Shall You Use that PDDL+ Language Feature? An Empirical Analysis on a Real-world Case Study

Anas El Kouaiti,[1] **Francesco Percassi,**[2] **Alessandro Saetti,**[1] **Mauro Vallati**[2]

[1] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy.
[2] School of Computing and Engineering, University of Huddersfield, United Kingdom.
a.elkouaiti@studenti.unibs.it, f.percassi@hud.ac.uk, alessandro.saetti@unibs.it, m.vallati@hud.ac.uk

Abstract

Automated planning is the field of AI that focuses on identifying sequences of actions allowing one to reach a goal state from a given initial state. To support the use of planning techniques in challenging real-world applications, that requires the ability to reason in terms of hybrid discrete and continuous changes, expressive languages such as PDDL+ have been introduced. PDDL+ includes a number of features designed to improve the readability and conciseness of the resulting knowledge models, but that are commonly doubted to have detrimental impact on the performance of domain-independent searches and heuristics.

To shed some light on the extent of the impact that some of these language features can have on well-known planning techniques, in this paper we perform an empirical analysis using challenging models from a real-world application, and a range of search and heuristics approaches. Surprisingly, our analysis indicates that the use of assignments and conditional effects, usually deemed to be detrimental to planning performance, positively affects the performance of the considered techniques.

Introduction

Automated planning is a prominent Artificial Intelligence challenge, which is concerned with the problem of finding a sequence of actions that can bring the agent into some goal state from a given initial condition. Automated planning is exploited in many real-world applications as it is a common capability requirement for intelligent autonomous agents (Ghallab, Nau, and Traverso 2016; McCluskey, Vaquero, and Vallati 2017). Recent example application domains include drilling (Fox et al. 2018), train dispatching (Cardellini et al. 2021), unmanned aerial vehicle control (Kiam et al. 2020) and pharmacokinetic optimisation (Alaboud and Coles 2019).

Real-world applications often require the ability to accurately represent aspects of the environment. This is commonly achieved by exploiting mixed representations that can model both continuous and discrete changes. In response to this need, the PDDL+ language was developed to facilitate the concise encoding of hybrid models for automated planning (Fox and Long 2006). The availability of this standard language supported the design and development of domain-independent search and heuristics, providing "off-the-shelf" technology that can be quickly used.

Notably, PDDL+ models are amongst the most advanced models of systems and the resulting problems are notoriously difficult for domain-independent planning engines to cope with. A well-established strategy for tackling hybrid PDDL+ problems involves breaking down complexity through discretisation (Della Penna, Magazzeni, and Mercorio 2012; Percassi, Scala, and Vallati 2023). This technique assumes that time is discrete, as are the actual numeric changes, contributing to a more manageable problem formulation. This allows for the handling and reasoning of highly complex environmental dynamics.

Complexity can also be exacerbated by the use of language features that have been designed to improve readability and maintenance for knowledge engineers, but that are poorly supported by existing planning engines or have the potential to make the search space more difficult to explore. Considering less expressive languages from the PDDL family, there is indeed a wealth of work that focuses on reformulating knowledge models by removing the use of some poorly supported language features (Helmert 2009; Ceriani and Gerevini 2015; Percassi and Gerevini 2019).

With the aim of supporting the knowledge engineering process of PDDL+ models, in this paper we empirically assess the impact of challenging language features on a range of domain-independent search and heuristic techniques. This assessment is carried out by considering a real-world application of planning in urban traffic control, building upon the work by El Kouaiti et al. (2024), utilising historical data for analysis, and introducing compact compilations to remove the target features from the model. In terms of considered language features, the focus is on *assignments*, which allows the direct assignment of a numeric value to a variable, and *conditional effects*, which can extend the effects of a performed action if some additional condition, beyond the preconditions, holds when applied. These two features of the PDDL+ language are commonly believed to have a detrimental impact on planning performance; however, they are deemed useful for knowledge engineers as they contribute to a more concise and readable representation. Contrary to common beliefs in the field, our analysis reveals that the utilisation of these features can have

a positive impact on performance, especially in addressing challenging real-world problems. Hence, their exploitation is strongly recommended to support both knowledge representation and problem-solving.

