


1 Preprocessing in SAT-Based Multi-Objective 2 Combinatorial Optimization

3 **Christoph Jabs** ✉ 🏠 

4 HIIT, Department of Computer Science, University of Helsinki, Finland

5 **Jeremias Berg** ✉ 🏠 

6 HIIT, Department of Computer Science, University of Helsinki, Finland

7 **Hannes Ihalainen** ✉

8 HIIT, Department of Computer Science, University of Helsinki, Finland

9 **Matti Järvisalo** ✉ 🏠 

10 HIIT, Department of Computer Science, University of Helsinki, Finland

11 — Abstract —

12 Building on Boolean satisfiability (SAT) and maximum satisfiability (MaxSAT) solving algorithms,
13 several approaches to computing Pareto-optimal MaxSAT solutions under multiple objectives have
14 been recently proposed. However, preprocessing in (Max)SAT-based multi-objective optimization
15 remains so-far unexplored. Generalizing clause redundancy to the multi-objective setting, we
16 establish provably-correct liftings of MaxSAT preprocessing techniques for multi-objective MaxSAT
17 in terms of computing Pareto-optimal solutions. We also establish preservation of Pareto-MCSes—the
18 multi-objective lifting of minimal correction sets tightly connected to optimal MaxSAT solutions—
19 as a distinguishing feature between different redundancy notions in the multi-objective setting.
20 Furthermore, we provide a first empirical evaluation of the effect of preprocessing on instance sizes
21 and multi-objective MaxSAT solvers.

22 **2012 ACM Subject Classification** Mathematics of computing → Combinatorial optimization; Theory
23 of computation → Constraint and logic programming

24 **Keywords and phrases** maximum satisfiability, multi-objective combinatorial optimization, prepro-
25 cessing, redundancy

26 **Digital Object Identifier** 10.4230/LIPIcs.CP.2023.44

27 **Supplementary Material** *Software (Source Code)*: <https://bitbucket.org/coreo-group/mo-prepro>

28 **Funding** Work financially supported by Academy of Finland under grants 322869, 342145 and
29 356046.

30 **Acknowledgements** The authors wish to thank the Finnish Computing Competence Infrastructure
31 (FCCI) for supporting this project with computational and data storage resources.

32 **1** Introduction

33 Boolean satisfiability (SAT) solving [7] is arguably a noticeable success story of constraint pro-
34 gramming. The impact of SAT solvers goes beyond merely deciding satisfiability. Incremental
35 use of SAT solvers [13] today enables efficiently solving, e.g., hard optimization problems via
36 maximum satisfiability (MaxSAT) [1]. While MaxSAT allows for finding optimal solutions in
37 terms of a single objective function, practical applications have motivated various algorithmic
38 advances and non-trivial generalizations of MaxSAT solving techniques to optimization under
39 multiple objectives [41, 38, 10, 25, 20, 11]. These algorithms allow for computing one or
40 several of the so-called Pareto-optimal solutions of multi-objective MaxSAT instances, i.e.,
41 solutions in which no objective can be improved without negatively affecting the value of
42 another objective.



© Christoph Jabs, Jeremias Berg, Hannes Ihalainen, and Matti Järvisalo;
licensed under Creative Commons License CC-BY 4.0

29th International Conference on Principles and Practice of Constraint Programming (CP 2023).

Editor: Roland H. C. Yap; Article No. 44; pp. 44:1–44:19

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

43 Preprocessing has become a central part of the SAT solving pipeline [8], pruning the
 44 instance through applying complex combinations of different inference and simplification
 45 rules based on fundamental notions of (clause) redundancy. Motivated by its success in
 46 SAT, preprocessing in MaxSAT solving, through both extensions of SAT-based simplification
 47 techniques [3], and novel MaxSAT-specific techniques [5, 23, 37], is becoming increasingly
 48 popular and better understood, especially through recent work generalizing fundamental
 49 notions of redundancy in SAT [28, 27, 21, 22] to MaxSAT [24]. The MaxSAT liftings of
 50 redundancy notions allow for uniformly establishing the formal correctness of a wide range
 51 of MaxSAT preprocessing techniques [24, 4].

52 The advances in SAT and MaxSAT preprocessing, together with the recent advances in
 53 extending the reach of SAT-based approaches to multi-objective combinatorial optimization,
 54 call for studying fundamental and practical aspects of preprocessing in multi-objective settings.
 55 So-far preprocessing for (Max)SAT-based multi-objective optimization remains unexplored,
 56 with several open research questions. Developing correct liftings of MaxSAT preprocessing
 57 techniques to multi-objective settings, where Pareto-optimal solutions are sought for, calls for
 58 redundancy notions in order to uniformly capture the correctness of such liftings. In analogy
 59 to work analysing the power of different redundancy notions in SAT and more recently
 60 in MaxSAT, understanding the relationship between different redundancy notions in the
 61 multi-objective setting is also fundamentally relevant. From a more practical perspective,
 62 the effect of preprocessing for multi-objective problems in terms of simplifications achieved
 63 and solver runtimes has also not been thoroughly explored.

64 We make contributions to each of these questions. We provide redundancy notions for
 65 the multi-objective setting based on the notions of reconstructible and literal-reconstructible
 66 clauses, allowing for establishing the correctness of a large number of preprocessing techniques
 67 for multi-objective MaxSAT in terms of computing Pareto-optimal solutions. Additionally, we
 68 identify the preservation of Pareto-MCSEs (the multi-objective lifting of minimal correction
 69 sets tightly connected to Pareto-optimal solutions [40]) as a distinguishing feature between
 70 the two proposed redundancy notions. We also consider liftings of MaxSAT preprocessing
 71 techniques which alter in a controlled way the objective functions at hand and thereby
 72 cannot be directly captured by the clause redundancy notions. Putting these preprocessing
 73 techniques lifted to the multi-objective setting into practice, we provide a first preprocessor
 74 implementation for multi-objective MaxSAT, and perform a first empirical evaluation of the
 75 effect of preprocessing both in terms of instance size reductions achieved and runtimes of
 76 recently proposed approaches to multi-objective MaxSAT solving.

77 **2 Multi-Objective MaxSAT**

78 For a Boolean variable x there are two literals, x and $\neg x$. A clause C is a set (or disjunction)
 79 of literals and a (CNF) formula F a set (or conjunction) of clauses. A (truth) assignment τ
 80 assigns variables to truth values 0 (false) or 1 (true). Assignments are extended to literals l ,
 81 clauses C , and formulas F , in the standard way: $\tau(\neg l) = 1 - \tau(l)$, $\tau(C) = \max\{\tau(l) \mid l \in C\}$,
 82 and $\tau(F) = \min\{\tau(C) \mid C \in F\}$, defining semantics for CNF formulas. An assignment τ is
 83 a solution to a CNF formula F if $\tau(F) = 1$; τ is complete for F if τ assigns a value to all
 84 variables in F , and otherwise partial for F . With slight abuse of notation, an assignment τ
 85 can be viewed as the set of the literals it assigns to 1. Then $\tau(x) = 1$ ($\tau(x) = 0$) is shorthand
 86 for $x \in \tau$ ($\neg x \in \tau$), $\neg C$ for $\{\neg l \mid l \in C\}$, and $\tau \supset \neg C$ means that τ falsifies a clause C .

87 We focus on the following natural generalization of the maximum satisfiability (MaxSAT)
 88 problem to multi-objective combinatorial optimization [42, 17, 15]. An instance $\mathcal{I} = (F, O)$ of

multi-objective MaxSAT (MO-MaxSAT) consists of a CNF formula F , the clauses of which need to be satisfied by any solution to the instance, and a tuple $O = (O_1, \dots, O_p)$ of p linear objective functions with positive coefficients over literals (or equivalently, pseudo-Boolean expressions) under minimization. We denote the set of literals appearing in O_i by $\mathcal{B}_i(\mathcal{I})$ and the set of literals appearing in at least one of the objectives by $\mathcal{B}(\mathcal{I}) = \bigcup_{i=1}^p \mathcal{B}_i(\mathcal{I})$. Furthermore, we denote by $c_i(l)$ the coefficient of literal l in O_i . If l does not appear in O_i , then $c_i(l) = 0$.

The cost $O(\tau) = (O_1(\tau), \dots, O_p(\tau))$ of a solution τ to \mathcal{I} (i.e., a solution to F) is obtained by evaluating each objective under τ . If τ is not a solution of F , we let $O(\tau) = (\infty, \dots, \infty)$. As a central notion of optimality in the multi-objective setting in general, we focus on Pareto-optimality, which is based on the following domination relation between solutions.

Definition 1 ((Weak) Domination). *Consider two solutions τ_1 and τ_2 with costs $O(\tau_1) = (O_1(\tau_1), \dots, O_p(\tau_1))$ and $O(\tau_2) = (O_1(\tau_2), \dots, O_p(\tau_2))$. The solution τ_1 weakly dominates τ_2 (denoted $\tau_1 \preceq \tau_2$) if $O_i(\tau_1) \leq O_i(\tau_2)$ holds for all $i = 1, \dots, p$. If additionally $O_i(\tau_1) < O_i(\tau_2)$ for some i , then τ_1 dominates τ_2 (denoted $\tau_1 \prec \tau_2$).*

Intuitively, the solution τ_1 weakly dominates another solution τ_2 if it is not worse in any objective. We use $\tau_1 \not\preceq \tau_2$ to denote that τ_1 does not weakly dominate τ_2 . Note that domination is not a total order on solutions, i.e., $\tau_1 \not\preceq \tau_2$ does not generally imply $\tau_2 \prec \tau_1$ or $\tau_2 \preceq \tau_1$.

A partial assignment τ^p dominates another (partial) assignment δ^p if for every extension $\delta \supset \delta^p$ there is an extension $\tau \supset \tau^p$ that dominates δ . A solution τ to an MO-MaxSAT instance \mathcal{I} is Pareto-optimal¹ if τ is not dominated by any other solution to \mathcal{I} .

The notion of the non-dominated set of an MO-MaxSAT instance characterizes the solutions of interest in terms of their (non-dominated) costs.

