



HAL
open science

An Analysis of Core-Guided Maximum Satisfiability Solvers Using Linear Programming

George Katsirelos

► **To cite this version:**

George Katsirelos. An Analysis of Core-Guided Maximum Satisfiability Solvers Using Linear Programming. 26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023), Jul 2023, Alghero, Italy. 10.4230/LIPIcs.SAT.2023.12 . hal-04199433

HAL Id: hal-04199433



<https://hal.science/hal-04199433>

Submitted on 7 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An Analysis of Core-guided Maximum Satisfiability Solvers Using Linear Programming

George Katsirelos  

Université Paris-Saclay, AgroParisTech, INRAE, UMR MIA Paris-Saclay, 91120, Palaiseau, France

Abstract

Many current complete MaxSAT algorithms fall into two categories: core-guided or implicit hitting set. The two kinds of algorithms seem to have complementary strengths in practice, so that each kind of solver is better able to handle different families of instances. This suggests that a hybrid might match and outperform either, but the techniques used seem incompatible. In this paper, we focus on PMRES and OLL, two core-guided algorithms based on max resolution and soft cardinality constraints, respectively. We show that these algorithms implicitly discover cores of the original formula, as has been previously shown for PM1. Moreover, we show that in some cases, including unweighted instances, they compute the optimum hitting set of these cores at each iteration. We also give compact integer linear programs for each which encode this hitting set problem. Importantly, their continuous relaxation has an optimum that matches the bound computed by the respective algorithms. This goes some way towards resolving the incompatibility of implicit hitting set and core-guided algorithms, since solvers based on the implicit hitting set algorithm typically solve the problem by encoding it as a linear program.

2012 ACM Subject Classification Theory of computation → Discrete optimization; Mathematics of computing → Combinatorial optimization; Theory of computation → Logic; Mathematics of computing → Solvers

Keywords and phrases maximum satisfiability, core-guided solvers, minimum hitting set problem, linear programming

Digital Object Identifier 10.4230/LIPIcs.SAT.2023.11

Funding This work has been partly funded by the “Agence nationale de la Recherche” (ANR-19-PIA3-0004 ANITI-DIL chair of Thomas Schiex).

George Katsirelos:

1 Introduction

MaxSAT is the optimization version of SAT, in which we are given a set of *hard* clauses which must always be satisfied, as well as a set of weighted soft clauses, with the objective to find an assignment which minimizes the weight of the falsified soft clauses. Much like the case for SAT, the performance of MaxSAT solvers has been steadily improving over the past few years [5]. Two classes of algorithms have contributed significantly to this improvement: implicit hitting set (IHS) solvers [12, 14, 13, 6, 8] and core-guided solvers [18, 2, 24, 23, 22, 19]. Both are based on iteratively calling a SAT solver on formulas derived from the original MaxSAT instance and extracting unsatisfiable cores, but they are very different in their operation. IHS solvers exploit the hitting set duality of cores and correction sets (solutions)[26], and they try to build up a collection of cores that are enough to make the minimum hitting set match the optimum solution. Crucially, IHS solvers only ask the SAT solver to extract cores from subsets of the initial MaxSAT instance, which are all approximately equally hard. Core-guided solvers, on the other hand, reformulate the input instance with each core they discover so that it exhibits a higher lower bound. The reformulation generates ever more constrained formulas, which get harder and harder.

Despite their different approaches, both classes of algorithms are competitive, but they

perform well in different families of instances. Hence, it would be desirable to understand exactly how they relate to each other and build algorithms with the strength of both. In that direction, Bacchus and Narodytska [7] showed that the cores discovered by the PM1 [18] algorithm correspond to a collection of cores of the original instance. Later, Narodytska and Bjørner [25] showed that for unweighted instances, PM1 actually discovers a hitting set of these cores of the original formula at every iteration. These results showed that there exists a close relationship between IHS and core-guided solvers.

Here, we focus on PMRES [24] and OLL [22]. Our contributions are as follows.

- We show that, like PM1, each core computed by PMRES and OLL corresponds to a set of cores of the original MaxSAT instance.
- We identify a condition for when the lower bound computed by PMRES or OLL matches the optimum hitting set of the set of cores of the original formula. This includes the case when the input instance is unweighted.
- We show that the hitting set problem over these cores can be formulated compactly as an integer linear program for both PMRES and OLL. Moreover, the linear relaxation of that ILP has a lower bound which is at least as great as the bound computed by PMRES or OLL, respectively.
- The linear program that we give is actually a subset of a higher level relaxation of that hitting set problem in the Sherali-Adams hierarchy [28].

The first two contributions match what has been done for PM1 previously, although our proofs are notably simpler, owing to the fact that the cores of PMRES and OLL have a much more regular structure than those of PM1. The latter two contributions provide further insight into the relationship between these core-guided algorithms and IHS. The LP formulation points the way to an algorithm that combines features of both core-guided and implicit hitting set solvers, since IHS solvers typically solve the hitting set problem with an ILP solver: any bounds computed by PMRES or OLL can be imported into IHS by way of this LP. The fact that this LP is a subset of a high level Sherali-Adams relaxation also shows IHS and core-guided solvers as being two extreme instantiations of the same algorithmic framework, where both solvers try to solve an implicit hitting set problem. But whereas IHS discovers only cores of the original formula and offloads solving of the hitting set problem to an external solver, PMRES very aggressively searches for a non-obvious set of new variables to add to the linear relaxation of the hitting set problem, in order to keep it as close as possible to the optimum integer solution, but places a great burden on the SAT solver. This suggests a more effective tradeoff could be found.

2 Background

In addition to the basics of MaxSAT, we also introduce necessary background on linear programming and weighted constraint satisfaction problems (WCSPs).

2.1 Satisfiability

A SAT formula ϕ in conjunctive normal form (CNF) is a conjunction of clauses and a clause is a disjunction of literals. We also view a CNF formula as a set of clauses and a clause as a set of literals. For a CNF formula F , we write $vars(F)$ for the set of all variables whose literals appear in the clauses of F . The Weighted Partial MaxSAT (WPMS) problem is a generalization of SAT to optimization. A WPMS formula is a triple $W = \langle H, S, w \rangle$ where H is a set of *hard* clauses, S is a set of *soft* clauses and $w : S \rightarrow \mathbb{R}_{\geq 0}$ is a cost function over the

89 soft clauses. We also write $H(W) = H, S(W) = S, vars(W) = vars(H) \cup vars(S)$. For an
 90 assignment I over $vars(W)$, we overload notation to write $w(I) \triangleq \sum_{c \in S: I \vdash \neg c} w(c)$ for the
 91 cost of the soft clauses that I falsifies. The objective is to find an assignment I to $vars(W)$
 92 such that all clauses in H are satisfied and the cost of the falsified soft clauses, i.e., $w(I)$, is
 93 minimized. We write $opt(W) \triangleq \min_I w(I)$ for this value. A WPMS formula $\langle H, S, w \rangle$ with
 94 $w(c) = 1$ for all $c \in S$, is a partial MaxSAT formula. If, additionally, H is empty, it is a
 95 MaxSAT formula.

96 Two WPMS formula $W = \{H, S, w\}$ and $W' = \{H', S', w'\}$ are *equivalent* if for each
 97 assignment I to $vars(W)$ that satisfies H , we can extend it to an assignment I' to $vars(W')$
 98 that satisfies H' and $w(I) = w'(I') + b$, for some constant b that is the same for all
 99 assignments. For example, $W = \{\emptyset, \{(x), (\bar{x})\}, w\}$, where $w((x)) = 5, w((\bar{x})) = 3$ is equivalent
 100 to $W' = \{\emptyset, \{(x)\}, w'\}$, where $w'((x)) = 2$, because the weight of all assignments differs by 3
 101 in W, W' . This notion of equivalence is important in our subsequent analysis.

102 Given an unsatisfiable CNF formula F , a set $C \subset F$ is a core of F if C is unsatisfiable.
 103 If C is minimal by set inclusion, it is a MUS (minimal unsatisfiable subset) of F . Given a
 104 WPMS formula $W = \langle H, S, w \rangle$, a set $C \subseteq S$ is a core of W if $H \cup C$ is unsatisfiable. In the
 105 rest of this paper, we assume for simplicity that H is satisfiable and $H \cup S$ is unsatisfiable.

106 In the sequel, we make some assumptions without loss of generality. First, we assume that
 107 all soft clauses in a MaxSAT formula $W = \langle H, S, w \rangle$ are unit. If there exists a clause $c_i \in S$
 108 which is not unit, we create the formula $W' = \langle H', S', w' \rangle$ with $H' = H \cup \text{cnf}(\neg c_i \iff b_i)$,
 109 $S' = S \cup \{(\bar{b}_i)\} \setminus \{c_i\}$, where b_i is a fresh variable, called the *blocking variable* for c_i , and
 110 $w'((\bar{b}_i)) = w(c_i), w'(c) = w(c)$ for all $c \in S \cap S'$. We see that W is equivalent to W' by noting
 111 that we can extend any assignment of W to W' by setting b_i so that it satisfies $b_i \iff \neg c_i$.
 112 Moreover, we assume that the unique literal in all soft clauses appears with negative polarity.
 113 If this does not hold, we can make it so by renaming. Because of this assumption, we identify
 114 each soft clause with the unique variable it contains and we use that literal to refer to it.
 115 Finally, we assume that there exist no soft clauses with cost 0, as we can remove those
 116 without affecting satisfiability or cost. However, we use the convention that $w(x) = 0$ for
 117 all positive literals and all negative literals of variables that do not appear in a soft clause.
 118 Given this convention, a WPMS instance can be written as $W = \langle H, w \rangle$, and S is implicitly
 119 $S = \{(\bar{x}_i \mid w(\bar{x}_i) > 0)\}$. We use the two formulations interchangeably.

