Exploiting Partial-Assignment Enumeration in Optimization Modulo Theories

Gabriele Masina(✉) and Roberto Sebastiani(✉)

DISI, University of Trento, Trento, Italy
{gabriele.masina,roberto.sebastiani}@unitn.it

Abstract. Optimization Modulo Theories (OMT) extends Satisfiability Modulo Theories (SMT) with the task of optimizing some objective function(s). In OMT solvers, a CDCL-based SMT solver enumerates theory-satisfiable total truth assignments, and a theory-specific procedure finds an optimum model for each of them; the current optimum is then used to tighten the search space for the next assignments, until no better solution is found.

In this paper, we analyze the role of truth-assignment enumeration in OMT. First, we spotlight that the enumeration of *total* truth assignments is suboptimal, since they may over-restrict the search space for the optimization procedure, whereas using *partial* truth assignments instead can improve the effectiveness of the optimization. Second, we propose some assignment-reduction techniques for exploiting *partial-assignment enumeration* within the OMT context. We implemented these techniques in the OptiMathSAT solver, and we conducted an experimental evaluation on OMT benchmarks. The results confirm the improvement in both the efficiency of optimal solving and the quality of the obtained solutions for anytime solving.

Keywords: Optimization Modulo Theories · Enumeration · Partial Assignments

1 Introduction

Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of a logical formula w.r.t. some background theory, such as linear and nonlinear arithmetic, bit-vectors, arrays, or uninterpreted functions [2]. Many SMT-encodable problems also require the capability of finding models that are optimal w.r.t. some objective functions. These problems are grouped under the term Optimization Modulo Theories (OMT) [6,25,31]. OMT has been successfully applied to a wide range of problems, such as verification of timed and hybrid systems [14,31], numeric [16] and temporal planning [26,27], optimal scheduling [7], constrained goal modelling [24], hybrid machine learning [37], GAS optimization for smart contracts [1], and optimum encodings for quantum annealing [3,10], establishing OMT solvers as powerful tools for solving complex constraint optimization problems in various domains.

OMT Solving. A general OMT-solving strategy [25,31,32] consists in performing a sequence of incremental SMT calls, progressively tightening the range of values for the

R. Thiemann and C. Weidenbach (Eds.): FroCoS 2025, LNAI 15979, pp. 117–134, 2026.
https://doi.org/10.1007/978-3-032-04167-8_7

objective function. Specifically, an SMT solver is used to enumerate $\mathcal{T}$-satisfiable truth assignments that propositionally satisfy the problem formula φ. For each such truth assignment, a $\mathcal{T}$-optimizer finds a $\mathcal{T}$-model of optimum cost within it. A constraint is then added to the formula to tighten the upper bound for the cost of the optimum model, and the search continues until the formula is found unsatisfiable. Besides optimal solving, an important feature of OMT solvers is the ability to provide the user with a good-enough solution within a given time budget. This capability, known as *anytime* OMT solving, is especially valuable in industrial applications where finding the optimum solution may be computationally impractical, and it is rather more important to obtain high-quality solutions quickly.

OMT techniques have been developed for $\mathcal{LRA}$ [6,32], $\mathcal{LIRA}$ [6,33], $\mathcal{NRA}$ [4], $\mathcal{NIA}$ [4], $\mathcal{BV}$ [23,39], and $\mathcal{FP}$ [39]. Also, OMT has been extended to deal with multiple objectives including lexicographic OMT [6,33], boxed OMT [6,17,33], min-max OMT [34], and Pareto OMT [6]. Recently, a Generalized OMT calculus has been proposed, extending the definition to objectives over partially ordered sets [41].

Partial Assignments Enumeration in SMT. The problem of truth assignment enumeration has been studied in recent years, mainly in the context of SAT and SMT enumeration (AllSAT and AllSMT). Typically, enumeration algorithms [12,13,15,35,36] rely their efficiency on the enumeration of partial assignments to reduce both the number of enumerated assignments and the computational time by up to an exponential factor. Several techniques have been proposed to find short satisfying partial assignments starting from a total assignment, trading off efficiency for effectiveness (e.g., [22,29,38]). Also, the impact of CNF-ization on the effectiveness of partial assignment reduction has been recently studied in [20,21].

Contributions. In this paper, we study the applicability of enumeration-based techniques to OMT solving, and, in particular, the usage of partial truth assignment reduction to improve the effectiveness and efficiency of OMT solving. First, we notice that OMT solvers typically invoke the $\mathcal{T}$-optimizer on total truth assignments, and we spotlight how this can be suboptimal in many cases. Second, we propose some ways to exploit partial truth assignments in OMT solving, tailoring existing techniques to the OMT context. We discuss the general idea for an arbitrary theory, and describe an implementation for the specific case of linear arithmetic over $\mathcal{LRA}$ and $\mathcal{LIRA}$, possibly combined with other theories. We implemented these strategies in the OMT solver OptiMathSAT [34], and we show through an empirical evaluation over OMT($\mathcal{LRA}$), OMT($\mathcal{LIRA}$), and OMT($\mathcal{LRA} \cup \mathcal{AR}$) benchmarks that they improve both the efficiency of optimal solving and the quality of obtained solutions for anytime solving.

Related Work. The idea of using partial assignments in OMT($\mathcal{LIRA}$) has been previously considered in [5,18]. In [5], the authors mention that in lazy OMT-solving, the truth assignments should preferably be prime implicants. This approach, however, is theory-blind; our reduction techniques, instead, specifically target theory-literals involved in optimization, which proves crucial for the solver efficiency. A similar idea was proposed in [18], where the truth assignments are reduced to prime implicants before invoking the $\mathcal{T}$-solver and $\mathcal{T}$-minimizer. Though our work is similar in flavour

to theirs, there are some key differences: (a) their enumeration algorithm only adds clauses blocking the minimized truth assignment, whereas modern OMT solvers add cost bounds ($\mathsf{cost} < \mathsf{ub}$) to prune the search space, which has been shown to be much more effective; (b) we also propose a cost-guided technique, which we show to perform much better in practice.

Organization. The rest of the paper is organized as follows. In Sect. 2, we provide the necessary background on SMT and OMT solving. In Sect. 3, we analyze the role of total and partial truth assignments in OMT solving. In Sect. 4, we propose two strategies to exploit partial truth assignments in OMT solving. In Sect. 5, we present an experimental evaluation of the proposed strategies over OMT($\mathcal{LRA}$), OMT($\mathcal{LIRA}$), and OMT($\mathcal{LRA} \cup \mathcal{AR}$) benchmarks. Finally, in Sect. 6, we conclude the paper and discuss future work.

2 Background

Notation and Terminology. We assume the standard setting with quantifier-free first-order formulas, and the standard notions of theory, satisfiability, logical consequence. We assume the reader is familiar with these notions and with the lazy CDCL-based SMT-solving approach, and refer to [2] for a comprehensive introduction to SMT.

In this paper, we denote SMT formulas by φ, theories by $\mathcal{T}$, variables by x, y, atoms by α, truth assignments by μ, η, and models by $\mathcal{M}$; all symbols possibly with subscripts or superscripts. We denote by $Atoms(\varphi)$ the set of atoms occurring in a formula φ.

