that the assignment returned by Optimize can be extended to a solution of the abstract cores and the definitions at some future iteration.

Lemma 4. *Assume $\mathcal{F}$ has a solution. There is an $r \geq 0$ such that the assignment $\gamma_{i+r}^A = \{\bar{x} \mid x \in \mathit{var}(O_{i+r}^R)\}$ can be extended to a solution to* $\text{DEF}(\mathcal{AB}^{i+r}) \cup \mathcal{K}^{i+r}$.

Proof. Assume for contradiction that γ_k^A cannot be extended to a solution to $\text{DEF}(\mathcal{AB}^k) \cup \mathcal{K}^k$ for any $k \geq i$. Then in each iteration $k \geq i$ Extract-AbstractCore will return a core. Let $\tau*$ be an optimal solution to $\mathcal{F}$ and, for a fixed k, τ_k^E its extension to a solution of $\text{DEF}(\mathcal{AB}^k) \cup \mathcal{K}^k$. By the properties of core-guided instantiations, we have $O(\tau*) = O_k^R(\tau_k^E)$. Furthermore, the constant term W^{LB} of the reformulated objective increases in each iteration when a core is obtained. Thus eventually for some iteration k' we have $W_{k'}^{\text{LB}} = O_{k'}^R(\tau_{k'}^E) = O(\tau*)$. ($W_{k'}^{\text{LB}} > O_{k'}^R(\tau_{k'}^E)$ is not possible, since W_k^{LB} is a lower bound for $O_k^R(\tau)$ for any solution τ.) Now $O_{k'}^R(\tau_{k'}^E) = W_{k'}^{\text{LB}}$ implies $\tau_{k'}^E \supseteq \{\bar{x} \mid x \in \mathtt{var}(O_{k'}^R)\}$. Thus $\{\bar{x} \mid x \in \mathtt{var}(O_{k'}^R)\}$ can be extended to a solution to $\text{DEF}(\mathcal{AB}^{k'}) \cup \mathcal{K}^{k'}$, contradicting the initial assumption. □

A simple corollary of the Lemmas 3 and 4 is that Optimize is guaranteed to compute an abstract candidate in some future iteration.

Corollary 1. *If $\mathcal{F}$ has a solution, then* Optimize *returns a $(\mathcal{K}^{i+r}, \mathcal{AB}^{i+r})$–abstract candidate in iteration $i + r$ for some integer $r \geq 0$.*

Proof. By Lemma 3 it suffices to show that there is an $r \geq 0$ such that the set $\gamma_{i+r}^A = \{\bar{x} \mid x \in \mathtt{var}(O_{i+r}^R)\}$ can be extended to a solution to $\text{DEF}(\mathcal{AB}^{i+r}) \cup \mathcal{K}^{i+r}$. This is established by Lemma 4. □

Finally, the proof of Theorem 2 follows from the previous statements.

Proof of Theorem 2. Observations 1 and 2 together with Corollary 1 imply that Optimize satisfies the correctness condition (assumption 1 of Theorem 1). Assumptions 2 and 3 of Theorem 1 follow directly from the definition of core-guided instantiations. □

In summary, Theorem 2 provides an alternative way of establishing the correctness of a range of core-guided algorithms by arguing that a core-guided algorithm at hand falls is a core-guided instantiation.

We also establish a similar result for a specific case of core-guided instantiations which we will refer to as cardinality-based CG instantiations. The notion of cardinality-based CG instantiations of UniMaxSAT captures in particular core-guided algorithms which introduce a single cardinality constraint over each extracted core.

Definition 6. *We say that an instantiation of* UniMaxSAT is *cardinality-based CG instantiation* if the following conditions hold.

(i) Extract-AbstractCore is a core-extracting SAT solver.

(ii) Given a core C as input, Add-AbstractionSets computes an abstraction set $\textsc{ab} = (in, D, out)$ for which the following hold.

- The set of variables of D intersects the set of previous variables only on the inputs, i.e., $\texttt{var}(D) \cap \texttt{var}(F \cup \mathrm{DEF}(\mathcal{AB})) \subseteq in$.
- The number of outputs is one less than the number of variables in the core, i.e., $|out| = |C| - 1$.
- $\sum_{b \in C} \tau(b) = 1 + \sum_{o \in out} \tau(o)$ holds for any solution τ to $\{C\} \cup \{D\} \cup \mathcal{K} \cup \mathrm{DEF}(\mathcal{AB})$.

Here $\mathcal{K}$ and $\mathcal{AB}$ are the set of cores computed and abstraction sets introduced by the iteration in which C is obtained.

(iii) Optimize is instantiated as Optimize-CB which maintains a reformulated objective O^R, initialized to be the objective O of the input MaxSAT instance. In each iteration i, Optimize-CB returns the assignment $\{\bar{x} \mid x \in \texttt{var}(O^R)\}$ that sets all variables in O^R to 0 and the constant term of O^R as the lower bound. Given an abstract core C and an abstraction set $\textsc{ab} = (in, D, out)$, Optimize-CB updates O^R as follows. (1) The coefficient of each $x \in C$ is decreased by the minimum over the coefficients in O of the variables in the core, i.e., by $w^C = \min_{x \in C}\{O^R(x)\}$. (2) Each $x \in out$ is added to O^R with the coefficient w^C. (3) The variables in O^R with coefficients 0 are removed. (4) The constant term of O^R is incremented by w^C.

We establish that cardinality-based CG instantiations of UniMaxSAT are a special case of core-guided instantiations.

Theorem 3. *Any cardinality-based CG instantiation (Definition 6) is a core-guided instantiation of* UniMaxSAT (Definition 5).

Note that this implies that ny cardinality-based CG instantiation will return an optimal solution to any MaxSAT instance by Theorem 2.

Proof. The non-trivial part of the proof is to argue that in each iteration i, we have $O^R_i(\tau) = O(\tau)$ for any solution τ to $\mathrm{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$, i.e., the definitions of the abstraction sets added and cores computed by iteration i. The proof is by induction on the iteration i. The base case $= 1$ directly follows from $O^R = O$. Now assume that the statement holds in iteration $i-1$. Let O^R_i be the reformulated objective in the beginning of iteration i and assume without loss of generality that an abstract core C_i is extracted in iteration i. Let $\textsc{ab}_{i+1} = (in_{i+1}, D_{i+1}, out_{i+1})$ be the abstraction set computed by Add-AbstractionSets on input C_i. Then

$$O^R_{i+1}(\tau) \overset{*}{=} O^R_{i+1}(\tau) + \sum_{x \in C_i} w^{C_i}\tau(x) - (\sum_{x \in out_{i+1}} w^{C_i}\tau(x) + w^{C_i}) = O^R_i(\tau).$$

Here $*$ follows by the fact that Add-AbstractionSets fits the definition of a cardinality-based CG instantiation stating that $\sum_{x \in C_i} \tau(x) = 1 + \sum_{x \in out_{i+1}} \tau(x)$, which implies

$$\sum_{x \in C_i} w^{C_i} \tau(x) = w^{C_i} + \sum_{x \in out_{i+1}} w^{C_i} \tau(x).$$

□

6.2 Contrasting Core-Guided and IHS Algorithms through UniMaxSAT

Before moving on to concretely capturing the core-relaxation steps of individual core-guided algorithms, we note that the definition of core-guided instantiations of UniMaxSAT allows for observing an interesting contrast between core-guided and IHS. In particular, in contrast to IHS, the cores extracted by core-guided instantiations of UniMaxSAT always refute all possible minimum-cost solutions to the cores accumulated by that iteration.

Proposition 2. *Consider the ith iteration of a core-guided instantiation of* UniMaxSAT invoked on a MaxSAT instance $\mathcal{F} = (F, O)$. Let $\mathcal{K}^i$ and $\mathcal{AB}^i$ be the set of abstract cores and abstraction sets accumulated by the beginning of iteration i, respectively. Assume that Optimize returns a $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate and Extract-AbstractCore a core C_i. Let then γ^A be any $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate and $\tau^E \supseteq \gamma^A$ an extension of γ^A to a solution to $\text{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$ Then $\tau^E(C_i) = 0$.

Proof. We show that $\tau^E \supseteq \{\bar{x} \mid x \in \texttt{var}(O_i^R)\}$; since $C_i \subseteq \texttt{var}(O_i^R)$, this implies $\tau^E(C_i) = 0$. Since Optimize returns a $(\mathcal{K}^i, \mathcal{AB}^i)$–abstract candidate at the ith iteration, we have $O_i^R(\tau^E) = O(\tau^E) = W_i$. As $O_i^R(\tau^E) = W_i$ holds if and only if $\tau(x) = 0$ for all $x \in \texttt{var}(O_i^R)$, we conclude that $\tau^E \supseteq \{\bar{x} \mid x \in \texttt{var}(O_i^R)\}$. □

Less formally, Proposition 2 states that whenever a core-guided instantiation of UniMaxSAT extracts a core falsified by a $(\mathcal{K}, \mathcal{AB})$–abstract candidate γ of $\mathcal{F}$ where $\mathcal{K}$ and $\mathcal{AB}$ are a set of cores and abstraction sets, respectively, the core refutes not only γ but all possible $(\mathcal{K}, \mathcal{AB})$–abstract candidates, and thereby all minimum-cost $(\mathcal{K}, \mathcal{AB})$–candidates. In contrast, the following example demonstrates that cores extracted by IHS over an abstract candidate do not necessarily refute all possible abstract candidates.

Example 9. *Consider the following execution of IHS (simulated in* UniMaxSAT) on the MaxSAT instance $\mathcal{F} = (F, O)$ with $F = \{(b_1 \vee b_2 \vee b_3), (b_2 \vee b_4)\}$ and $O = b_1 + b_2 + b_3 + b_4$. Assume that Extract-AbstractCore first extracts the core $C_1 = (b_1 \vee b_2 \vee b_3)$. Further, assume that Optimize then returns $\gamma = \{\bar{b}_1, \bar{b}_2, \bar{b}_4\}$ as the $(\{C_1\}, \emptyset)$–abstract candidate[6]. Finally, assume that Extract-AbstractCore next returns the core $C_2 = (b_2 \vee b_4)$. Although $\gamma(C_2) = 0$, C_2 does not refute all $(\{C_1\}, \emptyset)$–abstract candidates. This is because $\{\bar{b}_1, \bar{b}_3, \bar{b}_4\}$ is a $(\{C_1\}, \emptyset)$–abstract candidate and $\tau^E = \{\bar{b}_1, b_2, \bar{b}_3, \bar{b}_4\}$ an extension of it to a solution to $\{C_1\}$ that satisfies C_2.