120 2.1.1 Solving WPMS

121 Most current SAT solvers have the ability to not only report SAT or UNSAT for a given
 122 formula, but also, given a partition of its clauses so that $\phi = \psi \cup \chi$, report a subset of χ such
 123 that $\psi \cup \chi$ is unsatisfiable. In terms of WPMS, it means a modern SAT solver can give a
 124 subset of S such that $H \cup S$ is unsatisfiable, i.e., a core of the WPMS formula. Because we
 125 have assumed that S contains negative unit clauses only, it follows that each core of W is a
 126 positive clause entailed by H .

127 The *implicit hitting set* (IHS) algorithm for WPMS [12, 14, 13, 6, 8] is based on the
 128 observation that the set of soft clauses $CS \subseteq S$ violated by a solution I is a hitting set of
 129 the set of all cores of W [26]. Hence, an optimal solution is a minimum hitting set of the
 130 cores of W . Hitting sets of all cores are called correction sets.

131 The IHS algorithm maintains an initially empty set of discovered cores \mathcal{C} of W and a
 132 minimum hitting set of \mathcal{C} , $hs(\mathcal{C})$. If the SAT formula $H \cup (S \setminus hs(\mathcal{C}))$ is satisfiable, then
 133 its solutions are optimal solutions of W and $w(hs(\mathcal{C})) = w(W)$. Otherwise, a new core is
 134 extracted and added to \mathcal{C} and the loop repeats. Actual implementations of the IHS algorithm
 135 in MaxHS [12] and LMHS [27, 9] contain many optimizations over this basic loop.

136 A *core-guided* algorithm for WPMS [18, 24, 23, 22, 19] is an iterative algorithm that
 137 generates a sequence of WPMS instances $W^0 = \langle H^0, w^0 \rangle = W, \dots, W^m = \langle H^m, w^m \rangle$ and a
 138 sequence of lower bounds $lb^0 = 0 < lb^1 < \dots < lb^m$ such that $H^i \models H^{i-1}$ for all $i \in [1, m]$
 139 and W^0 is equivalent to W^i for all $i \in [1, m]$ and the weights of the assignments differ by lb_i ,
 140 therefore $opt(W) = lb_i + opt(W^i)$. Moreover, in the last iteration it holds $opt(W^m) = 0$, so
 141 $opt(W) = lb^m$. In words, a core-guided algorithm generates a sequence of equivalent WPMS
 142 instances such that each successive instance is used to derive an increased lower bound
 143 for the original instance, while decreasing the cost of every solution by the same amount.
 144 The final instance admits a solution with zero weight, and each such solution of W^m is an
 145 optimal solution of W . All such solutions are solutions of the SAT formula $H^m \upharpoonright_0$, defined as
 146 $H^m \cup (\bar{x}) \mid w(x) > 0$, i.e., with all soft clauses made into hard clauses. In order to derive each
 147 successive instance W^{i+1} in the sequence, it extracts a core from W^i and uses it to transform
 148 it into W^{i+1} and increase the lower bound, hence the name core-guided. The algorithms
 149 we study here, PMRES and OLL, are core-guided algorithms. Following Narodytska and
 150 Bjørner [25], we call cores of W^i for $i > 0$ *meta cores*, or *metas*, to distinguish them from
 151 cores of the original formula W^0 . We write m^i for the meta discovered at iteration i .

152 2.2 Linear programming and Weighted Constraint Satisfaction

153 An integer linear program (ILP) IP has the form $\min c^T x : Ax \geq b \wedge x \in \mathbb{Z}_{\geq 0}$, where x
 154 is a vector of n variables, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. For a given x , if $Ax \geq b$, then it
 155 is a feasible solution of IP. We write $c(x) = c^T x$ for the *cost*¹ of x . We write $c(IP)$ for
 156 the cost of a feasible solution with minimum cost. The *linear relaxation* P of IP is the
 157 problem $\min c^T x : Ax \geq b \wedge x \in \mathbb{R}_{\geq 0}^n$, i.e., one where we relax the integrality constraint
 158 $x \in \mathbb{Z}_{\geq 0}^n$. This is called a linear program (LP). Linear programs have the *strong duality*
 159 property, namely that for every linear program P in the above form, there exists another
 160 linear program $P^D = \max b^T y : A^T y \leq c \wedge y \in \mathbb{R}_{\geq 0}^m$, with the property that $c_{PD}(\hat{y}) \leq c_P(\hat{x})$
 161 for every feasible solution \hat{x} of P and \hat{y} of P^D and $c_{PD}(y^*) = c_P(x^*)$ for optimal solutions x^*
 162 and y^* . Given a feasible dual solution \hat{y} , the value $A_i^T \hat{y} - c_i$, the slack of the dual constraint
 163 corresponding to the primal variable x_i , is called the *reduced cost* of x_i , denoted $rc_i(\hat{y})$. A
 164 necessary condition for optimality called *complementary slackness* links the two solutions:
 165 $x_i^* rc_i(y^*) = 0$, i.e., for each variable x_i , either it is zero or its corresponding dual constraint
 166 ($A_i y \leq c_i$) is tight (has *zero slack*).

167 A Boolean Cost Function Network (CFN) is a pair $\langle V, D, C \rangle$, where V is a set of variables,
 168 D is a function mapping variables to domains, and C is a set of cost functions. If the domain
 169 of a variable v is binary, we write v for the value $v = 1$ and \bar{v} for $v = 0$. Each cost function
 170 is a pair $\langle S, c \rangle$ where $S \subseteq V$ is its scope and c_S is a function $\prod_{x \in S} D(x) \rightarrow \mathbb{R}_{\geq 0} \cup \infty$. We
 171 assume there exists at most one cost function for each scope, so c_S is a shortcut for $\langle S, c_S \rangle$.
 172 An assignment I_S to a scope S is a function which maps every variable $x \in S$ to a value
 173 in $D(x)$. When we omit S , it means $S = V$. When convenient, we also use I to denote
 174 the set $\{v = a \mid I(v) = a, v \in V\} \cup \{v \neq b \mid I(v) \neq b, v \in V, b \in D(x)\}$. For a scope S and
 175 assignment I , $I_{\downarrow S}$ is the projection of I to S . $t(S)$ denotes all possible assignments to S .

176 We use the convention that for a cost function c_S , $c_S(I) = c_S(I_{\downarrow S})$, i.e., we implicitly
 177 project to S . For a CFN P , we write $c_P(I) = \sum_{c_S \in F} c_S(I)$. The Weighted Constraint
 178 Satisfaction Problem (WCSP) is to find an assignment I such that $c_P(I) < \infty$ and that

¹ We stick to the terminology of *weights* in MaxSAT and *costs* in ILP and WCSP, even though they serve the same purpose.

179 minimizes c_P . The term WCSP is often used to refer both to the underlying CFN and to
 180 the optimization problem, and we do the same here. Additionally, we assume the existence
 181 of a unary cost function $c_{\{v\}}$ (abbreviated as c_v) for every variable $v \in V$ and a nullary cost
 182 function c_{\emptyset} , which is a lower bound for c_P , because all costs are non-negative. A CSP is a
 183 WCSP in which the domain of all cost functions is $\{0, \infty\}$.

184 A WCSP $P = \langle V, C \rangle$ can be formulated as the following ILP:

$$185 \quad \min \sum_{c_S \in C, l \in t(S)} c_S(l) x_{Sl} \quad (1)$$

$$186 \quad \text{s.t.} \quad (2)$$

$$187 \quad x_{\{v\},a} = \sum_{l \in t(S): v=a \in l} x_{Sl} \quad \forall v \in V, a \in D(v), c_S \in C \quad (3)$$

$$188 \quad \sum_{a \in D(v)} x_{\{v\},a} = 1 \quad \forall v \in V \quad (4)$$

$$189 \quad x_{Sl} \in \mathbb{Z}_{\geq 0} \quad \forall v \in V, c_S \in C, l \in t(S) \quad (5)$$

190 The linear relaxation of (1)–(5) defines the *local polytope* of P . A dual feasible solution
 191 of the local polytope LP has a particular interpretation: it defines a *reformulation* of the
 192 WCSP. A reformulation can be seen as a set of operations on a WCSP P that create a
 193 new WCSP \hat{P} with modified costs, but $c_P(I) = c_{\hat{P}}(I)$ for all I . Therefore, a reformulation
 194 is said to *preserve equivalence*. This notion of equivalence is identical to the equivalence
 195 preserved by core-guided algorithms, with the primary difference being that the lower bound
 196 is explicitly represented in a WCSP in c_{\emptyset} . These operations can intuitively be thought of as
 197 moving cost among cost functions:

- 198 ■ **Extention:** $ext(v = a, c_S, \alpha)$, with $v \in S, a \in D(v)$. This subtracts cost α from $c(\{v\}, a)$
 199 and adds it to $c(S, l)$ for all tuples $l \in t(S) : (v = a) \in l$. To see the correctness of this,
 200 consider the subset of the objective function $c_v(a)x_{\{v\},a} + \sum_{l \in t(S): (v=a) \in l} c_S(l)x_{Sl}$, as
 201 well as constraint (3). Since $x_{\{v\},a}$ is equal to the sum, the value of the objective remains
 202 unchanged by adding α to one and subtracting it from the other.
- 203 ■ **Projection:** $prj(c_S, v = a, \alpha)$, with $v \in S, a \in D(v)$. This is the same as $ext(v, c_S, -\alpha)$.
- 204 ■ **Nullary projection:** $prj_0(c_S, w)$. This subtracts cost α from each tuple $l \in t(S)$ and moves
 205 it to c_{\emptyset} . This is justified because $\sum_{l \in t(S)} x_{Sl} = 1$ and the cost of c_{\emptyset} is a constant in the
 206 objective function.

