A Sound (but Incomplete) Polynomial Translation from Discretised PDDL+ to Numeric Planning

Francesco Percassi,[1] **Enrico Scala,**[2] **Mauro Vallati**[1]

[1] School of Computing and Engineering, University of Huddersfield, UK.
[2] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy
f.percassi@hud.ac.uk, enrico.scala@unibs.it, m.vallati@hud.ac.uk

Abstract

PDDL+ is an expressive planning formalism that enables modelling of hybrid domains having both discrete and continuous dynamics, in which the agent and the environment description are sharply separated. Recently two mappings for translating a PDDL+ problem under discrete interpretation into a numeric task have been proposed. Such translations produce a task of exponential or polynomial size, with respect to the size of the native task.

In this work, starting from the above-mentioned polynomial translation, we introduce a sound but not generally complete variant that has the potential to improve the performance of numeric planning engines. However, it is guaranteed to be complete only in a subclass of PDDL+ problems. We define the subclass of problems where the variant is safely applicable, and we empirically assess the advantages of such translation. Further, we show that even in cases when the variant does not guarantee the completeness, it can speed up the planning process, at the cost of losing part of the solutions space.

Introduction

Automated planning is a solid branch of artificial intelligence that aims to design methodologies for synthesising, on the basis of a known model of the world, a sequence of actions capable to transform a given state, i.e., the initial state, into the desired state, i.e., the goal state. The planning community has developed several planning formalisms with the aim of progressively increasing their expressive power, to allow to accurately model complex application domains. An important step in this direction was made by PDDL2.1 (Fox and Long 2003), which enabled the possibility of giving to the planning domains temporal and numerical behaviour.

Many real-world systems are characterised by the coexistence of a discrete and continuous dynamic. Such physical systems can be modeled with a particular class of dynamic systems called *hybrid systems*. A well-established formal theory to describe this class of systems has been provided by *Hybrid Automata* (HA) (Henzinger 1996), which have found wide use within the model-checking research community with the purpose of developing robust methodologies to test the reachability of hazardous states.

PDDL+ (Fox and Long 2006) is a planning formalism that brings the planning capacity within the aforementioned class of hybrid systems. It can be argued that one of the major contributions introduced by PDDL+ lies also in the clear separation that subsists between the changes in the world description due to the actions performed by the agent and those ascribable to an exogenous reaction of the environment, thus outside the agent's control. However, PDDL+ planning problems are notoriously difficult to be solved, and there is a restricted set of planning engines that can support PDDL+ natively.

Recently, to increase the pool of planning engines that can tackle hybrid planning problems, an exponential and polynomial translation schemes have been proposed to translate a PDDL+ problem into a discrete numeric problem (Percassi, Scala, and Vallati 2021). This approach is part of a broader family of reformulation approaches that directly extend the set of methodologies that can be exploited to solved planning problems expressed in a given formalism (see for example (Keyder and Geffner 2009; Palacios and Geffner 2009; Cooper, Maris, and Régnier 2010; Taig and Brafman 2013; Percassi and Gerevini 2019)).

In the present work, we aim to deepen the understanding of the polynomial scheme proposed by Percassi, Scala, and Vallati (2021), hereinafter denoted with POLY, with the aim of understanding under what conditions it is possible to obtain encoding capable of making the search less expensive and therefore more efficient.

In particular, we propose a compact variant of this schema, namely POLY$^-$, which has the property of being *sound* but *incomplete* for a general discretised PDDL+ planning problem. After having formally presented the POLY$^-$ translation, we first identify which kind of solutions are excluded from the search space induced by the novel transformation, and second, we outline in which class of planning problems POLY$^-$ is both sound and complete. Our experimental analysis shows the performance gain that can be obtained by using the proposed POLY$^-$ translation.

Background

Problem Formalisation

In this section we formalise the problem of PDDL+ (Fox and Long 2006), and the problem of numeric planning as the one that can be specified in PDDL2.1 (level 2) (Fox and Long 2003), hereinafter simply referred to as PDDL2.1. We first describe the syntax of our problems, then detail the semantics of PDDL+ under discrete time. Our discussion follows Shin and Davis (2005)'s formalisation and terminology[1] in a way that is instrumental for our work. We detail our problems using propositional formulas over comparisons and Boolean variables[2]. A comparison is $\xi \bowtie 0$ where ξ is a mathematical expression, and $\bowtie \in \{\leq, <, =, >, \geq\}$.

