Compiling Temporal Numeric Planning into Discrete PDDL+

Andrea Micheli[1], **Enrico Scala**[2], **Alessandro Valentini**[1]

[1]Fondazione Bruno Kessler, Trento, Italy
[2]University of Brescia, Italy
amicheli@fbk.eu, enrico.scala@unibs.it, alvalentini@fbk.eu

Abstract

Since the introduction of the PDDL+ modeling language, it was known that temporal planning with durative actions (as in PDDL 2.1) could be compiled into PDDL+. However, no practical compilation was presented in the literature ever since. We present a practical compilation from temporal planning with durative actions into PDDL+, fully capturing the semantics and only assuming the non-self-overlapping of actions. Our compilation is polynomial, retains the plan length up to a constant factor and is experimentally shown to be of practical relevance for hard temporal numeric problems.

Introduction

Automated planning is the task of finding a course of actions to achieve a goal from an initial state given a model of the system specifying which actions are available, together with (i) their precondition, what needs to hold in order for an action to be applicable and (ii) their effect, what needs to hold when such actions are applied. Temporal planning is the extension in which actions are assumed to last for some interval of time, and so we look for a plan that is schedulable too, i.e., actions need to be done in specific points over a potentially unbounded timeline, with conditions required to hold at the beginning, at the end and during the execution of the action. Temporal planning problems can be compactly represented in the PDDL 2.1 language (Fox and Long 2003).

A number of solutions have been proposed to handle PDDL2.1 problems, ranging from forward heuristic search in the space of possible schedules (Coles et al. 2010; Benton, Coles, and Coles 2012; Valentini, Micheli, and Cimatti 2020) to satisfiability-based bounded reductions (Cardellini and Giunchiglia 2025). Among these approaches, compilation-based techniques are particularly appealing, as they allow temporal reasoning to be delegated to more expressive or better-supported target languages. Building on this line of work, this paper studies a compilation method that transforms a temporal planning problem specified in PDDL2.1 into an equivalent formulation in PDDL+.

PDDL+ is yet another extension of classical planning which provides a different take to the problem of representing timed and hybrid systems. Instead of having durative actions, in PDDL+ a system over time is modeled through a combination of processes, events and instantaneous actions. Processes model the system evolving over time through differential equations, and events dictate what needs to change instantaneously if some condition is satisfied.

Our compilation is rooted in the known observation that a durative action can be compiled into a combination of processes and events/actions (Fox and Long 2006), but we contribute the first fully spelled-out formal account to approach this problem rigorously. Moreover, the expressiveness of PDDL+ makes it easier to extend the input problems with a number of features that are not well supported by many temporal planners, such as numeric state variables, delayed effects and timed initial literals. Our compilation provides a comprehensive account for the full semantics of temporal planning resulting in an encoding that is sound, complete and polynomial on the size of the input problem.

To shed some light on the practical benefit of this compilation we run an experimental campaign on a number of temporal numeric domains. Surprisingly, PDDL+ planners prove competitive, and often superior, to state-of-the-art temporal planners on rich numeric temporal problems. This may inform a more precise understanding of the core difficulties in temporal planning.

Background

We start by defining a temporal planning problem, adapting the PDDL 2.1 level 3 language (Fox and Long 2003).

Definition 1. *A **temporal planning problem** $\mathcal{P}^t$ is a tuple (F, X, I, A^i, A^d, G) where:*

- *F is a finite set of boolean fluents (predicates);*
- *X is a finite set of numeric rational fluents;*
- *$I : F \cup X \to \mathbb{B} \cup \mathbb{Q}$ is the initial state, assigning each fluent to its initial value;*
- *A^i is a finite set of instantaneous actions; each $a \in A^i$ has a precondition formula pre_a over $F \cup X$ and a set of boolean and numeric effects eff_a of the form $f := \{\bot, \top\}$ if $f \in F$ or $f \leftarrow e$ with $\leftarrow \in \{:=, +=\}$ and e being a numeric expression over X, if $f \in X$.*
- *A^d is a finite set of durative actions; each $a \in A^d$ has lower and upper duration bounds $l_a \leq u_a \in \mathbb{Q}_{>0}$, a pair of starting $a_\vdash$ and ending $a_\dashv$ instantaneous "snap" actions, and an overall invariant formula γ_a over $F \cup X$.*
- *G is the goal expressed as a formula over $F \cup X$.*

