Journal of Artificial Intelligence Research 80 (2024) 1223-1269 Submitted 08/2023; published 08/2024

From Single-Objective to Bi-Objective Maximum Satisfiability Solving

Christoph Jabs CHRISTOPH.JABS@HELSINKI.FI
Jeremias Berg JEREMIAS.BERG@HELSINKI.FI
Andreas Niskanen ANDREAS.NISKANEN@HELSINKI.FI
Matti Järvisalo MATTI.JARVISALO@HELSINKI.FI
Department of Computer Science,
University of Helsinki,
Finland

Abstract

The declarative approach is key to efficiently finding optimal solutions to various types of NP-hard real-world combinatorial optimization problems. Most work on practical declarative solvers—ranging from classical integer programming to finite-domain constraint optimization and maximum satisfiability (MaxSAT)—has focused on optimization under a single objective; fewer advances have been made towards efficient declarative techniques for multi-objective optimization problems. Motivated by significant recent advances in practical solvers for MaxSAT, in this work we develop BiOptSat, an exact declarative approach for finding Pareto-optimal solutions to bi-objective optimization problems, with propositional logic as the underlying constraint language. BiOptSat can be viewed as an instantiation of the lexicographic method. The approach makes use of a single Boolean satisfiability solver that is incrementally employed throughout the entire search procedure, allowing for finding a single Pareto-optimal solution, finding one representative solution for each non-dominated point, and enumerating all Pareto-optimal solutions. We detail several algorithmic instantiations of BiOptSat, each building on recent algorithms proposed for single-objective MaxSAT. We empirically evaluate the instantiations compared to recently-proposed alternative approaches to multi-objective MaxSAT solving on several real-world domains from the literature, showing the practical benefits of our approach.

1. Introduction

The declarative approach—with its various instantiations, ranging from classical integer programming (Wolsey, 2020) to finite-domain constraint optimization (Rossi, van Beek, & Walsh, 2006) and Boolean satisfiability (SAT)-based (Biere, Heule, van Maaren, & Walsh, 2021) approaches such as maximum satisfiability (MaxSAT) (Bacchus, Järvisalo, & Martins, 2021), SAT modulo theories (Barrett, Sebastiani, Seshia, & Tinelli, 2021), and answer set programming (Niemelä, 1999)—is key to efficiently finding optimal solutions to various types of NP-hard real-world combinatorial optimization problems. In this work, we build on recent advances in MaxSAT solving (Bacchus et al., 2021). In particular, we develop an exact declarative programming based algorithmic approach to finding so-called Pareto-optimal solutions to bi-objective optimization problems encoded in propositional logic.

Much like how SAT solvers find various applications in solving NP-hard decision problems via compact propositional encodings, MaxSAT allows for succinctly encoding a wide range of NP-hard real-world optimization problems. Modern MaxSAT solvers can scale up to finding provably optimal solutions to instances of very significant size, and nowadays often outperform key competing approaches, especially when facing optimization problems the underlying constraints of which allow for natural encoding on the propositional level. This is in particular due to conflict-driven clause learning SAT solvers (Marques-Silva, Lynce, & Malik, 2021), the success of which has translated to increasing success of the Boolean optimization paradigm of MaxSAT (Bacchus et al., 2021).

Most work on practical algorithms and their implementations (i.e., solvers) for exact declarative optimization—including MaxSAT—has focused on optimization under a single objective. However, various real-world settings give rise to multiple, often conflicting objectives (Ehrgott, 2005). Whereas for a single objective function there is a clear minimum (or maximum) and objective values can be unambiguously compared, a notion of optimality of a solution becomes less obvious to define naturally for multi-objective settings and in particular when—as is generally the case—there is no clear preference over which objective should be considered the primary one. A commonly considered notion of "optimality" in the multi-objective case is that of Pareto optimality (also called Pareto efficiency in some contexts) (Arora, 2004; Ehrgott, 2005). Intuitively, a Pareto-optimal solution is one which cannot be improved with respect to any single objective without making it worse with respect to another objective.

Under Pareto optimality, several related tasks can be distinguished: (i) finding a single Pareto-optimal solution, (ii) finding a representative solution for each non-dominated point (i.e., tuple of objective values of Pareto-optimal solutions), and (iii) finding all Pareto-optimal solutions. Many approaches (Soh, Banbara, Tamura, & Le Berre, 2017; Terra-Neves, Lynce, & Manquinho, 2018b; Janota, Morgado, Santos, & Manquinho, 2021) to multi-objective optimization under Pareto optimality tend to focus on the second task where a single solution per non-dominated point is computed. The third task goes one step further and enumerates the full Pareto front (i.e., all Pareto-optimal solutions), even when multiple solutions share the same objective values. The techniques we develop in this work are applicable to each of these three tasks.

We focus in particular on bi-objective optimization, that is, the task of finding the Pareto-optimal solutions—or in other words, computing representative solutions for each so-called non-dominated point—under two conflicting objectives. While the solutions of interest can quickly become hard to grasp when the number of objectives is increased, bi-objective problems naturally arise in the real world. One topical setting is that of learning interpretable classifiers (Jin & Sendhoff, 2008; Malioutov & Meel, 2018; Narodytska, Ignatiev, Pereira, & Marques-Silva, 2018; Ignatiev, Pereira, Narodytska, & Marques-Silva, 2018; Hu, Siala, Hebrard, & Huguet, 2020; Yu, Ignatiev, Stuckey, & Bodic, 2021; Ignatiev, Marques-Silva, Narodytska, & Stuckey, 2021; Ghosh, Malioutov, & Meel, 2022) such as decision rules or other logically-oriented representations from data. In this context, interpretability—often understood as the size of a representation, with the intuition that "the smaller the representation, the easier it is for humans to interpret"—is intrinsically conflicting with the objective of accurately representing data at hand. Hence the two objectives of minimizing size of the representation and minimizing classification error give naturally rise to combinatorial bi-optimization problems. While some Pareto-optimal solutions may be found by minimizing a single linear combination of the objectives (i.e., with the weighted sum method (Ehrgott, 2005) used for example by Malioutov and Meel (2018) in the particular context of learning interpretable classifiers), doing so does not provide guarantees on which Pareto-optimal solutions are obtained and—even more severely—there can be certain Pareto-optimal solutions that can never be discovered through such means. This further motivates the development of effective optimization procedures that work directly and exactly on the bi-objective level.

As the main contribution of this work, we develop BiOptSat, an approach to SAT-based bi-objective optimization. The BiOptSat approach allows for computing representatives for each non-dominated point in an ordered fashion. The approach also extends naturally to enumerating *all* solutions at each non-dominated point, hence capturing a more generic setting than the multi-*level* setting (Marques-Silva, Argelich, Graca, & Lynce, 2011) of lexicographic optimization which assumes a preference order among the objectives. Our approach—which can be viewed as an instantiation of the lexicographic method (Wassenhove & Gelders, 1980; Marler & Arora, 2004) via SAT solving—allows for building on advances in MaxSAT solving algorithms. However, instead of using MaxSAT solvers as black boxes, we make use of incremental SAT solving (Eén & Sörensson, 2003; Marques-Silva et al., 2021) directly in implementing the approach. As the approach allows for making use of a MaxSAT algorithm of choice, we study the effectiveness of different algorithmic choices that include both solution-improving (sometimes called SAT-UNSAT) (Eén & Sörensson, 2006; Le Berre & Parrain, 2010; Bacchus et al., 2021) and core-guided (Marques-Silva & Planes, 2007; Ansótegui, Bonet, & Levy, 2009; Morgado, Dodaro, & Marques-Silva, 2014; Ignatiev, Morgado, & Marques-Silva, 2019) instantiations. In particular, we propose six different instantiations of BiOptSat that differ in how the minimization of what we refer to as the "increasing" objective among the two objectives is handled. The first four build on the SAT-UNSAT (Le Berre & Parrain, 2010), UNSAT-SAT (Fu & Malik, 2006), MSU3 (Marques-Silva & Planes, 2007) and OLL (Morgado et al., 2014) MaxSAT algorithms, respectively, and are adapted to the bi-objective setting by building on the fact that a bound on the other, so-called "decreasing" objective needs to be enforced during optimization. The final two instantiations are hybrids, switching from a core-guided approach (MSU3 and OLL, respectively) to the SAT-UNSAT-based instantiation during search with the aim of combining the advantages of the two search strategies. We also detail refinements for the approach as liftings from single-objective core-guided MaxSAT: lazy building of the pseudo-Boolean constraints for both objectives to reduce the number of clauses in the solver, and application-specific refinements of blocking clauses for speeding up enumeration of all Pareto-optimal solutions.

We provide an open-source implementation (available at `https://bitbucket.org/coreo-group/bioptsat/`) of all the described instantiations of BiOptSat. We also empirically evaluate the performance of BiOptSat on several benchmark domains against key exact SAT-based competitors implemented in the same code base, namely, enumeration of so-called *P*-minimal solutions (Soh et al., 2017) (as one of the closest ones to ours) originally proposed in the context of SAT-based constraint optimization (Koshimura, Nabeshima, Fujita, & Hasegawa, 2009), and an implicit hitting set style approach in the flavour of the recently-proposed Seesaw framework (Janota et al., 2021) (for more discussion on related work, see Sections 2.6 and 5). Furthermore, we compare the performance of BiOptSat also to existing implementations based on enumerating so-called Pareto-MCSes (Terra-Neves et al., 2018b), lower-bounding search, and a further recent approach coined as a form of implicit hitting set optimization (Cortes, Lynce, & Manquinho, 2023). While there are no evident standard benchmark sets in the context of multi-objective optimization, for the empirical evaluation we consider several benchmark domains: learning Pareto-optimal interpretable decision rules (as a generalization of settings for which MaxSAT-based solutions have been proposed) (Malioutov & Meel, 2018), bi-objective set covering (as earlier considered in the work presenting enumeration of P-minimal solutions (Soh et al., 2017)), the package upgradeability problem under various objectives (Janota, Lynce, Manquinho, & Marques-Silva, 2012), and truly bi-objective benchmark instances reverse-engineered from MaxSAT Lib. The empirical results suggest that our approach outperforms the competing approaches. Furthermore, the efficiency of our approach is impacted by the choice of the integrated MaxSAT algorithm within the approach as well as by further design choices such as how objective function coefficients are handled within the approach.

The rest of this article is organized as follows. We start with the necessary preliminaries on SAT, MaxSAT, and the extension of MaxSAT to bi-objective optimization, as well as an overview of related work on practical algorithmic approaches to MaxSAT-based multi-objective optimization (Section 2.1). We then turn to the main algorithmic contributions of this work, detailing the BiOptSat approach and several variants of instantiating one of its subroutines (Section 3). The results from an extensive empirical evaluation of the approach are presented in Section 4. Finally, before conclusions, we review related work (Section 5).

A preliminary version of this article was presented at 25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022) (Jabs, Berg, Niskanen, & Järvisalo, 2022). The present article considerably expands on the preliminary SAT 2022 version in several ways. In terms of algorithmic advances, as a notable improvement over the previously-proposed version of BiOptSat (Jabs et al., 2022), we propose the use of *incremental* pseudo-Boolean encodings rather than expanding pseudo-Boolean constraints into cardinality constraints, and outline a further hybrid instantiation of BiOptSat based on the OLL algorithm. In terms of empirical evaluation, we extend the evaluation with further benchmarks and further competing approaches (Cortes et al., 2023) that were published after the SAT 2022 paper, and also provide further empirical results on the impact of different implementation details on the efficiency of BiOptSat. Furthermore, the discussion overall has been expanded to be self-contained.

2. Preliminaries

As preliminaries, we recall Boolean satisfiability (SAT) and necessary related concepts, and detail bi-objective MaxSAT as the main focus of our work.

2.1 Boolean Satisfiability

For a Boolean variable v there are two literals, the positive v and the negative $\neg v$. A clause C is a set of (i.e., disjunction over) literals, and a CNF formula F is a set of (i.e., conjunction over) clauses. The set of variables and literals appearing in F are denoted by $\textsc{var}(F)$ and $\textsc{lit}(F)$, respectively. A truth assignment τ maps Boolean variables to 1 (true) or 0 (false). The semantics of truth assignments are extended to a negated variable $\neg v$, a clause C and a CNF formula F in the standard way: $\tau(\neg v) = 1 - \tau(v)$, $\tau(C) = \max\{\tau(l) \mid l \in C\}$, and $\tau(F) = \min\{\tau(C) \mid C \in F\}$. When convenient, we view an assignment τ over the set

$\text{VAR}(F)$ of variables as the set of literals assigned to 1, i.e., $\tau = \{v \mid v \in \text{VAR}(F), \tau(v) = 1\} \cup \{\neg v \mid v \in \text{VAR}(F), \tau(v) = 0\}$. Under this convention $\tau(l) = 1$ corresponds to $l \in \tau$ and $\tau(l) = 0$ to $\neg l \in \tau$. An assignment τ for which $\tau(F) = 1$ is a solution to F. A CNF formula F is satisfiable if it has a solution, and otherwise unsatisfiable. The Boolean satisfiability (SAT) problem asks to decide whether a given CNF formula is satisfiable.

As standardly employed, it is well-known that any propositional formula ϕ can be translated into a linear-size CNF formula using auxiliary variables naming the subformulas of ϕ, so that the CNF representation of ϕ is satisfied by exactly the same assignments on the variables in ϕ (Tseitin, 1983).

2.2 Incremental SAT Solving

An incremental SAT solver (Eén & Sörensson, 2003; Marques-Silva et al., 2021) is an implementation of a decision procedure (in our context, a conflict-driven clause learning (CDCL) algorithm (Marques-Silva et al., 2021)) that, given a CNF formula F and a set $\mathcal{A}$ of literals, can decide whether there is a solution τ to F that extends $\mathcal{A}$, i.e., for which $\tau \supset \mathcal{A}$. We abstract the use of an incremental SAT solver into the function `isSAT`. More precisely, $\texttt{isSAT}(F, \mathcal{A})$ denotes a call to an incremental SAT solver under the formula F and the *assumptions* specified by the set $\mathcal{A}$ of literals. The call returns a tuple $(\text{res}, \tau, \kappa)$ where res is either `SAT` ("satisfiable") or `UNSAT` ("unsatisfiable") depending on whether F is satisfiable under the assumptions or not. If res $=$ `SAT`, τ is populated by a satisfying assignment. If res $=$ `UNSAT`, then $\kappa \subset \{\neg l \mid l \in \mathcal{A}\}$ is populated with a subset of negated assumptions such that $F \wedge \bigwedge_{l \in \kappa}(\neg l)$ is unsatisfiable. Such a κ is an *unsatisfiable core* of F and can also be viewed as a clause containing negated assumptions entailed by F. When the formula F is clear from context, we will simply write $\texttt{isSAT}(\mathcal{A})$ for an invocation of an incremental SAT solver on F under the assumptions $\mathcal{A}$. In case the returned core of a specific call is not used, we will omit it as a return parameter.

2.3 Pseudo-Boolean Expressions and Constraints

A pseudo-Boolean (PB) expression $E = \sum_i c_i \cdot l_i$ is a linear combination of terms, each consisting of a literal l_i and a coefficient c_i. The set of literals appearing in E is denoted by $\text{LIT}(E)$, and the coefficient of a literal l in E is $\text{COEFF}(E, l)$. The sum of all coefficients in E is denoted by $\text{SUM-COEFF}(E) = \sum_{l \in \text{LIT}(E)} \text{COEFF}(E, l)$. Further, $E|_L = \sum_{l \in L} \text{COEFF}(E, l) \cdot l$ denotes the restriction of an expression E to a set of literals L.

A PB constraint is a PB expression the value of which is bounded by a constant. We will in particular make use of PB constraints of form $\sum_i c_i \cdot l_i < B$, where each l_i is a literal, c_i a positive integer, and B an integer bound. Such a constraint is satisfied by an assignment τ if $\sum_i c_i \cdot \tau(l_i) < B$. Further, we will make extensive use of CNF formulas equivalent to so-called reified PB constraints. More precisely, given an expression $E = \sum_i c_i \cdot l_i$ and a maximum bound `UB`, $\text{PBCNF}(E, \texttt{UB})$ denotes a CNF formula that defines a set $\{\langle E < k \rangle \mid k = 1, \ldots, \texttt{UB}\}$ of output literals such that any solution of $\text{PBCNF}(E, \texttt{UB})$ that sets $\langle E < k \rangle$ to 1 also satisfies $\sum_i c_i \cdot l_i < k$. Here $\text{PBCNF}(E, \texttt{UB})$ can be seen as the CNF representation of the set of reified PB constraints of the form $\langle E < k \rangle \rightarrow \sum_i c_i \cdot l_i < k$ for all $k = 1 \ldots \texttt{UB}$. For clarity of presentation, we often use $\langle E \leq k \rangle$ to denote $\langle E < k+1 \rangle$, and

$\textsc{PbCnf}(E)$ to denote $\textsc{PbCnf}(E, \textsc{sum-coeff}(E))$, i.e., the PB encoding using the maximum upper bound which is the sum of all coefficients in E.

As an important special case of PB constraints, we use $\textsc{CardCnf}(L, \texttt{UB})$ to denote a CNF formula equivalent to a cardinality constraint $\sum_{l \in L} l < B$ over a set L of literals with a maximum bound UB. In other words, $\textsc{CardCnf}(L, \texttt{UB})$ is a CNF formula that defines a set of output literals of the form $\langle L < k \rangle$ for $k = 1 \ldots \texttt{UB}$ such that any solution τ to $\langle L < k \rangle$ that sets $\tau(\langle L < k \rangle) = 1$ also assigns less than k of the literals in L to 1. Various CNF encodings of PB and cardinality constraints are known (Bailleux & Boufkhad, 2003; Eén & Sörensson, 2006; Tamura, Banbara, & Soh, 2013; Joshi, Martins, & Manquinho, 2015). Although the algorithmic approach developed in this work is not tied to a specific encoding, as described later in more detail, on the implementation level we make use of the so-called (generalized) totalizer encoding (Bailleux & Boufkhad, 2003; Joshi et al., 2015).

2.4 Maximum Satisfiability

Maximum satisfiability (MaxSAT) (Bacchus et al., 2021) is the optimization variant of SAT. A MaxSAT instance (F, O) consists of a set F of clauses and an *objective* $O = \sum_i c_i \cdot l_i$ represented as a pseudo-Boolean expression with positive constants, where each l_i is over a variable in F.[1] An assignment τ is a solution to a MaxSAT instance (F, O) if τ satisfies F. The objective value (or cost) $O(\tau)$ of τ is $O(\tau) = \sum_{l \in \textsc{lit}(O)} \tau(l) \cdot \textsc{coeff}(O, l)$. The task in MaxSAT is to find an optimal solution of (F, O), i.e., a solution τ of F that minimizes $O(\tau)$ over all solutions. In the context of MaxSAT, $\textsc{PbCnf}(O)$ denotes a CNF formula defining output literals of the form $\langle O < k \rangle$ such that $O(\tau) < k$ for any solution τ to $F \cup \textsc{PbCnf}(O)$ for which $\tau(\langle O < k \rangle) = 1$. This forms the basis for the so-called solution-improving approach to solving MaxSAT.

Example 1. *Consider the MaxSAT instance (F, O) with $F = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$ and $O = b_1 + 3 \cdot b_2 + 5 \cdot b_3$. Then $\textsc{PbCnf}(O, 9)$ defines the output literals $\langle O < k \rangle$ for $k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Now, **isSAT**$(F \wedge \textsc{PbCnf}(O, 9), \{\langle O < 4 \rangle\})$ reports satisfiability and the solution τ that sets $\tau(b_2) = 1$ and all others to 0. On the other hand, **isSAT**$(F \wedge \textsc{PbCnf}(O, 9), \{\langle O < 3 \rangle\})$ reports unsatisfiability. Hence the instance has minimum-cost 3.*

2.5 Bi-Objective Maximum Satisfiability

The main focus of this work is on developing practical algorithms for the following natural generalization of MaxSAT to bi-objective optimization. A bi-objective MaxSAT instance is a triple $I = (F, O_1, O_2)$, consisting of a formula F and two objectives, O_1 and O_2. We lift all terminology and concepts of the single-objective setting to the bi-objective case. Most notably, an assignment τ is a solution to I if $\tau(F) = 1$. The cost of τ with respect to O_1 is $O_1(\tau)$ and $O_2(\tau)$ with respect to O_2.

