Certifying Pareto-Optimality in Multi-Objective Maximum Satisfiability

Christoph Jabs[1], Jeremias Berg[1], Bart Bogaerts[2,3], and Matti Järvisalo[1]

[1] University of Helsinki, Helsinki, Finland
{christoph.jabs, jeremias.berg, matti.jarvisalo}@helsinki.fi
[2] KU Leuven, Leuven, Belgium
bart.bogaerts@kuleuven.be
[3] Vrije Universiteit Brussel, Brussels, Belgium

Abstract. Due to the wide employment of automated reasoning in the analysis and construction of correct systems, the results reported by automated reasoning engines must be trustworthy. For Boolean satisfiability (SAT) solvers—and more recently SAT-based maximum satisfiability (MaxSAT) solvers—trustworthiness is obtained by integrating proof logging into solvers, making solvers capable of emitting machine-verifiable proofs to certify correctness of the reasoning steps performed. In this work, we enable for the first time proof logging based on the VeriPB proof format for multi-objective MaxSAT (MO-MaxSAT) optimization techniques. Although VeriPB does not offer direct support for multi-objective problems, we detail how preorders in VeriPB can be used to provide certificates for MO-MaxSAT algorithms computing a representative solution for each element in the non-dominated set of the search space under Pareto-optimality, without extending the VeriPB format or the proof checker. By implementing VeriPB proof logging into a state-of-the-art multi-objective MaxSAT solver, we show empirically that proof logging can be made scalable for MO-MaxSAT with reasonable overhead.

Keywords: Multi-objective combinatorial optimization · maximum satisfiability · proof logging

1 Introduction

Automated reasoning is central in enabling the analysis and construction of correct systems. Practical solvers developed in the realm of automated reasoning, such as Boolean satisfiability (SAT) solvers [9], facilitate the development of more complex automated reasoning systems. One successful example of such generic SAT-based approaches are solvers developed for maximum satisfiability (MaxSAT) [3]—the optimization extension of SAT—enabling solving various NP-hard real-world optimization problems [3]. Further, SAT-based approaches are being generalized and developed for MaxSAT under multiple objectives, i.e., multi-objective MaxSAT [11,13,14,29,35,36,37,49,50], with the aim of extending the success of MaxSAT to more efficiently solving real-world multi-objective

optimization problems, from, e.g., staff scheduling [18] through package upgradeability [39] to finding interpretable classifiers [44].

The more SAT and MaxSAT solvers are used in real-world settings, the more important it is to be able to trust the results solvers provide. While solutions are generally easy to confirm, solvers should be trustworthy also when they report unsatisfiability or, in the context of optimization, when the solvers claim that a solution is optimal and hence no better solutions exist. In response to these concerns, proof logging and checking techniques for SAT solvers have been developed and widely adopted [28,15,16], among which DRAT [30,31] remains today the de facto standard in the context of SAT solving. However, DRAT and other SAT proof formats work purely on propositional clauses, which makes them unsuitable for proof logging MaxSAT solvers. Instead, the VeriPB format [27,10], which is based on pseudo-Boolean constraints (i.e., 0-1 linear inequalities) and offers direct support for reasoning about objective values in single-objective optimization problems, has enabled proof logging for various optimization contexts [7,10,17,22,23,24,25,26,27,32,33,52], including MaxSAT solving [6,7,33,51,52].

In this work, we enable proof logging for various recently-proposed multi-objective MaxSAT solving techniques. To the best of our knowledge, this is the first work enabling proof logging in multi-objective optimization.[4] Our solution builds on the VeriPB format. It is critical to note that VeriPB does not offer direct support for multiple objective functions, and is thereby seemingly restricted to proof logging single-objective optimization algorithms. However, as we will detail, proof logging for MO-MaxSAT can in fact be enabled without extending the VeriPB format or the proof checker. In particular, in order to provide certificates for MO-MaxSAT algorithms developed for computing a representative solution for each element in the so-called non-dominated set of the search space under Pareto-optimality [21], we make in a specific way use of preorders supported by VeriPB. While preorders were first introduced to VeriPB for certifying symmetry and dominance breaking [10], here we show that, in fact, a single preorder suffices for certifying that an MO-MaxSAT algorithm has computed a representative solution at each element of the non-dominated set. As representative MO-MaxSAT techniques, we detail VeriPB-based proof logging for variants of the P-minimal [43,49], BiOptSat [37], and LowerBound [14] approaches, as well as the recently-proposed MO-MaxSAT preprocessing/reformulation technique of core boosting [36] which has been shown to provide considerable runtime improvements to MO-MaxSAT solvers. By adding VeriPB proof logging to the implementations of these approaches in the MO-MaxSAT solver Scuttle [34,35,36], we show empirically that proof logging can be made scalable for MO-MaxSAT, with average proof logging overhead ranging from 14% to 29% depending on the solving approach.

[4] Although relatively distant to the present work, there is some work on certificates in the context of multi-objective queries in Markov decision processes [4].

2 Preliminaries

We begin with necessary preliminaries related to multi-objective MaxSAT and VeriPB proofs.

2.1 Clauses and Pseudo-Boolean Constraints

A literal ℓ is a $\{0,1\}$-valued Boolean variable x or its negation $\overline{x} \equiv 1 - x$. A propositional clause $C = (\ell_1 \vee \ldots \vee \ell_k)$ is a disjunction of literals. A formula in conjunctive normal form (CNF) $F = C_1 \wedge \ldots \wedge C_m$ is a conjunction of clauses. We often think of clauses as sets of literals and formulas as sets of clauses.

A (normalized) pseudo-Boolean (PB) constraint is a 0-1 linear inequality $C = \sum_i a_i \ell_i \geq b$ where a_i are positive integers and b a non-negative integer. We will assume wlog that all PB constraints are in *normal form*, meaning that the ℓ_i are over different variables and all coefficients a_i and the bound are positive. A pseudo-Boolean (PB) formula is a conjunction (or set) of PB constraints. We identify the propositional clause $C = (\ell_1 \vee \ldots \vee \ell_k)$ with the PB constraint $\sum_{i=1}^{k} \ell_i \geq 1$. This is convenient as the algorithms and solvers that we develop proof logging for expect propositional clauses as input, and—as detailed in this work—produce their proofs in pseudo-Boolean format. If C is the PB constraint $\sum_i a_i \ell_i \geq b$, we write $\neg C$ for its negation $\sum_i a_i \overline{\ell_i} \geq \sum_i a_i - b + 1$. If p is furthermore a variable, we write $p \Leftrightarrow C$ for the two constraints expressing that p implies C and vice versa, i.e., $Mp + \sum_i a_i \overline{\ell_i} \geq M$ and $b\overline{p} + \sum_i a_i \ell_i \geq b$ where $M = \sum_i a_i - b + 1$. An *objective* O is an expression $\sum_i a_i \ell_i + L$ where the a_i and L are integers.

A substitution ω maps each variable in its domain to a truth value (either 0 or 1) or to another literal. We denote by $C\restriction_\omega$ the constraint obtained from C by replacing each variable x in the domain of ω by $\omega(x)$; the notations $O\restriction_\omega$, $F\restriction_\omega$, and $\vec{x}\restriction_\omega$ for a tuple of variables $\vec{x}$ are defined analogously. An assignment α is a substitution that maps only onto $\{0,1\}$. When convenient, we view an assignment as the set of literals it sets to 1. An assignment α is *complete* for a constraint, formula, or objective if α maps each variable in them to a value, and partial otherwise. The assignment α satisfies a constraint C if the constraint $C\restriction_\alpha = \sum_i a'_i \ell'_i \geq b'$ obtained after normalization has $b' = 0$, and falsifies C if $\sum_i a'_i < b'$. In other words, α satisfies C if simplifying C by α leads to a trivial constraint and falsifies C if no extension of α satisfies C. An assignment α is a solution to a formula F if α satisfies all constraints in F. A constraint C is *implied* by F (denoted by $F \models C$) if all solutions of F also satisfy C.

2.2 Multi-Objective MaxSAT

An instance $(F, \mathbf{O})$ of multi-objective MaxSAT (MO-MaxSAT) consists of a CNF formula F and a set $\mathbf{O} = (O_1, \ldots, O_p)$ of p objectives under minimization. This definition for MO-MaxSAT captures standard (single-objective, weighted partial) MaxSAT by setting $p = 1$; see, e.g., [37]. Given two assignments α and β

$$F = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee x_5) \wedge (x_3 \vee x_4 \vee x_5)$$
$$O_1 = 3x_2 + 4x_3 + 2x_4 + 5x_5$$
$$O_2 = 7x_1 + 4x_2 + 1x_3 + 2x_4$$

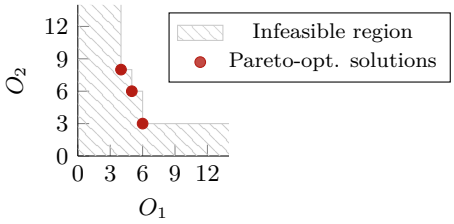

Fig. 1. A bi-objective MaxSAT instance and its Pareto-optimal solutions.

that are complete for each O_i, we say that α *weakly dominates* β (and write $\alpha \preceq_P \beta$) if $O_i\lfloor_\alpha \leq O_i\lfloor_\beta$ holds for each $i = 1, \ldots, p$. Note that $O\lfloor_\alpha$ is an integer value when α is complete for O. If additionally $O_t\lfloor_\alpha < O_t\lfloor_\beta$ for some t, we say that α *dominates* β (and write $\alpha \prec_P \beta$). A solution α to F is Pareto-optimal for $(F, \mathbf{O})$ if α is not dominated by any other solution to F. The *non-dominated set* of $(F, \mathbf{O})$ consists of the tuples of objective values of Pareto-optimal solutions, i.e., $\{(O_1\lfloor_\alpha, \ldots, O_p\lfloor_\alpha) \mid \alpha \text{ is Pareto-optimal for } (F, \mathbf{O})\}$. Note that each element in the non-dominated set can correspond to several Pareto-optimal solutions. We focus on the task of computing a representative solution for each element of the non-dominated set.

