Abstract Action Scheduling for Optimal Temporal Planning via OMT

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

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

Abstract

Given the model of a system with explicit temporal constraints, optimal temporal planning is the problem of finding a schedule of actions that achieves a certain goal while optimizing an objective function. Recent approaches for optimal planning reduce the problem to a series of queries to an Optimization Modulo Theory (OMT) solver: each query encodes a bounded version of the problem, with additional *abstract actions* representing an over-approximation of the plans beyond the bound. This technique suffers from performance issues, mainly due to the looseness of the over-approximation, which can include many non-executable plans.

In this paper, we propose a refined abstraction for solving optimal temporal planning via OMT by introducing *abstract scheduling* constraints, which have a double purpose. First, they enforce a partial ordering of abstract actions based on mutual dependencies between them, which leads to a better makespan estimation and allows to prove optimality sooner. Second, they implicitly forbid circular self-enabling of abstract actions, which is a common cause of spurious models that severely affects performance in existing approaches. We prove the soundness and completeness of the resulting approach and empirically demonstrate its superiority with respect to the state of the art.

1 Introduction

Automated temporal planning is the problem of finding a course of actions achieving a desired goal condition starting from a known initial state, for a system involving actions with durations and subject to temporal constraints. This problem mixes classical automated planning and scheduling (generalizing both problems) and is relevant in all the domains where time is an important dimension and parallelism between activities is possible. Example applications are flexible manufacturing (Ruml, Do, and Fromherz 2005) and robotics (Ingrand and Ghallab 2017). Several approaches to tackle temporal planning have been devised over the years, using either heuristic search (e.g. TFD (Eyerich, Mattmüller, and Röger 2012), OPTIC (Benton, Coles, and Coles 2012) and TAMER (Valentini, Micheli, and Cimatti 2020)), partial-order planning (e.g. EUROPA (Frank and Jónsson 2003), VHPOP (Younes and Simmons 2003), PLATINUM (Umbrico et al. 2018) and FAPE (Dvorak et al. 2014)) and reduction to satisfiability (e.g. ITSAT (Rankooh and Ghassem-Sani 2015), LCP (Bit-Monnot 2018) and the encodings of Shin and Davis (Shin and Davis 2005a) and Rintanen (Rintanen 2017)).

Optimal Temporal Planning (OTP) is the problem of finding a valid plan that optimizes an objective function. Common objective functions that have been considered in the literature are makespan minimization, consisting in finding the plan of minimum duration, and action cost minimization, which amounts to associate a cost to each action and find the plan of minimum cumulated cost. The planning approaches mentioned above are focused on quickly finding good-quality plans without offering formal guarantees of eventually finding optimal plans and proving their optimality[1]. Very few papers tackle the OTP problem and all of them focus on very specific cost functions. Some authors (Vidal 2011; Haslum 2006) address a special sub-case of OTP called "conservative planning" (Smith and Weld 1999), which is significantly simpler than the general OTP case because actions having conflicting preconditions or effects are forbidden to overlap in time. For general OTP, only some admissible heuristics for makespan minimization have been designed (Haslum 2009; Brandao et al. 2022). Recently, Panjkovic and Micheli presented an approach for solving an expressive class of OTP problems (2023); the technique exploits Optimization Modulo Theory (OMT) (Sebastiani and Tomasi 2015), a generalization of Satisfiability Modulo Theory (SMT) (Barrett et al. 2009) with cost functions. The basic idea of the approach (reported in Section 3) is to reduce the OTP problem to a series of queries to an OMT solver. Each query encodes a bounded version of the problem, with an additional abstract step representing an over-approximation of states that are reachable beyond the bound. If there is an optimal model of the formula that does not use abstract elements, then a provably optimal plan can be extracted from such a model. Conceptually, the abstract step of the OMT encoding is the symbolic analogous to an admissible heuristic, providing an under-estimation of the true minimal cost to the goal.

In this paper, we tackle a very general (and non-

[1] An exception is the minimization of plan length, for which satisfiability reductions are optimal and admissible heuristics exist.

conservative) OTP model, also supporting numeric features, and we make two major contributions. First, we refine the abstraction used in (Panjkovic and Micheli 2023) by encoding *abstract action schedules*. Our improved encoding tackles two key inaccuracies of the over-approximation step, namely the self-enabling of *abstract actions* which are not truly executable in future steps, and a lack of ordering constraints between abstract actions which can largely underestimate the makespan of the optimal plan. This allows the planning procedure to terminate with much smaller bounds, making it far more efficient. Second, we experimentally show that our approach is superior to previous works on both IPC and industrial domains. This paper focuses on providing a novel abstract step encoding for an OTP via OMT schema, and is therefore the symbolic analogous to providing a novel admissible heuristic in the context of search-based planning.

2 Problem Definition

In this section, we formalize the Optimal Temporal Planning (OTP) problem that we consider. As customary in planning, we define the syntax of a grounded problem, where a finite set of fluents with known initial states can be changed by actions and a formula represents the goal states.

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.*
- *A is a set of actions a of the form $\langle pre_a, eff_a, dur_a\rangle$:*
  - *pre_a is a set of conditions (Boolean expressions) 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 (assignments to fluents of the form $v := e$, where v is a fluent and e is an appropriately typed expression) 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.*

In order to define the semantics of the problem, we define a time-triggered plan as follows.