The two objectives O_1 and O_2 may be (and typically are) in conflict with each other in the sense that no solution is optimal for both O_1 and O_2 independently. With this in mind, commonly studied notions of optimality in the multi-objective setting are based on the

1. We note that this definition of MaxSAT in terms of a set of clauses and an objective captures the—arguably more classical—definition of MaxSAT in terms of hard and weighted soft clauses via the so-called blocking variable transformation (Leivo, Berg, & Järvisalo, 2020; Bacchus et al., 2021). Specifically, our definition corresponds to the so-called weighted partial MaxSAT problem.

$$
\begin{aligned}
L =& \{\neg i_1, \neg i_2, \neg i_3, \neg i_4, \neg d_1, \neg d_2, \neg d_3, \neg d_4\}, \\
F =& \textsc{CardCnf}(L, 5) \cup \Big\{ (\langle L \leq 4 \rangle), \\
& (i_1 \vee i_2), (i_2 \vee i_3), (i_2 \vee i_4) \\
& (d_1 \vee d_2), (d_2 \vee d_3), (d_2 \vee d_4) \Big\}, \\
O_{\mathrm{I}} =& i_1 + i_2 + i_3 + i_4, \\
O_{\mathrm{D}} =& d_1 + d_2 + d_3 + d_4.
\end{aligned}
$$

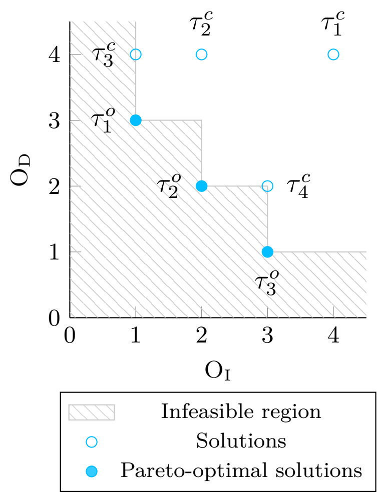

Figure 1: Left: An example bi-objective MaxSAT instance $I = (F, O_{\mathrm{I}}, O_{\mathrm{D}})$ Right: the feasible region of F in the objective space defined by O_{I} and O_{D}. The solutions τ_1^o and τ_2^o (solid points) are Pareto-optimal, while τ_i^c for $i = 1, \ldots, 4$ are not.

following dominance relation between solutions. A solution τ_1 dominates τ_2 if (i) $O_i(\tau_1) \leq O_i(\tau_2)$ for $i = 1, 2$ and (ii) either $O_1(\tau_1) < O_1(\tau_2)$ or $O_2(\tau_1) < O_2(\tau_2)$. A solution τ is *Pareto*-optimal if no other solution dominates it. The non-dominated set $\textsc{non-dom}(I)$ of I consists of the non-dominated points, i.e., the cost-pairs of all Pareto-optimal solutions, denoted by $\textsc{non-dom}(I) = \{(O_1(\tau), O_2(\tau)) \mid \tau \text{ is Pareto-optimal wrt. } I\}$.

Example 2 (Running Example). *A bi-objective MaxSAT instance $I = (F, O_I, O_D)$ is shown on the left in Figure 1. Here* $\textsc{CardCnf}(L, 5)$ *together with $(\langle L \leq 4 \rangle)$ enforces that any solution must assign at least 4 objective literals of O_I and O_D to 1. The solution space is illustrated in Figure 1 on the right. The three solid dots correspond to the three non-dominated points $\{(1, 3), (2, 2), (3, 1)\}$ of I. Examples of Pareto-optimal solutions corresponding to these points are $\tau_1^o = \{i_2, d_1, d_3, d_4, \neg i_1, \neg i_3, \neg i_4, \neg d_2\}$, $\tau_2^o = \{i_1, i_2, d_1, d_2, \neg i_3, \neg i_4, \neg d_3, \neg d_4\}$ and $\tau_3^o = \{i_1, i_3, i_4, d_2, \neg i_2, \neg d_1, \neg d_3, \neg d_4\}$, respectively. The solution $\tau_3^c = \{i_2, d_1, d_2, d_3, d_4, \neg i_1, \neg i_3, \neg i_4\}$ is dominated by τ_1^o because $O_I(\tau_1^o) \leq O_I(\tau_3^c)$ and $O_D(\tau_1^o) < O_D(\tau_3^c)$.*

The following three tasks can be identified for the multi-objective problem setting:

(i) finding a representative solution for every non-dominated point (earlier considered e.g. by Soh et al. (2017), Janota et al. (2021)),

(ii) finding all Pareto-optimal solutions (earlier considered e.g. by Isermann (1979)), and

(iii) finding a representative leximax-optimal solution (earlier considered e.g. by Cabral, Janota, and Manquinho (2022)).

Note that each non-dominated point can correspond to several Pareto-optimal solutions, leading to the two distinct tasks of either computing a single representative for each non-dominated point or computing all Pareto-optimal solutions. The algorithmic approach developed in this work is applicable to each of the three tasks.

In addition to the general problem of computing all Pareto-optimal solutions, our work relates to two important special cases of Pareto-optimality: lexicographic max-ordering (leximax) and lexicographic optimality (Ehrgott, 2005). The optimal solutions under both of these notions are a subset of the Pareto-optimal solutions. Thus finding an optimal solution under leximax and lexicographic optimality can be considered easier than finding a representative solution for every non-dominated point.

In leximax optimization (Ehrgott, 2005), the goal is to find solutions that—informally speaking—minimize the "worst" objective first. More precisely, given the non-dominated set $\textsc{non-dom}(I)$ of a bi-objective MaxSAT instance $I = (F, O_1, O_2)$ the Pareto-optimal solutions of I that correspond to the elements in $\textsc{non-dom}(I)$ with the lowest maximum objective value—i.e., the elements $(c_1, c_2) \in \textsc{non-dom}(I)$ that minimize $\max\{c_1, c_2\}$—are called leximax-optimal. For bi-objective instances, an alternative description of leximax-optimal solutions are the Pareto-optimal solutions that minimize the difference between the two objective values. Notice that different leximax-optimal solutions can correspond to different elements in $\textsc{non-dom}(I)$.

Example 3. *Consider again the formula F and the objectives O_I and O_D in Figure 1. The solution $\tau_2^o = \{i_1, i_2, d_1, d_2, \neg i_3, \neg i_4, \neg d_3, \neg d_4\}$ with costs $(O_I(\tau_2^o), O_D(\tau_2^o)) = (2, 2)$ is leximax-optimal: the greatest objective value for this solution is* 2, *and this is smaller than for the two other Pareto-optimal solutions for which the greatest objective value is* 3.

Finally, our work is also related to so-called lexicographic optimization (Ehrgott, 2005) in which the solutions to a bi-objective MaxSAT instance $I = (F, O_1, O_2)$ are ranked primarily based on their costs wrt. O_1 and secondarily based on their costs wrt. O_2. More precisely, a solution τ dominates another solution τ' in the lexicographic sense if (a) $O_1(\tau) < O_1(\tau')$ or (b) $O_1(\tau) = O_1(\tau')$ and $O_2(\tau) < O_2(\tau')$. A solution τ is lexicographically optimal if τ is not dominated (in the lexicographic sense) by any other solution. For an alternative description, the lexicographically optimal solutions of I are the Pareto-optimal solutions which correspond to the non-dominated point with the lowest value for the cost of O_1, i.e., the lowest first element of the tuple. Note that lexicographic optimization is reducible to a single-objective optimization problem via the so-called weighted sum method (Ehrgott, 2005). More precisely, given a bi-objective MaxSAT instance $I = (F, O_1, O_2)$, consider the single-objective problem $I^{\textsc{sum}} = (F, \lambda \cdot O_1 + O_2)$ obtained by multiplying O_1 with a coefficient $\lambda > \textsc{sum-coeff}(O_2)$ and summing the result with O_2. The optimal (minimum-cost) solutions of $I^{\textsc{sum}}$ are exactly the lexicographically optimal solutions of I and vice versa. Thus, while the algorithmic approach we develop in this work is applicable—as is any method capable of computing all Pareto-optimal solutions—to lexicographic optimization as a special case, lexicographic optimization is in this sense not a true bi-objective problem.

Example 4. *Consider again the bi-objective MaxSAT instance (F, O_I, O_D) from Figure 1. All solutions corresponding to the non-dominated point $(1, 3)$ (such as $\tau_1^o = \{i_2, d_1, d_3, d_4, \neg i_1, \neg i_3, \neg i_4, \neg d_2\}$) are lexicographically optimal. These are also the solutions of the single-objective MaxSAT problem (F, O_{sum}), where $O_{sum} = 5 \cdot O_I + O_D$. To see this, note that setting any $l \in O_I$ to 1 incurs cost $O_{sum}(l) = 5$, i.e., more than setting* all 4 *literals in O_D to* 1.

2.6 Existing SAT-Based Approaches to Bi-Objective MaxSAT

We continue with an overview of related approaches to bi-objective MaxSAT solving: enumerating P-minimal solutions, multi-objective lower-bounding search, enumerating Pareto-minimal correction sets, and approaches based on the implicit hitting set paradigm. Later, we will empirically compare the performance of the BiOptSat approach developed in this work to each of these approaches.

P-Minimal Solution Enumeration. The approach perhaps closest to the one developed in this work finds the non-dominated set by enumerating so-called P-minimal solutions (Soh et al., 2017; Koshimura et al., 2009). Originally, this approach was proposed in the context of solving constraint satisfaction problems (Rossi et al., 2006) encoded in CNF via the so-called order encoding (Tamura et al., 2013).

For a bi-objective MaxSAT instance $I = (F, O_1, O_2)$, let $F^{\mathrm{W}} = F \wedge \textsc{PbCnf}(O_1) \wedge \textsc{PbCnf}(O_2)$ be the CNF formula consisting of the clauses in F together with a (reified) PB constraint over each objective. Computing the non-dominated set of I corresponds to enumerating the solutions of F^{W} that are subset-minimal wrt. the outputs of $\textsc{PbCnf}(O_1)$ and $\textsc{PbCnf}(O_2)$ assigned to 0. In particular, if P is the set of those outputs, then a solution τ^m is P-minimal if for no other solution τ does it hold that $\{l \mid l \in P, \tau(l) = 0\} \subsetneq \{l \mid l \in P, \tau^m(l) = 0\}$. Each P-minimal solution τ^m corresponds to a non-dominated point (k_1, k_2) of I, where k_i for $i \in \{1, 2\}$ is the largest value for which $\langle O_i < k_i \rangle$ is set to 0 by τ^m.

Enumeration of P-minimal solutions (Koshimura et al., 2009) works by iteratively

(i) using a SAT solver to obtain a solution τ to F^{W},

(ii) iteratively minimizing the subset of variables of P set to 0 by τ, and

(iii) once a minimal solution τ^m has been found, adding to the working formula F^W the clause $(\langle O_1 < k_1 \rangle \vee \langle O_2 < k_2 \rangle)$, where $k_i = O_i(\tau^m)$ for $i = 1, 2$, resulting in ruling out the same non-dominated point from being discovered in subsequent iterations.

The procedure terminates when the working formula F^W becomes unsatisfiable. We refer to this algorithm for enumerating P-minimal solutions as "P-minimal" for short.

Enumeration of Pareto-Minimal Correction Sets. Another earlier proposed approach to multi-objective MaxSAT is based on the enumeration of so-called Pareto-minimal correction (Terra-Neves et al., 2018b). For a bi-objective MaxSAT instance (F, O_1, O_2), let $\mathcal{L} = \textsc{lit}(O_1) \cup \textsc{lit}(O_2)$ be the set of all literals in the objectives O_1 and O_2. A Pareto-MCS (Terra-Neves et al., 2018b; Terra-Neves, Lynce, & Manquinho, 2018a, 2018c; Guerreiro, Cortes, Vanderpooten, Bazgan, Lynce, Manquinho, & Figueira, 2023) (wrt. O_1 and O_2) is a set $M \subset \mathcal{L}$ of objective literals such that there is a Pareto-optimal solution τ to (F, O_1, O_2) that sets $\tau(l) = 1$ for all $l \in M$ and $\tau(l) = 0$ for all $l \in \mathcal{L} \setminus M$.

The computation of Pareto-optimal solutions can be reduced to the computation of Pareto-MCSes (Terra-Neves et al., 2018b). The task of computing Pareto-MCSes is in turn accomplished by enumerating all subsets $T \subset \mathcal{L}$ for which (i) $F \wedge \bigwedge_{l \in \mathcal{L} \setminus T} (\neg l)$ is satisfiable and (ii) $F \wedge \bigwedge_{l \in \mathcal{L} \setminus T'} (\neg l)$ is unsatisfiable for all $T' \subsetneq T$. Let $\mathcal{T}$ consist of all such sets. The Pareto-optimal solutions are obtained by extracting the solutions satisfying $F \wedge \bigwedge_{l \in \mathcal{L} \setminus T} (\neg l)$ for all $T \in \mathcal{T}$ and removing the dominated ones (Terra-Neves et al., 2018b). We will refer to this algorithm for enumerating Pareto-optimal solutions via enumerating Pareto-MCSes as "Pareto-MCS" for short.

The computation of $\mathcal{T}$ corresponds to the enumeration of (subset-)minimal correction sets, a problem for which numerous algorithms have been proposed (Morgado, Liffiton, & Marques-Silva, 2012; Previti, Mencía, Järvisalo, & Marques-Silva, 2017; Grégoire, Izza, & Lagniez, 2018; Bendík & Cerna, 2020; Koshimura & Satoh, 2020). Intuitively, the search performed by Pareto-MCS can be described as iteratively refining a set of solutions that dominates an increasing number of solutions, towards constructing the full set of Pareto-optimal solutions to the problem instance at hand. Importantly, note that any solution in such a set can be guaranteed to be Pareto-optimal only once the set $\mathcal{T}$ has been completely computed. More recent work (Guerreiro et al., 2023) aims to provide guarantees on the quality of a suboptimal set of solutions by reformulating the objective, similarly to P-minimal (Soh et al., 2017).

Multi-Objective Lower-Bounding Search. Cortes et al. (2023) recently proposed a lower-bounding approach to multi-objective MaxSAT. Similarly as P-minimal, to solve a bi-objective MaxSAT instance $I = (F, O_1, O_2)$ the approach builds the working formula $F^{\mathrm{W}} = F \wedge \textsc{PbCnf}(O_1) \wedge \textsc{PbCnf}(O_2)$. In contrast to P-minimal, the search is lower-bounding in the following sense. Initially, a bound of zero on both objectives is enforced via assumptions. Whenever the SAT solver reports `UNSAT`, the current bound of each objective whose output literals appear in the obtained core is increased. If the SAT solver instead returns `SAT`, the obtained solution is minimized into a Pareto-optimal solution in an inner loop. The intuition for why such an inner loop is necessary is that the bound relaxations performed after each `UNSAT` may increment both bounds simultaneously and may thus relax the constraints too much. Finally, when a Pareto-optimal solution is obtained, a clause that blocks solutions dominated by the found Pareto-optimal solution is added to the SAT solver. We will refer to this approach as CLM-LB.

Implicit Hitting Set based Bi-objective Optimization: Seesaw and CLM-IHS. The implicit hitting set (IHS) approach (Reggia, Nau, & Wang, 1983; Reiter, 1987; Parker & Ryan, 1996; Chandrasekaran, Karp, Moreno-Centeno, & Vempala, 2011; Moreno-Centeno & Karp, 2013) has, among other formalisms and problems, been successfully applied to MaxSAT (Davies & Bacchus, 2011, 2013b, 2013a; Saikko, Berg, & Järvisalo, 2016; Berg, Bacchus, & Poole, 2020) as well as various other NP-hard decision and optimization problems (Ignatiev, Previti, Liffiton, & Marques-Silva, 2015; Ignatiev, Morgado, & Marques-Silva, 2016; Saikko, Wallner, & Järvisalo, 2016; Saikko, Dodaro, Alviano, & Järvisalo, 2018; Fazekas, Bacchus, & Biere, 2018; Smirnov, Berg, & Järvisalo, 2021, 2022). Recently, two approaches generalizing the IHS framework to bi-objective optimization have been proposed: Seesaw (Janota et al., 2021) and CLM-IHS (Cortes et al., 2023). While the specifics of these approaches differ significantly, on a high level both work by maintaining a set $\mathcal{K}$ of cores which represent an undesirable or conflicting substructure of the problem at hand. These cores can but do not have to be unsatisfiable cores that have been extracted with a SAT solver. During search, both Seesaw and CLM-IHS compute solutions that satisfy the current set of cores in a cost-minimal way, and check which of the solutions are feasible for the input instance at hand. Feasible solutions are kept as Pareto-optimal solutions while the infeasible ones are ruled out by extracting new cores.

In more detail, in the context of bi-objective MaxSAT, the Seesaw algorithm (Janota et al., 2021) computes Pareto-optimal solutions of a bi-objective MaxSAT instance (F, O_1, O_2) by maintaining a collection $\mathcal{K}$ of cores that are subsets of $\textsc{lit}(O_1)$. Informally speaking, in the bi-objective setting, every solution τ that improves on O_2 needs to assign at least one literal from each core in $\mathcal{K}$ to 1. The algorithm works iteratively by computing a minimum-cost hitting set $\texttt{hs} \subset \textsc{lit}(O_1)$ of $\mathcal{K}$, i.e., a subset of the literals of O_1 that (i) intersects with each core in $\mathcal{K}$ and (ii) minimizes cost $\textsc{cost}(\texttt{hs})$ defined as the sum of coefficients in O_1 of the literals in $\texttt{hs}$, i.e., $\textsc{cost}(\texttt{hs}) = \textsc{sum-coeff}(O_1|_{\texttt{hs}})$. The hitting set is computed using an integer programming solver. Next, a solution τ is computed with $\tau(l) = 1$ for each $l \in \texttt{hs}$, $\tau(l) = 0$ for each $l \in O_1 \setminus \texttt{hs}$, and $O_2(\tau)$ the smallest possible value for all such solutions, if one exists. In our SAT-based instantiation, this step is instantiated by solution-improving search (often called SAT-UNSAT search) using a SAT solver. Seesaw then extracts a new core that $\texttt{hs}$ does not intersect with. We use the so-called SAT-based core extraction which was shown by Jabs (2022) to outperform the so-called improved strategy core extraction presented in the original publication on Seesaw (Janota et al., 2021). The Pareto-optimal solutions of F are identified by the cost of the hitting set increasing. If the previous hitting set $\texttt{hs}^{\texttt{old}}$ had cost $\textsc{cost}(\texttt{hs}^{\texttt{old}})$ and the new hitting set $\texttt{hs}^{\texttt{new}}$ has strictly greater cost $\textsc{cost}(\texttt{hs}^{\texttt{new}}) > \textsc{cost}(\texttt{hs}^{\texttt{old}})$, the solution τ found with $\texttt{hs}^{\texttt{old}}$ that has the smallest minimum value $O_2(\tau)$ is Pareto-optimal (Janota et al., 2021).

Turning to the CLM-IHS approach (Cortes et al., 2023), when solving a bi-objective MaxSAT problem $I = (F, O_1, O_2)$, a working instance $I^W = (F_{\text{simp}}, O_1, O_2)$ is maintained by CLM-IHS. Initially, I^W is unconstrained, i.e., $F_{\text{simp}} = \emptyset$. CLM-IHS maintains two invariants: (i) every Pareto-optimal solution of I^W that is also a solution of I is a Pareto-optimal solution of I and (ii) every solution of I is a solution of I^W. During each iteration of the search, a representative solution for each element in the non-dominated set of I^W is computed. As implemented by Cortes et al. (2023), such a solution is computed using CLM-LB. The obtained Pareto-optimal solutions of I^W that are not solutions of I are then blocked from subsequent consideration by adding clauses to F_{simp}. The search terminates when all Pareto-optimal solutions of I^W are also solutions of I. From the perspective of IHS, the set of Pareto-optimal solutions of I^W is the hitting set and the clauses blocking the infeasible solutions are the cores that the hitting set is computed over.

SAT-Based Leximax Optimization. A SAT-based approach to multi-objective optimization under leximax optimality called leximaxIST was earlier proposed by Cabral et al. (2022). The approach uses a single SAT solver working on encodings of cardinality constraints over each objective. The cardinality constraints are merged into an encoding for the *maximum* value out of all objectives. This allows leximaxIST to iteratively minimize this maximum with algorithmic ideas inspired by MaxSAT solving.

3. The BiOptSat Approach

With preliminaries in place, we detail BiOptSat, the MaxSAT-based approach to bi-objective optimization developed in this work. We start with an overview of the generic framework (Section 3.1) and then describe six specific instantiations (Section 3.2) based on established MaxSAT algorithms. Furthermore, we detail the use of incremental pseudo-Boolean encodings (Section 3.3) as an important refinement from the practical perspective.

