Optimal Planning Modulo Theories

Francesco Leofante[1] , **Enrico Giunchiglia**[2] , **Erika Ábrahám**[3] and **Armando Tacchella**[2]
[1]Imperial College London, United Kingdom
[2]University of Genoa, Italy
[3]RWTH Aachen University, Germany
f.leofante@imperial.ac.uk

Abstract

We consider the problem of planning with arithmetic theories, and focus on generating optimal plans for numeric domains with constant and state-dependent action costs. Solving these problems efficiently requires a seamless integration between propositional and numeric reasoning. We propose a novel approach that leverages Optimization Modulo Theories (OMT) solvers to implement a domain-independent optimal theory-planner. We present a new encoding for optimal planning in this setting and we evaluate our approach using well-known, as well as new, numeric benchmarks.

1 Introduction

In this work we focus on *numeric planning* [Fox and Long, 2003], an extension of classical planning where preconditions and effects of actions may require reasoning on fragments of arithmetic theories. Despite being undecidable in the general case [Helmert, 2002], admissible heuristics have been extended to handle *simple* numeric planning problems [Scala *et al.*, 2016a] in which actions have linear conditions and may only increase or decrease numeric variables by a constant – see, e.g, [Scala *et al.*, 2016a; Scala *et al.*, 2017; Piacentini *et al.*, 2018b]. Several heuristics have been proposed also for numeric planning problems where both conditions and effects are expressed as *linear* expressions over numeric state variables [Hoffmann, 2003; Illanes and McIlraith, 2017; Li *et al.*, 2018]. However these are inadmissible and cannot be used in the cost-optimal setting. Notably, cost-optimal planning with both simple and linear effects can be handled by [Piacentini *et al.*, 2018a] via a compilation to mixed-integer linear programming (MILP) that proved to be competitive with heuristic search approaches.

We propose a new planning approach that can solve to optimality numeric problems for which (integrated) algorithmic solutions have not been proposed yet. We focus on extending cost-optimal numeric planning to problems where conditions may be simple or linear and actions are equipped with constant and *state-dependent* costs, i.e., costs are encoded by arithmetic expressions over numeric state variables. Previous works have studied state-dependent action costs (SDAC) [Ivankovic *et al.*, 2014] in the classical setting [Geißer *et al.*, 2015; Geißer *et al.*, 2016] and in the presence of global numerical state constraints [Ivankovic *et al.*, 2014; Haslum *et al.*, 2018; Ivankovic *et al.*, 2019], but did not explore extensions towards numeric planning.

Support and scalability are challenging in the setting we target, but we show that such challenges are met by our approach leveraging recent advances in Optimization Modulo Theories (OMT) [Sebastiani and Tomasi, 2015], an extension of SMT that combines efficient propositional reasoning with dedicated procedures for theory-optimization. After a brief review of background notions in Section 2, Sections 3 and 4 detail our contributions summarized as follows:

- A novel SMT encoding of numeric planning that enables an efficient relaxed reachability analysis. We extend standard encodings with a suffix that performs a Boolean abstraction of the transition function of the planning problem. Reasoning on this abstraction, we can conclude whether a goal is reachable only with a modest increase in the size of the formula and without resorting to expensive decision procedures for theory-reasoning.
- An extension of this construction that leads to Optimal Planning Modulo Theories. We show how our encoding needs to be modified in the OMT setting and provide optimality guarantees for our approach. While this is not the first application of OMT to planning – see, e.g., [Leofante *et al.*, 2019; Leofante *et al.*, 2018] – this work presents the first *domain-independent* results based on OMT.
- An empirical analysis of our planner on domains previously reported in the literature [Scala *et al.*, 2017; Li *et al.*, 2018], and on a new numeric domain featuring SDAC which state-of-the-art tools struggle to solve.

2 Background

Numeric planning. We consider a fragment of numeric planning expressible in PDDL2.1 level 2 [Fox and Long, 2003]. A (*numeric*) *planning problem* is a tuple $\Pi = \langle \mathcal{V}_{\mathbb{B}}, \mathcal{V}_{\mathbb{Q}}, A, I, G \rangle$ where $\mathcal{V}_{\mathbb{B}}$ and $\mathcal{V}_{\mathbb{Q}}$ are finite disjoint sets of *propositional* and *numeric variables* of Π, respectively; for a variable $v \in \mathcal{V}_{\mathbb{B}} \cup \mathcal{V}_{\mathbb{Q}}$ let $dom(v)$ denote the *domain of* v; we use Boolean $\mathbb{B}$ for propositional variables and for numeric variables the rationals $\mathbb{Q}$ equipped with the usual

order and arithmetic operations. A *state* of Π is a function $s : (\mathcal{V}_\mathbb{B} \cup \mathcal{V}_\mathbb{Q}) \to (\mathbb{B} \cup \mathbb{Q})$ assigning to each variable $v \in \mathcal{V}_\mathbb{B} \cup \mathcal{V}_\mathbb{Q}$ a value $s(v) \in \mathit{dom}(v)$ from its corresponding domain. Let S denote the set of all states of Π. A *propositional constraint* is either a propositional variable $v \in \mathcal{V}_\mathbb{B}$ or its negation $\neg v$. *Arithmetic expressions* e are composed from numeric variables and constants using arithmetic operators. An arithmetic expression is *linear* iff for each multiplication operator in it, at least one of the operands contains no variables. A *numeric constraint* $e_1 \sim e_2$ compares two arithmetic expressions using a comparison operator $\sim \in \{<, \leq, =, \geq, >\}$. A *constraint* φ is either a propositional or a numeric constraint. A *condition* Φ is a (possibly empty) set of constraints. The evaluation function $[\![\cdot]\!]_s$ and the satisfaction relation $\models$ for expressions, constraints and conditions are as usual.