207 Because these operations preserve equivalence, they are called Equivalence Preserving
 208 Transformations (EPTs). A valid set of EPTs ensures that all cost functions are non-negative
 209 everywhere, but there are valid sets of EPTs for which any sequence of performing them
 210 leaves intermediate negative costs. A valid set of EPTs can be mapped to a feasible dual
 211 solution of the local polytope LP and vice versa. A set of EPTs which achieves the greatest
 212 increase in c_{\emptyset} , and hence the lower bound, can be mapped to an optimal dual solution of
 213 the local polytope LP [11]. Given a dual solution, the cost of each tuple $l \in t(S)$ is given by
 214 the reduced cost of the variable x_{Sl} .

215 For a WCSP P , let $Bool(P)$ be the CSP (not weighted) defined by accepting exactly
 216 those tuples which have cost 0, i.e., changing all costs which are greater than 0 to ∞ . Let
 217 \hat{P} be a reformulation of P . A consequence of complementary slackness is that if \hat{P} is an
 218 optimal reformulation, then $Bool(\hat{P})$ has a non-empty arc consistency closure [11, 15], in
 219 which case it is said that \hat{P} is virtually arc consistent (VAC). This is not a sufficient condition

11:6 Analysis of Core-guided MaxSAT Using Linear Programming

220 for optimality, however. Conversely, if \hat{P} is not VAC, therefore $Bool(\hat{P})$ has an empty arc
 221 consistency closure, there exists a reformulation with a higher c_\emptyset .

222 **3** PMRES

223 The PMRES algorithm is a core guided solver which was introduced by Narodytska and
 224 Bacchus [24] and is implemented primarily in the EVA solver. We describe it briefly here. In
 225 this description, we use the view of WPMS as hard and soft clauses, rather than hard clauses
 226 and an objective, because the transformations performed by PMRES temporarily violate
 227 the assumptions that allow us to take this alternative view. However, these assumptions are
 228 always restored at the end of each iteration.

229 **3.1 Max-Resolution**

230 Max resolution [20] is a complete inference rule for MAXSAT [10]. It consists of the following
 231 rule on soft clauses, in which the conclusions replace the premises:

$$\frac{(A \vee x, w) \quad (B \vee \bar{x}, w)}{(A \vee B, w) \quad (A \vee x \vee \bar{B}, w) \quad (B \vee \bar{x} \vee \bar{A}, w)}$$

232 The first clause in the conclusions is equivalent to what resolution derives. The latter
 233 two are called compensation clauses, as they compensate for the cost of assignments which
 234 do not falsify the conclusion $A \vee B$ but falsify one of the discarded premises. Depending on
 235 the exact form of A and B , the compensation “clauses” may not actually be in clausal form
 236 and would have to be converted to a set of clauses each. We ignore this complication here,
 237 as our presentation of PMRES mostly avoids this case.

238 The rule is generalized to clauses with different costs $w_1 > w_2$ by cloning the heavier
 239 clause into clauses with costs w_2 and $w_1 - w_2$. When one of the clauses is hard, e.g., $w_1 = \infty$,
 240 we keep it in the conclusions.

241 Max resolution has the property that if W and \hat{W} are the formulas before and after
 242 application of the rule, then they are equivalent.

243 **3.2 Max-Resolution with cores**

244 PMRES uses the specialization of this rule for a binary clause and a unit clause, i.e.,
 245 $|A| = 1, B = \emptyset$.

$$\frac{(A \vee x, w) \quad (\bar{x}, w)}{(A, w) \quad (\bar{x} \vee \bar{A}, w)}$$

246 As a core-guided solver, PMRES is an iterative algorithm and the first step in each
 247 iteration is to extract a meta core from W^i , or terminate if $H^i \cup S^i$ is satisfiable. Suppose
 248 that the meta is $m^i = \{b_1^i, b_2^i, \dots, b_{r^i}^i\} \subseteq S^{i-1}$ and $w_{\min}^i = \min_{b_j \in C} c^i(b_j^i)$. This implies
 249 the presence of the soft clauses $(\bar{b}_1^i, w_1), \dots, (\bar{b}_{r^i}^i, w_{r^i})$. PMRES first splits each soft clause
 250 (\bar{b}_j^i, w') with $w' > w_{\min}^i$ into $(\bar{b}_j^i, w_{\min}^i)$ and $(\bar{b}_j^i, w' - w_{\min}^i)$. This temporarily violates our
 251 assumption that each soft clause contains a unique literal, but as we will see, this invariant

$$\begin{array}{c} \hline (b_1 \vee b_2 \vee b_3 \vee b_4) \\ (b_5 \vee b_2) \\ (b_5 \vee b_3 \vee b_4) \\ \hline \end{array}$$

■ **Figure 1** Cores of the instance used in the running example

252 is restored before the next iteration starts. In the next step, it adds to H^{i+1} the hard clause
 253 corresponding to C using the CNF encoding of $(b_1^i \vee d_1^i), (d_1^i \iff b_2^i \vee d_2^i), \dots, (d_{r-2}^i \iff$
 254 $b_{r-1}^i \vee d_{r-1}^i), (d_{r-1}^i \iff b_r^i)$, where d_1^i, \dots, d_{r-1}^i are fresh variables. It is clear that we can
 255 recover the clause $(b_1^i \vee \dots \vee b_r^i)$ by resolving (not with max-resolution, as the clauses are
 256 all hard) the first two clauses on d_1^i , then on d_2^i , and so on, therefore the encoding and the
 257 clause are equivalent. PMRES then applies max-resolution as follows:

Premises		Conclusions
$(b_1^i \vee d_1, w_{\min}^i)$	$(\bar{b}_1^i, w_{\min}^i)$	$(d_1, w_{\min}^i), \boxed{(\bar{b}_1^i \vee \bar{d}_1, w_{\min}^i)}$
(d_1, w_{\min}^i)	$(\bar{d}_1 \vee b_2^i \vee d_2, w_{\min}^i)$	$(b_2^i \vee d_2, w_{\min}^i), ((b_2^i \vee d_2) \vee d_1, w_{\min}^i)$
⋮		
$(b_{r-1}^i \vee d_{r-1}, w_{\min}^i)$	$(\bar{b}_{r-1}^i, w_{\min}^i)$	$(d_{r-1}, w_{\min}^i), \boxed{(\bar{b}_{r-1}^i \vee \bar{d}_{r-1}, w_{\min}^i)}$
(d_{r-1}, w_{\min}^i)	$(\bar{d}_{r-1} \vee b_r^i, w_{\min}^i)$	$(b_r^i, w_{\min}^i), (\bar{b}_r^i \vee d_{r-1}, w_{\min}^i)$
(b_r^i, w_{\min}^i)	$(\bar{b}_r^i, w_{\min}^i)$	$\boxed{(\square, w_{\min}^i)}$

259 The non-clausal constraints in light gray are tautologies and can be discarded. For
 260 example, by $(d_1^i \iff b_2^i \vee d_2^i), (\bar{b}_2^i \vee \bar{d}_2^i) \vee d_1^i$ is equivalent to $(\bar{d}_1^i \vee d_1^i)$, a tautology. The
 261 clauses in gray are used as input for the next max-resolution step. The framed clauses are
 262 new soft clauses that are kept for the next iteration. Since they are not unary, they are
 263 reified using fresh variables and converted to unit soft clauses, e.g., $f \iff b_1^i \wedge d_1^i$ and (\bar{f}, w) ,
 264 where f is the fresh variable. Finally, the empty soft clause (\square, w_{\min}) is used to increase the
 265 lower bound for the next iteration by w_{\min} .

266 Consider now a clause (\bar{b}_j^i, w') that was split into two clones (\bar{b}_j^i, w_{\min}) and $(\bar{b}_j^i, w' - w_{\min})$.
 267 The former is consumed by max-resolution, therefore the invariant that each soft clause
 268 contains a unique literal is restored. This also allows us to implement the cloning process as
 269 a simple update: $w^{i+1}(b_j) = w^i(b_j) - w_{\min} = w' - w_{\min}$. If it happens that $w' = w_{\min}$, we
 270 maintain by the previously mentioned convention that $w^{i+1}(b_j) = 0$.

271 In the following, we write H_R^i for the formula consisting only of the clauses introduced
 272 by PMRES, therefore $H^i = H \cup H_R^i$. We also write F^i and D^i for the set of all variables,
 273 introduced to reify soft clauses (e.g. f above) or to encode the meta core clause (the d_j^i
 274 variables above), respectively. It has also been previously noted [25, 3] that the conjunction
 275 of the definitions of the F and D and the clauses $(b_1^i \vee d_1^i)$ define a monotone circuit, with a
 276 binary gate corresponding to each $v \in F^i \cup D^i$, an unnamed \vee gate corresponding to the
 277 clause $(b_1^i \vee d_1^i)$, and an implicit \wedge gate whose inputs are the unnamed \vee gates, which is the
 278 output of the circuit.

279 ► **Example 1** (Running Example). Consider an instance W with 5 soft clauses with cost 1
 280 each and corresponding literals b_1, \dots, b_5 , and the cores shown in Figure 1. We show a run
 281 of PMRES in Figure 2 (for readability, we show the objective function rather than the set of
 282 soft clauses) that discovers first the core $(b_1 \vee b_2 \vee b_3 \vee b_4)$. It increases the lower bound by
 283 1, adds the variables $D^1 = \{d_1^1, d_2^1, d_3^1\}$ and $F^1 = \{f_1^1, f_2^1, f_3^1\}$, defined as shown in the row
 284 corresponding to iteration 1. Since weights are unit, all original variables except b_5 disappear

Iteration	Meta	New clauses	Objective
1	$\{b_1, b_2, b_3, b_4\}$	$d_1^1 \iff b_2 \vee d_2^1, d_2^1 \iff b_3 \vee d_3^1,$ $d_3^1 \iff b_4,$ $f_1^1 \iff b_1 \wedge d_1^1, f_2^1 \iff b_2 \wedge d_2^1,$ $f_3^1 \iff b_3 \wedge d_3^1$	$1 + b_5 + f_1^1 + f_2^1 + f_3^1$
2	$\{f_2^1, b_5\}$	$d_1^2 \iff b_5$ $f_1^2 \iff b_7 \wedge d_1^2$	$2 + f_1^1 + f_3^1 + f_1^2$

■ **Figure 2** PMRES on the running example

285 from the objective. In the next iteration, PMRES discovers the meta $\{b_5, f_2^1\}$, increases the
 286 lower bound to 2, and introduces the variables d_1^2 and f_1^2 . In the next iteration, the instance
 287 is satisfiable. One of the possible solutions is b_4, b_5 , with cost 2, which matches the lower
 288 bound.