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 $t_i \in \mathbb{R}_{\geq 0}$ is the starting time, $a_i \in A$ is the action to be started, $d_i \in \mathbb{R}_{\geq 0}$ is the action duration, and $t_i \leq t_{i+1}$.*

A time-triggered plan is valid if each action can be executed (all its conditions are satisfied) at the prescribed time and with the given duration, and if after the end of the last action all the goal conditions are reached. For the sake of brevity, we omit the formalization of this semantics that is analogous to the ones presented in (Fox and Long 2003) and (Gigante et al. 2022). We adopt the *ϵ-separation* semantics (as in PDDL), requiring mutually exclusive events to be separated by a known amount of time indicated as ϵ.

Also following (Gigante et al. 2022), we need the definition of self-overlapping, because our techniques will be proven sound and complete assuming a semantics without self-overlapping. Intuitively, we will assume that two instances of the same ground action cannot overlap in time.

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 $t_i \leq t_j < t_i + d_i$.*

We now define our quality metrics. First, the makespan of a plan is the time at which the last action terminates.

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.*

Second, the action-cost minimization consists in associating a positive, real-valued cost to each action in a plan and to minimize the sum of the costs of all the actions in the plan.

Definition 5. *Given a map from actions to real costs $c : A \to \mathbb{R}_{>0}$, 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.*

3 OMT-based Optimal Temporal Planning

We now provide the needed background on SMT and OMT and describe previous OMT-based approaches for planning.

SMT and OMT Satisfiability Modulo Theory (SMT) is the problem of determining whether a first-order formula ψ expressed in some theory T is satisfiable (Barrett et al. 2009). A *model* of ψ is an assignment to the free variables of ψ that evaluates the formula to true. Several theories are supported by current SMT solvers, such as arithmetics with real and integer numbers, and data structures like arrays and bit-vectors. Here we focus on the theory of Quantifier-Free Linear Real Arithmetic ($\mathcal{QF\_LRA}$), that supports Boolean and real variables, comparisons with linear expressions (e.g. $\sum_i a_i x_i \leq c$), and the Boolean logical operators ($\wedge$, $\vee$, $\neg$). Optimization Modulo Theory (OMT) is an extension of the SMT problem where we want to find a model for a given formula ψ that minimizes or maximizes an objective function, which is a term expressed in some theory. Several OMT solvers are available (Sebastiani and Trentin 2018; Bjørner, Phan, and Fleckenstein 2015).

OTP via OMT Recently, several approaches for optimal planning based on OMT have been proposed, both for the numerical (Leofante et al. 2020; Giunchiglia and Taccchella 2022) and temporal (Panjkovic and Micheli 2023) cases. In these approaches, the main idea is that the algorithm proceeds in steps, and at each iteration h it considers a bounded encoding of the problem of length $h - 1$, with an extra step h, called the *abstract step*, used to represent an over-approximation of all the states that are reachable beyond the considered bound. In a numeric planning encoding, a step corresponds to the application of a set of (instantaneous) actions. In temporal planning, where actions have explicit durations, each step has an associated time and corresponds to a set of events, which can either be the start or the end of an action. The encoded SMT formula is given to an OMT solver, which returns an optimal model for it, and if this model does not rely on the *abstract step*, the solution plan that corresponds to it is provably optimal. This method is very similar to other approaches that reduce planning to a series of queries to a SAT or SMT solver (Kautz and Selman

1992; Shin and Davis 2005b), but the addition of the *abstract step* is a key difference. An optimal model for a standard bounded encoding of a problem (without the abstract step), represents an optimal solution among all the possible solutions of that length, but it could be the case that there exists a longer, more optimal plan, which is not captured by the current bounded encoding. The purpose of the abstract step is to be able to represent, using an over-approximation, all the possible plans beyond the considered bound, so that if the OMT solver returns an optimal model that is within this bound, it can be proved that the plan extracted from it is optimal for any possible length. In the following, we summarize the main technical details of the encoding for optimal temporal planning (Panjkovic and Micheli 2023), which generalizes the purely numeric case of (Leofante et al. 2020).

The encoding defines the following variables: t_i, for each step $i \in \{0, \dots, h\}$, representing the time of step i; f_i, for each step i and each fluent f, denoting the value of f at step i; the Boolean variable a_i, for each action a and each step i, indicating whether a is started at step i; d_i^a, representing the duration of the instance of the action a that is started at step i; mod_f, for each fluent f, denoting whether f is *potentially* modified after the last concrete step $h-1$ in the over-approximation. We use the notation $[e]_i$, where e is an expression and i is a step, for the SMT formula obtained by substituting each fluent f appearing in e with f_i. We write $vars(e)$ for the set of fluents appearing in e.

The formula that is produced at each step h is Π_h^{opt}, which is a conjunction of several subformulas that encode the features of the problem, among which the actions $\left(\varphi_h^{\text{ACTIONS}}\right)$, the mod_f variables $\left(\varphi_h^{\text{MOD}}\right)$, the abstract goals $\left(\varphi_h^{\text{GOALS}}\right)$ and the density axiom $\left(\varphi_h^{\text{DENSITY}}\right)$.