The PDDL+ Language

The planning research community has designed several languages to describe different classes of planning problems. One of the most prominent languages is PDDL+ (Fox and Long 2006), which encompasses a spectrum of very expressive constructs and allows the modelling of continuous and discrete changes. One major feature of this language is that it allows to model separately the changes that are induced by the agent's action on the world, and the exogenous changes that can occur in the world as a result of certain conditions or as a consequence of the flow of time. This separation allows to sharply isolate the modelling of the agent from the physics of the environment within which the agent operates, hence supporting the use of the resulting knowledge model also for the sake of simulation and validation (Bhatnagar et al. 2023).

A PDDL+ planning problem is formally defined by a tuple $\Pi = \langle \mathcal{F}, \mathcal{X}, \mathcal{I}, \mathcal{G}, \mathcal{A}, \mathcal{E}, \mathcal{P} \rangle$ in which each element is detailed as follows. $\mathcal{F}$ and $\mathcal{X}$ are sets of Boolean and numeric variables, respectively; the domain of a Boolean variable is $\mathbb{B} = \{\top, \bot\}$ where $\top$ and $\bot$ are the logical true and false, respectively; the domain of numeric variable is $\mathbb{Q}$.

$\mathcal{I}$ is the description of the initial state, expressed as a full assignment to all variables in X and F. $\mathcal{G}$ is the description of the goal, expressed as a formula.

$\mathcal{A}$ and $\mathcal{E}$ are the sets of actions and events, respectively, sharing the same syntax. An action or event is a pair $\langle p, e \rangle$, where p is a propositional formula using standard connectives from logic involving numeric and Boolean conditions, and e is a set of Boolean or numeric effects. Boolean conditions are of the form $\langle f = b \rangle$ with $f \in F$ and $b \in \mathbb{B}$. Numeric conditions are of the form $\langle \xi \bowtie 0 \rangle$, where ξ is a numeric expression over X and $\mathbb{Q}$, and $\bowtie \in \{\leq, <, =, >, \geq\}$. A Boolean assignment has the form $\langle f := b \rangle$, where $f \in F$ and $b \in \mathbb{B}$. A numeric assignment has the form $\langle op, x, \xi \rangle$, where $op \in \{asgn, inc, dec\}$, $x \in X$, and ξ is a numeric expression. Specifically, op can be the contraction of the keywords $assign$ ($x' := \xi$), $increase$ ($x' := x + \xi$) and $decrease$ ($x' := x - \xi$) where x' and x are the value assumed by the affected numeric variable after and before the application of the action/event, respectively.

P is a set of processes, and a process is a pair $\langle p, e' \rangle$, where p is a propositional formula involving numeric and Boolean conditions, and e' is a set of continuous numeric effects expressed as pairs $\langle x, \xi \rangle$, where $x \in X$ and ξ is a numeric expression defined as above. In the continuous formulation of PDDL+ ξ represents the additive contribution to the first derivative of x as time flows continuously. In the discrete setting, that is the semantics adopted in this work and by the majority of planning engines, ξ represents the finite difference of x when time flows by δ, i.e., $x' = x + \xi \cdot \delta$.

Note that the essential difference between $\mathcal{A}$ and $\mathcal{E} \cup \mathcal{P}$ is that actions in $\mathcal{A}$ prescribe changes that take place as the agent's choice, whereas the elements in $\mathcal{E} \cup \mathcal{P}$ prescribe changes outside the control of the agent. Also, note that

```
1  (:action action_name
2   :parameters (par1 - type1 par2 - type2 ... )
3   :precondition (condition_formula)
4   :effect
5     (and
6       (effect1) (effect2) ...
7       (assign (numVar) 3.0)
8       (when (cond) (eff))
9       ...))
```

Figure 1: PDDL syntax of an action with a conditional effect.

PDDL+ problems can be described compactly by means of a *lifted* representation. Lifted representations allow the representation of actions, processes, and events with free-typed variables, which need to be instantiated with objects specific to a particular instance of the problem. The lifted representation supports the knowledge engineering process of automated planning applications by providing a concise and easy-to-maintain representation of the knowledge needed by the automated reasoner (McCluskey, Vaquero, and Vallati 2017). The interested reader is referred to (Fox and Long 2006) for additional details on PDDL+.

A PDDL+ plan is a sequence of timed actions $\langle a, t \rangle$ where $a \in \mathcal{A}$ and t is a non-negative value representing the timestamp in which a is executed. A plan is also coupled with a non-negative value representing its duration.

Solving a PDDL+ problem Π corresponds to the task of finding a plan for Π such that all actions are applicable at the corresponding time and the goal condition is satisfied at the end of the plan. The states in which the actions are applied are the result of a combination of effects of continuously changing variables (given by the active processes through time) and discrete changes happening for effects of actions that get applied or events that get triggered.

