Optimization Modulo Non-Linear Arithmetic via Incremental Linearization

Filippo Bigarella[1], Alessandro Cimatti[2], Alberto Griggio[2], Ahmed Irfan[3], Martin Jonáš[2], Marco Roveri[1], Roberto Sebastiani[1], and Patrick Trentin[1]

[1] DISI, University of Trento, Italy
[2] Fondazione Bruno Kessler - FBK, Trento, Italy
[3] The author's contribution dates when he was still at FBK, Trento

Abstract. Incremental linearization is a conceptually simple, yet effective, technique that we have recently proposed for solving SMT problems on the theories of non-linear arithmetic over the reals and the integers. Optimization Modulo Theories (OMT) is an important extension of SMT which allows for finding models that optimize given objective functions.
In this paper, we show how incremental linearization can be extended to OMT in a simple way, producing an incomplete though effective OMT procedure. We describe the main ideas and algorithms, we provide an implementation within the OptiMathSAT OMT solver, and perform an empirical evaluation. The results support the effectiveness of the approach.

1 Introduction

Context. Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of a first-order formula with respect to some theories of interest (e.g. theory of linear arithmetic, of arrays, of bit-vectors,...) and their combination thereof [5]. Powerful and effective SMT techniques and tools are available for a large variety of theories, including the quantifier-free theories[1] of Uninterpreted Functions (UF) and Linear Arithmetic (LA), either over the reals (LRA) or the integers (LIA), as well as their combinations (UFLRA, UFLIA), of bit-vector (BV) and floating-point arithmetic (FP).

When dealing with arithmetic, a fundamental challenge is to go beyond the linear case, by introducing multiplications between variables, and hence between complex terms – over the reals (NRA) or over the integers (NIA). (We also use the term "NIRA" [resp. "LIRA"] when we do not distinguish between NRA and NIA [resp. LRA and LIA].) Unfortunately, dealing with non-linearity is a very hard challenge. Going from SMT(LRA) to SMT(NRA) yields a complexity gap that results in a computational barrier in practice – most available complete solvers rely on Cylindrical algebraic decomposition (CAD) techniques [15], which require double exponential time in worst case. Reasoning in NIA is even undecidable [30].

Incremental Linearization. Recently, we have proposed a conceptually-simple, incomplete yet effective practical approach for SMT dealing with the quantifier-free theory

[1] In the following, we only consider quantifier-free theories, and we abuse the accepted notation and omit the "QF_" prefix in the names of the theories.

of non-linear arithmetic over the reals, over the reals extended with transcendental functions, and over integers, called *Incremental Linearization* [13]. Its underlying idea is that of trading the use of expensive, exact solvers for non-linear arithmetic for an abstraction-refinement loop on top of much cheaper solvers for linear arithmetic and uninterpreted functions.

Optimization Modulo Theories. Many SMT problems of interest, however, require the capability of finding models that are optimum wrt. some objective functions. These problems are grouped under the umbrella term of Optimization Modulo Theories – OMT [32,34,8]. OMT techniques have been conceived for a variety of theories, including LRA [34,8], LIA [8,36], BV [31], FP [39]. In general, they work by performing sequences of incremental SMT calls, possibly combined with theory-specific optimization techniques for the conjunctive fragment of the given theory, which progressively tighten the range of values of the objective function.

OMT by Incremental Linearization. In this paper, we show how incremental linearization can be extended from SMT to OMT in a very simple way, producing an incomplete though effective OMT procedure. As with the SMT case in [13], the goal is to build an OMT(NIRA) solver on top of "cheap" ingredients: SMT(UFLIRA) and OMT(UFLIRA) incremental calls driven by an abstraction-refinement loop, with no expensive solver or optimizer for non-linear arithmetic, so that the task of progressing towards the optimum is performed by a combination of Boolean search and optimization in the abstract UFLIRA space. We describe the main ideas and algorithms.

We have implemented the novel OMT(NIRA) algorithms within the OMT solver OPTIMATHSAT [37], which is built on top of the MATHSAT5 SMT solver [14], where the incremental linearization for SMT(NIRA) procedures have been implemented [13]. We have experimentally validated our algorithm with an analysis of the performance of OPTIMATHSAT in dealing with OMT(NRA) and OMT(NIA) problems, and compared these results with those of another state-of-the-art OMT solver, Z3 [8]. Although quite preliminary to some extent, the results show that, despite the simplicity of the implemented procedures, OPTIMATHSAT performs comparably with this tool, thus supporting the effectiveness of the approach.

Related Work. Efficient SMT solving for non-linear arithmetic is an open research problem for which a variety of approaches have been proposed; these are often complementary with one another [13]. Methods for dealing with SMT(NRA) are typically based on: *cylindrical algebraic decomposition* (CAD) [15,23,25,3], *virtual substitution* (VS) [40], *interval constraint propagation* (ICP) [6,21], *bit-blasting* [20,41], *linearization* [10,28,19,33,9,11] and *incremental linearization* [13]. Methods for dealing with SMT(NIA) are based on the combination of branch-and-bound search with some SMT(NRA) solving technique. See, for example, [22] and [26] based on CAD and CAD+VS respectively, and [12] based on *incremental linearization*.

OMT for non-linear arithmetic, instead, is a largely unexplored territory.

The Z3 [8] OMT solver has some kind of support for optimization with NIRA objective functions and/or NIRA constraints. However, this functionality does not appear to be documented in any publication that we are aware of.

DREAL is a SMT(NRA) solver based on the notion of δ-satisfiability [21], that basically guarantees that there exists a variant (within a user-specified δ "radius") of the original problem such that it is satisfiable. Importantly, we notice that the approach cannot guarantee that the original problem is satisfiable, since it relies on numerical approximation techniques that only compute safe over-approximations of the solution space. DREAL supports optimization with NRA objective functions and/or in the presence of NRA constraints [24].

Content. The paper is organized as follows. In §2 we provide some background knowledge on OMT and incremental linearization; in §3 we present our main ideas and new OMT(NIRA) procedures; in §4 we perform the empirical evaluation, discussing the results; in §5 we draw some conclusions, and illustrate possible future research directions.

2 Background

2.1 Optimization Modulo Theories

We assume the reader is familiar with the main theoretical and algorithmic concepts in SAT and SMT solving (see e.g. [29,5]). Optimization Modulo Theories (OMT) is an extension of SMT which addresses the problem of finding a model for an input formula φ which is optimal wrt. some objective function obj [32,34]. (In this paper we consider optimization as minimization; the narration for maximization is dual.) A little more formally, given some theory $\mathcal{T}$ admitting some total order "$\leq$" over its domain, an OMT($\mathcal{T}$) problem is given by a pair $\langle \varphi, \mathsf{obj} \rangle$ where φ, obj are a formula and a term over $\mathcal{T}$, and consists in finding a model for φ (if any) which makes the value of obj minimum according to the order given by $\leq$. [2]

The basic minimization strategy implemented in all state-of-the-art OMT solvers, is known as *linear-search* strategy [32,34,35]. It requires solving an SMT problem with a feasible space that is progressively tightened by learning a sequence of unit clauses of the form $(\mathsf{obj} < \mathsf{ub})$, where ub is the current upper bound of the optimization search. At each iteration, the SMT solver can either find a model $\mathcal{M}$ of φ whose value of obj, denoted with $\mathcal{M}[\mathsf{obj}]$, is smaller than the current upper bound ub, or discover that the stack of formulas has become unsatisfiable. In the first case, the OMT solver invokes a $\mathcal{T}$-specific minimization procedure over the propositional truth assignment ψ induced by $\mathcal{M}$ on the atoms of the formula, so as to generate a model $\mathcal{M}'$ of φ such that the value of $\mathcal{M}'[\mathsf{obj}]$ is minimum for the given propositional assignment ψ. Then, the $\mathcal{M}'[\mathsf{obj}]$ becomes the new upper bound ub of the optimization search. The OMT search terminates when such procedure finds that obj is unbounded or when the SMT search is UNSAT, in which case the last model $\mathcal{M}_i$ of φ (if any) is the optimal solution.

