Unifying Core-Guided and Implicit Hitting Set Based Optimization

Hannes Ihalainen, Jeremias Berg and Matti Järvisalo
HIIT, Department of Computer Science, University of Helsinki, Finland
{hannes.ihalainen,jeremias.berg,matti.jarvisalo}@helsinki.fi

Abstract

Two of the most central algorithmic paradigms implemented in practical solvers for maximum satisfiability (MaxSAT) and other related declarative paradigms for NP-hard combinatorial optimization are the core-guided (CG) and implicit hitting set (IHS) approaches. We develop a general unifying algorithmic framework, based on the recent notion of abstract cores, that captures both CG and IHS computations. The framework offers a unified way of establishing the correctness of variants of the approaches, and can be instantiated in novel ways giving rise to new algorithmic variants of the core-guided and IHS approaches. We illustrate the latter aspect by developing a prototype implementation of an algorithm variant for MaxSAT based on the framework.

1 Introduction

The declarative paradigm of maximum satisfiability (MaxSAT) [Bacchus et al., 2021] is 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 Boolean satisfiability (SAT) based MaxSAT algorithms capable of computing provably optimal solutions. Two of the most popular and effective algorithmic approaches implemented in MaxSAT solvers are variants of the so-called core-guided (CG) [Fu and Malik, 2006; Marques-Silva and Platane, 2007; Heras et al., 2011; Ansótegui et al., 2013; Morgado et al., 2013; Morgado et al., 2014; Narodytska and Bacchus, 2014; Alviano et al., 2015; Ansótegui et al., 2016; Ansótegui and Gabas, 2017] and implicit hitting set (IHS) [Davies and Bacchus, 2011; Davies and Bacchus, 2013; Saikko et al., 2016] approaches.

Both CG and IHS are unsatisfiability-based approaches, relying on iteratively extracting unsatisfiable cores using a SAT solver as a core-extracting decision oracle [Één and Sörensson, 2003]. However, CG and IHS solvers deal with cores extracted during search differently. CG 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. The choice of the subset of constraints to consider at each iteration is dictated by a (minimum-cost) hitting set computer over the so-far accumulated set of cores. In practice, state-of-the-art core-guided and IHS MaxSAT solvers are both competitive in terms of runtime performance. However, their relative performance on distinct problem domains can vary noticeably, core-guided solvers outperforming IHS on specific domains, IHS outperforming core-guided on distinct other domains [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 CG search are tightly related to cores extracted in IHS search on the original instance [Bacchus and Narodytska, 2014; Narodytska and Björner, 2022].

Taking a different view, we develop a general algorithmic framework that captures CG and IHS computations in a unifying way. The framework is based on the recently-proposed notion of abstract cores originally presented as a performance-improving technique for IHS [Berg et al., 2020] that brings a flavor of CG reformulation into the representation of the hitting set problems solved during IHS search. Our framework provides a unified way of establishing the correctness of variants of core-guided and IHS approaches. Further, the framework can be instantiated in novel ways giving rise to new variants of unsatisfiability-based MaxSAT algorithms. As an illustration of its potential for obtaining novel types of unsatisfiability-based algorithms, we outline and implement a prototype of a core-guided variant for MaxSAT obtained through the framework which turns out to be promising also from a practical perspective.

While our discussion is grounded in MaxSAT, the framework is applicable also for capturing core-guided and IHS search approaches beyond MaxSAT, including ones developed e.g. for pseudo-Boolean optimization [Devriendt et al., 2021; Smirnov et al., 2021; Smirnov et al., 2022], finite-domain constraint optimization problems [Delisle and Bacchus, 2013; Gange et al., 2020] and answer set programming [Andres et al., 2012; Saikko et al., 2018; Alviano and...
core-extracting decision oracle, but the exact constraint language on which the core extraction is performed does not fundamentally influence the overall algorithmic approaches.

Our unifying framework can also be viewed as a further development in a longer line of research developing theoretical frameworks that capture the underpinning of automated reasoning engines, such ones proposed for SAT and SMT [Nieuwenhuis et al., 2006; Larroca et al., 2011; Järvisalo et al., 2012; Fazekas et al., 2018].

2 Maximum Satisfiability

For a Boolean variable \( x \) there are two literals, \( x \) and \( \bar{x} \). A clause \( C = l_1 \lor \ldots \lor l_n \) is a disjunction of literals, a conjunctive normal form (CNF) formula \( F = \{C_1, \ldots, C_m\} \) is a set of clauses. The set \( \text{var}(C) \) consists of the variables for which either \( x \in C \) or \( \bar{x} \in C \). An assignment \( \tau \) maps variables to \( 1 \) (true) or \( 0 \) (false). Assignments extend to a literal \( l \), clause \( C \) and formula \( F \) in the standard way: \( \tau(l) = 1 - \tau(l) \), \( \tau(C) = \max\{\tau(l) \mid l \in C\} \) and \( \tau(F) = \min\{\tau(C) \mid C \in F\} \), \( \tau \) satisfies \( F \) if \( \tau(F) = 1 \). We may treat the as the set of literals \( \tau \) assigns to \( 1 \), so that \( l \in \tau \) denotes \( \tau(l) = 1 \) and \( l \notin \tau \) denotes \( \tau(l) = 0 \). \( \tau \) is complete for \( F \) if it assigns each variable in \( F \), and otherwise partial.

