The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

Expressive Optimal Temporal Planning via Optimization Modulo Theory

Stefan Panjkovic[1, 2], **Andrea Micheli**[1]

[1] Fondazione Bruno Kessler, Trento, Italy
[2] University of Trento, Trento, Italy
{spanjkovic, amicheli}@fbk.eu

Abstract

Temporal Planning is the problem of synthesizing a course of actions given a predictive model of a system subject to temporal constraints. This kind of planning finds natural applications in the automation of industrial processes and in robotics when the timing and deadlines are important. Finding any plan in temporal planning is often not enough as it is sometimes needed to optimize a certain objective function: particularly interesting are the minimization of the makespan and the optimization of the costs of actions. Despite the importance of the problem, only few works in the literature tackled the problem of optimal temporal planning because of the complicated intermix of planning and scheduling.

In this paper, we address the problem of optimal temporal planning for a very expressive class of problems using a reduction of the bounded planning problem to Optimization Modulo Theory (OMT) a powerful discrete/continuous optimization framework. We theoretically and empirically show the expressive power of this approach and we set a baseline for future research in this area.

Introduction

Temporal planning is the problem of automatically synthesizing a course of actions to achieve a goal objective given a model of the system being controlled subject to temporal constraints. This AI problem is of both theoretical and practical interest as it can be used to frame interesting applications ranging from flexible manufacturing (Ruml, Do, and Fromherz 2005) to robotics (Ingrand and Ghallab 2017). Having synchronization constraints or deadlines is commonplace, naturally yielding temporal planning problems.

Over the years, several approaches and algorithms for dealing with temporal planning have been proposed – e.g. (Shin and Davis 2005; Coles et al. 2008, 2010; Valentini, Micheli, and Cimatti 2020). However, most of the papers focus on the problem of satisficing temporal planning, that is finding a plan as good as possible without giving optimality guarantees. Very few papers tackle the problem of finding temporal plans that are *optimal* with respect to an objective function. This landscape is due to the high computational complexity of temporal planning that, even without optimization criteria, is undecidable when time is considered dense, EXPSPACE-complete when time is considered discrete and PSPACE-complete when action self-overlapping is disallowed (Gigante et al. 2022).

The literature in classical and numeric planning (i.e. the untimed cases) recognizes different kinds of objective functions of interest to be optimized while planning: minimizing the cumulative action costs, minimizing or maximizing the final value of a fluent, maximizing the number of optional goals (so-called "oversubscription" planning) or optimizing trajectory preferences. In temporal planning, the only optimization metric that has been studied in depth is the minimization of the "makespan", that is finding a plan achieving the goal in the minimum possible time.

In this paper, we exploit the Optimization Modulo Theory (OMT) framework (Sebastiani and Tomasi 2015), a very general and powerful technique for mixed continuous/discrete optimization. We present the first OMT encoding of optimal temporal planning, capable of optimizing either the makespan or the sum of action costs of plans. Our approach greatly generalizes the work by Leofante et al. (2020), where the authors propose an OMT encoding for the numeric planning case, to temporal planning. Our technique is based on a bounded planning problem formulation in OMT with a concrete part (essentially a SAT-Plan encoding) and a single abstract step that over-approximates the reachable states after the end of the plan. In this way, we can conclude at a certain step that no plans with a better cost are possible even with the addition of more action instances, and thus we achieve optimality. More precisely, this paper makes three major contributions. First, we devise the first encoding in OMT for optimal temporal planning, that is capable of finding plans that are optimal with respect to either the makespan or the sum of action costs. Second, we theoretically prove the soundness and completeness of our approach for temporal planning under the assumption of non-self-overlapping. We also derive theoretical bounds for termination. Third, we empirically evaluate the merits of this approach on a novel set of benchmarks.

Background

SMT and OMT. Given a first-order formula ψ in a background theory T, the Satisfiability Modulo Theory (SMT) problem consists in deciding whether there exists a model

(i.e. an assignment to the free variables in ψ) that satisfies ψ (Barrett et al. 2009). SMT solvers can support different theories such as arithmetic, bit-vectors and arrays. In this paper, we are interested in Quantifier-Free Linear Real Arithmetic ($\mathcal{QF\_LRA}$). A term in $\mathcal{QF\_LRA}$ is of the form $\sum_i a_i x_i$, where every x_i is a real variable and every a_i is a rational constant. An atom is either a Boolean variable or an expression of the form $\varphi \bowtie c$, where φ is a term, $\bowtie \in \{>, <, \geq, \leq, \neq, =\}$, and c is a rational constant. Finally, a formula in $\mathcal{QF\_LRA}$ is a conjunction ($\wedge$), disjunction ($\vee$) or negation ($\neg$) of atoms.

Optimization Modulo Theory (OMT) generalizes SMT with optimization procedures to find a model of ψ that is optimal for an objective function f (or a combination of multiple objective functions) under all models of a formula ψ. The objective function f can be expressed as a term in different theories, but in this paper we focus on objective functions expressed as $\mathcal{QF\_LRA}$ terms. Finally, several OMT solvers (Sebastiani and Trentin 2018; Bjørner, Phan, and Fleckenstein 2015) can solve *Weigthed Max-SMT* problems with tailored algorithms, meaning that optimizing the cost over sets of weighted soft clauses can be efficiently handled.

Notation. Given a $\mathcal{QF\_LRA}$ formula ψ and two terms t_1 and t_2 we write $\text{ITE}(\psi, t_1, t_2)$ to indicate the if-then-else expression meaning "if ψ is true, then take the value t_1, otherwise take the value t_2" This term is natively supported by SMT and OMT solvers (Kim, Somenzi, and Jin 2009). Given three $\mathcal{QF\_LRA}$ terms t_1, t_2 and t_3, we write $t_1 \leq t_2 \leq t_3$ as a shorthand for $t_1 \leq t_2 \wedge t_2 \leq t_3$ (and the same for $<$, $>$ and $\geq$). In this paper we also consider conditions over fluents: we will abuse the terminology saying that these objects are in $\mathcal{QF\_LRA}$ even if they are not defined over SMT variables. Given a formula ϕ, we write $\text{vars}(\phi)$ to indicate the set of variables (or fluents) appearing in ϕ.

