Taming Discretised PDDL+ through Multiple Discretisations (Extended Abstract)*

Matteo Cardellini[1], **Marco Maratea**[2], **Francesco Percassi**[3], **Enrico Scala**[4], **Mauro Vallati**[3]

[1]DIBRIS, Università di Genova, Italy
[2]DeMaCS, Università della Calabria, Italy
[3]School of Computing and Engineering, University of Huddersfield, United Kingdom
[4]Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy
matteo.cardellini@edu.unige.it, marco.maratea@unical.it, f.percassi@hud.ac.uk,
enrico.scala@unibs.it, m.vallati@hud.ac.uk

Introduction

PDDL+ is an expressive planning formalism designed to represent problems involving numeric and temporal aspects (Fox and Long 2006). Its key feature is the possibility to model the environment dynamics using events and processes. Events dictate instantaneous and discrete changes when some preconditions hold. Processes specify how the numeric variables change continuously over time according to a set of ordinary differential equations. While these modelling capabilities make this language well-suited for representing realistic and complex scenarios, they also pose significant challenges on the reasoning side.

A well-established approach to breaking down this complexity is based on *discretisation* (Della Penna, Magazzeni, and Mercorio 2012; Percassi, Scala, and Vallati 2023). Specifically, the timeline is segmented according to a *discretisation step* $\delta \in \mathbb{Q}^+$ (shortened as *delta*). Actions can only be executed at timestamps multiples of δ and continuous changes, expressed as derivatives, are discretised into finite differences, i.e., $\dot{x} = v$ becomes $x(t + \delta) = x(t) + v(t) \cdot \delta$. Ideally, δ should be arbitrarily small to ensure a good approximation of the dynamics one intends to represent. However, as δ decreases, the computational intensity of the problem increases significantly. ENHSP mitigates this drawback by decoupling the execution delta (usually smaller), denoted by δ_e, with the planning delta (typically larger) δ_p (Scala et al. 2020). Intuitively, in some domains, an agent may make decisions less frequently than the environment's dynamics, reducing the planning workload.

In this extended abstract, we summarise the work by Cardellini et al. (2024), which introduces a framework for handling discretisation more flexibly. Besides supporting the decoupling between execution and planning delta, the framework allows using several deltas, one for each agent, which can also vary over time. Below, we introduce a motivating example that underscores the need for adopting the discussed framework, we outline a methodology based on reformulation to make it operational, followed by an experimental evaluation.

*We report on the work by Cardellini et al. (2024).

The COOPROVERS Domain

We introduce a new domain called COOPROVERS, where two agents, referred to as rovers (Red and Green), operate at different speeds and must coordinate to achieve a common goal. Figure 1 provides an illustration of the initial state (left) and the goal condition (right), where the rovers are engaged in separate experiments (A and B) located at different distances from their base camp. The rovers need to exchange a tool at the base camp. To ensure safety, they are restricted to move only between the base camp and their respective working zones. Each rover is equipped with a battery and solar panels for recharging, with the Green rover having a lighter, more efficient battery due to the greater distance it needs to cover. The rovers can recharge their batteries while moving or stationary, with the battery level required to remain above 20% for emergency operations. The rovers are also equipped with a container for transporting tools. Despite moving at the same speed, the differences in distances and battery discharge rates necessitate modelling and controlling their movements with different granularities. To model the domain using PDDL+, we define actions and events for rover movements, battery management, tool handling, and charging. These include actions such as $\mathrm{startMoving}$ and $\mathrm{startCharging}$, processes like moving and $\mathrm{discharge}$, and events like $\mathrm{endMovement}$.[1]

Dynamic-Planning Discretised PDDL+

Consider a discrete PDDL+ planning problem extended with an execution delta δ_e. Here we qualitatively define the dynamic planning-discretised PDDL+ problem, abbreviated as PDDLδ+ problem. Such a problem is represented as a pair $\langle \Pi, K_\delta = \langle J, \nabla \rangle \rangle$, where Π is a discrete PDDL+ problem and K_δ is the *discretisation knowledge*. J partitions actions and events into m classes, defining the discretisation variables $\{\delta_1, ..., \delta_m\}$. Intuitively, every δ_j handles a different aspect of the problem by controlling when actions from the j-th partition can be executed. For instance, in our motivating example, J categorises actions and events in $m = 2$ partitions according to the rover they refer to. ∇ controls how these variables change based on actions and events, mapping each to a positive rational number, or a special value indicating the persistence of the current discretisation value.

[1]Available at https://github.com/matteocarde/ICAPS24-Delta.

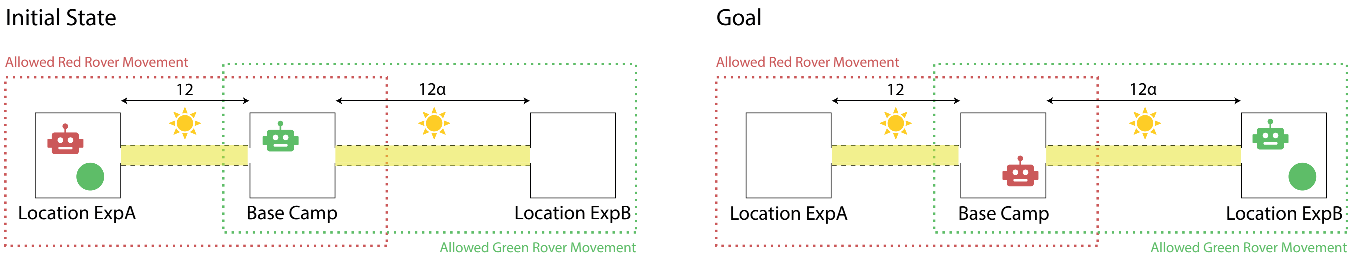

Figure 1: A representation of the initial state and goal condition of the COOPROVERS motivating example.