A *propositional assignment* has the form $v := e$, where $v \in \mathcal{V}_\mathbb{B}$ is a propositional variable and $e \in \{\top, \bot\}$. A *numeric assignment* has the form $v := e$, where $v \in \mathcal{V}_\mathbb{Q}$ is a numeric variable and e is an arithmetic expression; we say that $v := e$ is an assignment *to* v. An *assignment* is either a propositional or a numeric assignment. An *effect* Ψ is a set of assignments that contains at most one assignment $v := e$ for each variable $v \in \mathcal{V}_\mathbb{B} \cup \mathcal{V}_\mathbb{Q}$; we say that v is *assigned* in Ψ iff there is an assignment $v := e$ in Ψ. For any $v \in \mathcal{V}_\mathbb{Q}$, $d \in \mathit{dom}(v)$, and e a linear arithmetic expression, we call $v := v{+}d$ a *constant increment*, $v := v{-}d$ a *constant decrement*, $v := v{+}e$ a *linear increment*, and $v := v{-}e$ a *linear decrement*. Given a state $s \in S$ and an effect Ψ, the *successor* of s and Ψ is the (unique) state $s' \in S$ such that $s'(v) = [\![e]\!]_s$ for each $v := e$ in Ψ, and $s'(v) = s(v)$ for each $v \in \mathcal{V}_\mathbb{B} \cup \mathcal{V}_\mathbb{Q}$ that is not assigned in Ψ; we write $s, s' \models \Psi$.

A is the set of *actions* $a = (\mathit{pre}_a, \mathit{eff}_a, c_a)$, where pre_a is a condition, eff_a is an effect and $c_a : S \to \mathbb{Q}_{\geq 1}$ is a state-dependent positive *cost function*, specified by a numeric expression. An action $a = (\mathit{pre}_a, \mathit{eff}_a, c_a)$ is *applicable* in state s iff (i) $s \models \varphi$ for each $\varphi \in \mathit{pre}_a$ and (ii) $[\![e]\!]_s$ is defined for each assignment $v := e$ in eff_a. A numeric constraint $e \sim 0$ is *simple* iff e is linear and for each assignment in $\cup_{a \in A} \mathit{eff}_a$, either the assigned variable does not appear in e or the assignment is a constant increment or a constant decrement. A numeric constraint $e \sim 0$ is *linear* iff e is linear and for each assignment in $\cup_{a \in A} \mathit{eff}_a$, either the assigned variable does not appear in e or the assignment is a linear increment or a linear decrement. A set $M_A \subseteq A$ of actions is *independent* if any variable assigned in the effect of an action in the set appears in no other action in the set (neither in conditions nor in effects nor in cost functions). I is a condition called the *initial condition* which is satisfied by exactly one state called the *initial state*; G is a condition called the *goal condition*; states satisfying G are called *goal states*.

A (sequential) *plan* $\pi_n = \langle a_0, \ldots, a_{n\text{-}1} \rangle$ is a sequence of actions $a_0, \ldots, a_{n\text{-}1} \in A$ such that there exist (unique) states $s_0, \ldots, s_n \in S$ with $s_0 \models I$, $s_{i\text{-}1} \models \mathit{pre}_{a_{i\text{-}1}}$ and $s_{i\text{-}1}, s_i \models \mathit{eff}_{a_{i\text{-}1}}$ for each $i = 1, \ldots, n$, and $s_n \models G$. The *cost* of π_n is $C(\pi_n) = \sum_{i=0}^{n-1} c_{a_i}(s_i)$. A *parallel plan* $\pi_n = \langle A_0, \ldots, A_{n\text{-}1} \rangle$ is a sequence of independent action sets $A_i = \{a_{i,1}, \ldots, a_{i,k_i}\} \subseteq A$, for $i = 0, \ldots, n\text{-}1$, such that $\langle a_{0,1}, \ldots, a_{0,k_0}, \ldots, a_{n\text{-}1,1}, \ldots, a_{n\text{-}1,k_{n\text{-}1}} \rangle$ is a plan for Π. The cost of a parallel plan π_n is $C(\pi_n) = \sum_{i=0}^{n-1} \sum_{a \in A_i} c_{a_i}(s_i)$. A plan π_n is *optimal* for Π iff $C(\pi_n) \leq C(\pi')$ for all plans π' of Π.

