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
firstname.surname@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 Planes, 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 Gabàs, 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 [Eén 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 Bjørner, 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

Dodaro, 2020]: core-guided and IHS approaches rely on a 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; Larrosa *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 $C = l_1 \vee \ldots \vee l_n$ is a disjunction of literals, a conjunctive normal form (CNF) formula $F = \{C_1, \ldots, C_m\}$ is a set of clauses. The set $\mathtt{var}(C)$ consists of the 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 in the standard way: $\tau(\bar{l}) = 1 - \tau(l)$, $\tau(C) = \max\{\tau(l) \mid l \in C\}$ and $\tau(F) = \min\{\tau(C) \mid C \in F\}$. τ satisfies F if $\tau(F) = 1$. We may treat τ as the set of literals τ assigns to 1, so that $l \in \tau$ denotes $\tau(l) = 1$ and $\bar{l} \in \tau$ denotes $\tau(l) = 0$. τ 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 τ if $\sum_i \tau(x_i) \geq k$. We do not make assumptions on the CNF encoding used for cardinality constraints, and abstractly use $\textsc{asCNF}(\sum_i x_i \geq k)$ to denote a CNF formula that is satisfiable by an assignment τ iff $\sum_i \tau(x_i) \geq k$. Taking a name o to indicate whether a cardinality constraint is satisfied, we also use $\textsc{asCNF}(\sum_i x_i \leq k \leftrightarrow o)$ to denote a CNF formula that is satisfied by any assignment τ 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 Piotrów, 2019].

An instance $\mathcal{F} = (F, O)$ of maximum satisfiability (MaxSAT) consists of a CNF formula F and an objective function $O \equiv \sum_i w_i b_i$, 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 $\mathtt{var}(O)$ consists of the variables that appear in O. A complete satisfying assignment τ to F is a solution to $\mathcal{F}$ 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 $\textsc{opt}(\mathcal{F})$.

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

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., $\mathtt{var}(C) \subseteq \mathtt{var}(O)$) and every solution to F satisfies C (i.e., F logically entails C).

Algorithm 1 Core-guided approach to MaxSAT

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

1: $\textsc{Constrs} = \emptyset$
2: $O' = O$
3: **while** true **do**
4: $\quad \gamma^A = \{\bar{x} \mid x \in \mathtt{var}(O')\}$
5: $\quad (\text{res}, C, \tau) = \textsc{Extract-Core}(F \cup \textsc{Constrs}, \gamma^A)$
6: $\quad$ **if** res ='true' **then return** τ
7: $\quad (D, out) = \textsc{Create-Card-Constr}(C)$
8: $\quad \textsc{Constrs} = \textsc{Constrs} \cup D$
9: $\quad O' = \textsc{Refine-Objective}(C, out, O')$

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; Narodytska and Bacchus, 2014; Alviano *et al.*, 2015; Ansótegui and Gabàs, 2017] and implicit hitting set algorithms [Davies and Bacchus, 2011; Davies and Bacchus, 2013; Saikko *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 γ^A (constituting a set of assumptions, as a set of literals), we abstract the SAT solver into the subroutine $\textsc{Extract-Core}$ that returns a triplet (res, C, τ). Here res=true if there is a satisfying assignment $\tau \supseteq \gamma^A$ of F. If there is no such assignment, res=false and C is a clause over a subset of the variables in γ^A entailed by F for which $\gamma^A(C) = 0$. Invoked on F under a set of assumptions γ^A such that $F \wedge \gamma^A$ is unsatisfiable, modern SAT solvers provide such C at termination without computational overhead. In core-guided and IHS MaxSAT solving, $\textsc{Extract-Core}$ is used for extracting cores of a MaxSAT instance $\mathcal{F} = (F, O)$ by invoking it on F under a subset of $\mathtt{var}(O)$ as the assumptions.