OMTPlan. The encoding we present in this paper is inspired by the one presented in (Leofante et al. 2020) for numeric planning: here, we summarize this seminal work. A numeric planning instance Π is a tuple $\langle F, A, I, G\rangle$, where F is a set of fluents, A is a set of instantaneous actions, each a with a set of preconditions pre_a and a set of effects eff_a, I is an initial state and G is a set of goal conditions. The key idea of the approach is to combine a standard encoding for bounded planning as satisfiability (Kautz and Selman 1992; Shin and Davis 2005), which encodes all the plans up to a given depth h, with an abstract step, which encodes an over-approximation of all the possible reachable states after h.

Concretely, the encoding defines an SMT variable (with appropriate typing) f_i for every fluent f in the problem and for every step $i \in \{0, \ldots, h\}$, a Boolean variable a_i for every action a and for every step $i \in \{0, \ldots, h\}$, a Boolean variable mod_f for every fluent f, and a Boolean variable a^{abs} for every action a. The encoding is: $I_0 \wedge \bigwedge_{i=0}^{h-1} T_{i,i+1} \wedge T_h^{abs} \wedge G_h^{abs}$. I_0 encodes the initial states by enforcing appropriate constraints on variables at step 0. $T_{i,i+1}$ indicates the transition relation, constraining the action and fluent variables at steps i and $i+1$ as usual in planning as satisfiability. The formula $T_h^{abs} \wedge G_h^{abs}$ represents the abstract part of the transition relation and the abstract goals. T_h^{abs} is defined as $\bigwedge_{a \in A}(a^{abs} \rightarrow (\bigwedge_{\phi \in \text{pre}_a}(\phi_h \vee \bigvee_{f \in \text{vars}(\phi)} \text{mod}_f))) \wedge \bigwedge_{f \in F}(\text{mod}_f \leftrightarrow (\bigvee_{a \in A, f:=e \in \text{eff}_a} a^{abs}))$. Intuitively, the first part of the formula allows the "starting" of an abstract action a^{abs} if either its preconditions are satisfied at step h or if at least one fluent f appearing in the condition can be abstractly modified (by setting mod_f to true). The second part sets a mod_f variable iff at least one action having some effect on f is fired in the abstract step.

G_h^{abs} is defined as $\bigwedge_{\phi \in G}(\phi_h \vee \bigvee_{f \in \text{vars}(\phi)} \text{mod}_f)$. The intuitive meaning is that the goal can be either satisfied concretely, by the variables at step h, or abstractly by checking that some fluent f of the goal can be modified abstractly. The key intuition is that mod_f variables are used to mark whether a fluent f can be "touched" by an effect in the abstract part of the encoding by firing one abstract action, and abstract actions can be fired if their preconditions are either supported by the last concrete step or if at least one variable occurring in the precondition can be abstractly modified.

Leofante et al. assume that each action a has an associated cost given by $c(a) : S \rightarrow \mathbb{Q}_{\geq 1}$ and use the term $\sum_{a \in A}(\sum_{i=0}^{h-1} \text{ITE}(a_i, c(a), 0)) + \text{ITE}(a^{abs}, 1, 0)$ as an objective for OMT to optimize[1]. We indicate this OMT encoding as Π_h^{opt} and we report the OMTPLAN procedure in Algorithm 1. The algorithm incrementally augments the bound h and asks the OMT solver to decide the satisfiability of Π_h^{opt}. If the formula is unsatisfiable, it means that the goal is unreachable even in the abstract over-approximation, so no solution exists. If it is satisfiable and the goal is reached without using abstract actions, then an optimal model for the OMT formula yields an optimal plan (it suffices to apply the sequence of parallel actions where a_i is true in the model); otherwise, the check is inconclusive and we need to expand the bound. Numeric planning is in general an undecidable problem, so this procedure is not guaranteed to terminate.

Algorithm 1 Optimal Planning via OMT

```
1  procedure OMTPLAN(Π)
2      h ← 1
3      while True do
4          if Π_h^opt is UNSAT then
5              return Π does not admit solution
6          else
7              extract optimal model μ for Π_h^opt
8              if μ satisfies the concrete goal then
9                  return EXTRACTPLAN(μ)
10             else h ← h + 1
```

Problem Definition

We formalize our planning problem following the standard PDDL 2.1 conventions, a complete overview of the semantics of the language is given in (Fox and Long 2003).

Definition 1. *A **temporal planning problem** Π is a tuple $\langle F, A, I, G\rangle$:*

- *F is a finite set of Boolean and real fluents.*

[1] A lower-bound on $c(a)$ can be used instead of 1 in the ITE.

- *A is a set of actions a of the form*[2] *$\langle pre_a, eff_a, dur_a \rangle$:*
  - *pre_a is a set of conditions partitioned in three subsets $pre_{\vdash a}$, $pre_{\leftrightarrow a}$, $pre_{\dashv a}$, which consist of start, overall and end conditions respectively.*
  - *eff_a is a set of effects partitioned in two subsets $eff_{\vdash a}$, $eff_{\dashv a}$, which consist of start and end effects respectively.*
  - *dur_a is a set of duration constraints.*
- *$I : F \to \{\top, \bot\} \cup \mathbb{R}$ is the total function describing the initial value of the fluents.*
- *G is a set of goal conditions.*

Without loss of generality, a condition is a Boolean $\mathcal{QF}\_\mathcal{LRA}$ expression over the fluents and an effect is of the form $f := e$ where f is a fluent and e is an appropriately-typed $\mathcal{QF}\_\mathcal{LRA}$ expression over the fluents.

Starting from the initial assignment indicated by I, we can apply any action in A provided that the starting conditions are satisfied and ensuring (possibly by applying other actions) that the overall and ending conditions will be maintained and satisfied, respectively. The state changes either when an action starts, by applying the starting effects or when it terminates applying the ending effects.