We partition A^d into $A^{fix} \sqcup A^{var}$, with $A^{fix} = \{a \in A^d \mid l_a = u_a\}$ being the set of durative actions with a fixed duration (therefore, A^{var} is the set of durative actions with variable duration). Moreover, given a (snap) instantaneous action a, we write: $V_a^{pre} \stackrel{\text{def}}{=} vars(pre_a)$ for the set of fluents occurring in pre_a; $V_a^r \stackrel{\text{def}}{=} V_a^{pre} \cup \bigcup_{(f \leftarrow e) \in \mathit{eff}_a} vars(e)$ for the fluents read by the precondition or by any effect of a; $V_a^{:=} \stackrel{\text{def}}{=} \bigcup_{(f:=e) \in \mathit{eff}_a} \{f\}$ for the assigned fluents; $V_a^{+=} \stackrel{\text{def}}{=} \bigcup_{(f+=e) \in \mathit{eff}_a} \{f\}$ for the increased fluents; and $V_a^w \stackrel{\text{def}}{=} V_a^{:=} \cup V_a^{+=}$ for the effected (written) fluents.

Definition 2. *A temporal plan $\pi^t$is a finite set of triples of the form (t, a, d), where $t \in \mathbb{Q}_{\geq 0}$ is the starting time, $a \in A^i \cup A^d$ is the action to execute and $d \in \mathbb{Q}_{\geq 0}$ is the action duration, s.t. $l_a \leq d \leq u_a$ if $a \in A^d$, and $d = 0$ if $a \in A^i$.*

We present an adaptation of the non-self-overlapping semantics by Gigante et al. (2022). A state is a total assignment of values to fluents; given a formula e defined over $F \cup X$, we write $s(e)$ for the value of e in s obtained by substitution and constant propagation. Given a state s and an instantaneous action a, a is applicable in s if $s(pre_a)$ is true ($s \models pre_a$) and the successor state $s' \stackrel{\text{def}}{=} a(s)$ is such that $s'(f) = s(e)$ if $f := e \in \mathit{eff}_a$, $s'(f) = s(f) + s(e)$ if $f \mathrel{+}= e \in \mathit{eff}_a$, $s'(f) = s(f)$ otherwise.

Let $\pi^t \stackrel{\text{def}}{=} \{(t_1, a_1, d_1), \ldots, (t_n, a_n, d_n)\}$ be a temporal plan for a planning problem $\mathcal{P}^t = (F, X, I, A^i, A^d, G)$. Let H^{π^t} be the set of timed (snap) actions for π^t defined as $\{(t_i, a_i) \mid a_i \in A^i\} \cup \{(t_i, a_{i,\vdash}) \mid a_i \in A^d\} \cup \{(t_i + d_i, a_{i,\dashv}) \mid a_i \in A^d\}$. Let $t_0^s, t_1^s, \ldots, t_{m-1}^s \in \mathbb{Q}$ be the times appearing in H^{π^t} (that is, $\exists (t, a) \in H^{\pi^t}. t_j^s = t$) ordered s.t. $t_j^s < t_{j+1}^s$; moreover, we add an arbitrary final time: $t_m^s \stackrel{\text{def}}{=} t_{m-1}^s + 1$. We define the set of "happenings" at step j as the set $H_j = \{a \mid t_j^s = t \text{ and } (a, t) \in H^{\pi^t}\}$. We can now give the semantics of temporal planning: the plan π^t is valid if there exists a sequence of states $s_0, \ldots, s_m$ such that:

1. $s_0 = I$ and $s_m \models G$ (initial state and goal constraints),
2. for each $0 \leq j < m$ and each $a \in H_j$, $s_j \models pre_a$ (all conditions for happenings at step j are satisfied);
3. for each $0 \leq j < m$, $s_{j+1} = b_0(b_1(\cdots b_k(s_j)))$, with $H_j = \{b_0, b_1, \ldots, b_k\}$ for an arbitrary ordering (a state is the result of applying all the effects of all happenings);
4. for each $1 \leq i \leq n$ with $a_i \in A^d$, if $t_j^s = t_i$ and $t_k^s = t_i + d_i$, $s_w \models \gamma(a_i)$ for all $j < w \leq k$ (overall conditions);
5. for each $0 \leq j < m$ and each pair of instantaneous actions $a \neq b \in H_j$, $V_a^r \cap V_b^w = V_b^r \cap V_a^w = V_a^{:=} \cap V_b^{:=} = \emptyset$ (no interfering actions at the same time);
6. For all $j \neq k \in [1, n]$, $t_j > t_k + d_k$ or $t_k > t_j + d_j$ (no-self-overlapping constraint).

The target of our compilation is a PDDL+ (Fox and Long 2006) problem, formalized below.

Definition 3. *A **PDDL+ problem** is modeled as a tuple (F, X, I, G, A, E, P) where:*

- *F is a finite set of boolean fluents (predicates);*
- *X is a finite set of numeric rational fluents;*
- *$I : F \cup X \to \mathbb{B} \cup \mathbb{Q}$ is the initial state, assigning each fluent to its initial value;*
- *G is a goal condition expressed as a formula over $F \cup X$.*
- *A, E are two finite sets of instantaneous actions and events resp.; each action/event x is defined by a set of preconditions pre_x and a set of effects eff_x.*
- *P is a set of processes each $p \in P$ having a precondition formula pre_p over $F \cup X$ and a set of effects eff_p of the form $\frac{d}{dt} f \mathrel{+}= e$ with $f \in X$ and e is a formula over X.*

Definition 4. *A **PDDL+ plan** is a pair (π^+, t_e), where $t_e \in \mathbb{Q}$ is the plan makespan and π^+ is a finite sequence of pairs (t_i, a_i) with $t_i \leq t_e \in \mathbb{Q}_{\geq 0}$.*

Following Percassi, Scala, and Vallati (2025), we assume a discretization time quantum $\delta \in \mathbb{Q}$. A plan is well-formed if t_e and all times t_i are multiples of δ. A plan (π^+, t_e) is valid for a PDDL+ problem (F, X, I, G, A, E, P) if there exists a sequence of states (similarly to the temporal case) $\bar{s}_0, \ldots, \bar{s}_{\bar{m}}$ with $\bar{s}_0 = I$ and $\bar{m} = \frac{t_e}{\delta}$ defined as follows. Let $\bar{t}_j^s = \delta \cdot j$ be the time of state $\bar{s}_j$ and let the sequence of "action happenings" H_j^a at step j be the sequence of actions from π^+ happening at time $\bar{t}_j^s$; i.e., $H_j^a = (a_i, \ldots, a_{i+k})$ where $(t_i, a_i), \ldots, (t_{i+k}, a_{i+k})$ is a maximal sub-sequence of π^+ with $t_i = \cdots = t_{i+k} = \bar{t}_j^s$. Given a state $\bar{s}$, we inherit the definitions of action applicability from above and generalize them for events in the obvious way. We define the event completion of $\bar{s}$ (written $\bar{s}^{\rightarrow}$) as the fixed-point state reached after applying any applicable event in $\bar{s}$ and then any other applicable event in the resulting state, until no event is applicable anymore. For each j, we define the final state at step j as $\bar{s}_j^{end} \stackrel{\text{def}}{=} b_0(b_1(\cdots(b_k(\bar{s}_j^{\rightarrow})^{\rightarrow})^{\rightarrow})^{\rightarrow}$ with $H_j^a = (b_0, b_1, \ldots, b_k)$ (intuitively, we apply all actions ordered by π^+ to the event completion of $\bar{s}_j$ and after every action we perform an event completion). We can now define the state transition: for every $0 \leq j < \bar{m}$, we define $\bar{s}_{j+1}(f) = \bar{s}_j^{end}(f)$ for every $f \in F$, and for every $x \in X$:

$$\bar{s}_{j+1}(x) = \bar{s}_j^{end}(x) + \sum_{\substack{p \in P,\, (x+=e) \in \mathit{eff}_p \\ \bar{s}_j^{end} \models pre_p}} \bar{s}_j^{end}(e) \cdot \delta$$

Intuitively, we set every predicate to its final value at step j and compute the value of the numeric fluents after some passage of time by applying a discretized step aggregating the contribution of every active process. Finally, the plan is valid if $\bar{s}_m \models G$, and each $a_i \in H_j^a$ is applicable in $\bar{s}_j^{\rightarrow}$.

Compile Temporal Planning into PDDL+

In this section, we formally define our compilation from temporal planning into PDDL+. Intuitively, we formulate a PDDL+ problem where each durative action is emulated by a triplet (action, process, event) for fixed duration actions, and (action, process, action) for flexible duration ones. To only encode valid temporal plans, we need all temporal points from 1-6 (see previous section) to hold. We introduce a number of auxiliary boolean and numeric state variables, and use them to constrain actions properly, and propagate

processes and events when necessary. A key step is the introduction of lock preconditions-effects, a machinery employed to prevent interfering actions to happen at the same time. Below, we formalize the encoding.

In the following, we assume a temporal planning problem $\Pi \stackrel{\text{def}}{=} (F, X, I, A^i, A^d, G)$ is given, and we define the compiled PDDL+ problem $\bar{\Pi} \stackrel{\text{def}}{=} (\bar{F}, \bar{X}, \bar{I}, \bar{G}, \bar{A}, \bar{E}, \bar{P})$.
Let $FX \stackrel{\text{def}}{=} F \cup X$; we start with the fluents definition.

$$\bar{F} \stackrel{\text{def}}{=} F \cup \{ok\} \cup \{r_a \mid a \in A^d\} \cup \{rl_f, al_f, il_f \mid f \in FX\}$$
$$\bar{X} \stackrel{\text{def}}{=} X \cup \{oc, gc\} \cup \{c_a \mid a \in A^d\}$$

In addition to the fluents of Π, we add a new ok predicate that will be required by every action in the compiled model and by the new goal, this will be used to "abort" a plan that violated some constraints. We also add two numeric fluents, oc and gc, to count the number of actions started but not terminated and to distinguish the time in consecutive steps, respectively. Moreover, for every durative action a, we add a predicate r_a, which will be kept to true while a durative action is running, and a numeric fluent c_a which will be used to measure the time since the start of a. Finally, for every fluent f we define three "lock" predicates, rl_f, al_f and il_f, that will be used to enforce mutual exclusion constraints.

The initial state and goal condition are defined by simply augmenting the original initial state and goal as follows.

$$\bar{I} \stackrel{\text{def}}{=} \{ok = \top, oc = gc = 0\} \cup \{r_a = \bot, c_a = 0 \mid a \in A^d\} \cup \{rl_f = al_f = il_f = \top \mid f \in FX\} \cup I$$
$$\bar{G} \stackrel{\text{def}}{=} G \wedge ok \wedge oc = 0$$

Before defining the rest of the compilation, the following definition provides the key machinery for encoding the non-interference constraints.