Definition 2 (Non-dominated set). *The non-dominated set $\text{non-dominated}(\mathcal{I}) = \{O(\tau) \mid \tau \text{ is Pareto-optimal}\}$ of an MO-MaxSAT instance $\mathcal{I} = (F, O)$ consists of the costs of the Pareto-optimal solutions of \mathcal{I} .*

Practical algorithm for computing the non-dominated set of a given MO-MaxSAT instance also provide for each cost $o \in \text{non-dominated}(\mathcal{I})$ a Pareto-optimal solution having cost o . It is worth noting that for an $o \in \text{non-dominated}(\mathcal{I})$ there may be more than one Pareto-optimal solution with cost o and that for a single-objective MaxSAT instance \mathcal{I} the set $\text{non-dominated}(\mathcal{I})$ consists of the optimal (minimum) cost of \mathcal{I} .

3 Clause Redundancy in MO-MaxSAT

Preprocessing an MO-MaxSAT instance \mathcal{I} refers to the iterative application of a set of preprocessing techniques (inference/simplification rules) on \mathcal{I} , resulting in a preprocessed instance $\mathcal{P}(\mathcal{I})$ for which $\text{non-dominated}(\mathcal{I}) = \text{non-dominated}(\mathcal{P}(\mathcal{I}))$. In other words, correctness of preprocessing requires that the non-dominated set of the original \mathcal{I} does not change under the preprocessing techniques applied.

As fundamental notions for capturing, establishing the correctness of, and analysing the strengths of different MO-MaxSAT preprocessing techniques, we propose several (clause) redundancy properties in MO-MaxSAT. These properties can be viewed as multi-objective counterparts of earlier proposed redundancy notions in SAT [28, 27, 21, 22] and most recently

¹ Sometimes in the literature also referred to as *efficient* or *non-inferior* [14].

130 in MaxSAT [24], with similar motivations. In contrast to SAT (where clause redundancy
 131 notions are required to preserve satisfiability) and similarly as in MaxSAT, clause redundancy
 132 notions are required to preserve optimal costs. Compared to MaxSAT, however, the multiple
 133 objectives and Pareto optimization require additional care.

134 For an MO-MaxSAT instance $\mathcal{I} = (F, O)$ and a clause C , $\mathcal{I} \wedge C = (F \wedge C, O)$ is the
 135 instance obtained by adding C to \mathcal{I} . We begin with a general notion of redundancy for the
 136 problem of computing the non-dominated set in MO-MaxSAT.

137 **► Definition 3 (Redundant clauses).** *A clause C is redundant wrt an MO-MaxSAT instance*
 138 *\mathcal{I} if $\text{non-dominated}(\mathcal{I}) = \text{non-dominated}(\mathcal{I} \wedge C)$.*

139 Note that this definition does not require that all Pareto-optimal solutions should be preserved.

140 We propose two refined redundancy notions which turn out to differ in strength and thereby
 141 in terms of the preprocessing techniques they capture. The notions are based on the following
 142 alternative characterization of redundancy that essentially states that a clause C is redundant
 143 if every solution that falsifies it is weakly dominated by some solution that satisfies C .

144 **► Proposition 4.** *A clause C is redundant wrt an instance $\mathcal{I} = (F, O)$ iff, for any solution*
 145 *$\tau \supset \neg C$ to \mathcal{I} that falsifies C , there is a witnessing assignment (or simply witness) ω^τ for*
 146 *which $\omega^\tau(C) = 1$ and $\omega^\tau \preceq \tau$.*

147 **Proof.** We prove each of the directions separately.

148 *C is redundant \Rightarrow a witness exists:* Consider a solution $\tau \supset \neg C$ to \mathcal{I} . Then there
 149 exists a Pareto-optimal solution $\delta \preceq \tau$ (we can pick $\delta = \tau$ if τ is Pareto-optimal). Since
 150 $\text{non-dominated}(\mathcal{I}) = \text{non-dominated}(\mathcal{I} \wedge C)$ (as C is redundant), there is a solution ω^δ to
 151 $\mathcal{I} \wedge C$ with $O(\tau) = O(\omega^\delta)$. Such ω^δ satisfies C and weakly dominates δ . Thus, it also weakly
 152 dominates τ , fulfilling the requirements of the proposition.

153 *A witness exists $\Rightarrow C$ is redundant:* To show that C is redundant according to Defini-
 154 tion 3 we show that $\text{non-dominated}(\mathcal{I} \wedge C) = \text{non-dominated}(\mathcal{I})$. For the direction
 155 $\text{non-dominated}(\mathcal{I} \wedge C) \subset \text{non-dominated}(\mathcal{I})$, note that every Pareto-optimal solution τ
 156 to $\mathcal{I} \wedge C$ is also a solution to \mathcal{I} . Furthermore, τ is also Pareto-optimal wrt \mathcal{I} . If this was
 157 not the case, by the assumption the solution δ dominating τ wrt \mathcal{I} would have a witness
 158 ω^δ dominating τ wrt $\mathcal{I} \wedge C$. Since therefore every Pareto-optimal solution to $\mathcal{I} \wedge C$ is also
 159 Pareto-optimal wrt \mathcal{I} , it follows that $\text{non-dominated}(\mathcal{I} \wedge C) \subset \text{non-dominated}(\mathcal{I})$.

160 For the other direction consider an element $o \in \text{non-dominated}(\mathcal{I})$ and let τ be a Pareto-
 161 optimal solution to \mathcal{I} for which $O(\tau) = o$. For the interesting case, assume $\tau \supset \neg C$, i.e.,
 162 that it falsifies C . Then by the assumption τ is weakly dominated by some witness ω^τ
 163 that satisfies C . Now $O(\tau) = O(\omega^\tau) = o$ (as otherwise τ would not be Pareto-optimal)
 164 demonstrating that $o \in \text{non-dominated}(\mathcal{I} \wedge C)$ and thus that C is redundant. ◀

165 The (weakly) dominating witness ω^τ guaranteed by Proposition 4 for any redundant
 166 clause C might differ depending on the specific solution τ that falsifies C . The redundancy
 167 notions of *reconstructible* and *literal-reconstructible* clauses we propose next are based on
 168 placing stronger requirements on this witness.

169 **► Definition 5 (Reconstructible clauses).** *A clause C is reconstructible on the (partial)*
 170 *assignment ω wrt an MO-MaxSAT instance \mathcal{I} if (i) $\omega(C) = 1$, and (ii) $(\tau \setminus \neg\omega) \cup \omega \preceq \tau$ for*
 171 *every solution $\tau \supset \neg C$ to \mathcal{I} .*

172 In words, a clause C is reconstructible wrt an MO-MaxSAT instance \mathcal{I} if there is a *single*
 173 *witnessing assignment ω* that satisfies C and weakly dominates *all* solutions τ that do not.

174 Moreover, enforcing the partial assignment ω in any such solution τ allows for efficiently
 175 obtaining a solution to \mathcal{I} that satisfies C and weakly dominates τ . For the corner case, note
 176 that if there are no solutions to \mathcal{I} that falsify C , then C is reconstructible on any witness.

177 The fact that reconstructible clauses are redundant follows directly from Proposition 4.
 178 The next example demonstrates that the converse does not hold. Central to the example is
 179 to note that a direct consequence of Definition 5 is that if C is reconstructible on the partial
 180 assignment ω , then ω weakly dominates the partial assignment $\neg C$.

181 ► **Example 6.** Let $\mathcal{I} = (F, (O_1, O_2))$ be an MO-MaxSAT instance with $F = (a_1 \vee a_2) \wedge$
 182 $(b_1 \vee b_2) \wedge (a_1 \vee b_1) \wedge (a_1 \vee b_2) \wedge (a_2 \vee b_1) \wedge (a_2 \vee b_2)$, $O_1 = a_1 + a_2$, and $O_2 = b_1 +$
 183 b_2 . Then $\text{non-dominated}(\mathcal{I}) = \{(1, 2), (2, 1)\}$, and the Pareto-optimal solutions are $\tau_1 =$
 184 $\{a_1, \neg a_2, b_1, b_2\}$, $\tau_2 = \{\neg a_1, a_2, b_1, b_2\}$, $\tau_3 = \{a_1, a_2, \neg b_1, b_2\}$, and $\tau_4 = \{a_1, a_2, b_1, \neg b_2\}$.
 185 Consider the clause $C = (\neg a_2 \vee \neg b_2)$. Since τ_1 and τ_4 are solutions to $F \wedge C$, adding C
 186 does not change the non-dominated set of the instance. Thus, C is redundant wrt \mathcal{I} . To
 187 see that C is not reconstructible we show that no partial assignment ω that satisfies C
 188 weakly dominates $\neg C$. There are two possible candidates for such ω (as C contains two
 189 literals): $\omega_1 = \{\neg a_2\}$ and $\omega_2 = \{\neg b_2\}$. The only solution of \mathcal{I} that ω_1 can be extended to is
 190 τ_1 . However, $\neg C$ can be extended to τ_3 , which is not weakly dominated by τ_1 . Similarly,
 191 $\omega_2 = \{\neg b_2\}$ does not weakly dominate $\tau_2 \supset \neg C$, showing that $\omega_2 \not\leq \neg C$.

192 Contrasting Example 6, the next proposition shows that the notions of (clause) redundancy
 193 according to Definition 3 and reconstructible clauses according to Definition 5 coincide for
 194 single-objective MaxSAT instances that have solutions.

195 ► **Proposition 7.** *For a single-objective MaxSAT instance $\mathcal{I} = (F, (O_1))$ with at least one*
 196 *solution τ and clause C , it holds that C is reconstructible for \mathcal{I} iff C is redundant.*

197 **Proof (sketch).** For the non-trivial direction, assume that C is redundant. Then there is an
 198 optimal (minimum-cost) solution τ° to \mathcal{I} that satisfies C . As \mathcal{I} only has a single objective,
 199 τ° weakly dominates all solutions to \mathcal{I} . Therefore, C is reconstructible on τ° . ◀

200 As a further notion of redundancy, we consider *literal-reconstructible* clauses as a special
 201 case of reconstructible clauses where the witness is required to consist of a single literal. In
 202 Section 4 we discuss properties that literal-reconstructible clauses specifically satisfy and
 203 overview in Section 5.1 preprocessing techniques that can be characterized by adding and
 204 removing literal-reconstructible clauses.

205 ► **Definition 8 (Literal-reconstructible clauses).** *A clause C is literal-reconstructible wrt an*
 206 *instance $\mathcal{I} = (F, O)$ if either (i) all solutions to F satisfy C , or (ii) there is a non-objective*
 207 *literal $l \in C \setminus \mathcal{B}(\mathcal{I})$ s.t. if $\tau \supset \neg C$ is a solution to F , then $\tau_l = (\tau \setminus \{\neg l\}) \cup \{l\}$ is a solution*
 208 *to $F \wedge C$. If condition (ii) holds, we say that C is literal-reconstructible on the literal l .*

209 Note that the definition of literal-reconstructible clauses does not explicitly require that
 210 τ_l weakly dominates τ , as this follows from l not being an objective literal. The following
 211 proposition states that literal-reconstructible clauses are redundant in terms of Definition 3.