Conditional Effects

Conditional effects are an expressive PDDL language feature utilised for defining *state-dependent* effects in the action model. In essence, a conditional effect of an action represents an effect that occurs only when an additional condition holds at the time when the action is applied. Widely employed in complicated scenarios, conditional effects serve as a valuable tool for compactly representing complex application domains.

An example of how a conditional effect is defined in a PDDL action is provided in Figure 1. It is specified using the keyword `when`, and the effect (`eff`) takes place if and only if the condition (`cond`) holds when the action is applied. Otherwise, the conditional effect is ignored.

Due to the impact that conditional effects can have on search spaces, the traditional approach in literature for handling them is to compile them away (Gazen and Knoblock 1997; Nebel 2000).

Assignments

Assignments is a language feature introduced to support numeric reasoning in PDDL. An assignment is a statement that is defined as an effect of an action model, to indicate that as

```
(:action changeConf
 :parameters (?p - stage ?j - junction ?c1 ?c2 -
      configuration)
 :precondition (and
    (inter ?p)
    (controllable ?j)
    (endcycle ?j ?p)
    (availableconf ?j ?c2)
    (activeconf ?j ?c1)
    (not (activeconf ?i ?c2)))
 :effect (and
    (not (activeconf ?j ?c1))
    (activeconf ?i ?c2)))
```

Figure 2: PDDL+ *changeConf* action for changing the configuration of the junction j from c_1 to c_2.

```
(:action changeLimit
 :parameters (?p - stage ?j - junction ?l - limit)
 :precondition
   (and
     (inter ?p)
     (configurable ?j ?p)
     (not (= (cyclelimit ?j) (conflimit ?l))))
  :effect
    (and
      (assign (cyclelimit ?j) (conflimit ?l))
      (not (configurable ?j ?p))))
```

Figure 3: The *changeLimit* action making use of an assignment statement for changing the minimum number of times $?l$ a selected configuration must be repeated on junction j. This action can only be executed during the last intergreen of a cycle.

a result of the action execution, a numeric variable is changing its value to a new one. An example of an assignment is provided in Figure 1, where `(assign (numVar) 3.0)` indicated that `numVar` value is set to 3.0 in the state resulting from the action execution.

Case Study

In this work, we perform our empirical analysis on a version of the models presented in El Kouaiti et al. (2024), namely VARE. This model extends the one introduced by McCluskey and Vallati (2017) to address traffic signal optimisation through automated planning, in a way that ensures deployability. We first provide a brief description of the considered model, and then discuss how the potentially problematic PDDL+ language features of assignments and conditional effects have been compiled away.

A region of the road network is represented by a directed graph, where edges stand for road links and vertices stand for junctions. One special vertex is used for representing the *outside* of the modelled region. Intuitively, vehicles enter (leave) the network via road links connected with the outside. Each link has a given maximum *capacity*, i.e., the maximum number of vehicles that can be, at the same time, in the corresponding road, and the current number of vehicles of a road link, which is denoted as *occupancy*.

Traffic in junctions is regulated by *turn rates*, defined using a dedicated predicate, between couples of road links. Given two links r_x, r_y, a junction j, and a traffic signal stage p such that r_x is an incoming link to the junction i, r_y is an outgoing link from i, and the flow is active (i.e., the corresponding signal light is on green) during stage p, the turn rate associated with r_x, r_y, j and p, stands for the number of vehicles that can leave r_x, pass through j and enter r_y per time unit. Notably, turn rates are defined only for permitted traffic movements. The turn rate value is given in "passenger car units" PCUs per second. For the sake of simplicity, we assume that vehicles moving in the same direction will occupy the correct lane, thus avoiding the blockage of other vehicles travelling in different directions.

Junctions are associated with a sequence of traffic signal stages. The *next* predicate is used to define the sequence of stages. The active traffic signal stage determines the turn rates corresponding to the green lights. The PDDL+ model includes, for each junction, a set of cycle configurations that can be used to optimise traffic conditions. Let j be a junction, and let $\mathcal{S}_j = \langle s_1, ..., s_{k_j} \rangle$ be the sequence of stages associated with j, also referred to as a cycle. Additionally, let $\mathcal{G}_j = \{gt_1, ..., gt_{k_j}\}$ be a set of numeric variables that track the set duration of each stage within j. A *cycle configuration of* j is a complete assignment over $\mathcal{G}_j$. E.g., given a junction j having a cycle involving 3 stages $\mathcal{S}_j = \langle s_1, s_2, s_3 \rangle$, a cycle configuration of j is $\{\langle gt_1 = 30 \rangle, \langle gt_2 = 30 \rangle, \langle gt_3 = 30 \rangle\}$, assigning an uniform duration, 30 secs, to each of the three stages. Each stage ends with an "intergreen" period, which is the time required for a signal to change to green while allowing for stacked vehicles in the middle of the junction to clear and/or provide time for pedestrian crossings.