Definition 5. *Given an instantaneous action a, we define the **lock precondition** λ_a as:*

$$\bigwedge_{f \in V_a^r} (al_f \wedge il_f) \wedge \bigwedge_{f \in V_a^{:=}} (rl_f \wedge il_f \wedge al_f) \wedge \bigwedge_{f \in V_a^{+=}} (rl_f \wedge al_f).$$

*Moreover, we define the **lock effects** L_a as the set:*

$$\{al_f := \bot \mid f \in V_a^{:=}\} \cup \{il_f := \bot \mid f \in V_a^{+=}\} \cup \{rl_f := \bot \mid f \in V_a^r\}.$$

Intuitively, we will augment every happening associated with an instantaneous action or with the start or end of a durative action with its lock precondition and effects; if all locks are reset to $\top$ at every time step (see $\bar{e}^{\not t}$ below), no pair of interfering happenings can appear at the same time.

We now define the core of the translation starting from actions: we introduce a PDDL+ action for every instantaneous action, start of durative action and termination of non-fixed durative action in the original problem: $\bar{A} = \{\bar{a}^i \mid a \in A^i\} \cup \{\bar{a}^{\vdash} \mid a \in A^d\} \cup \{\bar{a}^{\dashv} \mid a \in A^{var}\}$ defined as follows.

$$\bar{a}^i \stackrel{\text{def}}{=} (pre_a \wedge ok \wedge \lambda_a, \mathit{eff}_a \cup L_a)$$

Instantaneous actions are simply augmented with the ok precondition (common to all other actions and events in the compilation) and with the lock preconditions and effects.

Durative actions are split into their starting and ending timepoints; fixed-duration actions will be terminated by an event ($\bar{e}_a^{\dashv}$ defined below), while variable-duration actions are terminated by the $\bar{a}^{\dashv}$ actions.

$$\bar{a}^{\vdash} \stackrel{\text{def}}{=} (pre_{a_{\vdash}} \wedge ok \wedge \lambda_{a_{\vdash}} \wedge \neg r_a, \mathit{eff}_{a_{\vdash}} \cup L_{a_{\vdash}} \cup \{r_a := \top, c_a := 0, oc \mathrel{+}= 1\})$$

The starting of every durative action a corresponds to the $a_{\vdash}$ snap action, we add the $\neg r_a$ precondition to enforce non-self-overlapping and we set r_a to true, we reset the clock c_a because we are just starting the action, and we increase oc to signal that a new action started but has not finished yet.

$$\bar{a}^{\dashv} \stackrel{\text{def}}{=} (pre_{a_{\dashv}} \wedge ok \wedge r_a \wedge l_a \leq c_a \leq u_a \wedge \lambda_{a_{\dashv}}, \mathit{eff}_{a_{\dashv}} \cup L_{a_{\dashv}} \cup \{r_a := \bot, oc \mathrel{+}= -1\})$$

To terminate a non-fixed durative action a, we require r_a and that the duration constraint is satisfied with $l_a \leq c_a \leq u_a$; we then reset r_a to false and decrement oc.

We have two types of very simple processes in our compilation that are used to keep track of the time passing while an action is running and to continuously increase gc for the mutex construction we will describe below. The compilation processes are then $\bar{P} = \{\bar{p}_a \mid a \in A^d\} \cup \{\bar{p}^{\not t}\}$ with:

$$\bar{p}_a \stackrel{\text{def}}{=} (ok \wedge r_a, \{\frac{d}{dt} c_a = 1\}) \qquad \bar{p}^{\not t} \stackrel{\text{def}}{=} (ok, \{\frac{d}{dt} gc = 1\})$$

In the compilation, events serve several purposes, each encoded in a different subset: $\bar{E} \stackrel{\text{def}}{=} \bar{E}^{\leftrightarrow} \cup \bar{E}_{\dashv}^{fix} \cup \bar{E}^{expire} \cup \{\bar{e}^{\not t}\}$. $\bar{E}_{\dashv}^{fix} \stackrel{\text{def}}{=} \{\bar{e}_a^{\dashv} \mid a \in A^{fix}\}$ encodes the termination of fixed-duration actions, $\bar{E}^{\leftrightarrow} \stackrel{\text{def}}{=} \{\bar{e}_a^{\leftrightarrow} \mid a \in A^d\}$ ensures that overall conditions of durative actions are not violated, $\bar{E}^{expire}$ ensure that no plan prefix has variable duration actions that are not terminated within the duration upper bound and $\bar{e}^{\not t}$ resets all the lock variables immediately after a time-elapse.