The $\varphi_h^{\text{ACTIONS}}$ formula controls the execution of the actions, specifying that if an action a is started at step s, the conditions and effects of the action are applied properly and the duration constraints are enforced.

$$\bigwedge_{a \in A} \bigwedge_{s \in \{1,\dots,h\}} a_s \rightarrow \left(\varphi_{s,h}^{\text{pre}_{\vdash a}} \wedge \varphi_{s,h}^{\text{pre}_{\leftrightarrow a}} \wedge \varphi_{s,h}^{\text{pre}_{\dashv a}} \wedge \varphi_{s,h}^{\text{eff}_{\vdash a}} \wedge \varphi_{s,h}^{\text{eff}_{\dashv a}} \wedge \varphi_{s,h}^{\text{dur}_a}\right)$$

The encoding of the start preconditions of an action a, started at step s is $\varphi_{s,h}^{\text{pre}_{\vdash a}}$ defined as follows.

$$\bigwedge_{e \in \text{pre}_{\vdash a}} \left((t_s \leq t_{h-1} \rightarrow [e]_s) \wedge \left(t_s > t_{h-1} \rightarrow \left([e]_{h-1} \vee \bigvee_{f \in \text{vars}(e)} \text{mod}_f\right)\right)\right)$$

The formula states that if the condition e needs to be checked in a concrete step (up to $h-1$), it is enforced at that step $(t_s \leq t_{h-1} \rightarrow [e]_s)$; if instead it is scheduled after the last concrete step h–1 $(t_s > t_{h-1})$, it must either hold at step h–1 $([e]_{h-1})$, or be potentially modifiable in the future, meaning that one of the mod_f variables corresponding to fluents f appearing in the condition must be true $\left(\bigvee_{f \in \text{vars}(e)} \text{mod}_f\right)$. Other conditions and effects are handled similarly.

The mod_f variables are controlled by the φ_h^{MOD} formula:

$$\bigwedge_{f \in F} \left(\text{mod}_f \leftrightarrow \left(\bigvee_{a \in A} \bigvee_{s \in \{1,\dots,h\}} \bigvee_{f := e \in \text{eff}_{\dashv a}} a_s \wedge t_s + d_s^a \geq t_{h-1}\right) \vee \left(\bigvee_{a \in A} \bigvee_{f := e \in \text{eff}_{\vdash a}} a_h\right)\right).$$

The main intuition is that a variable is potentially modifiable in the future (in the over-approximation), if there is an effect on that variable that is scheduled to happen after the last concrete step $h-1$. The formula states that mod_f is true, for a fluent f, if and only if there is an end effect on f of an action a beyond the concrete part of the encoding $(a_s \wedge t_s + d_s^a \geq t_{h-1})$, or there is a start effect on f of an action a that is started at the abstract step (a_h).

In the over-approximation, each goal condition is satisfied if it is true in the last concrete step or if it can potentially become true in the future, because a variable inside the condition is touched by a future effect. This is expressed by the φ_h^{GOALS} formula: $\bigwedge_{e \in G} \left([e]_{h-1} \vee \bigvee_{f \in \text{vars}(e)} \text{mod}_f\right)$.

Finally, the formula φ_h^{DENS} forbids the solver from leaving *empty* concrete steps (where no action is started or ended), if it returns a model that does not satisfy the goals *concretely*, meaning that all the goal conditions are satisfied at step $h-1$ and no action is left running (this is expressed with a formula denoted by φ_h^{CG}).

$$\varphi_h^{\text{DENS}} : \neg\varphi_h^{\text{CG}} \rightarrow \bigwedge_{i \in \{1,\dots,h-1\}} \bigvee_{a \in A} \left(a_i \vee \bigvee_{s \in \{1,\dots,i-1\}} (a_s \wedge t_s + d_s^a = t_i)\right)$$

$$\varphi_h^{\text{CG}} : \bigwedge_{e \in G} [e]_{h-1} \wedge \bigwedge_{a \in A} \bigwedge_{s \in \{1,\dots,h-1\}} (a_s \rightarrow t_s + d_s^a < t_{h-1})$$

The density axiom is essential to guarantee the termination of the overall algorithm: if the solver keeps returning solutions which rely on the *abstract step*, the requirement to fill all concrete steps leads to an increase of the value of the objective function, which eventually surpasses the value of the optimal solution, forcing the solver to return it.

With this encoding, it is possible to perform both makespan and total action cost minimization. When minimizing makespan, the objective function that is specified is t_{h-1}, while for total action cost the objective is $\sum_{a \in A} \sum_{i=1}^{h} \text{ITE}(a_i, c(a), 0)$, where $c(a)$ denotes the cost of action a and $\text{ITE}(a_i, c(a), 0)$ is an if-then-else term (supported by OMT solvers) that represents the cost of a if it is started (a_i true), otherwise 0.

Algorithm 1 reports the overall procedure. At each step h, if the OMT solver determines that Π_h^{opt} is unsatisfiable, it is possible to conclude that the problem Π is unsolvable, as the over-approximation encoded in Π_h^{opt} captures all the states that are reachable beyond the bound. If the solver returns an optimal model μ for Π_h^{opt}, then if μ is a valid solution for Π that does not rely on the abstract step to satisfy the goals $(\mu \models \varphi_h^{\text{CG}})$ an optimal solution plan can be extracted, otherwise the bound is increased and the procedure is repeated.