2.1 Satisfiability Modulo Theories

Given a first-order theory $\mathcal{T}$, a $\mathcal{T}$-atom is any atomic formula built over the signature of $\mathcal{T}$. A $\mathcal{T}$-literal is a $\mathcal{T}$-atom or its negation. A $\mathcal{T}$-formula is either a $\mathcal{T}$-literal or a combination of formulas by means of standard Boolean operators. From now on, we assume every formula is in Conjunctive Normal Form (CNF), i.e., it is a conjunction ($\wedge$) of clauses, where each clause is a disjunction ($\vee$) of literals. (If it is not, then it can be easily converted into CNF by applying the standard transformations [28,40]).

Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of a first-order formula w.r.t some first-order theory $\mathcal{T}$, or combination of first-order theories $\mathcal{T} \cup \mathcal{T}'$. A formula is $\mathcal{T}$-satisfiable if it is satisfiable in a model of $\mathcal{T}$ (a $\mathcal{T}$-model). Popular theories include linear and nonlinear arithmetic over the reals or integers ($\mathcal{LRA}$, $\mathcal{NRA}$, $\mathcal{LIRA}$, and $\mathcal{NIRA}$), bit-vectors ($\mathcal{BV}$), and floating-point ($\mathcal{FP}$).

Lazy SMT-Solving. Given a formula φ with $Atoms(\varphi) = \{\alpha_1, \ldots, \alpha_n\}$, a truth assignment $\mu : Atoms(\varphi) \rightarrow \{\top, \bot\}$ maps atoms in φ to truth values. A partial truth assignment is a partial mapping, and a total truth assignment is a total mapping. We represent a truth assignment μ also as a conjunction of literals $\bigwedge_{\mu(\alpha_i)=\top} \alpha_i \wedge \bigwedge_{\mu(\alpha_i)=\bot} \neg\alpha_i$. We say that μ *propositionally satisfies* φ iff μ satisfies all clauses in φ.

The CDCL($\mathcal{T}$) algorithm [19] is based on the so-called lazy approach to SMT (see e.g., [2,30]), which exploits the fact that a $\mathcal{T}$-formula φ is $\mathcal{T}$-satisfiable iff there exists

a truth assignment μ that propositionally satisfies φ and μ is $\mathcal{T}$-satisfiable. It combines a CDCL-based SAT-solver with a $\mathcal{T}$-specialized decision procedure called $\mathcal{T}$-solver to decide the consistency of a set of $\mathcal{T}$-literals. Whenever the SAT-solver finds a truth assignment μ propositionally satisfying φ, it invokes the $\mathcal{T}$-solver to check the $\mathcal{T}$-satisfiability of μ. If μ is $\mathcal{T}$-satisfiable, then the $\mathcal{T}$-solver returns a model $\mathcal{M}$, that is also a model of φ. Otherwise, the $\mathcal{T}$-solver returns a subset of μ that causes the $\mathcal{T}$-unsatisfiability, which is learned by the SAT-solver and used in subsequent iterations to prune the search space.

To maximize efficiency, most $\mathcal{T}$-solvers can be called incrementally via a stack-based interface, keeping the status of the search between calls. E.g., an efficient, incremental $\mathcal{LRA}$-solver, can be built on a variant of the Simplex algorithm designed to be integrated within a lazy SMT framework [11]. The combination of theories can be handled efficiently by delayed theory combination [8].

Another important feature of CDCL-based SMT solvers is that they provide a stack-based incremental interface, allowing to push and pop clauses and incrementally check the satisfiability of the formula conjoined with the pushed clauses, maintaining most of the learned information between calls.

2.2 Optimization Modulo Theories

Let $\mathcal{T}$ be a theory admitting some total order relation "$\leq$" over its domain, let φ be a $\mathcal{T}$-formula, and let cost be a $\mathcal{T}$-term which we call *objective function*. *Optimization Modulo Theories* (OMT($\mathcal{T}$)) is the problem of finding a model for φ that makes the value of cost minimum according to the order given by $\leq$ (maximization is dual) [4,31]. To simplify the presentation, we focus on minimization, but the same concepts apply to maximization as well. Notice that, in general, φ can be built on a combination of $\mathcal{T}$ with other theories (OMT($\mathcal{T} \cup \mathcal{T}'$)), and the same concepts apply to such cases [31,32].

Example 1. Consider the $\mathcal{LRA}$-formula on the rational variables x, y:

$$\begin{aligned} \varphi \stackrel{\text{def}}{=} & ((2x - 3y \leq 6) \vee (x \leq 4)) \wedge \\ & ((y \leq 2) \vee (y \leq -3x + 9) \vee (x < -2)). \end{aligned} \tag{1}$$

φ is $\mathcal{LRA}$-satisfiable, e.g., the $\mathcal{LRA}$-model $\mathcal{M} \stackrel{\text{def}}{=} \{x \mapsto 3, y \mapsto 0\}$ satisfies φ.

Consider the OMT($\mathcal{LRA}$) problem $\langle \varphi, \mathsf{cost} \rangle$ where φ is the $\mathcal{LRA}$-formula in (1), and $\mathsf{cost} \stackrel{\text{def}}{=} -2x$. Then the model $\mathcal{M} \stackrel{\text{def}}{=} \{x \mapsto 3, y \mapsto 0\}$ has $\mathsf{cost} = -6$. A better model of φ is, e.g., $\mathcal{M}' \stackrel{\text{def}}{=} \{x \mapsto 6, y \mapsto 2\}$, that has $\mathsf{cost} = -12$. This model is also the model of φ with minimum cost.

Lazy OMT Solving. A general optimization strategy implemented by state-of-the-art OMT solvers, and typically used for OMT($\mathcal{LRA}$) and OMT($\mathcal{LIRA}$), is the so-called *linear-search* strategy [25,31,32]. It consists in solving a sequence of SMT problems where the space of feasible solutions is progressively tightened by learning unit clauses in the form $(\mathsf{cost} < \mathsf{ub})$, ub being the currently-known upper bound for cost. At each iteration, the solver can either find a model $\mathcal{M}$ whose value of cost is smaller than ub, or detect the unsatisfiability of the current formula. In the first case, the solver invokes

a $\mathcal{T}$-specific procedure, called $\mathcal{T}$ *-minimizer*, to find an optimum model $\mathcal{M}'$ within the truth assignment induced by $\mathcal{M}$. Then, ub is set to $\mathcal{M}'(\mathsf{cost})$, and the search continues. In the second case, the formula has no models with cost lower than ub, and the search terminates as the last model found is optimum.

Alternatively, the solver could also follow a *binary-search* strategy [31]. In this case, a lower and upper bound lb and ub are kept s.t. the optimum model lies in the interval $(\mathsf{lb}, \mathsf{ub}]$. At each iteration, an intermediate value $\mathsf{pivot} \in (\mathsf{lb}, \mathsf{ub}]$ is chosen, and the solver checks if there exists a model with cost lower than pivot. If so, pivot becomes the new upper bound, otherwise, it becomes the new lower bound. The search terminates when lb and ub are equal, and the last model found is optimum. (In continuous domains, e.g., OMT($\mathcal{LRA}$), to guarantee termination, it is necessary to interleave binary-search steps with a linear-search step [31]). In this paper, we focus on the linear-search strategy, but the analysis applies to the binary-search strategy as well.