Example 1. Consider the bi-objective MaxSAT instance $(F, (O_1, O_2))$ in Fig. 1. Its non-dominated set is $\{(4, 8), (5, 6), (6, 3)\}$. Its three Pareto-optimal solutions are $\{x_1, \overline{x_2}, x_3, \overline{x_4}, \overline{x_5}\}$, $\{\overline{x_1}, x_2, \overline{x_3}, x_4, \overline{x_5}\}$, and $\{\overline{x_1}, \overline{x_2}, x_3, x_4, \overline{x_5}\}$.

2.3 Multi-Objective MaxSAT Solving

We consider the multi-objective MaxSAT problem of finding a representative solution of each element in the non-dominated set. Various algorithms for this problem setting have been proposed recently [13,14,37,49,50]. These algorithms make incremental use of a SAT solver [19,45] while adding constraints to the working formula, ending with an unsatisfiable working formula once all elements in the non-dominated set have been discovered.

For many of the existing algorithms for MO-MaxSAT, a crucial building block is what we call a Pareto-dominance cut, or PD cut for short. A PD cut is a (set of) constraint(s) that, given a solution α, is falsified exactly by all solutions that are weakly dominated by α (including α itself). Adding a PD cut to the working formula therefore excludes solutions weakly dominated by α from further consideration. Note that for the single-objective case ($p = 1$), a PD cut is identical to a solution-improving constraint, admitting only "better" solutions. This no longer holds when $p > 1$ since Pareto-dominance is not a total order: solutions that are incomparable to α will satisfy the PD cut.

For objectives $\mathbf{O} = (O_1, \ldots, O_p)$ and solution α, let w_i be fresh variables for each O_i with their semantics defined by

$$w_i \Leftrightarrow O_i \geq O_i\lfloor_\alpha \,. \tag{1}$$

The PD cut is the clause $(\overline{w_1} \vee \cdots \vee \overline{w_p})$. For encoding (1), MO-MaxSAT algorithms use a choice of various CNF encodings [5,20,41,42,46]. We will return in more detail to how PD cuts are employed in the P-minimal [43,49], LowerBound [14], and BiOptSat [37] algorithms in Section 3 when detailing VeriPB proof logging for each of these algorithms.

2.4 VeriPB

We now overview a simplified version of the VeriPB proof system, only discussing the rules that are relevant for our current exposition. For instance, while VeriPB supports *single-objective* optimization, we will only use the decision version of this proof system. We refer the interested reader to earlier work [10,27] for an exposition of the full proof system.

Given a PB input formula F, the VeriPB proof system maintains a *proof configuration* $\langle \mathcal{C}, \mathcal{D}, \mathcal{O}, \vec{z} \rangle$, consisting of two sets of constraints, the *core set* $\mathcal{C}$ and the *derived set* $\mathcal{D}$, a pseudo-Boolean formula $\mathcal{O}(\vec{u}, \vec{v})$ over two tuples of variables $\vec{u}, \vec{v}$ that do not appear in $\mathcal{C}$, and a tuple of variables $\vec{z}$. The core set can be thought of as being equal to F and the derived set as consisting of all constraints derived in the proof. The order $\mathcal{O}$ defines a preorder $\preceq$ on assignments as follows. If α and β are assignments, then $\alpha \preceq \beta$ iff $\mathcal{O}(\vec{z}\restriction_\alpha, \vec{z}\restriction_\beta)$ is true. The proof system will guarantee that $\preceq$ is indeed a preorder, i.e., a reflexive and transitive relation. The preorder in this configuration was originally introduced in the context of symmetry and dominance breaking [10]. Here we will use the preorder for a different purpose and will in fact not use the dominance rule introduced in [10]. The precise role of the preorder in our proofs for MO-MaxSAT will be detailed in Section 3.

The configuration is initialized by setting $\mathcal{C} = F$, $\mathcal{D} = \emptyset$, $\mathcal{O} = \emptyset$ (the empty, and hence trivially true formula) and $\vec{z} = ()$, the empty tuple. Afterwards, the configuration is updated using the rules detailed next.

New constraints can be added to $\mathcal{D}$ by deriving them from previously derived constraints in $\mathcal{C} \cup \mathcal{D}$ using the *cutting planes* proof system [12] consisting of the following rules.

Literal Axioms. For any literal ℓ, $\ell \geq 0$ is an axiom and can be derived.
Linear Combination. Any positive integer linear combination of two previously derived PB constraints can be inferred.
Division. Given the normalized PB constraint $\sum_i w_i \ell_i \geq A$ and a positive integer c, the constraint $\sum_i \lceil w_i/c \rceil \ell_i \geq \lceil A/c \rceil$ can be inferred.

Conveniently, VeriPB also allows adding an implied constraint without giving an actual cutting planes derivation, namely, when the constraint is implied by *reverse unit propagation (RUP)*, a generalization of the same notion in SAT [28]. RUP states that if applying integer bounds consistency propagation on $\mathcal{C} \cup \mathcal{D} \cup \{\neg C\}$ results in a contradiction, then C is implied by $\mathcal{C} \cup \mathcal{D}$ and can hence be derived.

Additionally, VeriPB also allows for deriving non-implied constraints as long as the constraints are guaranteed to preserve satisfiability. Specifically VeriPB

has the following generalization of the resolution asymmetric tautology (RAT) rule in SAT [40].

Redundance-Based Strengthening. The constraint C can be derived and added to $\mathcal{D}$ given a substitution ω and explicit (cutting planes) proofs for

$$\mathcal{C} \cup \mathcal{D} \cup \{\neg C\} \models (\mathcal{C} \cup \mathcal{D} \cup \{C\})\restriction_\omega \cup \mathcal{O}(\vec{z}\restriction_\omega, \vec{z}),$$

i.e., explicit proofs that the constraint on the right is implied by the premises $\mathcal{C} \cup \mathcal{D} \cup \{\neg C\}$.

Intuitively, this rule ensures that ω remaps any solution α of $\mathcal{C} \cup \mathcal{D}$ that does not satisfy C to a solution $\alpha \circ \omega$ of $\mathcal{C} \cup \mathcal{D}$ that (i) satisfies C and (ii) for which $\alpha \circ \omega \preceq \alpha$, i.e., $\alpha \circ \omega$ is least as good (in terms of the order) as α. A common application of redundance-based strengthening is *reification*: deriving two pseudo-Boolean constraints that encode $\ell \Leftrightarrow D$ for some PB constraint $D \in \mathcal{C} \cup \mathcal{D}$ and for some fresh literal ℓ.

In addition to adding constraints, previously derived constraints can also be *deleted* in order to reduce the number of constraints that the proof checker has to work with. However, deletion requires care. Without restrictions, deleting everything in $\mathcal{C}$ could make an unsatisfiable formula satisfiable, which would clearly be incorrect. Deletion is allowed using the following rules.

Derived Deletion. Any constraint can be removed from $\mathcal{D}$.

Core Deletion. A constraint C can be removed from $\mathcal{C}$ if C can be derived from $\mathcal{C} \setminus \{C\}$ with the redundance-based strengthening rule.

Solutions found are logged and excluded from further consideration by learning a constraint using the following rule.

Solution Logging. Given a solution α to $\mathcal{C} \cup \mathcal{D}$, we can derive the constraint $\sum_{\ell \in \alpha} \overline{\ell} \geq 1$ that excludes α and add this constraint to $\mathcal{C}$.

Constraints can always be moved from the derived set to the core set using the transfer rule in order to allow e.g. the application of core deletion.

Transfer Rule. If $\mathcal{D}' \subseteq \mathcal{D}$, we can transfer from configuration $\langle \mathcal{C}, \mathcal{D}, \mathcal{O}, \vec{z} \rangle$ to $\langle \mathcal{C} \cup (\mathcal{D} \setminus \mathcal{D}'), \mathcal{D}', \mathcal{O}, \vec{z} \rangle$.

Finally, the order can be changed, provided that the derived set is empty (which can always be achieved using the transfer rule).