Core-Guided Search. Algorithm 1 details a general abstraction of the core-guided approach to computing an optimal solution to a MaxSAT instance $\mathcal{F} = (F, O)$. First the algorithm initializes a set $\textsc{Constrs}$ 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 τ that (i) satisfies all clauses in F and all of the cardinality constraints in $\textsc{Constrs}$ 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 τ, it is returned as an optimal solution to the original instance (Line 6). Otherwise a core C of $(F \cup \textsc{Constrs}, O')$ is obtained. The core is then *relaxed* (Lines 7–9), transforming the current instance in a way that

Algorithm 2 IHS for MaxSAT

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

```
1: 𝒦 = ∅
2: while true do
3:     γ^A = {x̄ | x ∈ var(O) \ MINCOST-HS(𝒦)}
4:     (res, C, τ) = EXTRACT-CORE(F, γ^A)
5:     if res = 'true' then return τ
6:     else 𝒦 = 𝒦 ∪ C
```

enables (at most) one variable in the core C to incur cost in subsequent iterations. This is done by adding a new cardinality constraint over the core to CONSTRS (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 [Morgado *et al.*, 2014; Andres *et al.*, 2012] as arguably one of the most successful core-guided approaches. In OLL CREATE-CARD-CONSTR(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 $out = \{o_2^C \ldots o_{|C|}^C\}$ of *output* variables. Intuitively, the new cardinality constraints define output variables that count the number of literals in C assigned to 1 in subsequent iterations, since enforcing o_k^C to 0 limits the number of literals in C assigned to 1 to at most $k-1$. The output variable with index 1 is not introduced; since C is a core, every satisfying assignment assigns at least one literal to 1. In the objective reformulation step (REFINE-OBJECTIVE procedure of Algorithm 1), OLL adds the new outputs to the objective in a way that preserves the set of optimal solutions. The coefficient of each $x \in C$ is decreased by $w^C = \min_{x \in C_i}\{O'(x)\}$ (removing from O' every literal whose coefficient decreases to 0). Informally, the termination of OLL follows from observing that at least one literal bis removed from O' on each iteration, which allows the SAT solver to assign at least one more objective variable to true in subsequent iterations.

Example 2. *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 $i = 2, 3, 4, 5$ and the new objective is $O' = 2b_3 + b_5 + o_2^1 + o_3^1 + o_4^1 + o_5^1$. 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 $O' = b_3 + b_5 + o_3^1 + o_4^1 + o_5^1 + o_2^2$. The assumptions in the 3rd call to* EXTRACT-CORE *are $\gamma^A = \{\bar{b_3}, \bar{b_5}, \bar{o}_3^1, \bar{o}_4^1, \bar{o}_5^1, \bar{o}_2^2\}$. Now e.g. the assignment $\tau = \{b_1, \bar{b_2}, b_3, b_4, b_5\}$ (that also assigns $\{o_2^1, \bar{o}_3^1, \bar{o}_4^1, \bar{o}_5^1, \bar{o}_2^2\}$) is returned as an optimal solution to $\mathcal{F}$.*

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 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 the variables in hs, assigned to 1, satisfy all cores in $\mathcal{K}$. 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). If there is such a τ, 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 γ^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 $\mathcal{F} = (F, O)$ from Example 1. In the first iteration there are no cores, so* 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 iteration 2 there are three different MCHSs over $\mathcal{K} = \{C_1\}$. Assume that* MINCOST-HS *returns $\{b_1\}$. Then 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 is $C_2 = (b_3 \vee b_4 \vee b_5)$. In iteration 3 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 $C_3 = (b_1, b_2, b_3)$. In iteration 4 there are two possible MCHSs over $\mathcal{K} = \{C_1, C_2, C_3\}$. Assume that* MINCOST-HS *returns $\{b_2, b_4\}$, leading to the assumptions $\gamma^A = \{\bar{b_1}, \bar{b_3}, \bar{b_5}\}$. Now* EXTRACT-CORE *returns the solution $\tau = \{\bar{b_1}, b_2, \bar{b_3}, b_4, \bar{b_5}\}$ as an optimal solution to $\mathcal{F}$.*

4 Unifying CG and IHS

As our main contribution we present 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 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* 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 $in \cup out$, i.e., $\mathtt{var}(in \cup out) \subseteq \mathtt{var}(D)$. 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.*

Given an abstraction set AB $= (in, D, out)$ we call D 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, DEF$(\mathcal{AB}) = \bigcup_{i=1}^n D_i$ is the CNF formula consisting of the definitions in $\mathcal{AB}$ and OUTS$(\mathcal{AB}) = \bigcup_{i=1}^n out_i$ is the set of