If φ is built on a combination of theories, then the $\mathcal{T}$-minimizer is invoked on $\mu_{\mathcal{T}} \cup \mu_{ed}$—i.e., the subset of μ containing only the atoms in $\mathcal{T}$ and the interface (dis-)equalities [31,32] The implementation of $\mathcal{T}$-minimizers for $\mathcal{LRA}$ and $\mathcal{LIRA}$ is briefly described in the next paragraph.

T-minimizers. A $\mathcal{LRA}$-minimizer [31,32] can be implemented as a simple extension of the Simplex-based $\mathcal{LRA}$-solver [11]. For $\mathcal{LIRA}$, a minimizer can be built on top of a branch-and-bound $\mathcal{LIRA}$-solver [5,6,33], by replacing the $\mathcal{LRA}$-solver with a $\mathcal{LRA}$-solver&minimizer to solve each relaxed subproblem. To find an optimum model within the truth assignment, once a $\mathcal{LIRA}$-model $\mathcal{M}$ is found, a constraint $(\mathsf{cost} < \mathcal{M}(\mathsf{cost}))$ is pushed onto the $\mathcal{LIRA}$-solver, and the search is iteratively refined until no better model exists. An alternative strategy, implemented, e.g., in OPTIMATHSAT, is the so-called *truncated* optimization [33], where the $\mathcal{LIRA}$-minimizer stops after finding the first model $\mathcal{M}$. Although this model may be sub-optimal for the current truth assignment—allowing the same assignment to be found again by the CDCL($\mathcal{T}$) procedure—this approach is typically much faster and remains effective in practice.

Remark 1. Importantly, the lazy OMT solving approach allows for an *anytime* behavior, i.e., we can interrupt the search at any time and return the best model found so far.

2.3 SAT and SMT Enumeration

SAT enumeration (AllSAT) is the problem of finding all the truth assignments that propositionally satisfy a propositional formula. SMT enumeration (AllSMT) is the problem of finding all $\mathcal{T}$-satisfiable truth assignments that propositionally satisfy a $\mathcal{T}$-formula. Since a partial assignment can be extended to 2^k total truth assignments, k being the number of unassigned atoms, finding short partial truth assignments is a key point in reducing both the number of enumerated truth assignments and the computational time by up to an exponential factor.

Many enumeration algorithms find total truth assignments, and then extract partial truth assignments from them by some reduction procedure. A basic reduction procedure is illustrated in Algorithm 1. It consists in iteratively dropping literals one-by-one from the truth assignment, checking if it still satisfies the formula. The resulting partial

Algorithm 1. REDUCE-ASSIGNMENT(φ, η)
Input: CNF formula φ, $\mathcal{T}$-satisfiable total truth assignment η satisfying φ
Output: Reduced (minimal) partial truth assignment $\mu \subseteq \eta$ satisfying φ

```
1: μ ← η
2: for ℓ ∈ μ do
3:     if μ \ {ℓ} satisfies all clauses in φ then
4:         μ ← μ \ {ℓ}
5: return μ
```

assignment is minimal, i.e., it cannot be further reduced without violating the satisfaction of the formula. Notice that the order in which literals are dropped can have a significant impact on the effectiveness of the reduction procedure.

3 An Analysis of Enumeration in OMT

In the following, to simplify the notation and the presentation, we refer to one single theory $\mathcal{T}$, but the results can be straightforwardly extended to combinations of theories.

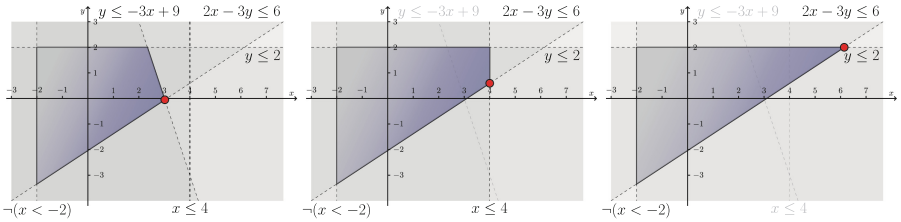
As described in Sect. 2.2, a basic OMT solving schema involves the interaction of a combinatorial and a theory-specific optimization components. In the combinatorial component, an SMT solver enumerates $\mathcal{T}$-satisfiable truth assignments that propositionally satisfy the problem formula φ conjoined with increasingly tighter bounds on the cost of the optimum solution. In the theory-specific component, a $\mathcal{T}$-minimizer finds a $\mathcal{T}$-model of minimum cost within the constraints imposed by the given truth assignment. This model is then used to tighten the upper bound for the cost of the optimum model and continue the search, until the formula is found unsatisfiable.

Since the enumeration is based on the CDCL($\mathcal{T}$) schema [19], these truth assignments are typically *total*, i.e., they assign a truth value to each atom of the formula. However, we point out that total truth assignments can often over-constrain the search space for the optimum model, whereas relying on *partial* truth assignments can be much more effective. Intuitively, *by removing from the current satisfying truth assignment $\mathcal{T}$-constraints that are not strictly necessary for the propositional satisfaction of the formula, we enlarge the area within which the optimum model is searched, thus increasing the chances of finding a better optimum model.* This means that the solver can add a tighter upper bound to the cost of the global optimum, potentially reducing the number of search iterations needed to find it, and consequently the overall solving time. Moreover, this improvement can be crucial for anytime OMT solving, as it allows the solver to converge faster to better solutions within the given time limit.

We illustrate this idea in the following example.

Example 2. Consider the OMT($\mathcal{LRA}$) problem $\langle \varphi, \mathsf{cost} \rangle$ where φ is the formula in (1) in Example 1, and $\mathsf{cost} \stackrel{\text{def}}{=} -2x$. Consider the following scenario, which is graphically represented in Fig. 1. Consider the $\mathcal{LRA}$-satisfiable total truth assignment that propositionally satisfies φ:

$$\mu \stackrel{\text{def}}{=} \{(2x - 3y \leq 6), (y \leq 2), \neg(x < -2), (y \leq -3x + 9), (x \leq 4)\}. \qquad (2)$$



(a) Total assignment μ (2) (b) Partial assignment μ' (3) (c) Partial assignment μ'' (4)

Fig. 1. Graphical representation of Example 2. For each step, the half-planes representing the constraints in the truth assignment are delimited by dashed lines and colored in grey. The intersection of these constraints is colored in blue, with a gradient that follows the value of cost (the lower the value of cost, the more intense the color), and the red dot represents the optimum model found within this region. (Color figure online)

The optimum model of μ is $\{x \mapsto 3, y \mapsto 0\}$ with $\mathsf{cost} = -6$ (Fig. 1a). We notice that some literals in μ are not strictly necessary for propositionally satisfying φ. In fact, we only need one true literal in each clause to propositionally satisfying the formula. Not every drop is equally effective, though. For instance, if we drop $(x \leq 4)$, $(y \leq 2)$, or $\neg(x < 2)$, then we get a truth assignment with the same optimum as μ, since these literals don't "oppose" to the optimization of cost in μ. Dropping at least one of the other two literals, instead, leads to a truth assignment with a better optimum model. For instance, if we drop $(y \leq -3x + 9)$, we get:

$$\mu' \stackrel{\text{def}}{=} \mu \setminus \{(y \leq -3x + 9)\} = \{(2x - 3y \leq 6), (y \leq 2), \neg(x < -2), (x \leq 4)\}. \quad (3)$$

The optimum model of μ' is $\{x \mapsto 4, y \mapsto 2/3\}$ with $\mathsf{cost} = -8$ (Fig. 1b). At this point, two constraints, $(x \leq 4)$ and $(2x - 3y \leq 6)$, oppose to the optimization of cost, and either of them can be safely dropped. Assume that we drop $(x \leq 4)$, then we get:

$$\mu'' \stackrel{\text{def}}{=} \mu' \setminus \{(x \leq 4)\} = \{(2x - 3y \leq 6), (y \leq 2), \neg(x < -2)\} \quad (4)$$

with optimum model $\{x \mapsto 6, y \mapsto 2\}$ and $\mathsf{cost} = -12$ (Fig. 1c). Now, the only literals opposing to the optimization of cost are $(2x - 3y \leq 6)$ and $(y \leq 2)$, but none of them can be dropped. Hence, no further improvement can be obtained.