Order Change Rule. Given a proof in VeriPB format that $\mathcal{O}'$ is *reflexive* (i.e., $\mathcal{O}'(\vec{u}, \vec{u})$ is trivial) and transitive (i.e., whenever $\mathcal{O}'(\vec{u}, \vec{v})$ and $\mathcal{O}'(\vec{v}, \vec{w})$ hold, so does $\mathcal{O}'(\vec{u}, \vec{w})$), and given a tuple of variables $\vec{z}'$ of the right length, we can transition from $\langle \mathcal{C}, \emptyset, \mathcal{O}, \vec{z} \rangle$ to $\langle \mathcal{C}, \emptyset, \mathcal{O}', \vec{z}' \rangle$.

3 Proof Logging for Multi-Objective MaxSAT

As our main contributions, we will now detail how VeriPB can be used for enabling proof logging in the multi-objective setting—despite the fact that VeriPB does not directly support multiple objectives. Our solution is based on a new type of use of the preorder $\mathcal{O}$ in VeriPB.

3.1 The General Setup

Preorders were originally introduced in VeriPB to enable proofs for symmetry breaking [10]. However, the preorder turns out to be applicable for multi-objective proof logging as well. Since all rules in VeriPB are guaranteed to preserve solutions that are minimal with respect to the defined preorder, the preorder generalizes a single objective O: computing a solution optimal wrt O is equivalent to computing a solution that is smallest in the order $\mathcal{O}^O$ defined by the formula that is true iff $O\lfloor_\alpha \leq O\lfloor_\beta$. As a first step towards the multi-objective setting, we introduce a suitable order for encoding Pareto-dominance. In the following definition, if $\vec{u}$ and $\vec{v}$ are two tuples of variables of equal length, we write $\omega_{\vec{u}\to\vec{v}}$ for the substitution that maps every u_i to v_i and all other variables to themselves.

Definition 1. *Let $\mathbf{O} = (O_1, \ldots, O_p)$ be a tuple of p objectives over variables $\vec{x} = (x_1, \ldots, x_k)$, and define $\mathcal{O}_P^{\mathbf{O}}(\vec{u}, \vec{v})$ over fresh variables $\vec{u}$ and $\vec{v}$ as the PB formula $\mathcal{O}_P^{\mathbf{O}}(\vec{u}, \vec{v}) = \{O_1\lfloor_{\omega_{\vec{x}\to\vec{u}}} \leq O_1\lfloor_{\omega_{\vec{x}\to\vec{v}}}, \ldots, O_p\lfloor_{\omega_{\vec{x}\to\vec{u}}} \leq O_p\lfloor_{\omega_{\vec{x}\to\vec{v}}}\}$.*

The following proposition summarizes the properties of $\mathcal{O}_P^{\mathbf{O}}$ that are important for our setting.

Proposition 1. *Let $\mathbf{O} = (O_1, \ldots, O_p)$ be a tuple of objectives and $\mathcal{O}_P^{\mathbf{O}}(\vec{u}, \vec{v})$ the PB formula from Definition 1. Then the following hold:*

- *$\mathcal{O}_P^{\mathbf{O}}$ encodes a preorder, i.e., a reflexive ($\mathcal{O}_P^{\mathbf{O}}(\vec{u}, \vec{u})$ is trivially satisfied) and transitive (if $\mathcal{O}_P^{\mathbf{O}}(\vec{u}, \vec{v})$ and $\mathcal{O}_P^{\mathbf{O}}(\vec{v}, \vec{w})$ also $\mathcal{O}_P^{\mathbf{O}}(\vec{u}, \vec{w})$ holds) relation.*
- *$\mathcal{O}_P^{\mathbf{O}}(\vec{x}\lfloor_\alpha, \vec{x}\lfloor_\beta)$ is satisfied if and only if $\alpha \preceq_P \beta$ wrt $\mathbf{O}$, i.e., if α weakly dominates β.*

When the objectives are clear from context, we drop the superscript and use $\mathcal{O}_P(\vec{u}, \vec{v})$ for the order that encodes Pareto-dominance over the objectives.

With the transfer and order change rules, the order can be changed arbitrarily in VeriPB proofs. In our setting, however, we will use $\mathcal{O}_P$ unchanged throughout the entire proof. From now on, a *VeriPB proof for* $(F, \mathbf{O})$ refers to a standard VeriPB proof for F that (i) as the first derivation step loads the order $\mathcal{O}_P$ over the variables $\vec{x}$ in the objectives $\mathbf{O}$, and (ii) at no other point in the proof changes the order. Our observations on valid VeriPB proofs assume that the conditions (i) and (ii) are satisfied. The VeriPB proof checker will not verify these two conditions for us; however, these are merely syntactic restrictions that can be verified easily (e.g., by checking that no other lines in the proof starts with `load_order`).

The following result now guarantees the correctness of the proofs produced for the different MO-MaxSAT algorithms.

Theorem 1. *Let P be a VeriPB proof for $(F, \mathbf{O})$ that derives a contradiction. Let S be the set of non-dominated solutions logged in P, i.e., logged solutions that are not dominated by other solutions logged in P. Then S contains a representative solution for each element in the non-dominated set of $(F, \mathbf{O})$.*

Let $\langle \mathcal{C}_i, \mathcal{D}_i, \mathcal{O}_i, \vec{z}_i \rangle$ be the proof configuration of P at step i. From the additional conditions imposed for multi-objective proofs, we have $\mathcal{O}_i = \mathcal{O}_P$ and $\vec{z}_i = \vec{x}$ for all $i \geq 1$, since $\mathcal{O}_P$ over the variables $\vec{x}$ in $\mathbf{O}$ is loaded in the first derivation step.

Theorem 1 follows by the following lemmas. Lemma 1 is a restatement of the properties of VeriPB proofs shown in [10], included here for completeness.

Lemma 1. *Let $\langle \mathcal{C}_i, \mathcal{D}_i, \mathcal{O}_P, \vec{x} \rangle$ be the i^{th} configuration of P. For every solution α of $\mathcal{C}_i$, there exists a solution β of $\mathcal{C}_i \cup \mathcal{D}_i$ for which $\mathcal{O}_P(\vec{x}\restriction_\beta, \vec{x}\restriction_\alpha)$.*

The next lemma establishes that no rule in VeriPB can "create" new non-dominated points.

Lemma 2. *Consider the i^{th} configuration $\langle \mathcal{C}_i, \mathcal{D}_i, \mathcal{O}_P, \vec{x} \rangle$ of P for fixed $i \geq 1$. Any solution of $\mathcal{C}_i$ is weakly dominated by a Pareto-optimal solution of $(F, \mathbf{O})$.*

Proof. By induction on i. The base case $i = 1$ follows by Proposition 1 from the first configuration having $\mathcal{C}_1 = F$. Assume that the statement holds for $i - 1$ and let $\langle \mathcal{C}_{i-1}, \mathcal{D}_{i-1}, \mathcal{O}_P, \vec{x} \rangle$ be the $i{-}1^{\text{th}}$ configuration. The rules of VeriPB that can alter the core set are solution logging, core deletion, and the transfer rule. For the transfer rule, the statement follows immediately by Lemma 1. For solution logging, the result follows from any solution of $\mathcal{C}_i$ being a solution of $\mathcal{C}_{i-1}$. Assume thus that $\mathcal{C}_i = \mathcal{C}_{i-1} \setminus \{C\}$ and let α be a solution of $\mathcal{C}_i$, that (for the non-trivial case) does not satisfy C. We show that α is weakly dominated by a Pareto-optimal solution of $(F, \mathbf{O})$. By the properties of redundance-based strengthening and core deletion, there is a substitution ω such that $\beta = \alpha \circ \omega$ is a solution to $\mathcal{C}_{i-1}$ for which $\mathcal{O}_P(\vec{x}\restriction_\beta, \vec{x}\restriction_\alpha)$. By the induction assumption, β is weakly dominated by a Pareto-optimal solution γ of $(F, \mathbf{O})$. Since $\beta \preceq_P \alpha$, α is weakly dominated by γ. □

Lemma 3 establishes that no rule except for solution logging can remove all representative solutions for an element in the non-dominated set of $(F, \mathbf{O})$.

Lemma 3. *Let $\langle \mathcal{C}_{i-1}, \mathcal{D}_{i-1}, \mathcal{O}_P, \vec{x} \rangle$ and $\langle \mathcal{C}_i, \mathcal{D}_i, \mathcal{O}_P, \vec{x} \rangle$ be the $i{-}1^{th}$ and i^{th} configurations of P, respectively. Assume that (i) the solutions of $\mathcal{C}_{i-1} \cup \mathcal{D}_{i-1}$ include a representative for the non-dominated point $\mathbf{b}$ and (ii) the i^{th} configuration is obtained by a rule other than solution logging. Then the solutions of $\mathcal{C}_i \cup \mathcal{D}_i$ include a representative solution for $\mathbf{b}$.*

Proof. Let α be the representative solution to $\mathcal{C}_{i-1} \cup \mathcal{D}_{i-1}$ for $\mathbf{b}$. We construct a solution to $\mathcal{C}_i \cup \mathcal{D}_i$ that is representative for $\mathbf{b}$. For the interesting case, assume that α is not a solution to $\mathcal{C}_i \cup \mathcal{D}_i$. By assumption the i^{th} configuration was reached by a rule other than solution logging, and hence $\mathcal{D}_i = \mathcal{D}_{i-1} \cup \{C\}$ for a constraint C added using the redundance-based strengthening rule and a substitution ω. By redundance-based strengthening $\beta = \alpha \circ \omega$ is a solution to $\mathcal{C}_i \cup \mathcal{D}_i$ for which $\mathcal{O}_P(\vec{x}\restriction_\beta, \vec{x}\restriction_\alpha)$. Hence by Proposition 1, β weakly dominates α. As α is representative for the non-dominated point $\mathbf{b}$, β must be representative for $\mathbf{b}$ as well. □

Using these three lemmas we can now establish Theorem 1 as follows.