Definition 1 (PDDL+ problem). *A PDDL+ problem Π is the tuple $\langle F, X, I, G, A, E, P\rangle$ where:*

- *F and X are the sets of Boolean and numeric variables, respectively.*
- *I and G are the description of the initial state, expressed as a full assignment to all variables in X and F, and the description of the goal, expressed as a formula, respectively.*
- *A and E are the sets of actions and events, respectively. Actions and events are pairs $\langle p, e\rangle$ where p is a propositional formula and e is a set of conditional effects of the form $c \triangleright e$ where (i) c is a formula and (ii) e is a set of Boolean ($f = \{\bot, \top\}$) or numeric ($\langle\{asgn, inc, dec\}, x, \xi\rangle$ where ξ is an expression over X) assignments.*
- *P is a set of processes. A process is a pair $\langle p, e'\rangle$ where p is a formula and e' is a set of numeric continuous effects expressed as pairs $\langle x, \xi\rangle$ with the meaning that ξ represents the time-derivative of x, with $x \in X$.*

Let $a = \langle p, e\rangle$ be an action or an event or a process, we use *pre*(a) to denote the precondition p of a, and *eff*(a) the effect e of a. In the following we will use a, ρ and ε for denoting an action, process and event, respectively. In order to lighten the notation, all conditional effects in the form $\top \triangleright e$ are written as e, by omitting the antecedent $\top$.

A plan for a PDDL+ is an ordered set of timed actions plus a time envelope, organised formally as following.

Definition 2 (PDDL+ plan). *A PDDL+ plan π_t is a pair $\langle \pi, \langle t_s, t_e\rangle\rangle$ where $\pi = \langle\langle a_0, t_0\rangle, \langle a_1, t_1\rangle, ..., \langle a_{n-1}, t_{n-1}\rangle\rangle$, with $t_i \in \mathbb{R}$, is a sequence of time-stamped actions and $\langle t_s, t_e\rangle$, with $t_s, t_e \in \mathbb{R}$, is a real interval, called envelope, within which π is performed.*

Definition 3 (PDDL2.1 problem). *A PDDL2.1 problem Π is the tuple $\langle F, X, I, G, A, c\rangle$ where all elements are as for PDDL+, yet there are neither processes nor events and c associates to each action a rational cost.*

Definition 4 (PDDL2.1 plan). *A PDDL2.1 plan is a sequence of actions $\langle a_0, ..., a_{n-1}\rangle$. The cost of the plan π is the sum of all action costs in π, cost$(\pi) = \sum_{a \in \pi} c(a)$.*

In the rest of the section we focus on the discrete semantic of PDDL+, mainly based on Percassi, Scala, and Vallati (2021), and in particular on its temporal dimension. We assume the reader is familiar with notions of action/event applicability, and simply use $\gamma(s, \cdot)$ for the state resulting by applying either an action/event ($\gamma(s, a)$) or a sequence of action/events ($\gamma(s, \langle a_0, \dots, a_n\rangle)$) in state s. For details on PDDL2.1, we refer the reader to Fox and Long (2003). Note that assuming the semantics provided in (Fox and Long 2003), an action is valid, and then applicable, if no numeric variables appears as an lvalue in more than one simple assignment effect, i.e., one using $asgn$. Since the numeric effect having the form $\langle inc, x, \xi\rangle$ ($\langle dec, x, \xi\rangle$) are normalised in $\langle asgn, x, x{+}\xi\rangle$ ($\langle asgn, x, x{-}\xi\rangle$), then an action a having at least two numeric assignment affecting the same variable and using inc or dec is to be considered invalid. This especially holds if the two aforementioned conflicting effects are subordinated to the triggering some conditional effects.

Moreover, we exploit the widely shared assumptions of boundness adopted in PDDL+ problems (Fox and Long 2006). Importantly, we assume there always is a finite (possibly empty) and unique acyclic sequence of events that can be triggered, and there is a bound on the number of spontaneous changes of processes over a closed interval. We start off by recalling the notion of time points, intervals and histories over intervals, which are used to define the validity of PDDL+ plan interpreted on a discrete timeline.

A time point T is a pair $\langle t, n\rangle$ where $t \in \mathbb{R}$ and $n \in \mathbb{N}$. Time points over $\mathbb{R} \times \mathbb{N}$ are ordered lexicographically.

A history $\mathcal{H}$ over $\mathcal{I} = [T_s, T_e]$ maps each time point in $\mathcal{I}$ into a situation. A "situation at time T" is the tuple $\mathcal{H}(T) = \langle \mathcal{H}_A(T), \mathcal{H}_s(T)\rangle$, where $\mathcal{H}_A(T)$ is the set of actions executed at time T and $\mathcal{H}_s(T)$ is a state, i.e., an assignment to all variables in X and F at time T. We denote by $\mathcal{H}_s(T, v)$ and $\mathcal{H}_s(T, \xi)$ the value assumed in the state at time T by $v \in F \cup X$ and by a numeric expression ξ, respectively. $E_{trigg}(T)$ indicates the set of active events in T.

T is a significant time point of $\mathcal{H}$ over $[T_s, T_e]$ iff, in such a time point, an action is applied, an event is triggered, a process has started or stopped or there has been a discrete change just before. A history $\mathcal{H}$ is monotonous over a real interval $\mathcal{I}_t$ if there are no significant time points in $\mathcal{I}_t$.

