Ordered Objectives in Maximum Satisfiability

Jeremias Berg[1], **André Schidler**[2], **Matti Järvisalo**[1]

[1]University of Helsinki, Finland
[2]University of Freiburg, Germany
jeremias.berg@helsinki.fi, schidler@cs.uni-freiburg.de, matti.jarvisalo@helsinki.fi

Abstract

Maximum satisfiability (MaxSAT) is a viable approach to solving NP-hard combinatorial optimization problems through propositional encodings. Understanding how problem structure and encodings impact the behaviour of different MaxSAT solving algorithms is an important challenge. In this work, we identify MaxSAT instances in which the constraints entail an ordering of the objective variables as an interesting instance class from the perspectives of problem structure and MaxSAT solving. From the problem structure perspective, we show that a non-negligible percentage of instances in commonly used MaxSAT benchmark sets have ordered objectives and further identify various examples of such problem domains to which MaxSAT solvers have been successfully applied. From the algorithmic perspective, we argue that MaxSAT instances with ordered objectives, provided an ordering, can be solved (at least) as efficiently with a very simplistic algorithmic approach as with modern core-based MaxSAT solving algorithms. We show empirically that state-of-the-art MaxSAT solvers suffer from overheads and are outperformed by the simplistic approach on real-world optimization problems with ordered objectives.

1 Introduction

Building on the success of Boolean satisfiability (SAT) solving (Biere et al. 2021; Fichte et al. 2023), maximum satisfiability (MaxSAT) is today a noteworthy declarative approach to solving computationally hard combinatorial optimization problems (Bacchus, Järvisalo, and Martins 2021; Li and Manyà 2021). Currently, the most successful algorithmic approaches to MaxSAT solving are (i) *core-based algorithms*—namely, *core-guided* (Andres et al. 2012; Ansótegui, Didier, and Gabàs 2015; Ansótegui and Gabàs 2017; Bjørner and Narodytska 2015; Fu and Malik 2006; Marques-Silva and Planes 2007; Morgado, Dodaro, and Marques-Silva 2014; Narodytska and Bacchus 2014) and *implicit hitting set (IHS) algorithms* (Davies and Bacchus 2013; Saikko, Berg, and Järvisalo 2016)—based on iteratively identifying sources of inconsistency in terms of unsatisfiable cores using a SAT solver; (ii) *solution-improving algorithms* (Berre and Parrain 2010; Eén and Sörensson 2006; Paxian, Reimer, and Becker 2018) based on querying a SAT solver for better solutions until there are none; and (iii) *branch-and-bound approaches* (Abramé and Habet 2014; Li, Manyà, and Planes 2005) with recent improvements through integration of clause learning techniques (Li et al. 2021, 2022). A majority of modern MaxSAT solvers implement variants of the core-based algorithms (Bacchus, Järvisalo, and Martins 2021).

Understanding how problem structure and encodings impact the efficiency of solvers is an important challenge towards further improving the current state of the art in MaxSAT solving. Previous work in this context includes structure-based heuristics for increased solver efficiency, such as the so-called at-most-ones technique (Ignatiev, Morgado, and Marques-Silva 2019) based on identifying cliques of objective variables, and multilevel optimization (Argelich, Lynce, and Marques-Silva 2009), based on partitioning the objective into independent subparts. Both are widely employed in current solvers.

We identify MaxSAT instances in which the constraints entail an ordering of the objective variables as an interesting instance class from the perspectives of problem structure, encodings, and MaxSAT solving. Various examples of domains in which MaxSAT encodings yielding instances with such *ordered objectives* can be identified. As we will explicate, one specific example are problems in which the underlying objective is to minimize the maximum of a specific measure, i.e., min-max type optimization problems. MaxSAT encodings for min-max type problems in the literature include e.g. *treewidth* (Berg and Järvisalo 2014; Robertson and Seymour 1986), *judgment aggregation in computational social choice* (Conati, Niskanen, and Järvisalo 2024; Endriss 2016), *inconsistency measurement of propositional knowledge bases* (Grant 1978; Niskanen et al. 2023), *interpretable clustering* (Shati, Cohen, and McIlraith 2023), *task allocation* (Zheng, Cherif, and Shibasaki 2024), as well as *graph coloring* (Glorian et al. 2019; Van Gelder 2008). Specifically, natural encodings of such objectives result in binary objective variables indicating that the maximum value under minimization being bounded by different constants. This yields an ordering among the MaxSAT objective variables that essentially hides the structure of the original objective. While this "hidden structure" in terms of the underlying semantics of such objective variables occurs naturally in various types of problems viewed as MaxSAT,

modern solvers do not use it, and—despite successful applications in such problem domains—actually incur runtime overheads on such encodings.

Our main contributions are five-fold: (i) We define MaxSAT instances with ordered and almost-ordered objectives, and relate these properties with minimal unsatisfiable cores of instances being unit and all optimal solutions incurring cost on the same soft constraints. (ii) We show empirically that a non-negligible percentage of MaxSAT instances in recent MaxSAT Evaluation benchmark sets can be identified to have ordered objectives by outlining algorithmic ways for identifying whether a given MaxSAT instance has an ordered objective without knowledge of the underlying problem structure. (iii) We identify various examples of MaxSAT encodings from the literature which yield instances with ordered objective, pointing out that care needs to be taken in the encodings to guarantee ordered objectives. (iv) We formally analyze the behaviour of core-based algorithms on MaxSAT instances having ordered objectives, and pointing out that the algorithms essentially "trivialize" to simplistic iterative approaches on such instances. (v) We empirically evaluate the performance various state-of-the-art MaxSAT solvers on problems yielding ordered objectives, showing that that state-of-the-art solvers can suffer from unnecessary performance overheads on such instances against a more simplistic iterative approach.

Formal proofs, additional details, empirical data and implementation code are available in an online supplement at https://doi.org/10.5281/zenodo.17601897.

2 Maximum Satisfiability

A literal ℓ is a Boolean variable x or its negation $\neg x$. A clause $C = \ell_1 \vee \cdots \vee \ell_n$ is a disjunction of literals and a (CNF) formula $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. A clause may be viewed as a set of literals and a formula as a set of clauses. An assignment α maps variables to 1 (TRUE) or 0 (FALSE). The semantics of assignments are extended to literals, clauses, and formulas standardly: $\alpha(\neg x) = 1 - \alpha(x)$, $\alpha(C) = \max\{\alpha(\ell) \mid \ell \in C\}$ and $\alpha(F) = \min\{\alpha(C) \mid C \in F\}$. α is a solution to F if $\alpha(F) = 1$. A formula is satisfiable if it has a solution and unsatisfiable otherwise. A solution α may be viewed as the set of literals it assigns to 1, i.e., write $\ell \in \alpha$ if $\alpha(\ell) = 1$.