Cardinality constraints are linear inequalities of the form \( \sum_i x_i \geq k \), where each \( x_i \) is a Boolean variable and \( k \) a positive constant. The constraint \( \sum_i x_i \geq k \) is satisfied by an assignment \( \tau \) if \( \sum_i \tau(x_i) \geq k \). We do not make assumptions on the CNF encoding used for cardinality constraints, and abstractly use \( \text{asCNF}(\sum_i x_i \geq k) \) to denote a CNF formula that is satisfiable by an assignment \( \tau \) iff \( \sum_i \tau(x_i) \geq k \).

Taking a name \( o \) to indicate whether a cardinality constraint is satisfied, we also use \( \text{asCNF}(\sum_i x_i \leq k \iff o) \) to denote a CNF formula that is satisfied by any assignment \( \tau \) that sets \( \tau(o) = 1 \) iff \( \sum_i \tau(x_i) \geq k \). Various CNF encodings of cardinality constraints have been proposed [Bailleux and Boufkhad, 2003; Sinz, 2005; Eén and Sörensson, 2006; Codish and Zazon-Ivry, 2010; Asín et al., 2011; Abío et al., 2013; Karpinski and Piotrōw, 2019].

An instance \( F = (F, O) \) of maximum satisfiability (MaxSAT) consists of a CNF formula \( F \) and an objective function \( O = \sum_i w_i b_i \), where \( w_i \) are positive constants and \( b_i \) is a variable of \( F \). (This view on MaxSAT is equivalent to the classical view on MaxSAT with hard and weighted soft clauses [Berg and Järvisalo, 2019].) The set \( \text{var}(O) \) consists of the variables that appear in \( O \). A complete satisfying assignment \( \tau \) to \( F \) is a solution to \( F \) with cost \( O(\tau) = \sum_i w_i \tau(b_i) \). A solution is optimal if there are no solutions with lower cost. The cost of optimal solutions is denoted by \( \text{opt}(F) \).

Example 1. Consider the MaxSAT instance \( F = (F, O) \) with \( F = \{b_1 \lor b_2 \lor b_3, b_1 \lor b_3 \lor b_5\} \), \( O = b_1 + b_2 + 3b_3 + b_4 + 2b_5 \). An optimal solution to \( F \) is \( \tau = \{b_1, b_2, b_3, b_5\} \), assigning all variables except \( b_4 \) to 0. The cost of \( \tau \) is \( O(\tau) = \tau(b_1) + \tau(b_2) + 3\tau(b_3) + \tau(b_4) + 2\tau(b_5) = 2 \).

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

3 Core-Guided and IHS Search for MaxSAT

We develop a unifying algorithmic framework for core-guided [Fu and Malik, 2006; Marques-Silva and Planes, 2007; Morgado et al., 2013; Morgado et al., 2014; Nardytska and Bacchus, 2014; Alviano et al., 2015; Ansotegui and Gabás, 2017] and implicit hitting set algorithms [Davies and Bacchus, 2011; Davies and Bacchus, 2013; Saikkonen et al., 2016]. As background, we describe these approaches in general terms; practical solver implementations employ various heuristics and optimizations which do not affect our main contributions.

Both core-guided and IHS algorithms use an incremental SAT solver that determines the satisfiability of CNF formulas under different sets of assumptions [Eén and Sörensson, 2003]. Given a CNF formula \( F \) and partial assignment \( \gamma^A \) (constituting a set of assumptions, as a set of literals), we abstract the SAT solver into the subroutine \( \text{extract-core} \) that returns a triplet \( (\text{res}, C, \tau) \). Here \( \text{res} = \text{true} \) if there is a satisfying assignment \( \tau \geq \gamma^A \) of \( F \). If there is no such assignment, \( \text{res} = \text{false} \) and \( C \) is a clause over a subset of the variables satisfied by \( \gamma^A \) entailed by \( F \) for which \( \gamma^A(C) = 0 \).\n
\text{Core-Guided Search.} Algorithm 1 details a general abstraction of the core-guided approach to computing an optimal solution to a MaxSAT instance \( F = (F, O) \). First the algorithm initializes a set \( \text{const} \) of cardinality constraints as empty and a working objective function \( O' \) 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 \( \tau \) that (i) satisfies all clauses in \( F \) and all of the cardinality constraints in \( \text{const} \) and (ii) falsifies all objective variables of the current working objective \( O' \), i.e., \( O'(\tau) = 0 \) (Lines 4–5). If there is such a \( \tau \), it is returned as an optimal solution to the original instance (Line 6). Otherwise a core \( C \) of \( (F \cup \text{const}, O') \) is obtained. The core is then relaxed (Lines 7–9), transforming the current instance in a way that

\begin{algorithm}
\caption{Core-guided approach to MaxSAT}
\begin{algorithmic}[1]
\Input MaxSAT instance \( F = (F, O) \).
\Output Optimal solution \( \tau \) to \( F \).
1: \( \text{const} = \emptyset \)
2: \( O' = O \)
3: \textbf{while \ true \ do}
4: \( \gamma^A = \{x \mid x \in \text{var}(O')\} \)
5: \( (\text{res}, C, \tau) = \text{extract-core}(F \cup \text{const}, \gamma^A) \)
6: \textbf{if \ res = 'true' then return } \tau \)
7: \( (D, \text{out}) = \text{create-card-constr}(C) \)
8: \( \text{const} = \text{const} \cup D \)
9: \( O' = \text{refine-objective}(C, \text{out}, O') \)
\end{algorithmic}
\end{algorithm}
Algorithm 2 IHS for MaxSAT

Input: MaxSAT instance \( F = (F, O) \).
Output: Optimal solution \( \tau \) to \( F \).