Planning as satisfiability. Given a planning problem $\Pi = \langle \mathcal{V}_\mathbb{B}, \mathcal{V}_\mathbb{Q}, A, I, G \rangle$, *planning as satisfiability* [Kautz and Selman, 1992] tries to solve Π by encoding plans of bounded length as the solutions of a logical formula. We consider *state-based* encodings to SAT [Rintanen *et al.*, 2006; Rintanen, 2009] and SMT [Shin and Davis, 2005]. To encode the existence of parallel plans of length up to n, we encode a sequence of n ground action sets as well as the semantics of their parallel execution. This requires n variable sets $A_0, \ldots, A_{n\text{-}1}$, where each A_i consists of a unique proposition a_i for each action $a \in A$ (stating whether or not a is executed in step i), and also $n{+}1$ copies $\mathcal{V}_i = \{v_i \mid v \in \mathcal{V}_\mathbb{B} \cup \mathcal{V}_\mathbb{Q}\}$, $i = 0, \ldots, n$ of the propositional and numeric variable sets, storing the initial variable values in $\mathcal{V}_0$ and the values after the ith step in $\mathcal{V}_i$.

For each constraint φ in the plan specification, we denote by φ_i the formula obtained from φ by replacing each variable $v \in \mathcal{V}$ with $v_i \in \mathcal{V}_i$. The same renaming applies to effects. Thus in formula (1) below, I_0 is the initial condition with each variable v replaced by v_0, and similarly G_n results from the goal condition G after replacing each v with v_n.

Furthermore, let $T_{i,i+1}$ be a formula describing how actions executed in the ith step of a plan affect states, i.e., $T_{i,i+1}$ enforces that each action implies its preconditions over $\mathcal{V}_i$ and its effects over $\mathcal{V}_{i+1}$. With $T_{i,i+1}$ we also encode *explanatory frame axioms* and mutual exclusion (*mutex*) axioms. The former state that variables not affected by actions do not change their values; the latter enforce that multiple actions can be performed simultaneously if and only if they are independent.

For a given planning problem Π, now we can encode bounded *plans* for horizon n with the following formula Π_n:

$$\Pi_n : \quad I_0 \wedge \bigwedge_{i=0}^{n-1} T_{i,i+1} \wedge G_n \tag{1}$$

Example 1. *Assume a planning problem with one proposition $\mathcal{V}_\mathbb{B} = \{b\}$ and one numeric variable $\mathcal{V}_\mathbb{Q} = \{x\}$), initial condition $b \wedge (x = 0)$, goal condition $x = 1$, and two actions $A = \{a^+, a^-\}$, $a^+ = (\emptyset, \{x := x{+}2\}, x)$ and $a^- = (\{b\}, \{x := x{-}1, b := \mathit{false}\}, x)$. Without considering costs, the encoding uses the following constructs:*

$$\begin{aligned}
I_0 \quad &: x_0 := 0 \wedge b_0 \quad G_n : x_n = 1 \\
T^+_{i,i+1} : \; &a^+_i \Rightarrow x_{i+1} := x_i{+}2 \\
T^-_{i,i+1} : \; &a^-_i \Rightarrow (b_i \wedge x_{i+1} := x_i{-}1 \wedge \neg b_{i+1}) \\
T_{i,i+1} : \; &(a^+_i \oplus a^-_i) \wedge T^+_{i,i+1} \wedge T^-_{i,i+1} \wedge (b_i \wedge \neg b_{i+1} \Rightarrow a^-_i) \\
&\wedge (\neg b_i \wedge b_{i+1} \Rightarrow \bot) \wedge (x_i = x_{i+1} \vee a^-_i \vee a^+_i)
\end{aligned}$$

where $\oplus$ denotes mutual exclusion. The formulas Π_0 and Π_1 are unsatisfiable, but Π_2 yields a plan executing both a^-_i and a^+_i once, in arbitrary order.

3 Optimal Planning Modulo Theories

When dealing with unit costs, the standard encoding above can be adapted to find plans with a minimal number of actions by enforcing that exactly one action is executed in each step. If $\Pi_0, \ldots, \Pi_{n-1}$ are unsatisfiable and Π_n has a solution, then the shortest plan has length n. However, for constant and state-dependent costs the shortest plan is not necessarily cost-optimal, therefore new conditions are needed to determine when to stop searching. To put a bound on this cost-optimal search, we need at least a sufficient condition to detect that no plans longer than a certain bound exist. The idea is that if after n steps no future action can modify any of the variables in the goal any more, then the problem does not admit any solution.

3.1 A Novel SMT Encoding

To formalize such a sufficient condition Π_n^{bound} for a given horizon n, we enforce the initial condition and transitions as per formula (1) and extend it with requirements T_n^{abs} and G_n^{abs} as explained below:

$$\Pi_n^{bound}: \; I_0 \wedge \bigwedge_{i=0}^{n-1} T_{i,i+1} \wedge T_n^{abs} \wedge G_n^{abs} \tag{2}$$

Having executed n steps, a necessary condition for satisfying the goal condition G at n or in the future is that for each constraint φ in G, either φ is true in n or at least one variable in φ will be modified by further steps. To express a necessary condition for the latter, we introduce for each variable $v \in \mathcal{V}_{\mathbb{B}} \cup \mathcal{V}_{\mathbb{Q}}$ a fresh proposition *mod_v*, and we ensure that if *mod_v* is false then there exists no plan that is able to modify the value of v after the first n steps:

$$G_n^{abs}: \quad \textstyle\bigwedge_{\varphi \in G} (\varphi_n \vee \bigvee_{v \in \varphi} mod\_v) \tag{3}$$