In general, partial truth assignments have an optimum model that is necessarily better or equal to that of the total truth assignments extending them. Since multiple partial truth assignments can be obtained from a total one, the choice of which constraints to drop can be crucial to improve the quality of the optimum model found.

4 Exploiting Partial Truth Assignments in OMT

The general schema of our approach is presented in Algorithm 2. This algorithm is a variant of the basic OMT linear-search schema [31,32] described in Sect. 2.2. The main

Algorithm 2. LINEAR-SEARCH OMT WITH PARTIAL ASSIGNMENTS(φ, cost)
Input: Formula φ, objective cost
Output: SAT/UNSAT, optimum model $\mathcal{M}$

```
1:  M ← ∅      // Best model found so far
2:  ub ← ∞     // Current upper bound
3:  res ← SAT  // Status of the search
4:  while res = SAT do
5:      ⟨res, η⟩ ← SMT.IncrementalSolve(φ ∧ (cost < ub))
6:      if res = SAT then
7:          μ ← OMT-REDUCE-ASSIGNMENT(φ, η, cost)
8:          M ← T-Solver.Minimize(μ, cost)
9:          ub ← M(cost)
10: if M = ∅ then
11:     return ⟨UNSAT, ∅⟩
12: else
13:     return ⟨SAT, M⟩
```

difference is the call to the OMT-REDUCE-ASSIGNMENT procedure (line 7), which is responsible for reducing the truth assignment to be fed to the $\mathcal{T}$-minimizer, provided that the resulting partial truth assignment still propositionally satisfies the formula. Depending on the implementation of this procedure, the assignment-reduction strategy can be more or less effective in improving the search for the global optimum.

In Sect. 4.1 and Sect. 4.2, we describe two possible implementations of this procedure.

4.1 Basic Assignment Reduction

The first approach is to reduce the truth assignment using Algorithm 1 in Sect. 2.3, i.e., iterating over all the literals in the current truth assignment η, and dropping them one by one, if possible. A straightforward improvement is to only try to drop $\mathcal{T}$-literals, since they are the ones that, if dropped, can potentially enlarge the area within which the optimum $\mathcal{T}$-model is searched. In the case of theory combination, we drop only literals from the theory $\mathcal{T}$ of the "$\leq$" symbol w.r.t. which we minimize. Possibly, heuristics can be used to choose an appropriate dropping order; in our implementation, we used the default strategy that follows the appearance order of the atoms in the formula. This procedure is simple and general, and comes with a limited overhead, as each truth assignment is scanned only once to find the literals to drop, and the $\mathcal{T}$-minimizer is called only once for each candidate assignment.

This approach, however, might not be very effective in practice, as it "blindly" removes literals from the truth assignment without taking into account the properties of the OMT search strategy. In particular, the search space may be enlarged in a direction that does not improve the objective, bringing no benefit to the search. Accurately choosing which literals to drop is crucial because the removal of a literal may prevent the removal of other literals that could potentially be more relevant for the optimization.

Algorithm 3. OMT-REDUCE-ASSIGNMENT-GUIDED (φ, η, cost)
Input: Formula φ, $\mathcal{T}$-satisfiable total truth assignment η satisfying φ, objective cost
Output: Reduced truth assignment $\mu \subseteq \eta$ satisfying φ

```
1: μ ← η
2: M ← T-Solver.MinimizeApprox(μ, cost)
3: ℓ ← T-Solver.ProposeLiteralToDrop()
4: while ℓ ≠ ⊥ do
5:   if μ \ {ℓ} satisfies all clauses in φ then
6:     μ ← μ \ {ℓ}
7:     M ← T-Solver.MinimizeApprox(μ, cost)
8:   ℓ ← T-Solver.ProposeLiteralToDrop()
9: return μ
```

4.2 Guided Assignment Reduction

We propose an ad-hoc assignment-reduction technique for OMT solving, which is outlined in Algorithm 3. Suppose that, after the $\mathcal{T}$-minimizer has found a minimum model within the current truth assignment μ (line 2), it returns also one (or more) literal(s) that limit the current minimum (line 3). These literals are part of some (possibly minimal) $\mu' \subseteq \mu$ such that $\mu' \cup \{\mathsf{cost} < \mathcal{M}(\mathsf{cost})\}$ is $\mathcal{T}$-unsatisfiable. Intuitively, the removal of any literal $\ell \in \mu'$ is very likely to lead to a better optimum model, provided that $\mu \setminus \{\ell\}$ still propositionally satisfies φ (line 5).

We can then iteratively drop these literals and re-run the $\mathcal{T}$-minimizer, until no more literals can be dropped (lines 4–8). Notice that instead of T-Solver.Minimize, here we call T-Solver.MinimizeApprox, suggesting that also a relaxed optimization algorithm could be used. In the next paragraph, we describe how this can be exploited for $\mathcal{LIRA}$.

Notice that, in Algorithm 3, we only drop one literal before every optimization call; nevertheless, in principle, more literals could be dropped. Experimentally, we found that dropping more than one literal was not beneficial. The reason is that, generally, when a literal is removed, most of the other literals in μ' do not limit the current minimum anymore. Hence, their removal not only does not lead to a better optimum model, but also can prevent the removal of other more-relevant literals.

On Proposing Literals to Drop. For an arbitrary theory $\mathcal{T}$, a generic implementation for T-Solver.ProposeLiteralToDrop (Algorithm 3, lines 3, 8) could be as follows. Once a minimum model $\mathcal{M}$ for μ is found, invoke the $\mathcal{T}$-solver on $\mu \wedge (\mathsf{cost} < \mathcal{M}(\mathsf{cost}))$ and return a (possibly-minimal) conflict set. For some theories, ad-hoc (possibly heuristic) procedures can be employed. We describe two such techniques for $\mathcal{LRA}$ and $\mathcal{LIRA}$.

As we recalled in Sect. 2.2, a $\mathcal{LRA}$-minimizer can be implemented as a variant of the Simplex method [11,31], by which an optimum model is always found on a vertex of the polytope defined by the conjunction of $\mathcal{LRA}$-constraints on which it is invoked. Thus, in this case, the candidate constraints to be dropped are those that form such a vertex. This information can be easily obtained from the Simplex tableau [11].