1: \( \mathcal{K} = \emptyset \)
2: \textsf{while} true \textsf{do}
3: \( \gamma^A = \{ x \mid x \in \text{var}(O) \setminus \text{MINCOST-HS}(\mathcal{K}) \} \)
4: \( (\text{res}, C, \tau) = \text{EXTRACT-CORE}(F, \gamma^A) \)
5: \textsf{if} res = 'true' \textsf{then return} \( \tau \)
6: \textsf{else} \( \mathcal{K} = \mathcal{K} \cup C \)

Conceptually, modern core-guided algorithms differ mainly in the specifics of the core relaxation step. We de-


defines cardinality constraints over the core to \( \mathcal{K} \) and \( O \) (Lines 7–8) and updates the core working objective (Line 9).

Consequently, modern core-guided algorithms differ mainly in the specifics of the core relaxation step. We de-


defines cardinality constraints over the core to \( \mathcal{K} \) and \( O \) (Lines 7–8) and updates the core working objective (Line 9).

Example 2. Invoke \( \textsc{OLL} \) on \( F = (F, O) \) from Example 1. The first call to \( \textsc{EXTRACT-CORE} \) is under the assumptions \( \gamma^A = \{ b_1, b_2, b_3, b_4, b_5 \} \). Let the first core obtained be \( C_1 = \{ b_1 \lor b_2 \lor b_3 \lor b_4 \lor b_5 \} \). Relaxing \( C_1 \) introduces the cardinality constraint \( \textsc{ASCNF}(\sum_{x \in C_1} x \geq i \iff o^i_2) \) for \( i = 2, 3, 4, 5 \) and the new objective is \( O' = 2b_3 + b_5 + o^1_3 + o^3_3 + o^4_3 + o^5_3 \). The next call to \( \textsc{EXTRACT-CORE} \) is under the assumptions \( \gamma^A = \{ b_1, b_3, o^3_3, o^2_1, o^5_3 \} \). Let the next core obtained be \( C_2 = \{ o^2_1 \lor b_3 \} \). Relaxing \( C_2 \) introduces the cardinality constraint \( \textsc{ASCNF}(\sum_{x \in C_2} x \geq 2 \iff o^3_2) \) and the new objective \( O' = b_3 + b_5 + o^1_3 + o^4_3 + o^5_3 \). The assumptions in the 3rd call to \( \textsc{EXTRACT-CORE} \) are \( \gamma^A = \{ b_1, b_3, o^3_3, o^2_1, o^5_3 \} \). Now e.g. the assignment \( \tau = \{ b_1, b_2, b_3, b_4, b_5 \} \) that also assigns \( \{ o^3_1, o^2_1, o^3_2, o^5_3 \} \) is returned as an optimal solution to \( F \).

IHS Search. Algorithm 2 is a generic abstraction of the IHS approach to MaxSAT. IHS iteratively extracts cores of an input MaxSAT instance and stores them in the set \( \mathcal{K} \). Instead of reformulating the objective, IHS invokes the \( \text{MINCOST-HS}(\mathcal{K}) \) 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 of the objective variables that the variables in \( h_s \), assigned to 1, satisfy all cores in \( \mathcal{K} \). In each iteration of the main loop (Lines 2–6), \( \text{EXTRACT-CORE} \) is queried for a solution that falsifies all objective variables that are not contained in the \( h_s \) computed over the current set of cores (Lines 3–4). If there is such a \( \tau \), it is an optimal solution on 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 \( O \). IHS iterates until the MCHS can be extended into a solution of \( F \), at which point it satisfies (or hits) all cores (not only the cores in \( \mathcal{K} \)) of the instance implicitly. Note that the assumptions \( \gamma^A \) set up on Line 3 constitute a partial assignment over the objective variables that can be extended into a solution of \( \mathcal{K} \) in a unique way.

Example 3. Invoke Algorithm 2 on the MaxSAT instance \( F = (F, O) \) from Example 1. In the first iteration there are no cores, so \( \text{MINCOST-HS}(\mathcal{K}) = \emptyset \). The first call to \( \text{EXTRACT-CORE} \) is under the assumptions \( \gamma^A = \{ b_1 \} \). There are a number of cores that could be returned: let the first core obtained be \( C_1 = \{ b_1 \lor b_2 \lor b_3 \lor b_4 \lor b_5 \} \). In iteration 2 there are three different MCHSs over \( \mathcal{K} \). Assume that \( \text{MINCOST-HS} \) returns \( \{ b_1 \} \). Then the assumptions for the next call to \( \text{EXTRACT-CORE} \) are \( \gamma^A = \{ b_2, b_3, b_4, b_5 \} \). Assume that the next core is \( C_2 = \{ b_3 \lor b_4 \lor b_5 \} \). In iteration 3 the only MCHS over \( \mathcal{K} = \{ C_1, C_2 \} \) is \( \{ b_1 \} \), the assumptions for the next call to \( \text{EXTRACT-CORE} \) are \( \gamma^A = \{ b_1, b_2, b_3, b_4 \} \). Assume that the next core is \( C_3 = \{ b_1 \lor b_2 \lor b_3 \} \). In iteration 4 there are two possible MCHSs over \( \mathcal{K} = \{ C_1, C_2, C_3 \} \). Assume that \( \text{MINCOST-HS} \) returns \( \{ b_2, b_3 \} \), leading to the assumptions \( \gamma^A = \{ b_2, b_3, b_4, b_5 \} \). Now \( \text{EXTRACT-CORE} \) returns the solution \( \tau = \{ b_1, b_2, b_3, b_4, b_5 \} \) as an optimal solution to \( F \).