Definition 5 (Valid PDDL+ plan). *π_t is valid plan for Π iff $\mathcal{H}^\pi_s(T_m) \models G$ and, for each $a \in \mathcal{H}^\pi_A(T)$ with $T \in \mathcal{I}$, $\mathcal{H}^\pi_s(T) \models$ pre(a).*

Given a mathematical expression ξ denoting the time derivative of a given variable x, and a rational value $\delta \in \mathbb{Q}$ we denote with $\Delta(\xi, \delta)$ the discretised update for x.

Definition 6 (PDDL+ plan discrete projection). *Let $\delta \in \mathbb{Q}$, let $\mathbb{H}^\pi$ be a history, let I be an initial state and let π_t be a PDDL+ plan. We say that $\mathbb{H}^\pi$ is a discrete projection of π_t which starts in I iff $\mathbb{H}^\pi$ induces the significant time points $T_\mathbb{H} = \langle T_0 = \langle t_0, n_0\rangle, \cdots, T_m = \langle t_m, n_m\rangle\rangle$ where either $t_{i+1} = t_i + \delta$ or $t_{i+1} = t_i$*

R1 *$E_{trigg}(T_i) \neq \emptyset$ iff $\mathcal{H}^\pi_s(T_{i+1}) = \gamma(\mathcal{H}^\pi_s(T_i), E_{trigg}(T_i))$, $\mathcal{H}^\pi_A(T_i) = \emptyset$, $t_{i+1} = t_i$ and $n_{i+1} = n_i + 1$;*

R2 *$\mathcal{H}^\pi_A(T_i) \neq \emptyset$ iff $\mathcal{H}^\pi_s(T_{i+1}) = \gamma(\mathcal{H}^\pi_s(T_i), \mathcal{H}^\pi_A(T_i))$, $E_{trigg}(T_i) = \emptyset$, $t_{i+1} = t_i$ and $n_{i+1} = n_i + 1$;*

R3 *for each $\langle a_i, t_i\rangle, \langle a_j, t_j\rangle \in \pi$, with $i < j$ and $t_i = t_j$ there exists $T_k, T_z \in T_\mathcal{H}$ such that $a_i \in \mathcal{H}^\pi_A(T_k)$ and $a_j \in \mathcal{H}^\pi_A(T_z)$ where $t_k = t_z = t_i$ and $n_k < n_z$;*

R4 *for each pair of contiguous significant time points $T_i = \langle t_i, n_i\rangle$, $T_{i+1} = \langle t_{i+1}, 0\rangle$ such that $t_{i+1} = t_i + \delta$, the value*

[1]As Shin and Davis (2005), we idealise the execution of actions and events to be truly instantaneous transitions as long as an order is imposed among transitions sharing the same time clock.

[2]With abuse of notations we also use positive and negative literals as shortcut for $f = \{\top\}$.

of each numeric variable $x \in X$ is updated as:

$$\mathbb{H}_s^\pi(T_{i+1}, x) = \mathbb{H}_s^\pi(T_i, x) + \sum_{\substack{\langle x', \xi\rangle \in \mathit{eff}(\rho),\, x'=x \\ \rho \in \{\rho \in P,\, \mathbb{H}_s^\pi(T_i) \models \mathit{pre}(\rho)\}}} \mathbb{H}_s^\pi(T_i, \Delta(\xi, \delta))$$

and values of unaffected variables remain unchanged (frame-axiom).

In the following we provide a more intuitive description of R1–R4 of Def. 6. R1 (R2) states that if an action (event) is executed (triggered) in a significant time point $T1 = \langle t, n\rangle$, then there necessary exists a successor of $T1$, i.e., $T2 = \langle t, n+1\rangle$ having the same clock t and the step increased by one unit, i.e., $n+1$; the successor state associated to $T2$ is calculated by simply applying the discrete effects of the action (event). R3 is used to enforce how actions of a PDDL+ plan π are projected over an history, preserving their original ordering in case they share the same time-stamp in π. R4 is used to enforce how a numeric variable changes continuously over time according to the active processes in those "monotonous" temporal intervals in which "nothing happens" (there is no action/event executed/triggered and there is no process which starts/ends).

Definition 7 (δ Discrete valid PDDL+ plan). *π_t is a valid plan for Π under δ discretisation iff $\mathbb{H}_s^\pi(T_m) \models G$ and, for each $T \in \mathcal{I}$ such that $\mathbb{H}_A^\pi(T) \neq \emptyset$, then $\mathcal{H}_s^\pi(T) \models \mathit{pre}(a)$.*

Polynomial Translation

As observed by Percassi, Scala, and Vallati (2021), it is possible to translate a discretised PDDL+ into a PDDL2.1 problem with a polynomial translation, namely POLY. The key idea in POLY consists in simulating, through the execution of a sequence of actions, the progress of a discrete amount of time $\delta \in \mathbb{Q}$.