Definition 2. *A **time-triggered plan** π for Π is a sequence $\langle \langle t_1, a_1, d_1 \rangle, \langle t_2, a_2, d_2 \rangle, \ldots, \langle t_n, a_n, d_n \rangle \rangle$, where $t_i \in \mathbb{Q}_{\geq 0}$ is the starting time, $a_i \in A$ is the action to be started, $d_i \in \mathbb{Q}_{\geq 0}$ is the action duration, and $t_i \leq t_{i+1}$.*

Intuitively, a plan π is valid for a planning problem Π if by starting from the initial state and by executing the actions prescribed by the plan at the indicated time and with the chosen durations, all the conditions of every action are fulfilled and after the end of the last action all the goal conditions are satisfied. We refer the reader to (Fox and Long 2003) for a thorough definition of plan validity for temporal planning.

Definition 3. *A plan $\langle \langle t_1, a_1, d_1 \rangle, \ldots, \langle t_n, a_n, d_n \rangle \rangle$ has **self-overlapping** if there exist $i, j \in \{1, \ldots, n\}$ such that $a_i = a_j$ and $t_i \leq t_j < t_i + d_i$.*

In our encoding, we target a decidable fragment of temporal planning where we disallow action self-overlapping (Gigante et al. 2022). Moreover, we adopt an ϵ-separation semantics between consecutive timepoints (like in PDDL 2.1). The encoding can be easily adapted to handle the non-zero separation case, but this could lead to some problems not admitting a concrete optimal solution (the optimum could be to start an action an infinitesimal time after another).

Definition 4. *Given a plan π, its **makespan** is $max_{\langle t,a,d \rangle \in \pi}(t + d)$. π is **makespan-optimal** if it is valid and there is no other valid plan with smaller makespan.*

Intuitively, the makespan is the time when the last action terminates in a valid plan.

Definition 5. *Given a map from actions to rational costs $c : A \to \mathbb{Q}_{>0}$, the **cumulative action cost** of a plan π is $\sum_{\langle t,a,d \rangle \in \pi} c(a)$. π is **action-cost-optimal** if it is valid and there is no other valid plan with smaller cost.*

In this paper, we want to find makespan-optimal and action-cost-optimal plans for temporal planning problems.

[2]For simplicity, we focus on durative actions; instantaneous actions are durative actions with only starting conditions and effects.

OMT Encoding

We will now describe our OMT encoding for optimal temporal planning (Figure 1). We follow the same procedure OMTPLAN presented in Algorithm 1, but generalize the construction of Π_h^{opt} to the temporal case. The main idea is that the first $h-1$ steps represent a standard encoding for bounded planning (Shin and Davis 2005), while step h is the abstract step encoding the over-approximation of all reachable states (unlike before, we represent *concrete* steps up to $h-1$, and use h as the abstract step).

We start by defining the variables of our encoding. For every step $i \in \{0, \ldots, h\}$ we use t_i to denote the time associated to step i; for every fluent f in the problem we define the variable f_i, representing the value of f at step i; for every action a we define the Boolean variable a_i, which indicates whether action a is started at step i; the variables d_i^a denote the duration of action a when started at step i. Finally, the mod_f variables for each fluent f are used to indicate whether in the over-approximation f can be potentially modified after the last *concrete* step $h-1$.

In the following, we will use the notation $[e]_i$, where e is an expression and i is a step, to denote the expression obtained by substituting each fluent $f \in \text{vars}(e)$ with f_i.

The overall formula Π_h^{opt} is made of several components which encode different aspects of the problem: φ^{INIT} (initial state); $\varphi_h^{\text{ACTIONS}}$ (actions encoding); φ_h^{TIME} (time monotonicity); φ_h^{FA} (frame axiom); φ_h^{NSO} (action non-self-overlapping); φ_h^{MOD} (mod_f variables); φ_h^{DENS} (density axiom); φ_h^{GOAL} (abstract goals).

The formula φ^{INIT} encodes the initial state, simply by setting the values f_0 for each fluent f to the initial values, as defined in the problem.

The formula $\varphi_h^{\text{ACTIONS}}$ iterates on all the actions $a \in A$ and all the steps $s \in \{1, \ldots, h\}$, adding the implication that if a is started at step s, then a conjunction of formulas must hold which enforce the duration constraints of the action and the proper application of all the action conditions and effects. $\varphi_{s,h}^{\text{pre}_{\vdash a}}$ encodes the start conditions of a, stating that if the action is started in a *concrete* step ($t_s \leq t_{h-1}$) then the condition must hold at that step ($[e]_s$), otherwise if the action is started in the abstract step ($t_s > t_{h-1}$) then the condition must either hold in the last *concrete* step ($[e]_{h-1}$) or at least one mod_f variable for a fluent f appearing in e must be true (the condition can potentially become true in the future). The overall conditions are encoded by $\varphi_{s,h}^{\text{pre}_{\leftrightarrow a}}$, which enforces three constraints: if a step p happens between the start and the end of action a ($t_s < t_p < t_s + d_s^a$) then the condition must hold in p ($[e]_p$); in the first step after the end of the action the condition must still hold, which handles cases where the open interval $(t_s, t_s + d_s^a)$ is skipped; finally, if the action is started in the abstract step, then the overall condition must either hold at step $h-1$ or at least one mod_f variable for a fluent f appearing in e must be true. The end conditions are encoded by $\varphi_{s,h}^{\text{pre}_{\dashv a}}$, stating that the condition must hold at the first step that happens at or after the end of the action ($t_{p-1} < t_s + d_s^a \leq t_p \to [e]_p$) and that if the end of the action is scheduled to happen after the last *concrete* step, then