Proof (of Theorem 1). Consider the first configuration $\langle \mathcal{C}_1, \mathcal{D}_1, \mathcal{O}_P, \vec{x} \rangle$ of P, where $\mathcal{C}_1 = F$ and $\mathcal{D}_1 = \emptyset$. By Proposition 1, there is a one-to-one correspondence between the Pareto-optimal solutions of $(F, \mathbf{O})$ and the solutions of $\mathcal{C}_1 \cup \mathcal{D}_1$ that are minimal wrt $\mathcal{O}_P$. Specifically, the solutions of $\mathcal{C}_1 \cup \mathcal{D}_1$ contain a representative solution for each element in the non-dominated set of $(F, \mathbf{O})$. Since P derives a contradiction, there are no solutions to the union of the core and derived set of the final configuration. Thus the theorem holds if (i) no solutions that dominate a Pareto-optimal solution of $(F, \mathbf{O})$ are logged in the proof, and (ii) a representative solution for each element in the non-dominated set is logged in the proof. (i) follows by Lemma 2 and (ii) by Lemma 3. □

We note that Theorem 1 also holds when an instance $(F, (O))$ only has a single objective. In this case $\alpha \preceq_P \beta$ is equivalent to $O\restriction_\alpha \leq O\restriction_\beta$. Hence Theorem 1 states that the simplified version of VeriPB using only rules for decision problems (recall Section 2.4) is sufficient for certifying single-objective MaxSAT solvers as well. Thereby it can be argued that the (more complicated) optimization proof system presented in [10] could be simplified without losing any of its expressiveness by replacing the explicit linear objective and objective-specific rules by the linear order-based rules we describe.

3.2 Proof Logging for Pareto Dominance Cuts

A key step in proof logging the different multi-objective algorithms is the derivation of a constraint called a PD cut given a solution α. In particular, given a solution α of F, we will derive a constraint that states that we are no longer interested in solutions worse than or equally good as α in terms of Pareto-dominance. In single-objective VeriPB, there is a dedicated rule that allows for deriving a so-called solution-improving constraint. However we will show that PD cuts can be derived only relying on the redundance-based strengthening and solution logging rules.

We will make use of some auxiliary variables w_i for each objective O_i (recall Equation (1)). Firstly, introducing such (reified) constraints can be done with redundance-based strengthening in a standard way [27]. The introduced constraints guarantee that iff β is worse than or equal to α in O_i, then variable w_i holds in β. Now let ω_α be the substitution that maps every variable to their value in α and each w_i to 1. We claim that with the redundance-based strengthening rule and this witness we can derive the constraint

$$C_\alpha := \sum_{i=1}^{p} |\alpha| \cdot \overline{w_i} + \sum_{\ell \in \alpha} \ell \geq |\alpha|.$$

Intuitively, this constraint maps each solution weakly dominated by α to α. In doing so it excludes all solutions the PD cut excludes (the solutions weakly dominated by α) except for α itself. To see that this constraint can be derived with redundance-based strengthening, we can verify that all proof obligations are indeed met.

Table 1. Example proof for certifying a PD cut.

ID	Pseudo-Boolean Constraint	Comment	Justification
	Input constraints and potential previous proof steps		
$[a]$	$11w_1^\alpha + 3\overline{x_2} + 4\overline{x_3} + 2\overline{x_4} + 5\overline{x_5} \geq 11$	$w_1^\alpha \Leftarrow O_1 \geq O_1\restriction_\alpha$	Redundance $\{w_1^\alpha\}$
$[b]$	$4\overline{w_1^\alpha} + 3x_2 + 4x_3 + 2x_4 + 5x_5 \geq 4$	$w_1^\alpha \Rightarrow O_1 \geq O_1\restriction_\alpha$	Redundance $\{\overline{w_1^\alpha}\}$
$[c]$	$7w_2^\alpha + 7\overline{x_1} + 4\overline{x_2} + 1\overline{x_3} + 2\overline{x_4} \geq 7$	$w_2^\alpha \Leftarrow O_2 \geq O_2\restriction_\alpha$	Redundance $\{w_2^\alpha\}$
$[d]$	$8\overline{w_2^\alpha} + 7x_1 + 4x_2 + 1x_3 + 2x_4 \geq 8$	$w_2^\alpha \Rightarrow O_2 \geq O_2\restriction_\alpha$	Redundance $\{\overline{w_2^\alpha}\}$
$[e]$	$5\overline{w_1^\alpha} + 5\overline{w_2^\alpha} + x_1 + \overline{x_2} + x_3 + \overline{x_4} + \overline{x_5} \geq 5$	C_α	Redundance ω_α
$[f]$	$\overline{x_1} + x_2 + \overline{x_3} + x_4 + x_5 \geq 1$		Log solution α
$[g]$	$\overline{w_1^\alpha} + \overline{w_2^\alpha} \geq 1$	PD cut	$([e] + [f])/5$

- For each constraint C in $\mathcal{C} \cup \mathcal{D}$ that does not mention w_i, $C\restriction_{\omega_\alpha}$ is trivially satisfied since α is a solution that satisfies those constraints.
- If C is one of the constraints (1), then clearly $C\restriction_{\omega_\alpha}$ holds too.
- Clearly also $C_\alpha\restriction_{\omega_\alpha}$ holds.
- Lastly, what we need to show is that $\mathcal{O}(\vec{x}\restriction_{\omega_\alpha}, \vec{x})$ holds if C_α is not satisfied. This constraint expresses that α weakly dominates any assignment β that satisfies all derived constraints so far but not C_α. As such an assignment β must assign all w_i to 1, from (1) we immediately have that $O_i\restriction_\beta \geq O_i\restriction_\alpha$, as desired.

Finally, after deriving C_α, we log the solution α to obtain a solution-excluding constraint $\sum_{\ell \in \alpha} \overline{\ell} \geq 1$. By adding $\sum_{\ell \in \alpha} \overline{\ell} \geq 1$ to C_α, we arrive at $\sum_{i=1}^{p} |\alpha| \cdot \overline{w_i} \geq 1$. Hence at least one of the w_i must be 0. Dividing the result by $|\alpha|$ yields the PD cut $(\overline{w_1} \vee \cdots \vee \overline{w_p})$.

Example 2. Recall the instance in Fig. 1. Table 1 shows the steps required in the VeriPB proof for certifying a PD cut based on solution $\alpha = \{x_1, \overline{x_2}, x_3, \overline{x_4}, \overline{x_5}\}$ with objective values $(4, 8)$. Steps $[a]$, $[b]$, $[c]$, and $[d]$ first introduce the definitions of the w_1^α and w_2^α variables from Equation (1) as normalized reified PB constraints. These steps are justified by redundance-based strengthening using the fresh variables w_1^α and w_2^α as witnesses. Next, step $[e]$ introduces C_α. Lastly, the solution α is logged (in $[f]$) and the PD cut derived (in $[g]$).

3.3 Proof Logging Multi-Objective MaxSAT Algorithms

To enforce bounds on the values of the different objectives, multi-objective MaxSAT algorithms make use of CNF encodings of (reified) pseudo-Boolean constraints. For certifying the correctness of MO-MaxSAT algorithms, the correctness of these encodings needs to be certified as well. All algorithms covered in this paper make use of the incremental (generalized) totalizer encoding [5,41,46]. The totalizer encoding can be visualized as a tree where each node has a set of output literals that "count" how many of the literals at the leaves of its subtree are 1. In earlier work [52] it has been shown how the clauses of the totalizer

encoding can be derived in the VeriPB proof system from the constraints describing the semantics of the output and internal variables. For certifying the *generalized* totalizer encoding, the semantics of the output variables are slightly different. Each output variable o_b of a node is defined by the two constraints $o_b \Leftrightarrow \sum_{i=1}^{n} a_i \ell_i \geq b$ over the n literals at the leaves of the subtree. In contrast to the unweighted case, a_i can be larger than 1 here and values of b that are not subsetsums of $\{a_i \mid i = 1, \ldots, n\}$ are omitted. Deriving the clauses of the generalized totalizer encoding now follows the cutting planes procedure described in [52].