6. Recall that IHS simulated in UniMaxSAT does not add abstractions sets.

6.3 Capturing Algorithm-Specific Core Relaxations of Core-Guided Algorithms

Having established general conditions for the correctness of core-guided algorithms, we move on to detailing how the algorithm-specific core relaxations of modern core-guided algorithms can be viewed as cardinality-based CG instantiations of UniMaxSAT. Specifically, we detail this individually for OLL (Andres et al., 2012; Morgado et al., 2014), WPM3 (Ansótegui & Gabàs, 2017), MSU3 (Marques-Silva & Planes, 2007), PMRES (Narodytska & Bacchus, 2014) and K (Alviano et al., 2015) as key representatives of modern core-guided algorithms. More precisely, as the definition of cardinality-based CG instantiations prescribes how Extract-AbstractCore and Optimize are instantiated in each of these algorithms, we now detail how to instantiate Add-AbstractionSets in ways that fit Lemma 2 and Definition 6, and at the same time match the core-relaxation step of the individual algorithms.

6.3.1 OLL

The OLL algorithm (recall Section 3.2) is viewed as a cardinality-based CG instantiation of UniMaxSAT by instantiating Add-AbstractionSets to introduce, given a core C as input, the abstraction set

$$\textsc{ab}^C = (C, \{\text{asCNF}(\sum_{x \in C} x \geq i \leftrightarrow o_i) \mid 2 \leq i \leq |C|\}, \{o_2, \ldots, o_{|C|}\}),$$

matching the OLL core relaxation. For any reasonable encoding of the cardinality constraint, $\textsc{ab}^C$ clearly satisfies condition (ii) of Definition 6. Hence OLL is a cardinality-based CG instantiation of UniMaxSAT.

Example 10. *Invoke the cardinality-based CG instantiation of* UniMaxSAT *that corresponds to OLL on the MaxSAT instance* $\mathcal{F} = (F, O)$ *from Example 1.* Before the main search loop, the reformulated objective O^R of Optimize is set to O, and the sets $\mathcal{K}^1$ and $\mathcal{AB}^1$ both to $\emptyset$. In the first iteration, Optimize$(O, \mathcal{AB}^1, \mathcal{K}^1)$ returns the assignment $\gamma_1^A = \{\bar{b_1}, \ldots, \bar{b_5}\}$ containing the negation of all variables in O^R, and $\texttt{lb}_1 = 0$ corresponding to the constant term of O^R. The call Extract-AbstractCore$(F, \text{DEF}(\mathcal{AB}^1), \gamma_1^A)$ then returns res ='false' and, e.g., the core $C_1 = (b_1 \lor b_2 \lor b_3 \lor b_4 \lor b_5)$. The set of cores is then updated by letting $\mathcal{K}^2 = \{C_1\}$, after which Add-AbstractionSets$(\mathcal{K}^2)$ forms the new abstraction set

$$\textsc{ab}^1 = (\{b_1, b_2, b_3, b_4, b_5\}, \{\text{asCNF}(\sum_{i=1}^{5} b_i \geq k \leftrightarrow o_k^1) \mid 2 \leq k \leq 5\}, \{o_2^1, \ldots, o_5^1\})$$

and UniMaxSAT sets $\mathcal{AB}^2 = \{\textsc{ab}^1\}$.

In the second iteration Optimize$(O, \mathcal{AB}^2, \mathcal{K}^2)$ uses the core C_1 and abstraction set $\textsc{ab}^1$ to update O^R following Definition 6. The new reformulated objective is $O^R = 2b_3 + b_5 + o_2^1 + o_3^1 + o_4^1 + o_5^1 + 1$ and so Optimize returns $\gamma_2^A = \{\bar{b_3}, \bar{b_5}, \bar{o}_2^1, \bar{o}_3^1, \bar{o}_4^1, \bar{o}_5^1\}$ and $\texttt{lb}_2 = 1$. The call Extract-AbstractCore$(F, \text{DEF}(\mathcal{AB}^2), \gamma_2^A)$ then returns res ='false' and, e.g., the (abstract) core $C_2 = (o_2^1 \lor b_3)$. Adding C^2 to $\mathcal{K}$, i.e., letting $\mathcal{K}^3 = \{C_1, C_2\}$, results in

Add-AbstractionSets introducing the abstraction set

$$\text{AB}^2 = (\{o_2^1, b_3\}, \{\text{AS}\text{CNF}(o_2^1 + b_3 \geq 2 \leftrightarrow o_2^2)\}, \{o_2^2\})$$

.

In the third iteration Optimize again updates O^R based on AB^2 and C_2 to be $O^R = b_3 + b_5 + o_3^1 + o_4^1 + o_5^1 + o_2^2 + 2$, and returns $\gamma_3^A = \{\bar{b_3}, \bar{b_5}, \bar{o}_3^1, \bar{o}_4^1, \bar{o}_5^1, \bar{o}_2^2\}$ and $\texttt{lb}_3 = 2$. This time the call Extract-AbstractCore$(F, \text{DEF}(\mathcal{AB}^3), \gamma_3^A)$ returns res ='true' and, e.g., the solution $\tau = \{b_1, \bar{b_2}, \bar{x}, \bar{b_3}, b_4, \bar{b_5}\} \cup \{o_2^1, \bar{o}_3^1, \bar{o}_4^1, \bar{o}_5^1, \bar{o}_2^2\}$ which is then returned as an optimal solution to $\mathcal{F}$.

6.3.2 PMRES

The PMRES algorithm is viewed as a cardinality-based CG instantiation of UniMaxSAT by instantiating Add-AbstractionSets to introduce for every core $C = (b_1 \vee \cdots \vee b_n)$ the abstraction set

$$\text{AB}^C = (C, D = \{b_i \wedge (b_{i+1} \vee \cdots \vee b_n) \leftrightarrow o_i \mid 1 \leq i \leq n-1\}, \{o_1, \ldots, o_{n-1}\}).$$

In practice, the definition of each o_i is represented in CNF in the style of the standard Tseitin encoding (Tseitin, 1983; Prestwich, 2021) by taking the name d_i for the disjunction $(b_{i+1} \vee \cdots \vee b_n)$, i.e., adding clauses equivalent to $d_i \leftrightarrow (b_{i+1} \vee \cdots \vee b_n)$.[7] To establish the correctness of PMRES in terms of UniMaxSAT through Theorem 3, we argue that this instantiation of Add-AbstractionSets satisfies condition (ii) of Definition 6. Consider a solution τ to $D \wedge \{C\}$. Let $\text{IN} = \{b_{i_1}, \ldots, b_{i_m}\}$ be the set of variables occurring in C assigned to 1 by τ. We show that the set of outputs that τ assigns to 1 is $\text{OUT} = \{o_{i_1}, \ldots o_{i_{m-1}}\}$. As τ is a solution to D, it assigns all $o \in \text{OUT}$ to 1. In the opposite direction, any other output $o_k \notin \text{OUT}$ assigned to 1 by τ would result in a variable of the core $b_k \notin \text{IN}$ also being assigned to 1 by τ, which is a contradiction.

6.3.3 K

The core relaxation of the K algorithm is intuitively a combination of the PMRES and OLL relaxations. K partitions the found cores into subsets of bounded size, relaxes each partition similarly to OLL and then merges the relaxed partitions similarly to PMRES. More precisely, given a core $C = (b_1 \vee \ldots \vee b_{mk})$, K partitions C into m subsets P_i, each of size k such that $C = P_1 \vee \ldots \vee P_m$. Each partition P is relaxed with a cardinality constraint $\{\text{AS}\text{CNF}(\sum_{b \in P} b \geq l \leftrightarrow o_l^P) \mid l = 1, \ldots, k\}$ similarly as in OLL. Finally, the cardinality constraints of each partition are "merged" by adding a PMRES-style constraint of the form $\{o_1^{P_i} \wedge (o_1^{P_{i+1}} \vee \cdots \vee o_1^{P_m}) \leftrightarrow o_i^R \mid 1 \leq i \leq m-1\}$ where o_1^P is the first output of the cardinality constraint introduced for the partition P.

For formalizing K as a cardinality-based instantiation of UniMaxSAT, consider an instantiation of Add-AbstractionSets that returns a single abstraction set that combines both of these relaxations. More precisely, consider a core C of size mk partitioned into $C = P_1 \vee \ldots \vee P_m$ by m subsets $P_1, \ldots, P_m$ with $P_i = (b_1^{P_i} \vee \ldots \vee b_k^{P_i})$. We define three

7. The auxiliary d_i variables do not obstruct the main observations made here.

different types of abstraction sets:

$$\textsc{ab}^{P_i} = (P_i, \{\textsc{as}\text{CNF}(\sum_{b \in P_i} b \geq l \leftrightarrow o_l^{P_i}) \mid l = 1, \ldots, k\}, \{o_1^{P_i}, \ldots, o_k^{P_i}\})$$

for each $i = 1 \ldots m$,

$$\textsc{ab}^R = (\{o_1^{P_i}, \mid i = 1 \ldots m\}, \{o_1^{P_i} \wedge (o_1^{P_{i+1}} \vee \cdots \vee o_1^{P_m}) \leftrightarrow o_i^R \mid 1 \leq i \leq m-1\}, \{o_1^R, \ldots, o_{m-1}^R\}),$$

and

$$\textsc{ab} = \{C, \text{DEF}(\{\textsc{ab}^R\}) \cup \bigcup_{j=1}^{m} \text{DEF}(\textsc{ab}^{P_j}), \mathit{out}\},$$

where $\mathit{out} = \{o_1^R, \ldots, o_{m-1}^R\} \cup \{o_i^{P_j} \mid j = 1, \ldots, m \text{ and } i = 2, \ldots, k\}$. The intuition underlying these sets is that each $\textsc{ab}^{P_i}$ corresponds to the OLL-style relaxation of the partition P_i and the set $\textsc{ab}^R$ to the PMRES-style relaxation that combines all of them. In other words, the set $\textsc{ab}$ collects all of the relaxations into a single abstraction set that satisfies the condition of Definition 6.

The fact that an instantiation of Add-AbstractionSets which on input C returns the abstraction set $\textsc{ab}$ satisfies condition (ii) of Definition 6 follows straightforwardly—albeit being somewhat tedious to formally prove—as a consequence of the arguments we already made for PMRES and OLL. For some intuition, note that the set *out* contains $n(k-1)+(m-1) = mk-1$ output variables, which is one less than the number of variables in C. Furthermore, any solution τ to the definitions of C and definitions of $\textsc{ab}$ assigns in each $\textsc{ab}^{P_i}$ exactly the same number of inputs and outputs to 1. The output with index 1 from each P_i is then further relaxed by the PMRES-style abstraction set $\textsc{ab}^R$. This ensures that exactly one less output in *out* will be assigned to 1.

6.3.4 MSU3

The MSU3 algorithm is specific to *unweighted* MaxSAT instances, i.e., instances in which objective coefficients are all equal. On an unweighted MaxSAT instance (F, O) MSU3 maintains a single cardinality constraint

$$\textsc{as}\text{CNF}\left(\sum_{b \in \textsc{active}} b \geq \textsc{bound} \leftrightarrow o_{\textsc{bound}}\right),$$

where the set active contains the objective variables that have occurred in the so-far extracted cores. The bound bound counts the number of cores that have been extracted so far. When a new core is extracted, the objective variables in the core are added to active and the bound is incremented by one. Informally speaking, in each iteration, the SAT solver is queried for a solution that sets exactly bound objective variables to 1. The increment is due to the fact that each new core obtained implies that the optimal cost of the instance is at least one higher than bound.