Let $\Pi = \langle F, X, I, G, A, \emptyset, P\rangle$ be an event-free PDDL+ problem, and a discretisation parameter $t = \delta$, POLY generates a new PDDL2.1 problem $\Pi_{\text{POLY}} = \langle F \cup D \cup \{\mathit{pause}\}, X \cup X^{cp}, I, G \wedge \neg \mathit{pause}, A_c \cup A_P \cup \{\mathit{start}, \mathit{end}\}, c\rangle$ such that:

$$X^{cp} = \{x^{copy} \mid x \in X\}$$

$$D = \bigcup_{\substack{ne \in \mathit{eff}(\rho) \\ \rho \in P}} \{\mathit{done}_{ne}\}$$

$$A_c = \{\langle \mathit{pre}(a) \wedge \neg \mathit{pause}, \mathit{eff}(a)\rangle \mid a \in A\}$$

$$\mathit{start} = \langle \neg \mathit{pause}, \{\mathit{pause}\} \cup \bigcup_{x \in X} \{\langle asgn, x_{copy}, x\rangle\}\rangle$$

$$\mathit{end} = \langle \bigwedge_{\mathit{done} \in D} \mathit{done} \wedge \mathit{pause}, \{\neg \mathit{pause}\} \cup \bigcup_{\mathit{done} \in D} \{\neg \mathit{done}\}\rangle$$

$$A_P = \bigcup_{\substack{ne:\langle x,\, \xi\rangle \in \mathit{eff}(\rho) \\ \rho \in P}} \{\langle \mathit{pause} \wedge \neg \mathit{done}_{ne}, \{\sigma(\mathit{pre}(\rho), X^{cp}) \triangleright \{\langle inc, x, \Delta(\delta, \sigma(\xi, X^{cp}))\rangle\}\} \cup \{\mathit{done}_{ne}\}\rangle\}$$

Whenever the passage of a discrete amount of time δ has to be simulated within Π_{POLY}, the sequence of actions $\mathit{wait} = \langle \mathit{start}, \mathit{seq}(A_P), \mathit{end}\rangle$, where $\mathit{seq}(A_P)$ is any permutation of all A_P actions, has to be performed. Such simulation consists of the following steps:

- *start*, this action enables the execution of all A_P actions and, at the same time, disables all those that do not belong to A_P through the use of the *pause* predicate;
- $\mathit{seq}(A_P)$, this sequence modifies the state of the world according to the dynamics of the active processes; to prevent the A_P actions from interfering with each other, the *start* action performs a copy of all the numeric variables X, assigning the current value to the corresponding X^{cp} variables; this allows to correctly modify the state of the world, regardless of the specific sorting chosen for $\mathit{seq}(A_P)$;
- *end*, this actions closes the simulation and can be executed, by using the *done* predicates, if all the A_P actions have been executed.

Sound (but Incomplete) Translation

For an event-free PDDL+ problem Π, the POLY$^-$ reformulation generates a PDDL2.1 problem $\Pi_{\text{POLY}^-} = \langle F, X, I, G, A \cup \{\mathit{SIM}\}, c\rangle$, discretised in $t = \delta$. Π_{POLY^-} is almost identical to Π but for the absence of processes and the presence of the special action *SIM* playing the role of the simulator, i.e., what changes when time goes forward (similarly to the *wait* sequence for POLY translation). *SIM* is defined as follows:

$$\mathit{pre}(\mathit{SIM}) = \top$$

$$\mathit{eff}(\mathit{SIM}) = \bigcup_{\rho \in P} \{\mathit{pre}(\rho) \triangleright \bigcup_{\langle x, \xi\rangle \in \mathit{eff}(\rho)} \{\langle inc, x, \Delta(\delta, \xi)\rangle\}\}$$

SIM action is always applicable and features a conditional effect for each process $\rho \in P$. Such conditional effect is triggered if the preconditions of ρ holds when *SIM* is applied, modifying, for each $\langle x, \xi\rangle \in \mathit{eff}(\rho)$, the affected numeric variable x according to the discretised effect expression, i.e $\Delta(\delta, \xi)$.

Hereinafter when we say "Π admits a valid solution" we imply "under discrete interpretation".

Proposition 1 (Soundness of POLY$^-$). *Let Π be a PDDL+ problem, and let Π_{POLY^-} be the PDDL2.1 problem obtained by using the POLY$^-$ translation discretised in $t = \delta$. If Π_{POLY^-} admits a solution then so does Π.*

Proposition 2 (Incompleteness of POLY$^-$). *Let Π be a PDDL+ problem, and let Π_{POLY^-} be the PDDL2.1 problem obtained by using the POLY$^-$ translation discretised in $t = \delta$. If Π admits a solution then Π_{POLY^-} may not admit it.*

Proof. To show the correctness of the proposition we build the simplest case in which a solution π_t for Π does not admit a corresponding solution π' for Π_{POLY^-}.