212 ► **Proposition 9.** *If a clause C is literal-reconstructible wrt an MO-MaxSAT instance*
 213 *$\mathcal{I} = (F, (O_1, \dots, O_p))$, then $\text{non-dominated}(\mathcal{I}) = \text{non-dominated}(\mathcal{I} \wedge C)$.*

214 **Proof (sketch).** For the interesting case, assume that there is a solution $\tau \supset \neg C$ to \mathcal{I} that
 215 does not satisfy C . Let $l \in C \setminus \mathcal{B}(\mathcal{I})$ be the literal on which C is literal-reconstructible and
 216 consider the solution $\tau_l = (\tau \setminus \{\neg l\}) \cup \{l\}$. Then by the assumption τ_l is a solution to $\mathcal{I} \wedge C$
 217 and as $l \notin \mathcal{B}(\mathcal{I})$ we have that $O_i(\tau_l) \leq O_i(\tau)$ for all objectives, i.e., for each $i = 1, \dots, p$. As
 218 τ_l is a solution to both \mathcal{I} and $\mathcal{I} \wedge C$, the result follows. ◀

219 A clause that is literal-reconstructible on l is also reconstructible on the witness $\omega = \{l\}$.
 220 The following example shows that the opposite does not hold in general, i.e., there are
 221 reconstructible clauses that are not literal-reconstructible.

222 ► **Example 10.** Consider the MO-MaxSAT instance $\mathcal{I} = (F, (O_1))$ with $F = (a_1 \vee a_2)$
 223 and $O_1 = a_1 + a_2$. The clause $C = (\neg a_1)$ is reconstructible on the witness $\omega = \{\neg a_1, a_2\}$.
 224 The assignment $\tau = \{a_1, \neg a_2\}$ is a solution to F but does not satisfy C . The only literal
 225 $l \in C \setminus \mathcal{B}(\mathcal{I})$ is $\neg a_1$, but $(\{a_1, \neg a_2\} \setminus \{a_1\}) \cup \{\neg a_1\} = \{\neg a_1, \neg a_2\}$ is not a solution to $F \wedge C$.
 226 It follows that C is not literal-reconstructible.

227 The relative generality of these three MO-MaxSAT redundancy notion can be summarized
 228 as follows. For any MO-MaxSAT instance \mathcal{I} , the set of redundant clauses $\text{Red}(\mathcal{I})$ is a superset
 229 of the set of reconstructible clauses $\text{Rec}(\mathcal{I})$, and $\text{Rec}(\mathcal{I})$ is a superset of the set of literal-
 230 reconstructible clauses $\text{LRec}(\mathcal{I})$. Furthermore, there are clauses that are reconstructible but
 231 not literal-reconstructible (i.e., there is an instance \mathcal{I} for which $\text{Rec}(\mathcal{I}) \supsetneq \text{LRec}(\mathcal{I})$), and
 232 clauses that are redundant (in terms of Definition 3) that are not reconstructible (i.e., there
 233 is an instance \mathcal{I}' for which $\text{Red}(\mathcal{I}') \supsetneq \text{Rec}(\mathcal{I}')$). In contrast to single-objective MaxSAT, the
 234 last statement holds also for instances that have solutions as for a single objective instance
 235 \mathcal{I}'' we have that $\text{Red}(\mathcal{I}'') \neq \text{Rec}(\mathcal{I}'')$ if and only if \mathcal{I}'' does not have solutions.

236 As a side-remark, literal-reconstructible clauses are related to so-called cost literal propaga-
 237 tion redundant clauses [24] recently proposed for (single-objective) MaxSAT: any cost literal
 238 propagation redundant clause is literal-reconstructible under a single objective. The opposite
 239 holds only when conditions (i) and (ii) in Definition 8 can be deterministically checked
 240 by standard Boolean constraint propagation on clauses (i.e., unit propagation). Intuitiv-
 241 ely, literal-reconstructible clauses extend and slightly generalize the concept of cost literal
 242 propagation redundant clauses for the multi-objective setting.

243 4 Redundancy and Pareto-MCSes

244 We move on to analysing the effect that adding (literal-)reconstructible clauses to an MO-
 245 MaxSAT instance has on the solution space in terms of so-called Pareto minimal correction
 246 sets (Pareto-MCSes) [40, 41], that—informally speaking—correspond to subset-minimal sets
 247 of objective literals that are assigned to 1 by at least one Pareto-optimal solution.

248 ► **Definition 11 (Pareto-MCS).** Consider an MO-MaxSAT instance $\mathcal{I} = (F, O)$. A subset
 249 $M \subset \mathcal{B}(\mathcal{I})$ of objective literals is a correction set if there is a solution τ of \mathcal{I} that assigns
 250 $\tau(l) = 0$ for every objective literal l not appearing in M . M is a minimal correction set
 251 (MCS) (or multi minimal correction subset as in [40]) if no $M' \subsetneq M$ is a correction set.
 252 Finally, M is a Pareto-MCS if each solution τ that assign $\tau(l) = 0$ for every $l \in \mathcal{B}(\mathcal{I}) \setminus M$ is
 253 Pareto-optimal. The set $\text{ParetoMCS}(\mathcal{I})$ consists of the Pareto-MCSes of \mathcal{I} .

254 For some intuition, note that assigning an objective literal to 1 can be seen as falsifying
 255 a soft constraint. If M is an MCS or Pareto-MCS, then for any solution with $\tau(l) = 0$ for
 256 every literal not in M we also have $\tau(l') = 1$ for every literal l' in M . From this point of
 257 view, these definitions of MCSes align with the (arguably more classical) ones in terms of
 258 subset-minimal sets of soft constraints falsified by some solution. Specifically, if \mathcal{I} only has a
 259 single objective, this definition is identical to MCSes in single-objective MaxSAT [34].

260 The relationship between Pareto-optimal solutions, Pareto-MCSes and elements of the
 261 non-dominated set is not one-to-one. For a Pareto-MCS M of an MO-MaxSAT instance
 262 $\mathcal{I} = (F, O)$, there is at least one corresponding Pareto-optimal solution τ to \mathcal{I} . The cost of

each such τ wrt each objective in O is the sum of the objective coefficients of the objective literals included in M . There can be multiple Pareto-optimal solutions that correspond to a Pareto-MCS which differ in how non-objective variables are assigned. Furthermore, for a single element (cost tuple) in the non-dominated set, there can be multiple corresponding Pareto-MCSES, since two different Pareto-MCSES can incur the same cost wrt each objective of \mathcal{I} . Hence, preserving the Pareto-MCSES of an input MO-MaxSAT instance \mathcal{I} is a sufficient *but not necessary* condition for preserving $\text{non-dominated}(\mathcal{I})$. For computing the non-dominated set, it suffices that at least one corresponding Pareto-MCS for each element in the non-dominated set is preserved.

We establish the fact that preservation of the set of Pareto-MCSES is a property of literal-reconstructible clauses, distinguishing this notion from the more general notion of reconstructible clauses which does not have this property. More precisely, the following summarizes the main theorem of this section: adding/removing literal-reconstructible clauses does not change the set of Pareto-MCSES.

► **Theorem 12.** *Assume that C is literal-reconstructible wrt an MO-MaxSAT instance $\mathcal{I} = (F, O)$. Then $\text{ParetoMCS}(\mathcal{I}) = \text{ParetoMCS}(\mathcal{I} \wedge C)$.*

A proof of Theorem 12 relies on showing that, given any Pareto-optimal solution $\tau \supset \neg C$ of \mathcal{I} that does not satisfy C , the weakly-dominating (Pareto-optimal) witness τ_l obtained by flipping the value of the literal $l \in C \setminus \mathcal{B}(\mathcal{I})$ that C is literal-reconstructible on corresponds to the exact same Pareto-MCS as τ . Toward formalizing this intuition, we show that if the negation of l is in any objective, then there is no Pareto-optimal solution that falsifies C .

► **Lemma 13.** *Let C be literal-reconstructible on l wrt $\mathcal{I} = (F, O)$ and $\neg l$ an objective literal, i.e., $\neg l \in \mathcal{B}(\mathcal{I})$. Then there is no Pareto-optimal solution $\tau \supset \neg C$ to \mathcal{I} that falsifies C .*

Proof of Lemma 13. As C is literal-reconstructible on l , $\tau' = (\tau \setminus \{\neg l\}) \cup \{l\}$ is a solution to \mathcal{I} . Because $\neg l \in \mathcal{B}(\mathcal{I})$ and therefore at least one of the objectives evaluates to less for τ' than for τ , $\tau' \prec \tau$. Therefore, τ is not Pareto-optimal. ◀

With the inverse of Lemma 13 covering the (special) case of some Pareto-optimal solutions falsifying C , we turn to the proof of Theorem 12.

Proof of Theorem 12. If C is literal-reconstructible because every solution of \mathcal{I} satisfies C , the solutions and therefore the set of Pareto-MCSES of \mathcal{I} and $\mathcal{I} \wedge C$ are the same. Otherwise, let C be literal-reconstructible on l and consider the following.

$\text{ParetoMCS}(\mathcal{I}) \subset \text{ParetoMCS}(\mathcal{I} \wedge C)$: Let $M \in \text{ParetoMCS}(\mathcal{I})$ and consider the Pareto-optimal solution $\tau^M \supset \{\neg b \mid b \in \mathcal{B}(\mathcal{I}) \setminus M\}$ to \mathcal{I} that sets $\tau(b) = 0$ for every objective literal b not in M . Since C is literal-reconstructible on l , there is a solution δ to $F \wedge C$ that weakly dominates τ^M . If τ^M satisfies C , then $\delta = \tau^M$. Otherwise, $\delta = (\tau^M \setminus \{\neg l\}) \cup \{l\}$, and since τ^M is Pareto-optimal and falsifies C , by Lemma 13 $\neg l$ is not an objective literal. In both cases δ corresponds to the same MCS (M) as τ^M . Furthermore, M must be a Pareto-MCS of $\mathcal{I} \wedge C$ as any solution dominating δ would also be a solution to \mathcal{I} and therefore M would not be a Pareto-MCS of \mathcal{I} .