To achieve such a result, we observe that having executed n steps, a necessary condition for executing an action a in the future is that for each precondition of a either it is true in state n or at least one of its variables will be modified by further executions. Since executing an action might in turn enable further actions, we encode fixed points for this chain to obtain an over-approximation of all variables whose value could still be potentially changed if a longer horizon were given:

$$T_n^{abs}: \bigwedge_{a \in A} \Big(a^{abs} \Rightarrow \Big(\bigwedge_{\varphi \in pre_a} (\varphi_n \vee \bigvee_{v \in \varphi} mod\_v)\Big)\Big) \\ \wedge \bigwedge_{v \in \mathcal{V}} \Big(mod\_v \Leftrightarrow \Big(\bigvee_{a \in A,\, v := e \in eff_a} a^{abs}\Big)\Big) \tag{4}$$

Note that a^{abs} represents only a necessary condition for executing a in the future, on a path starting in the nth state. However, this condition is not sufficient for the existence of plans with length n or longer, because after the first n steps we disregard the concrete effects of actions. Those steps can be seen as executing *abstract actions*, where for each concrete action $a \in A$ we define one abstract action a^{abs} by

- relaxing each constraint $\varphi \in pre_a$ in the precondition of a to $\varphi \vee \bigvee_{v \in \varphi} mod\_v$ and
- relaxing each assignment $v := e$ in the effect of a by $mod\_v := true$,

assuming that *mod_v* are initially false for all $v \in \mathcal{V}$. Note that, since all abstract action effects set a subset of the *mod_v* variables to true, they are not conflicting, therefore we do not encode mutex axioms.

Example 2. *Consider the planning problem of Example 1. Without considering costs, the encoding of* Π_n^{bound} *uses the following new constructs:*

$$G_n^{abs}: (x_n = 1 \vee mod\_x)$$
$$T_n^{abs}: \; a^{-,abs} \Rightarrow (b_n \vee mod\_b) \wedge (mod\_b \Leftrightarrow a^{-,abs}) \\ (mod\_x \Leftrightarrow a^{+,abs} \vee a^{-,abs})$$

Enforcing the smallest fixed point. Our encoding should express the existence of a plan of length n or longer when after the first n steps we switch from executing concrete actions to executing abstract actions. However, there is one remaining problem: formula (4) does not yet enforce the *smallest* fixed point. For example, consider an action a with a single precondition constraint that refers to a variable $v \in \mathcal{V}$ and a single assignment in its effect that assigns v. Then a solution which evaluates both a^{abs} and *mod_v* to true satisfies the corresponding formulas in formula 4, but this way a^{abs} might "enable itself". Though the encoding is already correct in its current form, in order to make it efficient, we compute the *smallest fixed point* of all facts that are true in n under the abstract transition relation to obtain all valid reachable states.

We borrow ideas from Answer Set Programming (ASP) [Gelfond and Lifschitz, 1988] to perform this computation in our framework[1]. Consider a program P corresponding to formula (4) where, for each abstract action a^{abs}, a set of rules is built such that: *(i)* premises contain one of the disjunct appearing in the precondition of a^{abs} and *(ii)* conclusions of each rule contain all *mod_v* appearing in the effects of a^{abs}.

Theorem 1. *(closure) For any fixed truth values of constraints in action preconditions and the goal condition, the smallest closure of the abstract transition relation corresponds to the answer set of* P.

Previous works on compilation from ASP to the satisfiability setting can be leveraged to encode this computation in our construction. Here we use an approach based on *loop formulas* as proposed by [Lin and Zhao, 2002].

We start by constructing a *dependency graph* for abstract actions as a directed graph D such that: *(i.)* the nodes of D correspond to all concrete constraints and all *mod_v*, $v \in \mathcal{V}$ in action preconditions and *(ii.)* for each abstract action a^{abs} and each variable *mod_v* assigned by a^{abs}, there is an edge connecting *mod_v* to nodes corresponding to a^{abs}'s precondition.

[1] Our idea bears some similarities to the relaxed reachability analysis of [Helmert, 2009], although here we propose a new construction that can be embedded in the Planning as SAT framework. Moreover, we use this construction with a different purpose: while Helmert uses this to perform reachability analysis, we use this abstraction to provide termination guarantees for our planning algorithm.

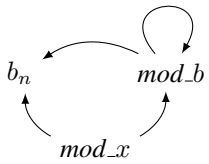
From [Lin and Zhao, 2002] we know that strongly connected components of D represent loops in the formula, i.e., circular dependencies that hinder the computation of the smallest set closed under action rules.

For each loop visiting nodes L, we forbid circular self-enabling of corresponding actions with the formula

$$(\bigvee_{l \in L} l) \Rightarrow (\bigvee_{l' \in R(L)} l') \tag{5}$$

where $R(L)$ is a set of variables appearing in premises of abstract rules such that: *(i.)* at least one variable in their conclusions intersect L but *(ii.)* none of the variables in their premises intersects L.

Example 3. *Consider the planning problem of Example 1, where action a^+ is now removed, i.e., $A = \{a^-\}$. The dependency graph for $a^{-,abs}$ is*



The graph contains one loop $L = \{mod\_b\}$ and the corresponding loop formula writes $mod\_b \Rightarrow b_n$

Correctness. In the following let $A^{abs} = \{a^{abs} \in \mathbb{B} \mid a \in A\}$ be the set of variables modelling (the execution of) abstract actions. Furthermore, let Π_n^{bound+} be the conjunction of Π_n^{bound} and the axioms (5) for each loop in the dependency graph D. We can formulate the following correctness statement for Π_n^{bound+} .