Let $\pi_t = \langle \pi, (0, t_e)\rangle$ be a valid solution for Π (assume w.l.o.g. $t_s = 0$) under δ discretisation and assume that π_t induces, according to Def. 6, a discrete projection $\mathbb{H}^\pi$ in which the following conditions are satisfied

- in the first significant time point of $\mathbb{H}^\pi$, i.e., $T_0 = \langle 0, 0\rangle$, two numeric continuous effect affecting the same variable

$x \in X$ are active, i.e, $\mathbb{H}^{\pi}_{s}(T_0) \models pre(\rho) \wedge pre(\rho')$ with $\rho, \rho' \in P$, $\langle x, \xi \rangle \in \mathit{eff}(\rho)$ and $\langle x, \xi' \rangle \in \mathit{eff}(\rho')$;
- $\mathbb{H}^{\pi}$ is monotonous over $(0, \delta)$.

Using the translation POLY$^-$ we generate a PDDL2.1 planning problem Π_{POLY^-} where the action *SIM* has, in reference to ρ and ρ', the following conditional effects:

$$\begin{aligned} \mathit{eff}(SIM) = \{&pre(\rho) \triangleright \{\langle op, x, \Delta(\delta, \xi)\rangle\} \\ &pre(\rho') \triangleright \{\langle op', x, \Delta(\delta, \xi')\rangle\}, ...\} \end{aligned}$$

Let π' be a PDDL2.1 plan constructed in such a way that: i) for each $\langle a, t \rangle \in \pi$ then $a' \in \pi'$ (where a' is the compiled version of a); ii) for each $\langle a_i, t_i \rangle, \langle a_j, t_j \rangle$ with $a_i \prec a_j$ in π then $a'_i \prec a'_j$ in π' iii) a sequence, possibly empty, of *SIM* actions has to be placed before each action $a'_i \in \pi'$ and at the end of π' according to the following structure:

$$\pi' = \langle \langle SIM \rangle \times \frac{t_0}{\delta}, a'_0, \langle SIM \rangle \times \frac{t_1 - t_0}{\delta}, ..., a'_{n-1}, \langle SIM \rangle \times \frac{t_e - t_{n-1}}{\delta} \rangle$$

where $\langle SIM \rangle \times k$ indicates k repetitions of *SIM*.

By using this mapping from π_t to π', we get that the first action of π' is *SIM*, which is used to simulate the passage of time from 0 to δ. Π and Π_{POLY^-} have the same initial state and then *SIM* is applied in a state $I \models pre(\rho) \wedge pre(\rho')$, thus activating the aforementioned conditional effects which both affect x. The state thus generated has to be considered, according to the PDDL2.1 semantic, inconsistent and it has to be pruned from the search space induced by Π_{POLY^-}. It follows that the action *SIM* is not applicable in I within Π_{POLY^-}, and then π' is not a valid solution for Π_{POLY^-}. □

Through this discussion, we have shown a minimal case in which the translation scheme POLY$^-$ generates a numeric problem whose induced search space does not contain some of the states present in the original problem.

Definition 8 (Forbidden states)**.** *Let Π be a PDDL+ problem, and let Π_{POLY^-} be the PDDL2.1 problem obtained by using the POLY$^-$ translation discretised in $t = \delta$. We define the set of forbidden states for Π_{POLY^-} as follows:*

$$FS(\Pi_{\text{POLY}^-}) = \{s \mid s \in states(\Pi), s \models \bigvee_{\langle \rho, \rho' \rangle \in FP(\Pi)} pre(\rho) \wedge pre(\rho')\}$$

where

$$\begin{aligned} FP(\Pi) = \{\langle \rho, \rho' \rangle \mid \langle x, \xi \rangle \in \mathit{eff}(\rho), \langle x', \xi' \rangle \in \mathit{eff}(\rho'), \\ x = x',\ \rho \neq \rho'\} \end{aligned}$$
$$states(\Pi) = \{\bot, \top\}^{|F|} \times (\mathbb{R} \cup \{\bot\})^{|X|}$$

The set of forbidden states for Π_{POLY^-} is defined as the set of states belonging to all possible tuples definable on $X \cup V$ such that at least one pair of forbidden processes is active. A pair of processes $\langle \rho, \rho' \rangle$ is said to be forbidden, i.e., $\langle \rho, \rho' \rangle \in FP(\Pi)$, iff ρ and ρ' have continuous numeric effects having the same variable as right-hand side value.

Note that these states are labelled as *forbidden* in the sense that, once reached, it is not possible to apply the *SIM* action as its execution would cause the generation of an undefined state. This means that a π' plan for Π_{POLY^-} can generate a forbidden state as long as no *SIM* action is applied to it.