If, for some theory $\mathcal{T}$, $\mathcal{T}$-specific minimization is hard to implement (e.g. for floating-point arithmetic [39]) or computationally too expensive (e.g., for linear integer arithmetic [36]), then we can rely on other strategies.

One possible approach for the latter case is to implement a cheaper though incomplete $\mathcal{T}$-specific minimization procedure, which may only improve the value of $\mathcal{M}[\mathsf{obj}]$

[2] More generally the formula can be built on a combination of $\mathcal{T}$ with other theories, see e.g. [35]. However, to simplify the narration and the notation, here we refer to one single theory.

with no guarantee to find a minimum one for the given truth assignment. This comes at the risk of generating and exploring the same truth assignment more than once, overall trading arithmetic minimization search for extra Boolean search [36].

An alternative general minimization approach is the *binary-search* strategy described in [34]. At the beginning of each binary search step, the OMT solver calls a function $\mathsf{ComputePivot}()$ to compute a pivoting value contained in the current search interval. (In its simplest implementation, $\mathsf{ComputePivot}()$ returns the value of $\frac{(\mathsf{ub}-\mathsf{lb})}{2}$, where lb and ub are the lower and the upper bound currently delimiting the optimization search respectively). Then, the OMT solver temporarily assumes a unit-clause of the form $(\mathsf{obj} < \mathsf{pivot})$. This effectively restricts the search space so that it includes only satisfiable truth assignments (if any) for which obj has a value included in the interval $[\mathsf{lb}, \mathsf{pivot})$. If any such solution is found by the OMT solver, the algorithm proceeds like in *linear-search* mode, and updates the current upper bound ub. Otherwise, if no such solution exists, then the pivoting unit-clause $(\mathsf{obj} < \mathsf{pivot})$ is replaced by its negation $\neg(\mathsf{obj} < \mathsf{pivot})$ and pivot becomes the new lower bound. Notice that, in case of continuous domains (e.g., LRA, NRA) the binary search alone may not terminate, so that it is necessary to interleave binary-search steps with linear-search ones [35].

2.2 SMT(NIRA) via Incremental Linearization

The main idea of incremental linearization [13] is to trade the use of expensive, exact solvers for non-linear arithmetic for an abstraction-refinement loop on top of much cheaper solvers for linear arithmetic and uninterpreted functions, UFLIRA. The pseudo-code of the baseline procedure is shown in Fig. 1.

First, the input SMT(NIRA) formula φ is abstracted to the SMT(UFLIRA) formula $\widehat{\varphi}$ (called its UFLIRA-abstraction) by substituting every non-linear multiplication term $x * y$ with $f_*(x, y)$, where both x and y are variables [3] and f_* is an uninterpreted function symbol (function SMT-INITIAL-ABSTRACTION in line 1). Then, the set of linearization lemmas Γ (i.e. UFLIRA-abstraction of NIRA-valid multiplication lemmas in Fig. 3) is initialized to the empty set, and the loop begins. [4]

First, the UFLIRA-satisfiability of $\widehat{\varphi}$ augmented with the linearization lemmas in Γ is checked. If the SMT(UFLIRA) check returns false, then the input formula is NIRA-unsatisfiable, because $\widehat{\varphi} \wedge \bigwedge \Gamma$ is an over-approximation of φ by construction (lines 4-6).

Otherwise, the abstract model $\hat{\mu}$ for $\widehat{\varphi}$ is used to build an UFLIRA under-approximation $\widehat{\psi}^*$ of φ, with the aim of finding a model μ for $\widehat{\psi}^*$, and thus for the original NIRA formula φ (function SMT-CHECK-REFINE in line 7). If this succeeds, then φ is also satisfiable. Otherwise, SMT-CHECK-REFINE returns a set Γ' of linear *lemmas* which

[3] As in [13] and with no loss of generality, hereafter we assume that all multiplications in φ are either between two variables or between one constant and one variable, because more complex terms occurring in a multiplication can be renamed by fresh variables. Notice that this assumption is not necessary in practice, but it simplifies the explanation.

[4] In order to keep the narration simple, in Fig. 1 we have omitted some details. First, the input formula can be simplified by some preprocessing steps. Furthermore, for each fresh $f_*(x, y)$ term, $\hat{\varphi}$ can be extended from the beginning with simple multiplication lemmas. We refer the reader to [13] for details.

function SMT-INCREMENTAL-LINEARIZATION(φ)

1: $\hat{\varphi} \coloneqq$ SMT-INITIAL-ABSTRACTION(φ) ; // $\hat{\varphi}$ over-approximates φ
2: $\Gamma \coloneqq \emptyset$; // linearization lemmas
3: **while** $true$ **do**
4: $\langle sat, \hat{\mu} \rangle \coloneqq$ SMT-UFLIRA-CHECK($\hat{\varphi} \wedge \bigwedge \Gamma$)
// $\hat{\varphi} \wedge \bigwedge \Gamma$ over-approximates φ
5: **if not** sat **then**
6: **return** $\langle false, \emptyset \rangle$
7: $\langle sat, \mu, \Gamma' \rangle \coloneqq$ SMT-CHECK-REFINE($\hat{\varphi}, \hat{\mu}$)
8: **if** sat **then**
9: **return** $\langle true, \mu \rangle$
10: **else**
11: $\Gamma \coloneqq \Gamma \cup \Gamma'$

Fig. 1. Solving SMT(NIRA) via incremental linearization.

function SMT-CHECK-REFINE($\hat{\varphi}, \hat{\mu}$)

1: $\hat{\psi} \coloneqq$ EXTRACTASSIGNMENT($\hat{\varphi}, \hat{\mu}$) ; // Eq. (1)
2: $\hat{\psi}^* \coloneqq \hat{\psi} \wedge$ LINEARIZATION-CONSTRAINTS($\hat{\psi}$) ; // Eq. (2)
3: $\langle sat, \hat{\mu}' \rangle \coloneqq$ SMT-UFLIRA-CHECK($\hat{\psi}^*$)
4: **if** sat **then**
5: **return** $\langle true, \hat{\mu}', \emptyset \rangle$; // $\hat{\mu}'$ is a model of φ
6: **else**
7: $\Gamma \coloneqq$ BLOCK-SPURIOUS-PRODUCT-TERMS($\hat{\varphi}, \hat{\mu}$)
8: **return** $\langle false, \emptyset, \Gamma \rangle$

Fig. 2. The procedure checking whether $\hat{\mu}$ can be refined into a model of φ.

are sufficient to rule out the spurious model $\hat{\mu}$. Γ' is then added to Γ, thus improving the precision of the abstraction, and another iteration of the loop is performed.

The lemmas added are instances of the axioms of Fig. 3, obtained by replacing the free variables with terms occurring in φ, selected among those that evaluate to false under the current spurious model μ.