Many implementations of MaxSAT algorithms employing the totalizer encoding only derive clauses for enforcing the $o_b \Leftarrow \sum_{i=1}^{n} a_i \ell_i \geq b$ constraint, since these are enough for enforcing upper bounds on the objective values by setting o_b to 0. As observed in [37], for the generalized totalizer it is additionally necessary to enforce *all* output variables in the range $[b, \max\{a_i \mid i = 1, \ldots, n\})$ to 0 in order to enforce $\sum_{i=1}^{n} a_i \ell_i < b$. With this modification, also for the generalized totalizer encoding, deriving only one "direction" of clauses is enough from the perspective of the solver. In the proof, however, we will make use of both directions of the definition of these output variables; this means that a solution found by the SAT solver is not necessarily a solution to the constraints in the proof. When using a solution found by the SAT solver as a witness for the redundance-based strengthening rule for deriving a PD cut (recall Section 3.2), we therefore need to adjust the assignment to satisfy the stricter semantics of the proof first. This is done during proof generation by traversing through all nodes of the (generalized) totalizer encodings and manually assigning the output variables to values following the strict semantics described above. This adjusted assignment is still guaranteed to satisfy all clauses in the SAT solver.

Next, we detail three state-of-the-art MO-MaxSAT algorithms and how to generate proofs for them.

P-minimal [43,49]. Starting from any solution α, the P-minimal algorithm introduces a PD cut excluding all solutions that are weakly dominated by α. A SAT solver is then queried while temporarily enforcing the next-found solution to dominate α. Both steps are achieved using the generalized totalizer encoding. If no solution dominating the latest one can be found, the previous solution is guaranteed to be Pareto-optimal. In this case P-minimal drops the temporary constraints and starts over. If the working formula is unsatisfiable at this point, P-minimal terminates. An example of a search path of P-minimal in objective space is illustrated on the left-hand side of Fig. 2. The blue circles and red dots represent the solutions found by the SAT solver, with the red dots representing Pareto-optimal solutions. In such an execution, P-minimal introduces a PD cut for each of these solutions.

For proof logging P-minimal we certify the generalized totalizer objective encodings and the added PD cuts in the VeriPB proof (as already described). This allows a contradiction to be derived in the proof iff a PD cut was added for each element in the non-dominated set. The temporary constraints that enforce domination are not required in the proof since they are merely heuristics guiding

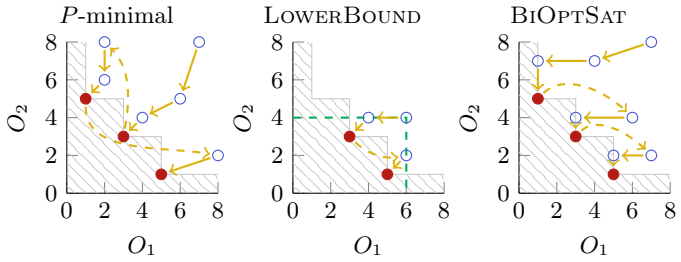

Fig. 2. Illustrations of the search path of MO-MaxSAT algorithms in objective space.

the SAT solver to certain regions temporarily and are not needed for reaching the final contradiction.

LowerBound [14]. The LowerBound algorithm restricts the search space by temporarily enforcing upper bounds of the form $O_i \leq b_i$ on each objective and then executes P-minimal within these bounds. Once P-minimal terminates, the bounds b_i are loosened and the process is repeated. LowerBound terminates once the bounds include the entire search space. In this last case, P-minimal is executed without the temporary bound constraints and will terminate with an unsatisfiable working instance. For proof logging LowerBound it therefore suffices to proof log the invocations of P-minimal as a subroutine. The search path for one invocation of P-minimal within a set of bounds is illustrated in green in Fig. 2 (middle).

BiOptSat [37]. The BiOptSat algorithm is specific to problems with two objectives (O_1, O_2). The algorithm enumerates non-dominated points under the guarantee that the values for one objective are monotonically increasing, while the values of the other objective are decreasing. BiOptSat first employs the subroutine `MinInc` to minimize O_1 without any additional constraints, returning the solution α. Next, the subroutine `MinDec` uses solution-improving search to minimize O_2 under the condition that $O_1 = O_1\lfloor_\alpha$. Let the final solution found by `MinDec` be β, which is guaranteed to be Pareto-optimal. BiOptSat then repeats this process after introducing the constraint $O_2 \leq O_2\lfloor_\beta - 1$. An example of a search path of BiOptSat is illustrated in Fig. 2 (right). Since `MinInc` finds the global minimum of O_1 for the current working formula, assume for now that we have a proof logging procedure for `MinInc` that results in the constraint $C_{\texttt{MinInc}} := O_1 \geq O_1\lfloor_\alpha$ being added to the proof. For any solution found during the execution of `MinDec`, we introduce a PD cut, which in turn is strengthened to the unit clause $(\overline{w_2})$ by combining it with $C_{\texttt{MinInc}}$. This strengthened PD cut is semantically equivalent to the constraint added during solution-improving search in `MinDec` and at the end before BiOptSat starts over.

Example 3. Recall again the instance in Fig. 1. Assume that we have executed `MinInc` and `MinDec` once already and that `MinDec` has yielded the Pareto-optimal

Table 2. Example proof for a strengthened PD cut in BiOptSat.

ID	Pseudo-Boolean Constraint	Comment	Justification
	Input constraint and potential previous proof steps		
$[lb]$	$3x_2 + 4x_3 + 2x_4 + 5x_5 \geq 4$		Derived during `MinInc`
$[a]$	$11w_1^\alpha + 3\overline{x_2} + 4\overline{x_3} + 2\overline{x_4} + 5\overline{x_5} \geq 11$	$w_1^\alpha \Leftarrow O_1 \geq O_1\restriction_\alpha$	Redundance $\{w_1^\alpha\}$
	Other PD cut certification steps from Table 1		
$[g]$	$\overline{w_1^\alpha} + \overline{w_2^\alpha} \geq 1$	PD cut	See Table 1
$[h]$	$w_2^\alpha \geq 1$		$(([lb] + [a])/11) + [g]$

$\alpha = \{x_1, \overline{x_2}, x_3, \overline{x_4}, \overline{x_5}\}$. Table 2 details the proof steps taken to certify the constraint $O_2 < O_2\restriction_\alpha$ added as the last step in BiOptSat. First, assume that $[lb]$ is $C_{\texttt{MinInc}}$ derived in the proof during `MinInc`. Next, we certify the PD cut for α in the proof as detailed in Table 1. Finally, by summing up the lower-bounding constraint from `MinInc`, the definition of w_1^α and the PD cut, we obtain $\overline{w_2^\alpha} \geq 1$ which is semantically equivalent to the constraint added by BiOptSat.

It remains to show that we can proof log `MinInc` to derive $C_{\texttt{MinInc}}$. The details depend on the implementation of `MinInc` for which several variants have been proposed [37]. For the SAT-UNSAT variant, where `MinInc` is implemented as solution-improving search, $C_{\texttt{MinInc}}$ is implicitly derived by the SAT solver during the final unsatisfiable query. Proof logging for the core-guided variants of `MinInc` can be implemented as already described for the single-objective setting in [7]; in short, $C_{\texttt{MinInc}}$ can be derived from the final reformulated objective.

3.4 Proof Logging Core Boosting

Core boosting [36] is a recently-proposed preprocessing/reformulation technique for MO-MaxSAT, consisting of applying the single-objective core-guided optimization algorithm OLL [2,47] wrt each objective individually, before executing an MO-MaxSAT algorithm on the reformulated objectives obtained from OLL. With this, core boosting shrinks the search space that needs to be considered by the MO-MaxSAT algorithm by deriving lower bounds for each objective. Core boosting also alters the structure of the CNF objective encodings since the totalizer structures built by OLL during core boosting can be reused in the objective encodings built by the MO-MaxSAT algorithm.

We give a brief overview of the single-objective core-guided OLL MaxSAT algorithm to the extent relevant for understanding how proof logging for core boosting works. Given an objective O, OLL invokes a SAT solver with the assumptions that none of the literals in O incurs cost. If these assumptions are not satisfiable, the SAT solver returns an implied clause C—referred to as an unsatisfiable core—over the objective literals. OLL now introduces counting variables $o_i \Leftarrow \sum_{\ell \in C} \ell \geq i$ for $i = 2, \ldots, |C|$ (encoded by the totalizer encoding) and reformulates the objective by adding $c_C \cdot (\sum_{\ell \in C} -\ell + \sum_{i=2}^{|C|} o_i + 1)$ to it, where c_C is the minimum objective coefficient of any literals in C. Iteratively applying

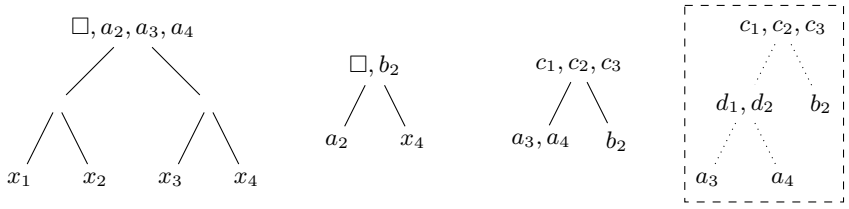

Fig. 3. An objective encoding structure built by core boosting. The dashed box shows the alternative structure that would be built without reusing a_3, a_4 as an internal node.

this process guarantees that the reformulated objective is always equal to the original objective and OLL terminates once there is a solution that does not incur cost on any of the literals in the reformulated objective.