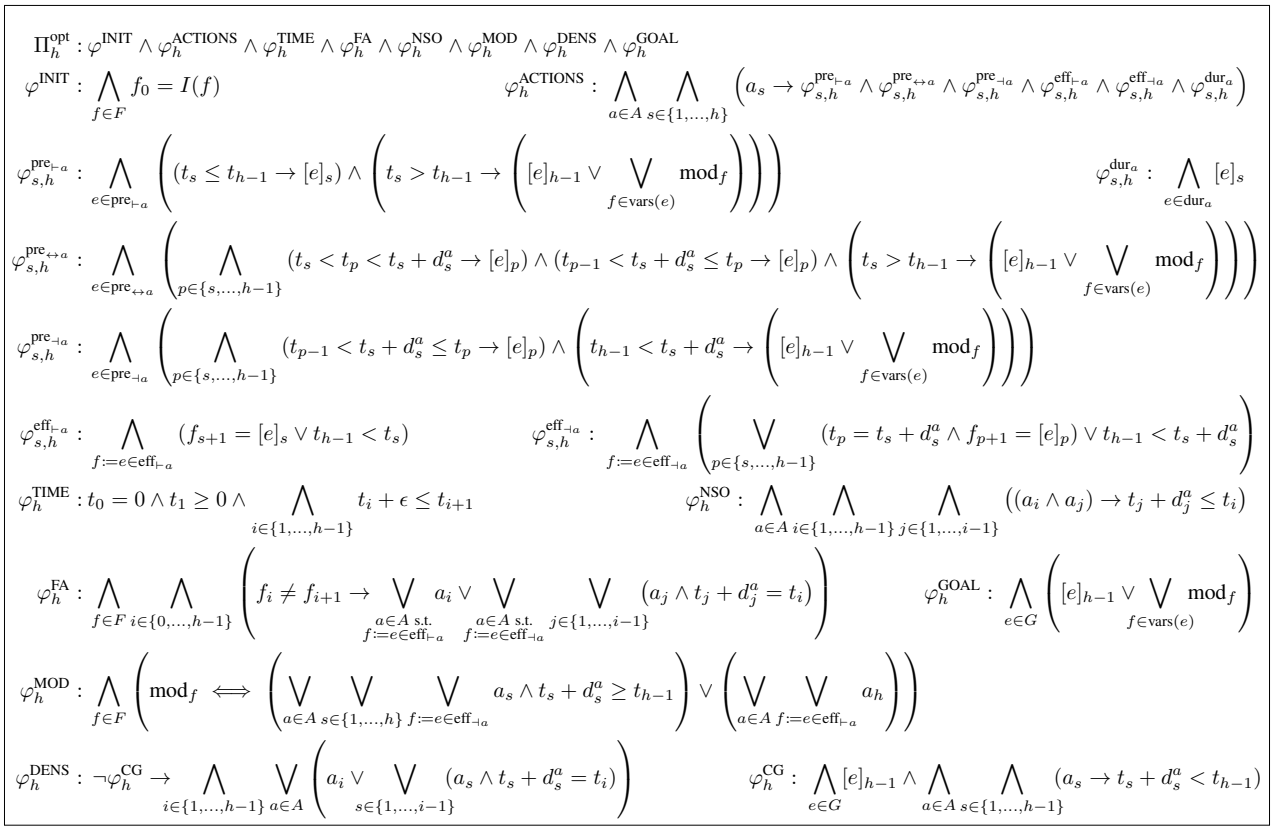

Figure 1: The OMT encoding formulation

the condition must hold at step $h-1$ or at least one appropriate mod_f variable must be true. The start and end effects are encoded by formulas $\varphi^{\text{eff}_{\vdash a}}_{s,h}$ and $\varphi^{\text{eff}_{\dashv a}}_{s,h}$ respectively, which enforce that there must be a step p in which the effect is applied ($f_{p+1} = [e]_p$) or the effect must be scheduled after the last *concrete* step ($t_{h-1} < t_s$ and $t_{h-1} < t_s + d^a_s$ respectively). Finally, $\varphi^{\text{dur}_a}_{s,h}$ simply enforces that all duration constraints must hold at step s, when the action is started. This restricts the values that d^a_s can take.

The formula φ^{TIME}_h encodes the fact that time is non-negative ($t_1 \geq 0$) and strictly increasing ($t_i + \epsilon \leq t_{i+1}$).

The *frame axiom* (Shanahan 1997) is expressed by φ^{FA}_h, which allows fluent variables to change only if an effect is applied to them. For every fluent f, if it changes from one step to the next ($f_i \neq f_{i+1}$) that implies that at step i either an action with a start effect on f is started, or an action with an end effect on f is ended.

The formula φ^{NSO}_h disallows action self-overlapping: if an action a is started at two steps j and i with $j < i$, then the execution which starts at j must conclude before the new execution starts at i ($t_j + d^a_j \leq t_i$).

The formula φ^{MOD}_h controls when mod_f variables can be set to true. Given a fluent f, mod_f is true iff an action with a start effect on f is started in the *abstract* step, or if an action with an end effect on f is started and the end of that action is scheduled to happen after t_{h-1}. The idea is that a true mod_f variable represents a necessary condition for f to be modifiable in the future.

The formula φ^{DENS}_h ensures that if the *concrete* goal φ^{CG}_h is not satisfied by a model of the formula (all goal conditions are satisfied at step $h-1$ and no action is left running), then each step up to $h-1$ must be filled with either the start or the end of an action. Without this restriction an OMT solver would be likely to return for each depth h an optimal solution which leaves most of the *concrete* steps empty and applies the necessary abstract actions to reach the goal in our over-approximation. This formula does not affect soundness, because a valid solution plan satisfies the *concrete* goal, and thus the requirement to fill all *concrete* steps does not apply.

Finally, φ^{GOALS}_h states that each goal condition must be either satisfied at step $h-1$, or at least one mod_f variable for a fluent f appearing in the condition must be true. An example of our encoding is given in Figure 2.