$\text{ParetoMCS}(\mathcal{I} \wedge C) \subset \text{ParetoMCS}(\mathcal{I})$: Given $M \in \text{ParetoMCS}(\mathcal{I} \wedge C)$ and a Pareto-optimal $\tau^M \supset \{\neg b \mid b \in \mathcal{B}(\mathcal{I}) \setminus M\}$ to $\mathcal{I} \wedge C$, τ^M is also Pareto-optimal for \mathcal{I} as any dominating solution could be reconstructed (by flipping the value of l) into a solution to $\mathcal{I} \wedge C$ dominating τ^M . ◀

Contrasting Theorem 12, we show that a similar result cannot be obtained for reconstructible clauses.

307 ► **Proposition 14.** *There is an MO-MaxSAT instance \mathcal{I} and a reconstructible clause C wrt*
 308 *\mathcal{I} for which $\text{ParetoMCS}(\mathcal{I} \wedge C) \subsetneq \text{ParetoMCS}(\mathcal{I})$.*

309 **Proof.** Consider the MO-MaxSAT instance $\mathcal{I} = (F, (O_1, O_2))$ with $F = (a_1 \vee b_1 \vee b_2)$,
 310 $O_1 = a_1$, $O_2 = b_1 + b_2$, and $C = (\neg b_2)$. We have that $\text{ParetoMCS}(\mathcal{I}) = \{\{a_1\}, \{b_1\}, \{b_2\}\}$
 311 and $\text{ParetoMCS}(\mathcal{I} \wedge C) = \{\{a_1\}, \{b_1\}\}$.

312 Since the only clause in F and C are both satisfied by $\omega = \{b_1, \neg b_2\}$, every superset of ω
 313 is a solution to $F \wedge C$. Furthermore, given a solution τ to F that falsifies C , the solution
 314 $\tau_\omega = (\tau \setminus \neg\omega) \cup \omega$ has $O_1(\tau_\omega) = O_1(\tau)$ and $O_2(\tau_\omega) \leq O_2(\tau)$, hence $\tau_\omega \preceq \tau$. It follows that C
 315 is reconstructible on ω wrt \mathcal{I} . ◀

316 This distinction between literal-reconstructible and reconstructible clauses in terms of
 317 the preservation of Pareto-MCSes provides two important insights.

318 Firstly, the fact that adding/removing literal-reconstructible clauses does not change
 319 the set of Pareto-MCSes implies that (single-objective) MaxSAT preprocessing techniques
 320 that can be viewed as sequences of adding and removing literal-reconstructible clauses are
 321 techniques that are “directly applicable” to MO-MaxSAT. In particular, it has been shown
 322 that the non-dominated set of an MO-MaxSAT instance $\mathcal{I} = (F, O)$ can be computed by
 323 enumerating its Pareto-MCSes, which can in turn be achieved by enumerating the MCSes of
 324 the (single-objective) MaxSAT instance (F, O^m) with the single objective $O^m = \sum_{O_i \in O} O_i$
 325 that sums all objectives of \mathcal{I} [41]. Thus, any preprocessing technique for single-objective
 326 MaxSAT that preserves MCSes is directly applicable to MO-MaxSAT by applying it to
 327 $(F, (O^m))$ and using the preprocessed formula in the MO-MaxSAT instance. The correctness
 328 of such techniques—which we will overview shortly—can either be directly argued on the MO-
 329 MaxSAT level by viewing them as sequences of adding and removing literal-reconstructible
 330 clauses, *or* by using (MCS-preserving) redundancy notions such as cost literal propagation
 331 redundancy on the level of single-objective MaxSAT. On the other hand, preprocessing
 332 techniques captured by reconstructible clauses but which cannot be captured by literal-
 333 reconstructible clauses—as detailed later on—go beyond preserving Pareto-MCSes, having
 334 the ability to eliminate Pareto-MCSes that are redundant in terms of the non-dominating
 335 set. Hence, reconstructible clauses are key in capturing the correctness of such techniques in
 336 a uniform way.

337 5 Preprocessing for MO-MaxSAT

338 We proceed with overviewing a range of preprocessing techniques for MO-MaxSAT, lifting
 339 earlier-proposed techniques from single-objective MaxSAT (some of which originate from
 340 SAT) to the multi-objective setting. We detail in short which of the techniques are captured
 341 by the notions of reconstructible or literal-reconstructible clauses by simulating the techniques
 342 via sequences of additions and removals of redundant clauses of a specific type.

343 5.1 Preprocessing Techniques Captured by Literal-Reconstructible 344 Clauses

345 First, we shortly recall well-known single-objective MaxSAT preprocessing techniques that are
 346 known to preserve MCSes [24]. We note again that the correctness of these techniques follows
 347 from the previously mentioned fact that each of them preserve MCSes in single-objective
 348 MaxSAT, which further follows naturally from earlier work on capturing these techniques
 349 in the setting of SAT solving via redundancy notions developed for SAT. Alternatively—as
 350 we will detail in the following—the correctness arguments can be directly made on the

351 MO-MaxSAT level by showing that each technique can be simulated via removing and adding
 352 literal-reconstructible clauses. For the following list of techniques, let $\mathcal{I} = (F, O)$ be an
 353 MO-MaxSAT instance with $O = (O_1, \dots, O_p)$.

354 **Bounded Variable Elimination (BVE)** [39, 12] as arguably the most important SAT
 355 preprocessing technique allows eliminating a non-objective variable $x \notin \mathcal{B}(\mathcal{I})$ (and $\neg x \notin \mathcal{B}(\mathcal{I})$)
 356 from \mathcal{I} . A step of BVE on \mathcal{I} and x results in the MO-MaxSAT instance $\text{bve}(\mathcal{I}, x) =$
 357 $(F \cup F_{\text{res}} \setminus (F_x \cup F_{\neg x}), O)$, where $F_x = \{C \in F \mid x \in C\}$, $F_{\neg x} = \{C \in F \mid \neg x \in C\}$ are the
 358 sets of clauses containing x and $\neg x$, respectively, and $F_{\text{res}} = \{(A \vee B) \mid (A \vee x), (B \vee \neg x) \in F\}$
 359 is the set of all non-tautological resolvents on x of the clauses in F , bounded in practice
 360 to eliminate variables when this decreases the number of clauses. Working directly on the
 361 MO-MaxSAT level, $\text{bve}(\mathcal{I}, x)$ can be obtained from $\mathcal{I} = (F, O)$ by a sequence of additions
 362 and removals of literal-reconstructible clauses as follows. First add F_{res} to F which does not
 363 change the non-dominated set because every clause in F_{res} is satisfied by any solution to F
 364 and therefore literal-reconstructible wrt \mathcal{I} and any instance obtained by adding clauses from
 365 F_{res} to \mathcal{I} . Note that for every $(A \vee B) \in F_{\text{res}}$, by construction of F_{res} , $(A \vee x), (B \vee \neg x) \in F$.
 366 Second, remove the clauses $F_x \cup F_{\neg x}$ from the intermediate instance $\mathcal{I}' = (F \cup F_{\text{res}}, O)$: every
 367 clause in F_x (resp. $F_{\neg x}$) is literal-reconstructible on x (resp. $\neg x$) wrt \mathcal{I}' and any instance
 368 obtained by removing clauses in $F_x \cup F_{\neg x}$ from \mathcal{I}' .

369 **Blocked Clause Elimination (BCE)** removes blocked clauses: a clause $(C \vee l) \in F$ is blocked
 370 on a literal $l \notin \mathcal{B}(\mathcal{I})$ if for every clause $(D \vee \neg l) \in F$ containing $\neg l$, the resolvent $(C \vee D)$ is
 371 a tautology. Note that a clause blocked on l is literal-reconstructible on l .

372 **Subsumption Elimination (SE).** A clause $C \in F$ is subsumed by another clause $D \in$
 373 F if $D \subset C$. One step of SE removes a subsumed clause C , resulting in the instance
 374 $\text{se}(\mathcal{I}, C) = (F \setminus \{C\}, O)$. Note that any solution to $\text{se}(\mathcal{I}, C)$ also satisfies C , and thus C is
 375 literal-reconstructible wrt $\text{se}(\mathcal{I}, C)$.

376 **Unit Propagation (UP).** Given a non-objective literal $l \notin \mathcal{B}(\mathcal{I})$ and a unit clause $(l) \in F$,
 377 unit propagation removes each clause $C \in F$ containing l ($l \in C$) and removes the negation
 378 $\neg l$ from the remaining clauses. Similarly as in SAT, UP can be viewed as an application of
 379 BVE on l (to remove negation $\neg l$ from all clauses) followed by an application SE (to remove
 380 resolvents introduced by BVE).

381 **Self-Subsuming Resolution (SSR).** Given two clauses $(x \vee A), (\neg x \vee B) \in F$ s.t. $A \subset B$,
 382 $x \notin \mathcal{B}(\mathcal{I})$, and $\neg x \notin \mathcal{B}(\mathcal{I})$, a step of SSR [36, 12] results in the formula $\text{ssr}(\mathcal{I}, (\neg x \vee B)) =$
 383 $((F \cup \{B\}) \setminus \{(\neg x \vee B)\}, O)$. Note that B is literal-reconstructible wrt \mathcal{I} and that $(\neg x \vee B)$
 384 is subsumed in $F \wedge B$.

385 **Failed Literal Elimination (FLE) and TrimMaxSAT.** FLE [44, 18, 32] and TrimMaxSAT [37]
 386 allow for detecting unit clauses entailed by F , i.e., clauses satisfied by every solution to \mathcal{I} .
 387 Such clauses are by definition literal-reconstructible.

388 **Equivalent Literal Substitution (ELS).** [33, 9, 43] Two literals l_1, l_2 are equivalent if
 389 $\tau(l_1) = \tau(l_2)$ for every solution τ . If neither literal nor their negation occur in objectives,
 390 equivalent literal substitution replaces every occurrence of l_2 with l_1 and $\neg l_2$ with $\neg l_1$.
 391 Viewed in terms of literal-reconstructible clauses, first add the clauses in which l_2 ($\neg l_2$) has

392 been replaced and then remove clauses containing l_2 ($\neg l_2$). Both of these sets of clauses
 393 are literal-reconstructible wrt the instance they are added to / removed from because
 394 $\tau(l_1) = \tau(l_2)$.