Example 4. Let $O = x_1+x_2+x_3+2x_4$ be one of the objectives that core boosting is performed on. Assume the first core extracted is $C_1 = (x_1 \vee x_2 \vee x_3 \vee x_4)$. OLL reformulates C_1 by adding counting variables $a_i \Leftarrow x_1 + x_2 + x_3 + x_4 \geq i$ for $i = 2, 3, 4$. Assume the next core extracted is $C_2 = (x_4 \vee a_2)$, which is reformulated by adding the counting variable $b_2 \Leftarrow x_4 + a_2 \geq 2$. The final reformulated objective is $O' = a_3 + a_4 + b_2 + 2$.

Proof logging for OLL is described in [7], yielding a constraint of form $O \geq O'$ associating the reformulated objective O' with the original objective O. After OLL has been executed, core boosting builds a CNF encoding for each reformulated objective. However, if the reformulated objective contains a sequence of literals that are outputs for the same totalizer, they should be reused as an internal node of the final encoding employed by the MO-MaxSAT algorithm rather than individual leaves. This avoids introducing new auxiliary variables that would end up being equivalent to the variables introduced by OLL.

Example 5. Fig. 3 (left and middle-left, respectively) shows the two totalizer encodings built by OLL, where □ denotes the output variable with value 1 that is omitted by OLL. Fig. 3 (middle-right) shows the encoding of the reformulated objective built by core boosting after executing OLL, where a_3, a_4 is reused as an internal node. The dashed box shows the encoding that would be built when treating a_3 and a_4 individually. Since a_3 and a_4 are already totalizer outputs, d_1 and d_2 in this structure are equivalent to a_3 and a_4 and therefore redundant.

Going beyond previous work, certifying the encoding of the reformulated objective built during core boosting requires special care. In particular, due to the process of reusing partial sequences of totalizer output variables as internal nodes, using the semantics of the totalizer outputs built by OLL does *not* allow us to derive the clauses required for the encoding. Instead, for a sequence of variables $o_r, \ldots, o_{r+n}$ reused as an internal node in the encoding, we introduce

ordering constraints

$$C_v^a := \sum_{i=r}^{v} [\overline{o_i} \geq 0] + \sum_{i=v+1}^{r+n} [\overline{o_i} + o_v \geq 1] = \left[M o_v + \sum_{i=r}^{r+n} \overline{o_i} \geq M \right],$$

where $M = r + n - v + 1$, and

$$C_v^b := \sum_{i=v+1}^{r+n} [o_i \geq 0] + \sum_{i=r}^{v-1} [\overline{o_v} + o_i \geq 1] = \left[(v-r)\overline{o_v} + \sum_{i=r}^{r+n} o_i \geq (v-r) \right],$$

for each $v = r, \ldots, r+n$. These constraints sum up axioms and ordering constraints, which can be derived from the semantic definitions of the totalizer output variables, and are therefore derivable in the proof. Furthermore, C_v^a, C_v^b are identical to $o_v \Leftrightarrow \sum_{i=r}^{r+n} o_i \geq (v-r)$. When deriving the clauses involving the reused output variable sequence, we therefore use C_v^a and C_v^b instead of the semantic definitions of the variables, which allows for deriving the clauses of the encoding.

After core boosting, the output variables of the objective encoding are now defined with respect to the *reformulated* objective rather than the original one. As a final step, after certifying a PD cut with respect to the original objectives, we therefore use the objective reformulation constraints derived while executing OLL to certify the PD cut with respect to the reformulated objectives.

Example 6. All required clauses for the encoding shown in Fig. 3 (middle-right) can be derived from the semantic definitions $c_i \Leftrightarrow a_3 + a_4 + b_2 \geq i$ for $i = 1, 2, 3$ while treating the individual variable b_2 as a leaf and using C^a and C^b as "pseudo semantics" for the node a_3, a_4. Note that the semantics for c_i are with respect to the variable part of the reformulated objective.

4 Experiments

We extended all algorithms implemented in the Scuttle [34,35,36] MO-MaxSAT solver—namely, P-minimal [43,49], LowerBound [14], and (the SAT-UNSAT variant of) BiOptSat [37], each with and without core boosting—with the just-described VeriPB proof logging. We used CaDiCaL 2.0.0 [8] as the SAT solver within Scuttle and the VeriPB 2.2.2 proof checker [1] for checking the produced proofs.[5] The proof logging Scuttle implementation is available in open source [34,38]. We evaluate the implementation on the same set of benchmark instances used in the original work proposing core boosting [36]. This set of benchmarks consists of 300 instances from 6 domains with the number of objectives ranging from 2 to 5. The experiments were run on 2.50-GHz Intel Xeon Gold 6248 machines with 381-GB RAM in RHEL under a 32-GB memory limit and 1 hour time limit for Scuttle.

[5] Both VeriPB and CaDiCaL contain bug-fixes obtained directly from their respective authors based on our reporting while preparing our experiments.

Table 3. Average proof logging overheads and average proof checking overheads.

Algorithm	with core boosting		without core boosting	
	$\frac{\text{solving w/proof log}}{\text{solving}}$	$\frac{\text{proof checking}}{\text{solving w/proof log}}$	$\frac{\text{solving w/proof log}}{\text{solving}}$	$\frac{\text{proof checking}}{\text{solving w/proof log}}$
P-minimal	1.233	47.52	1.176	47.89
BiOptSat	1.247	29.70	1.140	18.70
LowerBound	1.220	21.81	1.294	21.78

The per-instance Scuttle runtimes for each benchmark domain with and without proof logging are shown in Fig. 4 for all three MO-MaxSAT algorithms in Scuttle, both with and without core boosting. We observe that the runtime overhead of proof logging is in all cases quite tolerable, with average runtime increase ranging from 14% to 29% depending on the algorithm; see Table 3 for details. There are at most 3 instances (for P-minimal without core boosting) that were only solved without proof logging within the given 1-h time limit. While this work is *not* focussed on improving proof *checking* but rather realizing for the first time proof logging in a multi-objective setting, Table 3 also includes the proof checking overhead, i.e., (proof checking time)/(Scuttle runtime with proof logging), resulting from checking the Scuttle proofs with the VeriPB checker. With a time limit of 10 hours enforced for the VeriPB checker, we observed that checking takes on average 1–1.5 orders of magnitude more time than solving the instances with proof logging enabled. It should be noted that similar observations have been made, e.g., in the realm of VeriPB-based certified MaxSAT preprocessing [33]. Indeed, these observations motivate seeking improvements to the current runtime performance of the VeriPB checker. We observed that in cases, in particular for P-minimal (see the appendix for more details), the proof checking overhead appears to somewhat correlate with the number of PD cuts produced during search.

5 Conclusions

We realized for the first time proof logging for multi-objective MaxSAT solving. Circumventing the fact that VeriPB does not offer direct support for multiple objectives, we detailed how preorders in VeriPB can be used to provide certificates for MO-MaxSAT algorithms that compute a representative solution for each element of the non-dominated set (with respect to the Pareto order). We achieved this without changes to the VeriPB format or the proof checker. Integrating VeriPB proof logging into a state-of-the-art multi-objective MaxSAT solver, we empirically showed that proof logging can be made scalable for MO-MaxSAT. While we in this work detailed how VeriPB can be employed for proof logging SAT-based multi-objective approaches, the same concepts are applicable to enabling proof logging for similar algorithmic ideas instantiated for other contexts, e.g., in the context of pseudo-Boolean optimization. Developing proof logging methods that capture the computation of all Pareto-optimal solu-

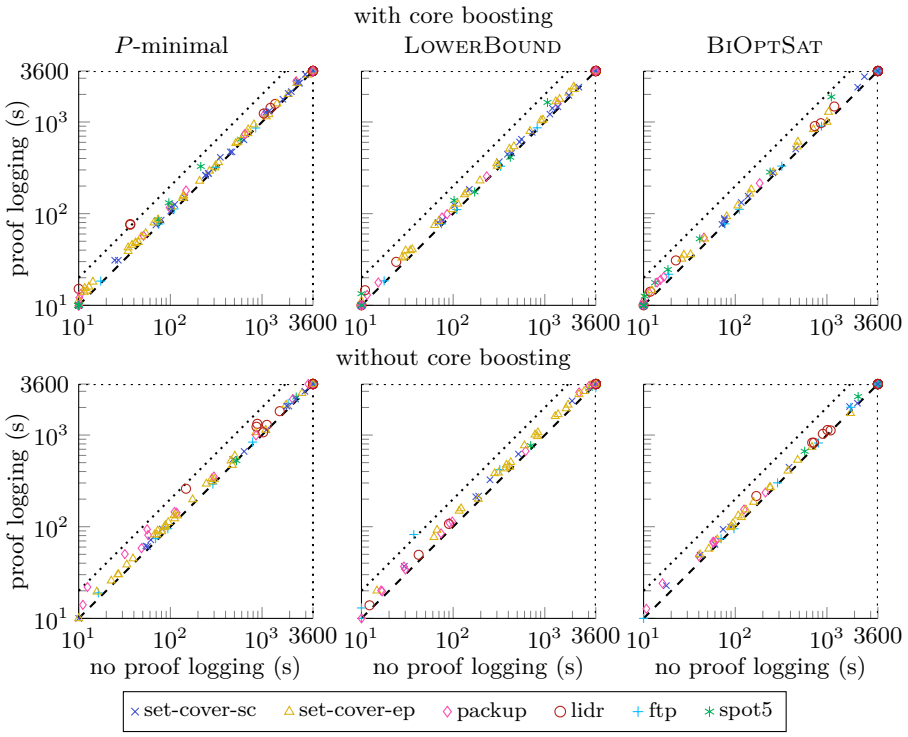

Fig. 4. Proof logging runtime overheads.

tions, i.e., every solution at each element in the non-dominated set, potentially by extending VeriPB, also remains part of future work.