An objective $O \equiv \sum_i c_i b_i$ is a pseudo-Boolean expression where wlog. each b_i is a variable and c_i a positive constant. An assignment α over the (objective) variables in O has cost $\alpha(O) = \sum_i c_i \alpha(b_i)$. For representational convenience, we use an "objective-based" view to (weighted partial) MaxSAT which is equivalent to the classical view of via soft and hard clauses (Ihalainen et al. 2024; Leivo, Berg, and Järvisalo 2020) and better captures the representation that core-based MaxSAT solvers work with. An instance (F, O) of (weighted partial) MaxSAT consists of a formula F and an objective O. The goal in MaxSAT is to find an optimal solution, i.e., a solution α of F minimizing $\alpha(O)$ over all solutions to F. The optimal cost of (F, O) is the cost of its optimal solutions. For representational simplicity, we assume that MaxSAT instances have solutions. The restriction of a solution α to objective variables is denoted by $\alpha|_O$.

Core-based MaxSAT solvers use *(unsatisfiable) cores*. For convenience, with slight abuse of terminology, we say that a clause C is a core of a MaxSAT instance (F, O) if (i) C contains only objective variables and (ii) every solution α of F satisfies C, i.e., $F \wedge \neg C$ is unsatisfiable (with $\neg C$ more typically called an unsatisfiable core). A core C is minimal (an MUS) if no subset $C_s \subset C$ is a core. A core C of an instance (F, O) can be seen as a local witness for a lower bound on its optimal cost: since any solution α must satisfy each core C, the cost of α cannot be lower than the smallest objective coefficient among the variables in C.

3 Ordered Objectives in MaxSAT Instances

We turn to MaxSAT instances with ordered objectives.

3.1 Ordered Objectives

As a running example, we will use graph coloring, where the task is to compute the chromatic number of a given undirected graph $G = (V, E)$, i.e., the minimum number of colors required to color each vertex without monochromatic edges. Encoded as MaxSAT, we obtain for a given graph G a MaxSAT instance (F^G, O^G) with a one-to-one correspondence between solutions of F^G and colorings of G so that the cost of a solution α is the number of colors used in the coloring χ^α given by α. A natural such encoding (see e.g. (Glorian et al. 2019)) defines the objective $O^G \equiv \sum_{k=1}^{n} b_k$, where n is the number of nodes in G and b_k is an indicator for using at least k colors. More precisely, α sets $\alpha(b_k) = 1$ if χ^α maps the nodes in V to at least k different colors. Evidently, if χ^α uses more than k colors, it uses more than $k - 1$ colors. Stated in terms of MaxSAT, if $\alpha(b_k) = 1$, then $\alpha(b_{k-1}) = 1$ also holds. The following definition of an *ordered objective* captures more generally MaxSAT instances in which the constraints entail such an ordering on the objective variables.

Definition 1. *A MaxSAT instance (F, O) has an ordered objective if there is an ordering $\prec$ of the variables in O such that $\alpha(b \vee \neg b') = 1$ (i.e., $\alpha(b' \rightarrow b) = 1$) holds whenever α is any solution to (F, O) and b and b' are objective variables for which $b \prec b'$.*

For a specific ordering $\prec$, an instance (F, O) has an ordered objective wrt $\prec$. We say that an instance $(F, \sum_{k=1}^{n} c_k b_k)$ has an ordered objective to mean that it has an ordered objective wrt the natural ordering $\prec$ for which $b_i \prec b_j$ whenever $i < j$.

Instances with ordered objectives have simple MUS structures; each MUS contains only a single literal. When instantiated for graph coloring, the following general proposition states that every coloring of a graph with a chromatic number t has to use at least k colors for all $k = 1, \ldots, t$.

Proposition 1. *Any MUS C of an instance $(F, \sum_{k=1}^{n} c_k b_k)$ that has an ordered objective contains a single literal.*

An ordered objective also implies that the MUSes themselves are ordered.

Corollary 1. *The set of MUSes of an instance $(F, \sum_{k=1}^{n} c_k b_k)$ that has an ordered objective is exactly the set of unit clauses $\{(b_i) \mid i < t\}$ for some $1 \leq t \leq n + 1$.*

As every optimal solution to a MaxSAT instance assigns at least one variable in each of its MUSes to 1, by Corollary 1 all optimal solutions to a MaxSAT instance $(F, \sum_{k=1}^{n} c_k b_k)$ with an ordered objective, assign $b_i = 1$ for $i = 1, \ldots, t$ to 1 for a fixed constant t and the rest to 0. In this sense, an ordered objective implies that fewer assignments of the objective variables need to be considered when computing optimal solutions.

While an ordered objective implies unit-MUSes, the converse does not hold in general.

Example 1. *Consider the instance (F, O) with $F = \{(b_1), (b_1 \vee b_2), (b_1 \vee b_3)\}$ and $O = b_1 + b_2 + b_3$. The only MUS of (F, O) is (b_1). The instance does not have an ordered objective. To see this, consider the two solutions $\alpha_1 = \{b_1, b_2, \neg b_3\}$ and $\alpha_2 = \{b_1, \neg b_2, b_3\}$ and observe that $\alpha_1(b_3 \vee \neg b_2) = 0$ while $\alpha_2(b_2 \vee \neg b_3) = 0$.*

3.2 Almost-Ordered Objectives

When devising MaxSAT encodings, the semantics of objective variables are often encoded using implications instead of bi-implications (equivalences). In case of graph coloring, e.g., implications are used to encode that b_k is set to 1 whenever at least k colors are used. The other direction would encode that b_k is set to to 0 whenever fewer than k colors are used. However, this is not necessary as it is covered through minimizing the objective. This does, however, mean that there can be non-optimal solutions that assign objective variables to 1 "unnecessarily". Consequently, the resulting instance may not have an ordered objective in the sense of Definition 1, even if the semantics of graph coloring imply that it should. The following definition of an almost-ordered objective captures more generally this intuition.

Definition 2. *An instance (F, O) has an almost-ordered objective if there is an ordering $\prec$ of the variables in O for which $\alpha(b \vee \neg b') = 1$ holds whenever α is an **optimal** solution to (F, O), and b, b' objective variables for which $b \prec b'$.*

For a specific ordering $\prec$, we say that an instance (F, O) has an almost-ordered objective wrt $\prec$. When writing an instance with an almost-ordered objective as $(F, \sum_{k=1}^{n} c_k b_k)$, we implicitly assume that the ordering is natural.

In contrast to an ordered objective, an almost-ordered objective does not generally imply that all MUSes are units.