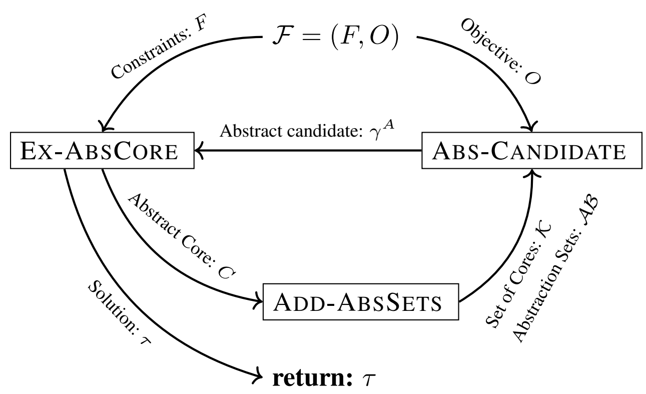

Figure 1: Schematic overview of UNIMAXSAT

outputs occurring in $\mathcal{AB}$. We say that $\mathcal{AB}$ is *feasible* for a MaxSAT instance $\mathcal{F} = (F, O)$ if $\mathrm{DEF}(\mathcal{AB})$ does not change the set of solutions to $\mathcal{F}$, i.e., if every solution τ to F can be uniquely extended into a solution $\tau^E \supseteq \tau$ to $F \cup \mathrm{DEF}(\mathcal{AB})$. 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 set. 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 $\mathcal{AB}$ of feasible abstraction sets, a clause C is an* abstract core *of $\mathcal{F}$ wrt $\mathcal{AB}$ if (i) $\mathtt{var}(C) \subseteq (\mathtt{var}(O) \cup \mathtt{var}(\mathrm{OUTS}(\mathcal{AB}))$ and (ii) $\tau(C) = 1$ for each solution τ of $F \cup \mathrm{DEF}(\mathcal{AB})$.*

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 (F, O) from Example 1 and the abstraction set* $\textsc{ab} = (\{b_1, \ldots, b_5\}, \{\mathrm{ASCNF}(\sum_{i=1}^5 b_i \geq j \leftrightarrow o_j) \mid j = 2, 3, 4, 5\}, \{o_2, \ldots, o_5\})$*. Then $C = (o_2 \vee b_3)$ is an abstract core as any satisfying assignment of $F \cup \mathrm{DEF}(\textsc{ab})$ 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 (F, O), a collection $\mathcal{AB}$ of abstraction sets and a set $\mathcal{K}$ of abstract cores, a complete satisfying assignment δ of $\mathcal{K} \cup \mathrm{DEF}(\mathcal{AB})$ that assigns each variable in $\mathtt{var}(O)$ is a *candidate* (solution) to $\mathcal{K}$ with cost $O(\delta)$. δ is minimum-cost if $O(\delta) \leq O(\delta')$ for all candidates δ' of $\mathcal{K}$.

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, $\mathcal{K}$ a set of abstract cores, and δ a minimum-cost candidate of $\mathcal{K}$. Then $O(\delta) \leq \textsc{opt}(\mathcal{F})$.*

Algorithm 3 UNIMAXSAT, a unified framework for core-guided and IHS-based MaxSAT algorithms.

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

$\mathcal{AB}^1 = \emptyset, \mathcal{K}^1 = \emptyset$
for $i = 1 \ldots$ **do**
  $\gamma^A = \textsc{Abs-Candidate}(\mathcal{AB}^i, \mathcal{K}^i)$
  $(\mathrm{res}, C, \tau) = \textsc{Ex-AbsCore}(F, \mathrm{DEF}(\mathcal{AB}^i), \gamma^A)$
  if res = 'true' **then return** τ
  $\mathcal{K}^{i+1} = \{C\} \cup \mathcal{K}^i$
  $\mathcal{AB}^{i+1} = \mathcal{AB}^i \cup \textsc{Add-AbsSets}(\mathcal{F}, \mathcal{K}^{i+1})$

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, $\mathcal{AB}$ a collection of feasible abstraction sets and $\mathcal{K}$ a set of abstract cores. A partial assignment γ^A over a subset of the variables in $\mathtt{var}(\mathcal{K}) \cup \mathtt{var}(O)$ is an abstract candidate of $\mathcal{K}$ if (i) there is at least one extension $\tau \supseteq \gamma^A$ which is a solution of $\mathrm{DEF}(\mathcal{AB}) \cup \mathcal{K}$, i.e., a candidate of $\mathcal{K}$ and (ii) all such extensions are minimum-cost candidates of $\mathcal{K}$.*