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 ⊨ φ_h^CG then
9                  return EXTRACTPLAN(μ)
10             else h ← h + 1
```

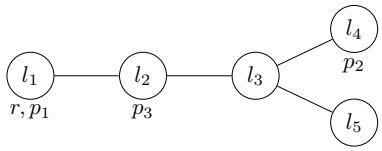

Figure 1: Pictorial representation of the running example.

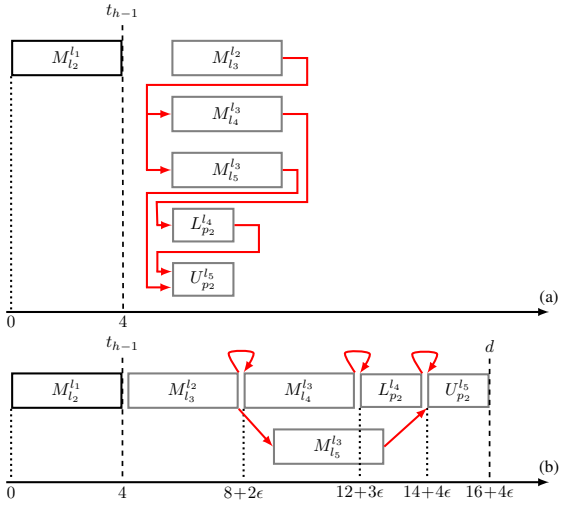

Figure 2: GANTTs of old (a) and new (b) OMT models.

4 Scheduling of Abstract Actions

The abstract step approximation of Section 3 has several sources of inaccuracy that can severely affect performance, especially for the case of makespan optimization in temporal planning. First, by specifying t_{h-1} as the minimization objective, anything that is applied in the abstract step has no impact on the overall cost. This fact can be "abused" by the OMT solver, which can fill the concrete steps with short actions, and perform all the actions that are necessary for reaching the goal at the abstract step, without paying any cost for them. Second, even if we considered the ending times of abstract actions in the objective (assuming they are started immediately after t_{h-1}), the solver could still parallelize all of them, and we would only increment the objective by the largest duration of an abstract action. Third, the encoding may allow disjoint groups of abstract actions to "enable" themselves, producing spurious models corresponding to plans that are not executable. These inaccuracies affect both the optimization of the makespan and the cumulative action cost. The abstraction allows the solver to return low-cost spurious models, which makes it necessary to reach very high bounds in the encoding in order to prove the optimality of a solution.

Example. *To illustrate these points, consider the example depicted in Figure 1. There are 5 locations $l_1, \ldots, l_5$, a robot r which is initially in location l_1, and 3 packages p_1, p_2, p_3 which are initially in locations l_1, l_4 and l_2, respectively. The actions available to the robot are: $M^{l_f}_{l_t}$ with a duration of 4 time units for moving from l_f to l_t (the two locations must be adjacent in the figure and the robot must be in l_f, as a result the robot will be in l_t); L^l_p with duration 2 for loading the package p in location l on the robot (both the package and the robot must be in location l, as a result the package will be onboard the robot); and U^l_p with duration 2 for unloading the package p on the robot in location l (the robot must be in location l and the package must be onboard the robot, as a result the package will be in l and no longer on the robot).*

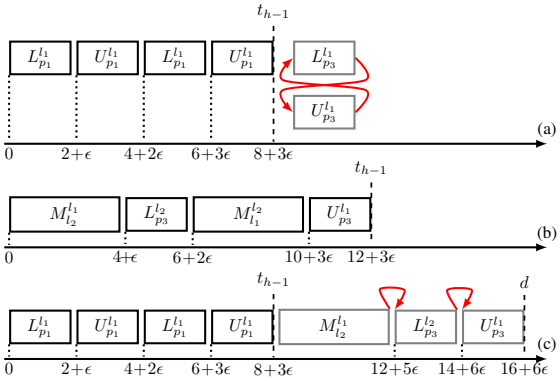

Figure 3: Witness models for removal of self-enabling loops.

Suppose the goal is to have package p_2 in location l_5, minimizing makespan. The GANTT in Figure 2a represents the optimal solution returned by the OMT solver at step $h = 3$. In the first two concrete steps, the action $M^{l_1}_{l_2}$ is started and ended. In the abstract step, the solver applies all the actions that are necessary for reaching the goal in the over-approximation: the action $M^{l_2}_{l_3}$ enables the actions $M^{l_3}_{l_4}$ and $M^{l_3}_{l_5}$, which in turn enable the loading of the package ($L^{l_4}_{p_2}$), and the unloading at the desired location ($U^{l_5}_{p_2}$), which satisfies the goal condition. The OMT objective is to minimize t_{h-1}, hence the cost of this model is 4. This does not consider the time spent to perform the abstract actions, nor whether they must be applied sequentially due to the causal relationships between them. The optimal valid plan has makespan 24, but since the abstraction is so loose, the algorithm is required to reach a very high step h in order to produce an encoding whose optimal model has cost 24.

Suppose now the goal is to have package p_3 in location l_1. Figure 3a shows the optimal solution returned by the solver at step $h = 9$. In the concrete part of the encoding, the robot loads and unloads the package p_1 twice, without moving. In the abstract step, only the actions $L^{l_1}_{p_3}$ and $U^{l_1}_{p_3}$ are applied, satisfying the goal condition in the over-approximation. This model shows a problematic behavior of the encoding: the package p_3 is never retrieved from l_2, but the combination of $L^{l_1}_{p_3}$ and $U^{l_1}_{p_3}$ can make it "appear" in l_1, because in the abstraction $U^{l_1}_{p_3}$ is enabled by $L^{l_1}_{p_3}$, which places the package on the robot, and $L^{l_1}_{p_3}$ is enabled by $U^{l_1}_{p_3}$, which places the package on the same location of the robot. With this loop of self-enabling actions, the solver can effectively "materialize" any package at the current location of the robot in the abstract step, and can fill the concrete part of the encoding with the cheapest available actions, achieving cost 8. The

optimal valid plan with makespan 12 could be encoded at the current step (Figure 3b), but since the abstraction allows such spurious models to appear, the algorithm needs more steps to return the optimal plan.