289 3.3 Cores and Hitting Sets of PMRES

290 We first observe that the f^i and d^i variables created on iteration i are functionally dependent
 291 on the b^i variables. Therefore, the formula H^i generated after the i th iteration is logically
 292 equivalent to H , i.e., every solution of H can be extended to exactly one solution of H^i .

293 ► **Lemma 2.** *There exists a set \mathcal{C}^i such that m^i is a core of H^i if and only if for each $c \in \mathcal{C}^i$,*
 294 *c is a core of ϕ .*

295 **Proof.** The set \mathcal{C}^i can be derived from m^i and H_R^i by forgetting the variables f and d
 296 that were introduced by PMRES. More concretely, let $E^0 = \{m^i\}$. If there exists $c \in E^j$
 297 such that $f \in c$ and f was introduced by PMRES and defined as $f \iff b \wedge d$, we set
 298 $E^{j+1} = E^j \setminus \{c\} \cup \{c \setminus \{f\} \cup \{b\}, c \setminus \{f\} \cup \{d\}\}$, i.e., we replace c by two clauses which have b
 299 and d , respectively, instead of f . If there exists $c \in E^j$ such that $d \in c$ and d was introduced
 300 by PMRES and defined as $d \iff b \vee d'$, we set $E^{j+1} = E^j \setminus \{c\} \cup \{c \setminus \{d\} \cup \{b, d'\}\}$, i.e.,
 301 we replace d by b, d' in c . The process eventually terminates because it removes one reference
 302 to a variable introduced by PMRES and replaces it by a variable corresponding to a gate at
 303 a deeper level of the Boolean circuit defined by H_R^i , hence all variables must eventually be
 304 original variables of W^0 . It is also confluent because the choice of variable to forget does not
 305 hinder other choices.

306 Since both forgetting variables and introducing functionally defined variables are satisfiability-
 307 preserving operations, we have $m^i \wedge H_R^i \models \mathcal{C}^i$ and $\mathcal{C}^i \models m^i \wedge H_R^i$. ◀

308 ► **Lemma 3.** *Let $hs \subseteq S$. Then hs as an assignment can be extended to a solution of H_R^i if*
 309 *and only if it is a hitting set of \mathcal{C}_\cup^i .*

310 **Proof.** This follows from lemma 2.

311 (\Rightarrow) hs satisfies H_R^i , hence it satisfies all clauses in \mathcal{C}^i , which are cores, so it hits all the
 312 cores.

313 (\Leftarrow) hs is a hitting set of \mathcal{C}^i , hence it satisfies all the corresponding clauses, hence it
 314 satisfies H_R^i . ◀

315 In the following, let $\mathcal{C}_\cup^i = \cup_{j \in [1, i]} \mathcal{C}^j$.

316 ► **Observation 4.** $\langle H_R^i, w^0 \rangle$ and $\langle H_R^i, w^i \rangle$ are equivalent.

317 **Proof.** Consider $H_0 = \langle H_R^i, w^0 \rangle$. We know that m^0 is a core of H_0 . By applying max
 318 resolution to m^0 as described in section 3.2, we get new variables and soft clauses. But these
 319 new variables are defined identically to the variables PMRES introduced to get H_R^1 , which
 320 is a subset of H_R^i . Hence, we can identify them. By correctness of PMRES, we get that
 321 $\langle H_R^i, w^1 \rangle$ is equivalent to $\langle H_R^i, w^0 \rangle$. We apply the same argument inductively to complete
 322 the proof. ◀

323 ▶ **Corollary 5.** *The WPMS $W_i^{hs} = \langle H_R^i, w^i \rangle$ encodes the minimum hitting set problem over*
 324 \mathcal{C}_\cup^i , *with weights shifted by lb^i . Hitting sets with cost lb^i , if they exist, are solutions of W_i^{hs}*
 325 *that use only soft clauses with soft 0.*

326 **Proof.** From Lemma 3, $\langle H_R^i, w^0 \rangle$ encodes minimum hitting set over \mathcal{C}_\cup^i . From Observation 4,
 327 $\langle H_R^i, w^0 \rangle$ and $\langle H_R^i, w^i \rangle$ are equivalent, therefore W_i^{hs} encodes minimum hitting set over \mathcal{C}_\cup^i .

328 The second part follows from the fact that, for any assignment I , $w^0(I) = lb_i + w^i(I)$, so
 329 if $w^0(I) = lb_i$, then $w^i(I) = 0$. ◀

330 Let us denote by $H_R^i \mid_0$ the formula H_R^i with all variables x such that $w(x) > 0$ set
 331 to false so that all models of $H_R^i \mid_0$ are minimum hitting sets of \mathcal{C}^i . Therefore if $H_R^i \mid_0$ is
 332 satisfiable, the bound computed by PMRES matches the cost of the minimum hitting set of
 333 \mathcal{C}_\cup^i .

334 ▶ **Lemma 6.** *If W is a PMS instance, $H_R^i \mid_0$ is satisfiable for all iterations i of PMRES.*

335 **Proof.** ■ All variables in D have cost 0.

336 ■ Moreover, all variables which appear in any meta have cost 0, because it is moved away
 337 by max-resolution.

338 ■ Therefore, all variables in $b_1^j, \dots, b_{r_j}^j$ for $j \in [1, i]$ have zero cost.

339 We construct a solution to $H_R^i \mid_0$ by setting to false all variables which are inputs to
 340 false \wedge -gates (which is done by unit propagation), then we set variables to true by traversing
 341 metas in reverse chronological order:

342 1. For m^i , we pick the first variable in $b_1^j, \dots, b_{r_j}^j$ and set it to true. We set all variables in
 343 F^i and D^i to false (the former is required for m^i because, as the last discovered core, all
 344 variables in F^i have non-zero weight).

345 2. Supposing we have satisfied all metas m^{j+1}, \dots, m^i , consider m^j . Suppose that $0 \leq$
 346 $q < |m^j|$ variables in F^j that have been set to true by previous steps, with indices
 347 $P^j = \{p_1, \dots, p_q^j\}$. For simplicity of notation, assume that if P^j is empty, then $p_q^j = 0$.

348 Then we set to true the variables $b_r^j \mid r \in P^j$ as well as $b_{p_q+1}^j$, and set the rest to false.
 349 When $p_q^j = 0$, this reduces to setting the first variable in b_1^j to true.

350 a. This assignment satisfies the constraints introduced in H_R^j .

351 b. Moreover, all the variables that appear in m^j have cost 0 after the j^{th} iteration.
 352 Therefore they cannot appear in any meta discovered in iterations $j+1, \dots, i$ and
 353 the assignment we have chosen here does not contradict the assignments chosen in
 354 iterations $j+1, \dots, i$. ◀

356 We can see where the proof of Lemma 6 breaks when applied to WPMS: the assertion 2b
 357 does not hold, because a variable whose cost has not been reduced to 0 may appear in later
 358 metas and our procedure may therefore create a conflicting assignment.

11:10 Analysis of Core-guided MaxSAT Using Linear Programming

Iteration	Core	New clauses	Objective
1	$\{b_1, b_2, b_3, b_4\}$	$d_1^1 \iff b_2 \vee d_2^1, d_2^1 \iff b_3 \vee d_3^1,$ $d_3^1 \iff b_4,$ $f_1^1 \iff b_1 \wedge d_1^1, f_2^1 \iff b_2 \wedge d_2^1,$ $f_3^1 \iff b_3 \wedge d_3^1$	$1 + b_2 + 2b_3 + 3b_4 + 5b_5 +$ $f_1^1 + f_2^1 + f_3^1$
2	$\{f_2^1, b_5\}$	$d_1^2 \iff b_5$ $f_1^2 \iff f_2^1 \wedge d_1^2$	$2 + b_2 + 2b_3 + 3b_4 + 4b_5 +$ $f_1^1 + f_3^1 + f_1^2$
3	$\{b_3, b_4, b_5\}$	$d_1^3 \iff b_4 \vee d_2^3, d_2^3 \iff b_5$ $f_1^3 \iff b_3 \wedge d_1^3, f_2^3 \iff b_4 \wedge d_2^3$	$4 + b_2 + b_4 + 2b_5 +$ $f_1^1 + f_3^1 + f_1^2 + 2f_1^3 + 2f_2^3$
4	$\{b_2, b_5\}$	$d_1^4 \iff b_5$ $f_1^4 \iff b_2 \wedge d_1^4$	$5 + b_4 + b_5 +$ $f_1^1 + f_3^1 + f_1^2 + 2f_1^3 + 2f_2^3 + f_1^4$

■ **Figure 3** PMRES on the running example with modified, non-unit weights.