While each minimum-cost candidate of $\mathcal{K}$ 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 $\mathcal{K} = \{(b_1 \vee b_2 \vee b_3), (b_3 \vee b_4 \vee b_5)\}$ of abstract cores. A minimum-cost candidate of $\mathcal{K}$ is $\delta = \{\bar{b}_1, b_2, \bar{b}_3, b_4, \bar{b}_5\}$. An abstract candidate is $\gamma^A = \{\bar{b}_1, \bar{b}_3, \bar{b}_5\}$ since the only extension of γ^A into a solution to $\mathcal{K}$ is δ. The set $\{\bar{b}_1, \bar{b}_3\}$ is not an abstract candidate since it extends to the solution $\{\bar{b}_1, b_2, \bar{b}_3, \bar{b}_4, b_5\}$ of $\mathcal{K}$ 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 $\mathcal{K} = \{b_1 \vee b_2 \vee b_3 \vee b_4 \vee b_5\}$, and the abstraction set* AB *detailed in Example 4. The set $\gamma^A = \{\bar{b}_3, \bar{b}_5, \bar{o}_2, \bar{o}_3, \bar{o}_4, \bar{o}_5\}$ is an abstract candidate as it can be extended into a solution to $\mathcal{K} \cup \mathrm{DEF}(\{\textsc{ab}\})$ 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 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 $\mathcal{AB}$ and $\mathcal{K}$ of abstraction sets and abstract cores, respectively. In each iteration, an abstract candidate γ^A of $\mathcal{K}$ is computed by the ABS-CANDIDATE subroutine. The EX-ABSCORE subroutine is invoked to check for an extension of γ^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 γ^A is obtained. The core is added to $\mathcal{K}$, thus blocking γ^A and all of its extensions from further consideration, and the ADD-ABSSETS subroutine adds new abstraction sets to $\mathcal{AB}$.

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 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 $\mathcal{F} = (F, O)$ be an input instance to* UNIMAXSAT *that has solutions. Assume that the following three properties hold on every iteration i of* UNIMAXSAT.

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

Then UNIMAXSAT *terminates and returns an optimal solution of $\mathcal{F}$.*

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 $\mathcal{K}^i$ and $\mathcal{AB}^i$ be the set of abstract cores and abstraction sets obtained so-far. Denote by* OBJ-SOLS^i *the restrictions of all solutions to $\mathcal{K}^i \cup \text{DEF}(\mathcal{AB}^i)$ onto $var(O)$. Then* OBJ-SOLS$^{i+1} \subsetneq$ OBJ-SOLS^i.

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

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

Now consider termination. As $\mathcal{F}$ has solutions and $\mathcal{AB}^i$ is feasible, $F \cup \text{DEF}(\mathcal{AB}^i)$ has solutions for all i. By definition of abstract cores, all solutions to $F \cup \text{DEF}(\mathcal{AB}^i)$ are solutions to $\text{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$. In each iteration, the abstract candidate γ^A obtained from ABS-CANDIDATE can be extended into at least one solution to $\text{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$. We argue that eventually γ^A can also be extended into a solution to $F \cup \text{DEF}(\mathcal{AB}^i)$. 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 $F \cup \text{DEF}(\mathcal{AB}^i)$ 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 $F \cup \text{DEF}(\mathcal{AB}^i)$, 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 Gabàs, 2017], PMRES [Narodytska and Bacchus, 2014] and K [Alviano *et al.*, 2015] are obtained.

The EX-ABSCORE subroutine of UNIMAXSAT is in general a core-extracting SAT solver: given a MaxSAT instance $\mathcal{F} = (F, O)$, a feasible collection $\mathcal{AB}$ of abstraction sets and an abstract candidate γ^A, EX-ABSCORE invokes a SAT solver on $F \cup \text{DEF}(\mathcal{AB}^i)$ under assumptions γ^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-ABSSETS, 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-ABSSETS to never add any abstraction sets and EX-ABSCORE as a procedure that–given a set $\mathcal{K}$ of cores—returns the abstract candidate $\gamma^A = \{\bar{x} \mid x \in O \setminus \text{MINCOST-HS}(\mathcal{K})\}$ that assigns all literals in the objective to 0 except the ones in a most recent minimum-cost hitting set MINCOST-HS$(\mathcal{K})$ over $\mathcal{K}$. The correctness of Algorithm 2 now follows by Theorem 1 by observing that the only extension of γ^A into a candidate of $\mathcal{K}$ is $\{\bar{x} \mid x \in O \setminus \text{MINCOST-HS}(\mathcal{K})\} \cup \{x \mid x \in \text{MINCOST-HS}(\mathcal{K})\}$ and is minimum-cost. Hence γ^A is an abstract candidate of $\mathcal{K}$.