As mentioned at the beginning, we follow the OMTPLAN procedure of Algorithm 1. If Π^{opt}_h is unsatisfiable, we return the fact that Π does not admit any solution. Otherwise, we extract an optimal model μ from Π^{opt}_h: if we want to minimize the makespan, we specify t_{h-1} as the minimization objective, while for minimizing the total action cost the objective is $\sum_{a\in A}\sum_{i=1}^{h} \text{ITE}(a_i, c(a), 0)$, where $c(a)$ is the cost of action a. If μ satisfies the *concrete* goal φ^{CG}_h then we ex-

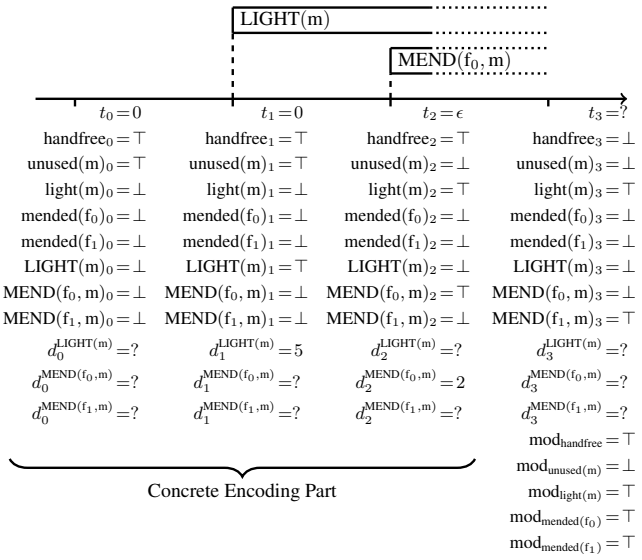

Figure 2: Visualization of the encoding model with horizon $h = 3$ for a MATCHCELLAR problem. Question marks indicate values that are irrelevant for the plan reconstruction.

tract a plan from μ and return it as an optimal solution for Π; otherwise, we increment the depth h and repeat the procedure. To extract a time-triggered plan π from a model μ (the EXTRACTPLAN(μ) function in Algorithm 1) we simply look at which variables a_i are set to true, and for each of them we add the tuple $\langle t_i, a, d_i^a \rangle$ to π.

In the remainder of this section we will prove that the proposed approach is a *sound* and *complete* procedure for finding makespan-optimal and action-cost-optimal plans for temporal planning problems. The following definitions of *timepoints* and *length* of a plan will be used in the proofs.

Definition 6. *Given a time-triggered plan* $\pi = \langle\langle t_1, a_1, d_1\rangle, \ldots, \langle t_n, a_n, d_n\rangle\rangle$*, we define the* ***set of timepoints*** *of* π *to be* $T_\pi = \bigcup_{i=1}^{n} \{t_i\} \cup \{t_i + d_i\}$*, i.e. all the times at which an action is started or ended.*
We define the ***length of*** π *to be the number of distinct timepoints of* π*, i.e. the cardinality of* T_π*.*

Theorem 1 (Validity and Soundness)**.** *For every temporal planning problem* Π*, if* OMTPLAN *(*Π*) terminates and returns plan* π*, then* π *is a valid and optimal solution for* Π*.*

Proof. (Sketch) Suppose that OMTPLAN (Π) returns a plan π at step h. Let μ be the optimal model of Π_h^{opt} from which π was extracted. Validity comes from the fact that the *concrete* part of the encoding Π_h^{opt} is a valid SMT encoding of plans of bounded length (Shin and Davis 2005) and because, since $\mu \models \phi_h^{\text{CG}}$, the goals are satisfied at the last *concrete* timepoint and no action is left running.

To prove optimality of π, we consider the two optimization problems separately. In both cases the intuition is that if there is a more optimal plan that can be found at a later step, then a relaxed version of that plan can be found at the current step, which would be more optimal than the solution returned by the OMT solver.

Minimum makespan. Let d be the makespan of π. Suppose for the sake of contradiction that there is a solution plan π' for Π with length l' and makespan $d' < d$. If $l' \leq h - 1$, then there exists a model μ' of Π_h^{opt} from which π' can be extracted, but this contradicts the assumption that μ is an optimal model for Π_h^{opt}. Suppose that $l' > h - 1$. From π' we can construct a plan π'' with length $h - 1$ and makespan d'', by starting in order the same actions as in π' until reaching length $h-1$, and then running an abstract action for each action that is left. We can observe that π'' can be extracted from a model μ'' of Π_h^{opt}: the preconditions of all the abstract actions are either already satisfied at step $h - 1$ or have a required mod variable set to true, since they were applicable in the plan π'; moreover, since goal conditions were all satisfied by π', they will be either satisfied or have a required mod variable set to true by π''. Additionally, since π'' is a shortened version of π', we have $d'' < d' < d$. This contradicts the assumption that μ is an optimal model for Π_h^{opt}.

Minimum action cost. The proof in this case is similar to the previous one. When considering the total action cost of π'' we observe that it is less than or equal to the total action cost of π', because the cost of an abstract action is always a lower bound to the cost of the same concrete action. □

Next, we prove *completeness*. The main intuition is that if the OMT solver keeps returning solutions which rely on the *abstract* component of the formulas, the value of the objective will keep increasing, because φ_h^{DENS} enforces the solver to fill all the *concrete* steps. Eventually, the value of the objective will exceed the value of the optimal solution, and the solver will be forced to return a *concrete* optimal plan.

Lemma 1. *Consider a temporal planning problem* Π*. Let* N *be the number of actions and let* d_{min} *be the minimum possible duration of an action in* Π*. Suppose* π^* *is a solution for* Π *with minimum makespan* d^**. Let* μ' *be an optimal model for* $\Pi_{h^*}^{opt}$ *where* $h^* = \left\lceil \frac{d^*}{d_{min}} \right\rceil \cdot N \cdot 2 + 1$*, and let* π' *be the plan extracted from* μ'*. Then* $\mu' \models \varphi_{h^*}^{CG}$ *and* π' *has makespan* d^**.*

Proof. (Sketch) First, we observe that π^* has *length* at most $\left\lceil \frac{d^*}{d_{\text{min}}} \right\rceil \cdot N \cdot 2$. Given that the makespan is d^*, an upper bound on the number of timepoints of π^* can be found by assuming that all N actions have duration d_{min} and that all are run in parallel for $\frac{d^*}{d_{\text{min}}}$ times. It is not possible to have more than N actions running in parallel because of the assumption of non-self-overlapping. Since there could be a timepoint each time a new action is started or ended, we can conclude that π^* has at most $\left\lceil \frac{d^*}{d_{\text{min}}} \right\rceil \cdot N \cdot 2$ distinct timepoints. Therefore there exists a model μ^* of $\Pi_{h^*}^{\text{opt}}$ from which π^* can be extracted, as h^* is large enough to encode all the timepoints of π^*. Since μ' is an optimal model of $\Pi_{h^*}^{\text{opt}}$, π' has makespan at most d^*.