359 ► **Example 7** (PMRES on a weighted formula). Consider the running example, but with the
360 modified weights (1, 2, 3, 4, 5), respectively. We assume the same trail as shown in figure 2,
361 and show in figure 3 the modified execution. After the first two iterations the lower bound will
362 be 2, as shown. The optimum hitting set is $\{b_2, b_3\}$ with cost 5, so the lower bound does not
363 match the optimum. Indeed, $H_R^2 \upharpoonright_0$ is unsatisfiable: the clause $(f_2^1 \vee b_5)$ can only be satisfied
364 by f_2^1 , because $w^2(b_5) > 0$. But $f_2^1 \iff b_2 \wedge (b_3 \vee b_4)$ and $w^2(b_2) > 0, w^2(b_3) > 0, w^2(b_4) > 0$,
365 therefore f_2^1 is forced to false. Hence, PMRES has to perform more iterations before matching
366 the bound of the hitting set. A possible trail finds the metas $\{b_3, b_4, b_5\}$ and $\{b_2, b_5\}$ (which
367 also happen to be cores of W^0), as shown.

368 We are now ready to state the main result of this section.

369 ► **Theorem 8.** *For a PMS instance, at each iteration, PMRES computes an optimum hitting*
370 *set of \mathcal{C}_\cup^i .*

371 **Proof.** Follows from Lemma 2, Corollary 5, and Lemma 6. ◀

372 For a WPMS instance, we can get a weaker result: since cores of $H_R^i \upharpoonright_0$ are also cores of
373 $H^i \upharpoonright_0$, we can extract cores of $H_R^i \upharpoonright_0$, which are metas of W until it becomes satisfiable, at
374 which point the bound is a hitting set of \mathcal{C}_\cup^i . It is not clear if that is a desirable thing to do
375 from a performance perspective.

376 3.4 PMRES and Linear Programming

377 In this section, we prove the following.

378 ► **Theorem 9.** *There exists an integer linear program $ILLP_P^i$ which (1) is logically equivalent*
379 *to the minimum hitting set problem with sets \mathcal{C}_\cup^i , (2) has size polynomial in $|H_R^i|$, and (3)*
380 *whose linear relaxation has an optimum which matches that derived by PMRES.*

381 Given the results of section 3.3, (1) is easy to show, since we can generate the set \mathcal{C}_\cup^i ,
382 then write the hitting constraint for each set in \mathcal{C}_\cup^i , and use w^0 as the objective. Call this
383 $ILLP_{hs}^i$. But $ILLP_{hs}^i$ may be exponentially larger than H_R^i . It is not much harder to show
384 that we can achieve (1) and (2). As Corollary 5 shows, H_R^i is logically equivalent to that
385 hitting set problem, so we can replace the constraints of $ILLP_{hs}^i$ by H_R^i (i.e., by the standard
386 encoding of clauses to linear constraints) and get an equivalent problem. Call that $ILLP_R^i$
387 and its linear relaxation LP_R^i .

388 However, we can see that LP_R^i is weak, specifically, that $c(LP_R^i) < c(ILLP_R^i)$.

389 ► **Example 10** (Running example, continued). Consider the ILPs $ILLP_{hs}^2$ and $ILLP_R^2$ corresponding to the hitting set problems for the 2^{nd} iteration of PMRES on the instance W in our running example. The optimum of both $ILLP_{hs}^2$ and $ILLP_R^2$ is 2, as expected, but the optimum of LP_R^2 is only 1.5.

393 In this specific example, since we have integer costs, the bound of the linear relaxation allows us to derive a bound of 2 for $ILLP_R^2$, but in general we can get an arbitrarily large difference. This is not surprising in general, but the fact that PMRES does compute an optimal hitting set at each iteration suggests that we should be able to do better. This is the objective of this section.

398 To construct an LP that meets the requirement of the theorem, we give a WCSP and its reformulation, which yield an LP (the local polytope) and a dual solution (one which is created from the formulation), as described in section 2.2. The result could be proved by directly giving an appropriate LP and dual solution, and proving the result on that, but it would be more cumbersome and would lack the existing intuitive understanding that has been developed in WCSP of dual solutions as reformulations.

404 **Proof of theorem 9.** We will give first a WCSP P^i which admits the same solutions as H_R^i and has unary costs such that its feasible solutions have the same cost as the hitting set problem entailed at iteration i of PMRES. This means that the optimum solution of P^i matches the minimum hitting set of \mathcal{C}_\cup^i . Further, we show that its linear relaxation $LP(P^i)$ admits a dual feasible solution whose cost matches the bound computed by PMRES. We give this dual solution as a sequence of equivalence preserving transformations of P^i , using the results presented in section 2.2. That linear program, $LP(P^i)$, satisfies the requirements of the theorem.

412 We first define P^i . The high level idea is that we encode the objective function of $ILLP_R^i$ directly as unary costs, and each meta using the well known decomposition into ternary constraints. The d variables have exactly the same semantics as the auxiliary variables used in that decomposition. The corresponding f variable corresponds to a single tuple of these ternary constraints, so we add an f variable to each ternary constraint in order to capture the cost of that ternary tuple into a unary cost. More precisely, let $P^0 = \emptyset$. At iteration i , where the core discovered is $\{b_1^i, b_2^i, \dots, b_r^i\} \subseteq S^{i-1}$, P^i is defined as P^{i-1} and additionally the following variables and cost functions:

- 420 ■ 0/1 variables $b_1, \dots, b_n, d_j^i, f_k^i$, corresponding to the propositional variables of the same name in W^i .
- 422 ■ Unary cost functions with scope b_i for each $b_i \in vars(W^0)$, with $c_{b_i}(0) = 0, c_{b_i}(1) = c^0(b_i)$
- 423 ■ A ternary cost function with scope $\{b_1^i, d_1^i, f_1^i\}$ where each tuple that satisfies $b_1^i \vee d_1^i$ and $f_1^i \iff d_1^i \wedge b_1^i$ has cost 0 and the rest have infinite cost.
- 425 ■ Quaternary cost functions with scope $\{b_j^i, d_{j-1}^i, d_j^i, f_j^i\}$, for $j \in [2, r-2]$, where each tuple that satisfies $d_{j-1}^i \iff d_j^i \vee b_j^i$ and $f_j^i \iff d_j^i \wedge b_j^i$ has cost 0 and the rest have infinite cost.
- 428 ■ A binary cost function with cost 0 for each tuple that satisfies $d_{r-1}^i = b_r^i$ and infinite cost otherwise.

430 It is straightforward to see that P^i is equivalent to $ILLP_R^i$: (i) they have the same set of variables, (ii) the only costs in P^i are in unary cost functions, so the objective functions are the same, (iii) the quaternary cost functions satisfy, by construction, the clauses included in the scope of these functions, and (iv) each clause is present in one cost function. Therefore,

434 solutions of P^i are hitting sets of \mathcal{C}_U^i and the cost of each solution matches the cost of the
435 corresponding hitting set.

436 It remains only to show that the LP optimum of $relax(P^i)$ matches that produced by PM-
437 RES. We show a slightly stronger result, namely that there exists a sequence of EPTs such that
438 in P^i , not only does the bound match that produced by PMRES, but the unary costs of each
439 variable match the weights computed by PMRES. We show this by induction on the number
440 of iterations. At iteration 0, this holds trivially, as the bound is 0 for both P^0 and PMRES
441 and the unary costs match the weights by construction. Suppose it holds at iteration $k - 1$.
442 Then, the core at iteration k is $\{b_1^k, b_2^k, \dots, b_r^k\} \subseteq S^{k-1}$. The EPT $ext(b_1^k, \{b_1^k, d_1^k, f_1^k\}, w_{\min}^k)$
443 enables the EPTs $prj(\{b_1^k, \bar{d}_1^k, f_1^k\}, f_1^k, w_{\min}^k)$ and $prj(\{b_1^k, d_1^k, f_1^k\}, \bar{d}_1^k, w_{\min}^k)$. For $j \in [2, r^k - 2]$,
444 in addition to extending cost from b_j^k , we also extend from \bar{d}_{j-1}^k , which has just received
445 this amount of cost: $ext(b_j^k, \{b_j^k, d_{j-1}^k, d_j^k, f_j^k\}, w_{\min}^k)$ and $ext(\bar{d}_{j-1}^k, \{b_j^k, \bar{d}_{j-1}^k, d_j^k, f_j^k\}, w_{\min}^k)$,
446 which enable $prj(\{b_j^k, d_{j-1}^k, d_j^k, f_j^k\}, f_j^k, w_{\min}^k)$ and $prj(\{b_j^k, d_{j-1}^k, d_j^k, f_j^k\}, \bar{d}_j^k, w_{\min}^k)$. Finally,
447 after $j = r - 2$, we are left with w_{\min}^k in \bar{d}_{r-1}^k . Using $d_{r-1}^k \iff b_r^k$, we move cost from b_r^k to
448 d_{r-1}^k (specifically: $ext((b_r^k, \{b_r^k, d_r^k\}), w_{\min}^k)$, then $prj(\{b_r^k, d_r^k\}, d_r^k, w_{\min}^k)$). Since both d_r^k and
449 \bar{d}_r^k have cost w_{\min}^k , we can apply $prj_0(d_r^k, w_{\min}^k)$ to increase the lower bound by w_{\min}^k .

450 After these EPTs, not only is the lower bound increased by w_{\min}^k , but the variables
451 b_1^k, \dots, b_r^k have their cost decreased by w_{\min}^k , the variables f_1^k, \dots, f_{r-1}^k receive cost w_{\min}^k ,
452 and the variables d_1^k, \dots, d_{r-1}^k stay at 0. This matches the effects of PMRES, as required by
453 the inductive hypothesis. ◀

454 ▶ **Example 11.** We move away from our running example here, as showing and explaining
455 all the cost moves would be tedious and space consuming. Instead, we give a small example
456 with the core $\{b_1^1, b_2^1, b_3^1\}$ in figure 4. All variables of this core have uniform weight w . We
457 show how the EPTs remove cost from b_1^1, b_2^1, b_3^1 and move it to f_1^1, f_2^1 and c_\emptyset , leaving all
458 other cost functions unchanged, even though they were used to make the cost moves possible.
459 The increase in c_\emptyset comes from a nullary projection from b_3^1 .