Our goal is to obtain a tighter over-approximation in the abstract step preventing these issues: we achieve this by introducing *abstract scheduling* constraints, and setting a new objective function for the case of makespan minimization.

The main intuition is that for each future condition that is false at the last *concrete* step, we require that it is preceded in time by an enabling future effect, i.e. an effect acting on a fluent in the condition. In this way, we obtain a more refined abstraction that considers how the future actions are arranged in time, based on the dependencies between them.

Encoding description For each action $a \in A$, we introduce a variable t_h^a representing the time in which a is started in the abstract step.

We will use the following notation. Given an event $ev \in \text{pre}_a \cup \text{eff}_a$ of an action a, we use t_s^{ev} to denote the timing of ev when a is started at step s. Thus, $t_s^{ev} = t_s$ if ev is a start condition, an overall condition or a start effect ($t_s^{ev} = t_h^a$ when $s = h$), and $t_s^{ev} = t_s + d_s^a$ if ev is an end condition or end effect ($t_s^{ev} = t_h^a + d_h^a$ when $s = h$).

Formula φ_h^{SCHED} (Figure 4) is a conjunction over all actions $a \in A$ and all steps $s \in \{1, \dots, h\}$. For each precondition φ of a, if it does not hold at the last *concrete* step $h-1$ $\left(\neg [\varphi]_{h-1}\right)$ and its timing is greater than t_{h-1} ($t_s^{\varphi} > t_{h-1}$), there must exist a variable v of φ modified (mod_v is true) by an effect of an action b started at step k (b_k is true), such that the effect is applied after step h–1 and before the condition is checked ($t_{h-1} \leq t_k^{v:=e} < t_s^{\varphi}$). All the abstract starting times must follow the time of the last *concrete* step ($t_{h-1} + \epsilon \leq t_h^a$).

With the additional variables representing the starting times of abstract actions, we can formulate a new objective for makespan minimization that takes into account the ending times of actions beyond t_{h-1}. We introduce a variable t_{obj} representing the makespan of the plan, and add the following lower bounds to it:

- the time of the last concrete step ($t_{obj} \geq t_{h-1}$);
- the ending times of all the actions started in the concrete part of the encoding ($a_s \rightarrow t_{obj} \geq t_s + d_s^a$);
- the ending times of all the actions started at the abstract step ($a_h \rightarrow t_{obj} \geq t_h^a + d_h^a$).

The minimum value of t_{obj} that satisfies these lower bounds will be the last ending time of an action in the plan, i.e. the makespan of the plan. We can now specify t_{obj} as the objective to be minimized for the case of makespan optimization. Note that also the minimization of cumulative action cost benefits from the superior precision of φ_h^{SCHED}, as will be explained in the following paragraphs.

We will now show that the inclusion of φ_h^{SCHED} in the encoding preserves the *soundness* and *completeness* of the overall approach. We consider the same OMTPLAN procedure, but the encoding that is produced at each step is $\Pi_h^{opt} \wedge \varphi_h^{\text{SCHED}}$, and we use t_{obj} as the objective to be minimized for the case of makespan optimization. Here we provide proof sketches for the theorems, while the full details are included in (Panjkovic and Micheli 2024).

Theorem 1 (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) Consider the case of makespan optimization (the case of action cost optimization is proved analogously). Let d be the makespan of π, and let h be the step at which π was returned by OMTPLAN. Suppose, for the sake of contradiction, that there exists a plan π' with a lower makespan $d' < d$. If π' can be found at step $h' \leq h$, then the OMT solver would have returned a model corresponding to that plan. If instead $h' > h$, then a *relaxed* version π'' of π' can be extracted from a model of $\Pi_h^{\text{opt}} \wedge \varphi_h^{\text{SCHED}}$ with makespan $d'' < d' < d$, which leads to a contradiction because the OMT solver returned π with makespan d at step h. The *relaxed* plan π'' can be obtained from π' by starting the same actions up to step $h-1$, and then applying an abstract action for each action a that is left, with the same starting time t_h^a as in π'. If an action is applied more than once beyond step $h-1$ in π', we apply it only once in π'' with the first starting time in which it occurs in π' after step $h-1$. These starting times satisfy the timing constraints in φ_h^{SCHED}, because each condition beyond step $h-1$ that is false at $h-1$ is preceded by at least one effect that makes it true in π', and this effect will have the same timing (or lower, if repeated) in π''. □

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. (Sketch) Completeness is a consequence of the *density axiom* φ_h^{DENS}, which forces the solver to fill all concrete steps with an action start or end if it uses the abstract step to achieve the goal. If the solver returns solutions with abstract actions at each step, the value of the objective increases until it surpasses the optimal value, at which point the OMT solver returns the *concrete* optimal plan. If the solver returns a *concrete* plan before this point, it is also an optimal plan by the soundness theorem. □