For $\mathcal{LIRA}$, directly using a $\mathcal{LIRA}$-minimizer and then identifying the limiting constraints presents some challenges: (a) a single call to the branch-and-bound-based

$\mathcal{LIRA}$-minimizer is worst-case exponential, hence calling it multiple times (as in Algorithm 3) is not feasible; (b) extracting the limiting constraints from a $\mathcal{LIRA}$-minimizer is not straightforward. We thus propose proceeding as follows. First, for the procedure T-Solver.MinimizeApprox a $\mathcal{LRA}$-minimizer is invoked on the relaxation of μ. Then, the procedure T-Solver.ProposeLiteralToDrop can be implemented as for $\mathcal{LRA}$. The intuition here is that reasoning on the relaxation of μ is much easier and cheaper—especially if incremental calls are used—and still allows for enlarging the search area in a favourable direction for the $\mathcal{LIRA}$-minimizer. We remark that, after the assignment reduction, in Algorithm 2 (line 8), the complete $\mathcal{LIRA}$-minimizer is called.

As a last aspect, we suggest that, if the $\mathcal{T}$-minimizer is able to find an optimum model μ, then we can use these limiting literals $l_1, \ldots, l_n$ also to learn a theory lemma $(\neg(\mathsf{cost} < \mathsf{ub}) \vee \neg l_1 \vee \ldots \vee \neg l_n)$ that blocks truth assignments that we know are not better than the current upper bound ub—thus preventing useless calls to the $\mathcal{T}$-solver. The idea is that, in order to find a model with $\mathsf{cost} < \mathsf{ub}$, we need to assign at least one of these literals to false. This is the case of $\mathcal{LRA}$, but not of $\mathcal{LIRA}$ if the truncated minimization method is used (see Sect. 2.2).

5 Experimental Evaluation

We implemented the above algorithms in the OMT solver OptiMathSAT [34], which is built on top of the MathSAT5 SMT solver [9]. We evaluated the proposed strategies on a set of OMT($\mathcal{LRA}$), OMT($\mathcal{LIRA}$), and OMT($\mathcal{LRA} \cup \mathcal{AR}$) benchmarks coming from different sources, evaluating both solving time for optimum solving, and the quality of the solutions found within the given timeout for anytime solving. All the experiments were run on an Intel Xeon Gold 6238R @ 2.20GHz 28 Core machine with 128 GB of RAM, running Ubuntu Linux 22.04. The timeout was set at 1200 s. The tool, benchmarks and results are available at https://optimathsat.disi.unitn.it/resources/optimathsat-partial-assignments.tar.gz.

5.1 Benchmarks

We evaluated the proposed strategies on two classes of OMT($\mathcal{LRA}$) benchmarks: OMT-encoded optimal temporal planning [26,27] and strip-packing problems [31, 32]. We also modified the strip-packing benchmarks to use OMT($\mathcal{LIRA}$) and OMT($\mathcal{LRA} \cup \mathcal{AR}$) encodings, to evaluate the effectiveness of our strategies in these theories.

Optimal Temporal Planning. In [26,27], the authors proposed a way to encode optimal temporal planning problems into a sequence of OMT($\mathcal{LRA}$) problems. Each problem encodes a bounded version of the problem up to a fixed horizon, with additional abstract actions representing an over-approximation of the plans beyond the bound, minimizing the makespan, i.e., the total time taken to reach the goal. If the optimal plan is found without using the abstract actions, then the plan is optimum for the original problem. Otherwise, the horizon is increased, and the process is repeated. We generated problems using the industrial problems Majsp (80 instances), MajspSimplified (80 instances), and Painter (30 instances) [27], with increasing horizon $h \in \{5, 10, 15, 20, 25, 30, 35, 40\}$, for a total of 1520 instances.

Strip-Packing. The strip-packing problem (SP) requires arranging N rectangles with widths W_i and heights H_i into a strip of fixed height H and unlimited length. The goal is to minimize the length of the used part of the strip, ensuring all rectangles are placed without overlap or rotation. An OMT($\mathcal{LRA}$) encoding for SP was proposed in [32]. Following [32], we sampled H_i uniformly in $(0, 1]$, W_i in $(1, 2]$, and set $H = \sqrt{N}/2$.

We also generated OMT($\mathcal{LIRA}$) SP problems, by randomly choosing with equal probability whether encoding the coordinates of each rectangle with integer or rational variables. Finally, we generated OMT($\mathcal{LRA} \cup \mathcal{AR}$) encodings for SP, by simply replacing the variables x_i in the OMT($\mathcal{LRA}$) encoding with a $\mathcal{LRA} \cup \mathcal{AR}$ term $read(x, i + offset)$, where x is an array mapping from rationals to rationals, i is a constant indicating the index of the rectangle, and $offset$ is a fresh rational variable.

For each of these encodings, we generated 25 random SP problems for each value of $N \in \{25, 50, 75, 100\}$, for a total of 100 instances per encoding.

5.2 Results

Figures 2, 3, 4 and 5 show the results on temporal planning and SP benchmarks for the different theories, respectively. For each benchmark set, we report a set of scatter plots.

On the rows, we have different metrics, namely the solving time in seconds (time(s)), the upper bound (u.b.)—i.e., the optimum value when the solver terminated within the time limit, or the value of the best solution found within the timeout otherwise—and the number of iterations (# iter) taken to reach the upper bound (see Algorithm 2).

On the columns, we compare the results obtained with the different truth-assignment-reduction strategies: in the left and center columns, we respectively compare the basic and the guided reductions with the plain algorithm without reductions. In the right column, we compare the two reduction strategies.

OMT($\mathcal{LRA}$) Benchmarks. The results are summarized in Figs. 2 and 3.

Optimal Temporal Planning (Fig. 2). In these benchmarks, with no truth-assignment reduction, OptiMathSAT reported 246 timeouts out of 1520 problems, 211 with the basic reduction, and 212 with the guided reduction.

From the plots (first row, left and center columns), we can see that applying either reduction almost uniformly improves the solving time with few exceptions, making optimal solving up to twice as fast as with no reduction.

Moreover, we observe that reducing truth assignments is very effective also for anytime solving (second row, left and center columns). Notice that when the solver terminated within the timeout with both strategies, then the corresponding points lie on the bisector, whereas when at least one strategy times out, the points are generally below the bisector. Indeed, this shows that, for anytime solving, both the basic and the guided reductions allow finding a much better upper bound than with no reduction.