Also note that Def. 8 is syntactic definition and it independent from I, G or from the reachability of such states, therefore we could have a Π problem such that $FS(\Pi_{\text{POLY}^-}) \neq \emptyset$ but in which such states are unreachable.

In the following we outline a syntactic property that a domain has to satisfy in order to ensure that POLY$^-$ is complete other than sound.

Definition 9 (Mono left-hand side Π)**.** *Let Π be a PDDL+ problem. We say that Π is a mono left-hand side PDDL+ problem (shortened in 1-lhs) if it does not have two processes having numeric continuous effects having as left-hand side value the same numeric variable.*

Proposition 3 (Soundness and completeness of POLY$^-$ under *1-lhs* condition)**.** *Let Π be a 1-lhs PDDL+ problem, and let Π_{POLY^-} be the PDDL2.1 problem obtained by using the POLY$^-$ translation discretised in $t = \delta$. Π_{POLY^-} admits a solution under δ discretisation iff so does Π.*

The property of completeness stated in Proposition 3 derives from the definition of *1-lhs* domain which implies that the set of forbidden states is necessarily empty as forbidden pair processes can not exist. The semantic property of being *1-lhs* is sufficient but not necessary (in the sense that there may be problems that are not *1-lhs* where POLY$^-$ is sound and complete).

In the following, we study in what relationship the solutions space of Π and Π_{POLY^-} are in order to actually understand what we can expect if we use POLY$^-$.

Definition 10 (Solutions space)**.** *Let Π be a 1-lhs PDDL+ problem, and let Π_{POLY^-} be the PDDL2.1 problem obtained by using the POLY$^-$ translation discretised in $t = \delta$. We define the the solutions space of Π and Π_{POLY^-} as follows:*

$$SS(\Pi) = \{\pi_t \mid \pi_t \textit{ is a valid solution for } \Pi\},$$
$$SS(\Pi_{\text{POLY}^-}) = \{map^{-1}(\pi') \mid \pi' \textit{ is a valid solution for } \Pi_{\text{POLY}^-}\}$$

where $map^{-1}(\pi)$ is a procedure that, given a PDDL2.1 plan, it builds the corresponding PDDL+ plan.

Note that the inverse mapping, realised through the function $map^{-1}(\cdot)$, is necessary in order to make the solutions spaces sets described in different formalism comparable. Starting from π' we can build $\pi_t = map^{-1}(\pi') = \langle \pi, \langle 0, t_e \rangle \rangle$ as follows: i) for each action $a'_i \in \pi'$ such that $a'_i \neq SIM$ then $\langle a_i, t_i \rangle \in \pi$, where t_i is equal to δ multiplied for the occurrences of *SIM* in π' before a'_i; ii) for each a'_i, a'_j such that $a'_i \prec a'_j$ in π' then $\langle a_i, t_i \rangle \prec \langle a_j, t_j \rangle$ in π and iii) t_e is equal to δ multiplied by the number of *SIM* in π'.

Proposition 4. *Let Π be a PDDL+ problem, and let Π_{POLY^-} be the PDDL2.1 problem obtained by using the POLY$^-$ translation discretised in $t = \delta$, then the following relationship hold:*

1. *in the general case it holds that $SS(\Pi_{\text{POLY}^-}) \subseteq SS(\Pi)$;*
2. *if Π is 1-lhs, then $SS(\Pi_{\text{POLY}^-}) = SS(\Pi)$;*
3. *if Π is not 1-lhs and then there exists a plan π for Π whose execution generates a $s \in FS(\Pi)$, in which time*

has to pass, then $\pi \notin SS(\Pi_{\text{POLY}^-})$ and thus $SS(\Pi_{\text{POLY}^-}) \subset SS(\Pi)$;

4. *if Π is not 1-lhs and each $\pi \in SS(\Pi)$ generates a state $s \in FS(\Pi)$, in which time has to pass, $SS(\Pi_{\text{POLY}^-}) = \emptyset$.*

Example 1 (POLY VS POLY$^-$ translation)**.** *Let $\Pi = \langle F, X, I, G, A, \emptyset, P\rangle$ be a PDDL+ problem without events encompassing one Boolean variable, i.e., $F = \{f_1\}$, four numeric variables, i.e., $X = \{x_1, x_2, x_3, x_4\}$ and two processes $P = \{\rho_1, \rho_2\}$ such that:*

$$\rho_1 = \langle x_1 > 0, \{\langle x_2, x_3\rangle\}\rangle,\ \rho_2 = \langle f_1, \{\langle x_2, x_4\rangle\}\rangle$$

According to R4 of Def. 6, ρ_1 affects x_2 according to $\frac{dx_2}{dt} = x_3$ when $x_1 > 0$ holds and, similarly, ρ_1 affects x_2 according to $\frac{dx_2}{dt} = x_4$ when f_1 holds. Finally, if $x_1 > 0 \wedge f_1$, then $\frac{dx_2}{dt} = x_3 + x_4$.