IHS with abstract cores. UNIMAXSAT gives IHS enhanced with abstract cores by instantiating EX-ABSCORE as in basic IHS, and ADD-ABSSETS 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 $D = \{\textsc{asCNF}(\sum_{x \in in} x \geq i \leftrightarrow o_i) \mid k = 1 \ldots n\}$, 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 γ to $\mathcal{K} \cup \mathrm{DEF}(\mathcal{AB})$ that assigns all variables in $\mathtt{var}(O) \cup \mathrm{OUTS}(\mathcal{AB})$, and then returns either γ_1^A as the restriction of γ onto the objective variables, 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. 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 $\mathcal{K} \cup \mathrm{DEF}(\mathcal{AB})$ is γ, γ_1^A is an abstract candidate. Finally, γ_2^A is an abstract candidate since (i) γ is an extension of γ_2^A into a solution of $\mathcal{K} \cup \mathrm{DEF}(\mathcal{AB})$ 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-ABSSETS relaxing a core C introduces abstraction set $\textsc{ab}^C = (C, D, out)$ where $D = \{\textsc{asCNF}(\sum_{x \in C} x \geq i \leftrightarrow o^i) \mid 2 \leq i \leq |C|\}$ 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'.

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 $\gamma^A = \{\bar{x} \mid x \in \mathtt{var}(O')\}$ 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 γ^A can be extended into a solution to $\mathrm{DEF}(\mathcal{AB}) \cup \mathcal{K}$, i.e., the definitions of the current collection $\mathcal{AB}$ of abstraction sets and set $\mathcal{K}$ of abstract cores. If it can, it is returned as an abstract candidate, otherwise a new core is obtained, added to $\mathcal{K}$, and relaxed as other cores.

Important for understanding assumption refinement is that the core extraction steps 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-ABSSETS with assumption refinement results in a partial assignment γ^A that is an abstract candidate of $\mathrm{DEF}(\mathcal{AB}) \cup \mathcal{K}$. We sketch a proof of this for an arbitrary unweighted MaxSAT instance $\mathcal{F} = (F, O)$ (all objective coefficients 1; the proof for weighted objectives is similar).

In the following, let $\textsc{inactive}^i$ 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 $\textsc{inactive}^i = \mathtt{var}(O')$ where O' is the reformulated objective (maintained by ABS-CANDIDATE) during iteration i. We say that the literals in $\textsc{inactive}^i$ are *inactive*. Now consider an iteration i of the UNIMAXSAT instantiated as OLL and let $\mathcal{K}^i$ be the set of abstract cores obtained so-far (either from EX-ABSCORE or from assumption refinement) and $\mathcal{AB}^i$ the set of all abstraction sets corresponding to cardinality constraints introduced so-far. We show that the partial assignment $\gamma^A = \{\bar{x} \mid x \in \textsc{inactive}^i\}$ that assigns all inactive literals in $\textsc{inactive}^i$ to 0 is an abstract candidate of $\mathcal{K}^i$. The assumption refinement step guarantees that there is at least one extension of γ^A into a solution to $\mathrm{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$. Thus what remains to show is that all such extensions will be minimum-cost. For this, let τ be a solution of $\mathrm{DEF}(\mathcal{AB}^i) \cup \mathcal{K}^i$. We argue that: (i) $O(\tau) \geq |\mathcal{K}^i|$ and, (ii) $O(\tau) \leq |\mathcal{K}^i|$ if $\tau \supseteq \gamma^A$.

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 τ assigns at least one literal in each $C \in \mathcal{K}^i$ 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-ABSSETS maintains a set 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 INACTIVE. A new abstraction set $\textsc{ab} = (\textsc{active}, \textsc{asCNF}(\sum_{x \in \textsc{active}} x \geq i+1 \leftrightarrow o_i), \{o_i\})$ where $\textsc{active} = \mathtt{var}(O) \setminus \textsc{inactive}$ and o_i is a fresh variable, is then introduced. The instantiation of ABS-CANDIDATE returns the abstract candidate $\gamma^A = \{\bar{x} \mid x \in \textsc{inactive}\} \cup \{\bar{o}_i\}$.