460 Note that theorem 9 does not prove that the optimum of (P^i) is identical to that of
461 PMRES at iteration i , but only that it is at least as high, as the following example shows.

462 ▶ **Example 12 (Running example, continued).** After iteration 2, in the running example, unit
463 propagation alone detects the core $\{b_3, b_4, b_5\}$. This means that when we set these variables
464 to false because their weight is non-zero, unit propagation generates the empty clause.

465 Let \hat{P}^i be the reformulation of P^i given by theorem 9. Then H_R^i and \hat{P}^i have the same
466 costs/weights. $H_R^i \upharpoonright_0$ is constructed from H_R^i in the same way as $Bool(P^i)$ is constructed
467 from $Bool(P)$: by making each non zero cost (weight) into an infinite cost (weight). so $H_R^i \upharpoonright_0$
468 admits the same solutions as $Bool(\hat{P}^i)$. Moreover, each clause of $H_R^i \upharpoonright_0$ is contained in at
469 least one constraint of \hat{P}^i , therefore arc consistency on $Bool(\hat{P}^i)$ is at least as strong as unit
470 propagation on $H_R^i \upharpoonright_0$. And since the core $\{b_3, b_4, b_5\}$ is not satisfied, the arc consistency
471 closure of $Bool(\hat{P}^i)$ is empty, therefore its bound can be improved further.

472 On the other hand, there is no reason to expect that the the optimum of (P^i) will
473 necessarily be higher than the bound computed by PMRES. For example, if $H_R^i \upharpoonright_0$ has no
474 cores that can be detected by unit propagation, the argument of example 12 does not apply.

475 4 OLL

476 OLL [22] is probably the most relevant core-guided algorithm currently, since solvers based on
477 it, like RC2 [19] and CASHWMaxSAT-CorePlus [21] have done very well in recent MaxSAT

b_1^1	d_1^1	f_1^1	⓪	①	②	b_2^1	d_1^1	d_2^1	f_2^1	⓪	③	④
0	1	0	0			0	0	0	0	0	w	0
1	0	0	0	w	0	0	1	1	0	0		
1	1	1	0	w	0	1	1	0	0	0	w	0
						1	1	1	1	0	w	0

b_1^1	⓪	①	d_1^1	⓪	②	③	f_1^1	⓪	②	b_2^1	⓪	③
0	0		0	0	w	0	0	0		0	0	
1	w	0	1	0			1	0	w	1	w	0

b_3^1	⓪	④	⑤	f_2^1	⓪	④	c_\emptyset	⓪	⑤
0	0	w	0	0	0			0	w
1	w	w	0	1	0	w			

■ **Figure 4** The evolution of cost functions that leads to the increase of the lower bound by w for the core $\{b_1^1, b_2^1, b_3^1\}$. Each table shows a cost function and how it evolves after each EPT. We omit the rows which would violate one of the clauses introduced by PMRES, as infinity absorbs all costs, so they are unaffected by EPTs. The column ⓪ gives the initial costs. Subsequent columns give the state of each cost function after all EPTs to that point. Only points in the sequence which affect a given cost function are given in the corresponding table. Since $d_2^1 = b_3^1$ for this core, we simplify the problem here and replace occurrences of d_2^1 by b_3^1 rather than include an extra binary cost function to enforce their equality. The sequence is ①: $ext(b_1^1, \{b_1^1, d_1^1, f_1^1\}, w)$, ②: $prj(\{b_1^1, d_1^1, f_1^1\}, \bar{d}_1^1, w)$ and $prj(\{b_1^1, d_1^1, f_1^1\}, f_1^1, w)$, ③: $ext(b_2^1, \{b_1^1, d_1^1, b_3^1, f_1^1\}, w)$ and $ext(\bar{d}_1^1, \{b_1^1, d_1^1, b_3^1, f_1^1\}, w)$, ④: $prj(\{b_1^1, d_1^1, b_3^1, f_1^1\}, \bar{b}_3^1, w)$ and $prj(\{b_1^1, d_1^1, b_3^1, f_1^1\}, f_2^1, w)$, ⑤: $prj_0(b_3^1, w)$

478 evaluations [5].

4.1 MaxSAT with soft cardinality constraints

480 OLL is an iterative algorithm, similar to PMRES. For the purposes of this discussion,
 481 it only differs in how it processes each meta that it finds. At iteration i , given the meta
 482 $m^i = \{b_1^i, b_2^i, \dots, b_{r^i}^i\} \subseteq S^{i-1}$, it adds fresh variables $o_1^i, \dots, o_{r^i-1}^i$ and constraints $o_j \iff$
 483 $\sum_{k=1}^{r^i} b_k^i > j$, then decreases the weight of each variable in m^i by w_{\min}^i , increases the lower
 484 bound by w_{\min}^i , and sets the weight of the fresh variables $o_1^i, \dots, o_{r^i-1}^i$ to w_{\min}^i . The o
 485 variables are called *sum variables*.

4.1.1 OLL with implied cores

487 We use here a minor modification of OLL, which we denote OLL'. In this variant, before
 488 processing a meta at iteration i , each sum variable o_k^j , $j < i, k \in [2, r^j - 1]$ is replaced by $o_{k'}^j$,
 489 where $k' < k$ is the lowest index for which $w(o_{k'}^j) > 0$. This is sound because $o_k^j \rightarrow o_{k'}^j$ for all
 490 $k' < k$, which can be written as $\neg o_k^j \vee o_{k'}^j$. We can resolve the meta at iteration i with this
 491 clause to effectively replace o_k^j by $o_{k'}^j$. This procedure can be repeated as long as it results in
 492 a meta with non-zero minimum weight, although that step is not required for the results we
 493 obtain next.

494 We argue that OLL' matches the behaviour of a realistic implementation like RC2, when
 495 used with an assumption-based solver such as MINISAT [16] or a derivative like GLUCOSE [4].
 496 In order to extract a core with MINISAT, RC2 asserts the negation of all literals which may
 497 appear in a core as assumptions. These literals are passed to MINISAT as a sequence. MINISAT
 498 returns a subset of these literals as a core. Crucially, MINISAT immediately propagates each
 499 assumption in sequence and never returns in a core a literal which is implied by earlier
 500 assumptions. Therefore, if the literals of the soft clauses introduced by OLL are given in the

11:14 Analysis of Core-guided MaxSAT Using Linear Programming

501 order $\langle o_1^i, \dots, o_{r^i}^i \rangle$, we get from $o_{j+1}^i \implies o_j^i$, or equivalently $\bar{o}_j^i \implies \bar{o}_{j+1}^i$, that all literals \bar{o}_j^i
 502 are implied by unit propagation from $o_{j'}^i$, with $j' < j$. Therefore, MINISAT will not return a
 503 core that contains o_j^i if $o_{j'}^i$ is in the assumptions. This means that OLL' is identical to OLL
 504 given these implementation details. By inspection of the code of RC2, we can confirm that
 505 it does indeed use this order of assumptions with MINISAT, and therefore implements OLL'.

506 4.2 Cores and Hitting Sets of OLL

507 In the following, we overload notation that we have used already for PMRES, but we use
 508 them now in the context of OLL', with the same meaning: $H_R^i, \mathcal{C}^i, \mathcal{C}_\cup^i$.

509 ► **Lemma 13.** *There exists a set \mathcal{C}^i such that m^i is a core of H^i if and only if for each*
 510 *$c \in \mathcal{C}^i$, c is a core of ϕ .*

511 **Proof Sketch.** We observe that $o_j^i = \bigvee_{S \subseteq m^i, |S| > j} (\bigwedge_{b \in S} b)$, therefore it is a monotone function
 512 of the inputs of the core. The entire formula constructed by OLL is therefore also monotone.
 513 We show the result using a similar variable forgetting argument as we did in lemma 2. ◀

514 The proofs of Lemma 3, Observation 4, and Corollary 5 transfer to OLL' immediately.
 515 These establish that the WPMS instance $\langle H_R^i, cost^i \rangle$ encodes the minimum hitting set
 516 problem over \mathcal{C}_\cup^i , where the cores are derived as described in lemma 13 this time.

517 In order to show that OLL' does compute minimum hitting sets at each iteration for
 518 PMS, we have to prove the equivalent of lemma 6.

519 ► **Lemma 14.** *If W is a PMS instance, $H_R^i \upharpoonright_0$ is satisfiable for all iterations i of OLL'.*

520 **Proof Sketch.** The following invariant holds in OLL': for each meta m^i , there exists
 521 $0 \leq k < r^i$ such that $w(o_{k'}^i) = 0$ for all $k' \leq k$ and $w(o_{k'}^i) > 0$ for all $k' > k$. Therefore, any
 522 assignment that sets $o_{k'}^i$, $k' < k$, to true can be extended by setting $o_{k''}^i$ to true as well for all
 523 $k'' < k'$ and exactly k' variables of m^i , so that all sum constraints of iteration i are satisfied.

524 From there, we use the same argument as we did in the proof of lemma 6 to show that,
 525 given an assignment to the variables of the metas m^j, \dots, m^i , $j < i$, we can extend to an
 526 assignment to the variables of m^{j-1} because any two sum constraints from different iterations
 527 sum over disjoint sets of variables. ◀

528 As was the case for the corresponding lemma in PMRES, Lemma 14 says nothing about
 529 instances with non-uniform weights.