The pseudo-code of the model search procedure (SMT-CHECK-REFINE) is reported in Fig. 2. In particular, $\widehat{\psi}^*$ is built by first generating the truth assignment $\widehat{\psi}$ for the atoms in $\widehat{\varphi}$ which is entailed by the current abstract model $\widehat{\mu}$, and then by adding *multiplication line constraints* that force all multiplications in ψ to be linear (lines 1-2):

$$\hat{\psi} = \bigwedge_{[\hat{A} \in atoms(\hat{\varphi}) \; s.t. \; \hat{\mu} \models \hat{A}]} \hat{A} \; \wedge \bigwedge_{[\hat{A} \in atoms(\hat{\varphi}) \; s.t. \; \hat{\mu} \not\models \hat{A}]} \neg \hat{A} \tag{1}$$

$$\hat{\psi}^* = \hat{\psi} \wedge \bigwedge_{f_*(x,y) \in \hat{\psi}} (x = \hat{\mu}[x] \wedge f_*(x,y) = \hat{\mu}[x] * y)) \vee (y = \hat{\mu}[y] \wedge f_*(x,y) = \hat{\mu}[y] * x)) \tag{2}$$

Then the UFLIRA-satisfiability of $\widehat{\psi}^*$ is checked: if satisfiable, then φ is NIRA-satisfiable and its model $\hat{\mu}'$ is also a model for φ (lines 3-5). Otherwise, a set Γ of lemmas ruling out the spurious model $\hat{\mu}'$ is produced and returned (lines 6-8).

Basic:	*Sign:*	$v_1 * v_2 = (-v_1 * -v_2)$
		$v_1 * v_2 = -(-v_1 * v_2)$
		$v_1 * v_2 = -(v_1 * -v_2)$
	Zero:	$(v_1 = 0 \vee v_2 = 0) \leftrightarrow v_1 * v_2 = 0$
		$((v_1 > 0 \wedge v_2 > 0) \vee (v_1 < 0 \wedge v_2 < 0)) \leftrightarrow v_1 * v_2 > 0$
		$((v_1 < 0 \wedge v_2 > 0) \vee (v_1 > 0 \wedge v_2 < 0)) \leftrightarrow v_1 * v_2 < 0$
	Neutral:	$(v_1 = 1 \vee v_2 = 0) \leftrightarrow v_1 * v_2 = v_2$
		$(v_2 = 1 \vee v_1 = 0) \leftrightarrow v_1 * v_2 = v_1$
	Proportionality:	$\lvert v_1 * v_2\rvert \geq \lvert v_2\rvert \leftrightarrow (\lvert v_1\rvert \geq 1 \vee v_2 = 0)$
		$\lvert v_1 * v_2\rvert \leq \lvert v_2\rvert \leftrightarrow (\lvert v_1\rvert \leq 1 \vee v_2 = 0)$
		$\lvert v_1 * v_2\rvert \geq \lvert v_1\rvert \leftrightarrow (\lvert v_2\rvert \geq 1 \vee v_1 = 0)$
		$\lvert v_1 * v_2\rvert \leq \lvert v_1\rvert \leftrightarrow (\lvert v_2\rvert \leq 1 \vee v_1 = 0)$
Order:		$(v_1 * v_2 \bowtie v_3 \wedge v_4 > 0) \rightarrow v_1 * v_2 * v_4 \bowtie v_3 * v_4$
		$(v_1 * v_2 \bowtie v_3 \wedge v_4 < 0) \rightarrow v_3 * v_4 \bowtie v_1 * v_2 * v_4$
Monotonicity:		$(\lvert v_1\rvert \leq \lvert v_2\rvert \wedge \lvert v_3\rvert \leq \lvert v_4\rvert) \rightarrow \lvert v_1 * v_3\rvert \leq \lvert v_2 * v_4\rvert$
		$(\lvert v_1\rvert < \lvert v_2\rvert \wedge \lvert v_3 \leq \lvert v_4\rvert \wedge v_4 \neq 0) \rightarrow \lvert v_1 * v_3\rvert < \lvert v_2 * v_4\rvert$
		$(\lvert v_1\rvert \leq \lvert v_2\rvert \wedge \lvert v_3\rvert < \lvert v_4\rvert \wedge v_2 \neq 0) \rightarrow \lvert v_1 * v_3\rvert < \lvert v_2 * v_4\rvert$
Tangent plane:		$v_1 = a \rightarrow v_1 * v_2 = a * v_2$
		$v_2 = b \rightarrow v_1 * v_2 = b * v_1$
		$(v_1 > a \wedge v_2 < b) \rightarrow v_1 * v_2 < b * v_1 + a * v_2 - a * b$
		$(v_1 < a \wedge v_2 > b) \rightarrow v_1 * v_2 < b * v_1 + a * v_2 - a * b$
		$(v_1 < a \wedge v_2 < b) \rightarrow v_1 * v_2 > b * v_1 + a * v_2 - a * b$
		$(v_1 > a \wedge v_2 > b) \rightarrow v_1 * v_2 > b * v_1 + a * v_2 - a * b$

Fig. 3. Axioms of the multiplication function.

3 Optimization Modulo Non-Linear Arithmetic

In this section, we present our novel Optimization Modulo Non-Linear Arithmetic procedure based on the combination of the optimization schema described in §2.1 and the incremental linearization approach described in §2.2. In §3.1, we describe the basic version of this new algorithm, based on the linear optimization search schema, and discuss the termination of the algorithm. Then, in §3.2, we describe some simple improvements over the basic approach that can significantly improve the effectiveness of the procedure, despite remaining incomplete.

3.1 Linear Optimization Search

The pseudo-code of the basic approach to Optimization Modulo Non-Linear Arithmetic, based on the linear optimization search schema, is shown in Fig. 4.

Input & Initialization. The algorithm takes as input a SMT(NIRA) formula φ, a LIRA objective obj and a threshold precision value ϵ. (We can assume wlog. that obj is a LIRA term because, if not so, we can rewrite $\langle \varphi, \mathsf{obj} \rangle$ into $\langle \varphi \wedge (v = \mathsf{obj}), v \rangle$, v being

function OMT-NIRA-LINEAR-SEARCH(φ, obj, ϵ)

1: $\hat{\varphi} \coloneqq$ SMT-INITIAL-ABSTRACTION(φ) // $\hat{\varphi}$ over-approximates φ
2: $\mathcal{M} \coloneqq \emptyset$ // current best model
3: $\Theta \coloneqq \emptyset$ // optimization constraints as (obj < ub)
4: $\Gamma \coloneqq \emptyset$ // linearization lemmas
5: **while** $true$ **do**
6: $\langle sat, \hat{\mu} \rangle \coloneqq$ SMT-UFLIRA-CHECK($\hat{\varphi} \wedge \bigwedge \Gamma \wedge \bigwedge \Theta$) // $\hat{\mu}$ abstract model
// $\hat{\varphi} \wedge \bigwedge \Gamma$ over-approximates φ
7: **if not** sat **then**
8: **break**
9: $\langle sat, \mu, \Gamma' \rangle \coloneqq$ SMT-CHECK-REFINE($\hat{\varphi} \wedge \bigwedge \Theta, \hat{\mu}$)
10: **if** sat **then**
11: $\mathcal{M} \coloneqq \mu$
12: $\hat{\psi} \coloneqq$ EXTRACTASSIGNMENT($\hat{\varphi}, \mu$) // Eq. (1)
13: $\hat{\mu}' \coloneqq$ UFLIRA-MINIMIZE($\hat{\psi}$, obj)
14: $\langle sat, \mu', \Gamma' \rangle \coloneqq$ OMT-CHECK-REFINE($\hat{\varphi} \wedge \bigwedge \Theta, \hat{\mu}'$, obj)
15: **if** sat **then**
16: $\mathcal{M} \coloneqq \mu'$ // $\hat{\mu}'[\text{obj}] \leq \mu'[\text{obj}] \leq \mu[\text{obj}]$
17: ub $\coloneqq$ GET-UPPER-BOUND($\hat{\mu}'[\text{obj}], \mathcal{M}[\text{obj}], \epsilon$)
18: $\Theta \coloneqq \Theta \cup \{(\text{obj} < \text{ub})\}$ // linear-search step
19: $\Gamma \coloneqq \Gamma \cup \Gamma'$
20: **if** $\mathcal{M} \neq \emptyset$ **then**
21: **return** $\langle$SAT, $\mathcal{M}\rangle$
22: **else**
23: **return** $\langle$UNSAT, $\emptyset\rangle$