POLY *Let $ne_1 = \langle x_2, x_3\rangle$ and $ne_2 = \langle x_2, x_4\rangle$ be the numeric continuous effects of ρ_1 and ρ_2, respectively. The PDDL2.1 problem obtained using POLY discretised in $t = \delta$ is $\Pi_{\text{POLY}} = \langle F \cup \{done_{ne_1}, done_{ne_2}\} \cup \{pause\}, X \cup \{x_1^{copy}, x_2^{copy}, x_3^{copy}, x_4^{copy}\}, I, G \wedge \neg pause, A_c \cup \{SIM\text{-}ne_1, SIM\text{-}ne_2\} \cup \{start, end\}, c\rangle$ such that:*

$$\begin{aligned}
start =& \langle \neg pause, \{\langle asgn, x_1^{copy}, x_1\rangle, \langle asgn, x_2^{copy}, x_2\rangle,\\
& \langle asgn, x_3^{copy}, x_3\rangle, \langle asgn, x_4^{copy}, x_4\rangle, pause\}\rangle\\
SIM\text{-}ne_1 =& \langle pause \wedge \neg done_{ne_1}, \{(x_1^{copy} > 0) \triangleright \{\langle inc, x_2, x_3^{copy} \cdot \delta\rangle\},\\
& done_{ne_1}\}\rangle\\
SIM\text{-}ne_2 =& \langle pause \wedge \neg done_{ne_2}, \{f_1 \triangleright \{\langle inc, x_2, x_4^{copy} \cdot \delta\rangle\},\\
& done_{ne_2}\}\rangle\\
end =& \langle pause \wedge done_{ne_1} \wedge done_{ne_2}, \{\neg pause, \neg done_{ne_1},\\
& \neg done_{ne_2}\}\rangle
\end{aligned}$$

POLY$^-$ *The PDDL2.1 problem obtained using POLY discretised in $t = \delta$ is $\Pi_{\text{POLY}^-} = \langle F, X, I, G, A \cup \{SIM\}, c\rangle$ such that:*

$$\begin{aligned}
SIM =& \langle \top, \{x_1 > 0 \triangleright \{\langle inc, x_2, x_3 \cdot \delta\rangle\},\\
& f_1 \triangleright \{\langle inc, x_2, x_4 \cdot \delta\rangle\}\}\rangle
\end{aligned}$$

Note that Π is not a 1-lhs because ne_1 and ne_2 affect the same variable, i.e., x_2, and so the transformation POLY$^-$ could not preserve the solutions space of Π. We can define the set of forbidden state for Π_{POLY^-} as follows:

$$FS(\Pi_{\text{POLY}^-}) = \{s \in states(\Pi),\ s \models f_1 \wedge x_1 > 0\}$$

Experimental Analysis

Our experimental analysis aims to evaluate the benefit that the proposed POLY$^-$ translation can provide to planning engines, also in comparison to the polynomial translation proposed in (Percassi, Scala, and Vallati 2021).

In this experimental evaluation, we consider the well-known numeric planning system METRIC-FF (Hoffmann 2003). Our experiments were run on an Intel Xeon Gold 6140M CPUs with 2.30 GHz. For each instance, we set a cutoff time of 900 seconds, and RAM was limited to 8 GB. As a benchmark, we consider the same suite examined in (Percassi, Scala, and Vallati 2021) which includes the following domains: Solar-Rover (Rover), Linear-Car (Lin-Car), Linear-Generator (Lin-Gen), Urban-Traffic-Control (UTC) from (Vallati et al. 2016), Baxter from (Bertolucci et al. 2019) and Overtaking-Car (OT-Car). With reference to the benchmark, Table 1 reports which of these domains satisfy the *1-lhs* property.

Table 2 shows the number of evaluated nodes, and the runtime needed by METRIC-FF to obtain a solution for problems translated through POLY and POLY$^-$. We measured the number of evaluated nodes and the search time on average by considering the instances solved by both approaches, only. When a domain is *1-lhs*, fewer and fewer nodes are evaluated, while, if not, either a solution is not found due to the incompleteness of POLY$^-$ (as in Lin-Gen and UTC) or it is necessary to evaluate more nodes, e.g., Baxter. As far as the search time is concerned, it emerges that, whenever METRIC-FF with POLY$^-$ finds a solution, it tends to be faster than when using POLY. In Lin-Car, although fewer nodes are evaluated using POLY$^-$, the time is essentially comparable. This may be due to the fact that the considered instances are too small to allow to notice significant differences. In Baxter, although it is not a *1-lhs* domain, solutions are generated faster using POLY$^-$ even if more nodes are evaluated; this is probably due to the more compact representation that makes search computationally lighter even if it prunes from the search space a large number of states, forcing the planning engine to explore a larger chunk of the search space to find a solution.