4 Unifying CG and IHS

As our main contribution we present \textsc{UniMaxSAT}, a general algorithmic framework unifying core-guided and IHS-based MaxSAT algorithms. The framework builds on the notion of abstract cores originally proposed as basis for a refinement of \( \text{IHS} \) [Berg et al., 2020]. We start with 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].

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

Given an abstraction set \( \text{AB} = (in, D, out) \) we call \( D \) the definitions of the outputs \( out \).

for a collection \( \text{AB} = \{(i_1, D_1, out_1), \ldots, (i_n, D_n, out_n)\} \) of abstraction sets, \( \text{DEF}(\text{AB}) = \bigcup_{i=1}^{n} D_i \) is the CNF formula consisting of the definitions in \( \text{AB} \) and \( \text{OUTS}(\text{AB}) = \bigcup_{i=1}^{n} out_i \) is the set of
outputs occurring in \(\mathcal{A}B\). We say that \(\mathcal{A}B\) is feasible for a MaxSAT instance \(\mathcal{F} = (F, O)\) if \(\text{DEF}(\mathcal{A}B)\) does not change the set of solutions to \(\mathcal{F}\), i.e., if every solution \(\tau\) to \(\mathcal{F}\) can be uniquely extended into a solution \(\tau^F \supseteq \tau\) to \(F \cup \text{DEF}(\mathcal{A}B)\). We will consider collections of abstractions sets that are feasible for specific MaxSAT instances.

An abstract core is a clause entailed by a formula together with the definitions of a feasible collection of abstraction sets. Importantly, an abstract core can contain both objective variables and outputs of abstraction sets.

**Definition 2.** Given MaxSAT instance \(\mathcal{F} = (F, O)\) and collection \(\mathcal{A}B\) of feasible abstraction sets, a clause \(C\) is an abstract core of \(\mathcal{F}\) wrt \(\mathcal{A}B\) if (i) \(\text{var}(C) \subseteq (\text{var}(O) \cup \text{var}(\text{OUTS}(\mathcal{A}B)))\) and (ii) \(\tau(C) = 1\) for each solution \(\tau\) of \(\mathcal{F} \cup \text{DEF}(\mathcal{A}B)\).

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

**Example 4.** Consider the \(\mathcal{F}\) from Example 1 and the abstraction set \(\mathcal{A}B = \{(b_1, \ldots, b_5), \{\text{ASCNF}(\sum_{j=1}^{5} b_j \geq j \leftrightarrow o_j = j) \mid j = 2, 3, 4, 5\}, \{o_2, \ldots, o_5\}\}\). Then \(C = (o_2 \vee b_5)\) is an abstract core as any satisfying assignment of \(\mathcal{F} \cup \text{DEF}(\mathcal{A}B)\) must assign either \(b_3 = 1\) or least two variables of \(\{b_1, b_2, b_3, b_4, b_5\}\) to 1, forcing \(o_2 = 1\). Note how \(C\) corresponds to the core \(C_2\) from Example 2.

Our general framework for core-guided and IHS search 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 the input MaxSAT instance from solutions to cores, we call solutions to a set of abstract cores candidate solutions (candidates for short): for a MaxSAT instance \(\mathcal{F} = (F, O)\), a collection \(\mathcal{A}B\) of abstraction sets and a set \(K\) of abstract cores, a complete satisfying assignment \(\delta\) of \(K \cup \text{DEF}(\mathcal{A}B)\) that assigns each variable in \(\text{var}(O)\) is a candidate (solution) to \(K\) with cost \(O(\delta)\). \(\delta\) is minimum-cost if \(O(\delta) \leq O(\delta')\) for all candidates \(\delta'\) of \(K\).

The following observation details how abstraction sets and abstract cores are used in our framework to compute lower bounds (which are used to prove optimality of solutions).

**Proposition 1.** Let \(\mathcal{F} = (F, O)\) be a MaxSAT instance, \(K\) a set of abstract cores, and \(\delta\) a minimum-cost candidate of \(K\). Then \(O(\delta) \leq \text{OPT}(\mathcal{F})\).

We will show that the correctness of IHS and core-guided algorithms follows from the fact that, instead of ruling out complete candidates on each iteration, it suffices to rule out partial assignments that extend solely to minimum-cost candidates. The notion of a (minimum-cost) abstract candidate is central in this respect.

**Definition 3.** Let \(\mathcal{F} = (F, O)\) be a MaxSAT instance, \(\mathcal{A}B\) a collection of feasible abstraction sets and \(K\) a set of abstract cores. A partial assignment \(\gamma^A\) over a subset of the variables in \(\text{var}(K) \cup \text{var}(O)\) is an abstract candidate of \(K\) if (i) there is at least one extension \(\tau \supseteq \gamma^A\) which is a solution of \(\text{DEF}(\mathcal{A}B) \cup K\), i.e., a candidate of \(K\) and (ii) all such extensions are minimum-cost candidates of \(K\).

While each minimum-cost candidate of \(K\) is also an abstract candidate, the converse need not hold.

**Example 5.** Consider the MaxSAT instance \(\mathcal{F} = (F, O)\) from Example 1 and the set \(K = \{(b_1 \lor b_2 \lor b_3, (b_1 \lor b_4 \lor b_5)\}\) of abstract cores. A minimum-cost candidate of \(K\) is \(\delta = \{b_1, b_2, b_3, b_4\}\). An abstract candidate is \(\gamma^A = \{b_1, b_3, b_5\}\) since the only extension of \(\gamma^A\) into a solution to \(K\) is \(\delta\). The set \(\{b_1, b_3\}\) is not an abstract candidate since it extends to the solution \(\{b_1, b_2, b_3, b_4, b_5\}\) of \(K\) which is not minimum-cost.