Example 2. *Consider the MaxSAT instance (F, O) with $F = \{(b_1 \vee x), \vee(\neg x \vee b_2)\}$ and $O = b_1 + 2b_2$. The unique optimal solution is $\alpha = \{\neg x, b_1, \neg b_2\}$. Thus $\alpha(b_1 \vee \neg b_2) = 1$ so (F, O) has an almost-ordered objective. One non-unit MUS of the the instance is $(b_1 \vee b_2)$.*

On the other hand, any instance that only has unit-MUSes has an almost-ordered objective. This follows from a more general observation: any MaxSAT instance for which all optimal solutions assign the objective variables in the same way has an almost-ordered objective. The following definition makes this precise.

Definition 3. *A MaxSAT instance (F, O) has a unique optimal objective-solution if $\alpha_1|_O = \alpha_2|_O$ holds for any optimal solutions α_1 and α_2 of (F, O).*

Having an almost-ordered objective is equivalent to having a unique optimal objective-solution.

Proposition 2. *A MaxSAT instance (F, O) has a unique optimal objective-solution if and only if it has an almost-ordered objective.*

As any instance for which all MUSes are unit has a unique optimal objective-solution, we have the following.

Corollary 2. *Assume that every MUS of an instance (F, O) is a unit. Then (F, O) has an almost-ordered objective.*

Any instance with an almost-ordered objective can be extended to an instance with an ordered objective by computing the ordered extension.

Definition 4. *Let (F, O) be an instance with an almost-ordered objective wrt $\prec$. The* ordered extension *of (F, O) is $(F \wedge \{(b_i \vee \neg b_j) \mid b_i \prec b_j\}, O)$.*

The following observation summarizes properties of ordered extensions.

Observation 1. *Let (F^E, O) be the ordered extension of an instance (F, O) with an almost-ordered objective wrt $\prec$. The following hold. (i) (F^E, O) has an ordered objective wrt $\prec$. (ii) An optimal solution α to (F^E, O) is an optimal solution to (F, O) and vice versa.*

We end this section by noting that (almost-)ordered objectives could also be studied through the lens of redundant clauses (Ihalainen, Berg, and Järvisalo 2022). From that perspective, an instance has an (almost-)ordered objective wrt. $\prec$ if $(b \vee \neg b')$ is entailed by the other clauses in (preserves the optimal solutions of) the instance for all $b \prec b'$.

4 Prevalence of Ordered Objectives

We show—both algorithmically and through a literature survey—that MaxSAT instances with (almost-)ordered objectives arise in practice from various settings. Algorithmically, we show that a non-negligible fraction of standard MaxSAT Evaluation (MSE) benchmarks have (almost-)ordered objectives. We also give concrete examples of problem domains in which natural MaxSAT encodings can be identified to result in instances with (almost-)ordered objectives without resorting to algorithmic identification.

4.1 Algorithmic Detection

To decide if an instance (F, O) has an ordered objective, we first let $\textsc{prec}(b) = \{\, b' : F \text{ entails } b \rightarrow b' \,\}$ for each objective variable b Then, we define $\prec$ by $b \prec b'$ iff $|\textsc{prec}(b)| \leq |\textsc{prec}(b')|$, with ties broken arbitrarily. Our approaches for detecting ordered objectives are based on the following proposition.

Proposition 3. *Let (F, O), $\textsc{prec}(b)$ and $\prec$ be as defined earlier. The following statements are equivalent.*
(1) (F, O) has an ordered objective.
(2) (F, O) has an ordered objective wrt $\prec$.
(3) $\textsc{prec}(b) \subseteq \textsc{prec}(b')$ for all $b \prec b'$.
$3 \Rightarrow 2 \Rightarrow 1$ also hold for any subsets of $\textsc{prec}(b)$.

Thus, given the sets $\textsc{prec}(b)$ for each objective variable, deciding if the instance has an ordered objective is

Instances	#	Ord.	Not Ord.	AO	Not AO
Unweighted	1558	134	1266	589	556
Weighted	1545	29	1421	616	439

Table 1: MaxSAT Evaluations 2022–2024 benchmarks: Instances determined (not) to have ordered and almost-ordered objectives are shown in the columns *Ord.*, (*Not Ord.*) and *AO*. (*Not AO*), respectively.

equivalent to checking if $\text{PREC}(b) \subseteq \text{PREC}(b')$ for all $b \prec b'$. We consider two methods for computing $\text{PREC}(b)$ for each objective variable b, the *exhaustive approach* and *unit-propagation-based lookahead*. The exhaustive approach uses a SAT solver to check if $F \wedge \neg(b \rightarrow b')$ is unsatisfiable for each pair b, b'. If it is, F entails $(b \rightarrow b')$ so $b' \in \text{PREC}(b)$. The exhaustive method correctly computes the full $\text{PREC}(b)$ for each b and thus, by Proposition 3, can identify any instance with an ordered objective. However, the number of SAT calls required is worst-case quadratic in the number of objective variables.

Standard unit-propagation (UP) based lookahead provides an alternative polytime incomplete approach to constructing a subset of each $\text{PREC}(b)$. Specifically, propagating a variable b on F returns some of the literals ℓ such that F implies $b \rightarrow \ell$. Any objective variables returned when unit propagating b are thus in $\text{PREC}(b)$. While UP cannot in general construct the full $\text{PREC}(b)$ for each b, by Proposition 3 checking if $\text{PREC}(b) \subseteq \text{PREC}(b')$ holds for all $b \prec b'$ and any subsets of $\text{PREC}(b)$ is sufficient for detecting an ordered objective.

By Proposition 2, almost-ordered objectives can be detected by first obtaining an optimal solution α with one call to an exact MaxSAT solver, and then calling the solver again after adding the clause $\{\neg\ell \mid \ell \in \alpha|_O\}$ that blocks the assignment α over objective variables. If the MaxSAT solver returns a higher optimal cost on the second call than on the first call, the instance has a unique optimal objective-solution and thus, by Proposition 2, an almost-ordered objective. Alternatively, by Corollary 1, some of the instances with almost-ordered objectives can be identified by checking if all MUSes are unit by enumeration (terminating early whenever a non-unit MUS is found).

We applied the just-described detection algorithms (more details provided in the online supplement) on the 1558 unweighted and 1545 weighted benchmarks MSE 2022-2024 benchmark sets (https://maxsat-evaluations.github.io) with duplicate instances filtered by filename. Table 1 reports the number of instances that could be detected to have an (almost-)ordered objective using a per-instance 5-h time and 32-GB memory limit. We detected ordered objectives in six unweighted domains and two weighted domains (in a total of 163 instances), and could determine the non-existence of ordered objectives on 1266 unweighted and 1421 weighted benchmarks, respectively. While unable to decide the non-existence of ordered objectives, UP-based lookahead was fast, taking on average 44 seconds. Turning to almost-ordered objectives, we found definitive answers for 1145 of 1558 unweighted and 1055 of 1545 weighted instances. Approximately half of the instances with a definitive answer have an almost-ordered objective. We conclude that almost-ordered objectives indeed appear in a non-negligible fraction of the instances.