Finally, we can see that both strategies are particularly effective in reducing the number of iterations needed to either find the optimum or to reach the best upper bound within the timeout (third row, left and center). Reducing the number of iterations is not

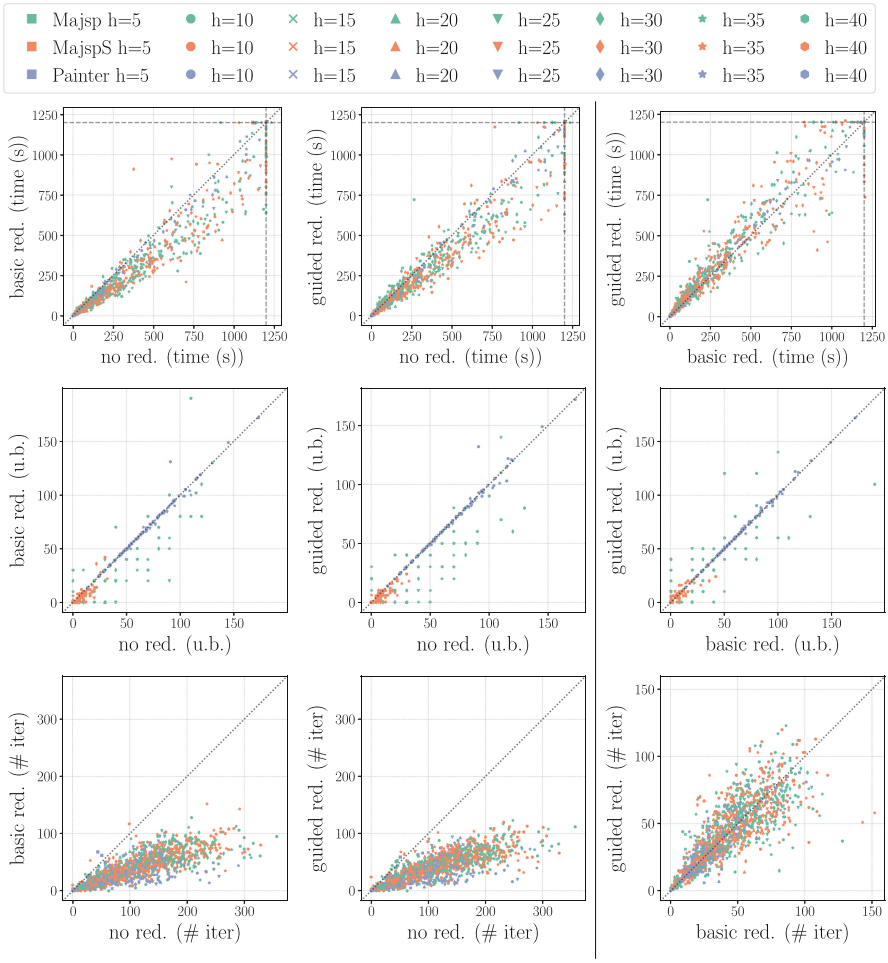


Fig. 2. Results on OMT($\mathcal{LRA}$)-encoded optimal temporal planning problems.

an advantage in itself, but it is a good indicator of the effectiveness of truth-assignment reduction strategies in OMT.

Overall, in these benchmarks there is no clear winner between the two reduction strategies (right column), but it is evident that applying either form of truth-assignment reduction can be beneficial in OMT, both for optimal and anytime solving.

Strip-Packing (Fig. 3). Since no instance in this set of benchmarks terminated within the timeout, for these benchmarks we omit the time plots. We can see that here the basic reduction strategy is not really effective, since the value of the upper bound is not improved compared to the no-reduction strategy (first row, left column). Also, the number of iterations only slightly decreases (second row, left column), suggesting that here blindly removing atoms from the truth assignment does not help much in finding

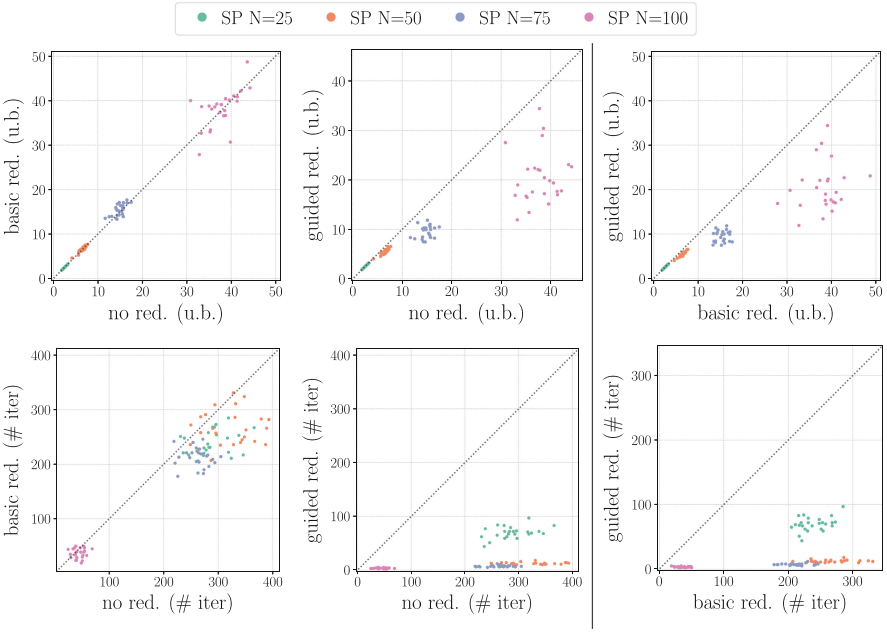


Fig. 3. Results on OMT($\mathcal{LRA}$)-encoded strip-packing problems.

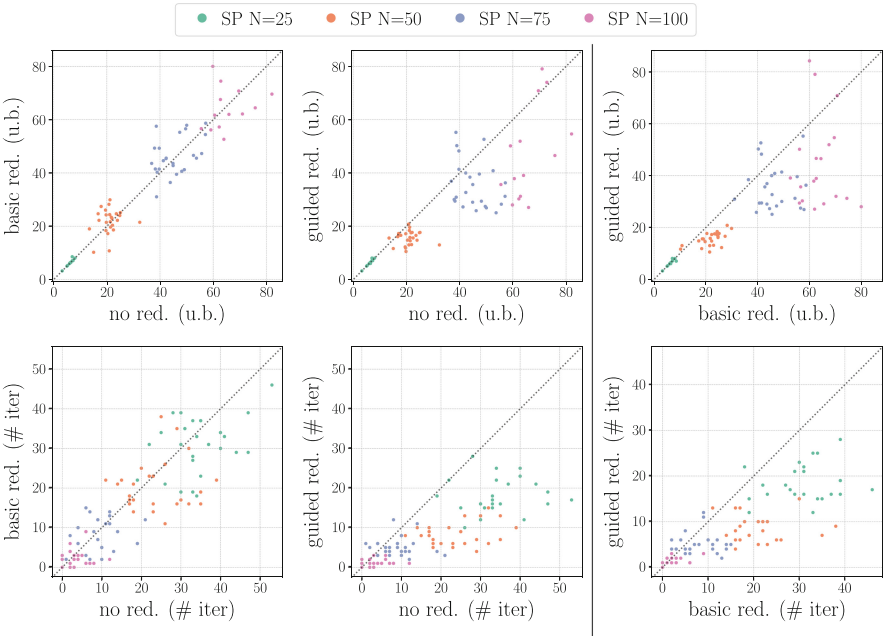


Fig. 4. Results on OMT($\mathcal{LIRA}$)-encoded strip-packing problems.