WPM3 is obtained very similarly as MSU3. However, instead of ADD-ABSSETS maintaining a single abstraction set with all objective variables not in INACTIVE as inputs, it maintains several abstraction sets with different disjoint subsets of the variables $x \in \mathtt{var}(O) \setminus \textsc{inactive}$. Whenever a core containing the outputs of two different abstraction sets $\textsc{ab}_1 = (in_1, \textsc{asCNF}(\sum_{x \in in_1} x \geq k \leftrightarrow o_k^1), \{o_k^1\})$ and $\textsc{ab}_2 = (in_2, \textsc{asCNF}(\sum_{x \in in_2} x \geq t \leftrightarrow o_t^2), \{o_t^2\})$ is extracted by EX-ABSCORE, a new abstraction set $(in_1 \cup in_2, \textsc{asCNF}(\sum_{x \in in_1 \cup in_2} x \geq (k+t+1) \leftrightarrow o_{k+t+1}^3), \{o_{k+t+1}^3\})$ is introduced; and $\textsc{ab}_1, \textsc{ab}_2$ removed (i.e., ignored by ABS-CANDIDATE).

PMRES is obtained by ADD-ABSSETS introducing for every core $C = (x_1, \ldots, x_n)$ an abstraction set $\textsc{ab} = (C, \{x_i \wedge (x_{i+1} \vee \cdots \vee x_n) \leftrightarrow o_i \mid 1 \leq i \leq n-1\}, \{o_1, \ldots, o_{n-1}\})$. The ABS-CANDIDATE simulates the objective reformulation steps of PMRES, which are very similar to how OLL reformulates 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, highlighting 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 $C = G_1 \cup \ldots \cup G_m$ so that all variables in a same set G_i have the same coefficients and the sets are ordered by decreasing coefficients. Starting from G_1 (corresponding to the largest coefficient in O'), AbstCG introduces in order for each G_i an abstraction set $\text{AB}_i = (in_i, \{\text{ASCNF}(\sum_{x \in in_i} x \geq j \leftrightarrow o^i_j) \mid 1 \leq j \leq |in_i|\}, out_i)$. The inputs $in_i = G_i \cup out_{i-1}$ consist of the variables in G_i and the outputs of AB_{i-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 o^m_1 is not be introduced.

To compute the next abstract candidate, AbstCG updates the reformulated objective O' by processing each abstraction set $\text{AB}_i = (in, D, out)$ very similarly to OLL. Starting from $i = 1$ the coefficient of each $x \in in$ is lowered by $w^i = \min(\{O'(x) \mid x \in in\})$ and each output $x \in out$ is included in O' with coefficient w^i. 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 $\gamma^A = \{\bar{x} \mid x \in \mathtt{var}(O')\}$ consisting of the negations of literals in O' with non-zero coefficients is returned. The correctness of the procedure follows by showing that γ^A is an abstract candidate, by similar arguments as for OLL.

Example 7. *Invoke* AbstCG *on $\mathcal{F} = (F, O)$ from Example 1 and assume the first core obtained is $C_1 = (b_1 \vee b_2 \vee b_3 \vee b_4 \vee b_5)$. Core relaxation divides the variables in this core into $G_1 = \{b_3\}$, $G_2 = \{b_5\}$ and $G_3 = \{b_1, b_2, b_4\}$ and $G_2 = \{b_1\}$. The abstraction set over G_1 has one output variable o^1_1 defined by $\text{ASCNF}(\sum_{x \in G_1} x \geq 1 \leftrightarrow o^1_1)$. (Practical implementations would use the variable b_3 directly as o^1_1.) The abstraction set over G_2 has b_5 and o^1_1 as inputs and two output variables o^2_i defined by $\text{ASCNF}(\sum_{x \in \{b_5, o^1_1\}} x \geq i \leftrightarrow o^2_i)$ for $i = 1, 2$. The abstraction set over G_3 has $G_3 \cup \{o^2_1, o^2_2\}$ as inputs, 4 output variables o^3_i defined by $\text{ASCNF}(\sum_{x \in G_3 \cup \{o^2_1, o^2_2\}} x \geq i \leftrightarrow o^3_i)$ for $i = 2 \ldots 5$. After reformulation, the objective O' is $O' \equiv o^1_1 + o^2_1 + o^2_2 + o^3_2 + o^3_3 + o^3_4 + o^3_5$ and the abstract candidate returned in the next iteration contains the negation of all these variables.*