The assumptions employed during iterations of core-guided algorithms can be seen as abstract candidates; for an example, consider the following, specific to OLL.

**Example 6.** Recall the MaxSAT instance \(\mathcal{F} = (F, O)\) from Example 1. Consider the set of cores \(K = \{b_1 \lor b_2 \lor b_3 \lor b_4 \lor b_5\}\), and the abstraction set \(\mathcal{A}B\) detailed in Example 4. The set \(\gamma^A = \{b_3, b_5, o_2, o_3, o_4\}\) is an abstract candidate as it can be extended into a solution to \(K \cup \text{DEF}(\{\mathcal{A}B\})\) by assigning exactly one literal in \(\{b_1, b_2, b_4\}\) to 1 and the rest to 0. Note that \(\gamma^A\) is exactly the set of assumptions that \textsc{Extract-Core} is queried under in iteration 2 of OLL in Example 2.

**UNIMAXSAT**

We now describe UNIMAXSAT, a general algorithmic framework for core-guided and IHS MaxSAT algorithms. Detailed as Algorithm 3 and Figure 1, given a MaxSAT instance \(\mathcal{F} = (F, O)\) as input, UNIMAXSAT outputs an optimal solution to \(\mathcal{F}\). The generic algorithm maintains an increasing set \(\mathcal{A}B\) and \(K\) of abstraction sets and abstract cores, respectively. In each iteration, an abstract candidate \(\gamma^A\) of \(K\) is computed by the \textsc{Abs-Candidate} subroutine. The \textsc{Ex-AbsCore} subroutine is invoked to check for an extension of \(\gamma^A\) into a solution to \(F\). If one exists, the algorithm terminates and returns the extension as an optimal solution. Otherwise a new abstract core falsified by \(\gamma^A\) is obtained. The core is added to \(K\), thus blocking \(\gamma^A\) and all of its extensions from further consideration, and the \textsc{Add-AbsSets} subroutine adds new abstraction sets to \(\mathcal{A}B\).

We formalize the correctness of Algorithm 3: it terminates on any MaxSAT instance and outputs an optimal solution of the input instance, subject to generic properties of its three subroutines. Importantly, correctness of the general framework allows for establishing the correctness of any
Algorithm 3 UnimaxSAT, a unified framework for core-guided and IHS-based MaxSAT algorithms.

Input: MaxSAT instance $F = (F, O)$.
Output: Optimal solution $\tau$ to $F$.

1. $AB^1 = \emptyset, K^1 = \emptyset$
2. for $i = 1 \ldots$ do
3. $\gamma^A = \text{ABS-CANDIDATE}(AB^i, K^i)$
4. $(\text{res}, C, \tau) = \text{EX-ABSCORE}(F, \text{DEF}(AB^i), \gamma^A)$
5. if $\text{res} = \text{true}$ then return $\tau$
6. $K^{i+1} = \{C\} \cup K^i$
7. $AB^{i+1} = AB^i \cup \text{ADD-ABSETS}(F, K^{i+1})$

of its instantiations—which include known core-guided and IHS algorithms for MaxSAT—by showing how each algorithm can be viewed as an instantiation of the UnimaxSAT.

Theorem 1. Let $F = (F, O)$ be an input instance to UnimaxSAT that has solutions. Assume that the following three properties hold on every iteration $i$ of UnimaxSAT.

1. $\text{ABS-CANDIDATE}(AB^i, K^i)$ computes an abstract candidate of $K^i$.
2. $\text{EX-ABSscore}(F, \text{DEF}(AB^i), \gamma^A)$ computes a solution $\tau \supseteq \gamma^A$ of $F \cup \text{DEF}(AB^i)$, or a core $C$ satisfied by all solutions to $F \cup \text{DEF}(AB^i)$ and falsified by $\gamma^A$.
3. $AB^i$ is feasible for $F$.

Then UnimaxSAT terminates and returns an optimal solution of $F$.

The proof of Theorem 1 relies the abstraction sets added and abstract cores obtained decreasing the set of candidates that ABS-CANDIDATE may provide on each iteration.

Lemma 1. Consider an iteration $i$ in which UnimaxSAT invoked on an instance $(F, O)$ does not terminate. Let $K^i$ and $AB^i$ be the set of abstract cores and abstraction sets obtained so far. Denote by $\text{OBJ-SOLS}^i$ the restrictions of all solutions to $K^i \cup \text{DEF}(AB^i)$ onto $\forall x(O)$. Then $\text{OBJ-SOLS}^{i+1} \subseteq \text{OBJ-SOLS}^i$.

Proof of Theorem 1. By assumptions 1–2, the clause $C$ obtained from EX-ABScore is an abstract core of $F$ and $AB^i$. Since $AB^i \subseteq AB^{i+1}$ holds for all $i$, $C$ is also an abstract core in all subsequent iterations. Thus all clauses in $K^i$ are abstract cores of $F$ and $AB^i$.

For optimality of returned solutions, assume that the algorithm terminates on iteration $i$ and returns a solution $\tau$. Then $\tau \supseteq \gamma^A$ for an abstract candidate $\gamma^A$ of $K^i$. Since $\tau$ is also a solution to $\text{DEF}(AB^i)$, $\tau$ is a minimum-cost candidate of $K^i$. Thus $O(\tau) \leq \text{OPT}(F) \leq O(\tau)$: the first inequality is by Proposition 1 and the second by $\tau$ being a solution to $F$. We conclude that $O(\tau) = \text{OPT}(F)$.