Algorithm 1 BiOptSat: MaxSAT-based bi-objective optimization

Input: A bi-objective MaxSAT instance $I = (F, O_{\mathrm{I}}, O_{\mathrm{D}})$.

Output: Either a single representative for each non-dominated point of I or all Pareto-optimal solutions.

```
1: InitSATsolver($F$)
2: $(\mathrm{res}, \tau) \leftarrow$ isSAT($\emptyset$)                      {Invokes the SAT solver on the formula}
3: if $\mathrm{res} =$ UNSAT then return "no solutions"
4: $b_{\mathrm{I}} \leftarrow -1, b_{\mathrm{D}} \leftarrow \infty$
5: while $\mathrm{res} =$ SAT do
6:     $(b_{\mathrm{I}}, \tau) \leftarrow$ Minimize-Inc($b_{\mathrm{D}}, b_{\mathrm{I}}, O_{\mathrm{I}}(\tau)$)      {Maintains PbCnf($O_{\mathrm{I}}$) (or similar)}
7:     $(b_{\mathrm{D}}, \tau) \leftarrow$ Sol-Impr-Search($b_{\mathrm{I}}, O_{\mathrm{D}}(\tau)$)             {Builds PbCnf($O_{\mathrm{D}}$)}
8:     yield $\tau$                                      {Optionally: yield EnumSols($b_{\mathrm{D}}, b_{\mathrm{I}}$)}
9:     $(\mathrm{res}, \tau) \leftarrow$ isSAT($\{\langle O_{\mathrm{D}} < b_{\mathrm{D}} \rangle\}$)
```

3.1 Overview of BiOptSat

Algorithm 1 describes in pseudocode the BiOptSat framework for computing the Pareto-optimal solutions of a given bi-objective MaxSAT instance $I = (F, O_{\mathrm{I}}, O_{\mathrm{D}})$. BiOptSat is an instantiation of the general lexicographic method (Wassenhove & Gelders, 1980; Marler & Arora, 2004) for multi-objective optimization, utilizing a SAT solver. To find a Pareto-optimal solution, the lexicographic method iteratively minimizes both objectives individually and in order, in our case starting from O_{I}. When minimizing the second objective (O_{D}), an additional constraint requiring the value of O_{I} to be not worse than the value found in the most-recent minimization procedure invocation is enforced. Once the minimum value under these current additional constraints for both objectives is found, the current solution is provably Pareto-optimal. By minimizing each objective in this way separately, the lexicographic method can enumerate all Pareto-optimal solutions in monotonically-increasing order of O_{I}. After finishing an iteration, the remaining Pareto-optimal solutions will have higher values of O_{I} but lower values of O_{D}. The search continues by enforcing a constraint stating that O_{D} must be improved in the next iteration. Search terminates when there are no solutions with lower values of O_{D}.

By enumerating solutions in increasing order of cost wrt. O_{I}, the Pareto-optimal solutions are enumerated in decreasing order for the other objective O_{D}. With this intuition, as formalized in the following observation, we will refer to objective O_{I} as *increasing* and O_{D} as *decreasing*.

Observation 1 (Adapted from (Hartert & Schaus, 2014)). *Sorting the Pareto-optimal solutions of a bi-objective optimization problem under the objectives O_1 and O_2 wrt. increasing values of O_1 amounts to sorting the solutions wrt. decreasing values of O_2, and vice-versa.*

In BiOptSat, the lexicographic method is instantiated in full using a single SAT solver. The SAT solver instantiation is invoked incrementally and thereby preserved (i.e., not reset) during the whole search. BiOptSat maintains the bounds b_{I} and b_{D} on the two objectives O_{I} and O_{D}, respectively. In each iteration, the `Minimize-Inc` procedure sets the value of b_{I} to the smallest value of O_{I} for which there is an undiscovered Pareto-optimal solution τ^o. The value of b_{D} is then set to $O_{\mathrm{D}}(\tau^o)$ by the `Sol-Impr-Search` procedure.

In its default configuration, detailed in pseudocode as Algorithm 1, BiOptSat solves the task of finding a single representative per non-dominated point. The first Pareto-optimal solution discovered by BiOptSat is guaranteed to be lexicographically optimal. For enumerating all Pareto-optimal solutions, the `EnumSols` procedure is extended to enumerate all Pareto-optimal solutions τ^o for which $O_\mathrm{I}(\tau^o) = b_\mathrm{I}$ and $O_\mathrm{D}(\tau^o) = b_\mathrm{D}$.

In detail, given a bi-objective MaxSAT instance $I = (F, O_\mathrm{I}, O_\mathrm{D})$, BiOptSat search (Algorithm 1) starts by initializing a SAT solver with all clauses in F on line 1. Satisfiability of the current set of clauses (i.e., the existence of Pareto-optimal solutions) is checked by invoking the SAT solver on its internal formula without assumptions via the `isSAT`$(\emptyset)$ function (line 2). If res is `UNSAT`, I has no solutions and the algorithm terminates. Otherwise, an assignment τ that satisfies F is obtained. Then, before the main enumeration procedure, the bounds b_I and b_D on O_I and O_D are initialized to -1 and ∞, respectively.

The main search loop (lines 5–9) is iterated over as long as there are Pareto-optimal solutions of I that have not yet been enumerated, i.e., while there is a solution τ for which $O_\mathrm{D}(\tau) < b_\mathrm{D}$. This termination criterion is checked by invoking the SAT solver under the assumptions $\langle O_\mathrm{D} < b_\mathrm{D}\rangle$ on line 9. As we will detail later, the PB constraint defining $\langle O_\mathrm{D} < b_\mathrm{D}\rangle$ is built and maintained by the `Sol-Impr-Search` subroutine.

In the beginning of each main loop iteration, the procedure `Minimize-Inc` is employed to minimize the increasing objective, i.e., to compute the smallest value b_I for which there is a solution τ^m with $O_\mathrm{I}(\tau^m) = b_\mathrm{I}$ and $O_\mathrm{D}(\tau^m) < b_\mathrm{D}$ (line 6). The parameters of the `Minimize-Inc` procedure are the strict upper bound b_D on solutions wrt. the decreasing objective, and b_I as a known lower and $O_\mathrm{I}(\tau)$ as a known upper bound on the minimum increasing objective value. In the following, we will assume that `Minimize-Inc` maintains a way of enforcing $\langle O_\mathrm{I} < b\rangle$ for any b and that BiOptSat and all of its subroutines have access to the literals required to do so; specific implementations of `Minimize-Inc` are detailed later in Section 3.2.

Next, the algorithm employs solution-improving search (Eén & Sörensson, 2006; Le Berre & Parrain, 2010; Bacchus et al., 2021) to minimize the decreasing objective, i.e., to compute the smallest b_D for which there is a solution τ^o with $O_\mathrm{I}(\tau^o) = b_\mathrm{I}$ and $O_\mathrm{D}(\tau^o) = b_\mathrm{D}$ (line 7). The pseudo-boolean constraint PbCnf$(O_\mathrm{D}, O_\mathrm{D}(\tau))$ is built the first time this subroutine is invoked. Building the constraint at this point allows for only building it up to bound $O_\mathrm{D}(\tau)$, which is sufficient since all Pareto-optimal solutions are known to have at most that value for O_D. Solution-improving search—starting from the known upper bound $b = O_\mathrm{D}(\tau)$—iteratively invokes the SAT solver under the assumptions $\{\langle O_\mathrm{D} < b\rangle, \langle O_\mathrm{I} \leq b_\mathrm{I}\rangle\}$ for decreasing values of b until the SAT solver reports unsatisfiability. As soon as unsatisfiability is reached, `Sol-Impr-Search` returns $b_\mathrm{D} = b$ and the latest solution τ for which we have $O_\mathrm{I}(\tau) = b_\mathrm{I}$ and $O_\mathrm{D}(\tau) = b_\mathrm{D}$. At this point, there is no solution of F that dominates τ, and hence τ is returned as Pareto-optimal on line 8. Optionally, to enumerate all solutions τ^o that have cost $(b_\mathrm{I}, b_\mathrm{D})$, the `EnumSols` procedure repeatedly invokes the SAT solver with the assumptions $\{\langle O_\mathrm{D} \leq b_\mathrm{D}\rangle, \langle O_\mathrm{I} \leq b_\mathrm{I}\rangle\}$ and blocking each found solution with a clause until no more solutions are found. Then the unit clause $(\langle O_\mathrm{D} \leq b\rangle)$ is added to the SAT solver. This is possible since the values wrt. $O_\mathrm{D}(\tau)$ monotonically decrease during the search. Compared to adding the output literal as an assumption, adding the unit clauses allows the solver to permanently simplify clauses through unit propagation for its subsequent invocations.

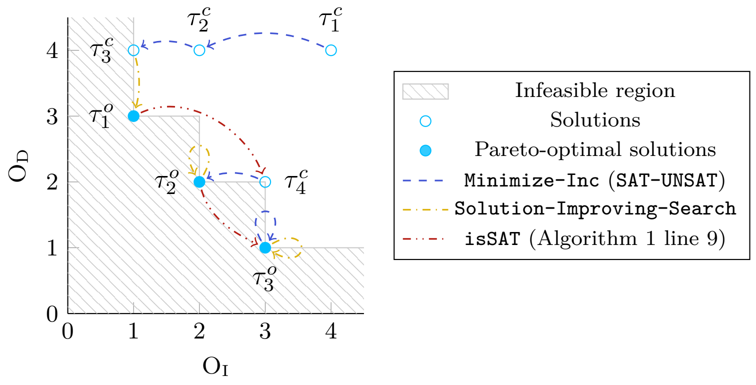

Figure 2: The search progression of the `SAT-UNSAT` instantiation of BiOptSat on the instance from Figure 1.

Example 5. *Consider invoking* BiOptSat *on the formula F and objectives O_I, O_D from Figure 1. The search starts by invoking the SAT solver on F. This call returns a solution, say $\tau_1^c = \{i_1, i_2, i_3, i_4, d_1, d_2, d_3, d_4\}$ (see Figure 2), for which $O_I(\tau_1^c) = O_D(\tau_1^c) = 4$. The first iteration of the main search loop starts with a call to* `Minimize-Inc`*. This returns $b_I = 1$ and, e.g., the solution $\tau_3^c = \{i_2, d_1, d_2, d_3, d_4, \neg i_1, \neg i_3, \neg i_4\}$ for which $O_I(\tau_3^c) = 1$ and $O_D(\tau_3^c) = 4$.* BiOptSat *then proceeds to the* `Sol-Impr-Search` *subroutine that initializes a PB encoding* PbCnf$(O_D, 4)$*. The first call to the SAT solver is made with the assumptions $\mathcal{A} = \{\langle O_I \leq 1\rangle, \langle O_D < 4\rangle\}$. The query is satisfiable. Say that the solver returns the solution $\tau_1^o = \{i_2, d_1, d_3, d_4, \neg i_1, \neg i_3, \neg i_4, \neg d_2\}$. Then, the solver is invoked with the assumptions $\mathcal{A} = \{\langle O_I \leq 1\rangle, \langle O_D < 3\rangle\}$. The query is unsatisfiable, so the procedure returns the Pareto-optimal τ_1^o and $b_D = O_D(\tau_1^o) = 3$. At the end of the iteration, the SAT solver is queried with the assumption $\{\langle O_D < 3\rangle\}$. As the query is satisfiable and the solver returns, e.g., the solution $\tau_4^c = \{i_1, i_2, i_3, d_1, d_2, \neg i_4, \neg d_3, \neg d_4\}$,* BiOptSat *continues similarly for two more iterations, finding, e.g., the Pareto-optimal $\tau_2^o = \{i_1, i_2, d_1, d_2, \neg i_3, \neg i_4, \neg d_3, \neg d_4\}$ with $O_I(\tau_2^o) = O_D(\tau_2^o) = 2$ in the second iteration and τ_3^o with $O_I(\tau_3^o) = 3$ and $O_D(\tau_3^o) = 1$. The algorithm then terminates as the SAT solver queried under the assumption $\{\langle O_D < 1\rangle\}$ reports unsatisfiability.*

3.2 Instantiations for Minimizing the Increasing Objective

We detail six different instantiations of the `Minimize-Inc` procedure for minimizing the increasing objective within BiOptSat. The first four, `SAT-UNSAT`, `UNSAT-SAT`, `MSU3` and `OLL`, are inspired by existing MaxSAT algorithms, while the latter two, `MSHybrid` and `OSHybrid`, switch between two types of MaxSAT-like algorithms with the aim of combining their advantages. We note that, unlike `Minimize-Inc`, the `Sol-Impr-Search` procedure is fixed within BiOptSat to perform solution-improving upper-bounding search. Intuitively, this choice is based on the fact that the constraints enforced on the increasing objective when

Algorithm 2 `SAT-UNSAT` instantiation of `Minimize-Inc`

Input: Most-recent bound b_{D} on O_{D} and known upper bound b on the constrained minimum value of O_{I}.

Output: Solution τ and $b = O_{\mathrm{I}}(\tau)$ minimal under $O_{\mathrm{D}}(\tau) < b_{\mathrm{D}}$.

```
1: build or extend $\textsc{PbCnf}(O_{\mathrm{I}}, b)$ if necessary
2: $(\text{res}, \tau) \leftarrow \texttt{isSAT}(\{\langle O_{\mathrm{I}} < b\rangle, \langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle\})$
3: while res = SAT do
4:     $b \leftarrow O_{\mathrm{I}}(\tau)$
5:     $(\text{res}, \tau) \leftarrow \texttt{isSAT}(\{\langle O_{\mathrm{I}} < b\rangle, \langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle\})$
6: return $(b, \tau)$
```

performing minimization in `Sol-Impr-Search` are iteratively weakened, which in general results in earlier-obtained lower bounds and cores becoming invalid.[2]

3.2.1 `SAT-UNSAT`

The `SAT-UNSAT` instantiation of `Minimize-Inc` performs solution-improving search (Eén & Sörensson, 2006; Le Berre & Parrain, 2010; Bacchus et al., 2021) similarly as employed in `Sol-Impr-Search`. As input, the `SAT-UNSAT` instantiation of `Minimize-Inc` takes the most-recent bound b_{D} on O_{D} and the upper bound $b = O_{\mathrm{I}}(\tau)$ on the constrained minimum value of the increasing objective known from the most recent SAT solver call. Since `SAT-UNSAT` is upper-bounding, it does not make use of the known lower bound. Since the latest query to the SAT solver was made on line 9 (Algorithm 1) with the assumptions $\{\langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle\}$, the solution τ has $O_{\mathrm{D}}(\tau) < b_{\mathrm{D}}$.

`SAT-UNSAT` is outlined in pseudocode as Algorithm 2. The procedure maintains the PB encoding $\textsc{PbCnf}(O_{\mathrm{I}})$ and starts on line 1 by checking whether the current upper bound on $\textsc{PbCnf}(O_{\mathrm{I}})$ is at least b. If this is not the case, the PB encoding is extended with the additional required clauses. Then the SAT solver is iteratively invoked with the assumptions $\{\langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle, \langle O_{\mathrm{I}} < b\rangle\}$ for decreasing values of b (line 5). The procedure terminates when the SAT solver reports unsatisfiability. At this point (on line 6) the value of b and the solution obtained from the last SAT solver call which reported satisfiability are returned as b_{I} and τ.

Example 6. *Consider again the invocation of* BiOptSat *from Example 5. We detail the invocations of* `Minimize-Inc` *instantiated as* `SAT-UNSAT`*. Figure 2 illustrates the search progression for this configuration. In the first iteration,* `SAT-UNSAT` *is invoked with $b_D = \infty$ and $b = O_I(\tau_1^c) = 4$. At this point, the PB encoding over O_I has not been built, so the procedure starts by adding* $\textsc{PbCnf}(O_I, 4)$ *to the solver. The first call to the SAT solver is made with the assumptions $\{\langle O_I < 4\rangle\}$ (as $b_D = \infty$ no assumptions on O_D are used). Assume that the solver returns the solution $\tau_2^c = \{i_1, i_2, d_1, d_2, d_3, d_4, \neg i_3, \neg i_4\}$. In the next iteration, the set of assumptions is $\{\langle O_I < 2\rangle\}$. The solver returns, e.g., the solution $\tau_3^c = \{i_2, d_1, d_2, d_3, d_4, \neg i_1, \neg i_3, \neg i_4\}$. The final (unsatisfiable) SAT solver call is then made under the assumptions $\{\langle O_I < 1\rangle\}$, resulting in the procedure returning $b_I = 1$ and τ_3^c.*

2. Investigating ways of making incremental use of other types of MaxSAT search approaches for the decreasing objective is left for further work.

Algorithm 3 `UNSAT-SAT` instantiation of `Minimize-Inc`

Input: Most recent bounds b_{D} on O_{D} and b_{I} on O_{I}.
Output: Solution τ and $b = O_{\mathrm{I}}(\tau)$ minimal under $O_{\mathrm{D}}(\tau) < b_{\mathrm{D}}$.

```
1: $b \leftarrow b_{\mathrm{I}}$
2: build or extend $\textsc{PbCnf}(O_{\mathrm{I}}, b+1)$
3: $(\mathrm{res}, \tau) \leftarrow \texttt{isSAT}(\{\langle O_{\mathrm{I}} \leq b+1\rangle, \langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle\})$
4: while res = UNSAT do
5:     $b \leftarrow b+1$
6:     extend $\textsc{PbCnf}(O_{\mathrm{I}}, b+1)$
7:     $(\mathrm{res}, \tau) \leftarrow \texttt{isSAT}(\{\langle O_{\mathrm{I}} \leq b+1\rangle, \langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle\})$
8: return $(b+1, \tau)$
```

*In the second and third iterations, the first SAT solver call is made by **`SAT-UNSAT`** (with assumptions $\{\langle O_D < 3\rangle, \langle O_I < 3\rangle\}$ and $\{\langle O_D < 2\rangle, \langle O_I < 3\rangle\}$ respectively) returns **`UNSAT`**, resulting in **`SAT-UNSAT`** immediately returning the current solution.*

3.2.2 `UNSAT-SAT`

`UNSAT-SAT` takes an analogous approach to `SAT-UNSAT` search but searches for the constrained minimum value of O_{I} by lower-bounding instead of upper-bounding. The procedure takes as input the most recent bounds b_{D} on O_{D} and b_{I} on O_{I} as a lower bound on O_{I} subject to $O_{\mathrm{D}} < b_{\mathrm{D}}$.

The `UNSAT-SAT` instantiation of `Minimize-Inc` is outlined in pseudocode as Algorithm 3. `UNSAT-SAT` maintains a PB encoding $\textsc{PbCnf}(O_{\mathrm{I}})$ for enforcing a bound b on O_{I}. On line 1, b is set to the known lower bound b_{I} and the SAT solver is then iteratively invoked on line 7 under the assumptions $\{\langle O_{\mathrm{I}} \leq b+1\rangle, \langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle\}$. If the SAT solver reports unsatisfiability, the bound b is increased by 1 and the SAT solver is invoked again. The search ends once the SAT solver reports satisfiability. At this time the solution and the updated bound are returned on line 8. Note that since the value of b is monotonically increasing, it suffices to build the pseudo-boolean encoding $\textsc{PbCnf}(O_{\mathrm{I}})$ up to the bound $b+1$ on each iteration (line 6). This way the SAT solver is always invoked without unnecessary clauses in terms of the PB encoding.