Suppose for the sake of contradiction that $\mu' \not\models \varphi_{h^*}^{\text{CG}}$. As a model of $\Pi_{h^*}^{\text{opt}}$, μ' satisfies $\varphi_{h^*}^{\text{DENS}}$, and therefore at each step between 1 and $h^* - 1$ an action must start or end. However, even assuming that all N actions have duration d_{min}

and running them all in parallel until all steps are filled, the makespan would still exceed d^*, given how h^* is defined. This contradicts the fact that π' has makespan at most d^*, and therefore it must be the case that $\mu' \models \varphi^{\text{CG}}_{h^*}$. Hence, π' is a valid solution plan for Π, and since d^* is the minimum makespan for a solution of Π and we already proved that π' has makespan at most d^*, π' has makespan exactly d^*. □

Lemma 2. *Consider a temporal planning problem* Π. *Let* c_{min} *be the minimum possible cost of an action in* Π *and suppose that* π^* *is a solution for* Π *with minimum total action cost* c^*. *Let* μ' *be an optimal model for* $\Pi^{opt}_{h^*}$ *where* $h^* = \left\lceil \frac{c^*}{c_{min}} \right\rceil \cdot 2 + 1$, *and let* π' *be the plan extracted from* μ'. *Then* $\mu' \models \varphi^{CG}_{h^*}$ *and* π' *has total action cost* c^*.

Proof. (Sketch) We begin by observing that π^* has *length* at most $\left\lceil \frac{c^*}{c_{\text{min}}} \right\rceil \cdot 2$. Given that the total action cost is c^*, an upper bound on the number of timepoints of π^* is obtained by applying $\frac{c^*}{c_{\text{min}}}$ times the action with cost c_{min}. As we can have a new timepoint each time an action is started or ended, π^* has at most $\left\lceil \frac{c^*}{c_{\text{min}}} \right\rceil \cdot 2$ timepoints. Therefore there exists a model μ^* of $\Pi^{\text{opt}}_{h^*}$ from which π^* can be extracted, as h^* is large enough to encode all the timepoints of π^*. Since μ' is an optimal model of $\Pi^{\text{opt}}_{h^*}$, π' has total action cost at most c^*.

As in the previous lemma, we can prove that $\mu' \models \varphi^{\text{CG}}_{h^*}$, because otherwise the requirement to fill each step with an action start or end would make the total action cost of π' exceed the value of c^*. It follows that π' is a valid solution plan for Π, and similarly to before we can conclude that π' has total action cost exactly c^*. □

Theorem 2 (Completeness)**.** *If there exists an optimal solution for* Π, *then* OMTPLAN *(*Π*) will eventually terminate and return an optimal solution for* Π.

Proof. This is a direct consequence of the previous two lemmas. If there exists an optimal solution π^* for Π, then at step h^* the obtained model μ' will satisfy $\varphi^{\text{CG}}_{h^*}$ so the algorithm will terminate, and the extracted plan will be an optimal solution for Π. If a plan is returned before step h^*, then by Theorem 1 it is also an optimal solution for Π. □

Encoding Optimizations

We added several extensions and optimizations to the encoding to improve its expressiveness and performance.

First, we added Intermediate Conditions and Effects (ICE) (Valentini, Micheli, and Cimatti 2020), that is conditions and effects happening at arbitrary points in time during the execution of an action. This can be captured in the encoding by adding the necessary timing constraints which imply the application of the event. E.g., if an effect is supposed to happen 4 units of time after the start of an action, then applying it at step p means that $t_p = t_s + 4$, where s is the step at which the action is started. The density axiom φ^{DENS}_h is also modified to enforce at each *concrete* step the starting/ending of an action or the application of an intermediate event.

A performance optimization for the action-cost case can be obtained by expressing the objective function as a Weighted Max-SMT problem: it suffices to rewrite the $\sum_{a \in A} \sum_{i=1}^{h} \text{ITE}(a^i, c(a), 0)$ objective as a set of weighted soft clauses $\neg a^i$ with cost $c(a)$ for each step i and action a.

Finally, major SMT and OMT solvers support incrementality, meaning they can efficiently re-use search artifacts derived from solving a formula ϕ to reason on a formula $\phi \wedge \psi$. Solvers offer a stack-based interface to add and retract formulas, and perform checks on the conjunction of formulas in the stack (Barrett et al. 2009). Since the OMTPLAN algorithm needs to solve a sequence of OMT queries by increasing the horizon h, we devised a partially-incremental version of the encoding that can exploit this feature. We write the encoding in two parts (with additional variables to allow this rewriting): a permanent part that is never retracted from the solver and a temporary part that is specific to the check with horizon h. The full incremental encoding with ICE support is available in (Panjkovic and Micheli 2023).

Related Work

Not many papers address optimal temporal planning. Some optimal temporal planners are limited to "conservative" planning models (Smith and Weld 1999) where action preconditions must hold throughout the whole action execution and the effects are applied at the end: this model is much simpler than the full PDDL 2.1 semantics that we address, because in a conservative model, actions having conflicting preconditions or effects cannot overlap in time in any way. CPT (Vidal 2011), TP4 and HSP* (Haslum 2006) are three optimal temporal planners that are limited to conservative models. (Haslum 2009) presents a technique to use admissible heuristics for conservative models as relaxations for non-conservative models, but to the best of our knowledge no planner currently implements this solution. The only planner capable of dealing with the full PDDL 2.1 and perform optimization is OPTIC (Benton, Coles, and Coles 2012): even though the main focus of the tool is on satisficing planning, the planner can generate increasingly better solutions until it is able to prove that no better solution exists. This is possible thanks to an admissible heuristic used for pruning in combination with a non-admissible one used for search with WA*. In addition, the authors very recently published a research on a novel merge-and-shrink heuristic (Brandao et al. 2022). We empirically compare with the OPTIC planner in the next section. Other optimal temporal planners include: TPSYS (Tejero and Onaindia 2006), which uses a graphplan-based technique and is optimal for a fragment of PDDL2.1; and CPPLANNER (Dinh 2004), which uses a temporally-expressive proprietary language. SCP2 (Lu et al. 2013) uses an encoding into Weighted Max-SAT for optimizing the makespan as well as action costs. This encoding assumes a discrete integer time model and requires a copy of all the fluents at each time, making it extremely sensitive to the scale of time. Our encoding, instead, uses the $\mathcal{QF\_LRA}$ theory to model *continuous* time and we create a copy of all the fluents for every action start or action termination only.