Discussion The addition of the *abstract scheduling* constraints results in a more refined abstraction, because they capture the causal and temporal relations between abstract events. The encoding described in Section 3 does not have a notion of time beyond the concrete part of the encoding, and the causality relation between effects and conditions is not ordered. Therefore, in a model of the formula, the abstract step contains an unordered set of actions, whose conditions are either true in the last step, or are touched by an effect of an action appearing in the set (see $\varphi_{s,h}^{\text{pre}_{\vdash a}}$ and φ_h^{MOD} in Section 3). With the φ_h^{SCHED} formula, instead, the abstract step becomes a schedule of actions, where conditions are required to appear after the first effect that affects them.

Figure 2b shows the optimal solution returned by the solver with the new encoding at step $h = 3$. The abstract actions are now scheduled in time, and the causal relationships are used to apply some actions in sequence rather than in parallel, leading to a better makespan estimation (e.g. $M_{l_4}^{l_3}$ is applied after $M_{l_3}^{l_2}$, because the start precondition of $M_{l_4}^{l_3}$ that

$$\bigwedge_{a\in A}\bigwedge_{s\in\{1,\dots,h\}}\left(a_s \rightarrow \bigwedge_{\varphi\in \text{pre}_a}\left(\left(\neg\left[\varphi\right]_{h-1}\wedge t_s^{\varphi} > t_{h-1}\right)\rightarrow \bigvee_{v\in\text{vars}(\varphi)}\left(\text{mod}_v \wedge \left(\bigvee_{b\in A}\bigvee_{k\in\{1,\dots,h\}}\bigvee_{v:=e\in\text{eff}_b} b_k \wedge t_{h-1} \le t_k^{v:=e} < t_s^{\varphi}\right)\right)\right)\right) \wedge \bigwedge_{a\in A} t_{h-1}+\epsilon \le t_h^a$$

Figure 4: The φ_h^{SCHED} formula

the robot must be in l_3 is enabled by the end effect of $M_{l_3}^{l_2}$). Note that the abstract step is still an over-approximation, because we only require for each abstract condition to be "touched" by an effect. For example, in Figure 2b there is no $M_{l_3}^{l_4}$ action once p_2 is loaded, and $M_{l_5}^{l_3}$ can appear as soon as $M_{l_3}^{l_2}$ ends. Still, the makespan estimation is much more precise than before (16 compared to 4), and in practice this means that the optimal solution can be found with fewer steps, as shown in the experimental evaluation.

An additional advantage of the *abstract scheduling* constraints is that they prevent models where the abstract actions are self-enabling (see Figure 3a). The issue of loops of self-enabling actions has already been identified in (Leofante et al. 2020) for the case of optimal numeric planning via OMT. They handled the problem by introducing *loop formulas*, which require that for each loop of self-enabling actions there is an action external to the loop that enables an action inside the loop. However, the number of additional constraints may be exponential in the size of the problem. Level ordering constraints are introduced in (Giunchiglia and Taccchella 2022), for the case of numeric planning, which disallow these loops and are also polynomial in size. Our encoding of the abstract step breaks the self-enabling loops polynomially as a side-effect of the abstract times that we introduce, without "artificial" variables or constraints. Consider the situation in Figure 3a and let $(t_h^{L_{p_3}^{l_1}}, d_h^{L_{p_3}^{l_1}})$ and $(t_h^{U_{p_3}^{l_1}}, d_h^{U_{p_3}^{l_1}})$ denote the abstract starting times and durations of the actions $L_{p_3}^{l_1}$ and $U_{p_3}^{l_1}$ respectively. If both actions are started at the abstract step, the φ_h^{SCHED} constraints require that the start preconditions of both actions are preceded in time by an effect which enables them. For both actions, their start condition regarding the position of p_3 can only be enabled by the end effect of the other action, so the φ_h^{SCHED} formula will enforce the constraints: $t_h^{U_{p_3}^{l_1}} + d_h^{U_{p_3}^{l_1}} < t_h^{L_{p_3}^{l_1}}$ and $t_h^{L_{p_3}^{l_1}} + d_h^{L_{p_3}^{l_1}} < t_h^{U_{p_3}^{l_1}}$. This leads to a contradiction:

$$t_h^{L_{p_3}^{l_1}} \le t_h^{L_{p_3}^{l_1}} + d_h^{L_{p_3}^{l_1}} < t_h^{U_{p_3}^{l_1}} \le t_h^{U_{p_3}^{l_1}} + d_h^{U_{p_3}^{l_1}} < t_h^{L_{p_3}^{l_1}}$$

Therefore, the self-enabling of these two actions is disallowed by the new encoding, and the algorithm actually terminates at step $h = 9$ and returns the optimal plan shown in Figure 3b. Note that the new encoding does not select the concrete prefix of Figure 3a, because it becomes more costly than the optimal plan, as shown in Figure 3c.

In general, φ_h^{SCHED} disallows loops of abstract actions that support each other's conditions with their effects, without having *external* enabling actions, or actions inside the loop which are applicable at $h - 1$: the precedence constraints between the conditions and effects of these actions identify a sequence of timings which are strictly decreasing, leading to a contradiction when a loop is considered.