Acknowledgments. We thank Armin Biere, Katalin Fazekas, and Florian Pollitt for their help in fixing some bugs in CaDiCaL, and Andy Oertel for fixing a bug in the VeriPB proof checker. This work is partially funded by Research Council of Finland (grants 356046 and 362987) and the European Union (ERC, CertiFOX, 101122653). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

1. VeriPB: Verifier for pseudo-Boolean proofs. `https://gitlab.com/MIAOresearch/software/VeriPB`
2. Andres, B., Kaufmann, B., Matheis, O., Schaub, T.: 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. LIPIcs, vol. 17, pp. 211–221. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2012). https://doi.org/10.4230/LIPIcs.ICLP.2012.211
3. Bacchus, F., Järvisalo, M., Martins, R.: Maximum satisfiabiliy. In: Biere et al. [9], pp. 929–991. https://doi.org/10.3233/FAIA201008
4. Baier, C., Chau, C., Klüppelholz, S.: Certificates and witnesses for multi-objective queries in markov decision processes. In: Hillston, J., Soudjani, S., Waga, M. (eds.) Quantitative Evaluation of Systems and Formal Modeling and Analysis of Timed Systems - First International Joint Conference, QEST+FORMATS 2024, Calgary, AB, Canada, September 9-13, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14996, pp. 1–18. Springer (2024). https://doi.org/10.1007/978-3-031-68416-6_1
5. Bailleux, O., Boufkhad, Y.: Efficient CNF encoding of boolean cardinality constraints. In: Principles and Practice of Constraint Programming – CP 2003, pp. 108–122. Springer Berlin Heidelberg (2003). https://doi.org/10.1007/978-3-540-45193-8_8
6. Berg, J., Bogaerts, B., Nordström, J., Oertel, A., Paxian, T., Vandesande, D.: Certifying without loss of generality reasoning in solution-improving maximum satisfiability. In: Shaw [48], pp. 4:1–4:28. https://doi.org/10.4230/LIPICS.CP.2024.4
7. Berg, J., Bogaerts, B., Nordström, J., Oertel, A., Vandesande, D.: Certified core-guided MaxSAT solving. In: Pientka, B., Tinelli, C. (eds.) Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14132, pp. 1–22. Springer (2023). https://doi.org/10.1007/978-3-031-38499-8_1
8. Biere, A., Faller, T., Fazekas, K., Fleury, M., Froleyks, N., Pollitt, F.: Cadical 2.0. In: Gurfinkel, A., Ganesh, V. (eds.) Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14681, pp. 133–152. Springer (2024). https://doi.org/10.1007/978-3-031-65627-9_7
9. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability - Second Edition, Frontiers in Artificial Intelligence and Applications, vol. 336. IOS Press (2021). https://doi.org/10.3233/FAIA336
10. Bogaerts, B., Gocht, S., McCreesh, C., Nordström, J.: Certified dominance and symmetry breaking for combinatorial optimisation. J. Artif. Intell. Res. **77**, 1539–1589 (2023). https://doi.org/10.1613/jair.1.14296
11. Cabral, M., Janota, M., Manquinho, V.: 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. LIPIcs, vol. 236, pp. 29:1–29:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPICS.SAT.2022.29
12. Cook, W.J., Coullard, C.R., Turán, G.: On the complexity of cutting-plane proofs. Discret. Appl. Math. **18**(1), 25–38 (1987). https://doi.org/10.1016/0166-218X(87)90039-4
13. Cortes, J.a., Lynce, I., Manquinho, V.: Slide&Drill, a New Approach for Multi-Objective Combinatorial Optimization. In: Shaw, P. (ed.) 30th International Conference on Principles and Practice of Constraint Programming (CP 2024). Leibniz International Proceedings in Informatics (LIPIcs), vol. 307, pp. 8:1–8:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2024). https://doi.org/10.4230/LIPIcs.CP.2024.8, https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CP.2024.8

14. Cortes, J., Lynce, I., Manquinho, V.M.: 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. Lecture Notes in Computer Science, vol. 13994, pp. 55–73. Springer (2023). https://doi.org/10.1007/978-3-031-30820-8_7
15. Cruz-Filipe, L., Heule, M.J.H., Jr., W.A.H., Kaufmann, M., Schneider-Kamp, P.: Efficient certified RAT verification. In: de Moura, L. (ed.) Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10395, pp. 220–236. Springer (2017). https://doi.org/10.1007/978-3-319-63046-5_14
16. Cruz-Filipe, L., Marques-Silva, J., Schneider-Kamp, P.: Efficient certified resolution proof checking. In: Legay, A., Margaria, T. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10205, pp. 118–135 (2017). https://doi.org/10.1007/978-3-662-54577-5_7
17. Demirovic, E., McCreesh, C., McIlree, M.J., Nordström, J., Oertel, A., Sidorov, K.: Pseudo-Boolean reasoning about states and transitions to certify dynamic programming and decision diagram algorithms. In: Shaw [48], pp. 9:1–9:21. https://doi.org/10.4230/LIPICS.CP.2024.9
18. Demirovic, E., Musliu, N., Winter, F.: Modeling and solving staff scheduling with partial weighted maxSAT. Ann. Oper. Res. **275**(1), 79–99 (2019). https://doi.org/10.1007/S10479-017-2693-Y
19. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers. Lecture Notes in Computer Science, vol. 2919, pp. 502–518. Springer (2003). https://doi.org/10.1007/978-3-540-24605-3_37
20. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into SAT. J. Satisf. Boolean Model. Comput. **2**(1-4), 1–26 (2006). https://doi.org/10.3233/sat190014
21. Ehrgott, M.: Multicriteria Optimization (2. ed.). Springer (2005). https://doi.org/10.1007/3-540-27659-9
22. Elffers, J., Gocht, S., McCreesh, C., Nordström, J.: Justifying all differences using pseudo-Boolean reasoning. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. pp. 1486–1494. AAAI Press (2020), https://ojs.aaai.org/index.php/AAAI/article/view/5507
23. Gocht, S., Martins, R., Nordström, J., Oertel, A.: Certified CNF translations for pseudo-Boolean solving. 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. LIPIcs, vol. 236, pp. 16:1–16:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.SAT.2022.16

24. Gocht, S., McBride, R., McCreesh, C., Nordström, J., Prosser, P., Trimble, J.: Certifying solvers for clique and maximum common (connected) subgraph problems. In: Simonis, H. (ed.) Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12333, pp. 338–357. Springer (2020). https://doi.org/10.1007/978-3-030-58475-7_20
25. Gocht, S., McCreesh, C., Nordström, J.: Subgraph isomorphism meets cutting planes: Solving with certified solutions. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020. pp. 1134–1140. ijcai.org (2020). https://doi.org/10.24963/ijcai.2020/158, scheduled for July 2020, Yokohama, Japan, postponed due to the Corona pandemic.
26. Gocht, S., McCreesh, C., Nordström, J.: An auditable constraint programming solver. In: Solnon, C. (ed.) 28th International Conference on Principles and Practice of Constraint Programming, CP 2022, July 31 to August 8, 2022, Haifa, Israel. LIPIcs, vol. 235, pp. 25:1–25:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.CP.2022.25
27. Gocht, S., Nordström, J.: Certifying parity reasoning efficiently using pseudo-Boolean proofs. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021. pp. 3768–3777. AAAI Press (2021), https://ojs.aaai.org/index.php/AAAI/article/view/16494
28. Goldberg, E.I., Novikov, Y.: Verification of proofs of unsatisfiability for CNF formulas. In: 2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany. pp. 10886–10891. IEEE Computer Society (2003). https://doi.org/10.1109/DATE.2003.10008
29. Guerreiro, A.P., Cortes, J., Vanderpooten, D., Bazgan, C., Lynce, I., Manquinho, V., Figueira, J.R.: Exact and approximate determination of the pareto front using minimal correction subsets. Comput. Oper. Res. **153**, 106153 (2023). https://doi.org/10.1016/J.COR.2023.106153
30. Heule, M., Jr., W.A.H., Wetzler, N.: Trimming while checking clausal proofs. In: Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013. pp. 181–188. IEEE (2013), https://ieeexplore.ieee.org/document/6679408/
31. Heule, M., Jr., W.A.H., Wetzler, N.: Verifying refutations with extended resolution. In: Bonacina, M.P. (ed.) Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7898, pp. 345–359. Springer (2013). https://doi.org/10.1007/978-3-642-38574-2_24
32. Hoen, A., Oertel, A., Gleixner, A.M., Nordström, J.: Certifying MIP-based presolve reductions for 0-1 integer linear programs. In: Dilkina, B. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 21st International Conference, CPAIOR 2024, Uppsala, Sweden, May 28-31, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14742, pp. 310–328. Springer (2024). https://doi.org/10.1007/978-3-031-60597-0_20
33. Ihalainen, H., Oertel, A., Tan, Y.K., Berg, J., Järvisalo, M., Myreen, M.O., Nordström, J.: Certified MaxSAT preprocessing. In: Benzmüller, C., Heule, M.J.H., Schmidt, R.A. (eds.) Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part I. Lecture