4.2 Problems with (Almost-)Ordered Objectives

Complementing detecting ordered objectives algorithmically, encodings of various problem domains can be identified by inspection to give rise to MaxSAT instances having (almost-)ordered objectives. We provide an overview of a(-n incomplete) selection of such MaxSAT applications from the literature.

In **min-max optimization** the goal is to compute a solution α of a CNF formula F that minimizes $\max\{g_i(\alpha) \mid i = 1..n\}$ where each g_i is a positive integer-valued function. As $\max\{g_i(\alpha) \mid i = 1..n\} \geq k$ implies $\max\{g_i(\alpha) \mid i = 1..n\} \geq k-1$, it follows that MaxSAT encodings of such problems naturally yield instances with almost-ordered objectives. A natural encoding that uses a low number of clauses encodes the (integer) values of each g_i into CNF using the order encoding (Bailleux and Boufkhad 2003; Crawford and Baker 1994) with variables o_k^i for which a solution α sets $\alpha(o_k^i) = 1$ if $g_i(\alpha) \geq k$. The minimization of the maximum value of the g_i is achieved by introducing (i) new variables b_k, (ii) clauses equivalent to $\neg b_k \rightarrow \bigwedge_{i=1}^{n}(\neg o_k^i)$, and (iii) the objective $O \equiv \sum_k b_k$. Note that if α is a solution with $\alpha(b_k) = 0$, then $\alpha(o_k^i) = 0$ for all i and thus $\max\{g_i(\alpha) \mid i = 1..n\} < k$. This abstract encoding scheme of min-max optimization results in instances with almost-ordered objectives but not ordered ones. Intuitively, this is because optimal solutions will not assign objective variables to 1 "unnecessarily", but non-optimal solutions can. There are several ways of turning the instance into one that has an ordered objective. The ordered extension (Definition 4) would add clauses of the form $b_k \rightarrow b_{k-1}$. Alternatively, one can add clauses equivalent to $b_k \rightarrow (o_k^1 \vee \ldots \vee o_k^n)$ and $o_k^i \rightarrow (g_i(\alpha) \geq k)$, effectively enforcing the semantics of the b_k and o_k^i variables with equivalences.

Various problem domains are intrinsically min-max optimization. Even graph coloring fits this encoding scheme by taking $g_i(\alpha)$ to be the index of the color of the ith node in the coloring α represents. Beyond coloring, we shortly overview further min-max optimization problem settings for which MaxSAT encodings yielding instances with (almost-)ordered objectives have been proposed.

Treewidth is a central graph-theoretic measure generally for problems where their underlying structure can be represented as a graph (Cygan et al. 2015). A proposed MaxSAT encoding for treewidth (Berg and Järvisalo 2014; Samer and Veith 2009) minimizes the maximum out-degree of the vertices in the so-called triangulated graph induced by elimination orderings. It includes for each vertex v the variables $c_{v,k}$, where $c_{v,k} = 1$ if the degree of v in the triangulated graph is larger than k, and thus fits the abstract encoding scheme of min-max optimization.

Judgment aggregation (JA) is a well-studied problem in computational social choice (COMSOC) research (Endriss 2016) that deals with aggregating judgment sets expressed

by individual agents regarding the validity of logical statements into a collective judgment. As one of the central rules studied in the COMSOC literature, the MaxHamming aggregation rule minimizes the maximum Hamming distance between the collective judgment set and the judgment sets of the voters. A recently-proposed MaxSAT encoding (Conati, Niskanen, and Järvisalo 2024) of this problem can be viewed as an instantiation of the abstract encoding scheme as its objective variables explicitly represent the range of possible values for the maximum Hamming distance.

Inconsistency measurement (IM) (Grant 1978; Grant and Martinez 2018; Thimm 2018; Thimm and Wallner 2019) refers to providing a quantitative characterization of the level of inconsistency of a knowledge base. A MaxSAT approach to IM (Niskanen et al. 2023), captures among others the so-called max-distance IM measure. While the problem encodings differ in terms of their underlying constraints, it can be identified that the MaxSAT encoding of the max-distance IM measure is analogous—in terms of its objective—to the encoding of JA under the MaxHamming rule.

A further example of min-max problems comes from **interpretable clustering** under the objective of minimizing the maximum diameter (MD) of the clusters, i.e., the maximum distance between two points assigned to the same cluster (Shati, Cohen, and McIlraith 2023). A recent MaxSAT encoding of the problem includes objective variables b_w^- for which a solution α that assigns $\alpha(b_w^-) = 0$ represents a clustering in which any pair of datapoints x, y with a pairwise distance larger than w are assigned to different clusters. This encoding yields instances with an almost-ordered objective by the semantics of clusterings; if $w_1 \leq w$ and α is optimal, then $\alpha(b_{w_1}^-) = 0$. Finally, MaxSAT has also been proposed for solving min-max problems related to **task allocation** with the aim to minimize the maximum power output of workstations (Zheng, Cherif, and Shibasaki 2024).

5 MaxSAT Solving on Ordered Objectives

We analyze the search performed by state-of-the-art MaxSAT algorithms on instances with ordered objectives.

5.1 Core-Based MaxSAT Solving

Essentially all core-guided and IHS-based MaxSAT solvers extract unsatisfiable cores with the assumption interface offered by modern CDCL SAT solvers (Eén and Sörensson 2003; Marques-Silva, Lynce, and Malik 2021). Given a MaxSAT instance (F, O) we abstract the core extraction step into the procedure $\texttt{ExtCore}(F, \mathcal{A})$ where $\mathcal{A}$ is a set of *assumptions* that contain negations of the variables in O. The call returns a triplet $(\mathit{sat?}, \alpha, C)$. The Boolean *sat?* indicates the satisfiability of $F \wedge \bigwedge_{\ell \in \mathcal{A}} \ell$. If $\mathit{sat?} = \text{TRUE}$, C is undefined and α is a solution of F that assigns each $\ell \in \mathcal{A}$ to 1, extending $\mathcal{A}$. If $\mathit{sat?} = \text{FALSE}$, α is undefined and C is an unsatisfiable core over the variables in $\mathcal{A}$, serving as an explanation for the unsatisfiability of $F \wedge \bigwedge_{\ell \in \mathcal{A}} \ell$ in terms of $\mathcal{A}$.