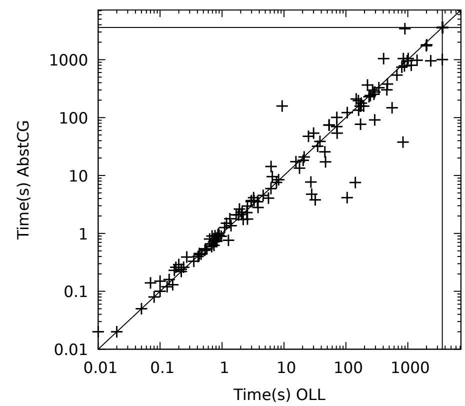

Figure 2: Runtime comparison of OLL and AbstCG

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

[Abío *et al.*, 2013] Ignasi Abío, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. A parametric approach for smaller and better encodings of cardinality constraints. In *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013*, volume 8124 of *Lecture Notes in Computer Science*, pages 80–96. Springer, 2013.

[Alviano and Dodaro, 2020] Mario Alviano and Carmine Dodaro. Unsatisfiable core analysis and aggregates for optimum stable model search. *Fundam. Informaticae*, 176(3-4):271–297, 2020.

[Alviano *et al.*, 2015] Mario Alviano, Carmine Dodaro, and Francesco Ricca. A MaxSAT algorithm using cardinality constraints of bounded size. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 2677–2683. AAAI Press, 2015.

[Andres *et al.*, 2012] Benjamin Andres, Benjamin Kaufmann, Oliver Matheis, and Torsten Schaub. Unsatisfiability-based optimization in clasp. In *Technical Communications of the 28th International Conference on Logic Programming, ICLP 2012*, volume 17 of *LIPIcs*, pages 211–221. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.

[Ansótegui and Gabàs, 2017] Carlos Ansótegui and Joel Gabàs. WPM3: An (in)complete algorithm for weighted partial MaxSAT. *Artificial Intelligence*, 250:37–57, 2017.

[Ansótegui *et al.*, 2013] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artif. Intell.*, 196:77–105, 2013.

[Ansótegui *et al.*, 2016] Carlos Ansótegui, Joel Gabàs, and Jordi Levy. Exploiting subproblem optimization in SAT-based MaxSAT algorithms. *J. Heuristics*, 22(1):1–53, 2016.

[Asín *et al.*, 2011] Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality networks: A theoretical and empirical study. *Constraints An Int. J.*, 16(2):195–221, 2011.

[Bacchus and Narodytska, 2014] Fahiem Bacchus and Nina Narodytska. Cores in core based MaxSat algorithms: An analysis. In *Theory and Applications of Satisfiability Testing - SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 7–15. Springer, 2014.

[Bacchus *et al.*, 2019] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. MaxSAT Evaluation 2018: New developments and detailed results. *J. Satisf. Boolean Model. Comput.*, 11(1):99–131, 2019.

[Bacchus *et al.*, 2021] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. *Maximum Satisfiability*, chapter 24, pages 929–991. Frontiers in Artificial Intelligence and Applications. IOS Press BV, 2021.

[Bailleux and Boufkhad, 2003] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In *Principles and Practice of Constraint Programming - CP 2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003.

[Berg and Järvisalo, 2019] Jeremias Berg and Matti Järvisalo. Unifying reasoning and core-guided search for maximum satisfiability. In *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019*, volume 11468 of *Lecture Notes in Computer Science*, pages 287–303. Springer, 2019.

[Berg *et al.*, 2020] Jeremias Berg, Fahiem Bacchus, and Alex Poole. Abstract cores in implicit hitting set MaxSat solving. In *Theory and Applications of Satisfiability Testing - SAT 2020*, volume 12178 of *Lecture Notes in Computer Science*, pages 277–294. Springer, 2020.

[Codish and Zazon-Ivry, 2010] Michael Codish and Moshe Zazon-Ivry. Pairwise cardinality networks. In *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16*, volume 6355 of *Lecture Notes in Computer Science*, pages 154–172. Springer, 2010.

[Davies and Bacchus, 2011] Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *Principles and Practice of Constraint Programming - CP 2011*, volume 6876 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2011.

[Davies and Bacchus, 2013] Jessica Davies and Fahiem Bacchus. Postponing optimization to speed up MAXSAT solving. In *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013*, volume 8124 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 2013.

[Delisle and Bacchus, 2013] Erin Delisle and Fahiem Bacchus. Solving weighted CSPs by successive relaxations. In *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013*, volume 8124 of *Lecture Notes in Computer Science*, pages 273–281. Springer, 2013.