Example 7. *Consider again the invocation of* BiOptSat *from Example 5. We detail the invocations of **`Minimize-Inc`** instantiated as **`UNSAT-SAT`**. In the first iteration, **`UNSAT-SAT`** is invoked with $b_D = \infty$ and $b_I = -1$. At this point, the PB encoding over O_I has not yet been built, so the procedure starts by initializing* $\textsc{PbCnf}(O_I, 0)$ *and invokes the SAT solver with the assumptions $\{\langle O_I \leq 0\rangle\}$. The query is unsatisfiable, so the PB encoding is extended to* $\textsc{PbCnf}(O_I, 1)$ *and the SAT solver is invoked with the assumptions $\{\langle O_I \leq 1\rangle\}$. The SAT solver now reports satisfiability, returning e.g. the solution $\tau_3^c = \{i_2, d_1, d_2, d_3, d_4, \neg i_1, \neg i_3, \neg i_4\}$. Then **`UNSAT-SAT`** returns $b_I = 1$ and τ_3^c. The second and third invocations extend the PB encoding further and return τ_2^o and τ_3^o respectively.*

Algorithm 4 `MSU3` instantiation of `Minimize-Inc`

Input: Most recent bounds b_{D} on O_{D} and b_{I} on O_{I}.
Output: Solution τ and $b = O_{\mathrm{I}}(\tau)$ minimal under $O_{\mathrm{D}}(\tau) < b_{\mathrm{D}}$.

```
1: $b \leftarrow \max\{b_{\mathrm{I}}, 0\}$
2: $(\mathrm{res}, \tau, \kappa) \leftarrow \texttt{isSAT}(\langle O_{\mathrm{I}}|_{\texttt{Act}} \leq b\rangle \cup \langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle \cup \{\neg l \mid l \in \textsc{lit}(O_{\mathrm{I}}) \setminus \texttt{Act}\})$
3: while $\mathrm{res} = \texttt{UNSAT}$ do
4:     $b \leftarrow b + 1$
5:     $\kappa \leftarrow \kappa \cap \textsc{lit}(O_{\mathrm{I}})$
6:     $\texttt{Act} \leftarrow \texttt{Act} \cup \kappa$
7:     build or extend $\textsc{PbCnf}(O_{\mathrm{I}}|_{\texttt{Act}}, b)$
8:     $(\mathrm{res}, \tau, \kappa) \leftarrow \texttt{isSAT}(\{\langle O_{\mathrm{I}}|_{\texttt{Act}} \leq b\rangle, \langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle\} \cup \{\neg l \mid l \in \textsc{lit}(O_{\mathrm{I}}) \setminus \texttt{Act}\})$
9: return $(b, \tau)$
```

3.2.3 `MSU3`

The `MSU3` instantiation of `Minimize-Inc` is a core-guided approach inspired by the MSU3 single-objective MaxSAT algorithm (Marques-Silva & Planes, 2007). As input `MSU3` takes the most recent bounds b_{D} on O_{D} and b_{I} on O_{I} as a lower bound on the constrained minimum value of O_{I}. `MSU3` maintains a set $\texttt{Act} \subset \textsc{lit}(O_{\mathrm{I}})$ of *active* objective literals and a PB encoding $\textsc{PbCnf}(O_{\mathrm{I}}|_{\texttt{Act}})$ built over the restriction of the increasing objective to the active literals. Initially all literals of O_{I} are inactive, i.e., $\texttt{Act} = \emptyset$. An inactive literal $l \in \textsc{lit}(O_{\mathrm{I}}) \setminus \texttt{Act}$ is assumed to the value 0 in every invocation of the SAT solver until l occurs in the core returned by the SAT solver. Algorithm 4 illustrates the search performed by `MSU3`. The algorithm starts from the value $b = b_{\mathrm{I}}$ computed in the previous iteration and invokes the SAT solver with the assumptions $\mathcal{A} = \{\langle \texttt{Act} \leq b\rangle, \langle O_{\mathrm{D}} < b_{\mathrm{D}}\rangle\} \cup \{\neg l \mid l \in \textsc{lit}(O_{\mathrm{I}}) \setminus \texttt{Act}\}$ on line 2. If the query is unsatisfiable, the SAT solver returns a core $\kappa \subset \{\neg l \mid l \in \mathcal{A}\}$. Next, the bound b is increased by one, the inactive literals in κ become active by adding them to `Act` and the PB encoding $\textsc{PbCnf}(O|_{\texttt{Act}})$ is extended (lines 4–7). The procedure continues until the query is satisfiable, at which point a solution τ with $O_{\mathrm{I}}(\tau) = b$ and $O_{\mathrm{D}}(\tau) < b_{\mathrm{D}}$ is found. At this point the value b is the minimum value $O_{\mathrm{I}}(\tau)$ for any solution τ subject to $O_{\mathrm{D}}(\tau) < b_{\mathrm{D}}$. This is because the value of b is increased monotonically, and the SAT solver reported unsatisfiability up to the second-to-last iteration. Note that the set `Act` of active literals is maintained between the invocations of `MSU3`.

Note that when using `MSU3`, $O_{\mathrm{I}}(\tau) \leq b_{\mathrm{I}}$ cannot be enforced in the other procedures within BiOptSat using a single literal. Instead, the algorithm uses a set of assumptions $\{\langle O_{\mathrm{I}}|_{\texttt{Act}} \leq b_{\mathrm{I}}\rangle\} \cup \{\neg l \mid l \in \textsc{lit}(O_{\mathrm{I}}) \setminus \texttt{Act}\}$ that restrict the cost of active literals set to 1 to b_{I} and fix the value of each inactive literal to 0. The following establishes that using these assumptions to enforce $O_{\mathrm{I}}(\tau) \leq b_{\mathrm{I}}$ does not remove any Pareto-optimal solutions from the search.

Proposition 1. *Invoke* BiOptSat *with `Minimize-Inc` instantiated as `MSU3` on an instance (F, O_I, O_D). Let b_D and b_I be the values returned by `MSU3` and `Sol-Impr-Search` on the i:th iteration of the search, respectively, and `Act` the set of active literals after the i:th invocation of `MSU3`. Consider a Pareto-optimal solution τ^o of the instance for which $O_I(\tau^o) = b_I$. Then $\tau^o(l) = 0$ for all $l \in \textsc{lit}(O_I) \setminus$ `Act`.*

Proof. (Sketch) Since b_{I} was returned by `MSU3`, we know that there is a Pareto-optimal τ^o for which $O_{\text{I}}(\tau^o) = b_{\text{I}}$ and $O_{\text{D}}(\tau^o) < b_{\text{D}}$. By the properties of cores, *any* solution τ^s of F for which $O_{\text{D}}(\tau^s) < b_{\text{D}}$ assigns literals in `Act` with at least cost b_{I} to 1. Thus, any τ^n that assigns $\tau^n(l) = 1$ for an inactive literal $l \in \text{LIT}(O_{\text{I}}) \setminus \texttt{Act}$ will have $O_{\text{I}}(\tau^n) > b_{\text{I}}$. □

Example 8. *Consider the invocation of* BiOptSat *from Example 5. We detail the invocations of* ***`Minimize-Inc`*** *instantiated as* ***`MSU3`****. In the first iteration of* BiOptSat*,* ***`MSU3`*** *is invoked with $b_D = \infty$ and $b_I = -1$. Initially, the set $\texttt{Act} = \emptyset$ of active literals is empty, so the first call to the SAT solver is made with the assumptions $\mathcal{A} = \{\neg i_1, \neg i_2, \neg i_3, \neg i_4\}$. The query is unsatisfiable. Assume that the SAT solver returns $\kappa = \{i_1, i_2\}$. The literals in κ are marked as active and the PB encoding* $\text{PbCnf}(O_I|_{\texttt{Act}}, 1)$ *is initialized. The SAT solver is then invoked with the assumptions $\mathcal{A} = \{\neg i_3, \neg i_4, \langle O_I|_{\texttt{Act}} \leq 1\rangle\}$. The query is satisfiable, so* ***`MSU3`*** *returns, e.g., $b_I = 1$ and the solution $\tau_3^c = \{i_2, d_1, d_2, d_3, d_4, \neg i_1, \neg i_3, \neg i_4\}$. In the next iteration of* BiOptSat*,* ***`MSU3`*** *is invoked with $b_D = 3$ and $b_I = 1$. The set $\texttt{Act} = \{i_1, i_2\}$ is kept from the previous iteration and the SAT solver is invoked with $\mathcal{A} = \{\langle O_I|_{\texttt{Act}} \leq 1\rangle, \langle O_D < 3\rangle, \neg i_3, \neg i_4\}$. The query is unsatisfiable. Assume that the core returned by the SAT solver is $\kappa = \{\langle O_D < 3\rangle, \langle O_I|_{\texttt{Act}} \leq 1\rangle, i_3, i_4\}$. The literals i_3 and i_4 are marked as active, and the PB encoding is extended to* $\text{PbCnf}(O_I|_{\texttt{Act}}, 2)$*. The next SAT solver call with assumptions $\mathcal{A} = \{\langle O_I|_{\texttt{Act}} \leq 2\rangle, \langle O_D < 2\rangle\}$ is satisfiable and* ***`MSU3`*** *returns $b_I = 2$ and, e.g., $\tau_2^o = \{i_1, i_2, d_1, d_2, \neg i_3, \neg i_4, \neg d_3, \neg d_4\}$. In the last iteration,* ***`MSU3`*** *is invoked with $b_D = 2$ and $b_I = 2$. At this point $\texttt{Act} = \text{LIT}(O_I)$ so the core found in the first SAT solver invocation will only result in the PB encoding being extended to* $\text{PbCnf}(O_I|_{\texttt{Act}}, 3)$*. The next SAT solver invocation reports satisfiability, and the procedure returns $b_I = 3$ and, e.g., $\tau_3^o = \{i_1, i_3, i_4, d_2, \neg i_2, \neg d_1, \neg d_3, \neg d_4\}$.*

3.2.4 `OLL`

The core-guided single-objective MaxSAT algorithm OLL (Andres, Kaufmann, Matheis, & Schaub, 2012; Morgado et al., 2014; Ignatiev et al., 2019), applied in the context of BiOptSat, iteratively reformulates the objective O_{I} based on extracted cores in a way that allows a minimum amount of cost to be incurred in future iterations. In more detail, `OLL` maintains a reformulated objective $O_{\text{I-ref}}$, initialized to O_{I}. Each invocation of the SAT solver is made under a set $\mathcal{A} = \{\neg l \mid l \in \text{LIT}(O_{\text{I-ref}})\} \cup \{\langle O_{\text{D}} < b_{\text{D}}\rangle\}$ of assumptions consisting of the negation of the literals in $O_{\text{I-ref}}$ and a bound on the decreasing objective. If a core κ is obtained and $\neg\langle O_{\text{D}} < b_{\text{D}}\rangle$ is part of the core, $\neg\langle O_{\text{D}} < b_{\text{D}}\rangle$ is removed from the core so that $\kappa \subset \text{LIT}(O_{\text{I-ref}})$. A cardinality constraint $\text{CardCnf}(\kappa)$ is built over the literals in κ and $O_{\text{I-ref}}$ is updated by replacing the terms $\sum_{l \in \kappa} \text{COEFF}(O_{\text{I-ref}}, l) \cdot l$ with $\sum_{l \in \kappa}((\text{COEFF}(O_{\text{I-ref}}, l) - w_\kappa^{\min}) \cdot l) + \sum_{i=2}^{|\kappa|} w_\kappa^{\min} \cdot \neg\langle \kappa < i\rangle$, where $w_\kappa^{\min} = \min\{\text{COEFF}(O_{\text{I-ref}}, l) \mid l \in \kappa\}$. Conceptually, this reformulation step (i) decreases the coefficient in $O_{\text{I-ref}}$ for each $l \in \kappa$ by $w_\kappa^{\min}$, and (ii) adds $|\kappa| - 1$ new literals that bound the number of literals in κ assigned to 1 to the objective with coefficient $w_\kappa^{\min}$. Note that the coefficient of at least one literal l in κ is lowered to 0 in the reformulation, thus removing such l from the objective and the assumptions in subsequent iterations. This essentially allows the SAT solver to incur cost on l by assigning it to 1. Furthermore, adding $\neg\langle \kappa < 2\rangle$ to the objective ensures that assigning more than one literal to 1 incurs more cost. If the SAT solver query is on the other hand satisfiable, the thereby obtained solution τ incurs no cost in $O_{\text{I-ref}}$ and thus

Algorithm 5 `MSHybrid` instantiation of `Minimize-Inc`

Input: Most recent bound b_{D} on O_{D}, known upper and lower bounds b and b_{I} on constrained minimum of O_{I}.

Output: Solution τ and $b = O_{\text{I}}(\tau)$ minimal under $O_{\text{D}}(\tau) < b_{\text{D}}$.

```
1: if $\textsc{sum-coeff}(O_{\text{I}}|_{\texttt{Act}}) < \texttt{thr} \cdot \textsc{sum-coeff}(O_{\text{I}})$ then
2:     $(b_{\text{I}}, \tau) \leftarrow \texttt{MSU3}(b_{\text{D}}, b_{\text{I}})$          {Immediately terminates once threshold is met}
3: if $\textsc{sum-coeff}(O_{\text{I}}|_{\texttt{Act}}) \geq \texttt{thr} \cdot \textsc{sum-coeff}(O_{\text{I}})$ then
4:     fully build or extend $\textsc{PbCnf}(O_{\text{I}}, b)$ if necessary
5:     $(b_{\text{I}}, \tau) \leftarrow \texttt{SAT-UNSAT}(b_{\text{D}}, b)$
6: return $(b_{\text{I}}, \tau)$
```

a minimum amount of cost in O_{I}, and hence at this stage τ is returned as a solution with minimum cost to O_{I} and $O_{\text{D}}(\tau) < b_{\text{D}}$.

Example 9. *Let $O_I = 3 \cdot i_1 + 2 \cdot i_2 + i_3$ be the increasing objective of a bi-objective MaxSAT instance. Consider invoking `Minimize-Inc` instantiated with `OLL` when solving such an instance with* BiOptSat. *Assume that the first core discovered within `OLL` is $\kappa = \{i_1, i_2\}$. `OLL` then adds* $\textsc{CardCnf}(\kappa)$ *and the new reformulated objective is $O_{I\text{-}ref} = i_1 + i_3 + 2 \cdot \neg\langle \kappa < 2 \rangle$.*

Similarly to `MSU3`, when instantiating `Minimize-Inc` with `OLL`, there is no single literal that can be used to enforce $O_{\text{I}}(\tau) \leq b_{\text{I}}$ in other parts of the BiOptSat algorithm. Instead, enforcing all literals in the reformulated objective to 0 will also bound O_{I} to at most b_{I}. This follows from the correctness of OLL for single-objective MaxSAT and an argument very similar to the one we made in Proposition 1: when `OLL` finds a constrained minimum b_{I}, any Pareto-optimal solution τ^o for which $O_{\text{I}}(\tau^o) = b_{\text{I}}$ has $\tau^o(l) = 0$ for all $l \in \textsc{lit}(O_{\text{I-ref}})$.

3.2.5 Hybrid Approaches `MSHybrid` and `OSHybrid`

The final two instantiations of `Minimize-Inc` we consider are hybrids that alternate between the core-guided approaches and `SAT-UNSAT`. A similar approach of combining core-guided and solution-improving search is known in single-objective MaxSAT solving as core-boosted linear search (Berg, Demirovic, & Stuckey, 2019).

The first one, `MSHybrid`, combines `MSU3` and `SAT-UNSAT`. The intuition underlying `MSHybrid` is that if `MSU3` reaches the stage where all literals of the objective are active, its search will essentially degenerate to `UNSAT-SAT`, i.e., a lower-bounding search where the bound on the PB encoding $\textsc{PbCnf}(O_{\text{I}})$ is increased by one every iteration until the SAT query is satisfiable.[3] Intuitively, `MSHybrid` aims to avoid this "degeneration" by switching to `SAT-UNSAT` instead when a considerable number of objective literals have become active.

With this intuition, we propose `MSHybrid` as a hybrid instantiation that starts with `MSU3` search and switches to `SAT-UNSAT` as soon as a certain percentage of the literals in O_{I} have become active. Outlined in pseudocode as Algorithm 5, `MSHybrid` takes the last bound b_{D}

3. Note that if a problem instance has literals in O_{I} that never appear in any cores, i.e., that are not constrained by F, these literals will never become active. As a result, `MSU3` will behave similarly as `UNSAT-SAT` on such instances, even before all literals are active.

on O_D as well as the known upper $b = O_\text{I}(\tau)$ and lower bounds b_I on the minimum of O_I as input Additionally, `MSHybrid` employs a configuration parameter `thr` that defines at which percentage of the objective O_I being active the algorithm switches from `MSU3` to `SAT-UNSAT`. Initially `MSHybrid` executes `MSU3` on line 2. If `MSU3` finds the minimum without meeting the threshold condition $\textsc{sum-coeff}(O_\text{I}|_{\texttt{Act}}) \geq \texttt{thr} \cdot \textsc{sum-coeff}(O_\text{I})$, the found minimum b_I and a corresponding solution τ are returned on line 6. In case the threshold condition is met during the execution of `MSU3`, it is immediately terminated. On line 4 the PB encoding $\textsc{PbCnf}(O_\text{I}, k)$ is fully built from the existing $\textsc{PbCnf}(O_\text{I}|_{\texttt{Act}})$ and `SAT-UNSAT` is invoked on line 5. From then on, every call to `MSHybrid` executes `SAT-UNSAT` on line 5.

Example 10. *Consider the invocation of* BiOptSat *from Example 5. We detail the invocations of* `Minimize-Inc` *instantiated as* `MSHybrid`*. Since* `MSHybrid` *starts out as* `MSU3`*, the first invocation follows the description in Example 8. Assume that* `MSHybrid` *is configured to switch as soon as 70% of O_I is active* ($\texttt{thr} = 0.7$)*. Since we have* $\texttt{Act} = \{i_1, i_2\}$ *after the first iteration of* BiOptSat*, less than 70% of the literals in O_I are active, and so* `MSU3` *is again employed in the second invocation of* `MSHybrid`*. As soon as i_3 and i_4 become active, with the first core in the second invocation of* `MSU3`*, the* `MSU3` *subroutine is terminated since the threshold for switching to* `SAT-UNSAT` *is reached. Since all literals in O_I are already active in this example and thereby included in* $\textsc{PbCnf}(O_I|_{\texttt{Act}})$*, the PB encoding does not need to be extended.* `SAT-UNSAT` *can directly be invoked as in the second iteration outlined in Example 6. In the third iteration of* BiOptSat*,* `MSHybrid` *will directly invoke* `SAT-UNSAT`*, which proceeds as described in Example 6.*

The second hybrid instantiation we consider is `OSHybrid`. Analogous to `MSHybrid`, which first employs `MSU3`, `OSHybrid` first employs `OLL`, and later invokes `SAT-UNSAT` once a certain portion of the literals in the original objective O_I have been extracted in a core. More precisely, in the context of `OLL`, we say that a literal $l \in \textsc{lit}(O_\text{I}) \setminus \textsc{lit}(O_\text{I-ref})$ is marked active once its coefficient in $O_\text{I-ref}$ drops to 0 and gets removed. For both `MSU3` and `OLL`, the active literals are the ones that are not enforced to 0 by assumptions. `OSHybrid` operates similarly to `MSHybrid`. When the fraction of active literals in $\textsc{lit}(O_\text{I})$ increases above the user-defined threshold `thr`, the following steps are taken: (i) `OLL` is terminated, (ii) a PB encoding $\textsc{PbCnf}(O_\text{I-ref})$ is built from the cardinality constraints constructed in `OLL`, and (iii) `SAT-UNSAT` is invoked using the previously-built $\textsc{PbCnf}(O_\text{I-ref})$. However, note that `MSHybrid` and `OSHybrid` are conceptually different in that while `SAT-UNSAT` is invoked in `MSHybrid` on the original objective O_I, `SAT-UNSAT` is in `OSHybrid` instead invoked on the reformulated objective $O_\text{I-ref}$.

3.3 Incremental Pseudo-Boolean Encodings in BiOptSat

At this stage, it is apparent that, regardless of how `Minimize-Inc` is instantiated, BiOptSat makes extensive use of encodings of pseudo-Boolean constraints. Due to this, the choice of how and when the pseudo-Boolean constraints are built can have a significant impact on the performance of BiOptSat in practice. In the implementation reported on in the preliminary conference version of this work (Jabs et al., 2022), the PB constraint $\textsc{PbCnf}(O)$ over an objective O was built by expanding each term $c \cdot l$ in O with $c > 1$ into c terms $l + l + \ldots + l$. Conceptually, this reduced the PB constraint into a (larger) cardinality constraint which was encoded with a CNF encoding for cardinality constraints.

Here we detail an improvement on this obtained via harnessing incremental PB encodings, i.e., encodings that are incrementally built only to the extent necessary for each SAT solver call within BiOptSat.

More formally, given an objective O, two subsets $L \subset L' \subset \textsc{lit}(O)$ and two bounds $b_1 < b_2$, an encoding of a PB constraint is incremental if (a) $\textsc{PbCnf}(O|_L, b) \subset \textsc{PbCnf}(O|_{L'}, b)$ and (b) $\textsc{PbCnf}(O, b_1) \subset \textsc{PbCnf}(O, b_2)$. The use of incremental PB encodings allows for invoking a SAT solver multiple times under different bounds on the PB constraint and allowing the solver to retain its state between the invocations. This can have a positive impact on the overall runtime of the SAT solver and hence also the empirical runtime performance of BiOptSat.