Finally, we highlight that abstract action scheduling is also helpful for action cost minimization. Consider Figure 3 pretending action costs are set equal to the durations. The OMT model (a) would cost 12, but is forbidden by the new encoding; (b) costs 12 and is the optimal plan; while (c) costs 16. With the same reasoning as above, the new algorithm would terminate with the correct plan at step 9.

5 Experiments

Implementation and setup We implemented the presented approaches in C++ using the Z3 OMT solver (Bjørner, Phan, and Fleckenstein 2015). Analogously to the approach in (Panjkovic and Micheli 2023), we implemented an incremental (indicated as $\text{Z3}_{\text{INC}}^{\text{OUR}}$) and a monolithic (indicated as Z3^{OUR}) version of the encoding. We compare against the solvers presented in (Panjkovic and Micheli 2023) (indicated as $\text{Z3}_{\text{INC}}^{\text{PM}}$and Z3^{PM} for the incremental and monolithic versions, respectively) and against the OPTIC planner (Benton, Coles, and Coles 2012).

Our implementation embodies two technical optimizations. The first improvement avoids the cost of optimization for bounds where no plan exists, relying on a classical SATPlan-like encoding instead, and then switching to the full OMT encoding after the first plan is found. This has the positive side-effect of generating a valid plan before starting the optimization search. We experimented with a version of our solver without this optimization and performed an ablation study; the solvers without this feature are indicated as $\text{Z3}_{\text{INC}}^{\text{OO}}$and Z3^{OO} (OO means "optimization only") for the incremental and monolithic cases, respectively. The second technical optimization augments the objective criterion passed to the OMT solver with a secondary objective, consisting of minimizing the number of mod_f variables set to true. OMT solvers such as Z3 support *lexicographic optimization*, that is finding models that minimize a primary objective function and among these return a model that minimizes a secondary objective. In this way, we force the OMT-PLAN algorithm to terminate as soon as possible if multiple equi-optimal models are present. Without this lexicographic constraint, the solver may be free to choose between a model using abstract "mod" variables and a concretizable model. We indicate with $\text{Z3}_{\text{INC}}^{\text{NS}}$and Z3^{NS} (NS means "no-secondary") the planners without this optimization using incrementality or being monolithic, respectively.

We experiment on Temporal IPC domains from the IPC-14 competition, and consider two versions for each domain, one with makespan and the other with action cost minimization. We also included the benchmark set that was used in (Panjkovic and Micheli 2023). We executed all the experiments on a cluster of identical machines equipped with Xeon E5-2440 2.4GHz and running Ubuntu Linux 20.04. We used a timeout of 3600s and a memory limit of 20GB.

Domain	OPTIC	$Z3^{PM}$	$Z3^{PM}_{INC}$	$Z3^{OO}$	$Z3^{OO}_{INC}$	$Z3^{NS}$	$Z3^{NS}_{INC}$	$Z3^{OUR}$	$Z3^{OUR}_{INC}$
DriverlogActionCost	0(11)	0	0	**1**	**1**	**1(7)**	**1(5)**	**1(7)**	**1(5)**
DriverlogMakespan	**1(15)**	0	0	0	0	**1(7)**	**1(5)**	**1(7)**	**1(5)**
FloortileActionCost	**10(14)**	1	1	3	3	3(11)	3(7)	3(11)	3(7)
FloortileMakespan	**5(20)**	1	1	2	1	2(11)	2(7)	3(11)	2(7)
Majsp	0(10)	0	0	**23**	**23**	22(76)	**23(75)**	**23(76)**	**23(74)**
MajspSimplified	0(10)	12	12	19	18	**20(76)**	19(72)	19(76)	**20(72)**
MatchActionCost	8(40)	19	19	**23**	**23**	**23(40)**	**23(40)**	**23(40)**	**23(40)**
MatchCellarActionCost	2(9)	3	**4**	3	3	3(6)	**4(6)**	**4(6)**	**4(6)**
MatchCellarMakespan	**3(9)**	**3**	**3**	**3**	**3**	**3(6)**	**3(6)**	**3(6)**	**3(6)**
MatchMakespan	**37(40)**	0	0	25	25	25(40)	25(40)	25(40)	25(40)
OptionalGoals	10(30)	25	12	**30**	**30**	**30(30)**	**30(30)**	**30(30)**	**30(30)**
Painter	0(4)	**14**	13	11	12	13(16)	11(13)	12(16)	11(13)
ParkingActionCost	0(19)	0	0	0	0	0(19)	0(8)	0(19)	0(8)
ParkingMakespan	**13(16)**	0	0	0	0	0(19)	0(8)	0(20)	0(8)
SatelliteActionCost	**1(15)**	0	0	0	0	**1(6)**	0(3)	**1(6)**	**1(3)**
SatelliteMakespan	0(15)	0	0	0	0	0(6)	0(3)	0(6)	0(3)
TurnAndOpenActionCost	0(10)	0	0	0	0	0(0)	0(0)	0(0)	0(0)
TurnAndOpenMakespan	0(10)	0	0	0	0	0(0)	0(0)	0(0)	0(0)
Total	90(297)	78	65	143	142	147(376)	145(328)	**148(377)**	147(327)

Table 1: Coverage table. For capable solvers, we write X(Y) indicating that X instances have been optimally solved and for Y instances only plans not proven optimal have been returned within the experimental limits.

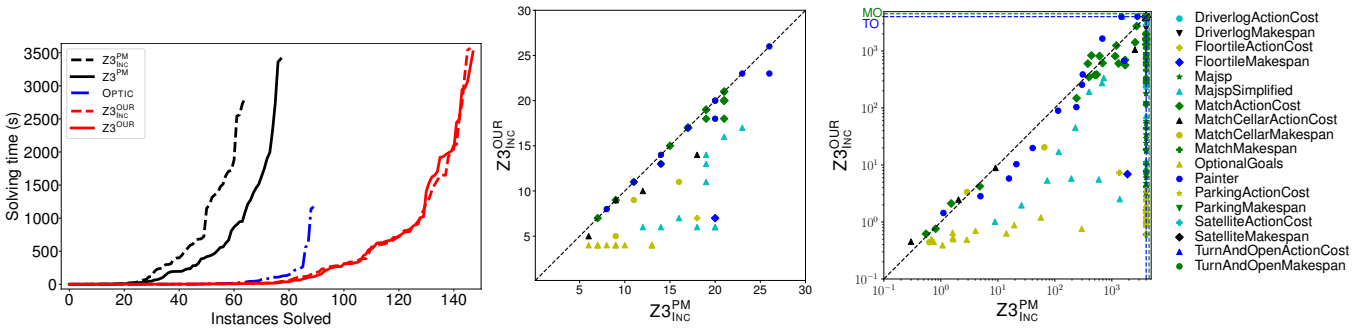

Figure 5: Cactus plot (left) of the cumulated solving time of the solvers compared with solved instances, sorted by time. Scatter plots for $Z3^{OUR}_{INC}$ vs $Z3^{PM}_{INC}$. Scatter plot comparison of runtime (right) and terminating bound (center) for $Z3^{OUR}_{INC}$ vs $Z3^{PM}_{INC}$. MO and TO denote memory out and time out cases respectively.