Theorem 2. *(unsolvability) For any $n \geq 0$, if Π_n^{bound+} is unsatisfiable then the planning problem Π does not admit solution.*

Proof sketch. The full proof of Theorem 2 is complex. Due to space constraint we only provide an intuition of the proof. Assume Π_n^{bound+} is unsatisfiable while Π admits a plan $\pi_m = \langle A_0, \ldots, A_{m-1} \rangle$ of length $m \geq n$. By construction the sequence $\langle A_0, \ldots, A_{n\text{-}1} \rangle$ satisfies Π_n. Abstraction-enabling axioms ensure that the abstract counterparts of the actions A_n are still applicable in the nth state and their effects enable the execution of subsequent abstract action sets in place of $\langle A_n, ..., A_{m\text{-}1} \rangle$. Since $A_{m\text{-}1}$ achieves the goal in Π, the abstract goal can be reached in Π_n^{bound+} as well, leading to a contradiction. □

3.2 Extension to OMT

In the first part of this section we introduced a new encoding that enables relaxed reachability analysis. In the following we show how to leverage the properties of this encoding to provide termination conditions and optimality guarantees for our planning algorithm. The main result can be informally stated as follows: if, for a given horizon, the OMT solver finds a solution that does not require abstract actions, this solution is also a globally optimal plan. To achieve this we extend formula Π_n^{bound+} as follows.

First, in the encoding T_n^{abs} of abstract actions a^{abs} we remove the abstraction of effects on cost variables and add instead a metric for plan quality: for each concrete action a with cost function c_a, the abstract action a^{abs} has the minimal value of c_a over all states, i.e. $c_{a^{abs}} := \min_{s \in S} c_a(s)$.

With the extensions above, we ensure that abstract actions always have a cost that is lower than or equal to the one of their concrete counterpart. Under this cost schema, the solver will favour the execution of abstract actions to achieve the (abstract) goal as these have minimum cost. While this is needed to ensure optimality of our solutions as we will show, we must make sure the solver does not abuse this and push the execution of all actions to the suffix. Not only this would affect optimal reasoning, but would also affect termination of bounded planning procedures. Indeed we would need to increase the planning horizon indefinitely hoping to find a valid plan (i.e., containing only concrete actions), but this would never happen. Hence to ensure termination, we augment the OMT encoding with the following axioms. For each action $a \in A$ let $M_a \subseteq A$ be the set of actions that are not independent from a. For each action $a \in A$ and for each step $0 < i < n$ we add

$$a_i \Rightarrow \left[a_{i\text{-}1} \vee \left(\bigvee_{\varphi \in pre_a} \neg \varphi_{i\text{-}1} \right) \vee \left(\bigvee_{a' \in M_a} a'_{i\text{-}1} \right) \right] \tag{6}$$

With axioms (6) an action a can be executed at step i only if: *(i.)* a was already performed at step i-1 or, *(ii.)* a was not applicable at step i-1 or, *(iii.)* another action a', mutex with a, was performed at i-1. We then ensure that abstract actions are executed only if all previous steps (i.e., $< n$) are filled with at least one action:

$$\left(\bigvee_{a^{abs} \in A^{abs}} a^{abs} \right) \Rightarrow \bigwedge_{0 \leq i < n} \bigvee_{a \in A} a_i \tag{7}$$

Let Π_n^{opt} denote the planning formula Π_n^{bound+} extended with the axioms above, the following result holds.

Lemma 1. *For any $n \geq 0$, Π_n^{bound+} and Π_n^{opt} are equisatisfiable.*

With the addition above we can formulate the following theorem.

Theorem 3. *(optimality) For any $n \geq 0$, let μ be the optimal solution of Π_n^{opt}. If $\mu \models G_n$ then μ is a valid* optimal *plan.*

Proof sketch. The intuition behind this proof is based on the fact that the goal state could be reached without resorting to abstract actions. Assume $\mu \models G_n$ but a cheaper plan $\pi_m = \langle A_0, \ldots, A_{n-1}, \ldots, A_{m-1} \rangle$ exists at horizon $m > n$. If a cheaper solution existed for $m > n$ then a relaxed version of it would be encoded by a model μ' of Π_n^{opt} where action sets $A_0, \ldots, A_{n-1}$ appear as in π_m and actions in $A_n, \ldots, A_{m-1}$ are abstracted by actions in A^{abs} which have lower (or same) cost by definition, i.e., $C(\mu') \leq C(\pi_m)$. We can now distinguish three cases: *(i)* $C(\mu') < C(\mu)$, which contradicts the assumption that μ is optimal, *(ii)* $C(\mu) = C(\mu') \leq C(\pi_m)$ and *(iii)* $C(\mu) < C(\mu') \leq C(\pi_m)$ which contradict the assumption that π_m is optimal. □

Algorithm 1 Optimal Planning Modulo Theories

```
1:  procedure OMTPLAN(Π, ub)
2:      set initial horizon n := 0
3:      while n ≤ ub do
4:          build formula Π_n^opt
5:          if Π_n^opt is UNSAT then
6:              return Π does not admit solution;
7:          else
8:              extract model μ for Π_n^opt ;
9:              if μ satisfies G_n then
10:                 return μ
11:             else
12:                 increase horizon n;
13:     return no plan found within bound ub
```

Informally, Theorem 3 states that if a goal can be achieved within a given horizon without using abstract actions, then the plan computed for said horizon is also a global optimum. This condition, in combination with the cost schema we adopt, guarantees optimality of our solutions. As an example, consider a problem where the goal could be achieved by a single, costly action or by a sequence of cheaper actions. Under such circumstances, Π_n^{opt} would be satisfiable for $n = 1$, however our cost schema forces the solver to satisfy the formula by using abstract actions which have lower costs. This, in turn, causes Theorem 3 to be violated and our planning algorithm to continue its search using longer horizons. These ideas are formalized in the following paragraph.