PDDL+ processes are used for modelling the flows of vehicles described by turn rates, that are activated when a corresponding traffic signal phase is on green. Dedicated processes are also used to measure the time spent on green by the traffic signal stages (or intergreens) on the considered junctions. PDDL+ events are used to stop flows of vehicles when the receiving link is completely full or the discharging link is empty.

The actions under the control of the agent for affecting the urban network are *changeConf* and *changeLimit* whose definitions are provided in Figure 2 and 3, respectively. The action *changeConf* is used to allow the planning system to change the current running configuration $c1$ to $c2$ on a given junction j, while the action *changeLimit* is used to define the minimum number of times the selected configuration must be repeated before allowing further changes.[1]

Plan Example

To illustrate the structure of traffic signal strategy that can be generated by reasoning upon the described model, Figure 4 provided an excerpt of a PDDL+ plan. The listing is organised as a sequence of timestamped actions (`timestamp: action`) closed by a special tag denoting the duration of the

[1]The complete domain model can be found here: https://github.com/anas-elkouaiti/utc-models-deployable

```
362.0: (changeConf j5_stage4 j5 conf_j5_1 conf_j5_2)
362.0: (changeLimit j5_stage4 j5 lim9)
464.0: (changeConf j4_stage4 j4 conf_j4_1 conf_j4_3)
464.0: (changeLimit j4_stage4 j4 lim8)
474.0: (changeConf j6_stage3 j6 conf_j6_1 conf_j6_2)
474.0: (changeLimit j6_stage3 j6 lim8)
477.0: (changeConf j2_stage5 j2 conf_j2_1 conf_j2_3)
628.0: (changeConf j1_stage4 j1 conf_j1_1 conf_j1_3)
628.0: (changeLimit j1_stage4 j1 lim10)
637.0: (changeConf j3_stage6 j3 conf_j3_1 conf_j3_6)
637.0: (changeLimit j3_stage6 j3 lim7)
969.0: (changeConf j2_stage5 j2 conf_j2_3 conf_j2_5)
969.0: (changeLimit j2_stage5 j2 lim9)
1370.0: (changeConf j5_stage4 j5 conf_j5_2 conf_j5_1)
...
2049.0: @PlanEND
```

Figure 4: Example of a PDDL+ plan encoding a signal traffic strategy based on the presented model.

```
(:event triggerChange
 :parameters
  (?p1 ?p2 - stage ?j - junction)
 :precondition
   (and
     (inter ?p2)
     (contains ?j ?p1)
     (next ?p ?p2)
     (>= (intertime ?j)
     (- (interlimit ?p1) 0.1)))
 :effect
   (and
     (not (inter ?p1))
     (active ?p2)
     (assign (intertime ?j) 0)))
```

Figure 5: PDDL+ *triggerChange* event for transitioning from a stage p_1 to the next one p_2 over junction j.

plan (`@PlanEnd`).

The plan reveals that the problem's goal is achieved in a duration of 2049.0 time units and that until time 362.0, no action is taken to change the configuration set in the initial state. Notably, all actions to change the running configuration on a junction are immediately followed by an action to set the minimum number of cycles for which it must be kept. For example, at time 362.0, the configuration of junction $j5$ is changed from `conf_j5_1` to `conf_j5_2`. This decision occurs during stage `stage_4` as it is the final stage of the cycle. Subsequently, an action is taken to set the minimum number of cycles for junction $j5$ to 9 (`lim9`).

Language Features and Compilations

We focus on two features of the language in this analysis, namely *assignments* and *conditional effects*. In the considered model, assignments are used to reset to 0 numeric variables or to update numeric values due to some changes in the configuration. The first case, exemplified in Figure 5 in the *triggerChange* event, can be compiled away by substituting the assignment effect with a subtraction of the numeric value by itself, such as `(decrease (intertime ?j) (intertime ?j))`. In our model, this type of assignment appears three times.

```
(:action changeLimit
 :parameters (?p - stage ?i - junction ?l1 ?l2 -
     limit)
 :precondition
   (and
     (inter ?p)
     (configurable ?i ?p)
     (activelimit ?i ?l1)
     (not (activelimit ?i ?l2)))
 :effect
   (and
     (not (activelimit ?i ?l1))
     (activelimit ?i ?l2)
     (not (configurable ?i ?p))))
```

Figure 6: The reformulated *changeLimit* action where the assignment statement is compiled away.