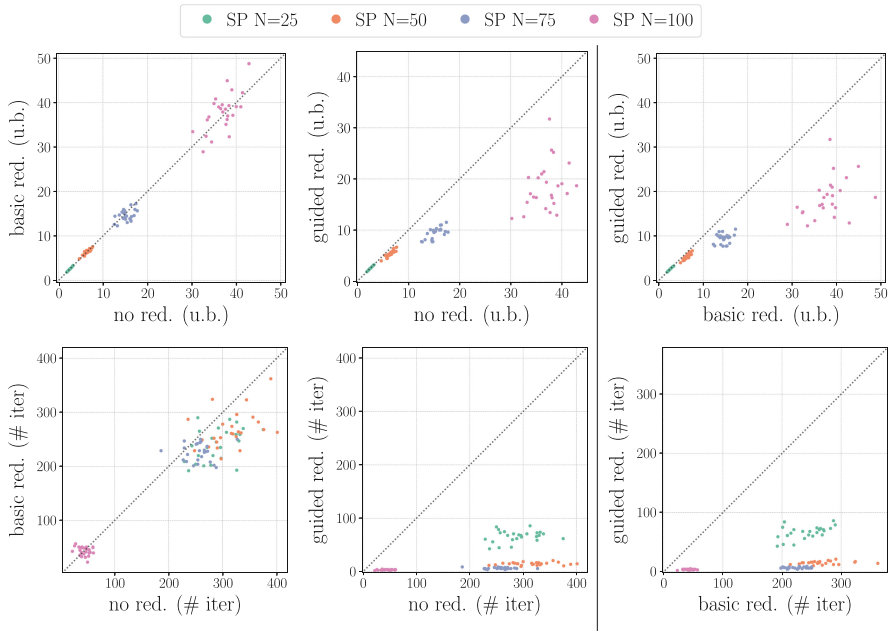


Fig. 5. Results on OMT($\mathcal{LRA} \cup \mathcal{AR}$)-encoded strip-packing problems.

better solutions. On the other hand, the guided reduction strategy is much more effective, since it allows finding a much better upper bound within the timeout (first row, center and right columns), and the number of iterations is drastically reduced (second row, center and right columns).

OMT($\mathcal{LIRA}$) Benchmarks. The results are summarized in Fig. 4.

Strip-Packing (Fig. 4). The plots show that also for $\mathcal{LIRA}$, the trend is similar to the one for $\mathcal{LRA}$. In fact, the basic reduction strategy is not very effective for improving the upper bound (first row, left column), and the number of iterations is only slightly reduced (second row, left column). On the other hand, the guided reduction allows finding a much better upper bound within the timeout (first row, center and right columns). Notice that, since these problems are much harder than the OMT($\mathcal{LRA}$) ones, the number of iterations completed within the timeout is much smaller, so that the upper bounds found are also bigger.

OMT($\mathcal{LRA} \cup \mathcal{AR}$) Benchmarks. The results are summarized in Fig. 5.

Strip-Packing (Fig. 5). Similarly, here we can see that the technique works also for combination of theories, such as $\mathcal{LRA} \cup \mathcal{AR}$. The results are similar to the ones for $\mathcal{LRA}$, and the advantage of the guided reduction is even more evident.

Discussion. The results show that applying either form of truth-assignment reduction can be beneficial in OMT, both for optimal and anytime solving, and that accurately

selecting which atoms to remove from the truth assignment can make a significant difference in finding better solutions in fewer iterations.

We observe, however, that a much smaller number of iterations, i.e. of truth assignments enumerated, not always correlates linearly with the solving time. This can be due to several reasons.

First, we remark that global efficiency of OMT depends on several factors, including the enumeration order of truth assignments, and different literal selections may alter this order. Also, the removal of some literals from the assignments can prevent the removal of others in subsequent iterations. The effects of these factors are quite unpredictable.

Second, we notice that in these problems, the number of truth assignments enumerated is typically contained to a few hundred. In fact, in OMT the bounds on the objective function already allow performing a very effective pruning of the search space.

Moreover, this pruning is typically done by theory reasoning, and most of it has to be done anyway, regardless of the number of truth assignments enumerated. Making it in a single iteration or in many iterations may not reflect as much on the solving time, because of the efficient incrementality of SMT solvers, which can reduce a lot the cost of consecutive iterations.

6 Conclusions and Future Work

In this paper, we have investigated the role of truth assignment enumeration in OMT solving, and proposed some ways for exploiting partial truth assignments for improving the efficiency and effectiveness of the search. In particular, we have proposed a truth assignment reduction strategy that takes advantage of the properties of the optimization problem to accurately choose the atoms to remove from the truth assignment.

We have implemented the proposed strategies in the OptiMathSAT solver, and evaluated them on a set of OMT($\mathcal{LRA}$), OMT($\mathcal{LIRA}$), and OMT($\mathcal{LRA} \cup \mathcal{AR}$) benchmarks. Our experimental results show that the proposed strategies can significantly improve the performance of the solver, uniformly reducing the overall solving time for optimal solving, and finding much better solutions for anytime solving for all the analyzed theories.

Other truth assignment reduction strategies, such as entailment-based methods [12], have shown significant benefits for SAT enumeration. Their extension to OMT problems could be a promising direction for future work.

Acknowledgements. RS was partially supported by the project "AI@TN" funded by the Autonomous Province of Trento. RS was partially supported by the MUR PNRR project FAIR - Future AI Research (PE00000013) funded by the NextGenerationEU. RS was partially funded under the NRRP, Mission 4 Component 2 Investment 1.4, by the European Union—NextGenerationEU (proj. nr. CN 00000013). RS was supported in part by the TANGO project funded by the EU Horizon Europe research and innovation program under GA No 101120763, funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union, the European Health and Digital Executive Agency (HaDEA) or The European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Albert, E., Correas, J., Gordillo, P., Román-Díez, G., Rubio, A.: GASOL: gas analysis and optimization for Ethereum smart contracts. In: TACAS 2020. LNCS, vol. 12079, pp. 118–125. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45237-7_7
2. Barrett, C., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Satisfiability, FAIA, vol. 336, pp. 1267–1329, 2 edn. IOS Press (2021). https://doi.org/10.3233/FAIA201017
3. Bian, Z., Chudak, F., Macready, W., Roy, A., Sebastiani, R., Varotti, S.: Solving SAT (and MaxSAT) with a quantum annealer: foundations, encodings, and preliminary results. Inf. Comput. **275**, 104609 (2020). https://doi.org/10.1016/j.ic.2020.104609
4. Bigarella, F., et al.: Optimization modulo non-linear arithmetic via incremental linearization. In: Konev, B., Reger, G. (eds.) FroCoS 2021. LNCS (LNAI), vol. 12941, pp. 213–231. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86205-3_12
5. Bjørner, N., Phan, A.D.: νZ - maximal satisfaction with Z3. In: Proceedings of the International Symposium on Symbolic Computation in Software Science, pp. 1–9 (2014). https://doi.org/10.29007/jmxj
6. Bjørner, N., Phan, A.-D., Fleckenstein, L.: νZ - an optimizing SMT solver. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 194–199. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_14
7. Bofill, M., Coll, J., Suy, J., Villaret, M.: An efficient SMT approach to solve MRCPSP/max instances with tight constraints on resources. In: Beck, J.C. (ed.) CP 2017. LNCS, vol. 10416, pp. 71–79. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66158-2_5
8. Bozzano, M., et al.: Efficient Theory Combination via Boolean Search. Inf. Comput. **204**(10), 1493–1525 (2006). https://doi.org/10.1016/j.ic.2005.05.011
9. Cimatti, A., Griggio, A., Schaafsma, B.J., Sebastiani, R.: The MathSAT5 SMT solver. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 93–107. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36742-7_7
10. Ding, J., Spallitta, G., Sebastiani, R.: Effective prime factorization via quantum annealing by modular locally-structured embedding. Sci. Rep. **14**(1), 3518 (2024). https://doi.org/10.1038/s41598-024-53708-7
11. Dutertre, B., de Moura, L.: A fast linear-arithmetic solver for DPLL(T). In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 81–94. Springer, Heidelberg (2006). https://doi.org/10.1007/11817963_11
12. Fried, D., Nadel, A., Sebastiani, R., Shalmon, Y.: Entailing generalization boosts enumeration. In: SAT 2024. LIPIcs, vol. 305, pp. 13:1–13:14. LZI (2024). https://doi.org/10.4230/LIPIcs.SAT.2024.13
13. Fried, D., Nadel, A., Shalmon, Y.: AllSAT for combinational circuits. In: SAT 2023. LIPIcs, vol. 271, pp. 9:1–9:18. LZI (2023). https://doi.org/10.4230/LIPIcs.SAT.2023.9
14. Henry, J., Asavoae, M., Monniaux, D., Maïza, C.: How to compute worst-case execution time by optimization modulo theory and a clever encoding of program semantics. In: LCTES 2014, pp. 43–52. ACM (2014). https://doi.org/10.1145/2597809.2597817
15. Lahiri, S.K., Nieuwenhuis, R., Oliveras, A.: SMT techniques for fast predicate abstraction. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 424–437. Springer, Heidelberg (2006). https://doi.org/10.1007/11817963_39
16. Leofante, F., Giunchiglia, E., Ábrahám, E., Tacchella, A.: Optimal planning modulo theories. In: IJCAI 2020, vol. 4, pp. 4128–4134 (2020). https://doi.org/10.24963/ijcai.2020/571
17. Li, Y., Albarghouthi, A., Kincaid, Z., Gurfinkel, A., Chechik, M.: Symbolic optimization with SMT solvers. In: POPL 2014, pp. 607–618. ACM (2014). https://doi.org/10.1145/2535838.2535857