We note that the use of incremental cardinality encodings is well-established (Martins, Joshi, Manquinho, & Lynce, 2014a). However, the use of incremental PB constraints as a generalization of cardinality constraints is less studied. Here we focus on the totalizer encoding for cardinality constraints (Bailleux & Boufkhad, 2003) and the generalized totalizer encoding (Joshi et al., 2015) for the PB constraints employed in our current implementation of BiOptSat. Both of these encodings can be viewed as binary trees. Each leaf of the tree is associated with a distinct input literal of the encoding. The root is associated with the set of output literals. An internal node N is associated with a set of auxiliary variables that—informally speaking—count the number (in the case of the totalizer) or sum of coefficients (in the case of the generalized totalizer) of the literals associated with the leaves of the subtree rooted at N that are assigned to 1.

To understand the challenge in employing the generalized totalizer incrementally, we need a more precise definition of the output literals. For a CNF formula $\textsc{CardCnf}(L)$ over a set L of literals produced by the totalizer encoding, and a bound k, first note that the definition $\langle L < k \rangle \to \sum_{l \in L} l < k$ of the output literals discussed in Section 2.3 is equivalent to $\sum_{l \in L} l \geq k \to \neg\langle L < k \rangle$. However, only a solution τ that assigns a subset of the literals of L containing *exactly* k elements to 1 is guaranteed to assign $\langle L < k \rangle$ to 0. For cardinality constraints, this also implies that any solution τ^b that assigns a subset $L^b \subset L$ containing more than k literals to 1 will also assign $\langle L < k \rangle$ to 0. This is due to the fact that any such L^b is guaranteed to contain another subset with exactly k literals.

However, the same reasoning does not hold for general PB constraints. Given an objective O and a bound k, we again have that any solution τ that assigns a subset of the literals in $\textsc{lit}(O)$ whose sum of coefficients add up to exactly k will assign $\langle O < k \rangle$ to 0. However, in contrast to the case of cardinality constraints, as illustrated in the following example, this does not imply that all solutions that satisfy a subset of literals with a larger sum of coefficients would necessarily assign $\langle O < k \rangle$ to 0.

Example 11. *Consider the objective $O = b_1 + b_2 + 2 \cdot b_3 + 3 \cdot b_4$ and the CNF formula $\textsc{PbCnf}(O)$ resulting from the generalized totalizer encoding. Any solution that assigns a subset of $\textsc{lit}(O)$ with sum-of-weights exactly 2 to 1 will also assign the literal $\langle O < 2 \rangle$ to 0. These include the solutions $\{b_1, b_2, \neg b_3, \neg b_4\}$, $\{b_1, b_2, b_3, \neg b_4\}$ and $\{\neg b_1, \neg b_2, b_3, \neg b_4\}$. However, the solution $\tau = \{\neg b_1, \neg b_2, \neg b_3, b_4\}$ need not assign $\langle O < 2 \rangle$ to 0 even though $O(\tau) = 4 > 2$. Specifically, both $\tau \cup \{\langle O < 2 \rangle\}$ and $\tau \cup \{\neg\langle O < 2 \rangle\}$ can be extended to solutions of $\textsc{PbCnf}(O)$.*

To the best of our knowledge, previous work (Joshi et al., 2015) making use of the generalized totalizer focuses on a static setting where a single bound B is to be enforced on the PB expression O. In such a more restrictive setting the issue illustrated in Example 11 is circumvented by slightly altering the encoding by adding clauses that explicitly enforce $O \geq k \rightarrow \neg\langle O < B\rangle$ for all $k \geq B$, and has been shown to be beneficial in practice (Joshi et al., 2015). However, in BiOptSat we need the ability to increase the bound on the PB constraint dynamically. Thus, for the purposes of potential runtime improvements for BiOptSat, we propose an incremental extension of the generalized totalizer encoding[4]. For this, we enforce a bound k on the coefficients of literals set to 1 not by the single literal $\langle O < k\rangle$, but instead by the set of literals $\{\langle O < k\rangle, \ldots, \langle O < k + c^{\max}\rangle\}$, where $c^{\max}$ is the largest coefficient among the coefficients of literals in O. The intuition here is that any subset of the literals in $\textsc{lit}(O)$ the coefficients of which sum up to more than k will contain a subset of literals with sum of weights equal to a number between k and $k + c^{\max}$.

In terms of specific instantiations of `Minimize-Inc`, for the `OSHybrid` variant, the fact that the objective reformulation steps performed during `OLL` are instantiated with totalizers is made use of when switching from the core-guided approach to `SAT-UNSAT`. When `OSHybrid` terminates its core-guided phase, all cardinality constraints resulting from objective reformulation are extended with the needed outputs that might not have been introduced yet due to the use of incremental totalizers. When subsequently building a generalized totalizer over the reformulated objective, the sets of outputs of totalizers are seen as internal nodes rather than separate leaves. This saves on the number of clauses that need to be added and—informally speaking—avoids building a sorting structure in the generalized totalizer over the totalizer outputs that are already known to represent a sorted unary number.

4. Empirical Evaluation

We turn to an empirical evaluation of our implementation of the BiOptSat approach. All experiments reported on in this section were run on 2.60-GHz Intel Xeon E5-2670 machines with 64-GB RAM in RHEL under a 1.5-hour per-instance time and 16-GB memory limit.

4.1 Implementation and Competing Approaches

Our implementations of BiOptSat and the competing P-minimal and Seesaw approaches are available in open source at `https://bitbucket.org/coreo-group/bioptsat/`. All of our implementations use the state-of-the-art SAT solver CaDiCaL 1.5.2 (Biere, Fazekas, Fleury, & Heisinger, 2020). We implemented all instantiations of BiOptSat described in Section 3 in C++. Our implementations of MSU3 and OLL are inspired by their single-objective MaxSAT implementations in Open-WBO v2.1 (Martins, Manquinho, & Lynce, 2014c). The other instantiations were implemented fully from scratch. Our implementations of the core-guided instantiations `MSU3` and `OLL` as well as the hybrid instantiations `MSHybrid` and `OSHybrid` make use of refinements commonly used in core-guided MaxSAT

4. We note that an incremental version of the sequential weight counter encoding (Hölldobler, Manthey, & Steinke, 2012) has been proposed earlier (Martins, Joshi, Manquinho, & Lynce, 2014b). Our motivation of an incremental version of the generalized totalizer encoding comes from the fact that the generalized totalizer encoding has been shown to outperform the sequential weight counter encoding (Joshi et al., 2015).

solving. Specifically, in addition to the use of incremental cardinality and pseudo-Boolean encodings discussed in Section 3.3, we use a heuristic form of core minimization known as core trimming (Ignatiev et al., 2019) during which we iteratively invoke the SAT solver with assumptions corresponding to the extracted core in the hopes of it computing a smaller core[5]. In `OLL` and `OSHybrid`, we also employ core exhaustion (Ansótegui, Bonet, Gabàs, & Levy, 2013; Ignatiev et al., 2019) and weight-aware core extraction (Berg & Järvisalo, 2017) as additional optimizations geared toward extracting cores that result in larger increases in the lower bound. The hybrids `MSHybrid` and `OSHybrid` are configured to switch between `MSU3`/`OLL` and `SAT-UNSAT` once 70% of O_I is active. This particular value was chosen based on preliminary experiments, with the aim of preventing the `MSU3` search phase from "degenerating" into `UNSAT-SAT`-style search (recall Section 3.2.5). In the preliminary experiments, the exact choice of the threshold value within reasonable scale did not appear to have a significant impact on performance.

We compare the empirical performance of our implementation of BiOptSat to all of the related approaches discussed in Section 2.6. For a fair comparison, we implemented both P-minimal and Seesaw in C++ under the same code based as BiOptSat. For Seesaw, we use CPLEX v20.10 for the hitting set computations in the experiments. As for Pareto-MCS and CLM-LB/CLM-IHS, we use the implementations available at `https://gitlab.ow2.org/sat4j/moco` and `https://gitlab.inesc-id.pt/u001810/moco`, respectively. Our main focus in the empirical evaluation is on the tasks of computing the non-dominated set and enumerating all Pareto-optimal solutions. However, we also include in the comparison the recently-proposed SAT-based leximax solver leximaxIST (Cabral et al., 2022) downloaded from `https://github.com/miguelcabral/leximaxIST`, although it should be noted that leximaxIST is only applicable to the arguably easier task of computing a leximax-optimal solution.

4.2 Benchmarks

We used four bi-objective benchmark domains in the empirical evaluation. In addition to three specific bi-objective problem domains (learning interpretable decision rules from data, bi-objective set covering, and package upgradeability), we reverse-engineered bi-objective optimization problem instances from single-objective MaxSAT instances originally submitted to the MaxSAT Evaluation (Bacchus, Järvisalo, & Martins, 2019) between 2006 and 2019 which actually encode a linear combination of two separate objectives. Beyond the following descriptions of the benchmarks, summary statistics for the number of variables and clauses as well as the objective weights are provided in Appendix A. The benchmarks are available at `https://bitbucket.org/coreo-group/bioptsat/src/master/jair24`.

4.2.1 Learning Interpretable Decision Rules

A variety of (Max)SAT-based approaches have recently been proposed for the intrinsically bi-objective problem of learning interpretable classifiers from data (Jin & Sendhoff, 2008; Malioutov & Meel, 2018; Narodytska et al., 2018; Ignatiev et al., 2018; Hu et al., 2020; Yu et al., 2021; Ignatiev et al., 2021; Ghosh et al., 2022). The two (conflicting) objectives are

5. In preliminary tests, we found that a complete core minimization procedure that iteratively tries to remove each variable in the core would be too expensive.

the classification error and the classifier size, the latter as a proxy for the interpretability of the classifier. As a representative of these types of bi-objective optimization problems, we consider learning of interpretable decision rules (LIDR) (Malioutov & Meel, 2018) where the classifier is a CNF formula over Boolean features. Previous work (Malioutov & Meel, 2018) considered the problem of computing individual non-dominated points of the size and error objectives by reducing the problem into a single-objective optimization problem with a method similar to the weighted sum method (recall Section 2.5). While the approach can be used for computing individual Pareto-optimal solutions, it offers no guarantees on finding all non-dominated points. Here we instead consider the more general problem of computing the entire non-dominated set in a structured manner, treating the two objectives separately, which gives rise to proper bi-objective MaxSAT instances.

In more detail, a data sample $\mathbf{x} = [x^1, \dots, x^m]$ over m binary features can be seen as a truth assignment over m variables $\{s^1, \dots, s^m\}$ that assigns $\mathbf{x}(s^j) = x^j$. Similarly, a CNF formula $\mathcal{R}$ over the variables $\{s^1, \dots, s^m\}$ can be viewed as a Boolean classifier over such samples that assigns the class $\mathbf{x}(\mathcal{R})$ to the sample $\mathbf{x}$. Such classifiers are called decision rules. A LIDR benchmark based on a given a set $\{(\mathbf{x}_i, y_i) \mid i = 1, \dots, n\}$ of n data samples $\mathbf{x}_i$, each associated with a known Boolean class y_i, is a bi-objective MaxSAT problem $I = (F, O_\mathrm{I}, O_\mathrm{D})$ obtained using the encoding from (Malioutov & Meel, 2018).[6] Each solution τ to I maps to a decision rule $\mathcal{R}^\tau$ for which $O_\mathrm{I}(\tau)$ matches the size of $\mathcal{R}^\tau$ measured as the total number of variables in $\mathcal{R}^\tau$. Furthermore, $O_\mathrm{D}(\tau)$ is the classification error, i.e., the number of data samples $\mathbf{x}_i$ that $\mathcal{R}^\tau$ does not assign the class y_i to.

When enumerating multiple solutions corresponding to the same non-dominated point, stronger domain-specific blocking clauses are employed: we identify a subset of the literals in the MaxSAT problem that uniquely define the decision rule $\mathcal{R}^\tau$ that a solution τ of the bi-objective MaxSAT instance corresponds to, and block each solution by negating only those literals. Furthermore, since BiOptSat enumerates the Pareto-optimal solutions in a known order, it is enough to block variables assigned to 0 in the found solution.

We generated the LIDR instances using 24 UCI (Dua & Graff, 2021) and Kaggle (`https://www.kaggle.com`) benchmark datasets, including ones used in the original evaluation of the base MaxSAT encoding (Malioutov & Meel, 2018), with the encoding configured to learn CNF decision rules consisting of two clauses. More details on the datasets are provided in Appendix B. We independently at random sampled subsets of $n \in \{50, 100, 1000, 5000, 10000\}$ data samples from the datasets, four of each size (given that the original dataset contained at least as many samples). This resulted in a total of 372 datasets. The datasets were discretized following (Malioutov & Meel, 2018): categorical features were one-hot encoded and continuous features discretized by comparing to a collection of thresholds.

4.2.2 Bi-Objective Set Covering

An instance of the bi-objective set covering problem consists of a ground set $N = \{1, \dots, n\}$ and a collection $\mathcal{S}$ of subsets of N with each element $e \in N$ assigned two weights c_1^e, c_2^e. A cover $\mathcal{C}$ of $\mathcal{S}$ is a subset of N that intersects with each $s \in \mathcal{S}$. We consider the problem

6. As an implementation detail, we extended the base encoding from (Malioutov & Meel, 2018) by breaking symmetries that arise from the encoding imposing a lexicographic ordering on the clauses of the decision rules being learned. The symmetries in the base encoding result in two rules $\mathcal{R}_1$ and $\mathcal{R}_2$ containing the same clauses in different orders being unnecessarily considered distinct.

of computing the covers of $\mathcal{S}$ that are Pareto-optimal with respect to the two objectives $c^1(\mathcal{C}) = \sum_{e \in \mathcal{C}} c_1^e$ and $c^2(\mathcal{C}) = \sum_{e \in \mathcal{C}} c_2^e$, following the original work describing the P-minimal approach (Soh et al., 2017). We encode the covering problem into bi-objective MaxSAT using the standard way of encoding set covering in propositional logic: for each set $s \in \mathcal{S}$, introduce the clause $\{v_e \mid e \in S\}$, where v_e is an indicator variable representing whether element e is in $\mathcal{C}$. The two objectives are $O_\mathrm{I} = \sum_e c_1^e \cdot v_e$ and $O_\mathrm{D} = \sum_e c_2^e \cdot v_e$.

We generated two types of bi-objective set covering problem instances.

SC-EP: instances were generated by enforcing a fixed probability p for an element appearing in a set.

SC-SC: instances were generated by enforcing fixed set cardinality s, with elements in a set chosen uniformly at random without replacement.

Both types of set covering instances were generated using combinations of the following parameters: number of elements $n \in \{100, 150, 200\}$, number of sets $m \in \{20, 40, 60, 80\}$, element probability $p \in \{0.1, 0.2\}$ and set cardinality $s \in \{5, 10\}$. For each parameter combination, we generated five instances, leading to a total of 120 instances of each type. The integer cost values c for the two objectives were chosen uniformly at random from the range $c \in [1, 100]$.

4.2.3 Package Upgradeability

The package upgradeability problem deals with automated package management of software packages in operating systems, with the goals of satisfying user requests and maintaining a consistent state of packages regarding their dependencies and conflicts. A user may also have preferences regarding how much their system is allowed to change in order to fulfil the specified request. PackUP (Janota et al., 2012) provides a Boolean encoding for the package upgradeability problem under five optional minimization objectives: the number of newly installed, removed, changed, and outdated packages, as well as the number of unmet recommendations. We used PackUP to generate instances based on 142 package upgradeability instances from Mancoosi International Solver Competition 2011 (`https://www.mancoosi.org/misc-2011/`). For bi-objective instances, we consider all pairs out of the 5 objectives, which led to 1057 instances after removing duplicates and instances with empty objectives.

4.2.4 Bi-objective Problems from MaxSAT Lib

As a basis for further bi-objective instances, we considered benchmark instances submitted to MaxSAT Evaluation (Bacchus et al., 2019) between 2006 and 2018, obtained from MaxSAT Lib (`http://www.cs.toronto.edu/maxsat-lib/maxsat-instances/`). In particular, we used the algorithm described by Paxian, Raiola, and Becker (2021) to detect whether the single objective O of each MaxSAT Lib instance (F, O) is of the form $O = O_\mathrm{D} + \lambda \cdot O_\mathrm{I}$ for some constant $\lambda > \textsc{sum-coeff}(O_\mathrm{D})$. Such instances are single-objective encoded lexicographic optimization instances (employing the weighted sum method) based on originally bi-objective problem instances. For our evaluation we reverse-engineered each such instance into a true bi-objective MaxSAT instance $(F, O_\mathrm{I}, O_\mathrm{D})$. We included in the benchmark set all MaxSAT Lib families for which each instance could be reverse engineered into a multi-objective instance with exactly two objectives. This resulted in a total of 254 bi-objective instances based on six MaxSAT Lib families: `spot5`, `drmx-atmostk` (`am-k`), `drmx-cryptogen`, `haplotyping-pedigrees` (`hap-ped`), `protein_ins` (`prot`), and `frb` (see Appendix C for more details on the instances).

4.3 Results

We turn to a detailed overview of our empirical results. We start with a performance comparison of BiOptSat and its direct competitors on the task of discovering one representative Pareto-optimal solution for each non-dominated point. Subsequently, we empirically analyze the best-performing BiOptSat instantiation in more detail.

4.3.1 Performance Comparison

Table 1 shows the number of solved instances per benchmark domain for all six instantiations of BiOptSat and all considered competing approaches. Furthermore, Table 2 lists the normalized PAR-2 scores for each solver.[7] The MaxSAT Lib families `drmx-cryptogen` and `frb` are excluded from the results as none of the solvers solved any instances from these families under the 1.5-h per-instance time limit. The best-performing approach for computing the entire non-dominated set per benchmark domain is highlighted in bold. We stress again that here the leximaxIST solver is instead computing a single *leximax*-optimal solution, hence solving a simpler problem than the other solvers; the results for leximaxIST are thereby not directly comparable to those for the other solvers. Overall, Pareto-MCS, Seesaw[8], and CLM-IHS are not competitive with P-minimal, CLM-LB, and BiOptSat, each solving significantly fewer instances from each of the benchmark domains.

For the LIDR domain, all instantiations of BiOptSat solve more instances than P-minimal and CLM-LB, the best-performing competitors of the approaches that find the entire non-dominated set. As can be observed from the runtime distributions in Figure 3 and the PAR-2 scores in Table 2, the performance difference between BiOptSat and P-minimal is relatively small, but especially the `SAT-UNSAT` instantiation clearly outperforms P-minimal, by more than 1000 seconds on the hardest instances.

Turning to set covering, we observe that the `SAT-UNSAT` and hybrid instantiations of BiOptSat outperform P-minimal and CLM-LB on both types of set covering instances. With the runtime distributions shown in Figure 4, we observe that the `OSHybrid` instantiation performs notably well on the SC-SC domain, solving > 11 % more instances than the second-best performing approach.

On the PackUP benchmark domain, all BiOptSat instantiations outperform CLM-LB. The `MSU3` and `MSHybrid` BiOptSat instantiations also solve more instances than P-minimal. However, by considering the PAR-2 scores, we observe that all BiOptSat instantiations score better than P-minimal, suggesting that BiOptSat solves the instances faster.

7. The normalized PAR-x score of a solver on a benchmark domain is obtained by summing the total runtime (in seconds) of the solver over all solved instances, adding x times the timeout of 5 400 seconds (i.e. 10 800 seconds for $x = 2$) for each instance on which the solver was unable to find all non-dominated points within the 1.5-h timeout, and dividing the result by the number of instances in the domain.

8. In the case of Seesaw, one should here note that the framework was primarily developed for capturing settings where one objective cannot be encoded directly but needs to be treated as a black box. Since here both objectives are directly encoded, these potential benefits of the framework do not apply here.

		Set Covering			MaxSAT Lib			
Domain	LIDR	SC-EP	SC-SC	PackUP	`spot5`	`hap-ped`	`am-k`	`prot`
# Inst.	371	120	120	1057	32	100	36	11
`SAT-UNSAT`	**222**	**94**	42	846	20	21	**24**	**11**
`UNSAT-SAT`	**222**	89	38	846	20	21	23	**11**
`MSU3`	221	89	38	**848**	18	22	23	**11**
`OLL`	221	73	41	847	**24**	22	23	**11**
`MSHybrid`	**222**	**94**	42	**848**	16	22	**24**	**11**
`OSHybrid`	**222**	90	**47**	847	10	21	23	**11**
P-minimal	220	89	40	846	20	**23**	23	**11**
CLM-LB	185	93	43	819	5	19	1	0
CLM-IHS	86	86	39	626	4	7	0	0
Seesaw	135	60	39	204	6	18	0	2
Pareto-MCS	34	0	0	277	0	1	0	0
leximaxIST	220	52	42	962	8	23	18	11

Table 1: Solved instances by approach and benchmark domain.