The more complicated assignment case refers to the update of a numeric value, that in the considered model is used in the *changeLimit* action shown in Figure 3. In this case, the reformulation requires to explicitly specify in the initial state the allowed *limit* values, e.g., 4, and the action is modified so that a limit can be activated on a considered junction, as shown in Figure 6. This compilation requires also the modification of the parameters list of the action, that now needs to include both the currently active limit $?l1$ and the limit to be assigned $?l2$. This class of assignment statements is only used once in the reference domain model.

We can now turn our attention to the language feature of conditional effects, used to allow some effects to be implemented only if some additional conditions are met in the environment. The only occasion where such a feature is used in the domain model is in the *trigger-change* event, shown in Figure 7. Conditional effects, denoted by the keyword `when`, are used to allow the event to take appropriate actions according to when, in a cycle, it has been triggered. The first conditional effect, i.e., `(when (endcycle ?i ?p1) (increase (countcycle ?i) 1))`, is employed to increment the variable that keeps track of how many times the configuration, currently selected for junction j, has been executed. The second conditional effect, i.e., `(when (endcycle ?i ?p) (not (configurable ?i ?p))`, not only narrows down the search space but also maintains the integrity of the problem's correctness constraints, preventing the *changeLimit* action in invalid stages.

For the reformulation of this language feature, we followed a methodology similar to an approach used in literature (Nebel 2000), which consists of multiplying out the construct, in this case, the *trigger-change* event, according to all the possible combinations of conditional effects. In the specific case, this leads to the encoding of 3 events: the original *trigger-change* with no conditional effects, and a new one event for each potential branch of the starting conditional effect.

It is worth noting that some of the considered reformulations lead to an increased number of PDDL+ constructs

or an increased number of parameters for some of the actions, which can have significant repercussions in terms of the size of the grounded representation (Scala and Vallati 2021). However, in works focusing on less expressive languages of the PDDL family, the performance benefits have greatly outweighed the potential issues.

Experimental Analysis

The experimental analysis aims at assessing the impact of the considered language features on a range of search strategies and heuristics.

We use an extended version of the benchmark used by Percassi et al. (2023). The modelled urban network area is situated in West Yorkshire, United Kingdom, specifically within the Kirklees council. It consists of a major corridor that links the Huddersfield ring road with the M1 highway and the southern part of the Kirklees council. It is heavily used by commuters and by delivery vans to get to the centre of the Huddersfield town, or to move between the M62 and the M1 highways. The corridor is approximately 1.3 kilometres long, and consists of 6 junctions and 34 road links. Each junction has between 4 and 6 stages, and between 10 and 17 valid traffic movements. A schema of the considered region is shown in Figure 8, in terms of links, junctions, and connections with the outside region. For this region, we have access to historical data collected by deployed sensors and the running traffic control infrastructure.

On this area, we consider six scenarios in two distinct days: the 26th (referred to as day *A*), which is a Wednesday, and the 30th (referred to as day *B*), a Sunday, both in January 2022. Each day was examined at three different time slots: the morning peak hour at 8:30 am, noon at 12:30 pm, and the evening peak hour at 4:30 pm. This provides variability in terms of traffic volumes and conditions. Further, we include an additional scenario involving exceptional traffic circumstances, pertaining to a concert held at John Smith's Stadium on Tuesday the 20th of June 2023, which attracted an approximate audience of $30,000$ people. The time considered is 4:00 pm, which is before the start of the concert. This scenario is interesting because there is a clash between commuters leaving the town and spectators arriving at the concert, creating two opposed traffic demands.

Following the work of Percassi et al. (2023), for each scenario we generated five planning problems, progressively expanding the set of explicitly considered corridor links in the goal. This allows for the enforcement of different behaviours in the planning systems and introduces more challenging goals to be achieved. For each link, we specified that a minimum of 350 vehicles should move through it as soon as possible. In terms of cycle configurations to be used to solve the planning problem, we generated six configurations per junction of the network, where in turn one stage is maximised. All the configurations have the same length of 90 seconds, to comply with legal requirements and to ensure that synchronisation between the junctions is maintained throughout the plan.