$$\bar{e}_a^{\dashv} \stackrel{\text{def}}{=} (pre_{a_{\dashv}} \wedge ok \wedge r_a \wedge c_a = l_a \wedge \lambda_{a_{\dashv}} \wedge gc = 0, \mathit{eff}_{a_{\dashv}} \cup L_{a_{\dashv}} \cup \{r_a := \bot, oc \mathrel{+}= -1\})$$

The termination of fixed-duration actions is analogous to the variable duration ones, but is an event scheduled at the fixed duration ($l_a = u_a$), measured by c_a. We also require gc to be 0 to execute this event after $\bar{e}^{\not t}$, as explained below.

$$\bar{e}_a^{\leftrightarrow} \stackrel{\text{def}}{=} (ok \wedge r_a \wedge \neg \gamma_a, \{ok := \bot\})$$

Overall conditions are enforced through the ok predicate: if we reach a state where action a is running (r_a is true) and its overall conditions are not satisfied, we set ok to false, making this prefix invalid, because ok can only be falsified, and never restored to true. Similarly, if a durative action with variable duration is not terminated within its duration upper bound ($r_a \wedge c_a > u$), we immediately set ok to false. (This is not strictly needed for correctness, as gc would never return to 0, but is useful for performance.)

$$\bar{E}^{expire} \stackrel{\text{def}}{=} \{(ok \wedge r_a \wedge c_a > u_a, \{ok := \bot\}) \mid a \in A^{var}\}$$

Domain	ENHSP LG	ENHSP WA	ARIES	OPTIC	TAMER	TFLAP	Next-FLAP	Patty
MatchCellar	7	12	**20**	9	7	**20**	2	4
MaJSP	**20**	19	18	N/A	**20**	N/A	N/A	N/A
T-Plant-Wat	**20**	16	15	20	12	0	0	13
T-Sailing	11	**20**	6	7	3	2	7	2
Total (80)	58	**67**	59	36	42	22	9	19

Table 1: Coverage analysis domain by domain, planner by planner. Bold for best, N/A for Not Applicable.

Finally, we have a single event that restores the locks:

$$\bar{e}^{\lightning} \stackrel{\text{def}}{=} (ok \wedge gc > 0, \{gc := 0\} \cup \{rl_x := \top, x_{wl} := \top, il_x := \top \mid x \in FX\})$$

The idea of this "lock" construction is that in a certain time we start the chains of happenings with the event $\bar{e}^{\lightning}$ that resets all the locks, then we can execute other actions or events, but every time we "read" a fluent f (either in a precondition or in the right-hand-side of an effect) we set rl_f to false, every time we have an assignment or increment effect of f we set al_f or il_f respectively to false. Thanks to the lock preconditions, we forbid reading a variable if it was previously (in the same "superdense" time) assigned or incremented, we forbid an assignment if it was previously assigned, increased or read, and we forbid increments if it was previously assigned or read. The whole trick is that gc will be continuously increased by a process, so in the subsequent times, the locks are automatically reset by the event $\bar{e}^{\lightning}$ and the locks are released. This faithfully captures the semantics of non-interference we outlined for temporal planning.

Given a plan $(\bar{\pi}^+, t_e)$ with $\bar{\pi}^+ = (t_1, a_1), \ldots (t_n, a_n)$ for $\bar{\Pi}$, we define the temporal plan $\tilde{\pi}$ solving Π as:

$$\begin{aligned}\tilde{\pi} \stackrel{\text{def}}{=} &\{(t, a, 0) \mid (t, \bar{a}^i) \in \bar{\pi}\} \cup \\ &\{(t_\vdash, a, l_a) \mid (t_\vdash, a^\vdash) \in \bar{\pi} \text{ and } a \in A^{fix}\} \cup \\ &\{(t_\vdash, a, t_\dashv - t_\vdash) \mid (t_\vdash, a^\vdash), (t_\dashv, a^\dashv) \in \bar{\pi} \text{ and } \\ &\quad \nexists t. t_\vdash < t' < t_\dashv, (t', a^\vdash) \in \bar{\pi}\}\end{aligned}$$