530 4.3 OLL and Linear Programming

531 We prove the equivalent of theorem 9 for OLL'.

532 ► **Theorem 15.** *There exists an integer linear program ILP_P^i which (1) is logically equivalent*
 533 *to the minimum hitting set problem with sets \mathcal{C}_\cup^i , (2) has size polynomial in $|H_R^i|$, and (3)*
 534 *whose linear relaxation has an optimum which matches that derived by OLL'.*

535 **Proof.** We construct a WCSP P^i . Its linear relaxation, the local polytope $LP(P^i)$, is the LP
 536 we want. Let $P^0 = \emptyset$. At iteration i , where the core discovered is $\{b_1^i, b_2^i, \dots, b_{r^i}^i\} \subseteq S^{i-1}$,
 537 P^i is defined as P^{i-1} and additionally the following variables and cost functions:

538 ■ 0/1 variables $b_1^i, \dots, b_n^i, o_1^i, \dots, o_{r^i-1}^i$, corresponding to the propositional variables of the
 539 same name in W^i .

- 540 ■ Unary cost functions with scope b_i for each $b_i \in \text{vars}(W^0)$, with $c(b_i, 0) = 0, c(b_i, 1) =$
541 $c^0(b_i)$
- 542 ■ A variable O^i with domain $[0, r^j]$, with $c(O^i, 0) = \infty$ and $c(O^i, j) = 0$ for all $j \in [1, r^i]$.
- 543 ■ A decomposition of the sum constraint $\sum_{j \in [1, r^i]} b_j^i = O^i$, as described by Allouche et al.
544 [1].
- 545 ■ Binary cost functions with scope $\{O^i, o_j^i\}$, for all $j \in [1, r^i - 1]$ where the tuples $\{j', 1\}$
546 and $\{j'', 0\}$, for all $1 \leq j' < j < j'' < r^i$, have infinite cost, and the rest have cost 0.
547 These encode the constraint $o_j^i \iff O^i > j$.

548 As before, the equivalence of P^i and H_R^i is immediate. We show that there exists a
549 reformulation of P^i that yields the same costs as the weights computed by OLL' , as well as
550 the same lower bound. The latter relies on previous results [1], which imply that, we can
551 move cost w_{\min}^i from b_1^i, \dots, b_n^i to O^i , so that we have $c(O^i, j) = jw_{\min}^i$. Since $c(O^i, 0) = \infty$,
552 we can apply $prj_0(O^i, w_{\min}^i)$. Finally, we can apply $ext(O^i = j', \{O^i, o_j^i\}, w_{\min}^i)$ for all $j' \geq j$,
553 followed by $prj(\{O^i, o_j^i, o_j^i, w_{\min}^i\})$. Once we complete this for all $j \in [1, r^i]$, there is no cost
554 in O^i , and each o_j^i has cost w_{\min}^i , as required. ◀

5 Connection to the Sherali-Adams hierarchy

556 The Sherali-Adams hierarchy of linear relaxations [28] of a 0/1 integer linear program is a
557 well known construction for building stronger relaxations. At its k^{th} level, it uses monomials
558 of degree k and it is known that the level n relaxation (where n is the number of variables
559 in the ILP) represents the convex hull of the original ILP, meaning that it solves the ILP
560 exactly. On the flip side, the size of the relaxations grows exponentially with the level of the
561 hierarchy, meaning that even low level SA relaxations tend to be impractical.

562 Formally, we derive the k^{th} level SA relaxation as follows. Let $SA_0^u(LP) = LP$, the
563 linear relaxation of the integer program. First, we define the set of multipliers $M_k =$
564 $\{\prod_{i \in P_1} x_i \prod_{i \in P_2} (1 - x_i) \mid P_1, P_2 \subseteq [1, n], |P_1 \cup P_2| = k, P_1 \cap P_2 = \emptyset\}$, i.e., the set of all
565 non-tautological monomials of degree k , using either x_i or $(1 - x_i)$ as factors. We then
566 multiply each constraint $c \in LP_0$ by each multiplier $m \in M_k$, simplify using $x^2 = 1$ and
567 $x(1 - x) = 0$, and finally we replace each higher order monomial by a single 0/1 variable to
568 get $SA_k^u(LP)$.

569 In this description, SA_k^u does not contain the variables and constraints of LP or any
570 $SA_j^u, j \in [1, k - 1]$. Here, we use instead $SA_k(LP) = \cup_{i=0}^k (SA_i^u(LP) \cup \text{cns}(k))$, where $\text{cns}(k)$
571 are constraints which ensure consistency between the variables at different levels, i.e., do not
572 allow $x_i x_j = 1$ and $x_i = 0$ at the same time.

573 To show the connection with PMRES, we define the *depth* measure for variables and,
574 by extension, cores and formulas. The set the depth of all variables appearing in W^0
575 to be 0, and we write $dp(b_j) = 0$, for $b_j \in \text{vars}(W^0)$. Consider a meta m^i . We define
576 $dp(f_j^i) = \max_{b_j \in m^i} dp(b_j) + 1$ for all $j \in [1, r^i - 1]$, and similarly for $d_j^i, j \in [1, r^i - 1]$. With
577 an overload of notation, we also write $dp(m^i) = dp(f_1^i)$. Finally, at iteration i , we write
578 $dp(W^i) = \max_{j \in [1, i]} dp(m^j)$. In words, the depth of a variable of the original instance has
579 depth 0, the variables introduced by a meta are one level deeper than variables that appear
580 in the meta, the depth of a meta is the same as that of the variables it introduces, and the
581 depth of the instance at iteration i is the deepest meta PMRES has discovered.

582 The result of this section, is that $LP(P^i)$, the linear relaxation that achieves the bound
583 computed by PMRES, is a subset of the $2^{dp(W^i)}$ level Sherali-Adams relaxation of a specific
584 linear formulation of the hitting set instance \mathcal{C}_{\cup}^i .

585 ► **Theorem 16.** *The variables f_j^i with $dp(f_j^i) = k$ are defined as a linear expression over*
 586 *variables of at most the level 2^k SA relaxation of the hitting set problem over \mathcal{C}_\cup^i .*

587 **Proof.** By induction. It holds for variables with depth 0, since they are variables of the
 588 original formula. Assume that it holds for variables of depth $k - 1$.

589 The main observation is that, since $f_j^i = b_j^i \wedge d_j^i$, we can write it as $f_j^i = b_j^i d_j^i$, i.e.,
 590 replace the conjunction by multiplication, which is valid for 0/1 variables. Then, since
 591 $d_j^i = b_{j+1}^i \vee \dots \vee b_{r^i}^i$, we can write it as $d_j^i = \max(b_{j+1}^i, \dots, b_{r^i}^i)$. The max operator is a
 592 piecewise linear function, so this expression is linear. Finally, we replace d_j^i in the definition
 593 of f_j^i to get $f_j^i = \max(b_j^i b_{j+1}^i, \dots, b_j^i b_{r^i}^i)$. Recall that $dp(b_l^i)$ for $l \in [j + 1, r^i]$ is at most 2^{k-1} ,
 594 so f_j^i can be written as a linear expression over monomials of degree at most 2^k , since it
 595 multiplies two variables which are themselves a linear expression over monomials of degree
 596 at most 2^{k-1} . ◀

597 Theorem 16 reflects the already known connection between Max-Resolution and the
 598 Sherali-Adams hierarchy in the context of proof systems for satisfiability [17]. Moreover, it is
 599 known that the k^{th} level of the Sherali Adams hierarchy based on the *basic LP relaxation*
 600 (BLP) of a CSP, another name for the local polytope LP, establishes k -consistency [29].

601 Theorem 16 is fairly weak. The upper bound is extremely loose and there is no lower
 602 bound. It is useful, however, as it suggests that discovering a meta of depth k involves
 603 potentially proving 2^k -inconsistency. It also hints towards minimizing the maximum degree
 604 of monomials entailed by a meta as a metric for choosing among different potential metas.

605 In the greater context of PMRES compared to IHS, one way to interpret the result of
 606 this section is that the two algorithms are instantiations of the same algorithm: they are
 607 both implicit hitting set algorithms, but where IHS extracts a single core at a time and
 608 offloads the hitting set computation to a specialized solver, PMRES shifts the burden to
 609 the SAT solver to not only extract cores, but discover a higher level relaxation so that the
 610 hitting set problem can be solved in polynomial time.

611 **6 Discussion**

612 **6.1 PM1**

613 The results of section 3.3 have of course already been shown for PM1 [7, 25]. The result we
 614 have shown here that is not shown for PM1 is the existence of a compact LP that computes
 615 the same bound as PM1. It is not easy to see how the results of section 3.4 could transfer.
 616 For PMRES and OLL, H_R^i logically entails all the implied cores. This allows us to create
 617 an ILP representation of the hitting set problem immediately, and then strengthen the LP
 618 relaxation using higher order cost functions to achieve the same bound. But for PM1, cores
 619 are solutions of a linear system, so it is not immediately obvious even how to create an ILP
 620 representation of the hitting set problem without enumerating the (potentially exponentially
 621 many) cores of the original formula.

622 **6.2 Practical Implications**

623 Besides revealing a tight connection between the operation of IHS and core-guided algorithms,
 624 there are potential practical implications, in particular from theorem 9. We first observe
 625 that the linear program used to prove theorem 9 is linear in the size of H_R^i , hence the size
 626 of the LP is not too great. Moreover, it can be further reduced by noting that, in order to
 627 replicate the bound of PMRES, the dual variable corresponding to several primal constraints

628 is always zero. Therefore, they can be removed from the LP without affecting the bound.
 629 After that, the LP can be further simplified by removing variables that appear in only 1
 630 constraint and forgetting (in the sense of the knowledge compilation operation of forgetting)
 631 variables that appear in only two constraints. In this way, the LP is reduced to contain only
 632 the d and f variables, and uses r^i constraints to relate them. In the running example, upon
 633 discovering the core $\{b_1, b_2, b_3, b_4\}$, the LP needs only the following constraints to satisfy the
 634 requirements of theorem 9:

$$635 \quad b_1^1 - f_1^1 - d_1^1 = 0$$

$$636 \quad b_2^1 - f_2^1 - d_2^1 + d_1^1 = 0$$

$$637 \quad b_3^1 - f_3^1 + b_4^1 + d_2^1 = 1$$

638 We omit the details of this mechanical reduction of the LP. But this suggests that the LP
 639 of theorem 9 is not just a theoretical construct, but a practical way to replicate the reasoning
 640 of PMRES. This allows a solver which runs PMRES until some heuristic condition is met,
 641 then passes its progress to IHS using theorem 9 to represent the hitting set problem and the
 642 lower bound. In the other direction, a solver can run IHS, then solve the hitting set problem
 643 once with PMRES to construct H_R^i , then continue solving starting from $\langle H_R^i \cup H, w^i \rangle$, in
 644 order to simplify solution of the ILP. However, running the two algorithms in sequence is the
 645 simplest form of combining them. Presumably, the greatest performance can be gained by
 646 an even deeper integration, using the LP to communicate progress.