Fig. 4. A baseline schema of our OMT(NIRA) procedure, with linear search.

function OMT-CHECK-REFINE($\hat{\varphi}$, $\hat{\mu}$, obj)

1: $\hat{\psi} \coloneqq$ EXTRACTASSIGNMENT($\hat{\varphi}, \hat{\mu}$) // Eq. (1)
2: $\hat{\psi}^* \coloneqq \hat{\psi} \wedge$ LINEARIZATION-CONSTRAINTS($\hat{\psi}$) // Eq. (2)
3: $\langle sat, \hat{\mu}' \rangle \coloneqq$ OMT-UFLIRA-CHECK-MINIMIZE($\hat{\psi}^*$, obj)
4: **if** sat **then**
5: **return** $\langle true, \hat{\mu}', \emptyset\rangle$ // $\hat{\mu}'$ is a model of φ
6: **else**
7: $\Gamma \coloneqq$ BLOCK-SPURIOUS-PRODUCT-TERMS($\hat{\varphi}, \hat{\mu}$)
8: **return** $\langle false, \emptyset, \Gamma\rangle$

Fig. 5. The OMT counterpart of the SMT-CHECK-REFINE in Fig. 5.

a fresh variable.) Lines 1-4 are part of the startup phase of the OMT(NIRA) algorithm. First, the UFLIRA abstraction $\hat{\varphi}$ over-approximating φ is computed at line 1. Next, the current best model $\mathcal{M}$, the set of optimization constraints Θ and the set of linearization lemmas Γ are initialized to the empty set (lines 2-4). [5]

[5] The same considerations as in Footnote 4 apply here as well.

Main Loop. Given the current best model $\mathcal{M}$, optimization constraints Θ and linearization constraints Γ, the algorithm enters into its main loop (lines 5-19), which can be virtually divided into two distinct blocks.

The first block, lines 6-9, corresponds to a single application of the incremental linearization schema presented in Fig. 1, lines 4-7. The goal of this phase is to find an initial abstract UFLIRA model $\hat{\mu}$ of $\hat{\varphi} \wedge \bigwedge \Gamma \wedge \bigwedge \Theta$ that is refined into a NIRA model μ of $\varphi \wedge \bigwedge \Theta$. This step can have three possible outcomes.

(i) SMT-UFLIRA-CHECK fails to find such $\hat{\mu}$ because $\hat{\varphi} \wedge \bigwedge \Gamma \wedge \bigwedge \Theta$ is UFLIRA-unsatisfiable. This implies that $\varphi \wedge \bigwedge \Theta$ is also NIRA-unsatisfiable, because the former is an over-approximation of the latter. Thus, there is no better model than the current one (if any), so that the execution breaks out of the loop (lines 6-8);

(ii) SMT-UFLIRA-CHECK succeeds in finding an UFLIRA model $\hat{\mu}$, but SMT-CHECK-REFINE fails to refine $\hat{\mu}$ into a NIRA model μ for $\varphi \wedge \bigwedge \Theta$ (lines 6,9). [6] If so, SMT-CHECK-REFINE returns a set of UFLIRA constraints Γ' ruling out $\hat{\mu}$ (and other spurious solutions) from the feasible search space, and the procedure skips the second block of lines 10-18, jumping directly to line 19;

(iii) $\hat{\mu}$ is successfully refined and rewritten into a NIRA model μ for $\varphi \wedge \bigwedge \Theta$, allowing for entering into the second block of lines 10-18.

The second block, lines 10-18, is responsible for advancing the optimization search. First, the new current best NIRA model μ for $\varphi \wedge \bigwedge \Theta$ is stored into $\mathcal{M}$ (line 11). Then the algorithm finds a new UFLIRA model $\hat{\mu}'$ for $\hat{\varphi} \wedge \bigwedge \Gamma \wedge \bigwedge \Theta$ with the best possible value of obj s.t. $\hat{\mu}'$ assigns the same truth values as μ to the atoms in $\hat{\varphi} \wedge \bigwedge \Gamma \wedge \bigwedge \Theta$ (lines 12-13). (The latter restriction forces the minimization procedure to search for an improving solution in the same region as the non-linear model μ.) To do this, the truth assignment $\hat{\psi}$ induced by μ on the atoms in $\hat{\varphi} \wedge \bigwedge \Gamma \wedge \bigwedge \Theta$ is extracted (line 12), and a standard minimization algorithm for linear arithmetic is invoked on $\hat{\psi}$ to find an UFLIRA model $\hat{\mu}'$ for $\hat{\psi}$ which minimizes obj (line 13).

Notice that, $\hat{\mu}'$ is a model for $\hat{\varphi} \wedge \bigwedge \Gamma \wedge \bigwedge \Theta$, but not necessarily so for $\varphi \wedge \Theta$. Therefore $\hat{\mu}'$ is checked for spuriousness with a call to a function OMT-CHECK-REFINE (see Fig. 5), the OMT counterpart of SMT-CHECK-REFINE (line 14). If the refinement process succeeds, then $\mathcal{M}$ is updated with the new current best NIRA model μ' (lines 15-16). Notice that $\hat{\mu}'[\mathsf{obj}] \leq \mu'[\mathsf{obj}] \leq \mu[\mathsf{obj}]$, because $\hat{\mu}'[\mathsf{obj}]$ is the lower bound for the value of obj in models sharing the same truth assignment $\hat{\psi}$. Otherwise, OMT-CHECK-REFINE returns a set of UFLIRA constraints Γ' ruling out $\hat{\mu}'$ (and other spurious solutions) from the feasible search space.

In either case, the optimization search is advanced by extending the set of optimization constraints Θ with a fresh linear constraint of the form $(\mathsf{obj} < \mathsf{ub})$, where ub is computed by an external call to GET-UPPER-BOUND (lines 17-18). In its simplest form, GET-UPPER-BOUND simply returns the value of obj in the current non-linear optimal model $\mathcal{M}$. (A more sophisticate version will be discussed later.)

[6] We stress the fact that SMT-CHECK-REFINE returning false does not mean that $\hat{\varphi} \wedge \bigwedge \Theta$ is NIRA-unsatisfiable, rather that it failed to prove it satisfiable.

At the end of each iteration of the loop (line 19), the set of linearization constraints Γ is extended with the set of UFLIRA constraints Γ', generated by either SMT-CHECK-REFINE (line 9) or OMT-CHECK-REFINE (line 14), so as to permanently rule out spurious solutions that have already been encountered.

The control-flow reaches lines 20-23 only after breaking the loop at line 8. At this point the algorithm returns SAT and the latest non-linear optimal model $\mathcal{M}$ if available, and UNSAT plus an empty model otherwise.

OMT-CHECK-REFINE in Fig. 5 is the OMT counterpart of SMT-CHECK-REFINE in Fig. 2. The only difference between these two functions is that OMT-CHECK-REFINE tries also to improve as much as possible the value of obj while refining the input model. This happens in line 3, where an UFLIRA OMT call is performed instead of an SMT one, so that the resulting model (if any) is the best possible among those allowed by the multiplication line constraint in (2).