Now consider termination. As $F$ has solutions and $AB^i$ is feasible, $F \cup \text{DEF}(AB^i)$ has solutions for all $i$. By definition of abstract cores, all solutions to $F \cup \text{DEF}(AB^i)$ are solutions to $\text{DEF}(AB^i) \cup K^i$. In each iteration, the abstract candidate $\gamma^A$ obtained from ABS-CANDIDATE can be extended into at least one solution to $\text{DEF}(AB^i) \cup K^i$. We argue that eventually $\gamma^A$ can also be extended into a solution to $F \cup \text{DEF}(AB^i)$. This follows from the finite number of possible assignments to the variables in $O$ and Lemma 1 by which, given that the algorithm does not terminate during specific iteration, the number of assignments to variables of $O$ that can be extended into candidates of the found cores will decrease. As the solutions to $F \cup \text{DEF}(AB^i)$ are candidates for any sets of cores, at some iteration ABS-CANDIDATE will return an abstract candidate that can be extended into a solution to $F \cup \text{DEF}(AB^i)$, resulting in termination.

5 Capturing Existing Algorithms

We detail existing unsatisfiability-based MaxSAT algorithms as instantiations of UnimaxSAT. Specifically, we explain how to instantiate the three subroutines of UnimaxSAT so that the assumptions of Theorem 1 hold to obtain IHS and core-guided algorithms. By Theorem 1, this yields uniform proofs of correctness for IHS (including its abstract-cores extension [Berg et al., 2020]) and modern core-guided algorithms. For core-guided instantiations, we detail OLL and more shortly explain how the further core-guided variants MSU3 [Marques-Silva and Planes, 2007], WPM3 [Ansótegui and Gabás, 2017], PMRES [Narodytska and Bacchus, 2014] and K [Alviano et al., 2015] are obtained.

The EX-ABSscore subroutine of UnimaxSAT is in general a core-extracting SAT solver: given a MaxSAT instance $F = (F, O)$, a feasible collection $AB$ of abstraction sets and an abstract candidate $\gamma^A$, EX-ABSscore invokes a SAT solver on $F \cup \text{DEF}(AB^i)$ under assumptions $\gamma^A$, fulfilling assumption 2 of Theorem 1. Feasibility of abstraction sets computed by each considered algorithm follows from that the definitions of new abstraction sets can only intersect with the input instance and previous abstraction sets on their inputs. We will thus assume abstraction sets to be feasible. With these considerations, we next argue that each of the recent unsatisfiability-based MaxSAT algorithms can be viewed as an instantiation of UnimaxSAT by (i) specifying the instantiations of ABS-CANDIDATE and ADD-ABSETS, and (ii) arguing that the instantiation of ABS-CANDIDATE correctly computes an abstract candidate.

5.1 Capturing IHS

IHS. UnimaxSAT gives the (basic) IHS (Algorithm 2) by instantiating ADD-ABSETS to never add any abstraction sets and EX-ABSscore as a procedure that—given a set $K$ of cores—returns the abstract candidate $\gamma^A = \{\bar{x} \mid x \in O \setminus \text{MINCOST-HS}(K)\}$ that assigns all literals in the objective to 0 except the ones in a most recent minimum-cost hitting set MINCOST-HS$(K)$ over $K$. The correctness of Algorithm 2 now follows by Theorem 1 by observing that the only extension of $\gamma^A$ into a candidate of $K$ is $\{\bar{x} \mid x \in O \setminus \text{MINCOST-HS}(K)\} \cup \{x \mid x \in \text{MINCOST-HS}(K)\}$ and is minimum-cost. Hence $\gamma^A$ is an abstract candidate of $K$.

IHS with abstract cores. UnimaxSAT gives IHS enhanced with abstract cores by instantiating EX-ABSscore as in basic IHS, and ADD-ABSETS to (heuristically) compute abstraction sets $(in, D, out)$ the inputs $in = \{x_1, \ldots, x_n\}$ of which 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 \( D = \{\text{asCNF}(\sum_{x \in \text{int}} x \geq i \iff o_i) \mid k = 1, \ldots, n\} \), resulting in the outputs counting the number of inputs assigned to 1 in all satisfying assignments. The ABS-CANDIDATE subroutine first computes a minimum-cost solution \( \gamma \) to \( K \cup \text{DEF}(AB) \) that assigns all variables in \( \text{var}(O) \cup \text{OUTS}(AB) \), and then returns either \( \gamma^A \) as the restriction of \( \gamma \) onto the objective variables, or \( \gamma^B \) as the restriction of \( \gamma \) onto the outputs of the current abstraction sets and objective variables that are not inputs to any abstraction sets. The correctness of IHS with abstract cores is now established by Theorem 1 as follows. Since the only extension of it into a solution of \( K \cup \text{DEF}(AB) \) is \( \gamma \), \( \gamma^A \) is an abstract candidate. Finally, \( \gamma^B \) is an abstract candidate since (i) \( \gamma \) is an extension of \( \gamma^A \) into a solution of \( K \cup \text{DEF}(AB) \) and (ii) every such extension has the same cost as the inputs to all abstraction sets have the same coefficients in \( O \).

5.2 Capturing Core-Guided Algorithms

Moving to core-guided algorithms, we first give a more detailed description on OLL, and briefly cover other algorithms.

OLL Key to viewing OLL through UnimaxSAT is that the cardinality constraints introduced by OLL are seen as abstraction sets, i.e., ADD-ABSETS relaxing a core \( C \) introduces abstraction set \( AB^C = (C, D, out) \) where \( D = \{\text{asCNF}(\sum_{x \in \text{active}} x \geq i \iff o_i) \mid 2 \leq i \leq |C|\} \) and \( out = \{o_2, \ldots, o_{|C|}\} \). The set of assumptions used in SAT solver calls are seen as abstract candidates. We thus assume that ABS-CANDIDATE maintains and updates reformulated objective \( O^t \).