The use of SMT for temporal planning has been pi-

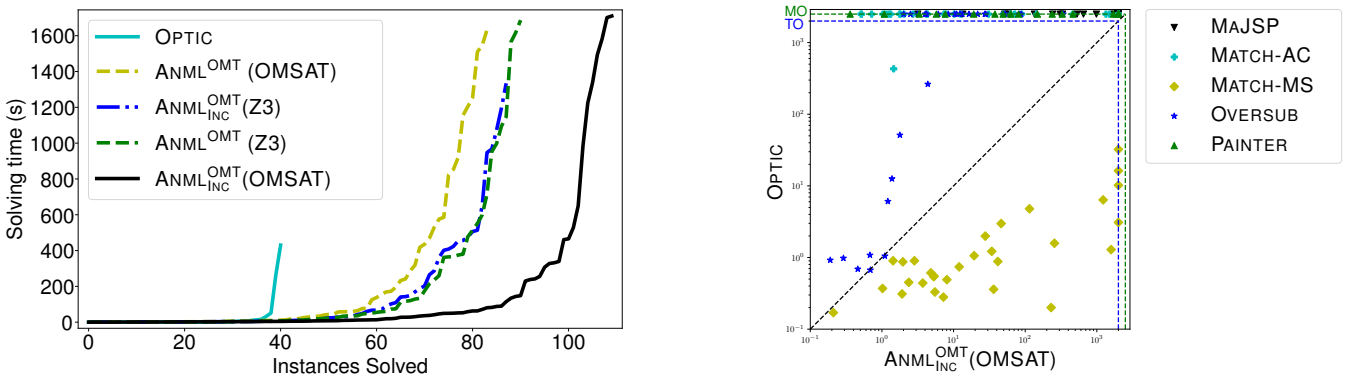

Figure 3: Result plots: cactus plot (left) for all the approaches and scatter plot (right) for OPTIC and $\text{ANML}^{\text{OMT}}_{\text{INC}}$(OMSAT).

oneered (without considering optimization) by Shin and Davis (2005) and other optimized temporal planning encodings have been presented (Rintanen 2017). The encoding we present is a generalization to the temporal planning case of the one introduced in (Leofante et al. 2020). The generalization is very non trivial. First, in temporal planning we need to deal with "future commitments", meaning that if a durative action is started, we need to ensure its overall and at-end conditions are respected in subsequent steps and we need to deal with the case in which these commitments are beyond the concrete part of the encoding. Moreover, we use OMT to formulate optimization criteria for both optimal-makespan and optimal-action-costs problems. Finally, we generalize and optimize the encoding to support ICE, to be incremental and to use a more efficient Max-SMT formulation.

Experiments

We implemented the planning algorithm via OMT in a C++ tool called ANML^{OMT}, which can read either ANML (Smith, Frank, and Cushing 2008) or PDDL 2.1 specifications and can use either OPTIMATHSAT (Sebastiani and Trentin 2018) or Z3 (Bjørner, Phan, and Fleckenstein 2015) as OMT solvers. It is also possible to choose whether or not to use the incremental encoding. We compare our tool in four different configurations: with or without incrementality (incremental solvers are indicated as $\text{ANML}^{\text{OMT}}_{\text{INC}}$) and with either OPTIMATHSAT or Z3 as OMT engines. As a competitor, we compare against the OPTIC (Benton, Coles, and Coles 2012) planner as it is the only planner publicly available that is capable of dealing with our benchmarks. We ran the experiments on an Intel Core i9-9900KS workstation with a 1800s time limit and 20GB of memory limit.

Domain	OPTIC	ANML^{OMT} OMSAT	ANML^{OMT} Z3	$\text{ANML}^{\text{OMT}}_{\text{INC}}$ OMSAT	$\text{ANML}^{\text{OMT}}_{\text{INC}}$ Z3
MAJSP (80)	0(10)	11	**12**	**12**	**12**
PAINTER (30)	0(4)	14	13	**15**	12
MATCH-MS (30)	**30**(30)	27	28	26	28
MATCH-AC (30)	1(30)	25	24	**27**	24
OVERSUB (30)	10(30)	7	14	**30**	12
Total (200)	41(104)	84	91	**110**	88

Table 1: Coverage table. We write X(Y) to indicate that X instances were proven optimal and Y instances were solved.

We consider five domains. MAJSP and PAINTER are industrial-inspired domains taken from (Valentini, Micheli, and Cimatti 2020), where we optimize the makespan. MATCH-MS is a classical "MatchCellar" IPC domain, while MATCH-AC is a variation of the same domain where matches have different costs and different burning durations, and the objective is to minimize the total action cost. OVERSUB, instead, is a synthetic problem in which we encode an oversubscription planning problem as action-cost optimization. The benchmarks and the tools used for the experimental evaluation are available in (Panjkovic and Micheli 2023).

The coverage results are reported in Table 1 while in Figure 3 we plot the performance of the solvers. With the notable exception of MATCH-MS where OPTIC performs really well proving the optimum for all the instances, $\text{ANML}^{\text{OMT}}_{\text{INC}}$(OMSAT) is consistently faster and can solve many more instances than the other competitors. This is clearly shown in the scatter plot of Figure 3 where we differentiated the scatter points per-domain. An interesting emerging fact is that the use of the incremental encoding is very useful when OMSAT is used, but is detrimental for Z3 .