18. Ma, F., Yan, J., Zhang, J.: Solving generalized optimization problems subject to SMT constraints. In: Snoeyink, J., Lu, P., Su, K., Wang, L. (eds.) AAIM/FAW -2012. LNCS, vol. 7285, pp. 247–258. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29700-7_23
19. Marques-Silva, J., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Handbook of Satisfiability, FAIA, vol. 336. IOS Press (2021). https://doi.org/10.3233/FAIA200987
20. Masina, G., Spallitta, G., Sebastiani, R.: On CNF conversion for disjoint SAT enumeration. In: SAT 2023. LIPIcs, vol. 271, pp. 15:1–15:16. LZI (2023). https://doi.org/10.4230/LIPIcs.SAT.2023.15
21. Masina, G., Spallitta, G., Sebastiani, R.: On CNF conversion for SAT and SMT enumeration. J. Artif. Intell. Res. **83** (2025). https://doi.org/10.1613/jair.1.16870, to appear, preprint arXiv:2303.14971
22. Morgado, A., Marques-Silva, J.: Good learning and implicit model enumeration. In: ICTAI 2005, pp. 131–136. IEEE Computer Society (2005). https://doi.org/10.1109/ICTAI.2005.69
23. Nadel, A., Ryvchin, V.: Bit-vector optimization. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 851–867. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_53
24. Nguyen, C.M., Sebastiani, R., Giorgini, P., Mylopoulos, J.: Multi-objective reasoning with constrained goal models. Requirements Eng. **23**(2), 189–225 (2016). https://doi.org/10.1007/s00766-016-0263-5
25. Nieuwenhuis, R., Oliveras, A.: On SAT modulo theories and optimization problems. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 156–169. Springer, Heidelberg (2006). https://doi.org/10.1007/11814948_18
26. Panjkovic, S., Micheli, A.: Expressive optimal temporal planning via optimization modulo theory. In: AAAI 2023, vol. 37, no. 10, pp. 12095–12102 (2023). https://doi.org/10.1609/aaai.v37i10.26426
27. Panjkovic, S., Micheli, A.: Abstract action scheduling for optimal temporal planning via OMT. In: AAAI 2024, vol. 38, no. 18, pp. 20222–20229 (2024). https://doi.org/10.1609/aaai.v38i18.30002
28. Plaisted, D.A., Greenbaum, S.: A Structure-preserving clause form translation. J. Symb. Comput. **2**(3), 293–304 (1986). https://doi.org/10.1016/S0747-7171(86)80028-1
29. Ravi, K., Somenzi, F.: Minimal assignments for bounded model checking. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 31–45. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24730-2_3
30. Sebastiani, R.: Lazy satisfiability modulo theories. JSAT **3**(3–4), 141–224 (2007). https://doi.org/10.3233/SAT190034
31. Sebastiani, R., Tomasi, S.: Optimization in SMT with LA(Q) cost functions. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS (LNAI), vol. 7364, pp. 484–498. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31365-3_38
32. Sebastiani, R., Tomasi, S.: Optimization modulo theories with linear rational costs. ACM Trans. Comput. Logic **16**(2), 12:1–12:43 (2015). https://doi.org/10.1145/2699915
33. Sebastiani, R., Trentin, P.: Pushing the envelope of optimization modulo theories with linear-arithmetic cost functions. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 335–349. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_27
34. Sebastiani, R., Trentin, P.: OptiMathSAT: a tool for optimization modulo theories. J. Autom. Reason. **64**(3), 423–460 (2018). https://doi.org/10.1007/s10817-018-09508-6
35. Spallitta, G., Sebastiani, R., Biere, A.: Disjoint partial enumeration without blocking clauses. In: AAAI 2024, vol. 38, pp. 8126–8135 (2024). https://doi.org/10.1609/aaai.v38i8.28652

36. Spallitta, G., Sebastiani, R., Biere, A.: Disjoint projected enumeration for SAT and SMT without blocking clauses. Artif. Intell. **345**, 104346 (2025). https://doi.org/10.1016/j.artint.2025.104346
37. Teso, S., Sebastiani, R., Passerini, A.: Structured learning modulo theories. Artif. Intell. **244**, 166–187 (2017). https://doi.org/10.1016/j.artint.2015.04.002
38. Toda, T., Soh, T.: Implementing efficient all solutions SAT solvers. ACM J. Exp. Algorithmics **21**, 1–44 (2016). https://doi.org/10.1145/2975585
39. Trentin, P., Sebastiani, R.: Optimization modulo the theories of signed bit-vectors and floating-point numbers. J. Autom. Reason. **65**(7), 1071–1096 (2021). https://doi.org/10.1007/s10817-021-09600-4
40. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Siekmann, J.H., Wrightson, G. (eds.) Automation of Reasoning. Symbolic Computation, pp. 466–483. Springer, Heidelberg (1983). https://doi.org/10.1007/978-3-642-81955-1_28
41. Tsiskaridze, N., Barrett, C., Tinelli, C.: Generalized optimization modulo theories. In: Benzmüller, C., Heule, M.J., Schmidt, R.A. (eds.) IJCAR 2024. LNCS, vol. 14739, pp. 458–479. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-63498-7_27