In OLL, the dependencies between the output variables introduced during core relaxation require extra care when instantiating OLL in UnimaxSAT. More specifically, to ensure that \( \gamma^A = \{\bar{x} \mid x \in \text{var}(O')\} \) is an abstract candidate in each iteration, we include an assumption refinement step as part of ABS-CANDIDATE. After receiving an abstraction set for a core \( C \), ABS-CANDIDATE with assumption refinement checks if \( \gamma^A \) can be extended into a solution to \( \text{DEF}(AB) \cup K \), i.e., the definitions of the current collection \( AB \) of abstraction sets and set \( K \) of abstract cores. If it can, it is returned as an abstract candidate, otherwise a new core is obtained, added to \( K \), and relaxed as other cores.

Important for understanding assumption refinement is that the core extraction step performed during it do not consider the input clauses. Even so, all cores extracted during assumption refinement remain abstract cores of the input instance, since the cores are satisfied by all solutions to the definitions of the current abstraction sets and cores, which in turn are satisfied by all solutions to the input instance. Each new core discovered during assumption refinement decreases the number of literals in \( O' \) by at least one. Thus the procedure is guaranteed to terminate eventually.

Correctness of OLL now follows by Theorem 1 by arguing that ADD-ABSETS with assumption refinement results in a partial assignment \( \gamma^A \) that is an abstract candidate of \( \text{DEF}(AB) \cup K \). We sketch a proof of this for an arbitrary unweighted MaxSAT instance \( \mathcal{F} = (P, O) \) (all objective coefficients 1; the proof for weighted objectives is similar).

In the following, let \( \text{INACTIVE} \) be the set of objective variables and abstraction set outputs that have not been encountered in any cores during the first \( i - 1 \) iterations. Then \( \text{INACTIVE} = \text{var}(O') \) where \( O' \) is the reformulated objective (maintained by ABS-CANDIDATE) during iteration \( i \). We say that the literals in \( \text{INACTIVE} \) are inactive. Now consider an iteration \( i \) of the UnimaxSAT instantiated as OLL and let \( K^i \) be the set of abstract cores obtained so-far (either from EX-ABSCORE or from assumption refinement) and \( AB^i \) the set of all abstraction sets corresponding to cardinality constraints introduced so-far. We show that the partial assignment \( \gamma^A = \{\bar{x} \mid x \in \text{INACTIVE} \} \) that assigns all inactive literals in \( \text{INACTIVE} \) to 0 is an abstract candidate of \( K^i \). The assumption refinement step guarantees that there is at least one extension of \( \gamma^A \) into a solution to \( \text{DEF}(AB^i) \cup K \). Thus what remains to show is that all such extensions will be minimum-cost. For this, let \( \tau \) be a solution of \( \text{DEF}(AB^i) \cup K \). We argue that: (i) \( O(\tau) \geq |K| \) and, (ii) \( O(\tau) \leq |K| \) if \( \tau \geq \gamma^A \).

Since the output literals of the abstraction sets are defined by cardinality constraints, each new output literal assigned to 1 will result in one more objective variable being assigned to 1, thus incurring cost. Now (i) follows by observing that \( \tau \) assigns at least one literal in each \( C \in K^i \) to 1. (ii) follows from the fact that each new abstract core obtained during search is always falsified when assigning the current set of inactive literals to 0. The abstraction set corresponding to a cardinality constraint that is introduced after each core allows exactly one more of the previously inactive literals to be assigned to 1 in subsequent iterations.

Finally, we outline instantiation of UnimaxSAT which exactly correspond to other modern core-guided algorithms.

MSU3 is obtained by instantiating UnimaxSAT as follows. ADD-ABSETS maintains a set \( \text{INACTIVE} \) of objective literals that have not appeared in any cores. Given a new core \( C \) on iteration \( i \) all objective literals in \( C \) are removed from \( \text{INACTIVE} \). A new abstraction set \( AB = (\text{active}, \text{asCNF}(\sum_{x \in \text{active}} x \geq i + 1 \iff o_i), \{o_i\}) \) where \( \text{active} = \text{var}(O) \setminus \text{INACTIVE} \) and \( o_i \) is a fresh variable, is then introduced. The instantiation of ABS-CANDIDATE returns the abstract candidate \( \gamma^A = \{\bar{x} \mid x \in \text{INACTIVE}\} \cup \{o_i\} \).

WPM3 is obtained very similarly as MSU3. However, instead of ADD-ABSETS maintaining a single abstraction set with all objective variables not in \( \text{INACTIVE} \) as inputs, it maintains several abstraction sets with different disjoint subsets of the variables \( x \in \text{var}(O) \). Whenever a core containing the outputs of two different abstraction sets \( AB_1 = (i_{11}, \text{asCNF}(\sum_{x \in \text{int}} x \geq k \iff o_{1k}), \{o_{1k}\}) \) and \( AB_2 = (i_{22}, \text{asCNF}(\sum_{x \in \text{int}} x \geq t \iff o_{2t}), \{o_{2t}\}) \) is extracted by EX-ABCORE, a new abstraction set \( (i_{11} \cup i_{22}, \text{asCNF}(\sum_{x \in \text{int}} x \geq (k + t + 1) \iff o_{k+t+1}), \{o_{k+t+1}\}) \) is introduced; and \( AB_1, AB_2 \) removed (i.e., ignored by ABS-CANDIDATE).