Example 3. *Consider the instance (F, O) with $F = (b_1 \vee x) \wedge (\neg x \vee b_2 \vee b_3) \wedge (b_2 \vee y) \wedge (\neg y \vee b_4)$ and $O = b_1 + 3b_2 + 5b_3 + 5b_4$. An optimal solution α to (F, O) assigns $\alpha(x) = \alpha(b_2) = 1$ and the other variables to 0, with optimal cost 3. The clause $C = (b_1 \vee b_2 \vee b_3)$ is an MUS of (F, O). $\texttt{ExtCore}(F, \{\neg b_i \mid i = 1..4\})$ returns sat? $= \text{FALSE}$ and, e.g., the (non-minimal) core $C_2 = (b_1 \vee b_2 \vee b_4)$.*

Algorithm 1: CG, core-guided MaxSAT search

Input: Instance (F, O)
Output: Optimal solution α_{best}
1: $(\text{CARD}, O^W) \leftarrow (\emptyset, O)$
2: **while** TRUE **do**
3: $\quad \mathcal{A} \leftarrow \{\neg b \mid cb \in O^W\}$
4: $\quad (\mathit{sat?}, \alpha_{best}, C) \leftarrow \texttt{ExtCore}(F \wedge \text{CARD}, \mathcal{A})$
5: $\quad$ **if** *sat?* **then return** α_{best}
6: $\quad (O^W, N^C) \leftarrow \texttt{Relax}(C, O^W)$
7: $\quad \text{CARD} \leftarrow \text{CARD} \wedge N^C$

Algorithm 1 details an abstraction of the core-guided approach to computing an optimal solution to a MaxSAT instance (F, O). Initially the set CARD of additional clauses is empty and the working objective O^W is O (Line 1). Each iteration of the main loop (Lines 2–7) starts by invoking a SAT solver (Line 4) on $F \wedge \text{CARD}$ assuming the negation of each variable in the working objective O^W. This check returns TRUE if there is a solution α of $F \wedge \text{CARD}$ with $\alpha(O^W) = 0$. If so, α is optimal for F, resulting in termination (Line 5). Otherwise, a new core C is obtained and the instance is relaxed (Line 6). The Relax subroutine returns a new working objective O^W and additional clauses N^C that are added to CARD (Line 7) before the loop is reiterated.

Core-guided algorithms differ in the specifics of Relax (see the online supplement for the OLL instantiation (Andres et al. 2012; Morgado, Dodaro, and Marques-Silva 2014)). Our observations apply generally to core-guided algorithms via the intuition that for a core-guided algorithm to make progress, the relaxation must allow at least one variable in a core to be assigned to 1 in subsequent iterations. Specifically, when relaxing a core of size 1, core-guided algorithms remove the variable in the core from O^W and do not add new clauses.

Algorithm 2 details the implicit hitting set approach to MaxSAT. First (Line 1) the set CORES of cores is empty. Each iteration of the main loop (Lines 2–7) begins by computing wrt O a minimum-cost solution γ of CORES (Line 3). A SAT solver is then invoked (Line 5) on the clauses in F assuming the negation of each objective variable in O assigned to 0 by γ. This query returns TRUE if there is a solution α of

Algorithm 2: IHS, IHS-based MaxSAT search

Input: Instance (F, O)
Output: Optimal solution α_{best}
1: $\text{CORES} \leftarrow \emptyset$
2: **while** TRUE **do**
3: $\quad \gamma \leftarrow \texttt{MinCost-Sol}(\text{CORES}, O)$
4: $\quad \mathcal{A} \leftarrow \{\neg b \mid \gamma(b) = 0\}$
5: $\quad (\mathit{sat?}, \alpha_{best}, C) \leftarrow \texttt{ExtCore}(F, \mathcal{A})$
6: $\quad$ **if** *sat?* **then return** α_{best}
7: $\quad \text{CORES} \leftarrow \text{CORES} \cup \{C\}$

F with $\alpha(O) = \gamma(O)$. If so, α is optimal and the algorithm terminates (Line 6). Otherwise, a new core falsified by $\mathcal{A}$ is obtained, added to CORES (Line 7) and the loop reiterated.

Example 4. *Invoke* IHS *on the instance from Example 3. Take* $(b_1 \vee b_2 \vee b_3)$ *as the first core extracted on Line 5. Then* MinCost-Sol *returns the solution* γ *assigning* $\gamma(b_1) = 1$ *and all other objective variables to* 0*. The next call to* ExtCore *is made with* $\mathcal{A} = \{\neg b_2, \neg b_3, \neg b_4\}$ *as assumptions. The call returns e.g.* $(b_2 \vee b_4)$*. In the next iteration the only minimum-cost solution* γ *to* CORES *assigns* $\gamma(b_2) = 1$ *and others to* 0*, and* IHS *terminates after calling* ExtCore*.*

We use CG-mus and IHS-mus to denote variants of CG and IHS, respectively, in which the call to ExtCore on Line 5 is guaranteed to return an MUS. In practice, extracting an MUS by complete core minimization requires further SAT solver calls.

5.2 Core-Based Search and Ordered Objectives

We analyze the search of CG and IHS on instances with ordered objectives. By Proposition 1 such instances also have unit-MUSes. Unit cores trivialize the computation of both CG and IHS; the Relax subroutine of CG simply removes the variable in each MUS from the working objective, while the only minimum-cost solution over a set of unit-MUSes that the MinCost-Sol subroutine of IHS can return assigns the variables that occur in the discovered MUSes to 1 and the rest to 0. As a consequence (as formalized next), CG-mus and IHS-mus, the versions of CG and IHS guaranteed to extract MUSes on each iteration, perform essentially the same search on instances with unit-MUSes.

Proposition 4. *Consider invoking* CG-mus *or* IHS-mus *on a unit-MUS instance* (F, O)*. Assume that (i) there are* m *MUSes in total, and that (ii) when given a unit-core* (b)*,* Relax *removes* b *from* O^W *and does not add any clauses to* CARD*. Then the following hold.*
1) The main loop (Lines 2 to 7) of both algorithms is repeated $m + 1$ *times.*
2) The set $\mathcal{A}$ *formed on iteration* i *consists in both algorithms of exactly the negations of the objective variables that did not appear in the* $i - 1$ *first MUSes extracted.*
3) CARD *in* CG-mus *will be empty on every iteration.*

Consequently, the CG-mus and IHS-mus algorithms exhibit best-case performance in the number of iterations required by CG and IHS when solving instances with unit-MUSes, which include instances with ordered objectives. More specifically, by Proposition 4, both CG-mus and IHS-mus require $m+1$ iterations when solving a unit-MUS instance (F, O) that has m MUSes. To see that IHS and CG need to perform at least $m + 1$ iterations, note that any optimal solution to (F, O) assigns exactly m objective variables to 1 and that the number of variables of O that can be assigned to 1 by (potential) solutions extending the assumptions $\mathcal{A}$ used in the ExtCore invocations of both CG and IHS will be 0 on the first iteration and increase by at most one on subsequent ones.