To see that MSU3 is a cardinality-based CG instantiation of UniMaxSAT, we define an instantiation of Add-AbstractionSets that corresponds to the core relaxation performed by MSU3 as just-described. When the first core C_1 is extracted, Add-AbstractionSets initializes a bound $\textsc{bound}$ to 1 and introduces the abstraction set

$$\textsc{ab}_1 = (in_1, \{\textsc{as}\text{CNF}\left(\sum_{b \in in_1} b \geq \textsc{bound} + j \leftrightarrow o_j\right) \mid 1 \leq j \leq |C| - 1\}, out_1),$$

where $in_1 = C_1$ and $out_1 = \{o_1, \ldots o_{|C|-1}\}$.

In iteration $i > 1$ the obtained core C_i is first extended with all outputs in out_{i-1} of the previous abstraction set $\textsc{ab}_{i-1}$. The resulting $C_i' = C_i \cup out_{i-1}$ is clearly a core as well. Then the bound is incremented by one and a new abstraction set

$$\textsc{ab}_i = (in_i, \{\textsc{as}\text{CNF}\left(\sum_{b \in in_i} b \geq \textsc{bound} + j \leftrightarrow o_j\right) \mid 1 \leq j \leq |C_i'| - 1\}, \{o_1^i, \ldots o_{|C_i'|-1}^i\}),$$

introduced. Here $in_i = in_{i-1} \cup (C \cap \texttt{var}(O))$ contains the objective variables that are in C_i and the inputs in_{i-1} of the previous abstraction set. Notice that due to the core-extension step and the instance being unweighted, all outputs of $\textsc{ab}_{i-1}$ are removed from the reformulated objective, and as such ignored in subsequent iterations.

The core extension step is a minor technical detail required to fit the formalization of MSU3 into the definition of a cardinality-based CG instantiation of UniMaxSAT. It does not affect the algorithm in any meaningful way. In the formalization Optimize-CB returns $\gamma_1^A = \{\bar{o}_t^i \mid t = 1, \ldots, |C_i'| - 1\}$ in iteration i. An exact correspondence to the description of MSU3 given at the beginning of the section would instead return $\gamma_2^A = \{\bar{o}_1^i\}$. These two are, however, essentially equal since $\bar{o}_t^i \rightarrow \bar{o}_{t+1}^i$ holds for all t. Specifically, there is no $(\mathcal{K}, \mathcal{AB})$–candidate of the instance for current set $\mathcal{K}$ and $\mathcal{AB}$ of cores and abstraction sets, respectively, that would extend γ_2^A but not γ_1^A.

MSU3 can be seen as a special case of the WPM3 algorithm discussed next. As such a formal proof of the fact that the abstraction set $\textsc{ab}_i$ satisfies condition (ii) of Definition 6 follows from the corresponding proof for WPM3, provided in Appendix A. Informally, the proofs make use of the fact that if k inputs are assigned to 1 by a solution τ, then $k \geq \textsc{bound}$ and $k - \textsc{bound}$ outputs are assigned to 1 by τ.

6.3.5 WPM3

The WPM3 algorithm combines elements of OLL and MSU3 in that it maintains a set of several cardinality constraints, but only over objective variables. More precisely, assume that WPM3 on a MaxSAT instance (F, O) extracts the core $C = (o_{t_1}^{C_1} \vee \ldots \vee o_{t_n}^{C_n} \vee b_1 \vee \ldots \vee b_m)$. Each $o_{t_i}^{C_i}$ is an output of a cardinality constraint introduced in the relaxation of a previous core C_i and each b_i an objective variable. WPM3 relaxes C by introducing a new cardinality constraint

$$\textsc{as}\text{CNF}\left(\sum_{b \in in^C} b \geq \textsc{bound}^C \leftrightarrow o_{\textsc{bound}^C}\right),$$

where in^C contains the objective variables of C and the inputs of all cardinality constraints whose outputs appear in C. The bound $\textsc{bound}^C$ is defined recursively as

$$\textsc{bound}^C = \begin{cases} 1 & \text{if } C \text{ only contains objective variables,} \\ 1 + \sum_{i=1}^{n} \textsc{bound}^i & \text{else.} \end{cases}$$

Here $\textsc{bound}^i$ is the bound of the abstraction set introduced when relaxing a previously found core C_i. All of the cardinality constraints the outputs of which appear in C are removed from the working instance. Conceptually, the new cardinality constraint merges the inputs of the constraints whose outputs appear in C, and the bound is the number of inputs that can be inferred to 1 by the cores extracted so far[8].

The formalization of WPM3 as a cardinality-based CG instantiation of UniMaxSAT is similar to the formalization of MSU3. In terms of UniMaxSAT, a core extracted by WPM3 is of the form $C = (o_{t_1}^{\text{AB}_1} \vee \ldots \vee o_{t_n}^{\text{AB}_n} \vee b_1 \vee \ldots \vee b_m)$. Here $o_{t_i}^{\text{AB}_i}$ is an output of the abstraction set $\textsc{ab}_i$ introduced earlier. The instantiation of Add-AbstractionSets that corresponds to WPM3 first extends C to $C' \supseteq C$ by adding, for each output $o_{t_i}^{\text{AB}_i} \in C$, all of the outputs of $\textsc{ab}_i$ that are in the reformulated objective maintained by Optimize-CB. The resulting C' remains a core since all outputs of a fixed abstraction set have the same coefficient when introduced to the reformulated objective. Thus C' either contains all of the outputs of a previously introduced abstraction set or none of them. A new abstraction set

$$\textsc{ab}^{C'} = (in^{C'}, D^{C'} = \{\textsc{as}\text{CNF}\left(\sum_{b \in in_i} b \geq \textsc{bound}^{C'} + j \leftrightarrow o_j^{C'_i}\right) \mid 1 \leq j \leq |C'_i| - 1\}, out^{C'})$$

is then introduced. The inputs $in^{C'} = \{b_1, \ldots, b_m\} \cup \bigcup_{i=1}^{n} in_i$ consist of the objective variables in C and the inputs in_i of all previous abstraction sets $\textsc{ab}_i$ whose outputs appear in C'. The outputs $out^{C'} = \{o_2^{C'_i}, o_3^{C'_i}, \ldots, o_{|C'_i|}^{C'_i}\}$ are new variables. The bound $\textsc{bound}^{C'}$ of the abstraction set $\textsc{ab}^{C'}$ is defined analogously to the bounds on cardinality constraints as

$$\textsc{bound}^{C'} = \begin{cases} 1 & \text{if } C \text{ only contains objective variables,} \\ 1 + \sum_{i=1}^{n} \textsc{bound}^i & \text{else.} \end{cases}$$

Here $\textsc{bound}^i$ is the bound of the abstraction set $\textsc{ab}_i$. The definitions $D^{C'}$ ensure that the outputs count the number of new inputs in addition to $\textsc{bound}^{C'}$ assigned to 1.

A formal proof of the fact that the abstraction set $\textsc{ab}^{C'}$ satisfies condition (ii) of Definition 6 is provided in Appendix A. Informally, the result follows from three observations: (i) The definitions $D^{C'}$ ensure that any solution assigning $k \geq \textsc{bound}^{C'}$ inputs to 1 will assign $k - \textsc{bound}^{C'}$ outputs to 1; (ii) the definitions of previous abstraction sets ensure that such a solution will assign $k - \textsc{bound}^{C'}$ variables of C' to 1; and (iii) the set of accumulated cores ensures that at least $\textsc{bound}^{C'}$ inputs will be assigned to 1 in any solution.

8. As a minor technical remark, if objective variables with different coefficients appear in cores together, they may end up as inputs in different cardinality constraints. Whenever the inputs of cardinality constraints merged contain the same variable, the union operator should be understood as additive union $\uplus$ for which, e.g., $\{x, y\} \uplus \{x\} = \{x, x, y\}$.

6.4 On the Frequency of Abstract Candidates

We end this section with observations on how frequently core-guided algorithms compute abstract candidates. Recall that whenever an instantiation of UniMaxSAT computes a $(\mathcal{K}, \mathcal{AB})$–abstract candidate of the instance with the current set $\mathcal{K}$ and $\mathcal{AB}$ of cores and abstraction sets, respectively, a lower bound as high as possible given the cores extracted so-far is obtained. Intuitively, the more frequently $(\mathcal{K}, \mathcal{AB})$–abstract candidates are computed during the search, the faster in terms of iterations new lower bounds are obtained.

First, we will show that in the general case, cardinality-based CG instantiations of UniMaxSAT will not (necessarily) compute a $(\mathcal{K}, \mathcal{AB})$–abstract candidate in every iteration.

Example 11. *Consider the MaxSAT instance $\mathcal{F} = (F, O)$ with $F = \{(b_1), (b_2)\}$ and $O = b_1 + 2b_2$. Invoke a cardinality-based CG instantiation of* UniMaxSAT *on $\mathcal{F}$.* Assume that the first core extracted is $C_1 = (b_1 \vee b_2)$. Then Add-AbstractionSets introduces an abstraction set AB_1 with one output o_1. Further, Optimize-CB updates its reformulated objective to $O_1^R = b_2 + o_1$ and returns the set of assumptions $\gamma^A = \{\bar{b}_2, \bar{o}_1\}$. Since Add-AbstractionSets fulfills condition (ii) of Definition 6 and any solution to F assigns two of the variables in C_1 to 1, any solution assigns the variable o_1 to 1. Thus $C_2 = (o_1)$ is an abstract core that can be extracted in the next iteration, which results in an abstraction set AB_2 without any outputs. In the next call to Optimize-CB its objective is updated to $O_2^R = b_2$ and the set $\gamma^A = \{\bar{b}_2\}$ is returned. Now γ^A is not a $(\{C_1, C_2\}, \{\text{AB}_1, \text{AB}_2\})$–abstract candidate since all solutions to $\text{DEF}(\{\text{AB}_1\}) \cup \{C_1, C_2\}$ assign b_2 to 1.

Example 11 is stated in terms of a generic instantiation of UniMaxSAT and hence applies to all cardinality-based CG instantiations, including OLL, PMRES, K, WPM3, and MSU3. In other words, each of these algorithms may compute intermediate lower bounds that are weaker than what could be inferred based on the cores extracted so far. In contrast, it turns out that these algorithms differ in this respect when restricting to unweighted MaxSAT instances in which all objective coefficients are equal. Specifically, PMRES and WPM3 are guaranteed to always compute abstract candidates when invoked on an unweighted instance, thereby obtaining as strong lower bounds as possible, while this is not the case for OLL. To establish this, we first show that there are unweighted instances on which UniMaxSAT instantiated as OLL may not always compute abstract candidates. For some intuition on the reasons for this, in contrast to PMRES, the outputs of abstraction sets introduced by OLL can lead to situations where already-extracted cores imply other cores—irrespectively of the input instance. In contrast to WPM3, at the same time the outputs introduced by OLL need not be removed from the reformulated objective.