Figure 1 shows coverage over time for the considered translations. In particular, we have reported the results obtained for METRIC-FF using POLY and POLY$^-$ and a third composite translation, denoted as POLY*, which consists in preceding the compiler with a selector that switches to POLY$^-$ if the domain considered is *1-lhs* and to POLY otherwise. Notably, this easy selector can help in combining the strengths of the two translation, without losing completeness. For reference, Figure 1 also shows the coverage obtained by the planning engine ENHSP (Scala et al. 2020) when run on the original PDDL+ discretised models using the same machine and the runtime cutoff of METRIC-FF. Finally, ENHSP solves some problems within the time window of 1 second, while POLY, POLY$^-$ and POLY* do not because of the overhead introduced by the translation.

Domain	Rover	Lin-Car	Lin-Gen	UTC	Baxter	OT-car
1-lhs	✓	✓	✗	✗	✗	✓

Table 1: Benchmark structure analysis.

Domain	METRIC-FF			
	Evaluated Nodes (*000's*)		Search Time (*seconds*)	
	POLY	POLY$^-$	POLY	POLY$^-$
Rover	14.73	4.22	7.72	2.67
Lin-Car	2.62	0.40	3.08	3.20
Lin-Gen	—	—	—	—
UTC	—	—	—	—
Baxter	27.08	68.25	27.19	7.14
OT-Car	106.28	15.35	8.36	3.45

Table 2: Average evaluated nodes and search time per domain achieved by METRIC-FF when run using models translated by POLY and POLY$^-$. "—" denotes that POLY$^-$ did not allow to find any solution.

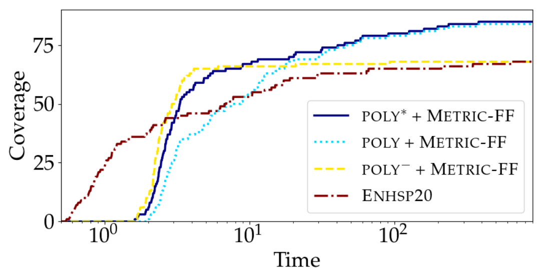

Figure 1: Total number of instances solved by each of the considered planning approaches, over time.

Conclusion

In this work, we introduced a sound but incomplete translation from PDDL+ to numeric planning, with the aim of better investigating how an existing polynomial translation can be optimised (Percassi, Scala, and Vallati 2021). The introduced POLY$^-$ can generate more compact encodings, but its completeness is guaranteed only on a subclass of PDDL+ problems, that we defined as *1-lhs*. Our experimental analysis, run on a variety of PDDL+ benchmarks, confirms the benefits of using the proposed POLY$^-$ translation. Further, it shows that POLY$^-$ can be beneficial also in some cases where the the model is not *1-lhs*, if the solutions space of a problem admits solutions that do not require intermediate states in which two continuous numerical effects that affect the same variable are simultaneously active.

Future work will focus on optimising the considered polynomial translations, and in investigating advanced ways for selecting and combining different translations at runtime, according to the characteristics of the PDDL+ problem to be solved, and of the planning engine in use.

References

Bertolucci, R.; Capitanelli, A.; Maratea, M.; Mastrogiovanni, F.; and Vallati, M. 2019. Automated Planning Encodings for the Manipulation of Articulated Objects in 3D with Gravity. In *AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence*, 135–150.

Cooper, M. C.; Maris, F.; and Régnier, P. 2010. Compilation of a High-level Temporal Planning Language into PDDL 2.1. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, 181–188.

Fox, M.; and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research* 20: 61–124.

Fox, M.; and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *Journal of artificial intelligence research* 27: 235–297.

Henzinger, T. A. 1996. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, 278–292. doi:10.1109/LICS.1996.561342.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J. Artif. Intell. Res.* 20: 291–341.

Keyder, E.; and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research* 36: 547–556.

Palacios, H.; and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research* 35: 623–675.

Percassi, F.; and Gerevini, A. E. 2019. On Compiling Away PDDL3 Soft Trajectory Constraints without Using Automata. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 320–328.

Percassi, F.; Scala, E.; and Vallati, M. 2021. Translations from Discretised PDDL+ to Numeric Planning. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling*. AAAI press.

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

Shin, J.; and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artif. Intell.* 166(1-2): 194–253.

Taig, R.; and Brafman, R. I. 2013. Compiling conformant probabilistic planning problems into classical planning. In *Twenty-Third International Conference on Automated Planning and Scheduling*.

Vallati, M.; Magazzeni, D.; Schutter, B. D.; Chrpa, L.; and McCluskey, T. L. 2016. Efficient Macroscopic Urban Traffic Models for Reducing Congestion: A PDDL+ Planning Approach. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 3188–3194.