Furthermore, CG and IHS may require more iterations if the algorithms do not always extract MUSes (which is indeed the case in practical implementations), since this can lead to CG adding unnecessary clauses and IHS making the hitting-set instance unnecessarily large. To see this, consider an instance (F_n, O_n) with $O_n = \sum_{i=1}^{n} i \cdot b_i$ and assume that (b_n) is its only MUS. An optimal solution α of the instance assigns $\alpha(b_n) = 1$ and $\alpha(b_i) = 0$ for all other i. By Proposition 4, CG-mus and IHS-mus require two iterations to return an optimal solution of (F_n, O_n). In contrast, the next example describes a (worst-case) execution of IHS that requires $n + 1$ iterations.

Algorithm 3: SimpleUS, simple search over an ordered objective.

Input: Instance (F, O) with O ordered wrt $\prec$
Output: An optimal solution α_{best}
1: $\{b_1, \ldots, b_n\} \leftarrow \texttt{sort}(\{b \mid c \cdot b \in O\}, \prec)$;
2: **for** $i = 1..n$ **do**
3: $\quad (sat?, \alpha_{best}, \_) \leftarrow \texttt{ExtCore}(F, \{\neg b_i\})$
4: $\quad$ **if** *sat?* **then return** α_{best}

Example 5. *Invoke* IHS *on* (F_n, O_n)*. Assume that the core extracted in the first iteration is* $C_1 = b_1 \vee \cdots \vee b_n$*. The only minimum-cost solution* γ *to* $\{C_1\}$ *assigns* $\gamma(b_1) = 1$ *and* $\gamma(b_i) = 0$ *for all other* i *so the next assumptions used on Line 5 will be* $\mathcal{A} = \{\neg b_2, \ldots, \neg b_n\}$*. Assume that the second core extracted is* $C_2 = b_2 \vee \cdots \vee b_n$*. Then the next call to* MinCost-Sol *returns the solution* γ *that assigns* $\gamma(b_2) = 1$ *and* $\gamma(b_i) = 0$ *for all other* i*, which in turns leads to the next call to* ExtCore *having* $\{\neg b_1, \neg b_3, .., \neg b_n\}$ *as assumptions. If IHS continues in the same manner, it will require in total* $n + 1$ *iterations before terminating.*

An analogous example for the OLL algorithm, showing that a worst-case quadratic number of extra variables and clauses are introduced in the relaxation steps necessary for termination, is provided in the online supplement.

5.3 Ordered Objective-Specific MaxSAT Solving

We have established that on MaxSAT instances with ordered objectives (i) CG-mus and IHS-mus perform essentially the same search, and that (ii) CG and IHS algorithms might incur overhead in terms of iterations required and additional constraints added. We now discuss two simple MaxSAT algorithm variants for instances with ordered objectives. The algorithms can be instantiated without (costly) MUS extraction, but still avoid the potential overheads of CG and IHS.

The first algorithm, SimpleUS outlined as Algorithm 3, starts by sorting the variables in the objective according to $\prec$. Then, for increasing i, the algorithm invokes ExtCore with the negation $\neg b_i$ of the ith objective variable b_i in the sorted order as a single assumption. If *sat?* $=$ FALSE, the only core that ExtCore can return is the MUS (b_i). If *sat?* $=$ TRUE, the obtained solution α is optimal as it assigns $\alpha(b_j) = 1$ for each $j < i$ (since (b_j) is an MUS), and $\alpha(b_k) = 0$ for all $k \geq i$ (as $(b_i \vee \neg b_k)$ holds due to the instance having an ordered objective). The natural counterpart of this lower-bounding search performed by SimpleUS is SimpleSIS which—as a simple form of solution-improving search—tries the objective variables

Solver	UW (108) #	UW (108) P2	W (17) #	W (17) P2	Col (415) #	Col (415) P2	JA (50) #	JA (50) P2	TW (287) #	TW (287) P2
SimpleSIS	88	1.8	13	2.4	387	0.5	27	3.8	143	3.8
SimpleUS	86	2.0	14	1.8	386	0.5	27	4.1	136	4.0
EvalSCIP	84	2.2	12	2.5	392	0.5	19	5.0	123	4.4
EvalNoSCIP	83	2.2	13	2.3	387	0.5	19	5.0	126	4.2
MaxHS	86	2.0	14	2.0	387	0.5	23	4.4	133	4.0
MaxCDCL	81	2.4	12	2.6	385	0.6	21	4.9	115	4.5
Pacose	70	2.9	11	3.3	380	0.6	13	5.7	85	5.2
Virtual Best	89	1.7	14	1.6	393	0.4	28	3.7	143	3.8

Table 2: Runtime comparison. #: number of solved instances. P2: PAR-2, average runtime in 1000-s units each unsolved instance contributing 2x the time limit.

in reverse order and iterates until ExtCore returns UNSAT at which point the final solution is optimal.

Note that we detailed SimpleSIS and SimpleUS to enable a comparison to modern core-based MaxSAT algorithms, with the aim of understanding the behaviour of core-based algorithms on instances with ordered objectives. SimpleSIS and SimpleUS are simplifications of solution-improving search (SIS) (Berre and Parrain 2010; Eén and Sörensson 2006; Paxian, Reimer, and Becker 2018) and UNSAT/SAT search, respectively. Both use extra constraints to represent the cost of solutions, and refine either a lower (UNSAT/SAT) or an upper (SIS) bound until an optimal solution is found. SimpleUS and SimpleSIS avoid the use of extra constraints by using the fact that an instance has an ordered objective.

5.4 Overhead of Core-Based MaxSAT in Practice

We evaluate the behavior of state-of-the-art core-based MaxSAT solvers on real-world problems yielding MaxSAT instances having ordered objectives. Specifically, we use both the unweighted (UW) and weighted (W) MSE 2022-2024 benchmark identified to have ordered objective and, as specific domains, judgment aggregation (JA) (Conati, Niskanen, and Järvisalo 2024), graph coloring (Col) (Glorian et al. 2019; Van Gelder 2008) and treewidth (TW) (Berg and Järvisalo 2014; Fichte, Hecher, and Szeider 2020; Samer and Veith 2009). The 50 JA instances based on real-world PrefLib data are from (Conati, Niskanen, and Järvisalo 2024), encoding judgment aggregation under the MaxHamming rule. The 287 treewidth instances based on TreewidthLIB graphs were obtained from the authors of (Fichte, Lodha, and Szeider 2017), limited due to problem hardness to graphs with at most 200 vertices. For graph coloring, we obtained a total of 415 instances by using both the TreewidthLIB graphs and the DIMACS graph coloring instances (https://sites.cc.gatech.edu/dimacs10/).