Example 12. *Consider an invocation of* UniMaxSAT instantiated as OLL on an unweighted MaxSAT instance $\mathcal{F}$. Assume two cores C and D are extracted, both containing three variables. This results in the introduction of the abstraction sets

$$\text{AB}_C = (C, \{\text{asCNF}(\sum_{x \in C} x \geq i \leftrightarrow o_i^C) \mid i = 2, 3\}, \{o_2^C, o_3^C\}) \text{ and}$$

$$\text{AB}_D = (D, \{\text{asCNF}(\sum_{x \in D} x \geq i \leftrightarrow o_i^D) \mid i = 2, 3\}, \{o_2^D, o_3^D\}).$$

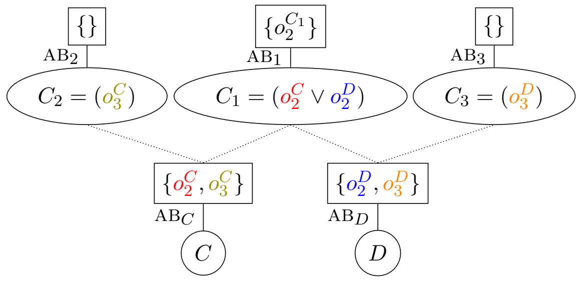


Figure 3: Structure of cores and abstraction sets of Example 12. Each pair of connected ellipse and rectangle nodes corresponds to an abstraction set, with the inputs (the extracted core) of that set appearing in the ellipse and the outputs in the rectangle. The dashed edges visualize how outputs of AB_C and AB_D appear as inputs to new abstraction sets.

Assume that $C_1 = (o_2^C \vee o_2^D)$ is the next core extracted with the corresponding abstraction set $\text{AB}_1 = (C_1, \text{ASCNF}(\sum_{x \in C_1} x \geq 2 \leftrightarrow o_2^{C_1}), \{o_2^{C_1}\})$. Finally, let the next two cores extracted be $C_2 = (o_3^C)$ and $C_3 = (o_3^D)$, resulting in the two abstraction sets $\text{AB}_2 = (\{(o_3^C\}, \emptyset, \emptyset)$ and $\text{AB}_3 = (\{o_3^D\}, \emptyset, \emptyset)$. Figure 3 illustrates the abstraction sets added after extracting these cores. In the subsequent iteration, the partial assignment δ^E computed by Optimize-CB will include $\bar{o}_2^{C_1}$. However, then δ^E is *not* a $(\mathcal{K}, \mathcal{AB})$–abstract candidate, where $\mathcal{K} = \{C, D, C_1, C_2, C_3\}$ and $\mathcal{AB} = \{\text{AB}_C, \text{AB}_D, \text{AB}_1, \text{AB}_2, \text{AB}_3\}$. This is because there is no extension of δ^E to a solution to $\mathcal{K} \cup \text{DEF}(\mathcal{AB})$. To see this, note that any solution τ to $\mathcal{K} \cup \text{DEF}(\mathcal{AB})$ has to assign $\tau(o_3^C) = \tau(o_3^D) = 1$ and therefore also $\tau(o_2^C) = \tau(o_2^D) = 1$. By the definitions in AB_1 this implies $\tau(o_2^{C_1}) = 1$.

In contrast to OLL, when invoked on an unweighted MaxSAT instance, both PMRES and WPM3 are guaranteed to compute abstract candidates in each iteration, and thus both algorithms are guaranteed to obtain as strong lower bounds as possible. This is formalized in the following proposition, the proof of which is provided in Appendix A.

Proposition 3. *Invoke* UniMaxSAT instantiated as PMRES or WPM3 on an unweighted MaxSAT instance $\mathcal{F} = (F, O)$. *In the* i*th iteration of search, let* O_i^R *be the reformulated objective maintained by* Optimize-CB. *Let also* $\mathcal{K}^i$ *be the set of cores, and* $\mathcal{AB}^i$ *the collection of abstraction sets collected so far. The partial assignment* $\gamma_i^A = \{\bar{x} \mid x \in \texttt{var}(O_i^R)\}$ *returned by* Optimize-CB *is a* $(\mathcal{K}^i, \mathcal{AB}^i)$*–abstract candidate.*

The results of Proposition 3 and Example 12 demonstrate the potential of UniMaxSAT for analyzing existing SAT-based MaxSAT solving algorithms.

7. UniMaxSAT as Basis for New Algorithmic Variants

We emphasize that the main contributions of this work are the formal UniMaxSAT framework and the unifying proofs of correctness for established SAT-based MaxSAT solving approaches the framework yields. However, beyond the already-presented main contributions, we more shortly point out that the framework can also be used for obtaining new algorithmic variants of the SAT-based MaxSAT solving approaches and thereby to provide proofs of correctness for such variants. To illustrate this further potential of the UniMaxSAT framework, we describe a novel variant AbstCG of core-guided search as an instantiation of UniMaxSAT. While AbstCG could be designed on its own, viewing it as an instantiation of UniMaxSAT immediately implies that this new algorithmic variant is correct, highlighting the usefulness of UniMaxSAT in developing new correct MaxSAT algorithms.

For AbstCG, similarly as for other core-guided algorithms, Extract-AbstractCore is a core-extracting SAT solver, Add-AbstractionSets introduces abstraction sets for each core and Optimize maintains a reformulated objective O^R and always returns $\{\bar{x} \mid x \in \texttt{var}(O^R)\}$ as assumptions. Given a core C consisting of variables that have m different coefficients in O^R, the instantiation of Add-AbstractionSets in AbstCG first partitions C into m disjoint sets $C = G_1 \vee \ldots \vee G_m$ so that all variables in the same set G_i have the same coefficients, with the sets G_i indexed by decreasing coefficients. Starting from G_1 (corresponding to the largest coefficient in O^R), AbstCG introduces for each G_i an abstraction set

$$\textsc{ab}_i = (in_i, \{\textsc{as}\text{CNF}(\sum_{x \in in_i} x \geq j \leftrightarrow o_{i,j}) \mid 1 \leq j \leq |in_i|\}, out_i).$$

The inputs $in_i = G_i \cup out_{i-1}$ consist of the variables in G_i and the outputs of $\textsc{ab}_{i-1}$. Since C is an abstract core, at least one of its variables is assigned to 1 in any solution. Hence the first output $o_{m,1}$ of the last abstraction set is not included in out_m.

The Optimize instantiation in AbstCG updates the reformulated objective O^R by processing each abstraction set $\textsc{ab}_i = (in_i, D_i, out_i)$ in order, starting from $i = 1$. The coefficient of each $x \in in_i$ is decreased by $w_i = \min(\{O^R(x) \mid x \in in_i\})$ and each output $x \in out_i$ is included in O^R with coefficient w_i. After processing each $\textsc{ab}_i$, a constant w_m is added to O^R.

Example 13. *Invoke* AbstCG on the MaxSAT instance $\mathcal{F} = (F, O)$ with

$$F = \{(b_1 \vee b_2 \vee b_3 \vee b_4 \vee b_5)\} \cup \textsc{as}\text{CNF}(\sum_{2 \leq i \leq 5} b_i \geq 3)\}$$

and $O = b_1 + 5b_2 + 5b_3 + 5b_4 + 5b_5$. Assume that the first core is $C = (b_1 \vee b_2 \vee b_3 \vee b_4 \vee b_5)$. Core relaxation divides the variables of C into $G_1 = \{b_2, b_3, b_4, b_5\}$ and $G_2 = \{b_1\}$. The abstraction set over G_1 has four output variables $o_{1,j}$ with $1 \leq j \leq 4$, defined by $\textsc{as}\text{CNF}(\sum_{x \in G_1} x \geq j \leftrightarrow o_{1,j})$. The abstraction set over G_2 has five output variables $o_{2,j}$ with $1 \leq j \leq 5$, defined by $\textsc{as}\text{CNF}(\sum_{x \in G_2 \cup out_1} x \geq j \leftrightarrow o_{2,j})$. After reformulation, the objective O^R is $O^R = 4o_{1,1} + 4o_{1,2} + 4o_{1,3} + 4o_{1,4} + o_{2,2} + o_{2,3} + o_{2,4} + o_{2,5}$. The assumptions γ^A for the next call to Extract-AbstractCore consist of the negations of variables in O^R, $\gamma^A = \{\bar{o}_{1,1}, \bar{o}_{1,2}, \bar{o}_{1,3}, \bar{o}_{1,4}, \bar{o}_{2,2}, \bar{o}_{2,3}, \bar{o}_{2,4}, \bar{o}_{2,5}\}$. Assume that next, the unit cores $(o_{1,1})$, $(o_{1,2})$, $(o_{1,3})$, $(o_{2,2})$, $(o_{2,3})$ are extracted. After this, the assumptions γ^A for the next call to Extract-AbstractCore are $\gamma^A = \{\bar{o}_{1,4}, \bar{o}_{2,4}, \bar{o}_{2,5}\}$. Then Extract-AbstractCore returns, for example, the solution $\tau = \{\bar{b}_1, b_2, b_3, b_4, \bar{b}_5\}$ as an optimal solution to $\mathcal{F}$.

For an informal connection between AbstCG and OLL, note that when the variables C all have the same coefficient in O^R, the reformulation performed by AbstCG is the same

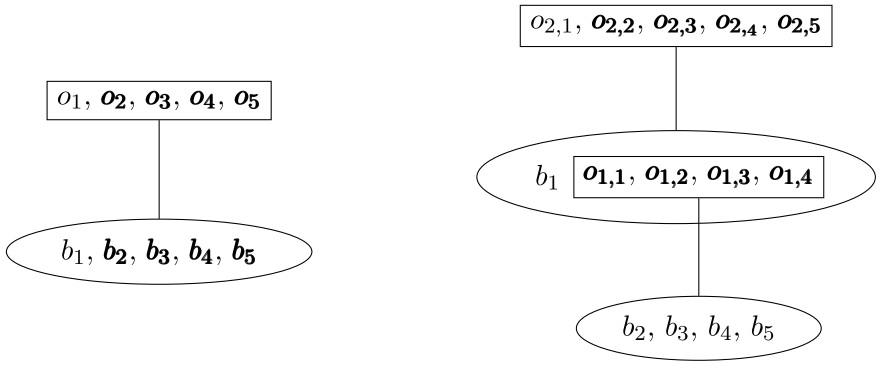

o_1, $\boldsymbol{o_2}$, $\boldsymbol{o_3}$, $\boldsymbol{o_4}$, $\boldsymbol{o_5}$
b_1, $\boldsymbol{b_2}$, $\boldsymbol{b_3}$, $\boldsymbol{b_4}$, $\boldsymbol{b_5}$
$o_{2,1}$, $\boldsymbol{o_{2,2}}$, $\boldsymbol{o_{2,3}}$, $\boldsymbol{o_{2,4}}$, $\boldsymbol{o_{2,5}}$
b_1
$\boldsymbol{o_{1,1}}$, $\boldsymbol{o_{1,2}}$, $\boldsymbol{o_{1,3}}$, $\boldsymbol{o_{1,4}}$
b_2, b_3, b_4, b_5