		Set Covering			MaxSAT Lib			
Domain	LIDR	SC-EP	SC-SC	PackUP	`spot5`	`hap-ped`	`am-k`	`prot`
`SAT-UNSAT`	**4434**	2872	7358	2201	4122	8610	**3730**	182
`UNSAT-SAT`	4453	3149	7567	2188	4166	8582	3996	518
`MSU3`	4484	3171	7584	**2159**	4839	8483	4002	389
`OLL`	4478	4823	7385	2168	**2822**	8509	4072	102
`MSHybrid`	4439	**2844**	7293	2161	5465	8506	3746	158
`OSHybrid`	4437	3294	**6870**	2173	7464	8575	3992	**101**
P-minimal	4531	3578	7583	2212	4105	**8408**	4032	161
CLM-LB	5502	2872	7156	2563	9183	8771	10583	10800
CLM-IHS	8386	3660	7527	4494	9450	10071	10800	10800
Seesaw	6980	5981	7711	8717	8779	8900	10800	8853
Pareto-MCS	9832	10800	10800	8012	10800	10800	10800	10800
leximaxIST	4513	6889	7423	987	8254	8395	5411	386

Table 2: Normalized PAR-2 scores by approach and benchmark domain.

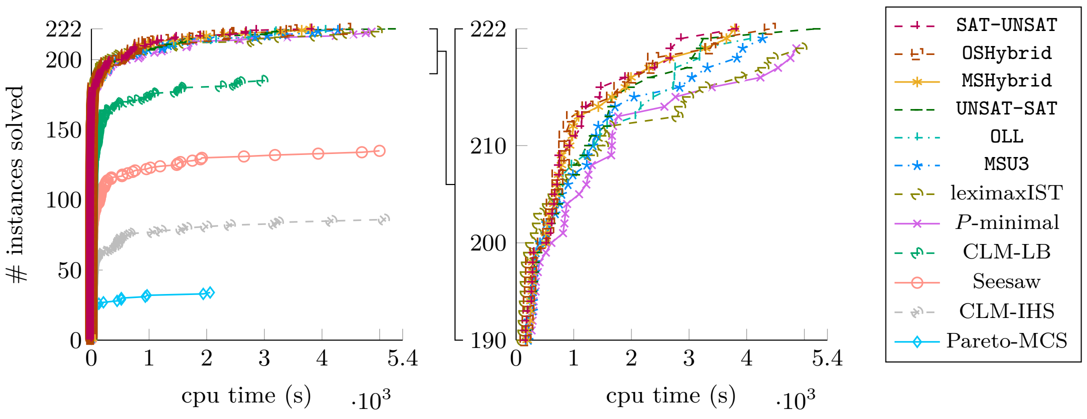

Figure 3: Solver runtime distributions on the LIDR benchmark domain.

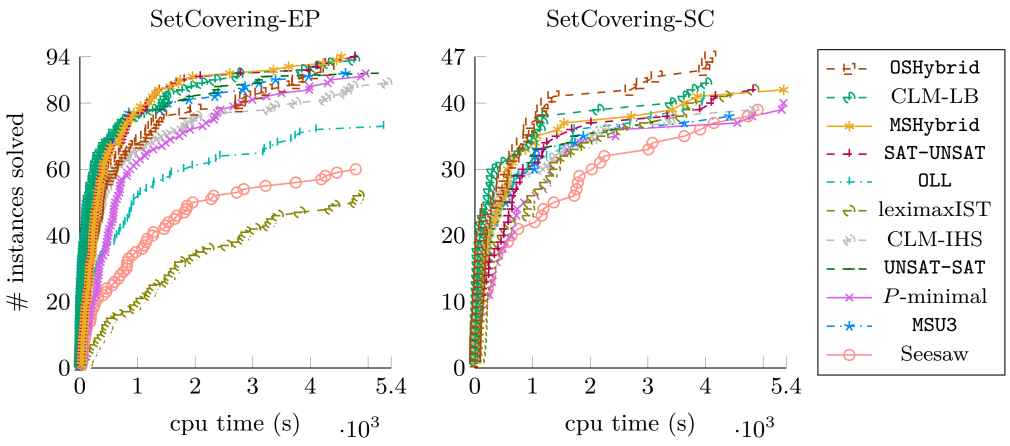

Figure 4: Solver runtime distributions on the set covering benchmark domains.

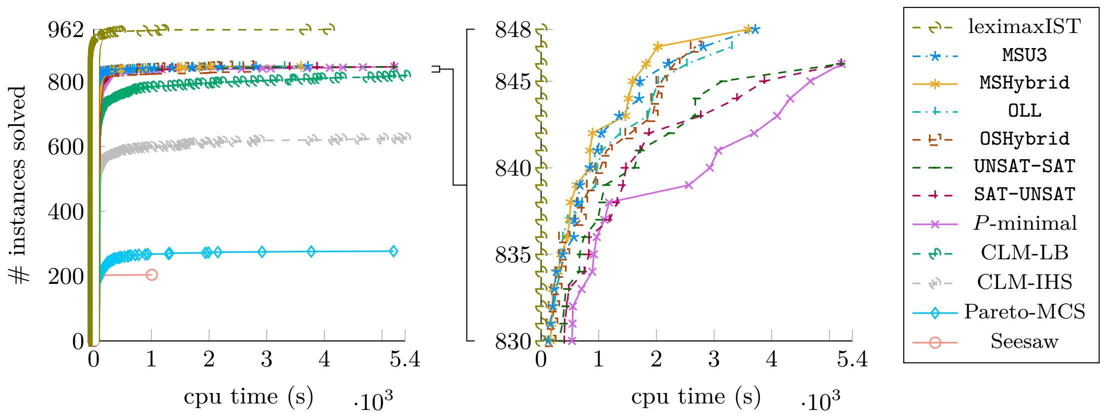

Figure 5: Solver runtime distribution on the PackUP benchmark domain.

This can also be observed in the runtime distributions shown in Figure 5: all BiOptSat instantiations clearly outperform *P*-minimal up to a runtime limit of about 4000 seconds.

Turning to the reverse engineered bi-objective instances from MaxSAT Lib, on the **`hap-ped`**, **`am-k`**, and **`prot`** families the performance of the BiOptSat instantiations and *P*-minimal is very similar. The runtime distributions for each of the families are shown in Figure 6. On the **`spot5`** family, the two hybrid instantiations of BiOptSat are outperformed by the other BiOptSat instantiations and *P*-minimal. On the other hand, BiOptSat outperforms CLM-LB on each of the families.

The percentage of instances on which the hybrid variants `MSHybrid` and `OSHybrid` of BiOptSat switch to `SAT-UNSAT` varies significantly between the benchmark domains. On SC-EP, **`spot5`**, **`am-k`**, **`prot`** both hybrids switch to `SAT-UNSAT` on almost all solved instances. On **`hap-ped`** both hybrids switch on almost no instance. For the remaining domains, both `MSHybrid` (that uses `MSU3` as the core-guided component) and `OSHybrid` (that uses `OLL`) switched on similar percentages of solved instances: $\sim 77\%$ (LIDR), $\sim 53\%$ (SC-SC), and $\sim 16\%$ (PackUP) of solved instances.

Turning to comparing BiOptSat to leximaxIST, recall again that leximaxIST is specific to the task of computing a single leximax-optimal solution. This task is arguably easier than the tasks of computing a single representative solution for each non-dominated point.[9] However, comparing the runtime performance of BiOptSat on the more general task of enumerating the entire non-dominated set to the runtime performance of leximaxIST on the task of computing a single leximax-optimal solution (see Tables 1–2 and Figures 3–5) we observe that, on all but the PackUP and the **`hap-ped`** benchmark domains, at least one and often most of the BiOptSat instantiations are able to find a representative solution for all elements of the non-dominated set—and thus also a leximax-optimal solution—for more instances than leximaxIST can find a single leximax-optimal solution for.

9. What comes to BiOptSat, note that given a representative Pareto-optimal solution for each non-dominated point, the one(s) with the smallest maximum objective value is (are) leximax-optimal. As such BiOptSat, can also be used for computing a leximax-optimal solution by simply enumerating the entire non-dominated set and returning one of the found solutions that obtains the smallest maximum objective value.

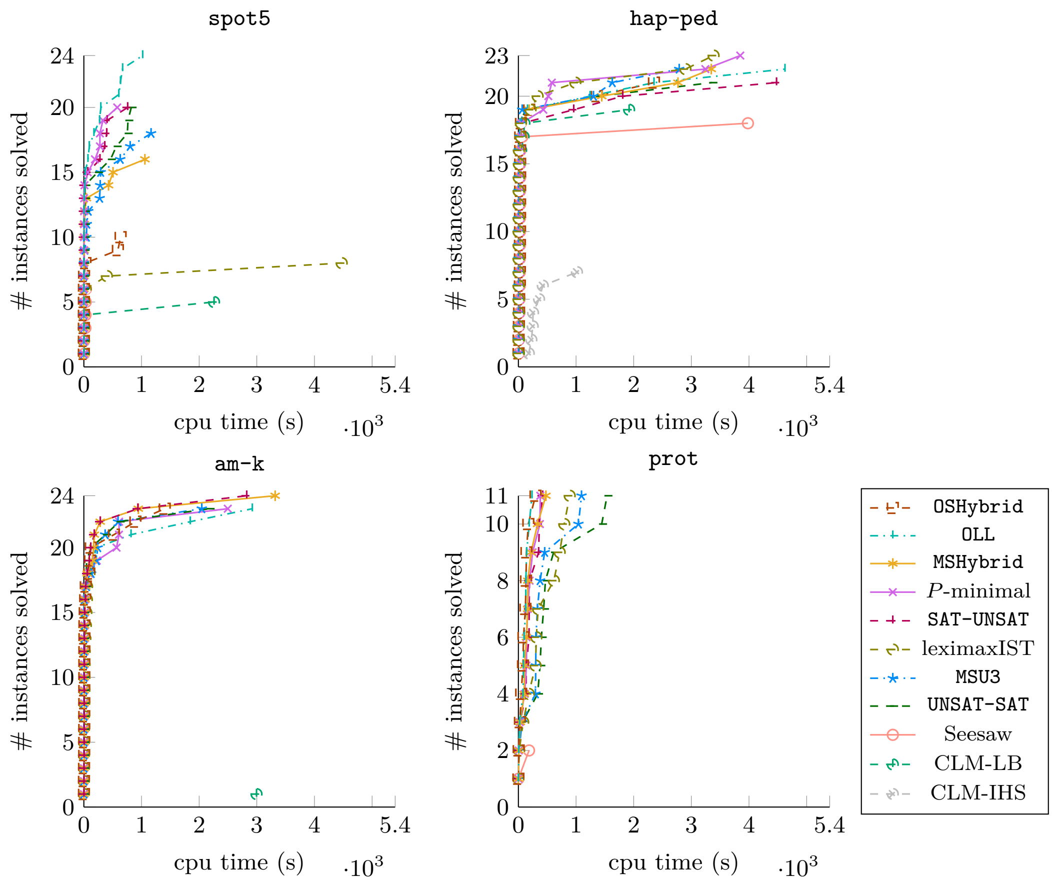

Figure 6: Solver runtime distributions on the MaxSAT Lib benchmark families.

4.3.2 Detailed Analysis of the Best-performing BiOptSat Instantiation

Finally, we provide more detailed empirical analysis on the `MSHybrid` instantiation of BiOptSat. In the remaining of this section we will refer to BiOptSat instantiated with `MSHybrid` simply by BiOptSat. Since a detailed analysis of each of the six instantiations would consume significant computational resources, we chose to focus on `MSHybrid` based on the observation that it solved the most instances on most of the benchmark domains. Specifically, we provide (i) a detailed comparison of BiOptSat with *P*-minimal and leximaxIST, (ii) details on how much time BiOptSat spends in on the two subroutines `Minimize-Inc` and `Sol-Impr-Search`, (iii) an overview of the overhead incurred when enumerating all Pareto-optimal solutions compared to enumerating one representative per non-dominated point, and (iv) discussion on the impact of directly encoding PB constraints as CNF compared to expanding them into cardinality constraint (recall Section 3.3) as the main difference between the current implementation of BiOptSat and the one presented in the preliminary version of this work (Jabs et al., 2022).

For a comparison of the per-instance runtimes of BiOptSat with *P*-minimal, CLM-LB and leximaxIST, respectively, see Figure 7. (Similar comparisons individually between BiOptSat in the `MSHybrid` variant and CLM-IHS, Seesaw, and Pareto-MCS are provided in

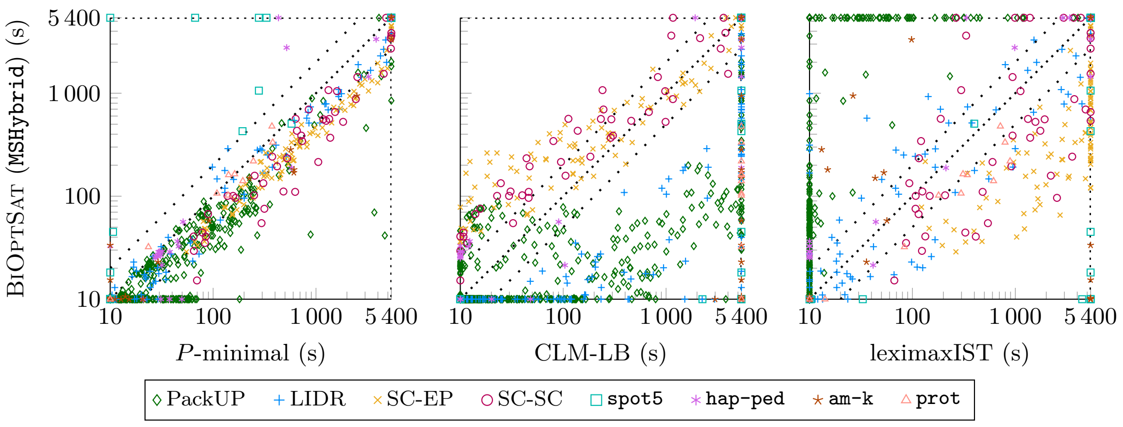

Figure 7: Runtime comparisons of BiOptSat instantiated with `MSHybrid` and (left) P-minimal, (middle) CLM-LB, (right) leximaxIST.

Appendix D.) BiOptSat outperforms P-minimal on all domains apart from the MaxSAT Lib family `spot5`. Overall, BiOptSat solves 18 instances that P-minimal is unable to solve, while the converse holds for 10 instances. CLM-LB outperforms BiOptSat on the set covering domains, while BiOptSat is significantly more efficient at solving PackUP and LIDR instances. Overall, CLM-LB solves only 5 instances that BiOptSat does not solve while the converse holds for 128 instances. Compared to leximaxIST, BiOptSat performs significantly better on both set covering domains. BiOptSat solves 68 instances that leximaxIST is unable to solve. This is especially notable for the benefit of BiOptSat since, again, BiOptSat computes one solution for *each* non-dominated point while leximaxIST only computes a single leximax-optimal solution. Excluding PackUP, which is the benchmark domain leximaxIST was originally evaluated on (Cabral et al., 2022), there were only 11 instances for which leximaxIST was able to compute a single leximax-solution while on which BiOptSat was not able to compute a solution for each non-dominated point.

Turning to analyzing the internal runtime behavior of BiOptSat, Figure 8 (left) provides a per-instance comparison of the time BiOptSat spends in each of the two subroutines `Minimize-Inc` (minimizing the “increasing objective”) and `Sol-Impr-Search` (minimizing the “decreasing objective”). For most domains, a clear majority of the runtime is spent in one of the subroutines, but the subroutine in question clearly depends on the benchmark domain. For LIDR, PackUP and `spot5` more time is spent in the `Sol-Impr-Search` routine while for the other domains the majority of the time is spent in the `Minimize-Inc` routine. This suggests investigating potential instance-specific heuristics for deciding which objective to treat as increasing and decreasing. In an attempt to understand how the choice of which objective to treat as the increasing and decreasing affects runtimes of BiOptSat, we ran all instances with the objectives switched. We found that the performance BiOptSat is in fact relatively robust in terms of how the objectives are treated: the number of solved instances varied only by 20 between the virtual best and worst objective choice per instance.

Figure 8 (right) provides a per-instance runtime comparison of using BiOptSat for the tasks of finding (i) a single representative solution per non-dominated point and (ii) all

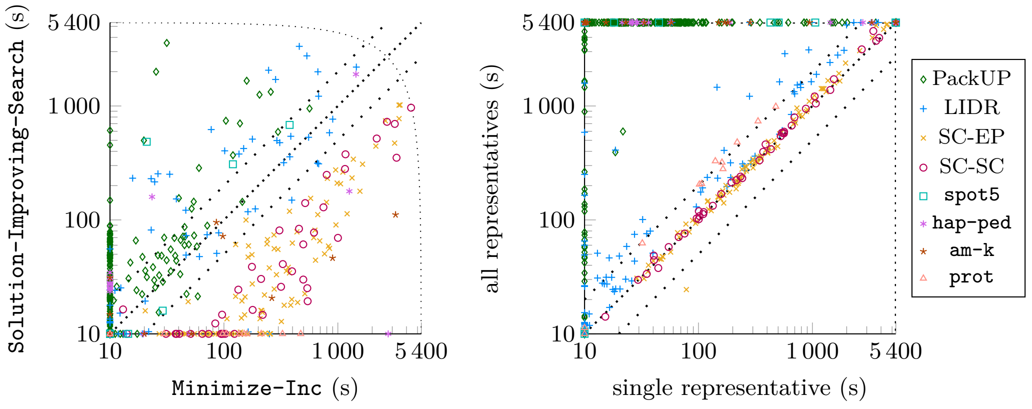

Figure 8: Left: Time (in seconds) spent in the `Minimize-Inc` and `Sol-Impr-Search` subroutines. Right: Runtime comparison for enumerating a single representative per non-dominated point or all Pareto-optimal solutions. Both for the `MSHybrid` instantiation of BiOptSat.

Pareto-optimal solutions. Overall, the enumeration of all Pareto-optimal solutions could be achieved for 38% of the instances for which the non-dominated set could be discovered. However, the additional overhead incurred by enumerating all Pareto-optimal solutions (compared to a single representative solution per non-dominated point) strongly depends on the benchmark domain. Out of the 990 instances for which not all Pareto-optimal solutions could be enumerated, 921 are from the PackUP and 24 from the `am-k` domain. In contrast, for LIDR, set covering, and the `prot` family of MaxSAT Lib, enumerating all Pareto-optimal solutions is achieved by BiOptSat with relatively minor overhead. We note that, interestingly, the domain-specific blocking clauses (recall Section 4.2) applicable in BiOptSat in the LIDR domain turned out to be crucial for enumeration of all Pareto-optimal decision rules. In fact, without domain-specific blocking full enumeration was possible for only nine instances. This suggests that specific domain knowledge when available has high potential for speeding up BiOptSat in full enumeration of all Pareto-optimal solutions in other problems domains as well.

Turning to the runtime impact of the choice of PB encodings in BiOptSat (recall Section 3.3), Figure 9 (left) provides a per-instance runtime comparison of BiOptSat when employing the totalizer cardinality encoding (as earlier implemented in BiOptSat) and the generalized totalizer PB encoding described in Section 3.3. Overall, employing the generalized totalizer appears to be often a better choice than employing the totalizer cardinality encoding. The generalized totalizer leads to a significant speed-up in particular on the set covering benchmarks. Note that in the other benchmark domains apart from set covering, objective coefficients of terms are small (see Table 3 in Appendix A for details). It seems reasonable to assume that the generalized totalizer PB encoding will not provide significant improvements over the totalizer cardinality encoding on such benchmarks. On the PackUP domain, all instances of which have small objective coefficients (up to a maximum of four), the generalized totalizer results in an overall runtime overhead on some of the relatively easy-to-solve instances in the lower left-hand side of Figure 9 (left). This may be due to an

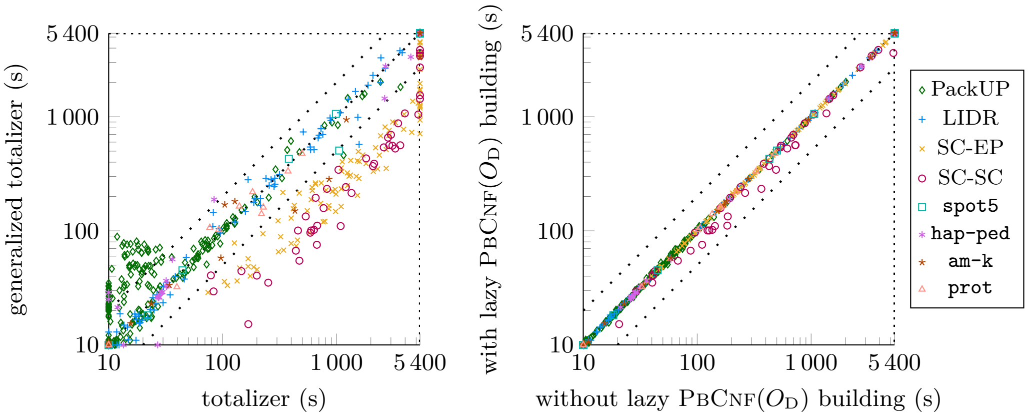

Figure 9: Performance impact of using the generalized totalizer PB encoding (left) and lazily building the PB encoding over the decreasing objective (right).

overhead incurred from initially building the more complicated generalized totalizer PB encoding. In contrast, the objective coefficients are more varied in the set covering instances, allowing for BiOptSat gaining more in performance from employing the generalized totalizer PB encoding. Expectedly, the performance improvement due to using a PB encoding on the set covering domains is likely to translate to other domains containing objectives with highly varying coefficients.