395 5.2 Preprocessing Techniques Captured by Reconstructible Clauses

396 We now turn to techniques that do not preserve all Pareto-MCSES and therefore require a
 397 more general notion of redundancy. Specifically, we lift (group-)subsumed label elimination
 398 ((G)SLE) [5, 30] from MaxSAT, extending subsumption to objective literals, to MO-MaxSAT
 399 and show that it is captured by adding *reconstructible* clauses.

400 ► **Definition 15.** *An objective literal $l \in \mathcal{B}(\mathcal{I})$ of an MO-MaxSAT instance $\mathcal{I} = (F, (O_1,$
 401 $\dots, O_p))$ is subsumed if there is a group of objective literals $S \subset \mathcal{B}(\mathcal{I})$ for which (i) $c_i(l) \geq$
 402 $c_i(\neg l) + \sum_{s \in S} c_i(s)$ for all objectives $(i = 1, \dots, p)$, (ii) every clause $C \in F$ that contains l
 403 also contains some literal $s \in S$, and (iii) every clause $C \in F$ that contains the negation of
 404 any $s \in S$ also contains $\neg l$.*

405 Informally speaking, a step of GSLE on an MO-MaxSAT instance \mathcal{I} wrt a subsumed literal l
 406 fixes $l = 0$. More formally, it results in the instance $\text{gsle}(\mathcal{I}, l) = \mathcal{I} \wedge (\neg l)$.

407 In contrast to the preprocessing techniques discussed in the preceding subsection, GSLE
 408 cannot be lifted from single-objective MaxSAT to MO-MaxSAT by simply combining multiple
 409 objectives into a sum. To see this, consider the MO-MaxSAT instance from the proof
 410 of Proposition 14. When applying *single-objective* GSLE by summing the objectives as
 411 $O_1 + O_2 = a_1 + b_1 + b_2$, a_1 is subsumed by $\{b_1\}$. However, adding the clause $(\neg a_1)$ removes
 412 $(1, 0)$ from the non-dominated set.

413 The following example demonstrates that GSLE can remove Pareto-MCSES.

414 ► **Example 16.** Consider the MO-MaxSAT instance in the proof of Proposition 14. According
 415 to Definition 15 b_2 is subsumed by $\{b_1\}$, hence $\text{gsle}(\mathcal{I}, b_2) = (F \wedge (\neg b_2), (O_1, O_2))$, and thus
 416 $\text{ParetoMCS}(\text{gsle}(\mathcal{I}, b_2)) \subsetneq \text{ParetoMCS}(\mathcal{I})$

417 For an alternative proof of the fact that GSLE cannot be viewed as a sequence of adding/re-
 418 moving literal-reconstructible clauses, consider Example 10 where it was argued that the
 419 clause $(\neg a_1)$ is reconstructible but not literal-reconstructible. Note that in the example a_1 is
 420 subsumed by $\{a_2\}$, and hence applying GSLE on the instance wrt a_1 would result in adding
 421 exactly the clause $(\neg a_1)$ into the instance.

422 The correctness of GSLE for MO-MaxSAT follows by observing that it can be viewed as
 423 the addition of a reconstructible clause.

424 ► **Proposition 17.** *If l is subsumed in an MO-MaxSAT instance $\mathcal{I} = (F, O)$, then the clause
 425 $C = (\neg l)$ is reconstructible wrt \mathcal{I} .*

426 **Proof.** Let $S \subset \mathcal{B}(\mathcal{I})$ be the group of literals that subsumes l , and $\tau \supset \neg C$ with $\tau(F) = 1$ a
 427 solution that falsifies C . Consider the witness $\omega = S \cup \{\neg l\}$ and the solution $\tau_\omega = (\tau \setminus \neg \omega) \cup \omega$.
 428 Since all clauses that l appears in contain at least one literal $s \in S$ (condition (ii) of
 429 Definition 15) and all clauses that a negated literal from S appears in also contain $\neg l$
 430 (condition (iii) of Definition 15), τ_ω is a solution to $F \wedge C$. Because l (note that $\tau(l) = 1$)
 431 increases every objective more than all of ω (condition (i) of Definition 15), τ_ω weakly
 432 dominates τ . ◀

433 5.3 Preprocessing with Changes to Objectives

434 All techniques we have so far considered solely change the formula of an instance and not
 435 the objectives. However, towards practical preprocessing for MO-MaxSAT, we note that
 436 MaxSAT preprocessing techniques that may change the single objective in MaxSAT can also
 437 be lifted to MO-MaxSAT. These include unit propagation and equivalent literal substitution
 438 on objective literals, intrinsic at-most-ones, and binary core removal (BCR). Alike their
 439 MaxSAT counterparts, these liftings cannot be expressed directly as a sequence of additions
 440 and removals of redundant clauses due to the very fact that redundant clauses by definition
 441 do not change costs of instances.

442 **Unit Propagation on an Objective Literal** $l \in \mathcal{B}(\mathcal{I})$, in addition to removing all clauses
 443 containing l and removing $\neg l$ from all clauses, replaces the terms $c_i(l) \cdot l$ with the respective
 444 constant $c_i(l)$ in each objective O_i for $i = 1, \dots, p$. It is straightforward to see that by doing
 445 so costs of solutions are left unchanged.

446 **Equivalent Literal Substitution on Objective Literals** replaces a literal l_2 with another
 447 literal l_1 if they are equivalent—regardless if the literals or their negations appear in an
 448 objective. Specifically, every occurrence of l_2 (resp. $\neg l_2$) is replaced by l_1 (resp. $\neg l_1$) and in
 449 every objective O_i ($i = 1, \dots, p$) l_1 (resp. $\neg l_1$) gets the coefficient $c_i(l_1) + c_i(l_2)$ (resp. $c_i(\neg l_1) +$
 450 $c_i(\neg l_2)$). In this way, the costs of solutions to the preprocessed instance are left unchanged.

451 **Intrinsic At-Most-One Technique** [23, 24] lifted to MO-MaxSAT works as follows. Given
 452 a set L of objective literals at most one of which are falsified in each solution to \mathcal{I} (i.e., at
 453 least $|L| - 1$ of the literals in L will incur cost in all solutions; such an L is sought heuristically
 454 using unit propagation), (i) a new literal l_L is introduced, (ii) the clause $(l_L \vee \bigvee_{l \in L} \neg l)$ is
 455 added to F , and (iii) for every objective O_i ($i = 1 \dots, p$) the coefficients of all literals in L
 456 are reduced by the minimum of these coefficients $c_i^m = \min\{c_i(l) \mid l \in L\}$ and the terms
 457 $c_i^m \cdot l_L + (|L| - 1) \cdot c_i^m$ are added. For intuition, the preliminary condition implies that for
 458 any solution to \mathcal{I} , at least $|L| - 1$ of the literals in L will incur cost at least c_i^m wrt each of
 459 the objectives. The added clause (ii) enforces that l_L must be true when all literals in L are
 460 true, incurring additional cost c_i^m wrt each of the objectives.

461 **Binary Core Removal (BCE)** [19, 30] can be phrased as first applying a restriction of
 462 the intrinsic at-most-one technique and then applying BVE. In detail, assume that a set
 463 $L = \{l_1, l_2\}$ with $c_i(l_1) = c_i(l_2)$ for all objectives satisfies the condition required for the
 464 intrinsic-at-most-ones and that $\neg l_1$ and $\neg l_2$ do not appear in F . Then BCR can be viewed
 465 as an application of intrinsic at-most-ones on L followed by applying BVE to eliminate l_1
 466 and l_2 (in practice when the size of the instance does not increase). Thus, its correctness for
 467 MO-MaxSAT follows directly from the correctness of intrinsic at-most-ones and BVE.

468 6 Experiments

469 Complementing our theoretical observations on redundancy notions for MO-MaxSAT, we
 470 detail results from an empirical evaluation of the combined effect of the various MO-MaxSAT
 471 preprocessing techniques overviewed in the preceding section in terms of their ability to
 472 reduce the size of real-world MO-MaxSAT instances and effect on runtime behaviour of
 473 recent MO-MaxSAT solvers. To the best of our understanding, this is the first evaluation

474 on the effect of preprocessing on MO-MaxSAT instances and the runtime performance of
475 MO-MaxSAT solvers.

476 We extended the MaxSAT preprocessor MaxPre 2 [30, 24] to MaxPre 2.1, covering MO-
477 MaxSAT. The preprocessor implementation, empirical data, and benchmarks are available
478 via <https://bitbucket.org/coreo-group/mo-prepro>. As the technique specification for
479 MaxPre 2.1 in the experiments we used `[[uvsrgc]VRTG]`, including unit propagation, BVE,
480 SE, SSR, GSLE, BCR, TrimMaxSAT, FLE, ELS, and intrinsic at-most-ones.²

481 We consider four MO-MaxSAT solvers: LEXIMAXIST³ [10], BIOPTSAT⁴ [25], CLM⁵ [11],
482 and SCUTTLE, our own implementation of a SAT-based approach based on enumerating so-
483 called *P*-minimal models [38, 31]. BIOPTSAT, CLM, and SCUTTLE compute a Pareto-optimal
484 solution for each element in the non-dominating set. BIOPTSAT is specific for MO-MaxSAT
485 under two objectives (i.e., bi-objective MaxSAT), while SCUTTLE and CLM handle any number
486 of objectives. LEXIMAXIST restricts to the simpler task of computing a leximax-optimal
487 solution, which corresponds to computing a specific element in the non-dominated set [16].
488 As BIOPTSAT and LEXIMAXIST offer different configurations, for BIOPTSAT we consider the
489 three central variants of a linear SAT-UNSAT based algorithm (denoted LSU), a core-guided
490 (MSU3) algorithm (denoted CG), and a hybrid between the two that was found to perform
491 best in [25] (denoted Hybrid). For LEXIMAXIST, we consider both a linear SAT-UNSAT and
492 a core-guided version of the approach. For CLM we evaluate both the core-guided (denoted
493 CG) and the implicit hitting set algorithm (denoted IHS). To achieve a tight integration, we
494 included MaxPre 2.1 as a library into the source code of each solver. The modified source code
495 of each solver is also available through <https://bitbucket.org/coreo-group/mo-prepro>.