In the extended version of this paper (Micheli, Scala, and Valentini 2026), we prove that the compilation is sound and complete. Indeed, if a plan $(\bar{\pi}^+, t_e)$ is found for $\bar{\Pi}$, so is $\tilde{\pi}$ for Π. This follows by proving that constraints 1-6 for the validity of a plan are all implied by the existence of a state sequence induced by $(\bar{\pi}^+, t_e)$. Completeness is more challenging: we prove that for every temporal plan, there exists a sufficiently small δ for which there is a corresponding valid PDDL+ plan for $\bar{\Pi}$. Finally, we note that the compilation is polynomial in size and the plan length is at most doubled.

Experimental Evaluation

We experimented with our compilation over a selection of temporal numeric domains. We focused our attention on problems requiring intertwined reasoning between numeric and temporal aspects, where the concurrency of the durative actions is necessary to solve the instances. As representative of temporally interesting domains, we took the classic Matchcellar IPC domain and MAJSP from (Micheli and Scala 2019). Then we introduce two new domains, T-Sailing, and T-Plant-Watering. T-Sailing extends Sailing (Scala et al. 2020a) by requiring a boat not only to rescue the persons in some specific area of the cartesian space, but also to do so under a specific deadline. If the boat arrives too late, the person cannot be saved anymore. T-Plant-Watering extends Plant-Watering (Francès and Geffner 2015) by imposing temporal constraints between pouring and opening the tap. The task becomes a collaboration involving two distinct agents: one carries the pump used to water the plants, but can begin watering only when the other agent simultaneously performs the task of opening the tap. Benchmarks are available at https://github.com/hstairs/time2processes.

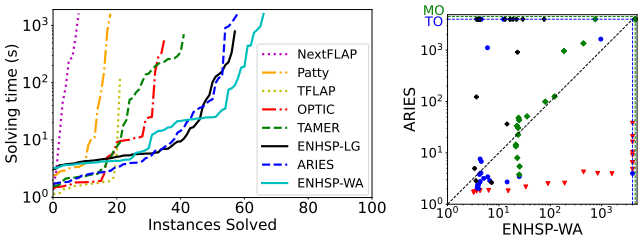

Figure 1: Cactus plot (left) and run-time scatter plot for highest-coverage planners (right): we indicate MatchCellar as ▼, MaJSP as ●, T-Plant-Wat. as ♦ and T-Sailing as ✚.

For each domain we have 20 instances, mostly scaling with the number of objects. Our analysis focuses on coverage (number of solved instances per domain) and run-time. ENHSP is used as the PDDL+ planner, run with two different engines, i.e., lazy greedy best-first search (ENHSP-LG) and WA* (ENHSP-WA), both with the h^{mrp} heuristic (Scala et al. 2020b). In LG, we used focus search as in Scala and Bonassi (2025); in WA* we use $w = 4$. The compiler is implemented within the `unified_planning` library (Micheli et al. 2025), and also supports delayed effects and timed initial literals; roughly, we emulate them with events triggered at the proper time (we omit the formal description due to space constraints). We compare the compilation with native state-of-the-art temporal planners, i.e., ARIES (Bit-Monnot 2023), NextFLAP and TFLAP (Sapena, Onaindia, and Marzal 2024), OPTIC (Benton, Coles, and Coles 2012), TAMER (Valentini, Micheli, and Cimatti 2020) and Patty (Cardellini and Giunchiglia 2025). Experiments were run on an AMD EPYC 7413; 1800s timeout, 20 GB memory limit.