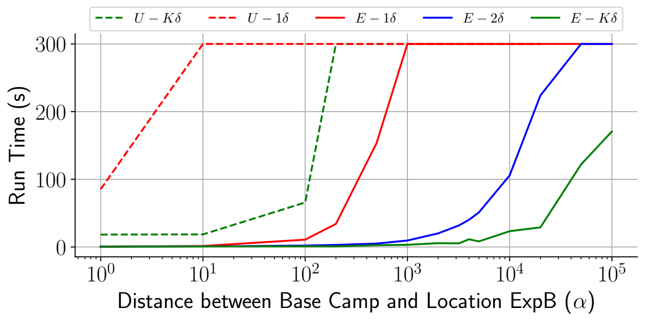

Figure 2: Average runtime (CPU-time seconds) achieved by search approaches implemented in ENHSP (E) and UPMURPHI (U) while relying on different discretisation approaches.

A PDDL+ plan, expressed as $\pi_t = \langle \pi, t_e \rangle$ where π is a sequence of timestamped actions and $t_e \in \mathbb{Q}_0^+$ is the duration of the plan, is valid for $\langle \Pi, K_\delta \rangle$ if (i) π_t is a valid plan for Π and (ii) each action is compliant with K_δ, i.e., it is executed in a timestamp compatible to what prescribed by ∇. We distinguish different levels of control based on how K_δ is defined. If $m = 1$, it is termed *Unitary*, otherwise *Multiple*. If, given a partition, its delta is constant, it is termed *Flat*; otherwise, *Dynamic*. Existing PDDL+ planners only support unitary-static control. We propose a reformulation, FLAT, for handling all the other cases that, given a $\langle \Pi, K_\delta \rangle$, produces an equivalent PDDL+ problem. In essence, the problem Π is extended with numeric variables representing the variables $\{\delta_1, ...\delta_m\}$ updated consistently with ∇ when an action (event) is applied (triggered). These variables restrict the timestamps at which actions can be executed.

Empirical Evaluation

This analysis aims to evaluate the benefits of translating PDDLδ+ to PDDL+ in the COOPROVERS domain, and assess the overall utility of the proposed framework. We considered UPMURPHI (Della Penna, Magazzeni, and Mercorio 2012) and ENHSP (run with different heuristics, i.e., blind, h^{max}, h^{mrp} and h^{aibr}, and searches, i.e., GBFS and A*), varying $\alpha \in \{10^0, ..., 10^5\}$, which controls the distance between the base camp and location B. We also considered different types of discretisation control. In the traditional 1δ approach, both the agent and environment granularity are defined by a single delta, where $\delta_e = \delta_p = 1$. In the 2δ approach, the planner natively decouples the environment and agent, i.e., $\delta_e = 1$ and $\delta_p = 3$ (only supported by ENHSP). Lastly, the K_δ approach employs varying deltas depending on the agent, which can change over time. For instance, movement actions for the red rover are assigned a value of 3, while for the green rover, is 3α. Charging actions, however, maintain the same time scale for both rovers, assigned a value of 30. This scenario falls within the *Multiple-Dynamic* case, as it involves two deltas that can change over time based on the actions taken. Since no planner supports this level of discretisation control, K_δ is enforced through the FLAT reformulation. Figure 2 shows the achieved results as α varies. For ENHSP, we reported the results of the fastest configuration. A more sophisticated discretisation control enables improved scalability in finding a solution. Regarding ENHSP, the K_δ approach, which diversifies the granularity based on executed actions besides the considered rover, is preferable to 2δ allowing us to find a solution for $\alpha = 10^5$. This analysis demonstrated that the proposed framework offers benefits in PDDL+ problems with complex dynamics, and that the FLAT reformulation can make it operational.

Acknowledgements

Francesco Percassi and Mauro Vallati were supported by a UKRI Future Leaders Fellowship [grant number MR/T041196/1]. Enrico Scala was supported by Climate Change AI project (No. IG-2023-174), and by H2020377 project AIPlan4EU (No. 101016442). Marco Maratea was supported by PNRR MUR project PE0000013-FAIR.

References

Cardellini, M.; Maratea, M.; Percassi, F.; Scala, E.; and Vallati, M. 2024. Taming Discretised PDDL+ through Multiple Discretisations. In *Proc. of ICAPS*.

Della Penna, G.; Magazzeni, D.; and Mercorio, F. 2012. A universal planning system for hybrid domains. *Appl. Intell.*, 36(4): 932–959.

Fox, M.; and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *J. Artif. Intell. Res.*, 27: 235–297.

Percassi, F.; Scala, E.; and Vallati, M. 2023. A Practical Approach to Discretised PDDL+ Problems by Translation to Numeric Planning. *J. Artif. Intell. Res.*, 76: 115–162.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramirez, M. 2020. Subgoaling techniques for satisficing and optimal numeric planning. *J. Artif. Intell. Res.*, 68: 691–752.