[Devriendt *et al.*, 2021] Jo Devriendt, Stephan Gocht, Emir Demirovic, Jakob Nordström, and Peter J. Stuckey. 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*, pages 3750–3758. AAAI Press, 2021.

[Eén and Sörensson, 2003] Niklas Eén and Niklas Sörensson. Temporal induction by incremental SAT solving. In *First International Workshop on Bounded Model Checking, BMC@CAV 2003*, volume 89 of *Electronic Notes in Theoretical Computer Science*, pages 543–560. Elsevier, 2003.

[Eén and Sörensson, 2006] Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into SAT. *J. Satisf. Boolean Model. Comput.*, 2(1-4):1–26, 2006.

[Fazekas *et al.*, 2018] Katalin Fazekas, Fahiem Bacchus, and Armin Biere. Implicit hitting set algorithms for maximum satisfiability modulo theories. In *Automated Reasoning - 9th International Joint Conference, IJCAR 2018*, volume 10900 of *Lecture Notes in Computer Science*, pages 134–151. Springer, 2018.

[Fu and Malik, 2006] Zhaohui Fu and Sharad Malik. On solving the partial MAX-SAT problem. In *Proc. SAT*, volume 4121 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2006.

[Gange *et al.*, 2020] Graeme Gange, Jeremias Berg, Emir Demirovic, and Peter J. Stuckey. Core-guided and core-boosted search for CP. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research, CPAIOR 2020*, volume 12296 of *Lecture Notes in Computer Science*, pages 205–221. Springer, 2020.

[Heras *et al.*, 2011] Federico Heras, António Morgado, and João Marques-Silva. Core-guided binary search algorithms for maximum satisfiability. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011*. AAAI Press, 2011.

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

[Järvisalo *et al.*, 2012] Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In *Automated Reasoning - 6th International Joint Conference, IJCAR 2012*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012.

[Karpinski and Piotrów, 2019] Michal Karpinski and Marek Piotrów. Encoding cardinality constraints using multiway merge selection networks. *Constraints An Int. J.*, 24(3-4):234–251, 2019.

[Larrosa *et al.*, 2011] Javier Larrosa, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. A framework for certified boolean branch-and-bound optimization. *J. Autom. Reason.*, 46(1):81–102, 2011.

[Marques-Silva and Planes, 2007] João Marques-Silva and Jordi Planes. On using unsatisfiability for solving maximum satisfiability. *CoRR*, abs/0712.1097, 2007.

[Morgado *et al.*, 2013] António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints An Int. J.*, 18(4):478–534, 2013.

[Morgado *et al.*, 2014] António Morgado, Carmine Dodaro, and João Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2014.

[Narodytska and Bacchus, 2014] Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proc. AAAI*, pages 2717–2723. AAAI Press, 2014.

[Narodytska and Bjørner, 2022] Nina Narodytska and Nikolaj S. Bjørner. Analysis of core-guided MaxSat using cores and correction sets. In *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022*, volume 236 of *LIPIcs*, pages 26:1–26:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Nieuwenhuis *et al.*, 2006] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(t). *J. ACM*, 53(6):937–977, 2006.

[Saikko *et al.*, 2016] Paul Saikko, Jeremias Berg, and Matti Järvisalo. LMHS: A SAT-IP hybrid MaxSAT solver. In *Theory and Applications of Satisfiability Testing, SAT 2016*, volume 9710 of *Lecture Notes in Computer Science*, pages 539–546. Springer, 2016.

[Saikko *et al.*, 2018] Paul Saikko, Carmine Dodaro, Mario Alviano, and Matti Järvisalo. A hybrid approach to optimization in answer set programming. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018*, pages 32–41. AAAI Press, 2018.

[Sinz, 2005] Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *Principles and Practice of Constraint Programming - CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005.

[Smirnov *et al.*, 2021] Pavel Smirnov, Jeremias Berg, and Matti Järvisalo. Pseudo-boolean optimization by implicit hitting sets. In *27th International Conference on Principles and Practice of Constraint Programming, CP 2021*, volume 210 of *LIPIcs*, pages 51:1–51:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Smirnov *et al.*, 2022] Pavel Smirnov, Jeremias Berg, and Matti Järvisalo. Improvements to the implicit hitting set approach to pseudo-boolean optimization. In *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022*, volume 236 of *LIPIcs*, pages 13:1–13:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.