496 We used three real-world benchmarks from the literature: package upgradeability
497 (PackUP) [26], learning interpretable decision rules (LIDR) [35], and development assurance
498 level (DAL) [6]. For PackUP we used the set of 3692 instances from [10], obtained from
499 Mancoosi International Solver Competition (<https://www.mancoosi.org/misc-2011/>) in-
500 stances using all combinations of 2–5 of the 5 original objectives. The 366 LIDR benchmark
501 instances with two objectives originate from [25], encoding the classification task for public
502 benchmark datasets. The 96 DAL benchmark instances originate from the LION9 challenge
503 (<https://www.cristal.univ-lille.fr/LION9/challenge.html>), each with 7 objectives.
504 The pseudo-boolean constraints in the DAL instances were encoded with a (generalized)
505 totalizer encoding [2, 29].

506 All runtime experiments were executed on 2.60-GHz Intel Xeon E5-2670 machines with
507 64-GB RAM in RHEL under a 1.5-hour per-instance time and 16-GB memory limit. Times
508 reported include the runtimes of MaxPre 2.1 whenever preprocessing is used.

509 6.1 Effect of Preprocessing on Instance Characteristics

510 We first consider the effect of preprocessing on four central characteristics of MO-MaxSAT
511 instances: the number of variables, the number of clauses, the sum of objective coefficients,
512 and the number of Pareto-MCSes.

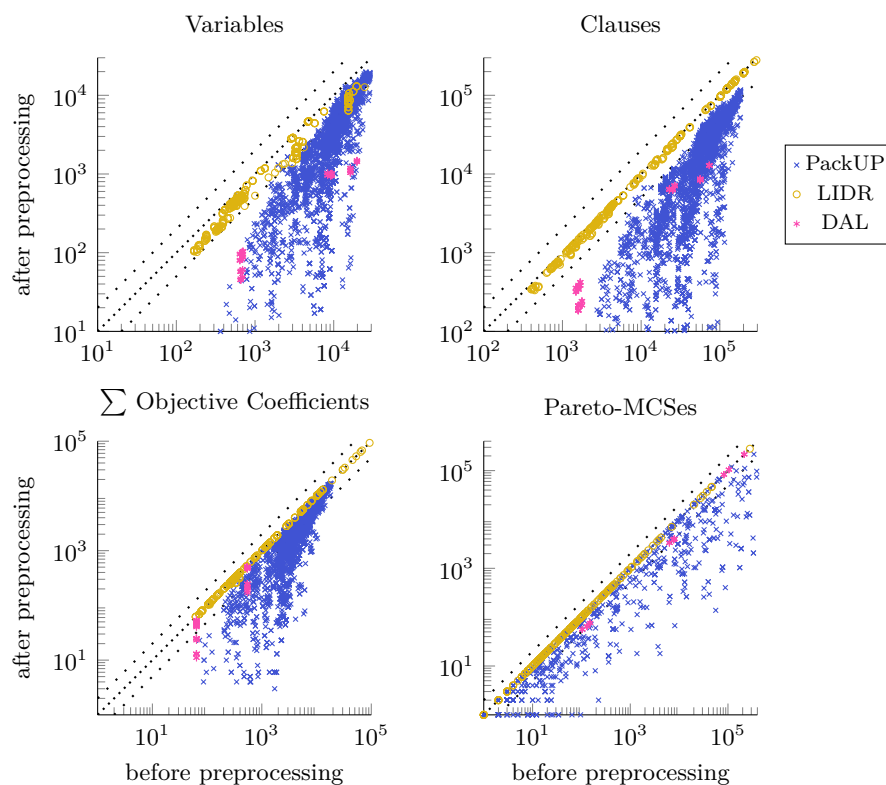
513 Figure 1 shows the reduction of variables (top left), number of clauses (top right), the
514 sum of objective coefficients (bottom left), and the number of Pareto-MCSes (bottom right)
515 obtained with MaxPre 2.1 for each of the three benchmark domains. The number of variables

² We excluded BCE as using it in preliminary testing led to slightly worse runtimes overall.

³ LEXIMAXIST obtained from <https://github.com/miguelcabral/leximaxIST>.

⁴ BIOPTSAT obtained from <https://bitbucket.org/coreo-group/bioptsat>

⁵ CLM obtained from <https://gitlab.inesc-id.pt/u001810/moco>



■ **Figure 1** Per-instance instance size reductions achieved by preprocessing.

516 (Figure 1 top left) is reduced significantly on each benchmark domain. In terms of medians,
 517 after preprocessing 9.5% of the original variables remain for DAL, 32% for PackUP, and
 518 64% for LIDR instances. In terms of clauses (Figure 1 top right), the reductions are very
 519 significant for both DAL and PackUP, with 9.3% and 24% of the original number of clauses
 520 remaining after preprocessing in terms of the median, respectively. For LIDR the number of
 521 clauses is reduced less significantly, although a reduction can still be observed; 93% of the
 522 original number of clauses remain.

523 Given an instance $\mathcal{I} = (F, (O_1, \dots, O_p))$, we measure the sum of objective coefficients, i.e.,
 524 $\sum_{i=1}^p \sum_{l \in \mathcal{B}_i(\mathcal{I})} c_i(l)$. Note that preprocessing can change the sum of objective coefficients
 525 both by inferring that some objective literals can be set to 0—conceptually decreasing the
 526 trivial upper bound on the objective—and by inferring that some literal must be assigned to
 527 1—conceptually increasing the lower bound. Figure 1 (bottom left) shows the reduction in
 528 the sum of objective coefficients achieved by preprocessing on each benchmark instance. The
 529 magnitude of reductions achieved by preprocessing depend significantly on the benchmark
 530 domain. For LIDR, preprocessing only seldom reduces objective coefficients. For PackUP a
 531 significant reduction is observed; the median sum of objective coefficients after preprocessing
 532 is 57% of the original. Furthermore, on 297 of the PackUP instances preprocessing reduced
 533 at least one of the objectives to *zero*, removing it from the instance. For DAL, while for
 534 some instances the objective coefficients are reduced only slightly, on every single instance
 535 preprocessing reduced at least one of the objectives to zero. The median sum of objective
 536 coefficients after preprocessing is 54% of the original for DAL.

537 For investigating how preprocessing affects the number of Pareto-MCSes, we used SCUTTLE

■ **Table 1** Solved instances (#), uniquely solved instances (uniq.), and cumulative runtimes over solved ($\sum t$) in 10^3 seconds, with and without preprocessing (Prepro.).

Solver	Prepro.	PackUP			LIDR			DAL		
		#	uniq.	$\sum t$	#	uniq.	$\sum t$	#	uniq.	$\sum t$
BiOPTSAT (LSU) (bi opt.)	no	1134	0	61.4	220	1	52.4	–	–	–
	yes	1161	27	47.7	223	4	34.1	–	–	–
BiOPTSAT (CG) (bi opt.)	no	1154	1	40.9	222	1	43.9	–	–	–
	yes	1159	6	34.5	222	1	38.4	–	–	–
BiOPTSAT (Hybrid) (bi opt.)	no	1154	1	46.6	222	0	40.4	–	–	–
	yes	1159	6	33.0	222	0	35.5	–	–	–
SCUTTLE (multi opt.)	no	1772	40	284.5	219	1	51.3	66	0	5.9
	yes	1778	46	244.1	218	0	44.1	67	1	5.3
CLM (CG) (multi opt.)	no	1593	88	301.3	206	2	48.0	60	7	8.1
	yes	1588	83	315.8	206	2	49.4	53	0	12.8
CLM (IHS) (multi opt.)	no	1301	91	258.5	134	19	26.8	48	7	0.3
	yes	1282	72	189.8	115	0	23.5	41	0	0.2
LEXIMAXIST (LSU) (leximax opt.)	no	2276	2	434.6	224	0	28.4	72	0	4.3
	yes	2347	73	268.5	224	0	29.6	72	0	5.2
LEXIMAXIST (CG) (leximax opt.)	no	2450	13	140.7	220	2	43.7	72	1	12.9
	yes	2453	16	140.0	218	0	38.0	73	2	9.7

538 to enumerate Pareto-MCSES of each benchmark instance under a 1.5-h per-instance time
539 limit. On the LIDR instances, no reduction in the number of Pareto-MCSES was observed.
540 For PackUP and DAL, respectively, preprocessing reduced the number of Pareto-MCSES
541 significantly, by more than one third for 33% and 60% of the instances, respectively. Further-
542 more, considering the per-instance reduction shown in Figure 1 (bottom right), we observed
543 that for PackUP the number of Pareto-MCSES is often reduced significantly further.

544 6.2 Effect of Preprocessing on Solver Runtimes

545 We now turn to investigating the effect of preprocessing on the runtime performance of
546 MO-MaxSAT solvers.

547 Table 1 shows the number of solved instances, number of instances uniquely solved with
548 or without preprocessing, and cumulative runtimes over solved instances (in 10^3 seconds) for
549 each solver. We emphasize that here one should focus on comparing the effect of preprocessing
550 on each individual solver and configuration. Most importantly, the numbers reported for
551 the four different solvers—BiOPTSAT, SCUTTLE, CLM, and LEXIMAXIST—are not directly
552 comparable to each other as they solve different variants of MO-MaxSAT: LEXIMAXIST
553 computes a solution corresponding to a single element in the non-dominated set, while
554 BiOPTSAT, SCUTTLE, and CLM enumerate the whole non-dominated set. Furthermore,
555 since BiOPTSAT supports two objectives only, data for BiOPTSAT on PackUP is restricted to
556 the 1420 instances with two objectives, and data on DAL is unavailable as the DAL instances
557 involve more than two objectives.