We report that we experimented with other IPC domains, but our encoding is inefficient when long, sequential plans are needed. This is not surprising as SAT/SMT planning encodings poorly scale on the number of objects, while gracefully dealing with complex constraints. Despite this limitation, our results indicate a strong complementarity in terms of coverage and performance (see scatter plot in Figure 3).

Conclusions

In this paper, we presented the first OMT encoding for *optimal* temporal planning. Our encoding can optimize either the plan makespan or the sum of action costs and is proved to be sound and complete for the non-self-overlapping semantics. We also described three optimizations of the encoding and we experimentally evaluated the approach.

In the future, we plan to hybridize our approach with (Rintanen 2017) to better tackle IPC domains and with (Cashmore, Magazzeni, and Zehtabi 2020) for continuous change.

Acknowledgements

This work has been partly supported by the project "AI@TN" funded by the Autonomous Province of Trento and by the "AIPlan4EU" project funded by EU Horizon 2020 research and innovation programme under GA n. 101016442.

References

Barrett, C. W.; Sebastiani, R.; Seshia, S. A.; and Tinelli, C. 2009. Satisfiability Modulo Theories. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, 825–885. IOS Press. ISBN 978-1-58603-929-5.

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *ICAPS 2012*.

Bjørner, N. S.; Phan, A.; and Fleckenstein, L. 2015. νZ - An Optimizing SMT Solver. In Baier, C.; and Tinelli, C., eds., *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9035 of *Lecture Notes in Computer Science*, 194–199. Springer.

Brandao, M.; Coles, A. J.; Coles, A.; and Hoffmann, J. 2022. Merge and Shrink Abstractions for Temporal Planning. In Kumar, A.; Thiébaux, S.; Varakantham, P.; and Yeoh, W., eds., *ICAPS 2022, Singapore (virtual), June 13-24, 2022*, 16–25. AAAI Press.

Cashmore, M.; Magazzeni, D.; and Zehtabi, P. 2020. Planning for Hybrid Systems via Satisfiability Modulo Theories. *J. Artif. Intell. Res.*, 67: 235–283.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. Planning with Problems Requiring Temporal Coordination. In *AAAI*.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *ICAPS 2010*.

Dinh, T. B. 2004. Solution Extraction with the "Critical Path" in Graphplan-Based Optimal Temporal Planning. In Wallace, M., ed., *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, 793. Springer.

Fox, M.; and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*.

Gigante, N.; Micheli, A.; Montanari, A.; and Scala, E. 2022. Decidability and complexity of action-based temporal planning over dense time. *Artif. Intell.*, 307: 103686.

Haslum, P. 2006. Improving heuristics through relaxed search-an analysis of TP4 and HSP* a in the 2004 planning competition. *Journal of Artificial Intelligence Research*, 25: 233–267.

Haslum, P. 2009. Admissible makespan estimates for pddl2. 1 temporal planning. In *Proceedings of the ICAPS Workshop on Heuristics for Domain-Independent Planning*.

Ingrand, F.; and Ghallab, M. 2017. Deliberation for autonomous robots: A survey. *Artif. Intell.*, 247: 10–44.

Kautz, H. A.; and Selman, B. 1992. Planning as Satisfiability. In Neumann, B., ed., *10th European Conference on Artificial Intelligence, ECAI 92, Vienna, Austria, August 3-7, 1992. Proceedings*, 359–363. John Wiley and Sons.

Kim, H.; Somenzi, F.; and Jin, H. 2009. Efficient Term-ITE Conversion for Satisfiability Modulo Theories. In *SAT*, 195–208.

Leofante, F.; Giunchiglia, E.; Ábrahám, E.; and Tacchella, A. 2020. Optimal Planning Modulo Theories. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 4128–4134. ijcai.org.

Lu, Q.; Huang, R.; Chen, Y.; Xu, Y.; Zhang, W.; and Chen, G. 2013. A SAT-based approach to cost-sensitive temporally expressive planning. *ACM Trans. Intell. Syst. Technol.*, 5(1): 18:1–18:35.

Panjkovic, S.; and Micheli, A. 2023. Expressive Optimal Temporal Planning via Optimization Modulo Theory: Additional Material. https://es-static.fbk.eu/people/panjkovic/aaai23. Accessed: 2022-11-28.

Rintanen, J. 2017. Temporal Planning with Clock-Based SMT Encodings. In Sierra, C., ed., *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 743–749. ijcai.org.

Ruml, W.; Do, M. B.; and Fromherz, M. P. 2005. On-line Planning and Scheduling for High-speed Manufacturing. In *ICAPS*, 30–39.

Sebastiani, R.; and Tomasi, S. 2015. Optimization Modulo Theories with Linear Rational Costs. *ACM Trans. Comput. Log.*, 16(2): 12:1–12:43.

Sebastiani, R.; and Trentin, P. 2018. OptiMathSAT: A Tool for Optimization Modulo Theories. *Journal of Automated Reasoning*.

Shanahan, M. 1997. *Solving the frame problem - a mathematical investigation of the common sense law of inertia*. MIT Press. ISBN 978-0-262-19384-9.

Shin, J.-A.; and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artificial Intelligence*.

Smith, D.; Frank, J.; and Cushing, W. 2008. The ANML language. In *KEPS 2008*.

Smith, D. E.; and Weld, D. S. 1999. Temporal planning with mutual exclusion reasoning. In *IJCAI*, volume 99, 326–337.

Tejero, A. G.; and Onaindia, E. 2006. Domain-independent temporal planning in a planning-graph-based approach. *AI Commun.*, 19(4): 341–367.

Valentini, A.; Micheli, A.; and Cimatti, A. 2020. Temporal Planning with Intermediate Conditions and Effects. In *AAAI 2020*.

Vidal, V. 2011. CPT4: An optimal temporal planner lost in a planning competition without optimal temporal track. *The Seventh International Planning Competition: Description of Participant Planners of the Deterministic Track*, 25–28.