We evaluate the following state-of-the-art MaxSAT solvers: the OLL solver *EvalMaxSAT* (Avellaneda 2020) (both its pure OLL implementation *EvalNoSCIP* and the variation *EvalSCIP* that first attempts to solve the instance with the MIP solver SCIP (Bolusani et al. 2024) for 500 s before starting OLL search); *MaxHS* (Davies and Bacchus 2013) as the state-of-the-art IHS solver; the solution-improving solver *Pacose* (Paxian and Becker 2023; Paxian, Reimer, and Becker 2018); and the clause learning branch-and-bound MaxSAT solver MaxCDCL (Li et al. 2022). We compare the runtime performance of these solvers and our implementations (available in the online supplement) of the the simple SimpleUS and SimpleSIS algorithms (recall Section 5) using the PySAT 1.8.dev13 interface (Ignatiev, Morgado, and Marques-Silva 2018) and Cadical 1.9.5 (Biere et al. 2024) as the SAT solver. The experiments were run under Ubuntu 18.04 on 10-core 2.4-GHz Intel Xeon E5-2640 v4 CPUs and 160 GB memory, enforcing a per-instance 3600-s time and a 32-GB memory limit.

The results are shown in Table 2. The simplistic solvers perform slightly better than the state-of-the-art solvers on all benchmark domains. This is surprising as state-of-the-art MaxSAT solvers make use of a number of additional search heuristics and are typically considered more performant than performing linear search on the objective function range (Bacchus, Järvisalo, and Martins 2021). These results are in line with our theoretical observations for core-based algorithms: on ordered MaxSAT instances the typically performance improving techniques implemented in state-of-the-art implementations of of core-based MaxSAT solvers are essentially rendered useless, as the performance of the solvers is at best similar to (or weaker than) the simplistic SimpleUS and SimpleSIS algorithms.

5.5 On Solving Almost-Ordered Objectives

Neither SimpleUS and SimpleSIS are directly guaranteed to in general compute optimal solutions on instances with almost-ordered objectives, to see this invoke SimpleUS on the MaxSAT instance (F, O) from Example 2. However, both can be used to solve instances that have unit-MUSes. We also evaluated the performance of SimpleUS and SimpleSIS against state-of-the-art solvers on instances with almost-ordered objectives and unit-MUSes. The results show (see the online supplement) that the simplistic solvers outperform state-of-the-art solvers on instances with almost-ordered objectives, and that state-of-the-art solvers perform better on the ordered objective variants than on the almost-ordered variants.

6 Conclusions

We identified MaxSAT instances with ordered objectives as an interesting class of MaxSAT. A non-negligible fraction of MSE benchmarks have ordered objectives, as various problem encodings (when taking care) yield ordered objectives. Core-based algorithms—both core-guided and implicit hitting set approaches—essentially trivialize (at best) to simplistic iterative search on MaxSAT instances with ordered objectives. Empirically, state-of-the-art MaxSAT solvers can suffer from overheads on MaxSAT instances having ordered objectives even against simplistic iterative search, suggesting that knowledge of ordered objectives can be beneficial when solving MaxSAT instances.

Acknowledgements

The first and third authors are supported by the Research Council of Finland (grants 362987 and 356046). The second author is supported by an Amazon Research Award (Fall/2023). Further, the authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant INST 35/1597-1 FUGG. The initial idea for this work was conceived during the extended reunion of the program "Satisfiability: Theory, Practice, and Beyond" in the spring of 2023 at the Simons Institute for the Theory of Computing at UC Berkeley.

References

Abramé, A.; and Habet, D. 2014. Ahmaxsat: Description and Evaluation of a Branch and Bound Max-SAT Solver. *J. Satisf. Boolean Model. Comput.*, 9(1): 89–128.

Andres, B.; Kaufmann, B.; Matheis, O.; and Schaub, T. 2012. Unsatisfiability-based optimization in clasp. In *ICLP (Technical Communications)*, volume 17 of *LIPIcs*, 211–221. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Ansótegui, C.; Didier, F.; and Gabàs, J. 2015. Exploiting the Structure of Unsatisfiable Cores in MaxSAT. In *IJCAI*, 283–289. AAAI Press.

Ansótegui, C.; and Gabàs, J. 2017. WPM3: An (in)complete algorithm for weighted partial MaxSAT. *Artif. Intell.*, 250: 37–57.

Argelich, J.; Lynce, I.; and Marques-Silva, J. 2009. On Solving Boolean Multilevel Optimization Problemse. In *IJCAI*, 393–398.

Avellaneda, F. 2020. A short description of the solver EvalMaxSAT. In *MaxSAT Evaluation 2020*, 8–9.

Bacchus, F.; Järvisalo, M.; and Martins, R. 2021. Maximum Satisfiability. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, 929–991. IOS Press.

Bailleux, O.; and Boufkhad, Y. 2003. Efficient CNF Encoding of Boolean Cardinality Constraints. In *CP*, volume 2833 of *Lecture Notes in Computer Science*, 108–122. Springer.

Berg, J.; and Järvisalo, M. 2014. SAT-Based Approaches to Treewidth Computation: An Evaluation. In *ICTAI*, 328–335. IEEE Computer Society.

Berre, D. L.; and Parrain, A. 2010. The Sat4j library, release 2.2. *J. Satisf. Boolean Model. Comput.*, 7(2-3): 59–6.

Biere, A.; Faller, T.; Fazekas, K.; Fleury, M.; Froleyks, N.; and Pollitt, F. 2024. CaDiCaL 2.0. In Gurfinkel, A.; and Ganesh, V., eds., *CAV*, volume 14681 of *Lecture Notes in Computer Science*, 133–152. Springer.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. ISBN 978-1-64368-160-3.

Bjørner, N. S.; and Narodytska, N. 2015. Maximum Satisfiability Using Cores and Correction Sets. In *IJCAI*, 246–252. AAAI Press.

Bolusani, S.; Besançon, M.; Bestuzheva, K.; Chmiela, A.; Dionísio, J.; Donkiewicz, T.; van Doornmalen, J.; Eifler, L.; Ghannam, M.; Gleixner, A.; Graczyk, C.; Halbig, K.; Hedtke, I.; Hoen, A.; Hojny, C.; van der Hulst, R.; Kamp, D.; Koch, T.; Kofler, K.; Lentz, J.; Manns, J.; Mexi, G.; Mühmer, E.; Pfetsch, M. E.; Schlösser, F.; Serrano, F.; Shinano, Y.; Turner, M.; Vigerske, S.; Weninger, D.; and Xu, L. 2024. The SCIP Optimization Suite 9.0. Technical report, Optimization Online.

Conati, A.; Niskanen, A.; and Järvisalo, M. 2024. Declarative Approaches to Outcome Determination in Judgment Aggregation. *J. Artif. Intell. Res.*, 81: 793–836.