PMRES is obtained by ADD-ABSETS introducing for every core \( C = (x_1, \ldots, x_n) \) an abstraction set \( AB = (C, \{x_i \land (x_{i+1} \lor \cdots \lor x_n) \mid 1 \leq i \leq n - 1\}, \{o_1, \ldots, o_{n-1}\}) \). The ABS-CANDIDATE simulates the objective reformulation steps of PMRES, which are very similar to how OLL refor-
mulates the objective. In contrast to OLL, an assumption refinement step is not needed for simulating PMRES; this is essentially due to fact that the abstraction sets introduced do not introduce dependencies that could lead to cores independent of the input clauses.

K combines the cardinality constraints used by OLL and PMRES, and can hence be obtained from UniMAXSAT by the preceding discussion on OLL and PMRES.

6 Formulating New Algorithms

To highlight further the 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, helping the usefulness of UniMAXSAT in developing correct new MaxSAT algorithms.

ABSTCG differs from OLL in how a core C containing variables with different coefficients in the reformulated objective O′ are handled. When the literals in C contain m distinct coefficients in O′, C is partitioned into disjoint sets C = G1 ∪ ... ∪ Gm so that all variables in a same set Gi have the same coefficients and the sets are ordered by decreasing coefficients. Starting from G1 (corresponding to the largest coefficient in O′), ABSTCG introduces in order for each Gi an abstraction set ABi = (ini, {ascNF(∑x∈ini, x ≥ j ↔ oj′) | 1 ≤ j ≤ |ini|, outi}). The inputs ini = Gi ∪ outi−1 consist of the variables in Gi and the outputs of ABi−1. Since C is an abstract core, at least one of its variables has to be assigned to 1 by any solution; the first output of the last abstraction set o1m is not be introduced.

To compute the next abstract candidate, ABSTCG updates the reformulated objective O′ by processing each abstraction set ABi = (in, D, out) very similarly to OLL. Starting from i = 1 the coefficient of each x ∈ in is lowered by wi = min{|O′(x)| x ∈ ini} and each output x ∈ out is included in O′ with coefficient wi. The inputs of each abstract set consist solely of objective variables and outputs of the previous level; hence this procedure is well-defined. Finally, the set γA = {x | x ∈ var(O′)} consisting of the negations of literals in O′ with non-zero coefficients is returned. The correctness of the procedure follows by showing that γA is an abstract candidate, by similar arguments as for OLL.

Example 7. Invoke ABSTCG on F = (F, O) from Example 1 and assume the first core obtained is C1 = {b1 ∨ b2 ∨ b3 ∨ b4 ∨ b5}. Core relaxation divides the variables in this core into G1 = {b3}, G2 = {b5} and G3 = {b1, b2, b4} and G2 = {b1, b2, b4}. The abstraction step over G1 has one output variable o11 defined by ascNF(∑x∈G1, x ≥ 1 ↔ o11). (Practical implementations would use the variable b1 directly as o11.) The abstraction step over G2 has b5 and o11 as inputs and two output variables o21 defined by ascNF(∑x∈[b5, o11], x ≥ i ↔ o21) for i = 1, 2. The abstraction step over G3 has G3 ∪ {o21, o22} as inputs, 4 output variables o31 defined by ascNF(∑x∈G3∪[o21, o22], x ≥ i ↔ o31) for i = 2...5. After reformulation, the objective O′ is O′ ≡ o11 + o21 + o22 + o31 + o32 + o33 + o34 + o35 and the abstract candidate returned in the next iteration contains the negation of all these variables.

We developed an open-source implementation (available at https://bitbucket.org/coreo-group/cgss2/src/abstcg/) of ABSTCG on top of CGSS2, a state-of-the-art C++ implementation of OLL when compared to solvers in the 2022 MaxSAT Evaluations [Ihalainen, 2022]. To take advantage of the special feature of ABSTCG, the implementation invokes ABSTCG when an instance includes at least three different weights on the soft clauses and the median number of soft clauses for each weight is more than 25 (as a heuristic for instances that likely to contain cores which ABSTCG can divide into weight-sets in a meaningful way), and otherwise OLL.

We empirically compare the runtimes of this prototype to those of CGSS2 (employing OLL) on all 607 weighted instances from MaxSAT Evaluation 2022 using 2.6-GHz Intel Xeon E5-2670 processors under per-instance 3600-s time and 32-GB memory limit. For the 237 instances on which the prototype heuristically invoked ABSTCG (see Figure 2), ABSTCG solved 125 instances, OLL 124. ABSTCG reduced runtime by at least 2x for 13 instances, against OLL being 2x as fast on only 4 instances. The competitiveness of the prototype suggests as a proof of concept that UniMAXSAT allows for novel algorithmic instantiations that can also be interesting from the perspective of practical solvers.

7 Conclusions

We developed a general algorithmic framework that captures in a unifying way the computations of variants of core-guided and implicit hitting sets algorithms for MaxSAT. The correctness of the framework provides a uniform way of proving the correctness of various unsatisfiability-based MaxSAT algorithms. 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 also from a practical perspective interesting solver variant for MaxSAT. Beyond MaxSAT, the framework can also be similarly instantiated for related constraint optimization paradigms for which core-guided and IHS style solvers can and have been developed based on the constraint-agnostic notion of unsatisfiable cores.
Acknowledgements
This work has been financially supported by University of Helsinki Doctoral Programme in Computer Science DoCS and Academy of Finland under grants 322869 and 342145. The authors wish to thank the Finnish Computing Competence Infrastructure (FCCI) for supporting this project with computational and data storage resources.

References