Planning algorithm. We embed our encoding in the OMTPLAN procedure shown in Algorithm 1. Given a planning problem Π and a user-specified upper bound ub, our procedure builds bounded encodings for increasing horizons (lines $2-4$). At each iteration, we check if formula Π_n^{opt} is satisfiable. If it is not the case, the procedure terminates according to Theorem 2 and signals that the planning problem does not admit a solution (lines $5-6$). Notice that, for the purpose of checking reachability, this check could be done only once for $n = 0$. If formula Π_n^{opt} is satisfiable instead, we extract a model μ for it in line 8. Notice that μ has minimum cost among all possible models of Π_n^{opt}, being the result of an OMT check. We then check the condition expressed in Theorem 3: if μ does not contain abstract actions (i.e., $\mu \models G_n$, the goal can be achieved within the given horizon without resorting to abstraction) we return the optimal plan represented by μ, otherwise we increment the horizon for the next iteration (lines $9-12$). Notice that the strategy used to increase n does not affect the optimality results of OMTPLAN as Theorem 3 holds for any $n \geq 0$. Finally, if no solution can be found within the given upper bound ub, the procedure terminates signalling this fact in line 13.

4 Empirical Evaluation

To evaluate OMTPLAN, we developed a prototypical implementation in Python[2]. Our implementation leverages the modules developed in [Eyerich *et al.*, 2009] for parsing, and uses the Python API[3] of νZ [Bjørner *et al.*, 2015] to build and solve OMT formulas. Our experimental analysis compares with search based approaches implemented in the ENHSP planner [Scala *et al.*, 2016b] and with the MILP compilation (C^{SC}) of [Piacentini *et al.*, 2018a]. Experiments are carried out using a timeout of 30 minutes and 4 GB memory limits on a machine running Debian 3.16 with processor Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz. Our analysis considers numeric problems with simple and linear conditions, and also numeric domains with SDACs: simple numeric domains are taken from [Scala *et al.*, 2017]; linear domains are from [Li *et al.*, 2018], with two additions, ROVER-METRIC and FO-ZENOTRAVEL, developed starting from their simple counterparts; finally, planning problems with SDACs are developed specifically to test OMTPLAN.

Domain	#	$\hat{h}^{rmax}$		C^{SC}		OMTPlan	
		C	T	C	T	C	T
COUNTERS	15	6	28.22	**15**	1.36	7	524.59
DEPOTS	20	**3**	1050.02	1	4.9	1	78.48
FARMLAND	30	**30**	193.68	28	32.86	1	211.57
GARDENING	63	**63**	599.85	**63**	887.33	18	3031.23
SAILING	20	16	2101.13	**17**	2813.55	5	345.23
SATELLITE	20	2	293.1	**4**	459.8	1	17.85
ROVER	20	**4**	25.91	**4**	10.93	**4**	61.5
ZENOTRAVEL	20	6	579.3	**7**	699.65	4	107.74
Total	213	130	4871.21	**139**	4910.38	31	4378.19

Table 1: Coverage (C) and total solving time (T) in seconds for domains with simple conditions.

Simple and linear numeric domains. For domains with simple effects we compare against the $\hat{h}^{rmax}$ heuristic of [Scala *et al.*, 2017] and the MILP compilation (C^{SC}) of [Piacentini *et al.*, 2018a]. Table 1 shows coverage and the total solving time. Results confirm the efficiency of C^{SC} on simple numeric problems, outperforming other approaches on almost all instances. On the other hand, our approach suffers from two main drawbacks. The first one is that domains like FARMLAND, GARDENING and SAILING feature optimal plans with relatively many steps and little parallelism. Such "long and narrow" plans force us to produce large encodings that exceed the capabilities of νZ before finding optimal solutions. The second drawback has to do with axioms (6), and affects OMTPLAN adversely even in domains, e.g., COUNTERS, where optimal plans are "short and wide", i.e., featuring relatively few steps and lots of parallelism. Indeed, while (6) tries to make sure that actions are taken before entering the suffix, it may still happen that the optimal solution for a fixed horizon is an abstract solution which also satisfies (6). In such cases, we are still forced to increment the horizon until we exceed the capability of the underlying solver. Note that the interaction between abstraction and axioms (6) is not always harmful as the adverse effect depends on the structure of the domain and the associated costs.

In domains with linear effects we compare our encodings with C^{SC} and with a blind search using a simple goal sensitive heuristic (h^{blind}) that returns 0 if the state is a goal state and 1 otherwise. Table 2 reports results obtained in linear do-

[2]OMTPlan is available at: https://github.com/fraleo/OMTPlan.