Progress. The progress towards an optimum solution within the same truth assignment $\hat{\psi}$ is achieved in two distinct steps.

First, in Fig. 4 line 13, UFLIRA-MINIMIZE searches for the best abstract model $\hat{\mu}'$ which is compatible with the current truth assignment $\hat{\psi}$, so that to search for a refined model μ' starting from a point $\hat{\mu}'$ which is positioned in the direction indicated by obj.

Second, in Fig. 5 line 3, OMT-UFLIRA-CHECK-MINIMIZE finds the best possible among the possible refinements of $\hat{\psi}$ allowed by the multiplication line constraint in (2).

Termination. We notice that the algorithm in Fig. 4 is not guaranteed to terminate, even when the objective function is lower-bounded.

First, the SMT(NIRA) decision procedure based on incremental linearization is incomplete, as described in [13]. Therefore, it is possible for the algorithm to get indefinitely stuck in the main loop enumerating one spurious solution after another.[7]

Second, whereas an OMT(NIRA) problem may admit irrational minimum values for obj, the algorithm in Fig. 4 can return only rational values because it is based only on UFLIRA SMT/OMT calls, so that it may produce an infinite sequence of rational solutions progressively approaching the irrational minimum one.

Third, the linear search strategy in the algorithm of Fig. 4 is not guaranteed to converge towards the optimum value of obj in φ. In fact, each linear step may improve the value of the objective function by a negligible amount only (in particular when working on a continuous domain), even maintaining the same truth assignment $\hat{\psi}$, i.e., without toggling the truth values of the atoms of $\hat{\varphi}$ (and hence of φ) induced by the current models. As a result, the algorithm may end up enumerating an infinite sequence of improving solutions within the same propositional branch of the search.

The latter fact deserves some more explanation. Let $\Delta \stackrel{\text{def}}{=} \mathcal{M}[\mathsf{obj}] - \hat{\mu}'[\mathsf{obj}]$ be the difference between the optimization search upper bound ub computed at line 17 by the basic implementation of GET-UPPER-BOUND ($\mathcal{M}[\mathsf{obj}]$, i.e. either $\mu'[\mathsf{obj}]$ or $\mu[\mathsf{obj}]$), and the UFLIRA-optimum $\hat{\mu}'[\mathsf{obj}]$ computed by OMT-UFLIRA-MINIMIZE at line 13. Then $\Delta \geq 0$ because $\hat{\mu}'[\mathsf{obj}] \leq \mu'[\mathsf{obj}] \leq \mu[\mathsf{obj}]$. When $\Delta = 0$, the unit clause

[7] For the sake of simplicity, here we do not take into consideration the special termination condition based on a finite budget described in [13].

$(\mathsf{obj} < \mathsf{ub})$ learned at line 18 forces the change to a new propositional branch $\hat{\psi}$ of the search because $\hat{\psi} \wedge (\mathsf{obj} < \mathsf{ub})$ is LIRA-inconsistent, and moves the optimization search to explore a new region of the search space. Instead, when $\Delta > 0$ this is not the case, so that the OMT solver may keep looking for an improving NIRA solution in the same region.

Early termination. We remark that, when forced to terminate by external events (e.g., a timeout), our procedure can return the current best result as a partial-optimum solution.

3.2 Algorithm Improvements

Incrementality of SMT(UFLIRA) and OMT(UFLIRA) calls. In the algorithms of Fig. 4 and Fig. 5, all the calls to SMT(UFLIRA) and OMT(UFLIRA) (functions SMT-UFLIRA-CHECK and OMT-UFLIRA-CHECK-MINIMIZE) can be performed incrementally (in our implementation, by exploiting the incremental interface of OPTIMATHSAT), so that to avoid exploring the same portions of the search space multiple times.

Linear-Search Strengthening. In order to increase the pruning power of the linear search constraints learned during the optimization search, we modify the behavior of the function GET-UPPER-BOUND called at line 17 in Fig. 4 as follows. First, we compute

$$\gamma \stackrel{\text{def}}{=} \frac{|\hat{\mu}'[\mathsf{obj}] - \mathcal{M}[\mathsf{obj}]|}{\max\{|\hat{\mu}'[\mathsf{obj}]|, |\mathcal{M}[\mathsf{obj}]|\}}, \tag{3}$$

where $\mathcal{M}$ is the current best NIRA model for $\varphi \wedge \bigwedge \Theta$ and $\hat{\mu}'$ is the UFLIRA-optimal model for $\hat{\varphi} \wedge \bigwedge \Gamma \wedge \bigwedge \Theta$. (We recall that $\hat{\mu}'[\mathsf{obj}]$ is the lower bound for the value of obj in models sharing the same truth assignment $\hat{\psi}$.)

If the value of γ is greater than the input threshold precision value ϵ, then GET-UPPER-BOUND returns $\mathcal{M}[\mathsf{obj}]$ as above. Otherwise, we are not interested in further improving the current best solution in the interval $[\hat{\mu}'[\mathsf{obj}], \mathcal{M}[\mathsf{obj}])$. Therefore, GET-UPPER-BOUND returns $\hat{\mu}'[\mathsf{obj}]$ instead of $\mathcal{M}[\mathsf{obj}]$ as the new upper bound ub, which is such that $\hat{\psi} \wedge (\mathsf{obj} < \mathsf{ub})$ is UFLIRA-inconsistent. Thus in the next loop the procedure is forced to search for models in a new propositional branch.

Henceforth, $\mathcal{M}$ is considered the current best model unless/until some new model $\mathcal{M}'$ is found s.t. $\mathcal{M}'[\mathsf{obj}] < \hat{\mu}'[\mathsf{obj}]$. If this is the case, then the search proceeds. Otherwise, if the procedure concludes that no such model exists (line 8) then $\mathcal{M}$ is returned as best model, within the relative error margin of γ (3).

Notice that, in this case, it is also possible to further refine the search by setting $\Theta := \Theta \setminus \{(\mathsf{obj} < \hat{\mu}'[\mathsf{obj}])\} \cup \{\neg(\mathsf{obj} < \hat{\mu}'[\mathsf{obj}]), (\mathsf{obj} < \mathcal{M}[\mathsf{obj}])\}$, and then proceed the search for a better solution within the interval $[\hat{\mu}'[\mathsf{obj}], \mathcal{M}[\mathsf{obj}])$, either by linear or binary search.

Binary Search. We use the binary search strategy to gain some control on the amount of progress that is made at each step of the optimization search. The ComputePivot() procedure is displayed in Fig. 6.

function COMPUTEPIVOT(obj, lb, ub, $\hat{\mu}'$, $\mathcal{M}$)

1: **if** lb $\neq -\infty$ **then**
2: **return** $^{(\mathsf{lb}+\mathsf{ub})}/_2$
3: **else if** $\hat{\mu}'[\mathsf{obj}] \neq -\infty$ **then**
4: **return** $\hat{\mu}'[\mathsf{obj}]$
5: **else if** $0 < \mathcal{M}[\mathsf{obj}]$ **then**
6: **return** 0
7: **else**
8: **return** $2 \cdot \mathcal{M}[\mathsf{obj}]$

Fig. 6. The COMPUTEPIVOT function for OMT(NIRA) in OPTIMATHSAT.