The solver and benchmarks are available in (Panjkovic and Micheli 2024).

Results Table 1 reports the coverage results for the considered approaches and Figure 5 (left) shows a cactus plot of the experiments (excluding the ablation study solvers to avoid cluttering the plot). It is evident that, while OPTIC performs quite well on some IPC benchmarks, the presented approaches perform very well in general and are consistently superior to the encoding of (Panjkovic and Micheli 2023). We regard this as a very strong result, because satisfiability-based planners are notoriously very weak on IPC instances compared to heuristic-search approaches, like OPTIC . The ablation study results show that the two "technical" improvements described above are useful, but that the vast majority of the speedup is gained because of the abstract scheduling encoding. The scatter plot in the center of Figure 5 compares the bound at which the optimal plan is proven by $Z3^{OUR}_{INC}$ and $Z3^{PM}_{INC}$. It is evident that the new encoding never worsens the bound and in many cases it improves it significantly; this is then reflected in the runtime of the whole procedure (which heavily depends on the bound) as shown by the scatter plot on the right in Figure 5.

6 Conclusions

In this paper, we presented a novel encoding of the abstract step for Optimal Temporal Planning via OMT, which is the symbolic analogous of devising a new admissible heuristic for optimal planning in a search-based approach. Our abstract encoding is much more precise, allows to find optimal plans with fewer encoding step, and we formally proved its soundness and completeness. We empirically demonstrated the effectiveness of the approach compared to previous OMT encodings and state-of-the-art tools, and we also performed a detailed ablation study.

As future work, we will investigate the use of the abstract encoding as an informative heuristic in a search schema, and will compare the performance of OMT solvers with Mixed Integer Linear Programs solvers on our encoding.

Acknowledgments

This work has been partly supported by the project "AI@TN" funded by the Autonomous Province of Trento and by the PNRR projects "FAIR" – Future AI Research (PE00000013) and iNEST – Interconnected Nord-Est Innovation Ecosystem (ECS00000043) funded by the European Union NextGenerationEU program.

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*.

Bit-Monnot, A. 2018. A Constraint-Based Encoding for Domain-Independent Temporal Planning. In *CP 2018*, 30–46.

Bjørner, N. S.; Phan, A.; and Fleckenstein, L. 2015. νZ - An Optimizing SMT Solver. In Baier, C.; and Tinelli, C., eds., *TACAS*, 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.

Dvorak, F.; Barták, R.; Bit-Monnot, A.; Ingrand, F.; and Ghallab, M. 2014. Planning and Acting with Temporal and Hierarchical Decomposition Models. In *ICTAI 2014*.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2012. Using the Context-Enhanced Additive Heuristic for Temporal and Numeric Planning. In Prassler, E.; Zöllner, J. M.; Bischoff, R.; Burgard, W.; Haschke, R.; Hägele, M.; Lawitzky, G.; Nebel, B.; Plöger, P.; and Reiser, U., eds., *Towards Service Robots for Everyday Environments*, volume 76 of *Springer Tracts in Advanced Robotics*, 49–64. Springer.

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

Frank, J.; and Jónsson, A. 2003. Constraint-based attribute and interval planning. *Constraints*.

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.

Giunchiglia, E.; and Taccchella, A. 2022. Optimal Planning as Constraint Optimization. In *ICAPS 2022 Workshop on Heuristics and Search for Domain-independent Planning*.

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., *ECAI*, 359–363. John Wiley and Sons.

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.

Panjkovic, S.; and Micheli, A. 2023. Expressive Optimal Temporal Planning via Optimization Modulo Theory. In *AAAI 2023 Proceedings*. AAAI Press.

Panjkovic, S.; and Micheli, A. 2024. https://es-static.fbk.eu/people/panjkovic/aaai24/.

Rankooh, M. F.; and Ghassem-Sani, G. 2015. ITSAT: an efficient sat-based temporal planner. *Journal of Artificial Intelligence Research*.

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. J. 2005. Online Planning and Scheduling for High-speed Manufacturing. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), June 5-10 2005, Monterey, California, USA*, 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*.

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

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

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

Umbrico, A.; Cesta, A.; Mayer, M. C.; and Orlandini, A. 2018. Integrating Resource Management and Timeline-Based Planning. In *ICAPS*.

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.

Younes, H. L.; and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research*.