Figure 4: Left: An abstraction set introduced by OLL (left) and AbstCG (right) when relaxing core $C_1 = \{b_1, b_2, b_3, b_4, b_5\}$ from Example 13. Connected ellipse and rectangle nodes correspond to an abstraction set, with the inputs of a set mentioned within the ellipse and the outputs within the rectangle. Variables with a positive coefficient in the reformulated objective O^R after the reformulation are highlighted in boldface.

as the one performed by OLL. The two algorithms differ in how cores with more than one distinct coefficient are processed. To illustrate this difference, consider the abstraction sets introduced by AbstCG and OLL when relaxing the core $C_1 = (b_1 \lor b_2 \lor b_3 \lor b_4 \lor b_5)$ from Example 13. Figure 4 illustrates the abstraction sets introduced by OLL (left) and AbstCG (right). In the figure, each pair of connected ellipse and rectangle nodes corresponds to an abstraction set. The inputs of each abstraction set are represented by an ellipse and the outputs by a rectangle. Additionally, the variables with a positive coefficient in the reformulated objectives—$O^R = 4b_2 + 4b_3 + 4b_4 + 4b_5 + o_2 + o_3 + o_4 + o_5$ for OLL and $O^R = 4o_{1,1} + 4o_{1,2} + 4o_{1,3} + 4o_{1,4} + o_{2,2} + o_{2,3} + o_{2,4} + o_{2,5}$ for AbstCG—are highlighted in boldface. We observe that OLL keeps most of the variables in C_1 in the reformulated objective. In contrast, AbstCG removes all of the variables in the core from the objective, and instead introduces new variables that keep track of how many of the variables in the set $\{b_2, b_3, b_4, b_5\}$ are assigned to 1 in subsequent iterations.

We establish the correctness of AbstCG as a core-guided instantiation of UniMaxSAT via Theorem 2. The non-trivial part is to argue that the reformulated objective O^R maintained by AbstCG preserves the costs of all solutions.

Proposition 4. *Assume that the* AbstCG *instantiation of* UniMaxSAT *is invoked on a MaxSAT instance. Let $O_i^R(\tau)$ be the reformulated objective in iteration i and $\mathcal{AB}^i$ and $\mathcal{K}^i$ the set of abstraction sets introduced and cores obtained by iteration i, respectively. Let τ be a solution to $\text{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$. Then $O_i^R(\tau) = O(\tau)$.*

Proof. By induction on the iteration i. The base case $i = 1$ follows from $O^R = O$. Assume then that the statement holds in iteration $i - 1$. Let O_i^R be the reformulated objective in iteration i (i.e., the reformulated objective in the beginning of iteration i). Assume without loss of generality that an abstract core C_i is extracted in iteration i and partitioned into m subsets $G_1, \ldots, G_m$. Let $\text{AB}_j^i = (in_j^i, D_j^i, out_j^i)$ for $j = 1, 2, \ldots, m$ be the abstraction sets computed by Add-AbstractionSets on input C_i and let w_j^i be the minimum weights of each G_j for $j = 1, 2, \ldots, m$. Then

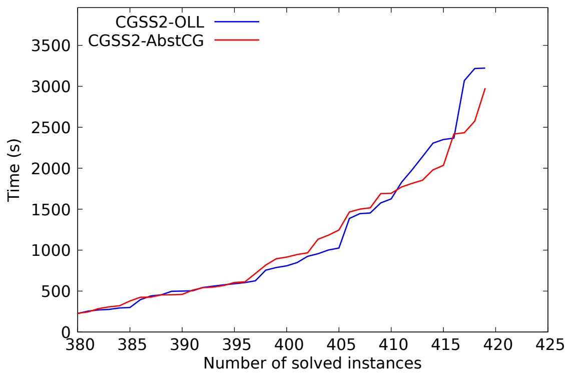
$$O^R_{i+1}(\tau) \overset{*}{=} O^R_{i+1}(\tau) + \sum_{j=1}^{m}\Big(\sum_{b \in in^i_j} w^i_j \tau(b) - \sum_{o \in out^i_j} w^i_j \tau(o)\Big) - w^i_m = O^R_i(\tau).$$

Step $*$ follows from the observation that $\sum_{b \in in^i_j} \tau(b) = \sum_{o \in out^i_j} \tau(o)$ for all $j < m$, which implies $\sum_{b \in in^i_j} w^i_j \tau(b) - \sum_{o \in out^i_j} w^i_j \tau(o) = 0$. Furthermore, since C_i is a core, we have $\sum_{b \in in^i_m} \tau(b) \geq 1$. Recalling that $o^i_{m,1} \notin out^i_m$ we have $\sum_{b \in in^i_m} w^i_m \tau(b) - \sum_{o \in out^i_m} w^i_m \tau(o) - w^i_m = 0$, and can conclude that $\sum_{b \in in^i_m} \tau(b) = \sum_{o \in out^i_m} \tau(o) + 1$. □

While the main focus of this work is on the UniMaxSAT framework, we developed a prototype implementation of AbstCG on top of CGSS2, a state-of-the-art C++ implementation of OLL (Ihalainen, 2022). This implementation of AbstCG is available online at `https://bitbucket.org/coreo-group/cgss2/` as a command line option of CGSS2. When a new core is extracted, the prototype implementation dynamically selects between relaxing it in the style of OLL and relaxing it in the style of AbstCG, always choosing the relaxation that results in fewer clauses added. Intuitively, this choice aims to balance the benefits of relaxing cores in the style of AbstCG (such as removing all literals in the core from the reformulated objective) with the potentially smaller size of the core-relaxation constraint of OLL. We empirically compare the runtimes of CGSS2-AbstCG (our prototype implementation of AbstCG) to those of CGSS2-OLL, i.e., the base CGSS2 implementation. We emphasize that the goal of these experiments is to demonstrate that UniMaxSAT allows for novel algorithmic instantiations that can be used to obtain practical solvers competitive with the state-of-the-art. As benchmarks, we used the 558 instances weighted instances[9] from the exact track of MaxSAT Evaluation 2023 (`https://maxsat-evaluations.github.io/2023/`). The experiments were run using 2.6-GHz AMD EPYC 7H12 processors under a per-instance 3600-second time and 16-GB memory limit.

Figure 5 (left) provides a runtime comparison for the two solvers with all additional heuristics implemented in CGSS2 enabled. We observe that CGSS2-AbstCG is competitive with base CGSS2 (referred to as CGSS2-OLL in the following). CGSS2-AbstCG exhibits somewhat faster runtimes on instances that take around 2000 seconds to solve, CGSS2-AbstCG solving 415 instances within 2100 seconds compared to CGSS2-OLL solving 412. Both solvers solve 419 instances within the per-instance time limit. For a more fine-grained view, we also ran both solvers with the weight-aware core extraction (WCE) (Berg & Järvisalo, 2017) and structure sharing (SS) (Ihalainen et al., 2021) techniques disabled. We note that with these two techniques CGSS2 delays the core-relaxation steps and heuristically attempts to order the literals in the individual core relaxations in a way that allows reusing constraints between multiple core relaxations. Figure 5 (right) provides a runtime comparison for the two solvers with WCE and SS disabled. Interestingly, disabling these heuristics seems to degrade the performance of CGSS2-OLL more than of CGSS2-AbstCG,

9. Note that when all literals in a core to be relaxed have the same coefficients, the reformulation used by AbstCG is exactly the same as the one used by OLL. We hence excluded the unweighted benchmarks from the experiments.



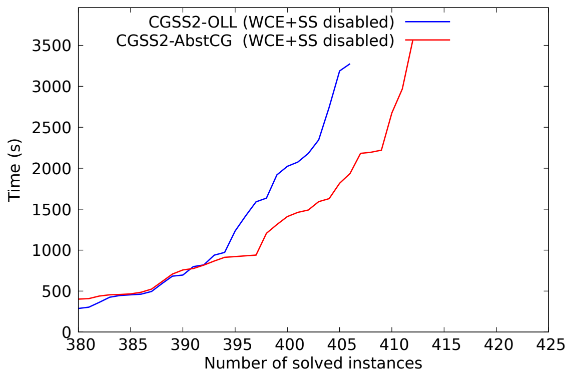


Figure 5: Left: runtime comparison of OLL and AbstCG with WCE and SS enabled. Right: Runtime comparison of OLL and AbstCG with WCE and SS disabled.

with the former solving 406 instances and the latter 412 within the time limit. The results suggest that a solver using purely OLL benefits more from WCE and SS than a solver using the AbstCGcore relaxation. Overall, as a proof of concept, AbstCG appears an interesting example of a new instantiation of the general framework. Beyond the main focus of this work, we note that a more fine-grained integration of the AbstCG relaxation within CGSS2 could lead to further improvements in solver runtimes.

8. Conclusions

Building on the recently-proposed notion of abstract cores, we developed a general algorithmic framework that captures in a unifying way the computations performed by SAT-based MaxSAT solvers. The framework covers the three most popular modern practical algorithmic variants to MaxSAT, namely, the core-guided, the implicit hitting set (IHS), and the objective-bounding approaches, variants of which are today implemented in various publicly-available MaxSAT solvers. The framework provides a uniform way of proving the correctness of the current and potential forthcoming algorithms of the three approaches, as well as algorithms combining techniques of the different approaches. To illustrate this, we formally detailed how the framework captures various existing instantiations of the approaches. The framework also suggests novel algorithmic variants through different instantiations; we detailed one such instantiation and showed as a proof of concept that it resulted in a potentially interesting solver variant for MaxSAT from a practical perspective.

While the framework developed in this work captures quite generally the current mainstream approaches to SAT-based MaxSAT solving, as a potential direction for further study it would be interesting to develop further understanding on the distinguishing features of branch-and-bound based MaxSAT solvers and their connnections to our framework. Furthermore, while our discussion was grounded in MaxSAT, we note that the framework could be extended to cover related constraint optimization paradigms—such as pseudo-Boolean optimization (Devriendt et al., 2021; Smirnov et al., 2021, 2022), finite-domain constraint optimization (Delisle & Bacchus, 2013; Gange et al., 2020), and answer set programming (Andres et al., 2012; Saikko et al., 2018; Alviano & Dodaro, 2020)—for which core-guided and IHS-style solvers have been developed. Such extensions would seem reasonable, since—as implemented in the already-proposed solvers for PBO, COP and ASP—the core-guided and IHS approaches are essentially agnostic to the constraint language at hand, assuming that a suitable decision oracle for core extraction exists. In particular, the UniMaxSAT framework could analogously be presented on the level of these more high-level constraint languages instead of the propositional representation focused on in this article. Studying new ways of instantiating the framework towards developing novel practical SAT-based algorithms for MaxSAT and related constraint optimization paradigms is also an interesting direction for further work.