If a lower bound lb $\neq -\infty$ is available, then $^{(\mathsf{lb}+\mathsf{ub})}/_2$ is returned (lines 1-2). The main difference with the binary search strategy described in §2.1 is in how an initial pivot is computed when no lower bound lb is available (lines 3-8). In this case, the procedure determines the best pivoting value heuristically:

- if $\hat{\mu}'[\mathsf{obj}] \neq -\infty$, then its value is returned as the new pivot (lines 3-4). Intuitively, this results in a pivoting constraint that is just strong enough to force the OMT solver to look for a model of $\hat{\varphi} \wedge \bigwedge \Gamma \wedge \bigwedge \Theta$ on a different propositional branch;
- otherwise, if the current interval contains 0, then a pivoting step on 0 is forced (lines 5-6). The idea is to discover as quickly as possible the sign of the optimum solution;
- if none of the above apply, so that the actual optimal value of obj can be anywhere within the negative interval $(-\infty, \mathcal{M}[\mathsf{obj}]]$, then the double of the (negative) value of $\mathcal{M}[\mathsf{obj}]$ is returned (lines 7-8). Doubling the negative value at each iteration ensures that the search proceeds as quickly as possible to the discovery of an initial lower bound, or to a concrete solution as close as possible to $-\infty$ if no such lower bound exists.

4 Experimental Evaluation

Tools under test. We have implemented the novel OMT(NIRA) algorithm described in §3 within the OMT solver OPTIMATHSAT [37]. We have experimentally validated our algorithm with an analysis of the performance of OPTIMATHSAT in dealing with OMT(NRA) and OMT(NIA) problems, and compared these results with the other available OMT solver, Z3 4.8.10 [8]. We did not compare against DREAL [21] because it supports neither OMT(NIA) problems, nor our generated OMT(NRA) problems coming from planning applications. We did not modify default options of any of the tools, besides using the option `smt.arith.solver=2` for Z3.[8] We have tested OPTIMATHSAT with two configurations: OPTIMATHSAT(LIN), using linear search for optimization, and OPTIMATHSAT(BIN), using binary search. For OPTIMATHSAT we have set the relative-error value ϵ (see §3) to its default value $\epsilon \stackrel{\text{def}}{=} 10^{-6}$.

[8] This option has the effect of enabling the legacy arithmetic solver. Without this option, Z3 produced a significant number of incorrect results. The issues are being reported to the Z3 developers.

We have run the experimental evaluation on a cluster consisting of 20 identical machines. Each of the machines is equipped with Intel Xeon CPU E5-2440 0 2.40GHz CPU and 96 GB of RAM. The time limit for finding the optimal solution was set to 300 seconds for each job pair. Memory limit per job pair was set to 8 GiB.

The benchmark-sets, the results and the scripts necessary to reproduce the experiment are made publicly available and can be downloaded from [1].

Benchmark sets. We have automatically generated OMT(NRA) benchmark set using the tool OMTPLAN [27] extended to dump OMT(NRA) problems in files to enable experimenting with different solvers. [9] OMTPLAN is an AI Planner, which uses Z3 as backend OMT engine, that searches for a sequence of actions of increasing length from one initial completely-specified condition to a goal state, so that a given cost function is minimized. The tool encodes the search for a given plan length leveraging on the Z3 Python API to build and solve the OMT problem. To generate the OMT(NRA) problems, we took existing planning problems from the AI planning literature, and we adapted them to include non-linear constraints in action preconditions, effects and cost functions. We then ran the tool to generate OMT problem files corresponding to problem encodings of increasing length (namely 10,15,20,25,35,50,75,100). This yielded 752 OMT(NRA) benchmarks that were used for the evaluation.

To evaluate the proposed approach also on OMT(NIA), we have automatically generated OMT(NIA) benchmark sets by starting from the SMT problems contained in the SMT-LIBV2 repository [4]. First, we selected 6680 SMT(NIA) instances that are marked as satisfiable in the repository. In order to transform a SMT instance into an OMT problem, we randomly select an arithmetic variable and use it as the objective of the optimization search. We have repeated this step up to 5 times for each instance, depending on the available number of variables, and generated 33397 OMT(NIA) problems. The vast majority – 92.3% – of the generated OMT(NIA) benchmarks comes from the benchmark family *VeryMax*. Therefore, to keep the evaluation time reasonable, we randomly selected 10% of the benchmarks from this family and evaluated the tools only on this subset. We used all the benchmarks from the other families. In total, we thus evaluated the tools on 5744 OMT(NIA) benchmarks.

Verification of results. We have independently verified the correctness of the optimal results found in this experiment using a portfolio of three SMT solvers: CVC4 [38], MATHSAT [14] and Z3 [17]. OPTIMATHSAT and Z3 produce in output a minimum-cost value $\min$. [10] Thus, in order to verify the correctness of such solution, we act as follows. For every OMT problem $\langle \varphi, \mathsf{obj} \rangle$ s.t. the solver terminated and returned a model $\mathcal{M}$ with minimum value $\min \stackrel{\text{def}}{=} \mathcal{M}[\mathsf{obj}]$, we verify (i) that there exists indeed a solution of $\mathsf{obj} = \min$ by checking if $\varphi \wedge (\mathsf{obj} = \min)$ is NIRA-satisfiable and (ii) that $\min$ is a minimum solution by checking that $\varphi \wedge (\mathsf{obj} < \min)$ is NIRA-unsatisfiable. In some cases, OPTIMATHSAT and Z3 can also decide that the optimization problem is unbounded, i.e., has models with arbitrarily big negative $\mathcal{M}[\mathsf{obj}]$. In these cases, we

[9] The modified version of OMTPLAN, together with the planning domain and problems used to generate this benchmark set, is available at [2].

[10] Here we describe the case for minimization; the case for maximization is dual.

checked whether the formula $\varphi \wedge (\mathsf{obj} < -10^6)$ is NIRA-satisfiable. Note that this is not sufficient to prove that the problem indeed is unbounded; this would require quantified reasoning. On the OMT(NRA) formulas, the solvers can also decide that the optimum is infinitesimally close to a given real number, i.e., return an optimum $k + \varepsilon$. For these cases, we checked that (i) the formula $\varphi \wedge (\mathsf{obj} > k) \wedge (\mathsf{obj} < k + 10^{-6})$ is NIRA-satisfiable and that (ii) the formula $\varphi \wedge (\mathsf{obj} \leq k)$ is NIRA-unsatisfiable. Note that this would also require quantified reasoning to confirm the real optimum. However, if the result passes these checks, we consider it as *verified*.

During the verification, we imposed a timeout of 1200 seconds on the portfolio's execution for each problem, i.e., 600 seconds for checking (i) and 600 seconds for checking (ii). To get more independent verification results, we did not stop the portfolio after the first obtained result and let all the solvers finish. We did not observe any incorrect results; unverified results are discussed in the presentation of the results.

Result Tables and Scatter-plots. The results of the experimental evaluation are summarized in the two tables in Figures 7-8. The columns list the total number of instances (col. *instances*), the number of timeouts (col. *timeout*), the number of timeouts after which the solver was able to provide a partial minimum (col. *partial*), the number of formulas decided as unsatisfiable, if any (col. *unsat.*), the number of formulas with an unverified optimal result (col. *unverif.*), the number of formulas with a verified incorrect optimal result (col. *incor.*), the number of formulas with a verified correct optimal result (col. *correct*), the total solving time including all formulas solved correctly (col. *time (s.)*) and the number of formulas that were uniquely solved by the given solver configuration (col *unique s.*). (The *unique s.* column for the V. BEST(OPTIMATHSAT) configuration reports the number of formulas that were solved only by one or both of OPTIMATHSAT versions.)