647 **7 Conclusion**

648 We have narrowed the gap between implicit hitting set and core-guided algorithms for
 649 MaxSAT. We have shown that the core-guided algorithms PMRES and OLL, the latter
 650 of which is the basis for the winning solvers of some recent maxsat evaluations, implicitly
 651 compute a potentially exponentially large set of cores of the original MaxSAT formula at
 652 each iteration and a minimum hitting set of those cores under some conditions. Moreover,
 653 we showed that they build a WPMS instance which is logically equivalent to the minimum
 654 hitting set problem over those cores and can therefore be seen as a compressed, polynomial
 655 sized, encoding of that problem. In addition, we showed how this problem is solved: by
 656 generating a subset of a higher level of the Sherali-Adams linear relaxation of that hitting
 657 set problem. These results open up the possibility for tighter integration between PMRES
 658 and IHS.

659 **References**

-
- 660 **1** David Allouche, Christian Bessiere, Patrice Boizumault, Simon de Givry, Patricia Gutierrez,
 661 Jimmy H. M. Lee, Ka Lun Leung, Samir Loudni, Jean-Philippe Métivier, Thomas Schiex,
 662 and Yi Wu. Tractability-preserving transformations of global cost functions. *Artif. Intell.*,
 663 238:166–189, 2016. doi:10.1016/j.artint.2016.06.005.
 - 664 **2** Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Solving (weighted) partial maxsat
 665 through satisfiability testing. In *International conference on theory and applications of*
 666 *satisfiability testing*, pages 427–440. Springer, 2009.
 - 667 **3** Carlos Ansótegui and Joel Gabàs. WPM3: an (in)complete algorithm for weighted partial
 668 maxsat. *Artif. Intell.*, 250:37–57, 2017. doi:10.1016/j.artint.2017.05.003.
 - 669 **4** Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers.
 670 In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages
 671 399–404, 2009.

- 672 5 F. Bacchus, J. Berg, M. Järvisalo, R. Martins, and A. (eds) Niskanen. MaxSAT evaluation 2022:
673 Solver and benchmark descriptions. Technical Report vol. B-2022-2, Department of Computer
674 Science, University of Helsinki, Helsinki, 2022. URL: <http://hdl.handle.net/10138/318451>.
- 675 6 Fahiem Bacchus, Antti Hyttinen, Matti Järvisalo, and Paul Saikko. Reduced cost fixing in
676 maxsat. In J. Christopher Beck, editor, *Principles and Practice of Constraint Programming -*
677 *23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September*
678 *1, 2017, Proceedings*, volume 10416 of *Lecture Notes in Computer Science*, pages 641–651.
679 Springer, 2017. doi:10.1007/978-3-319-66158-2_41.
- 680 7 Fahiem Bacchus and Nina Narodytska. Cores in core based maxsat algorithms: An analysis.
681 In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing - SAT*
682 *2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014,*
683 *Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *Lecture Notes in Computer*
684 *Science*, pages 7–15. Springer, 2014. doi:10.1007/978-3-319-09284-3_2.
- 685 8 Jeremias Berg, Fahiem Bacchus, and Alex Poole. Abstract cores in implicit hitting set
686 maxsat solving. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of*
687 *Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-*
688 *10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 277–294.
689 Springer, 2020. doi:10.1007/978-3-030-51825-7_20.
- 690 9 Jeremias Berg, Paul Saikko, and Matti Järvisalo. Improving the effectiveness of sat-based
691 preprocessing for maxsat. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings*
692 *of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015,*
693 *Buenos Aires, Argentina, July 25-31, 2015*, pages 239–245. AAAI Press, 2015. URL: <http://ijcai.org/Abstract/15/040>.
- 694 10 Maria Luisa Bonet, Jordi Levy, and Felip Manyà. A complete calculus for max-sat. In Armin
695 Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT*
696 *2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*,
697 volume 4121 of *Lecture Notes in Computer Science*, pages 240–251. Springer, 2006. doi:
698 10.1007/11814948_24.
- 700 11 M. C. Cooper, S. de Givry, M. Sanchez, T. Schiex, M. Zytnicki, and T. Werner. Soft
701 arc consistency revisited. *Artificial Intelligence*, 174(7-8):449–478, May 2010. URL: <http://dx.doi.org/10.1016/j.artint.2010.02.001>,
702 doi:10.1016/j.artint.2010.02.001.
- 703 12 Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT
704 instances. In Jimmy Ho-Man Lee, editor, *Principles and Practice of Constraint Programming*
705 *- CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011.*
706 *Proceedings*, volume 6876 of *Lecture Notes in Computer Science*, pages 225–239. Springer,
707 2011. doi:10.1007/978-3-642-23786-7_19.
- 708 13 Jessica Davies and Fahiem Bacchus. Exploiting the power of mip solvers in maxsat. In
709 Matti Järvisalo and Allen Van Gelder, editors, *Theory and Applications of Satisfiability*
710 *Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013.*
711 *Proceedings*, volume 7962 of *Lecture Notes in Computer Science*, pages 166–181. Springer,
712 2013. doi:10.1007/978-3-642-39071-5_13.
- 713 14 Jessica Davies and Fahiem Bacchus. Postponing optimization to speed up MAXSAT solving.
714 In Christian Schulte, editor, *Principles and Practice of Constraint Programming - 19th*
715 *International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*,
716 volume 8124 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 2013. doi:
717 10.1007/978-3-642-40627-0_21.
- 718 15 Tomás Dlask and Tomás Werner. On relation between constraint propagation and block-
719 coordinate descent in linear programs. In Helmut Simonis, editor, *Principles and Practice*
720 *of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve,*
721 *Belgium, September 7-11, 2020, Proceedings*, volume 12333 of *Lecture Notes in Computer*
722 *Science*, pages 194–210. Springer, 2020. doi:10.1007/978-3-030-58475-7_12.

- 723 16 Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proceedings of Theory and*
724 *Applications of Satisfiability Testing (SAT)*, pages 502–518, 2003.
- 725 17 Yuval Filmus, Meena Mahajan, Gaurav Sood, and Marc Vinyals. Maxsat resolution and
726 subcube sums. *ACM Trans. Comput. Logic*, 24(1), jan 2023. doi:10.1145/3565363.
- 727 18 Zhaohui Fu and Sharad Malik. On solving the partial MAX-SAT problem. In Armin
728 Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT*
729 *2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*,
730 volume 4121 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2006. doi:
731 10.1007/11814948\25.
- 732 19 Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient maxsat solver.
733 *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019. doi:10.3233/SAT190116.
- 734 20 Javier Larrosa and Federico Heras. Resolution in Max-SAT and its relation to local consistency
735 in weighted CSPs. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference*
736 *on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 193–198,
737 2005.
- 738 21 Zhendong Lei, Yiyuan Wang, Shiwei Pan, Shaowei Cai, and Minghao Yin. CASHWMaxSAT-
739 CorePlus: Solver description. Technical report, Department of Computer Science, University
740 of Helsinki, Helsinki, 2022. URL: <http://hdl.handle.net/10138/318451>.
- 741 22 António Morgado, Carmine Dodaro, and João Marques-Silva. Core-guided maxsat with soft
742 cardinality constraints. In Barry O’Sullivan, editor, *Principles and Practice of Constraint*
743 *Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014.*
744 *Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer,
745 2014. doi:10.1007/978-3-319-10428-7\41.
- 746 23 António Morgado, Alexey Ignatiev, and João Marques-Silva. MSCG: robust core-guided maxsat
747 solving. *J. Satisf. Boolean Model. Comput.*, 9(1):129–134, 2014. doi:10.3233/sat190105.
- 748 24 Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided maxsat
749 resolution. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence,*
750 *July 27 -31, 2014, Québec City, Québec, Canada.*, pages 2717–2723, 2014. URL: <http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8513>.
- 751 25 Nina Narodytska and Nikolaj S. Bjørner. Analysis of core-guided maxsat using cores and
752 correction sets. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference*
753 *on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel,*
754 volume 236 of *LIPICs*, pages 26:1–26:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik,
755 2022. doi:10.4230/LIPICs.SAT.2022.26.
- 756 26 Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95,
757 1987. doi:10.1016/0004-3702(87)90062-2.
- 758 27 Paul Saikko, Jeremias Berg, and Matti Järvisalo. LMHS: A SAT-IP hybrid maxsat solver.
759 In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability*
760 *Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016,*
761 *Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 539–546. Springer,
762 2016. doi:10.1007/978-3-319-40970-2\34.
- 763 28 Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations between the continuous and
764 convex hull representations for zero-one programming problems. *SIAM Journal on Discrete*
765 *Mathematics*, 3(3):411–430, 1990. doi:10.1137/0403036.
- 766 29 Johan Thapper and Stanislav Zivný. Sherali-adams relaxations for valued cps. In Magnús M.
767 Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata,*
768 *Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan,*
769 *July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages
770 1058–1069. Springer, 2015. doi:10.1007/978-3-662-47672-7\86.
- 771