Acknowledgments

This work was financially supported by University of Helsinki Doctoral Programme in Computer Science DoCS and Research Council of Finland (grants 342145 and 356046). The authors thank the Finnish Computing Competence Infrastructure (FCCI) for computational and data storage resources.

Appendix A. Proofs

A.1 Correctness of WPM3 via UniMaxSAT

The following proposition establishes that the instantiation of Add-AbstractionSets that simulates core-relaxation steps performed by WPM3 (as detailed in Section 6.3) satisfies condition (ii) of a cardinality-based CG instantiation (Definition 6).

In the following, assume that WPM3 instantiated in UniMaxSAT is invoked on a MaxSAT instance. Fix an iteration and let $\mathcal{AB}$ and $\mathcal{K}$ be the set of abstraction sets introduced and cores accumulated at the beginning of the iteration. Assume that a core $C = (b_1 \lor, \ldots, \lor b_m \lor o_{t_1}^{\text{AB}_1} \lor o_{t_n}^{\text{AB}_n})$, where each b_i is an objective variable and $o_{t_k}^{\text{AB}_k}$ is an output of a previously-introduced abstraction set AB_k, is extracted. As described in Section 6.3, the core C is extended to

$$C' = (b_1 \lor \ldots \lor b_m) \lor \bigvee_{i=1}^{n} (o_1^{\text{AB}_i} \lor \ldots \lor o_{|out_i|-1}^{\text{AB}_i})$$

by adding all outputs of the abstractions sets appearing in C. The new abstraction set

$$\text{AB}^{C'} = (in^{C'} = \{b_1, \ldots, b_m\} \cup \bigcup_{i=1}^{n} in_i, D^{C'}, out^{C'} = \{o_1^{C'}, \ldots, o_{|C'|-1}^{C'}\})$$

is introduced. Here in_i is the input of the abstraction set AB_i introduced previously,

$$D^{C'} = \{\text{asCNF}\left(\sum_{b \in in_i} b \geq \text{BOUND}^{C'} + j \leftrightarrow o_j^{C_i'}\right) \mid 1 \leq j \leq |C_i'| - 1\}$$

and

$$\text{BOUND}^{C'} = \begin{cases} 1 & \text{if } C \text{ only contains objective variables,} \\ 1 + \sum_{i=1}^{n} \text{BOUND}^{i} & \text{else.} \end{cases}$$

We show that C' fulfills condition (ii) of Definition 6. The non-trivial part to show is that the number of variables in C' assigned to 1 by any satisfying assignment to the cores and definitions found so far is one more than the number of outputs of the abstraction set introduced in its relaxation. This is formalized in the following proposition.

Proposition 5. *Let τ be an assignment satisfying to* $\mathrm{DEF}(\mathcal{AB}) \cup \mathcal{K} \cup \{D^{C'}\} \cup C'$. *We have that*

$$\sum_{b \in C'} \tau(b) = \sum_{o \in out^{C'}} \tau(o) + 1.$$

The proof of Proposition 5 employes the following lemma.

Lemma 5. *Let τ be an assignment satisfying to* $\mathrm{DEF}(\mathcal{AB}) \cup \mathcal{K} \cup \{D^{C'}\} \cup C'$. *We have that*

$$\sum_{b \in in^{C'}} \tau(b) \geq \textsc{bound}^{C'}.$$

Proof. By induction on the added abstraction sets.

Base Case. C' only contains objective variables. Now $\textsc{bound}^{C'} = 1$ and the statement follows by noting that $C' = C = in^{C'}$ and that τ satisfies C'.

Induction step. Assume that C' contains outputs of previous abstraction sets (for which the statement holds by the induction). Then

$$\sum_{b \in in^{C'}} \tau(b) = \sum_{i=1}^{m} \tau(b) + \sum_{j=1}^{n} \sum_{b \in in_j} \tau(b) \overset{*}{\geq} \sum_{i=1}^{m} \tau(b) + \sum_{j=1}^{n} \textsc{bound}^{j} = \sum_{i=1}^{m} \tau(b) + \textsc{bound}^{C'} - 1.$$

The induction step is marked with $*$. If $\sum_{i=1}^{m} \tau(b) \geq 1$, we are done. Otherwise, assume that $\sum_{i=1}^{m} \tau(b) = 0$. We establish that there exists an abstraction set $\textsc{ab}_k$ with more than $\textsc{bound}^{k}$ inputs set to true by τ. Since $\tau(C') = 1$, we can fix k to be an index for which $\tau(o_1^{\mathrm{AB}_k}) = 1$. Now, if the core C_k that prompted the addition of $\textsc{ab}_k$ does not contain any outputs of previous abstraction sets, we are done, since then $\textsc{bound}^{k} = 1$ and by the definitions of abstraction sets (satisfied by τ) $\tau(o_1^{\mathrm{AB}_k}) = 1$ implies at least two inputs are assigned to 1. Otherwise, we recurse. Since $\tau(C_k) = 1$, there is another abstraction set for which τ assigns at least two inputs to true. At some point, the core corresponding to the found abstraction set will not contain any output literals, at which point the recursion is guaranteed to end. □

Corollary 2.

$$\sum_{b \in in^{C'}} \tau(b) - \textsc{bound}^{C'} = \sum_{o \in out^{C'}} \tau(o).$$

Proof. Follows by Lemma 5 and the fact that τ satisfies $D^{C'}$. □

Proof of Proposition 5. If C includes no outputs from previous abstraction sets, the statement follows by noting that $C = C' = in^{C'}$. Otherwise we have

$$\sum_{b\in C'}\tau(b) = \sum_{i=1}^{m}\tau(b_i) + \sum_{i=1}^{n}\sum_{o\in out_i}\tau(o) \overset{*}{=} \sum_{i=1}^{m}\tau(b_i) + \sum_{i=1}^{n}\left(\sum_{b\in in_i}\tau(b) - \textsc{bound}^i\right)$$
$$= \sum_{b\in in^{C'}}\tau(b) - \sum_{i=1}^{n}\textsc{bound}^i = \sum_{b\in in^{C'}}\tau(b) - (\textsc{bound}^{C'} - 1) \overset{*}{=} \sum_{o\in out^{C'}}\tau(o) + 1,$$

where the * steps follow by Corollary 2. □

A.2 PMRES and WPM3 Compute Abstract Candidates on Unweighted Instances

We prove Proposition 3 separately for PMRES and WPM3. For the following, let $\mathcal{AB}^i$ and $\mathcal{K}^i$ be the set of abstraction sets and abstract cores extracted so far, respectively. Both proofs make use of the fact that, by Lemma 3, it suffices to show that there exists a solution $\tau_i \supseteq \gamma_i^A = \{\bar{x} \mid x \in \texttt{var}(O_i^R)\}$ to $\mathrm{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$ that extends γ_i^A.

Proof of Proposition 3 for PMRES. We prove the existence of such a τ_i by induction on i. More precisely, for every i we construct an assignment τ_i^o to the objective variables that extends to such a τ_i. As a tool for the induction, we use a function β_i that maps each variable $x \in \texttt{var}(\gamma_i^A)$ to a unique objective variable $\beta_i(x) = y \in \texttt{var}(O)$. The mapping has the following properties: (1) $\tau_i^o(y) = 0$ and (2) for every $x \in \texttt{var}(\gamma_i^A)$, any solution δ to $\mathrm{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$ that agrees with τ_i on every variable in $\texttt{var}(O)$ except $\beta_i(x)$ also agrees with τ_i on every variable in $\texttt{var}(O_i^R)$ except for x.

Base case $(i = 1)$**:** We have $\mathcal{K}^1 \cup \mathrm{DEF}(\mathcal{AB}^1) = \emptyset$ so $\tau_1^o = \gamma_1^A = \{\bar{x} \mid x \in \texttt{var}(O)\}$ and $\beta_1(x) = x$ for each $x \in \texttt{var}(O)$.

Induction Step: Let the core obtained in iteration i be $C_i = (x_1 \vee \cdots \vee x_n)$ and the abstraction set introduced be $\textsc{ab}_i = (C_i, \{x_j \wedge (x_{j+1} \vee \cdots \vee x_n) \leftrightarrow o_j^i \mid 1 \leq j \leq n-1\}, \{o_1^i, \ldots, o_{n-1}^i\})$. By the induction assumption, there exists an assignment τ_i^o to the objective variables that extends to a solution $\tau_i \supseteq \gamma_i^A$ that satisfies $\mathrm{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$ and a function β_i that satisfies properties (1) and (2) outlined in the beginning of this proof.

Let then $b = \beta_i(x_n)$ and $\tau_{i+1}^o = \tau_i^o \cup \{b\} \setminus \{\bar{b}\}$ and consider the (unique) extension of $\tau_{i+1} \supset \tau_{i+1}^o$ to a solution of $\mathrm{DEF}(\mathcal{AB}^{i+1}) = \mathrm{DEF}(\mathcal{AB}^i \cup \{\textsc{ab}_i\})$. By property (2) of β_i, τ_{i+1} agrees with τ_i on all variables in $\texttt{var}(O_i^R)$ except for x_n that τ_i assigns to 0 and τ_{i+1} to 1. As τ_i is a solution to $\mathcal{K}^i$, and τ_{i+1} assigns $\tau_{i+1}(x_n) = 1$, it follows that τ_{i+1} is a solution to $\mathcal{K}^{i+1} = \mathcal{K}^i \cup \{C_i\}$. Finally, let $\beta_{i+1}(x) = \beta_i(x)$ for all x except for every $x_i \in C_i$ (as these variables will no longer be part of the assumptions in subsequent iterations) and $\beta_{i+1}(o_j^i) = \beta_i(x_j)$ for $o_j^i \in \{o_1^i, \ldots, o_{n-1}^i\}$. Then the mapping β_{i+1} has the properties (1) and (2) outlined in the beginning of the proof. □

Proof of Proposition 3 for WPM3. We show the existence of a solution $\tau_i \supset \gamma_i^A = \{\bar{x} \mid x \in \texttt{var}(O_i^R)\}$ to $\mathcal{K}^i \cup \mathrm{DEF}(\mathcal{AB}^i)$. The proof is by induction on the iteration i.

Base case $(i = 1)$**:** Immediate by $\mathcal{K}^1 \cup \mathrm{DEF}(\mathcal{AB}^1) = \emptyset$ and $O = O^R$.

Induction step: Assume that the (extended) core

$$C_i = (b_1 \vee \ldots \vee b_m) \vee \bigvee_{i=1}^{n} (o_1^{\text{AB}_i} \vee \ldots \vee o_{|out_i|-1}^{\text{AB}_i})$$

is extracted on iteration i, and the new abstraction set $\text{AB}_i = (in_i, D_i, out_i)$ introduced as detailed in Section 6.3.5. By the induction assumption, there exists a solution $\tau_i \supset \gamma_i^A = \{\bar{x} \mid x \in \mathtt{var}(O_i^R)\}$ to $\mathcal{K}^i \cup \text{DEF}(\mathcal{AB}^i)$.