Finally, we note that preliminary work on BiOptSat (Jabs et al., 2022; Jabs, 2022) considered an approach of lazily building the totalizer cardinality encoding over the decreasing objective in the core-guided BiOptSat instantiations if the two objectives share literals. It turns out that by employing the generalized totalizer PB encoding described in Section 3.3 this refinement has much less of an impact. In detail, Figure 9 (right) shows the impact of this refinement for the `MSHybrid` instantiation when employing the generalized totalizer encoding. When employing the totalizer cardinality encoding, lazily building the encoding over the decreasing objective had a positive impact, especially on the SC-SC domain (as reported by Jabs et al. (2022, Section 3.3 and Figure 5) and Jabs (2022, Section 4.3.1 and Figure 5.7)), a similar positive effect is not observed when employing the generalized totalizer PB encoding. This suggests that the performance improvements previously gained by lazily building the encoding over the decreasing encoding can also be achieved by employing a PB encoding instead of expanding the objective into a cardinality constraint.

5. Related Work

Before conclusions, we discuss further related work, adding on what was covered in the previous sections and in particular in Sections 1–2.

SAT-based approaches have previously been developed for computing lexicographically optimal solutions by reducing the problem into single-objective MaxSAT (Argelich, Lynce, & Silva, 2009; Marques-Silva et al., 2011). Many modern MaxSAT solvers exploit properties of instances encoded by the weighted sum method to improve search efficiency (Ansótegui, Bonet, Gabàs, & Levy, 2012; Paxian et al., 2021). In contrast—and while BiOptSat can also be employed for finding leximax-optimal solutions via early termination—we focus on the more generic setting of computing the non-dominated set and its representative solutions.

Beyond SAT-based approaches, multi-objective optimization has been studied in the context of other declarative optimization paradigms. An early algorithm used in constraint programming (Rossi et al., 2006) is based on the lexicographic method (Wassenhove & Gelders, 1980; Marler & Arora, 2004). A branch-and-bound-based algorithm that outperforms the previous algorithm was presented later (Gavanelli, 2002). This improved filtering algorithm was improved again by the Pareto constraint (Schaus & Hartert, 2013; Hartert & Schaus, 2014). The resulting search algorithm is similar to Pareto-MCS in that it maintains a set $\mathcal{T}$ of solutions that do not dominate each other. When a new solution is found, any solution it dominates is removed from $\mathcal{T}$. Constraint programming approaches have also been proposed for finding lexicographically and leximax-optimal solutions (Ehrgott & Gandibleux, 2000; Argelich et al., 2009; Bouveret & Lemaître, 2009; Marques-Silva et al., 2011). The leximax optimization approaches developed include ones based on branch-and-bound and others based on adding constraints to encode the sorted objective value (Bouveret & Lemaître, 2009). Multi-objective optimization has also been studied in the context of linear programming, mixed integer programming and zero-one-programming (Ehrgott, 2005; Rasmussen, 1986; Alves & Clímaco, 2007). The underlying algorithmic approaches considered include ones based on Simplex (Evans & Steuer, 1973; Ehrgott, 2005), branch-and-bound (Santis, Eichfelder, Niebling, & Rocktäschel, 2020; Adelgren & Gupte, 2022), as well as some based on reducing the problem of finding a Pareto-optimal solution to single-objective mixed integer programming (Soland, 1979; Sun, 2017; Lu, Mizuno, & Shi, 2020).

Beyond exact approaches, incomplete search algorithms for multi-objective optimization have also been proposed (Zitzler & Thiele, 1998; Dubois-Lacoste, López-Ibáñez, & Stützle, 2012, 2015; Jaszkiewicz, 2018; Saini & Saha, 2021). In contrast to exact approaches (including the one we developed in this work), incomplete approaches are *not* guaranteed to return the exact non-dominated set, and instead aim to return a set of solutions that dominates as many of the solutions of the instance as possible under given resource limits. Literature on incomplete optimization algorithms is vast; for a survey, see e.g. (Saini & Saha, 2021). Incomplete algorithms that have been extended to multiple objectives include, e.g., Pareto local search (Dubois-Lacoste et al., 2012, 2015; Jaszkiewicz, 2018), simulated annealing (Kirkpatrick, Gelatt Jr., & Vecchi, 1983; Bandyopadhyay, Saha, Maulik, & Deb, 2008; Sengupta & Saha, 2018), and evolutionary algorithms (Dasgupta & Michalewicz, 1997; Storn & Price, 1997; Zitzler & Thiele, 1998; Deb, Agrawal, Pratap, & Meyarivan, 2002). All three of these categories of algorithms can be described as "local search style", where one or more solutions are iteratively modified to find better solutions.

6. Conclusions

We presented BiOptSat, an approach to exact bi-objective optimization under Pareto-optimality. The structured search of BiOptSat builds on algorithms for MaxSAT and makes incremental use of a SAT solver. BiOptSat allows for both finding the non-dominated set (with a representative solution for each element) and enumerating all Pareto-optimal solutions of NP-hard bi-objective optimization problems encoded in propositional logic. Furthermore, the first solution the approach finds is guaranteed to be lexicographically optimal. BiOptSat constitutes an algorithmic framework that can be instantiated in multiple ways. We detailed four instantiations of BiOptSat that are based on individual SAT-UNSAT/UNSAT-SAT and core-guided algorithms proposed for MaxSAT, as well as two novel hybrids between core-guided and SAT-UNSAT search. We provided an open-source implementation of all six instantiations. We empirically evaluated our implementation of BiOptSat on four well-motivated application settings of bi-objective optimization, comparing its performance to that of previously-proposed approaches solving the same problem setting. In the experiments, the hybrid `MSHybrid` instantiation of BiOptSat outperformed both all the five other instantiations of BiOptSat and each of its direct competitors. We also detailed the use of incremental PB encodings, which in the context of BiOptSat turned out to be a significant factor in improving the runtime performance of the approach, which we showed also empirically for the best-performing `MSHybrid` instantiation.

There are various possibilities for further work towards potential further practical improvements. For example, we based BiOptSat on combinations of solution-improving and core-guided MaxSAT approaches. The question of whether recent developments in branch-and-bound style MaxSAT solving (Li, Xu, Coll, Manyà, Habet, & He, 2021) could be integrated into the framework remains open. Secondly, the potential of employing recently-developed incremental MaxSAT solving techniques (Niskanen, Berg, & Järvisalo, 2021, 2022) for speeding up BiOptSat search further could be studied. Towards more fine-grained improvements, evaluating the practical impact of different heuristic choices within the BiOptSat subroutines, such as core minimization in the context of core-guided search, might provide further insights. Finally, extending BiOptSat to multi-objective optimization beyond two objectives remains a non-trivial challenge from the practical perspective.

Acknowledgments

Work financially supported by Academy of Finland under grants 322869, 328718, 342145, and 356046. The authors wish to thank the Finnish Computing Competence Infrastructure (FCCI) for supporting this project with computational and data storage resources.

Appendix A. Benchmark Statistics

Table 3 provides for each benchmark family the minimum, median, and maximum values of the number of variables and the number of clauses; and the minimum, maximum, and median of the weights over both objectives and the number of unique weights.

Appendix B. LIDR Datasets

Table 4 provides a summary of the datasets based on which the LIDR benchmark were generated, including their origin, number of data samples (# samp.), number of features before (# feat.) and after (# disc. feat.) discretization, and the sizes of the resulting bi-objective MaxSAT instances in terms of number of clauses (# cls) and number of variables

Family	# Inst.	# Variables			# Clauses			O_I Weight				O_D Weight			
		min	med	max	min	med	max	# uniq	min	med	max	# uniq	min	med	max
LIDR	371	168	2116	736k	341	7718	11M	unit coefficients				unit coefficients			
SC-EP	120	86	150	200	20	50	80	100	1	51	100	100	1	51	100
SC-SC	120	57	120	200	20	50	80	100	1	51	100	100	1	51	100
PackUP	1057	364	7199	21k	1243	38k	127k	3	1	1	4	3	1	1	4
`spot5`	32	101	609	3004	427	7532	37k	3	1	1	5	2	1	2	2
`am-k`	36	181	505	3013	381	1474	4422	unit coefficients				unit coefficients			
`drmx-cryptogen`	40	1600	10k	23k	7244	36k	79k	unit coefficients				unit coefficients			
`hap-ped`	100	62k	172k	216k	275k	1.1M	4.1M	unit coefficients				unit coefficients			
`prot`	11	171	2236	3016	13k	2.1M	3.8M	unit coefficients				unit coefficients			
`frb`	35	60	325	760	601	11k	42k	unit coefficients				unit coefficients			

Table 3: Benchmark family summary statistics: minimum (min), median (med), maximum (max), and unique values for the number of variables and clauses as well as minimum, median and maximum objective weights and the number of unique weights.

Dataset	Origin	# samp.	# feat.	# disc. feat.	# cls (10^3)	# vars (10^3)
Adult	UCI	32 561	14	144	635	98.1
Bank Marketing	UCI	45 211	16	88	1329	136
Banknote Authentication	UCI	372	4	16	6.67	4.16
Connect 4	UCI	67 557	42	126	2052	203
Default of Credit Card Clients	UCI	30 000	23	110	878	90.3
Dota 2 Games Results	UCI	92 650	115	345	11 164	279
FIFA 2018 Man of the Match	Kaggle	128	26	106	3.00	0.708
Heart Disease	Kaggle	303	13	31	3.72	1.00
Indian Liver Patient Dataset	UCI	583	10	14	6.67	1.79
Ionosphere	UCI	351	33	144	9.90	1.49
Iris	UCI	150	4	11	1.08	0.483
MAGIC Gamma Telescope	UCI	19 020	10	79	273	57.3
Medical Hospital Readmissions	Kaggle	25 000	64	125	1641	75.4
Mushroom	UCI	8124	22	115	190	24.7
Parkinsons	UCI	195	22	51	2.81	0.738
Pima Indians Diabetes	Kaggle	768	8	30	7.25	2.39
Skin Segmentation	UCI	245 057	3	119	745	736
Tic-Tac-Toe Endgame	UCI	958	9	27	7.75	2.96
Buzz in Social Media (Toms Hardware)	UCI	28 179	96	910	3712	87.3
Buzz in Social Media (Twitter)	UCI	49 999	77	1511	5406	155
Blood Transfusion Service Center	UCI	748	4	6	4.39	2.26
Travel Insurance	Kaggle	63 326	10	211	1188	191
Wisconsin Diagnostic Breast Cancer	UCI	569	30	88	20.7	1.97
Rain in Australia	Kaggle	107 696	16	141	2952	339

Table 4: The datasets used in the decision rule experiments and summary statistics on them and the CNF formulas generated from them.

Domain / Directory	Shortened name	# instances
`spot5`	`spot5`	34 (−2)
`drmx-atmostk`	`am-k`	36
`drmx-cryptogen`	`drmx-cryptogen`	40
`haplotyping-pedigrees`	`hap-ped`	100
`protein_ins`	`prot`	11
`frb`	`frb`	35

Table 5: MaxSAT Lib families with underlying bi-objective problems and their number of instances.

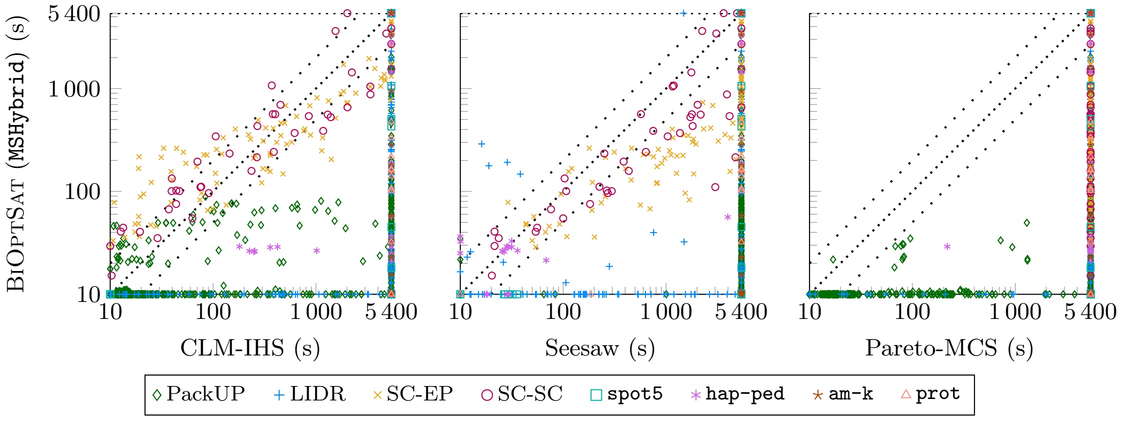

Figure 10: Runtime comparisons of BiOptSat in the `MSHybrid` variant and CLM-IHS (left), Seesaw (middle), and Pareto-MCS (right).

(# vars) in 10^3. The original datasets were downloaded from the UCI Machine Learning Repository (Dua & Graff, 2021) and from Kaggle (`https://www.kaggle.com`). Links to the original datasets as well as the discretized versions used as basis for the bi-objective MaxSAT instances used in the experiments are available at `https://bitbucket.org/coreo-group/bioptsat/src/master/jair24`.

Appendix C. MaxSAT Lib Benchmarks

Table 5 lists all MaxSAT Lib families for which we were able to identify an original bi-objective problem. For the `spot5` family, two out of the 34 instances (`28.wcsp.dir.wcnf.gz` and `28.wcsp.log.wcnf.gz`) could be split into three instead of two objectives and hence were excluded from the benchmark set.

Appendix D. Additional Empirical Data

A pairwise per-instance runtime comparison of the `MSHybrid` variant of BiOptSat and CLM-IHS, Seesaw, and Pareto-MCS is shown in Figure 10.

References

Adelgren, N., & Gupte, A. (2022). Branch-and-bound for biobjective mixed-integer linear programming. *INFORMS J. Comput.*, *34*(2), 909–933.

Alves, M. J., & Clímaco, J. C. N. (2007). A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, *180*(1), 99–115.

Andres, B., Kaufmann, B., Matheis, O., & Schaub, T. (2012). Unsatisfiability-based optimization in clasp. In Dovier, A., & Costa, V. S. (Eds.), *Technical Communications of the 28th International Conference on Logic Programming, ICLP 2012, September 4–8, 2012, Budapest, Hungary*, Vol. 17 of *LIPIcs*, pp. 211–221. Schloss Dagstuhl—Leibniz-Zentrum für Informatik.

Ansótegui, C., Bonet, M. L., Gabàs, J., & Levy, J. (2012). Improving SAT-based weighted MaxSAT solvers. In Milano, M. (Ed.), *Principles and Practice of Constraint Programming—18th International Conference, CP 2012, Québec City, QC, Canada, October 8–12, 2012. Proceedings*, Vol. 7514 of *Lecture Notes in Computer Science*, pp. 86–101. Springer.

Ansótegui, C., Bonet, M. L., Gabàs, J., & Levy, J. (2013). Improving WPM2 for (weighted) partial MaxSAT. In Schulte, C. (Ed.), *Principles and Practice of Constraint Programming—19th International Conference, CP 2013, Uppsala, Sweden, September 16–20, 2013. Proceedings*, Vol. 8124 of *Lecture Notes in Computer Science*, pp. 117–132. Springer.

Ansótegui, C., Bonet, M. L., & Levy, J. (2009). Solving (weighted) partial MaxSAT through satisfiability testing. In Kullmann, O. (Ed.), *Theory and Applications of Satisfiability Testing—SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 – July 3, 2009. Proceedings*, Vol. 5584 of *Lecture Notes in Computer Science*, pp. 427–440. Springer.

Argelich, J., Lynce, I., & Silva, J. P. M. (2009). On solving boolean multilevel optimization problems. In Boutilier, C. (Ed.), *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11–17, 2009*, pp. 393–398.

Arora, J. S. (2004). Multiobjective optimum design concepts and methods. In Arora, J. S. (Ed.), *Introduction to Optimum Design (Second Edition)* (Second Edition edition)., pp. 543–563. Academic Press, San Diego.

Bacchus, F., Järvisalo, M., & Martins, R. (2019). MaxSAT Evaluation 2018: New developments and detailed results. *Journal on Satisfiability, Boolean Modeling and Computation*, *11*(1), 99–131.

Bacchus, F., Järvisalo, M., & Martins, R. (2021). Maximum satisfiabiliy. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 929–991. IOS Press.

Bailleux, O., & Boufkhad, Y. (2003). Efficient CNF encoding of boolean cardinality constraints. In Rossi, F. (Ed.), *Principles and Practice of Constraint Programming—CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 – October 3, 2003, Proceedings*, Vol. 2833 of *Lecture Notes in Computer Science*, pp. 108–122. Springer.

Bandyopadhyay, S., Saha, S., Maulik, U., & Deb, K. (2008). A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, *12*(3), 269–283.

Barrett, C. W., Sebastiani, R., Seshia, S. A., & Tinelli, C. (2021). Satisfiability modulo theories. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 1267–1329. IOS Press.

Bendík, J., & Cerna, I. (2020). Rotation based MSS/MCS enumeration. In Albert, E., & Kovács, L. (Eds.), *LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, May 22–27, 2020*, Vol. 73 of *EPiC Series in Computing*, pp. 120–137. EasyChair.

Berg, J., Bacchus, F., & Poole, A. (2020). Abstract cores in implicit hitting set MaxSat solving. In Pulina, L., & Seidl, M. (Eds.), *Theory and Applications of Satisfiability Testing—SAT 2020—23rd International Conference, Alghero, Italy, July 3–10, 2020, Proceedings*, Vol. 12178 of *Lecture Notes in Computer Science*, pp. 277–294. Springer.

Berg, J., Demirovic, E., & Stuckey, P. J. (2019). Core-boosted linear search for incomplete MaxSAT. In Rousseau, L., & Stergiou, K. (Eds.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research—16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4–7, 2019, Proceedings*, Vol. 11494 of *Lecture Notes in Computer Science*, pp. 39–56. Springer.

Berg, J., & Järvisalo, M. (2017). Weight-aware core extraction in SAT-based MaxSAT solving. In Beck, J. C. (Ed.), *Principles and Practice of Constraint Programming—23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 – September 1, 2017, Proceedings*, Vol. 10416 of *Lecture Notes in Computer Science*, pp. 652–670. Springer.

Biere, A., Fazekas, K., Fleury, M., & Heisinger, M. (2020). CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Balyo, T., Froleyks, N., Heule, M., Iser, M., Järvisalo, M., & Suda, M. (Eds.), *Proceedings of SAT Competition 2020—Solver and Benchmark Descriptions*, Vol. B-2020-1 of *Department of Computer Science Report Series B*, pp. 51–53. University of Helsinki.

Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.). (2021). *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Bouveret, S., & Lemaître, M. (2009). Computing leximin-optimal solutions in constraint networks. *Artificial Intelligence*, *173*(2), 343–364.

Cabral, M., Janota, M., & Manquinho, V. M. (2022). SAT-based leximax optimisation algorithms. In Meel, K. S., & Strichman, O. (Eds.), *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2–5, 2022, Haifa, Israel*, Vol. 236 of *LIPIcs*, pp. 29:1–29:19. Schloss Dagstuhl — Leibniz-Zentrum für Informatik.

Chandrasekaran, K., Karp, R. M., Moreno-Centeno, E., & Vempala, S. S. (2011). Algorithms for implicit hitting set problems. In Randall, D. (Ed.), *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23–25, 2011*, pp. 614–629. SIAM.

Cortes, J., Lynce, I., & Manquinho, V. M. (2023). New core-guided and hitting set algorithms for multi-objective combinatorial optimization. In Sankaranarayanan, S., & Sharygina, N. (Eds.), *Tools and Algorithms for the Construction and Analysis of Systems — 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part II*, Vol. 13994 of *Lecture Notes in Computer Science*, pp. 55–73. Springer.