[3]https://github.com/Z3Prover/z3/wiki/Documentation

Domain	#	h^{blind}		C^{SC}		OMTPlan	
		C	T	C	T	C	T
FO-COUNT	20	4	339.84	3	223.83	**9**	2104.71
FO-COUNT-INV	20	3	77.29	2	48.82	**6**	937.41
FO-COUNT-RND	60	14	1411.79	10	520.29	**23**	2835.46
FO-FARMLAND	50	**13**	1035.09	2	47.07	1	6.21
FO-SAILING	20	**2**	610.85	0	-	1	71.56
ROVER-METRIC (1-10)	10	4	151.69	4	14.02	**5**	303.39
TPP-METRIC (1-10)	10	**5**	20.51	n.a.	n.a.	3	524.12
ZENOTRAVEL-LINEAR	20	**4**	145.5	2	1.55	**4**	888.24
Total	220	49	3792,29	23	855.58	**52**	7671.10

Table 2: Coverage (C) and total solving time (T) in seconds for domains with linear conditions. Entries reporting n.a. indicate that the planner could not be run on the corresponding domain.

mains. Solving 52 problems, OMTPLAN outperforms other approaches, still leaving room for improvement on specific domains. The overall results is probably due to the increased complexity in the numerical part which OMTPLAN handles comparatively better than the other approaches, except on domains like TPP-METRIC, a variant of the Travelling Salesman Problem with no parallelism.

A new SDAC domain. We introduce a numeric planning domain with SDACs called SECURITY CLEARANCE. In this domain, an intelligence agency has to manage clearance authorizations for several documents across different security levels. The agency can authorize a level to read a document, but doing so changes the clearance of the document: authorizations at lower levels are revoked, while those at higher levels remain unchanged. Authorizing a level has a cost which directly corresponds to the level involved, e.g., authorizing level 2 costs 2. Since some documents may be more important than others, each document is initially assigned a priority. When needed, the agency can increase its priority incurring in a cost proportional to the current one. If a document has high priority, the agency can decide to authorize all levels at once by paying the appropriate price. Starting from an initial state where no level is authorized, the goal is to authorize all levels to read all documents while minimizing expenses.

Experiments on SDACs. For our experiments we generated 36 instances of the domain, varying the number of documents (from 2 to 10) and the number of levels (from 2 to 5). Exploring this domain both in depth and breadth, we can investigate weaknesses of constraint and search-based methods respectively. Here, C^{SC} cannot be considered for our analysis as it does not provide support for state-dependent cost structures. Hence, we compare only with h^{blind}. Figure 1 shows a *cactus plot* of the result obtained: for each planner, we sort the instances according to the run time of the planners and we plot them. In this arrangement, two samples on the same abscissa are not necessarily the result of the same sample, but they correspond to the same percentile. The domain is challenging for both approaches, with OMT being able to solve 26 instances and h^{blind} solving 16. The performance of h^{blind} degrades when the number of documents is increased, incurring in what could be explained as a worst-case behaviour of A^*. Indeed, the planner produces timeouts for almost all instances having strictly more than 5 documents, while already failing to solve some problem with

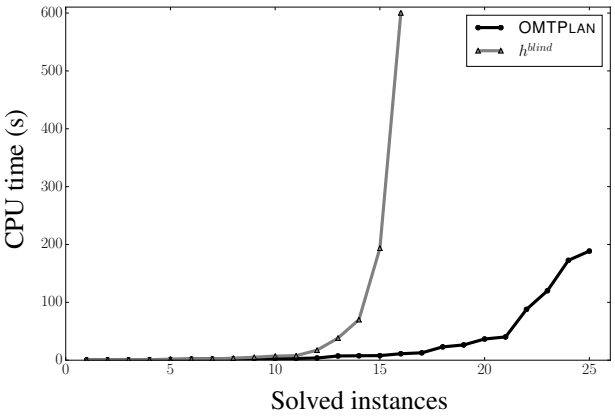


Figure 1: Cactus plot for the SECURITY CLEARANCE domain. Instances are ordered by increasing CPU time, reported in seconds.

less documents. OMTPLAN's performance is comparable to h^{blind} for instances with up to 4 documents (all levels), while a considerable difference can be noticed for instances with higher number of documents. In particular, OMTPLAN always manages to solve instances with 2 or 3 levels, even in domains with 10 documents. Still, domains having 4 or 5 levels proved challenging and could not be solved for instances having 6 documents or more.

5 Conclusion

We considered the problem of generating optimal plans for numeric domains with constant and state-dependent action costs. Since solving these problems requires an efficient interplay between propositional and arithmetic reasoners, we proposed Optimization Modulo Theories as the framework of choice. We presented a novel encoding of planning problems that enables efficient reasoning about optimality via OMT and abstraction. We further provided empirical evidence of the usefulness of our approach, demonstrating state-of-the-art results on some expressive classes of numeric problems.

Our future work will focus on extending the Optimal Planning Modulo Theories framework to problems dealing with other theories, such as that of structured data types. OMT solvers support reasoning over such theories, and we intend to leverage these capabilities to extend our approach. This step could be seen both as a straight-forward implementation of the Planning Modulo Theories framework of [Gregory *et al.*, 2012] and its extension to the optimal setting.

References

[Bjørner *et al.*, 2015] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. νz - An optimizing SMT solver. In *TACAS*, pages 194–199, 2015.