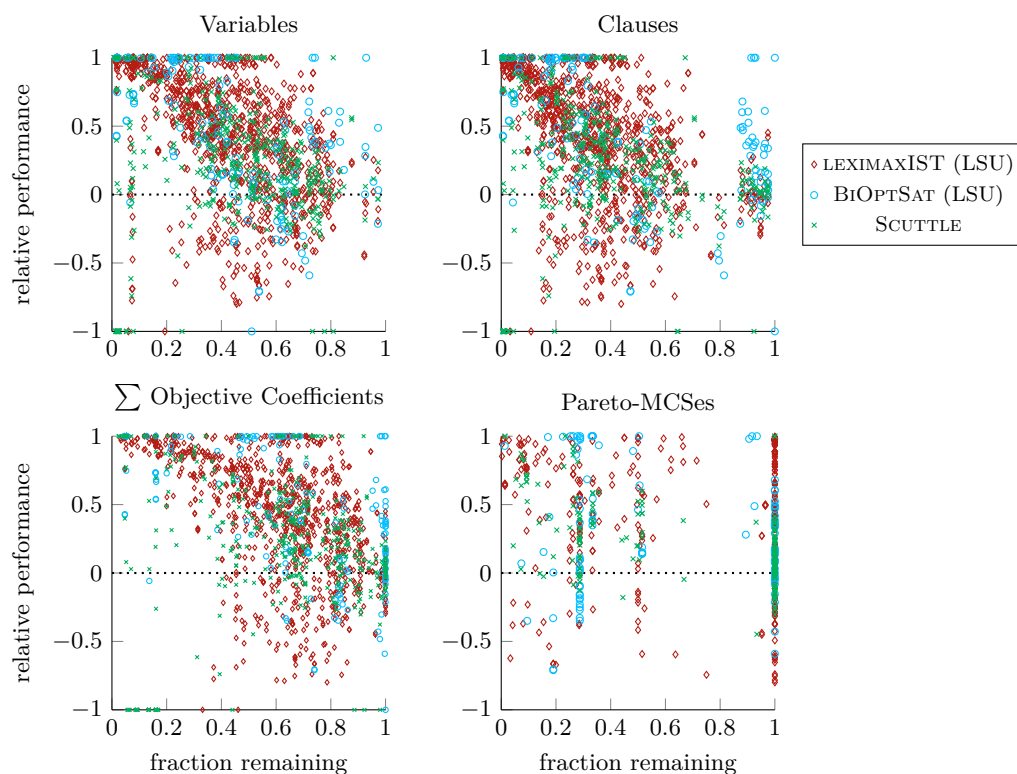
558 For PackUP, preprocessing has a clear positive impact on both the number of instances
559 solved and the runtimes of all solvers except for CLM: the solvers use less cumulative

560 runtime after preprocessing for solving more instances than what can be solved without
 561 preprocessing. We observe that for each of the three variants of BIOPTSAT as well as the LSU
 562 variant of LEXIMAXIST, preprocessing strictly increases the number of PackUP instances
 563 solved. There are close to no uniquely solved instances when preprocessing is not employed.
 564 Interestingly, the runtime improvement obtained using preprocessing for the LSU variants of
 565 BIOPTSAT, which without preprocessing is outperformed by the other BIOPTSAT variants, is
 566 so significant on PackUP instances that the LSU variant ends up even slightly outperforming
 567 the other variants. For SCUTTLE and the CG variant of LEXIMAXIST the number of uniquely
 568 solved instances with preprocessing is also higher than without, although there is a more
 569 significant number of instances that are uniquely solved without preprocessing. For LIDR,
 570 preprocessing speeds up all three configurations of BIOPTSAT and also increases the number
 571 of instances solved for the LSU variant. However, preprocessing does not consistently improve
 572 the performance of SCUTTLE, CLM, and LEXIMAXIST on LIDR and DAL.

573 Overall, although somewhat modestly, preprocessing appears to have the most signific-
 574 ant positive impact on linear SAT-UNSAT type algorithms, namely, the LSU variants of
 575 BIOPTSAT and LEXIMAXIST. This finding is in fact in-line with [24] where, in the con-
 576 text of MaxSAT, the strongest positive impact of preprocessing was observed for a linear
 577 SAT-UNSAT (solution-improving) MaxSAT solver.

578 Finally, we investigate potential correlations between the impact of preprocessing on
 579 solver runtimes and the instance characteristics of number of clauses, number of variables,
 580 sum of objective coefficients, and number of Pareto-MCSes. As a metric for the impact
 581 of preprocessing on solver runtimes, we use *relative solver performance*, defined for a fixed
 582 instance and solver as $(t_{\text{no prepro}} - t_{\text{prepro}}) / (t_{\text{no prepro}} + t_{\text{prepro}})$, where $t_{\text{(no) prepro}}$ is the solving
 583 time with (without) preprocessing. This metric takes values from -1 to 1 . A positive value
 584 implies that runtime with preprocessing was shorter than without preprocessing, and the value
 585 1 (value -1) means that the solver was able to solve the instance with (without) preprocessing,
 586 but timed out without (with) it; the closer to 1 (-1), the more significant a positive (negative)
 587 effect preprocessing has on overall runtime. As a metric for the impact of preprocessing
 588 on instance characteristics, we use *fraction remaining*. For a specific instance and instance
 589 characteristic, let $f_{\text{(no) prepro}}$ be the value of the feature with (without) preprocessing. The
 590 fraction remaining is then $f_{\text{prepro}} / f_{\text{no prepro}}$, taking values from 0 to 1 . For some intuition, the
 591 closer to 0 the value is, the more significantly preprocessing affects the instance characteristic:
 592 e.g., the value 0 for the number of clauses means that preprocessing removes all clauses from
 593 an instance, and a value of 0.5 (1) means that the preprocessed instance contains half as
 594 many (exactly as many) clauses as the original instance.

595 Figure 2 relates relative solver performance and the fractions remaining for the four
 596 instance characteristics for each solver using the configuration the runtimes of which were
 597 improved the most by preprocessing: BIOPTSAT (LSU), SCUTTLE, and LEXIMAXIST (LSU),
 598 focusing on “non-trivial” instances with runtimes > 60 seconds either with or without
 599 preprocessing. We observe that a lower fraction of variables remaining (Figure 2 top left),
 600 clauses (top right), or objective coefficient sum (bottom left) by preprocessing often also
 601 somewhat tends to result in faster solver runtimes (i.e., a higher relative performance of the
 602 solver), especially for LEXIMAXIST. Interestingly, the data for the LSU variant of BIOPTSAT
 603 as well as for SCUTTLE are quite scattered, with no clear correlations observed between
 604 relative solver performance and changes in instance characteristics. Finally, we note that the
 605 number of Pareto-MCSes appears to have little to no impact on the relative performance
 606 of these specific solvers. One possible explanation for this observation is that none of these
 607 specific solvers explicitly enumerate Pareto-MCSes in their search. On the other hand, based



■ **Figure 2** Relating solver runtimes with instance characteristic.

608 on the data, reducing the sum of the objective coefficients by preprocessing may be beneficial
 609 for solver performance; also in light of this developing further techniques that are capable of
 610 reducing the objective ranges appears to be an interesting direction for further work.

611 **7** Conclusions

612 Motivated by recent advances in (Max)SAT-based approaches to multi-objective optimization,
 613 we proposed redundancy notions and liftings of MaxSAT preprocessing techniques for
 614 the multi-objective setting. We showed that the redundancy notions capture different
 615 preprocessing techniques, with the (in)ability to remove Pareto-MCSes as the underlying
 616 differentiating property. We provided a stand-alone preprocessor implementation of the
 617 preprocessing techniques, and empirically evaluated the impact of preprocessing in multi-
 618 objective MaxSAT. The preprocessor can significantly reduce the size of real-world multi-
 619 objective MaxSAT instances and also has in cases a positive effect on runtimes of current
 620 state-of-the-art multi-objective MaxSAT solvers. Interesting directions for future work include
 621 developing redundancy notions that can capture changes to objectives; more fine-grained
 622 analysis of preprocessing for the restricted case of leximax optimization; and empirical
 623 evaluation of preprocessing on further problem settings with varying instance properties such
 624 as different distributions of objective coefficients.

625 — References —

- 626 1 Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maximum satisfiability. In Armin
627 Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability—*
628 *Second Edition*, volume 336 of *FAIA*, chapter 24, pages 929–991. IOS Press, 2021. doi:
629 10.3233/FAIA201008.
- 630 2 Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality con-
631 straints. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming—*
632 *9th International Conference, CP*, volume 2833 of *LNCS*, pages 108–122. Springer, 2003.
633 doi:10.1007/978-3-540-45193-8_8.
- 634 3 Anton Belov, António Morgado, and João Marques-Silva. SAT-based preprocessing for
635 MaxSAT. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Logic for*
636 *Programming, Artificial Intelligence, and Reasoning—19th International Conference, LPAR*,
637 volume 8312 of *LNCS*, pages 96–111. Springer, 2013. doi:10.1007/978-3-642-45221-5_7.
- 638 4 Jeremias Berg and Matti Järvisalo. Unifying reasoning and core-guided search for maximum
639 satisfiability. In Francesco Calimeri, Nicola Leone, and Marco Manna, editors, *Logics in*
640 *Artificial Intelligence—16th European Conference, JELIA*, volume 11468 of *LNCS*, pages
641 287–303. Springer, 2019. doi:10.1007/978-3-030-19570-0_19.
- 642 5 Jeremias Berg, Paul Saikko, and Matti Järvisalo. Subsumed label elimination for maximum
643 satisfiability. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia
644 Dignum, Frank Dignum, and Frank van Harmelen, editors, *22nd European Conference on*
645 *Artificial Intelligence, ECAI 2016*, volume 285 of *FAIA*, pages 630–638. IOS Press, 2016.
646 doi:10.3233/978-1-61499-672-9-630.
- 647 6 Pierre Bieber, Remi Delmas, and Christel Seguin. DALculus—Theory and tool for development
648 assurance level allocation. In Francesco Flammini, Sandro Bologna, and Valeria Vittorini, edi-
649 tors, *Computer Safety, Reliability, and Security—30th International Conference, SAFECOMP*,
650 volume 6894 of *LNCS*, pages 43–56. Springer, 2011. doi:10.1007/978-3-642-24270-0_4.
- 651 7 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of*
652 *Satisfiability—Second Edition*, volume 336 of *FAIA*. IOS Press, 2021. doi:10.3233/FAIA336.
- 653 8 Armin Biere, Matti Järvisalo, and Benjamin Kiesl. Preprocessing in SAT solving. In Armin
654 Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability—*
655 *Second Edition*, volume 336 of *FAIA*, chapter 9, pages 391–435. IOS Press, 2021. doi:
656 10.3233/FAIA200992.
- 657 9 Ronen I. Brafman. A simplifier for propositional formulas with many binary clauses. *IEEE*
658 *Trans. Syst. Man Cybern. Part B*, 34(1):52–59, 2004. doi:10.1109/TSMCB.2002.805807.
- 659 10 Miguel Cabral, Mikolás Janota, and Vasco M. Manquinho. SAT-based leximax optimisation
660 algorithms. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on*
661 *Theory and Applications of Satisfiability Testing, SAT*, volume 236 of *LIPICs*, pages 29:1–29:19.
662 Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.SAT.2022.29.
- 663 11 João Cortes, Inês Lynce, and Vasco M. Manquinho. New core-guided and hitting set algorithms
664 for multi-objective combinatorial optimization. In Sriram Sankaranarayanan and Natasha
665 Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 29th*
666 *International Conference, TACAS*, volume 13994 of *LNCS*, pages 55–73. Springer, 2023.
667 doi:10.1007/978-3-031-30820-8_7.
- 668 12 Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elim-
669 ination. In Fahiem Bacchus and Toby Walsh, editors, *Theory and Applications of Satisfiability*
670 *Testing, 8th International Conference, SAT*, volume 3569 of *LNCS*, pages 61–75. Springer,
671 2005. doi:10.1007/11499107_5.
- 672 13 Niklas Eén and Niklas Sörensson. Temporal induction by incremental SAT solving. In Ofer
673 Strichman and Armin Biere, editors, *First International Workshop on Bounded Model Checking,*
674 *BMC@CAV*, volume 89 of *Electronic Notes in Theoretical Computer Science*, pages 543–560.
675 Elsevier, 2003. doi:10.1016/S1571-0661(05)82542-3.