Dasgupta, D., & Michalewicz, Z. (1997). *Evolutionary algorithms in engineering applications*. Springer.

Davies, J., & Bacchus, F. (2011). Solving MAXSAT by solving a sequence of simpler SAT instances. In Lee, J. H. (Ed.), *Principles and Practice of Constraint Programming—CP 2011—17th International Conference, CP 2011, Perugia, Italy, September 12–16, 2011. Proceedings*, Vol. 6876 of *Lecture Notes in Computer Science*, pp. 225–239. Springer.

Davies, J., & Bacchus, F. (2013a). Exploiting the power of MIP solvers in MAXSAT. In Järvisalo, M., & Gelder, A. V. (Eds.), *Theory and Applications of Satisfiability Testing—SAT 2013—16th International Conference, Helsinki, Finland, July 8–12, 2013. Proceedings*, Vol. 7962 of *Lecture Notes in Computer Science*, pp. 166–181. Springer.

Davies, J., & Bacchus, F. (2013b). Postponing optimization to speed up MAXSAT solving. In Schulte, C. (Ed.), *Principles and Practice of Constraint Programming—19th International Conference, CP 2013, Uppsala, Sweden, September 16–20, 2013. Proceedings*, Vol. 8124 of *Lecture Notes in Computer Science*, pp. 247–262. Springer.

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Dua, D., & Graff, C. (2021). UCI machine learning repository. `http://archive.ics.uci.edu/ml`.

Dubois-Lacoste, J., López-Ibáñez, M., & Stützle, T. (2012). Pareto local search algorithms for anytime bi-objective optimization. In Hao, J., & Middendorf, M. (Eds.), *Evolutionary Computation in Combinatorial Optimization—12th European Conference,* *EvoCOP 2012, Málaga, Spain, April 11–13, 2012. Proceedings*, Vol. 7245 of *Lecture Notes in Computer Science*, pp. 206–217. Springer.

Dubois-Lacoste, J., López-Ibáñez, M., & Stützle, T. (2015). Anytime Pareto local search. *European Journal of Operational Research*, *243*(2), 369–385.

Eén, N., & Sörensson, N. (2003). Temporal induction by incremental SAT solving. *Electronic Notes in Theoretical Computer Science*, *89*(4), 543–560.

Eén, N., & Sörensson, N. (2006). Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, *2*(1–4), 1–26.

Ehrgott, M. (2005). *Multicriteria Optimization (2. ed.)*. Springer.

Ehrgott, M., & Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spectrum*, *22*(4), 425–460.

Evans, J. P., & Steuer, R. E. (1973). A revised simplex method for linear multiple objective programs. *Mathematical Programming*, *5*(1), 54–72.

Fazekas, K., Bacchus, F., & Biere, A. (2018). Implicit hitting set algorithms for maximum satisfiability modulo theories. In Galmiche, D., Schulz, S., & Sebastiani, R. (Eds.), *Automated Reasoning—9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14–17, 2018, Proceedings*, Vol. 10900 of *Lecture Notes in Computer Science*, pp. 134–151. Springer.

Fu, Z., & Malik, S. (2006). On solving the partial MAX-SAT problem. In Biere, A., & Gomes, C. P. (Eds.), *Theory and Applications of Satisfiability Testing—SAT 2006, 9th International Conference, Seattle, WA, USA, August 12–15, 2006, Proceedings*, Vol. 4121 of *Lecture Notes in Computer Science*, pp. 252–265. Springer.

Gavanelli, M. (2002). An algorithm for multi-criteria optimization in CSPs. In van Harmelen, F. (Ed.), *Proceedings of the 15th European Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002*, pp. 136–140. IOS Press.

Ghosh, B., Malioutov, D., & Meel, K. S. (2022). Efficient learning of interpretable classification rules. *Journal of Artificial Intelligence Research*, *74*, 1823–1863.

Grégoire, É., Izza, Y., & Lagniez, J. (2018). Boosting MCSes enumeration. In Lang, J. (Ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 1309–1315. ijcai.org.

Guerreiro, A. P., Cortes, J., Vanderpooten, D., Bazgan, C., Lynce, I., Manquinho, V. M., & Figueira, J. R. (2023). Exact and approximate determination of the Pareto front using minimal correction subsets. *Computers & Operations Research*, *153*, 106153.

Hartert, R., & Schaus, P. (2014). A support-based algorithm for the bi-objective Pareto constraint. In Brodley, C. E., & Stone, P. (Eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27–31, 2014, Québec City, Québec, Canada*, pp. 2674–2679. AAAI Press.

Hölldobler, S., Manthey, N., & Steinke, P. (2012). A compact encoding of pseudo-boolean constraints into SAT. In Glimm, B., & Krüger, A. (Eds.), *KI 2012: Advances in Artificial Intelligence—35th Annual German Conference on AI, Saarbrücken, Germany,* *September 24–27, 2012. Proceedings*, Vol. 7526 of *Lecture Notes in Computer Science*, pp. 107–118. Springer.

Hu, H., Siala, M., Hebrard, E., & Huguet, M. (2020). Learning optimal decision trees with MaxSAT and its integration in AdaBoost. In Bessiere, C. (Ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 1170–1176. ijcai.org.

Ignatiev, A., Marques-Silva, J., Narodytska, N., & Stuckey, P. J. (2021). Reasoning-based learning of interpretable ML models. In Zhou, Z. (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19–27 August 2021*, pp. 4458–4465. ijcai.org.

Ignatiev, A., Morgado, A., & Marques-Silva, J. (2016). Propositional abduction with implicit hitting sets. In Kaminka, G. A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., & van Harmelen, F. (Eds.), *ECAI 2016—22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands—Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, Vol. 285 of *Frontiers in Artificial Intelligence and Applications*, pp. 1327–1335. IOS Press.

Ignatiev, A., Morgado, A., & Marques-Silva, J. (2019). RC2: an efficient MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, *11*(1), 53–64.

Ignatiev, A., Pereira, F., Narodytska, N., & Marques-Silva, J. (2018). A SAT-based approach to learn explainable decision sets. In Galmiche, D., Schulz, S., & Sebastiani, R. (Eds.), *Automated Reasoning—9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14–17, 2018, Proceedings*, Vol. 10900 of *Lecture Notes in Computer Science*, pp. 627–645. Springer.

Ignatiev, A., Previti, A., Liffiton, M. H., & Marques-Silva, J. (2015). Smallest MUS extraction with minimal hitting set dualization. In Pesant, G. (Ed.), *Principles and Practice of Constraint Programming—21st International Conference, CP 2015, Cork, Ireland, August 31 – September 4, 2015, Proceedings*, Vol. 9255 of *Lecture Notes in Computer Science*, pp. 173–182. Springer.

Isermann, H. (1979). The enumeration of all efficient solutions for a linear multiple-objective transportation problem. *Naval Research Logistics Quarterly*, *26*(1), 123–139.

Jabs, C. (2022). A maximum satisfiability based approach to bi-objective boolean optimization. Master's thesis, University of Helsinki, Finland.

Jabs, C., Berg, J., Niskanen, A., & Järvisalo, M. (2022). MaxSAT-based bi-objective boolean optimization. In Meel, K. S., & Strichman, O. (Eds.), *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022), Haifa, Israel, August 2–5, 2022*, Vol. 236 of *LIPIcs*. Schloss Dagstuhl—Leibniz-Zentrum für Informatik.

Janota, M., Lynce, I., Manquinho, V. M., & Marques-Silva, J. (2012). PackUp: Tools for package upgradability solving. *Journal on Satisfiability, Boolean Modeling and Computation*, *8*(1/2), 89–94.

Janota, M., Morgado, A., Santos, J. F., & Manquinho, V. M. (2021). The Seesaw algorithm: Function optimization using implicit hitting sets. In Michel, L. D. (Ed.), *27th Interna-*

tional Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25–29, 2021, Vol. 210 of *LIPIcs*, pp. 31:1–31:16. Schloss Dagstuhl—Leibniz-Zentrum für Informatik.

Jaszkiewicz, A. (2018). Many-objective Pareto local search. *European Journal of Opererational Research*, *271*(3), 1001–1013.

Jin, Y., & Sendhoff, B. (2008). Pareto-based multiobjective machine learning: An overview and case studies. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, *38*(3), 397–415.

Joshi, S., Martins, R., & Manquinho, V. M. (2015). Generalized totalizer encoding for pseudo-boolean constraints. In Pesant, G. (Ed.), *Principles and Practice of Constraint Programming—21st International Conference, CP 2015, Cork, Ireland, August 31 – September 4, 2015, Proceedings*, Vol. 9255 of *Lecture Notes in Computer Science*, pp. 200–209. Springer.

Kirkpatrick, S., Gelatt Jr., D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Koshimura, M., Nabeshima, H., Fujita, H., & Hasegawa, R. (2009). Minimal model generation with respect to an atom set. In Peltier, N., & Sofronie-Stokkermans, V. (Eds.), *Proceedings of the 7th International Workshop on First-Order Theorem Proving, FTP 2009, Oslo, Norway, July 6–7, 2009*, Vol. 556 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Koshimura, M., & Satoh, K. (2020). A simple yet efficient MCSes enumeration with SAT oracles. In Nguyen, N. T., Jearanaitanakij, K., Selamat, A., Trawinski, B., & Chittayasothorn, S. (Eds.), *Intelligent Information and Database Systems - 12th Asian Conference, ACIIDS 2020, Phuket, Thailand, March 23-26, 2020, Proceedings, Part I*, Vol. 12033 of *Lecture Notes in Computer Science*, pp. 191–201. Springer.

Le Berre, D., & Parrain, A. (2010). The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, *7*(2–3), 59–6.

Leivo, M., Berg, J., & Järvisalo, M. (2020). Preprocessing in incomplete MaxSAT solving. In *ECAI*, Vol. 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 347–354. IOS Press.

Li, C., Xu, Z., Coll, J., Manyà, F., Habet, D., & He, K. (2021). Combining clause learning and branch and bound for MaxSAT. In Michel, L. D. (Ed.), *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25–29, 2021*, Vol. 210 of *LIPIcs*, pp. 38:1–38:18. Schloss Dagstuhl—Leibniz-Zentrum für Informatik.

Lu, K., Mizuno, S., & Shi, J. (2020). A new mixed integer programming approach for optimization over the efficient set of a multiobjective linear programming problem. *Optimization Letters*, *14*(8), 2323–2333.

Malioutov, D., & Meel, K. S. (2018). MLIC: A MaxSAT-based framework for learning interpretable classification rules. In Hooker, J. N. (Ed.), *Principles and Practice of Constraint Programming—24th International Conference, CP 2018, Lille, France, Au-*

gust 27–31, 2018, Proceedings, Vol. 11008 of *Lecture Notes in Computer Science*, pp. 312–327. Springer.

Marler, R., & Arora, J. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26*, 369–395.

Marques-Silva, J., Argelich, J., Graca, A., & Lynce, I. (2011). Boolean lexicographic optimization: Algorithms & applications. *Annals of Mathematics and Artificial Intelligence*, *62*(3–4), 317–343.

Marques-Silva, J., Lynce, I., & Malik, S. (2021). Conflict-driven clause learning SAT solvers. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 133–182. IOS Press.

Marques-Silva, J., & Planes, J. (2007). On using unsatisfiability for solving maximum satisfiability. *Computing Research Repository*, *abs/0712.1097*.

Martins, R., Joshi, S., Manquinho, V. M., & Lynce, I. (2014a). Incremental cardinality constraints for MaxSAT. In O'Sullivan, B. (Ed.), *Principles and Practice of Constraint Programming—20th International Conference, CP 2014, Lyon, France, September 8–12, 2014. Proceedings*, Vol. 8656 of *Lecture Notes in Computer Science*, pp. 531–548. Springer.

Martins, R., Joshi, S., Manquinho, V. M., & Lynce, I. (2014b). On using incremental encodings in unsatisfiability-based MaxSAT solving. *J. Satisf. Boolean Model. Comput.*, *9*(1), 59–81.

Martins, R., Manquinho, V. M., & Lynce, I. (2014c). Open-WBO: A modular MaxSAT solver. In Sinz, C., & Egly, U. (Eds.), *Theory and Applications of Satisfiability Testing—SAT 2014—17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14–17, 2014. Proceedings*, Vol. 8561 of *Lecture Notes in Computer Science*, pp. 438–445. Springer.

Moreno-Centeno, E., & Karp, R. M. (2013). The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment. *Operations Research*, *61*(2), 453–468.

Morgado, A., Dodaro, C., & Marques-Silva, J. (2014). Core-guided MaxSAT with soft cardinality constraints. In O'Sullivan, B. (Ed.), *Principles and Practice of Constraint Programming—20th International Conference, CP 2014, Lyon, France, September 8–12, 2014. Proceedings*, Vol. 8656 of *Lecture Notes in Computer Science*, pp. 564–573. Springer.

Morgado, A., Liffiton, M. H., & Marques-Silva, J. (2012). MaxSAT-based MCS enumeration. In Biere, A., Nahir, A., & Vos, T. E. J. (Eds.), *Hardware and Software: Verification and Testing—8th International Haifa Verification Conference, HVC 2012, Haifa, Israel, November 6–8, 2012. Revised Selected Papers*, Vol. 7857 of *Lecture Notes in Computer Science*, pp. 86–101. Springer.

Narodytska, N., Ignatiev, A., Pereira, F., & Marques-Silva, J. (2018). Learning optimal decision trees with SAT. In Lang, J. (Ed.), *Proceedings of the Twenty-Seventh Inter-*

national Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden, pp. 1362–1368. ijcai.org.

Niemelä, I. (1999). Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, *25*(3-4), 241–273.

Niskanen, A., Berg, J., & Järvisalo, M. (2021). Enabling incrementality in the implicit hitting set approach to MaxSAT under changing weights. In Michel, L. D. (Ed.), *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25–29, 2021*, Vol. 210 of *LIPIcs*, pp. 44:1–44:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Niskanen, A., Berg, J., & Järvisalo, M. (2022). Incremental maximum satisfiability. In Meel, K. S., & Strichman, O. (Eds.), *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022), Haifa, Israel, August 2–5, 2022*, Vol. 236 of *LIPIcs*, pp. 14:1–14:19. Schloss Dagstuhl—Leibniz-Zentrum für Informatik.

Parker, M., & Ryan, J. (1996). Finding the minimum weight IIS cover of an infeasible system of linear inequalities. *Annals of Mathematics and Artificial Intelligence*, *17*(1-2), 107–126.

Paxian, T., Raiola, P., & Becker, B. (2021). On preprocessing for weighted MaxSAT. In Henglein, F., Shoham, S., & Vizel, Y. (Eds.), *Verification, Model Checking, and Abstract Interpretation—22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17–19, 2021, Proceedings*, Vol. 12597 of *Lecture Notes in Computer Science*, pp. 556–577. Springer.

Previti, A., Mencía, C., Järvisalo, M., & Marques-Silva, J. (2017). Improving MCS enumeration via caching. In Gaspers, S., & Walsh, T. (Eds.), *Theory and Applications of Satisfiability Testing—SAT 2017—20th International Conference, Melbourne, VIC, Australia, August 28 – September 1, 2017, Proceedings*, Vol. 10491 of *Lecture Notes in Computer Science*, pp. 184–194. Springer.

Rasmussen, L. M. (1986). Zero-one programming with multiple criteria. *European Journal of Operational Research*, *26*(1), 83–95.

Reggia, J. A., Nau, D. S., & Wang, P. Y. (1983). Diagnostic expert systems based on a set covering model. *International Journal of Man-Machine Studies*, *19*(5), 437–460.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, *32*(1), 57–95.

Rossi, F., van Beek, P., & Walsh, T. (Eds.). (2006). *Handbook of Constraint Programming*, Vol. 2 of *Foundations of Artificial Intelligence*. Elsevier.

Saikko, P., Berg, J., & Järvisalo, M. (2016). LMHS: A SAT-IP hybrid MaxSAT solver. In Creignou, N., & Berre, D. L. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, Vol. 9710 of *Lecture Notes in Computer Science*, pp. 539–546. Springer.

Saikko, P., Dodaro, C., Alviano, M., & Järvisalo, M. (2018). A hybrid approach to optimization in answer set programming. In Thielscher, M., Toni, F., & Wolter, F. (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth*

International Conference, KR 2018, Tempe, Arizona, 30 October – 2 November 2018, pp. 32–41. AAAI Press.

Saikko, P., Wallner, J. P., & Järvisalo, M. (2016). Implicit hitting set algorithms for reasoning beyond NP. In Baral, C., Delgrande, J. P., & Wolter, F. (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25–29, 2016*, pp. 104–113. AAAI Press.

Saini, N., & Saha, S. (2021). Multi-objective optimization techniques: A survey of the state-of-the-art and applications. *The European Physical Journal Special Topics*, *230*(10), 2319–2335.

Santis, M. D., Eichfelder, G., Niebling, J., & Rocktäschel, S. (2020). Solving multiobjective mixed integer convex optimization problems. *SIAM Journal on Optimization*, *30*(4), 3122–3145.

Schaus, P., & Hartert, R. (2013). Multi-objective large neighborhood search. In Schulte, C. (Ed.), *Principles and Practice of Constraint Programming—19th International Conference, CP 2013, Uppsala, Sweden, September 16–20, 2013. Proceedings*, Vol. 8124 of *Lecture Notes in Computer Science*, pp. 611–627. Springer.

Sengupta, R., & Saha, S. (2018). Reference point based archived many objective simulated annealing. *Information Sciences*, *467*, 725–749.

Smirnov, P., Berg, J., & Järvisalo, M. (2021). Pseudo-boolean optimization by implicit hitting sets. In Michel, L. D. (Ed.), *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, Vol. 210 of *LIPIcs*, pp. 51:1–51:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Smirnov, P., Berg, J., & Järvisalo, M. (2022). Improvements to the implicit hitting set approach to pseudo-boolean optimization. In Meel, K. S., & Strichman, O. (Eds.), *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, Vol. 236 of *LIPIcs*, pp. 13:1–13:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Soh, T., Banbara, M., Tamura, N., & Le Berre, D. (2017). Solving multiobjective discrete optimization problems with propositional minimal model generation. In Beck, J. C. (Ed.), *Principles and Practice of Constraint Programming—23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 – September 1, 2017, Proceedings*, Vol. 10416 of *Lecture Notes in Computer Science*, pp. 596–614. Springer.

Soland, R. M. (1979). Multicriteria optimization: A general characterization of efficient solutions. *Decision Sciences*, *10*(1), 26–38.

Storn, R., & Price, K. V. (1997). Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.

Sun, E. (2017). On optimization over the efficient set of a multiple objective linear programming problem. *Journal of Optimization Theory and Applications*, *172*(1), 236–246.

Tamura, N., Banbara, M., & Soh, T. (2013). Compiling pseudo-boolean constraints to SAT with order encoding. In *25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2013, Herndon, VA, USA, November 4–6, 2013*, pp. 1020–1027. IEEE Computer Society.

Terra-Neves, M., Lynce, I., & Manquinho, V. M. (2018a). Enhancing constraint-based multi-objective combinatorial optimization. In McIlraith, S. A., & Weinberger, K. Q. (Eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2–7, 2018*, pp. 6649–6656. AAAI Press.

Terra-Neves, M., Lynce, I., & Manquinho, V. M. (2018b). Multi-objective optimization through Pareto minimal correction subsets. In Lang, J. (Ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden*, pp. 5379–5383. ijcai.org.

Terra-Neves, M., Lynce, I., & Manquinho, V. M. (2018c). Stratification for constraint-based multi-objective combinatorial optimization. In Lang, J. (Ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden*, pp. 1376–1382. ijcai.org.

Tseitin, G. S. (1983). On the complexity of derivation in propositional calculus. In Siekmann, J. H., & Wrightson, G. (Eds.), *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, pp. 466–483, Berlin, Heidelberg. Springer.

Wassenhove, L. N. V., & Gelders, L. F. (1980). Solving a bicriterion scheduling problem. *European Journal of Operational Research*, *4*(1), 42–48.

Wolsey, L. A. (2020). *Integer Programming*. Wiley.

Yu, J., Ignatiev, A., Stuckey, P. J., & Bodic, P. L. (2021). Learning optimal decision sets and lists with SAT. *Journal of Artificial Intelligence Research*, *72*, 1251–1279.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—A comparative case study. In Eiben, A. E., Bäck, T., Schoenauer, M., & Schwefel, H. (Eds.), *Parallel Problem Solving from Nature—PPSN V, 5th International Conference, Amsterdam, The Netherlands, September 27–30, 1998, Proceedings*, Vol. 1498 of *Lecture Notes in Computer Science*, pp. 292–304. Springer.