Notes in Computer Science, vol. 14739, pp. 396–418. Springer (2024). https://doi.org/10.1007/978-3-031-63498-7_24
34. Jabs, C.: Scuttle: A multi-objective MaxSAT solver. https://bitbucket.org/coreo-group/scuttle
35. Jabs, C., Berg, J., Ihalainen, H., Järvisalo, M.: Preprocessing in SAT-based multi-objective combinatorial optimization. In: Yap, R.H.C. (ed.) 29th International Conference on Principles and Practice of Constraint Programming, CP 2023, August 27-31, 2023, Toronto, Canada. LIPIcs, vol. 280, pp. 18:1–18:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPICS.CP.2023.18
36. Jabs, C., Berg, J., Järvisalo, M.: Core boosting in SAT-based multi-objective optimization. In: Dilkina, B. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 21st International Conference, CPAIOR 2024, Uppsala, Sweden, May 28-31, 2024, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14743, pp. 1–19. Springer (2024). https://doi.org/10.1007/978-3-031-60599-4_1
37. Jabs, C., Berg, J., Niskanen, A., Järvisalo, M.: From single-objective to bi-objective maximum satisfiability solving. Journal of Artificial Intelligence Research **80**, 1223–1269 (Aug 2024). https://doi.org/10.1613/jair.1.15333
38. Jabs, C.J., Berg, J., Bogaerts, B., Järvisalo, M.: Certifying pareto-optimality in multi-objective maximum satisfiability—TACAS 2025 artefact (Jan 2025). https://doi.org/10.5281/zenodo.14731485
39. Janota, M., Lynce, I., Manquinho, V., Marques-Silva, J.: PackUp: Tools for package upgradability solving. J. Satisf. Boolean Model. Comput. **8**(1/2), 89–94 (2012). https://doi.org/10.3233/SAT190090
40. Järvisalo, M., Heule, M., Biere, A.: Inprocessing rules. In: Gramlich, B., Miller, D., Sattler, U. (eds.) Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7364, pp. 355–370. Springer (2012). https://doi.org/10.1007/978-3-642-31365-3_28
41. Joshi, S., Martins, R., Manquinho, V.M.: 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. Lecture Notes in Computer Science, vol. 9255, pp. 200–209. Springer (2015). https://doi.org/10.1007/978-3-319-23219-5_15
42. Karpinski, M., Piotrów, M.: Encoding cardinality constraints using multiway merge selection networks. Constraints An Int. J. **24**(3-4), 234–251 (2019). https://doi.org/10.1007/S10601-019-09302-0
43. Koshimura, M., Nabeshima, H., Fujita, H., Hasegawa, R.: 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. CEUR Workshop Proceedings, vol. 556. CEUR-WS.org (2009), http://ceur-ws.org/Vol-556/paper06.pdf
44. Malioutov, D., Meel, K.S.: 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, August 27–31, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11008, pp. 312–327. Springer (2018). https://doi.org/10.1007/978-3-319-98334-9_21
45. Marques-Silva, J., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Biere et al. [9], pp. 133–182. https://doi.org/10.3233/FAIA200987

46. Martins, R., Joshi, S., Manquinho, V., Lynce, I.: Incremental cardinality constraints for MaxSAT. In: Lecture Notes in Computer Science, pp. 531–548. Springer International Publishing (aug 2014). https://doi.org/10.1007/978-3-319-10428-7_39
47. Morgado, A., Dodaro, C., Marques-Silva, J.: 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. Lecture Notes in Computer Science, vol. 8656, pp. 564–573. Springer (2014). https://doi.org/10.1007/978-3-319-10428-7_41
48. Shaw, P. (ed.): 30th International Conference on Principles and Practice of Constraint Programming, CP 2024, September 2-6, 2024, Girona, Spain, LIPIcs, vol. 307. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2024)
49. Soh, T., Banbara, M., Tamura, N., Le Berre, D.: 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. Lecture Notes in Computer Science, vol. 10416, pp. 596–614. Springer (2017). https://doi.org/10.1007/978-3-319-66158-2_38
50. Terra-Neves, M., Lynce, I., Manquinho, V.: Multi-objective optimization through pareto minimal correction subsets. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization (jul 2018). https://doi.org/10.24963/ijcai.2018/757
51. Vandesande, D.: Towards Certified MaxSAT Solving: Certified MaxSAT solving with SAT oracles and encodings of pseudo-Boolean constraints. Master's thesis, Vrije Universiteit Brussel (VUB) (2023), https://researchportal.vub.be/nl/studentTheses/towards-certified-maxsat-solving
52. Vandesande, D., De Wulf, W., Bogaerts, B.: QMaxSATpb: A certified MaxSAT solver. In: Gottlob, G., Inclezan, D., Maratea, M. (eds.) Logic Programming and Nonmonotonic Reasoning - 16th International Conference, LPNMR 2022, Genova, Italy, September 5-9, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13416, pp. 429–442. Springer (2022). https://doi.org/10.1007/978-3-031-15707-3_33

A Additional Empirical Data on Proof Checking

Fig. 5 shows a comparison between the per-instance solving runtimes with proof logging and the runtime of the VeriPB proof checker on the produced proofs. The color scale represents the range of the number of PD cuts introduced during solving. We observe that for P-minimal and BiOptSat without core boosting, a high number of PD cuts leads to a higher proof checking overhead, but for the other algorithms there is no such clear connection.

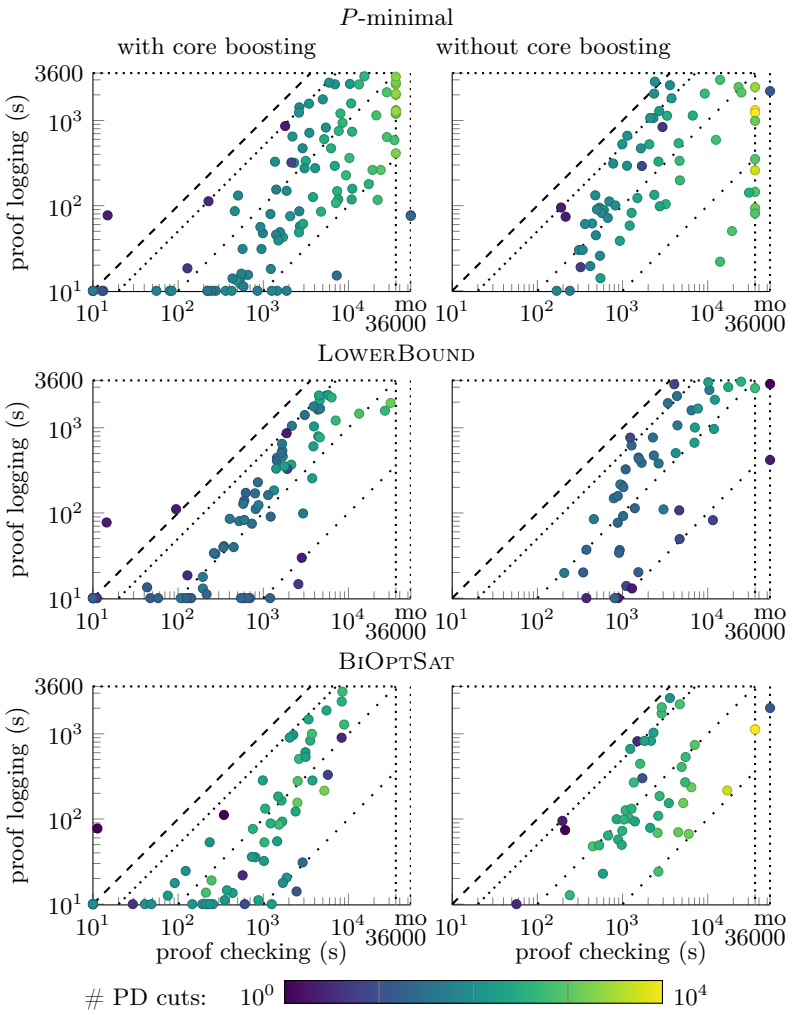

Fig. 5. Runtime comparison between solving with proof logging and proof checking wrt to the number of PD cuts.