Results. Figure 1 shows per-domain coverage. ENHSP-WA got the highest coverage. (Some planners do not support MAJSP for lack of delayed effects support.) For purely temporal domains, temporal planners are faster, yet both ENHSP engines proved competitive, especially in MAJSP. Over temporal numeric domains, ENHSP-WA provided superior performance overall. Figure 1 (right) shows a pairwise analysis on run-time for the two best performing planners ARIES and ENHSP-WA. ARIES scales better in Matchcellar, ENHSP-WA better over the temporal numeric domains, highlighting a great deal of complementarity. Finally, Figure 1 (left) shows the number of instances solved over time, confirming the strength of our compilation.

Acknowledgments

Andrea Micheli and Alessandro Valentini have been partially supported by the STEP-RL project funded by the European Research Council under GA n. 101115870. Enrico Scala has been supported by the Italian Ministry of University and Research within the PRIMA 2024 programme project "Optimizing Water Resources in Coastal Areas using Artificial Intelligence" (AI4WATER – D53C25000510006).

References

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *Proceedings International Conference on Automated Planning and Scheduling, ICAPS 2012*.

Bit-Monnot, A. 2023. Enhancing Hybrid CP-SAT Search for Disjunctive Scheduling. In Gal, K.; Nowé, A.; Nalepa, G. J.; Fairstein, R.; and Radulescu, R., eds., *ECAI 2023 - 26th European Conference on Artificial Intelligence*, 255–262. IOS Press.

Cardellini, M.; and Giunchiglia, E. 2025. Temporal Numeric Planning with Patterns. In *AAAI-25 Conference on Artificial Intelligence*, 26481–26489. AAAI Press.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Proceedings International Conference on Automated Planning and Scheduling, ICAPS 2010*.

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

Fox, M.; and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *Journal of Artificial Intelligence Research*.

Francès, G.; and Geffner, H. 2015. Modeling and Computation in Planning: Better Heuristics from More Expressive Languages. In Brafman, R. I.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015*, 70–78. AAAI Press.

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.

Micheli, A.; Bit-Monnot, A.; Röger, G.; Scala, E.; Valentini, A.; Framba, L.; Rovetta, A.; Trapasso, A.; Bonassi, L.; Gerevini, A. E.; Iocchi, L.; Ingrand, F.; Köckemann, U.; Patrizi, F.; Saetti, A.; Serina, I.; and Stock, S. 2025. Unified Planning: Modeling, manipulating and solving AI planning problems in Python. *SoftwareX*, 29: 102012.

Micheli, A.; and Scala, E. 2019. Temporal Planning with Temporal Metric Trajectory Constraints. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, 7675–7682. AAAI Press.

Micheli, A.; Scala, E.; and Valentini, A. 2026. Compiling Temporal Numeric Planning into Discrete PDDL+: Extended Version. *CoRR*, abs/2603.12188.

Percassi, F.; Scala, E.; and Vallati, M. 2025. On the Notion of Plan Quality for PDDL+. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 35, 102–111.

Sapena, Ó.; Onaindia, E.; and Marzal, E. 2024. A hybrid approach for expressive numeric and temporal planning with control parameters. *Expert Syst. Appl.*, 242: 122820.

Scala, E.; and Bonassi, L. 2025. On Using Lazy Greedy Best-First Search with Subgoaling Relaxation in Numeric Planning Problems. In *Proceedings International Conference on Automated Planning and Scheduling, ICAPS 2025*, 245 – 249.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2020a. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *J. Artif. Intell. Res.*, 68: 691–752.

Scala, E.; Saetti, A.; Serina, I.; and Gerevini, A. E. 2020b. Search-Guidance Mechanisms for Numeric Planning Through Subgoaling Relaxation. In *Proceedings International Conference on Automated Planning and Scheduling, ICAPS 2020*, 226–234. AAAI Press.

Valentini, A.; Micheli, A.; and Cimatti, A. 2020. Temporal Planning with Intermediate Conditions and Effects. In *AAAI-20 Conference on Artificial Intelligence*.