In the scatter-plots (Fig. 9), we compare the size of the partial minima found by OPTIMATHSAT(LIN) and OPTIMATHSAT(BIN) vs. those of Z3, when the solvers were able to provide at least a partial minimum after the timeout, on OMTPlan problems (1st row) and OMT(NIA) problems (2nd row) respectively. The plots also include results where one of the solvers reported the minimum but the other only finished with a partial minimum. As OMT(NIA) problems are not bounded from below, their partial minima can happen to be arbitrarily big negative numbers. We thus show partial minima smaller than -10^4 on the very left and very bottom lines marked as < -10000. Because OMT(NIA) problems are also discrete, we apply a small random jitter to their results, to better show the number of benchmarks with identical results, which would otherwise overlap.

OMTPlan Results. Fig. 7 presents the results for the OMTPlan benchmark set. We note that the generated problems are very difficult for all the solvers; only 17 benchmarks were solved by any of the solvers. On the whole, OPTIMATHSAT solved the largest number of benchmarks within the timeout. On the other hand, there is one benchmark[11] that was correctly solved by Z3 but not by any of the OPTIMATHSAT configurations.

While OPTIMATHSAT(LIN) solves one more instance than OPTIMATHSAT(BIN), our verification portfolio solver was not able to verify the correctness of this result.

[11] `nl_counters_simple/fn-counters-simp__instance_2_75.smt2`

tool & configuration	instances	timeout	partial	terminated				time (s.)	unique s.
				unsat	unverif.	incor.	correct		
OPTIMATHSAT(LIN)	752	360	231	144	1	0	**16**	887	0
OPTIMATHSAT(BIN)	752	360	236	140	0	0	**16**	827	0
Z3	752	266	312	161	6	0	7	315	1
V. BEST(OPTIMATHSAT)	752	356	235	144	1	0	16	810	10
V. BEST(ALL)	752	260	318	161	6	0	17	899	-

Fig. 7. Experimental results over the OMTPLAN benchmark-set.

tool & configuration	instances	timeout	partial	terminated				time (s.)	unique s.
				unverif.	incor.	correct			
OPTIMATHSAT(LIN)	5744	1449	1019	14	0	**3262**		97011	85
OPTIMATHSAT(BIN)	5744	1433	1047	11	0	3247		96329	72
Z3	5744	2664	1105	18	0	1957		55319	127
V. BEST(OPTIMATHSAT)	5744	1415	972	15	0	3342		104454	1512
V. BEST(ALL)	5744	1130	1122	23	0	3469		95367	-

Fig. 8. Experimental results over the OMT(NIA) benchmark-set.

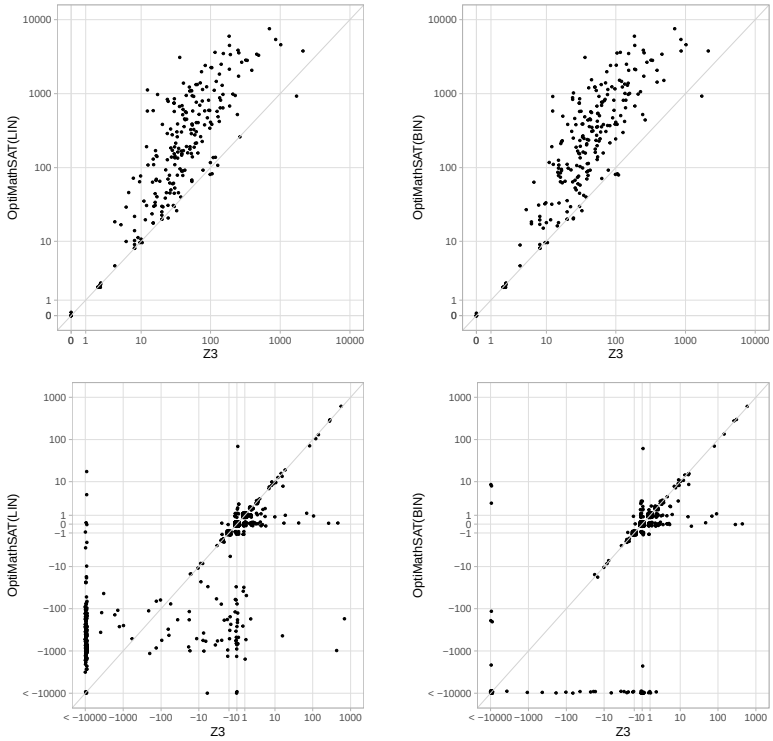

Fig. 9. Size of the partial minima found by OPTIMATHSAT(LIN) (left) and OPTIMATHSAT(BIN) (right) on Y axis vs. those of Z3 on X axis.
First row shows OMTPlan problems, second row shows OMT(NIA) problems.

More generally, 16 out of 17 results of OPTIMATHSAT(LIN) were verified (7 by one solver, 2 by two solvers, and 7 by all three solvers); all 16 results of OPTIMATHSAT(BIN) were verified (7 by one solver, 2 by two solvers, and 7 by all three solvers); and 7 out of 13 results of Z3 were verified (all by all three solvers). The remaining 6 results of Z3 on which the verification did not finish were of form $k + \varepsilon$.

Note that by the nature of how the benchmarks were generated, some of them are not satisfiable. Although the numbers of unsatisfiable benchmarks are reported in the table, they are not relevant to the evaluation, as they depend only on the performance of the base SMT(NRA) solver and do not compare capabilities of its OMT(NRA) extension.

Looking at the scatterplots (Fig 9) we notice that, whereas OPTIMATHSAT finds more minima, Z3 partial minima within the timeout are generally much better than those of OPTIMATHSAT. We do not have a clear-cut explanation of this apparently-contradictory fact. We conjecture that this is due to the fact that, unlike Z3, OPTIMATHSAT does not have a NRA-mimimization procedure working on single assignments, which typically get near to local solutions quickly.

OMT(NIA) Results. Fig. 8 presents the results for the OMT(NIA) benchmark set. In general, almost all the results produced by the solvers were verified to be correct. For OPTIMATHSAT(LIN), 3262 of total 3276 results were verified (128 by one of the solvers, 1076 by two of the solvers, and 2058 by all three); for OPTIMATHSAT(BIN), 3247 of total 3258 results were verified (124 by one of the solvers, 1078 by two of the solvers, and 2045 by all three); for Z3, 1957 of total 1975 results were verified (28 by one of the solvers, 518 by two of the solvers, and 1411 by all three).

On the whole, OPTIMATHSAT solved the largest number of benchmarks within the timeout. We note that there is not a significant difference between the performance of OPTIMATHSAT(LIN) and OPTIMATHSAT(BIN).

Looking at the scatterplots (Fig 9) we notice that, unlike with the OMTPlan, there is no tool whose partial minima are definitely better than the others.

5 Conclusions and Future Work

In this paper, we have shown how incremental linearization can be extended from SMT(NIRA) to OMT(NIRA) in a very simple way, producing an incomplete though effective OMT procedure. We believe that this procedure, in its simplicity, can also be used as a baseline for more elaborated procedures.

To this extent, we believe that many possible extension are possible. In a short term, we plan to extend it to work also with transcendental functions, exploiting the full expressive power of incremental-linearization approach in SMT presented in [13], and then to test the effectiveness of the procedures on real word verification problems, in particular of cyber-physical systems. In the middle term, we plan to extend our encoders from MINIZINC to OMT and vice-versa [16] to work with non-linear constraints, so that to be able to compare with tools and problems coming from the Constraint Solving & Optimization community. In a longer term, we plan to integrate our approach with more elaborated –though possibly expensive– procedures.