- 676 **14** Matthias Ehrgott. Efficiency and nondominance. In *Multicriteria Optimization (2. ed.)*,
677 chapter 2, pages 23–64. Springer, 2005. doi:10.1007/3-540-27659-9.
- 678 **15** Matthias Ehrgott. Multiobjective combinatorial optimization. In *Multicriteria Optimization*
679 *(2. ed.)*, chapter 8, pages 197–220. Springer, 2005. doi:10.1007/3-540-27659-9.
- 680 **16** Matthias Ehrgott. Other definitions of optimality—nonscalarizing methods. In *Multicriteria*
681 *Optimization (2. ed.)*, chapter 5, pages 127–150. Springer, 2005. doi:10.1007/3-540-27659-9.
- 682 **17** Matthias Ehrgott and Xavier Gandibleux. A survey and annotated bibliography of mul-
683 tiobjective combinatorial optimization. *OR Spectr.*, 22(4):425–460, 2000. doi:10.1007/
684 s002910000046.
- 685 **18** Jon William Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD
686 thesis, University of Pennsylvania, USA, 1995. UMI Order No. GAX95-32175.
- 687 **19** James F. Gimpel. A reduction technique for prime implicant tables. In *5th Annual Symposium*
688 *on Switching Circuit Theory and Logical Design, SWCT*, pages 183–191. IEEE Computer
689 Society, 1964. doi:10.1109/SWCT.1964.4.
- 690 **20** Andreia P. Guerreiro, João Cortes, Daniel Vanderpooten, Cristina Bazgan, Inês Lynce,
691 Vasco M. Manquinho, and José Rui Figueira. Exact and approximate determination of
692 the pareto front using minimal correction subsets. *Comput. Oper. Res.*, 153:106153, 2023.
693 doi:10.1016/j.cor.2023.106153.
- 694 **21** Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere. Clause
695 elimination for SAT and QSAT. *J. Artif. Intell. Res.*, 53:127–168, 2015. doi:10.1613/jair.
696 4694.
- 697 **22** Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Strong extension-free proof systems. *J.*
698 *Autom. Reason.*, 64(3):533–554, 2020. doi:10.1007/s10817-019-09516-0.
- 699 **23** Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient maxsat solver.
700 *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019. doi:10.3233/SAT190116.
- 701 **24** Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo. Clause redundancy and preprocessing
702 in maximum satisfiability. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors,
703 *Automated Reasoning—11th International Joint Conference, IJCAR*, volume 13385 of *LNCS*,
704 pages 75–94. Springer, 2022. doi:10.1007/978-3-031-10769-6_6.
- 705 **25** Christoph Jabs, Jeremias Berg, Andreas Niskanen, and Matti Järvisalo. MaxSAT-based
706 bi-objective boolean optimization. In Kuldeep S. Meel and Ofer Strichman, editors, *25th*
707 *International Conference on Theory and Applications of Satisfiability Testing, SAT*, volume
708 236 of *LIPICs*, pages 12:1–12:23. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2022.
709 doi:10.4230/LIPICs.SAT.2022.12.
- 710 **26** Mikolás Janota, Inês Lynce, Vasco M. Manquinho, and João Marques-Silva. PackUp: Tools
711 for package upgradability solving. *J. Satisf. Boolean Model. Comput.*, 8(1/2):89–94, 2012.
712 doi:10.3233/sat190090.
- 713 **27** Matti Järvisalo, Armin Biere, and Marijn Heule. Simulating circuit-level simplifications on
714 CNF. *J. Autom. Reason.*, 49(4):583–619, 2012. doi:10.1007/s10817-011-9239-9.
- 715 **28** Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In Bernhard Gram-
716 lich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning—6th International Joint*
717 *Conference, IJCAR*, volume 7364 of *LNCS*, pages 355–370. Springer, 2012. doi:10.1007/
718 978-3-642-31365-3_28.
- 719 **29** Saurabh Joshi, Ruben Martins, and Vasco M. Manquinho. Generalized totalizer encoding for
720 pseudo-boolean constraints. In Gilles Pesant, editor, *Principles and Practice of Constraint*
721 *Programming—21st International Conference, CP*, volume 9255 of *LNCS*, pages 200–209.
722 Springer, 2015. doi:10.1007/978-3-319-23219-5_15.
- 723 **30** Tuukka Korhonen, Jeremias Berg, Paul Saikko, and Matti Järvisalo. MaxPre: An extended
724 MaxSAT preprocessor. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications*
725 *of Satisfiability Testing—20th International Conference, SAT*, volume 10491 of *LNCS*, pages
726 449–456. Springer, 2017. doi:10.1007/978-3-319-66263-3_28.

- 727 31 Miyuki Koshimura, Hidetomo Nabeshima, Hiroshi Fujita, and Ryuzo Hasegawa. Minimal model
728 generation with respect to an atom set. In Nicolas Peltier and Viorica Sofronie-Stokkermans,
729 editors, *7th International Workshop on First-Order Theorem Proving, FTP*, volume 556 of
730 *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. URL: [http://ceur-ws.org/Vol-556/
731 paper06.pdf](http://ceur-ws.org/Vol-556/paper06.pdf).
- 732 32 Daniel Le Berre. Exploiting the real power of unit propagation lookahead. *Electron. Notes
733 Discret. Math.*, 9:59–80, 2001. doi:10.1016/S1571-0653(04)00314-2.
- 734 33 Chu Min Li. Integrating equivalency reasoning into davis-putnam procedure. In Henry A.
735 Kautz and Bruce W. Porter, editors, *Proceedings of the Seventeenth National Conference
736 on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial
737 Intelligence*, pages 291–296. AAAI Press / The MIT Press, 2000. URL: [http://www.aaai.
738 org/Library/AAAI/2000/aaai00-045.php](http://www.aaai.org/Library/AAAI/2000/aaai00-045.php).
- 739 34 Mark H. Liffiton and Karem A. Sakallah. Algorithms for computing minimal unsatisfiable
740 subsets of constraints. *J. Autom. Reason.*, 40(1):1–33, 2008.
- 741 35 Dmitry Malioutov and Kuldeep S. Meel. MLIC: A maxsat-based framework for learning
742 interpretable classification rules. In John N. Hooker, editor, *Principles and Practice of
743 Constraint Programming—24th International Conference, CP*, volume 11008 of *LNCS*, pages
744 312–327. Springer, 2018. doi:10.1007/978-3-319-98334-9_21.
- 745 36 Richard Ostrowski, Éric Grégoire, Bertrand Mazure, and Lakhdar Sais. Recovering and
746 exploiting structural knowledge from CNF formulas. In Pascal Van Hentenryck, editor,
747 *Principles and Practice of Constraint Programming—8th International Conference, CP*, volume
748 2470 of *LNCS*, pages 185–199. Springer, 2002. doi:10.1007/3-540-46135-3_13.
- 749 37 Tobias Paxian, Pascal Raiola, and Bernd Becker. On preprocessing for weighted MaxSAT. In
750 *Verification, Model Checking, and Abstract Interpretation, VMCAI*, volume 12597 of *LNCS*,
751 pages 556–577. Springer, 2021.
- 752 38 Takehide Soh, Mutsunori Banbara, Naoyuki Tamura, and Daniel Le Berre. Solving multi-
753 objective discrete optimization problems with propositional minimal model generation.
754 In J. Christopher Beck, editor, *Principles and Practice of Constraint Programming—23rd
755 International Conference, CP*, volume 10416 of *LNCS*, pages 596–614. Springer, 2017.
756 doi:10.1007/978-3-319-66158-2_38.
- 757 39 Sathiamoorthy Subbarayan and Dhiraj K. Pradhan. NiVER: Non-increasing variable elimina-
758 tion resolution for preprocessing SAT instances. In Holger H. Hoos and David G. Mitchell,
759 editors, *Theory and Applications of Satisfiability Testing, 7th International Conference, SAT*,
760 volume 3542 of *LNCS*, pages 276–291. Springer, 2004. doi:10.1007/11527695_22.
- 761 40 Miguel Terra-Neves, Inês Lynce, and Vasco M. Manquinho. Introducing pareto minimal
762 correction subsets. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of
763 Satisfiability Testing—20th International Conference, SAT*, volume 10491 of *LNCS*, pages
764 195–211. Springer, 2017. doi:10.1007/978-3-319-66263-3_13.
- 765 41 Miguel Terra-Neves, Inês Lynce, and Vasco M. Manquinho. Multi-objective optimization
766 through pareto minimal correction subsets. In Jérôme Lang, editor, *27th Joint Conference on
767 Artificial Intelligence, IJCAI*, pages 5379–5383. ijcai.org, 2018. doi:10.24963/ijcai.2018/
768 757.
- 769 42 Ekunda L. Ulungu and Jacques Teghem. Multi-objective combinatorial optimization problems:
770 A survey. *Journal of Multi-criteria Decision Analysis*, 3:83–104, 1994.
- 771 43 Allen Van Gelder. Toward leaner binary-clause reasoning in a satisfiability solver. *Ann. Math.
772 Artif. Intell.*, 43(1):239–253, 2005. doi:10.1007/s10472-005-0433-5.
- 773 44 Ramin Zabih and David A. McAllester. A rearrangement search strategy for determining
774 propositional satisfiability. In Howard E. Shrobe, Tom M. Mitchell, and Reid G. Smith,
775 editors, *Proceedings of the 7th National Conference on Artificial Intelligence, AAAI*, pages
776 155–160. AAAI Press / The MIT Press, 1988. URL: [http://www.aaai.org/Library/AAAI/
777 1988/aaai88-028.php](http://www.aaai.org/Library/AAAI/1988/aaai88-028.php).