By the properties of the Extract-AbstractCore subroutine, τ_i falsifies C_i. Let τ_i^o be the restriction of τ_i onto the objective variables and select any $b \in in_i$ for which $\tau_i^o(b) = 0$, at least one such b exists as $\tau_i(C_i) = 0$. Finally, let τ_{i+1}^o be the assignment to the objective variables that agrees with τ_i^o on all variables except b and consider the (unique) extension $\tau_{i+1} \supset \tau_{i+1}^o$ of τ_{i+1}^o to a solution of $\text{DEF}(\mathcal{AB}^{i+1}) = \text{DEF}(\mathcal{AB}^i \cup \{\text{AB}_i\})$. The proposition follows from showing two things: (a) τ_{i+1} satisfies $\mathcal{K}^{i+1} = \mathcal{K}^{i-1} \cup \{C_i\}$, and (b) $\tau_i \supset \gamma_{i+1}^A = \{\bar{x} \mid x \in \mathtt{var}(O_{i+1}^R)\}$.

(a) For the non-trivial case, assume $b \notin C_i$. Then b is an input to some abstraction set AB_j for which $o_1^{\text{AB}_j} \in C$. Then τ_{i+1} assigns $\text{BOUND}^j + 1$ inputs of AB_j to 1. This follows by a recursive argument similar to the one made in the proof of Lemma 5. By the construction of the abstraction sets–and by recursing if needed–we can assume that the core C_j that prompted the addition of AB_j only contains objective variables. As $\tau_i(C_j) = 1$ we have that τ_i assigns $\tau_i(b_o) = 1$ for at least one variable $b_o \in C_j$ to 1. As $b \in C_j$, $\tau_i(b) = 0$ and τ_{i+1} agrees with τ_i on all objective variables except b, we have that τ_{i+1} assigns at least 2 variables in C_j to 1. As $\text{BOUND}^j = 1$ τ_{i+1} also assigns $\tau_i(o_1^{\text{AB}_j}) = 1$ and satisfies C_i. Furthermore, by the definitions of abstraction sets, τ_{i+1} assigns at least the same variables to 1 as τ_i, thus it clearly also satisfies all other cores. We conclude that τ_{i+1} is a solution to $\mathcal{K}^{i+1}$.

(b) Follows from the properties of core-guided instantiations. As τ_{i+1} assigns exactly one more objective variable to 1 than τ, we have that $O(\tau_{i+1}) = O(\tau_i) + 1$ which implies $O_{i+1}^R(\tau_{i+1}) = O_i^R(\tau_{i+1}) = O_i^R(\tau_i) + 1$. By the induction assumption $O(\tau_i)$ is equal to the constant term W_i^{LB} of O_i as τ_i assigns every variable in $\mathtt{var}(O_i^R)$ to 0. Since $W_i^{\text{LB}} + 1 = W_{i+1}^{\text{LB}}$ it follows that $O_{i+1}^R(\tau_{i+1}) = W_{i+1}^{\text{LB}}$ so it assigns every variable in τ_{i+1} to 0. □

References

Abío, I., Nieuwenhuis, R., Oliveras, A., & Rodríguez-Carbonell, E. (2013). A parametric approach for smaller and better encodings of cardinality constraints. In Schulte, C. (Ed.), *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, Vol. 8124 of *Lecture Notes in Computer Science*, pp. 80–96. Springer.

Abramé, A., & Habet, D. (2014). Ahmaxsat: Description and evaluation of a branch and bound Max-SAT solver. *J. Satisf. Boolean Model. Comput.*, *9*(1), 89–128.

Abramé, A., & Habet, D. (2016). Learning nobetter clauses in Max-SAT branch and bound solvers. In *28th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2016, San Jose, CA, USA, November 6-8, 2016*, pp. 452–459. IEEE Computer Society.

Alviano, M., & Dodaro, C. (2020). Unsatisfiable core analysis and aggregates for optimum stable model search. *Fundamenta Informaticae*, *176*(3-4), 271–297.

Alviano, M., Dodaro, C., & Ricca, F. (2015). A MaxSAT algorithm using cardinality constraints of bounded size. In Yang, Q., & Wooldridge, M. J. (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 2677–2683. AAAI Press.

Andres, B., Kaufmann, B., Matheis, O., & Schaub, T. (2012). Unsatisfiability-based optimization in clasp. In Dovier, A., & Costa, V. S. (Eds.), *Technical Communications of the 28th International Conference on Logic Programming, ICLP 2012, September 4-8, 2012, Budapest, Hungary*, Vol. 17 of *LIPIcs*, pp. 211–221. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Ansótegui, C., Bonet, M. L., & Levy, J. (2013). SAT-based MaxSAT algorithms. *Artificial Intelligence*, *196*, 77–105.

Ansótegui, C., & Gabàs, J. (2017). WPM3: An (in)complete algorithm for weighted partial MaxSAT. *Artificial Intelligence*, *250*, 37–57.

Ansótegui, C., Gabàs, J., & Levy, J. (2016). Exploiting subproblem optimization in SAT-based MaxSAT algorithms. *Journal of Heuristics*, *22*(1), 1–53.

Asín, R., Nieuwenhuis, R., Oliveras, A., & Rodríguez-Carbonell, E. (2011). Cardinality networks: A theoretical and empirical study. *Constraints An International Journal*, *16*(2), 195–221.

Audemard, G., Lagniez, J., & Simon, L. (2013). Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In Järvisalo, M., & Gelder, A. V. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, Vol. 7962 of *Lecture Notes in Computer Science*, pp. 309–317. Springer.

Bacchus, F., Järvisalo, M., & Martins, R. (2019). MaxSAT Evaluation 2018: New developments and detailed results. *Journal on Satisfiability, Boolean Modeling and Computation*, *11*(1), 99–131.

Bacchus, F., Järvisalo, M., & Martins, R. (2021). Maximum satisfiability. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 929–991. IOS Press.

Bacchus, F., & Narodytska, N. (2014). Cores in Core Based MaxSat Algorithms: An Analysis. In Sinz, C., & Egly, U. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, Vol. 8561 of *Lecture Notes in Computer Science*, pp. 7–15. Springer.

Bailleux, O., & Boufkhad, Y. (2003). Efficient CNF encoding of boolean cardinality constraints. In Rossi, F. (Ed.), *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, Vol. 2833 of *Lecture Notes in Computer Science*, pp. 108–122. Springer.

Bailleux, O., Boufkhad, Y., & Roussel, O. (2009). New encodings of pseudo-boolean constraints into CNF. In Kullmann, O. (Ed.), *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, Vol. 5584 of *Lecture Notes in Computer Science*, pp. 181–194. Springer.

Barrett, C. W., Sebastiani, R., Seshia, S. A., & Tinelli, C. (2021). Satisfiability modulo theories. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability* (2 edition)., Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 1267–1329. IOS Press.

Baselice, S., Bonatti, P. A., & Gelfond, M. (2005). Towards an integration of answer set and constraint solving. In Gabbrielli, M., & Gupta, G. (Eds.), *Logic Programming, 21st International Conference, ICLP 2005, Sitges, Spain, October 2-5, 2005, Proceedings*, Vol. 3668 of *Lecture Notes in Computer Science*, pp. 52–66. Springer.

Berg, J., Bacchus, F., & Poole, A. (2020). Abstract cores in implicit hitting set MaxSat solving. In Pulina, L., & Seidl, M. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, Vol. 12178 of *Lecture Notes in Computer Science*, pp. 277–294. Springer.

Berg, J., Demirovic, E., & Stuckey, P. J. (2019). Core-boosted linear search for incomplete MaxSAT. In Rousseau, L., & Stergiou, K. (Eds.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4-7, 2019, Proceedings*, Vol. 11494 of *Lecture Notes in Computer Science*, pp. 39–56. Springer.

Berg, J., & Järvisalo, M. (2017). Weight-aware core extraction in SAT-based MaxSAT solving. In Beck, J. C. (Ed.), *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, Vol. 10416 of *Lecture Notes in Computer Science*, pp. 652–670. Springer.

Berre, D. L., & Parrain, A. (2010). The sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, *7*(2-3), 59–6.

Bjørner, N. S., & Fazekas, K. (2023). On incremental pre-processing for SMT. In Pientka, B., & Tinelli, C. (Eds.), *Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings*, Vol. 14132 of *Lecture Notes in Computer Science*, pp. 41–60. Springer.

Cai, S., Luo, C., Thornton, J., & Su, K. (2014). Tailoring local search for partial MaxSAT. In Brodley, C. E., & Stone, P. (Eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pp. 2623–2629. AAAI Press.

Chu, Y., Cai, S., & Luo, C. (2023). NuWLS: Improving local search for (weighted) partial MaxSAT by new weighting techniques. In Williams, B., Chen, Y., & Neville, J. (Eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 3915–3923. AAAI Press.

Cimatti, A., Franzén, A., Griggio, A., Sebastiani, R., & Stenico, C. (2010). Satisfiability modulo the theory of costs: Foundations and applications. In Esparza, J., & Majumdar, R. (Eds.), *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, Vol. 6015 of *Lecture Notes in Computer Science*, pp. 99–113. Springer.

Codish, M., & Zazon-Ivry, M. (2010). Pairwise cardinality networks. In Clarke, E. M., & Voronkov, A. (Eds.), *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010, Revised Selected Papers*, Vol. 6355 of *Lecture Notes in Computer Science*, pp. 154–172. Springer.

Davies, J., & Bacchus, F. (2011). Solving MAXSAT by solving a sequence of simpler SAT instances. In Lee, J. H. (Ed.), *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings*, Vol. 6876 of *Lecture Notes in Computer Science*, pp. 225–239. Springer.

Davies, J., & Bacchus, F. (2013). Postponing optimization to speed up MAXSAT solving. In Schulte, C. (Ed.), *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, Vol. 8124 of *Lecture Notes in Computer Science*, pp. 247–262. Springer.

Delisle, E., & Bacchus, F. (2013). Solving weighted CSPs by successive relaxations. In Schulte, C. (Ed.), *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, Vol. 8124 of *Lecture Notes in Computer Science*, pp. 273–281. Springer.

Devriendt, J., Gocht, S., Demirovic, E., Nordström, J., & Stuckey, P. J. (2021). Cutting to the core of pseudo-boolean optimization: Combining core-guided search with cutting planes reasoning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 3750–3758. AAAI Press.

Eén, N., & Sörensson, N. (2003). Temporal induction by incremental SAT solving. In Strichman, O., & Biere, A. (Eds.), *First International Workshop on Bounded Model Checking, BMC@CAV 2003, Boulder, Colorado, USA, July 13, 2003*, Vol. 89 of *Electronic Notes in Theoretical Computer Science*, pp. 543–560. Elsevier.

Eén, N., & Sörensson, N. (2006). Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, *2*(1-4), 1–26.

Fazekas, K., Bacchus, F., & Biere, A. (2018). Implicit hitting set algorithms for maximum satisfiability modulo theories. In Galmiche, D., Schulz, S., & Sebastiani, R. (Eds.), *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, Vol. 10900 of *Lecture Notes in Computer Science*, pp. 134–151. Springer.