[Eyerich *et al.*, 2009] Patrick Eyerich, Robert Mattmüller, and Gabriele Röger. Using the context-enhanced additive heuristic for temporal and numeric planning. In *ICAPS*, pages 130–137, 2009.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing

temporal planning domains. *J. Artif. Intell. Res. (JAIR)*, 20:61–124, 2003.

[Geißer *et al.*, 2015] Florian Geißer, Thomas Keller, and Robert Mattmüller. Delete relaxations for planning with state-dependent action costs. In *IJCAI*, pages 1573–1579, 2015.

[Geißer *et al.*, 2016] Florian Geißer, Thomas Keller, and Robert Mattmüller. Abstractions for planning with state-dependent action costs. In *ICAPS*, pages 140–148, 2016.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP*, pages 1070–1080, 1988.

[Gregory *et al.*, 2012] Peter Gregory, Derek Long, Maria Fox, and J. Christopher Beck. Planning modulo theories: Extending the planning paradigm. In *ICAPS*, pages 65–73, 2012.

[Haslum *et al.*, 2018] Patrik Haslum, Franc Ivankovic, Miquel Ramírez, Dan Gordon, Sylvie Thiébaux, Vikas Shivashankar, and Dana S. Nau. Extending classical planning with state constraints: Heuristics and search for optimal planning. *J. Artif. Intell. Res.*, 62:373–431, 2018.

[Helmert, 2002] Malte Helmert. Decidability and undecidability results for planning with numerical state variables. In *AIPS*, pages 44–53, 2002.

[Helmert, 2009] Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artif. Intell.*, 173(5-6):503–535, 2009.

[Hoffmann, 2003] Jörg Hoffmann. The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. *J. Artif. Intell. Res.*, 20:291–341, 2003.

[Illanes and McIlraith, 2017] Leon Illanes and Sheila A. McIlraith. Numeric planning via abstraction and policy guided search. In *IJCAI*, pages 4338–4345, 2017.

[Ivankovic *et al.*, 2014] Franc Ivankovic, Patrik Haslum, Sylvie Thiébaux, Vikas Shivashankar, and Dana S. Nau. Optimal planning with global numerical state constraints. In *ICAPS*, pages 145–153, 2014.

[Ivankovic *et al.*, 2019] Franc Ivankovic, Dan Gordon, and Patrik Haslum. Planning with global state constraints and state-dependent action costs. In *ICAPS*, pages 232–236, 2019.

[Kautz and Selman, 1992] Henry A. Kautz and Bart Selman. Planning as satisfiability. In *ECAI*, pages 359–363, 1992.

[Leofante *et al.*, 2018] Francesco Leofante, Erika Ábrahám, and Armando Tacchella. Task planning with OMT: an application to production logistics. In *IFM*, pages 316–325, 2018.

[Leofante *et al.*, 2019] Francesco Leofante, Erika Ábrahám, Tim Niemueller, Gerhard Lakemeyer, and Armando Tacchella. Integrated synthesis and execution of optimal plans for multi-robot systems in logistics. *Information Systems Frontiers*, 2019.

[Li *et al.*, 2018] Dongxu Li, Enrico Scala, Patrik Haslum, and Sergiy Bogomolov. Effect-abstraction based relaxation for linear numeric planning. In *IJCAI*, pages 4787–4793, 2018.

[Lin and Zhao, 2002] Fangzhen Lin and Yuting Zhao. ASSAT: computing answer sets of a logic program by SAT solvers. In *AAAI*, pages 112–118, 2002.

[Piacentini *et al.*, 2018a] Chiara Piacentini, Margarita P. Castro, André Augusto Ciré, and J. Christopher Beck. Compiling optimal numeric planning to mixed integer linear programming. In *ICAPS*, pages 383–387, 2018.

[Piacentini *et al.*, 2018b] Chiara Piacentini, Margarita P. Castro, André Augusto Ciré, and J. Christopher Beck. Linear and integer programming-based heuristics for cost-optimal numeric planning. In *AAAI*, pages 6254–6261, 2018.

[Rintanen *et al.*, 2006] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä. Planning as satisfiability: parallel plans and algorithms for plan search. *Artif. Intell.*, 170(12-13):1031–1080, 2006.

[Rintanen, 2009] Jussi Rintanen. Planning and SAT. In *Handbook of Satisfiability*, pages 483–504. 2009.

[Scala *et al.*, 2016a] Enrico Scala, Patrik Haslum, and Sylvie Thiébaux. Heuristics for numeric planning via subgoaling. In *IJCAI*, pages 3228–3234, 2016.

[Scala *et al.*, 2016b] Enrico Scala, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramírez. Interval-based relaxation for general numeric planning. In *ECAI*, pages 655–663, 2016.

[Scala *et al.*, 2017] Enrico Scala, Patrik Haslum, Daniele Magazzeni, and Sylvie Thiébaux. Landmarks for numeric planning problems. In *IJCAI*, pages 4384–4390, 2017.

[Sebastiani and Tomasi, 2015] Roberto Sebastiani and Silvia Tomasi. Optimization modulo theories with linear rational costs. *ACM Trans. Comput. Log.*, 16(2):12:1–12:43, 2015.

[Shin and Davis, 2005] Ji-Ae Shin and Ernest Davis. Processes and continuous change in a sat-based planner. *Artif. Intell.*, 166(1-2):194–253, 2005.