Crawford, J. M.; and Baker, A. B. 1994. Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems. In *AAAI*, 1092–1097. AAAI Press / The MIT Press.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer. ISBN 978-3-319-21274-6.

Davies, J.; and Bacchus, F. 2013. Exploiting the Power of MIP Solvers in MaxSat. In *SAT*, volume 7962 of *Lecture Notes in Computer Science*, 166–181. Springer.

Eén, N.; and Sörensson, N. 2003. Temporal induction by incremental SAT solving. In *BMC@CAV*, volume 89 of *Electronic Notes in Theoretical Computer Science*, 543–560. Elsevier.

Eén, N.; and Sörensson, N. 2006. Translating Pseudo-Boolean Constraints into SAT. *J. Satisf. Boolean Model. Comput.*, 2(1-4): 1–26.

Endriss, U. 2016. Judgment Aggregation. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*, 399–426. Cambridge University Press.

Fichte, J. K.; Berre, D. L.; Hecher, M.; and Szeider, S. 2023. The Silent (R)evolution of SAT. *Commun. ACM*, 66(6): 64–72.

Fichte, J. K.; Hecher, M.; and Szeider, S. 2020. Breaking Symmetries with RootClique and LexTopSort. In *CP*, volume 12333 of *Lecture Notes in Computer Science*, 286–303. Springer.

Fichte, J. K.; Lodha, N.; and Szeider, S. 2017. SAT-Based Local Improvement for Finding Tree Decompositions of Small Width. In *SAT*, volume 10491 of *Lecture Notes in Computer Science*, 401–411. Springer.

Fu, Z.; and Malik, S. 2006. On Solving the Partial MAX-SAT Problem. In *SAT*, volume 4121 of *Lecture Notes in Computer Science*, 252–265. Springer.

Glorian, G.; Lagniez, J.; Montmirail, V.; and Szczepanski, N. 2019. An Incremental SAT-Based Approach to the Graph Colouring Problem. In *CP*, volume 11802 of *Lecture Notes in Computer Science*, 213–231. Springer.

Grant, J. 1978. Classifications for Inconsistent Theories. *Notre Dame Journal of Formal Logic*, 19(3): 435–444.

Grant, J.; and Martinez, M. V., eds. 2018. *Measuring Inconsistency in Information*, volume 73 of *Studies in Logic*. College Publications.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2018. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In *SAT*, 428–437.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2019. RC2: an Efficient MaxSAT Solver. *J. Satisf. Boolean Model. Comput.*, 11(1): 53–64.

Ihalainen, H.; Berg, J.; and Järvisalo, M. 2022. Clause Redundancy and Preprocessing in Maximum Satisfiability. In *IJCAR*, volume 13385 of *Lecture Notes in Computer Science*, 75–94. Springer.

Ihalainen, H.; Oertel, A.; Tan, Y. K.; Berg, J.; Järvisalo, M.; Myreen, M. O.; and Nordström, J. 2024. Certified MaxSAT Preprocessing. In *IJCAR (1)*, volume 14739 of *Lecture Notes in Computer Science*, 396–418. Springer.

Leivo, M.; Berg, J.; and Järvisalo, M. 2020. Preprocessing in Incomplete MaxSAT Solving. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 347–354. IOS Press.

Li, C.; Xu, Z.; Coll, J.; Manyà, F.; Habet, D.; and He, K. 2021. Combining Clause Learning and Branch and Bound for MaxSAT. In *CP*, volume 210 of *LIPIcs*, 38:1–38:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

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

Li, C. M.; and Manyà, F. 2021. MaxSAT, Hard and Soft Constraints. In *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, 903–927. IOS Press.

Li, C. M.; Manyà, F.; and Planes, J. 2005. Exploiting Unit Propagation to Compute Lower Bounds in Branch and Bound Max-SAT Solvers. In *CP*, volume 3709 of *Lecture Notes in Computer Science*, 403–414. Springer.

Marques-Silva, J.; Lynce, I.; and Malik, S. 2021. Conflict-Driven Clause Learning SAT Solvers. In *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, 133–182. IOS Press, 2 edition.

Marques-Silva, J.; and Planes, J. 2007. On Using Unsatisfiability for Solving Maximum Satisfiability. *CoRR*, abs/0712.1097.

Morgado, A.; Dodaro, C.; and Marques-Silva, J. 2014. Core-Guided MaxSAT with Soft Cardinality Constraints. In *CP*, volume 8656 of *Lecture Notes in Computer Science*, 564–573. Springer.

Narodytska, N.; and Bacchus, F. 2014. Maximum Satisfiability Using Core-Guided MaxSAT Resolution. In *AAAI*, 2717–2723. AAAI Press.

Niskanen, A.; Kuhlmann, I.; Thimm, M.; and Järvisalo, M. 2023. MaxSAT-Based Inconsistency Measurement. In *ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, 1779–1786. IOS Press.

Paxian, T.; and Becker, B. 2023. Pacose: An Iterative SAT-based MaxSAT Solver. In *MaxSAT Evaluation 2023*, 20.

Paxian, T.; Reimer, S.; and Becker, B. 2018. Dynamic Polynomial Watchdog Encoding for Solving Weighted MaxSAT. In *SAT*, volume 10929 of *Lecture Notes in Computer Science*, 37–53. Springer.

Robertson, N.; and Seymour, P. D. 1986. Graph Minors. II. Algorithmic Aspects of Tree-Width. *J. Algorithms*, 7(3): 309–322.

Saikko, P.; Berg, J.; and Järvisalo, M. 2016. LMHS: A SAT-IP Hybrid MaxSAT Solver. In *SAT*, volume 9710 of *Lecture Notes in Computer Science*, 539–546. Springer.

Samer, M.; and Veith, H. 2009. Encoding Treewidth into SAT. In Kullmann, O., ed., *SAT*, volume 5584 of *Lecture Notes in Computer Science*, 45–50. Springer.

Shati, P.; Cohen, E.; and McIlraith, S. A. 2023. Optimal Decision Trees For Interpretable Clustering with Constraints. In *IJCAI*, 2022–2030. ijcai.org.

Thimm, M. 2018. On the Evaluation of Inconsistency Measures. In Grant, J.; and Martinez, M. V., eds., *Measuring Inconsistency in Information*, volume 73 of *Studies in Logic*. College Publications.

Thimm, M.; and Wallner, J. P. 2019. On the complexity of inconsistency measurement. *Artificial Intelligence*, 275: 411–456.

Van Gelder, A. 2008. Another look at graph coloring via propositional satisfiability. *Discret. Appl. Math.*, 156(2): 230–243.

Zheng, Z.; Cherif, S.; and Shibasaki, R. S. 2024. Optimizing Power Peaks in Simple Assembly Line Balancing Through Maximum Satisfiability. In *ICTAI*, 363–370. IEEE.