Fazekas, K., Biere, A., & Scholl, C. (2019). Incremental inprocessing in SAT solving. In Janota, M., & Lynce, I. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, Vol. 11628 of *Lecture Notes in Computer Science*, pp. 136–154. Springer.

Fu, Z., & Malik, S. (2006). On solving the partial MAX-SAT problem. In Biere, A., & Gomes, C. P. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, Vol. 4121 of *Lecture Notes in Computer Science*, pp. 252–265. Springer.

Gange, G., Berg, J., Demirovic, E., & Stuckey, P. J. (2020). Core-guided and core-boosted search for CP. In Hebrard, E., & Musliu, N. (Eds.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21-24, 2020, Proceedings*, Vol. 12296 of *Lecture Notes in Computer Science*, pp. 205–221. Springer.

Gebser, M., Ostrowski, M., & Schaub, T. (2009). Constraint answer set solving. In Hill, P. M., & Warren, D. S. (Eds.), *Logic Programming, 25th International Conference, ICLP 2009, Pasadena, CA, USA, July 14-17, 2009. Proceedings*, Vol. 5649 of *Lecture Notes in Computer Science*, pp. 235–249. Springer.

Heras, F., Larrosa, J., & Oliveras, A. (2008). MiniMaxSAT: An efficient weighted Max-SAT solver. *J. Artif. Intell. Res.*, *31*, 1–32.

Heras, F., Morgado, A., & Marques-Silva, J. (2011). Core-guided binary search algorithms for maximum satisfiability. In Burgard, W., & Roth, D. (Eds.), *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011.* AAAI Press.

Hölldobler, S., Manthey, N., & Steinke, P. (2012). A compact encoding of pseudo-boolean constraints into SAT. In Glimm, B., & Krüger, A. (Eds.), *KI 2012: Advances in Artificial Intelligence - 35th Annual German Conference on AI, Saarbrücken, Germany, September 24-27, 2012. Proceedings*, Vol. 7526 of *Lecture Notes in Computer Science*, pp. 107–118. Springer.

Ignatiev, A., Morgado, A., Manquinho, V. M., Lynce, I., & Marques-Silva, J. (2014). Progression in maximum satisfiability. In Schaub, T., Friedrich, G., & O'Sullivan, B. (Eds.), *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, Vol. 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 453–458. IOS Press.

Ihalainen, H. (2022). Refined core relaxations for core-guided maximum satisfiability algorithms. Master's thesis, University of Helsinki. http://hdl.handle.net/10138/351207.

Ihalainen, H., Berg, J., & Järvisalo, M. (2021). Refined core relaxation for core-guided MaxSAT solving. In Michel, L. D. (Ed.), *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, Vol. 210 of *LIPIcs*, pp. 28:1–28:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Ihalainen, H., Berg, J., & Järvisalo, M. (2022). Clause redundancy and preprocessing in maximum satisfiability. In Blanchette, J., Kovács, L., & Pattinson, D. (Eds.), *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings*, Vol. 13385 of *Lecture Notes in Computer Science*, pp. 75–94. Springer.

Ihalainen, H., Berg, J., & Järvisalo, M. (2023). Unifying core-guided and implicit hitting set based optimization. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pp. 1935–1943. ijcai.org.

Järvisalo, M., Heule, M., & Biere, A. (2012). Inprocessing rules. In Gramlich, B., Miller, D., & Sattler, U. (Eds.), *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, Vol. 7364 of *Lecture Notes in Computer Science*, pp. 355–370. Springer.

Jiang, Y., Kautz, H., & Selman, B. (1995). Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT. In *Proceedings of the 1st International Joint Workshop on Artificial Intelligence and Operations Research*.

Joshi, S., Kumar, P., Rao, S., & Martins, R. (2019). Open-WBO-Inc: Approximation strategies for incomplete weighted MaxSAT. *J. Satisf. Boolean Model. Comput.*, *11*(1), 73–97.

Joshi, S., Martins, R., & Manquinho, V. M. (2015). Generalized totalizer encoding for pseudo-boolean constraints. In Pesant, G. (Ed.), *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, Vol. 9255 of *Lecture Notes in Computer Science*, pp. 200–209. Springer.

Karpinski, M., & Piotrów, M. (2019). Encoding cardinality constraints using multiway merge selection networks. *Constraints An International Journal*, *24*(3-4), 234–251.

Katsirelos, G. (2023). An analysis of core-guided maximum satisfiability solvers using linear programming. In Mahajan, M., & Slivovsky, F. (Eds.), *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*, Vol. 271 of *LIPIcs*, pp. 12:1–12:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Koshimura, M., Zhang, T., Fujita, H., & Hasegawa, R. (2012). QMaxSAT: A partial Max-SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, *8*(1/2), 95–100.

Lei, Z., & Cai, S. (2018). Solving (weighted) partial MaxSAT by dynamic local search for SAT. In Lang, J. (Ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 1346–1352. ijcai.org.

Li, C. M., & Manyà, F. (2021). MaxSAT, hard and soft constraints. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 903–927. IOS Press.

Li, C. M., Manyà, F., & Planes, J. (2005). Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers. In van Beek, P. (Ed.), *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, Vol. 3709 of *Lecture Notes in Computer Science*, pp. 403–414. Springer.

Li, C. M., Manyà, F., & Planes, J. (2006). Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pp. 86–91. AAAI Press.

Li, C., Xu, Z., Coll, J., Manyà, F., Habet, D., & He, K. (2021). Combining clause learning and branch and bound for MaxSAT. In Michel, L. D. (Ed.), *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, Vol. 210 of *LIPIcs*, pp. 38:1–38:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Li, C., Xu, Z., Coll, J., Manyà, F., Habet, D., & He, K. (2022). Boosting branch-and-bound MaxSAT solvers with clause learning. *AI Commun.*, *35*(2), 131–151.

Manthey, N., Philipp, T., & Steinke, P. (2014). A more compact translation of pseudo-boolean constraints into CNF such that generalized arc consistency is maintained. In Lutz, C., & Thielscher, M. (Eds.), *KI 2014: Advances in Artificial Intelligence - 37th Annual German Conference on AI, Stuttgart, Germany, September 22-26, 2014. Proceedings*, Vol. 8736 of *Lecture Notes in Computer Science*, pp. 123–134. Springer.

Marques-Silva, J., Lynce, I., & Malik, S. (2021). Conflict-driven clause learning SAT solvers. In *Handbook of Satisfiability* (2 edition)., Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 133–182. IOS Press.

Marques-Silva, J., & Planes, J. (2007). On using unsatisfiability for solving maximum satisfiability. *CoRR*, *abs/0712.1097*.

Morgado, A., Dodaro, C., & Marques-Silva, J. (2014). Core-guided MaxSAT with soft cardinality constraints. In O'Sullivan, B. (Ed.), *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, Vol. 8656 of *Lecture Notes in Computer Science*, pp. 564–573. Springer.

Morgado, A., Heras, F., Liffiton, M. H., Planes, J., & Marques-Silva, J. (2013). Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints An International Journal*, *18*(4), 478–534.

Nadel, A. (2020). Anytime algorithms for MaxSAT and beyond. In *2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21-24, 2020*, p. 1. IEEE.

Narodytska, N., & Bacchus, F. (2014). Maximum satisfiability using core-guided MaxSAT resolution. In Brodley, C. E., & Stone, P. (Eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pp. 2717–2723. AAAI Press.

Narodytska, N., & Bjørner, N. S. (2022). Analysis of core-guided MaxSat using cores and correction sets. In Meel, K. S., & Strichman, O. (Eds.), *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, Vol. 236 of *LIPIcs*, pp. 26:1–26:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Nieuwenhuis, R., Oliveras, A., & Tinelli, C. (2006). Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(t). *Journal of the ACM*, *53*(6), 937–977.

Ogawa, T., Liu, Y., Hasegawa, R., Koshimura, M., & Fujita, H. (2013). Modulo based CNF encoding of cardinality constraints and its application to MaxSAT solvers. In *25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2013, Herndon, VA, USA, November 4-6, 2013*, pp. 9–17. IEEE Computer Society.

Paxian, T., Reimer, S., & Becker, B. (2018). Dynamic polynomial watchdog encoding for solving weighted MaxSAT. In Beyersdorff, O., & Wintersteiger, C. M. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, Vol. 10929 of *Lecture Notes in Computer Science*, pp. 37–53. Springer.

Piotrów, M. (2020). UWrMaxSat: Efficient solver for MaxSAT and pseudo-boolean problems. In *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9-11, 2020*, pp. 132–136. IEEE.

Planes, J. (2003). Improved branch and bound algorithms for Max-2-SAT and weighted Max-2-SAT. In Rossi, F. (Ed.), *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, Vol. 2833 of *Lecture Notes in Computer Science*, p. 991. Springer.

Prestwich, S. D. (2021). CNF encodings. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability* (2 edition)., Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 75–100. IOS Press.

Saikko, P., Berg, J., & Järvisalo, M. (2016). LMHS: A SAT-IP hybrid MaxSAT solver. In Creignou, N., & Berre, D. L. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, Vol. 9710 of *Lecture Notes in Computer Science*, pp. 539–546. Springer.

Saikko, P., Dodaro, C., Alviano, M., & Järvisalo, M. (2018). A hybrid approach to optimization in answer set programming. In Thielscher, M., Toni, F., & Wolter, F. (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, pp. 32–41. AAAI Press.

Silva, J. P. M., & Sakallah, K. A. (1999). GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, *48*(5), 506–521.

Sinz, C. (2005). Towards an optimal CNF encoding of boolean cardinality constraints. In van Beek, P. (Ed.), *Principles and Practice of Constraint Programming - CP 2005, 11th*

International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings, Vol. 3709 of *Lecture Notes in Computer Science*, pp. 827–831. Springer.

Smirnov, P., Berg, J., & Järvisalo, M. (2021). Pseudo-boolean optimization by implicit hitting sets. In Michel, L. D. (Ed.), *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, Vol. 210 of *LIPIcs*, pp. 51:1–51:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Smirnov, P., Berg, J., & Järvisalo, M. (2022). Improvements to the implicit hitting set approach to pseudo-boolean optimization. In Meel, K. S., & Strichman, O. (Eds.), *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, Vol. 236 of *LIPIcs*, pp. 13:1–13:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Tseitin, G. (1983). On the complexity of derivation in propositional calculus. In Siekmann, J. H., & Wrightson, G. (Eds.), *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, pp. 466–483. Springer Berlin Heidelberg, Berlin, Heidelberg.

Warners, J. P. (1998). A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, *68*(2), 63–69.

Zhang, L., Madigan, C. F., Moskewicz, M. W., & Malik, S. (2001). Efficient conflict driven learning in boolean satisfiability solver. In Ernst, R. (Ed.), *Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2001, San Jose, CA, USA, November 4-8, 2001*, pp. 279–285. IEEE Computer Society.