References

1. https://es-static.fbk.eu/people/mjonas/papers/frocos21_oms/.
2. https://github.com/roveri-marco/OMTNPlan.
3. E. Ábrahám, J. H. Davenport, M. England, and G. Kremer. Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings. *J. Log. Algebraic Methods Program.*, 119:100633, 2021.
4. C. Barrett, S. Ranise, A. Stump, and C. Tinelli. The satisfiability modulo theories library (smt-lib). http://www.smtlib.org, 2010.
5. C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. *Satisfiability Modulo Theories*, chapter 26, pages 825–885. Volume 185 of Biere et al. [7], February 2009.
6. F. Benhamou and L. Granvilliers. Continuous and interval constraints. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 571–603. Elsevier, 2006.
7. A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*. IOS Press, February 2009.
8. N. Bjorner, A.-D. Phan, and L. Fleckenstein. νZ - An Optimizing SMT Solver. In *Proc. TACAS*, volume 9035 of *LNCS*. Springer, 2015.
9. C. Borralleras, D. Larraz, E. Rodríguez-Carbonell, A. Oliveras, and A. Rubio. Incomplete SMT techniques for solving non-linear formulas over the integers. *ACM Trans. Comput. Log.*, 20(4):25:1–25:36, 2019.
10. C. Borralleras, S. Lucas, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. SAT modulo linear arithmetic for solving polynomial constraints. *J. Autom. Reasoning*, 48(1):107–131, 2012.
11. F. Brauße, K. Korovin, M. V. Korovina, and N. T. Müller. A cdcl-style calculus for solving non-linear constraints. In *FroCos*, volume 11715 of *Lecture Notes in Computer Science*, pages 131–148. Springer, 2019.
12. A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani. Experimenting on solving nonlinear integer arithmetic with incremental linearization. In *SAT*, volume 10929 of *Lecture Notes in Computer Science*, pages 383–398. Springer, 2018.
13. A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani. Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions. *ACM Trans. Comput. Log.*, 19(3):19:1–19:52, 2018.
14. A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. The MathSAT 5 SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'13.*, volume 7795 of *LNCS*, pages 95–109. Springer, 2013.
15. G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition-preliminary report. *ACM SIGSAM Bulletin*, 8(3):80–90, 1974.
16. F. Contaldo, P. Trentin, and R. Sebastiani. From minizinc to optimization modulo theories, and back. In *Proc. Seventeenth International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research, CPAIOR*, 2020.
17. L. M. de Moura and N. Bjørner. Z3: An efficient smt solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
18. C. Dixon and M. Finger, editors. *Frontiers of Combining Systems - 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings*, volume 10483 of *Lecture Notes in Computer Science*. Springer, 2017.
19. P. Fontaine, M. Ogawa, T. Sturm, and X. Vu. Subtropical satisfiability. In Dixon and Finger [18], pages 189–206.
20. C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, and H. Zankl. SAT solving for termination analysis with polynomial interpretations. In J. Marques-Silva and

K. A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, volume 4501 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 2007.
21. S. Gao, S. Kong, and E. M. Clarke. dreal: An SMT solver for nonlinear theories over the reals. In *CADE*, volume 7898 of *Lecture Notes in Computer Science*, pages 208–214. Springer, 2013.
22. D. Jovanovic. Solving nonlinear integer arithmetic with MCSAT. In A. Bouajjani and D. Monniaux, editors, *Verification, Model Checking, and Abstract Interpretation - 18th International Conference, VMCAI 2017, Paris, France, January 15-17, 2017, Proceedings*, volume 10145 of *Lecture Notes in Computer Science*, pages 330–346. Springer, 2017.
23. D. Jovanovic and L. de Moura. Solving non-linear arithmetic. *ACM Comm. Computer Algebra*, 46(3/4):104–105, 2012.
24. S. Kong, A. Solar-Lezama, and S. Gao. Delta-decision procedures for exists-forall problems over the reals. In H. Chockler and G. Weissenbacher, editors, *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, volume 10982 of *Lecture Notes in Computer Science*, pages 219–235. Springer, 2018.
25. G. Kremer and E. Ábrahám. Fully incremental cylindrical algebraic decomposition. *J. Symb. Comput.*, 100:11–37, 2020.
26. G. Kremer, F. Corzilius, and E. Ábrahám. A generalised branch-and-bound approach and its application in SAT modulo nonlinear integer arithmetic. In V. P. Gerdt, W. Koepf, W. M. Seiler, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing - 18th International Workshop, CASC 2016, Bucharest, Romania, September 19-23, 2016, Proceedings*, volume 9890 of *Lecture Notes in Computer Science*, pages 315–335. Springer, 2016.
27. F. Leofante, E. Giunchiglia, E. Ábrahám, and A. Tacchella. Optimal planning modulo theories. In C. Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4128–4134. ijcai.org, 2020.
28. A. Maréchal, A. Fouilhé, T. King, D. Monniaux, and M. Périn. Polyhedral approximation of multivariate polynomials using handelman's theorem. In B. Jobstmann and K. R. M. Leino, editors, *Verification, Model Checking, and Abstract Interpretation - 17th International Conference, VMCAI 2016, St. Petersburg, FL, USA, January 17-19, 2016. Proceedings*, volume 9583 of *Lecture Notes in Computer Science*, pages 166–184. Springer, 2016.
29. J. P. Marques-Silva, I. Lynce, and S. Malik. *Conflict-Driven Clause Learning SAT Solvers*, chapter 4, pages 131–153. Volume 185 of Biere et al. [7], February 2009.
30. Y. V. Matiyasevich. *Hilbert's Tenth Problem*. Foundations of computing. MIT Press, 1993.
31. A. Nadel and V. Ryvchin. Bit-Vector Optimization. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2016*, volume 9636 of *LNCS*. Springer, 2016.
32. R. Nieuwenhuis and A. Oliveras. On SAT Modulo Theories and Optimization Problems. In *Proc. Theory and Applications of Satisfiability Testing - SAT 2006*, volume 4121 of *LNCS*. Springer, 2006.
33. A. Reynolds, C. Tinelli, D. Jovanovic, and C. W. Barrett. Designing theory solvers with extensions. In Dixon and Finger [18], pages 22–40.
34. R. Sebastiani and S. Tomasi. Optimization in SMT with LA(Q) Cost Functions. In *IJCAR*, volume 7364 of *LNAI*, pages 484–498. Springer, July 2012.
35. R. Sebastiani and S. Tomasi. Optimization Modulo Theories with Linear Rational Costs. *ACM Transactions on Computational Logics*, 16(2), March 2015.
36. R. Sebastiani and P. Trentin. Pushing the Envelope of Optimization Modulo Theories with Linear-Arithmetic Cost Functions. In *Proc. Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'15*, volume 9035 of *LNCS*. Springer, 2015.
37. R. Sebastiani and P. Trentin. OptiMathSAT: A Tool for Optimization Modulo Theories. *Journal of Automated Reasoning*, 64:423–460, 2020.

38. A. Stump, C. W. Barrett, and D. L. Dill. CVC: A Cooperating Validity Checker. In *Proc. CAV'02*, number 2404 in LNCS. Springer Verlag, 2002.
39. P. Trentin and R. Sebastiani. Optimization modulo the theory of floating-point numbers. In *In proc. The 27th International Conference on Automated Deduction - CADE-27*, LNAI/LNCS. Springer, 2019.
40. V. Weispfenning. Quantifier elimination for real algebra - the quadratic case and beyond. *Appl. Algebra Eng. Commun. Comput.*, 8(2):85–101, 1997.
41. H. Zankl and A. Middeldorp. Satisfiability of non-linear (ir)rational arithmetic. In E. M. Clarke and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010, Revised Selected Papers*, volume 6355 of *Lecture Notes in Computer Science*, pages 481–500. Springer, 2010.