Experiments were run on a machine with a 2.3 GHz Intel Xeon Gold 6140M CPU and 8 GB of RAM. As planning engine, we use ENHPS (Scala et al. 2020a) version 20.

```
(:event trigger-change
:parameters
    (?p ?p1 - stage ?i - junction)
 :precondition (and
   (inter ?p)
   (contains ?i ?p)
   (next ?p ?p1)
   (>= (intertime ?i)
   (- (interlimit ?p) 0.1)))
  :effect (and
    (not (inter ?p))
    (active ?p1)
    (decrease (intertime ?i) (intertime ?i))
    (when (endcycle ?i ?p1)
        (increase (countcycle ?i) 1))
    (when (endcycle ?i ?p)
        (not (configurable ?i ?p)))))
```

Figure 7: The *trigger-change* event that is triggered when an intergreen finished, to allow moving to the next stage for the junction $?i$.

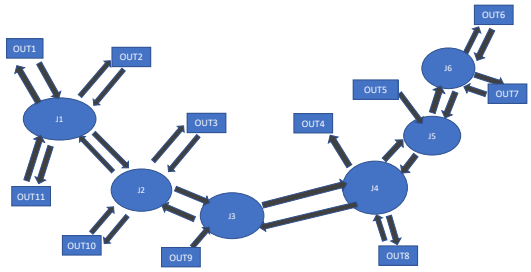

Figure 8: A simplified overview of the modelled urban region, in terms of junctions (circles), links, and boundaries (rectangles). For the sake of readability, the map is not correctly scaled.

This system implements a large number of heuristics and search techniques, providing the ideal testbed for assessing the impact of PDDL+ features on performance. The considered search strategies are greedy best-first search (GBFS) and A*, and the adopted heuristics include h^{add} (Scala et al. 2016), h^{max} (Scala et al. 2016), and h^{mrp} (Scala et al. 2020b). These are state-of-the-art approaches in hybrid planning. We also considered blind search, implemented by ENHPS, but no problem has been solved using that technique.

Table 1 provides an overview of the results in terms of the number of solved instances, and IPC score for the quality of generated plans, i.e., their durations, expanded nodes, and runtime. Here the IPC score is calculated as in the IPC 2014 (Vallati, Chrpa, and McCluskey 2018). In short, for each planning instance, a considered combination of model and search configuration can get between 1.0 and 0.0 points. 1.0 is awarded to the combination that obtained the best result amongst all the considered, 0.0 to the combination that fails to solve the instance, and a value in between according to how close is the result of the combination to the best observed. The better the result, the closer the score gets to

	h^{max}		h^{mrp}		h^{add}		Σ (210)
	GBFS	A⋆	GBFS	A⋆	GBFS	A⋆	
Number of Solved Instances							
B (35)	35	0	34	0	1	0	70
-ce	35	0	34	0	1	0	70
-asgn	34	0	31	0	0	0	65
-ce -asgn	30	0	25	0	0	0	55
Σ (105)	**134**	0	124	0	2	0	
Score(Quality)							
B (35)	32.91	0	32.62	0	1.00	0	**66.53**
-ce	32.94	0	32.49	0	1.00	0	65.43
-asgn	31.99	0	29.62	0	0.00	0	61.61
-ce -asgn	22.09	0	18.34	0	0.00	0	40.43
Σ	**119.93**	0	113.07	0	2.00	0	
Score(Expanded Nodes)							
B (35)	20.00	0	12.80	0	0.02	0	32.82
-ce	20.95	0	11.10	0	0.02	0	32.07
-asgn	21.52	0	12.71	0	0.00	0	**34.23**
-ce -asgn	13.10	0	5.75	0	0.00	0	18.85
Σ	**75.57**	0	42.36	0	0.05	0	
Score(Runtime)							
B (35)	18.42	0	15.02	0	0.12	0	**33.56**
-ce	18.69	0	14.66	0	0.16	0	33.51
-asgn	13.77	0	10.20	0	0.00	0	23.97
-ce -asgn	6.78	0	5.07	0	0.00	0	11.85
Σ	**57.66**	0	44.96	0	0.28	0	

Table 1: Results in terms of coverage, quality of generated plans, expanded nodes, and planning time for each considered domain model. "B" denotes the basic formulation while *-ce* and *-asgn*, denotes, respectively, a variant where conditional effects and/or numeric assignments have been compiled away. Aggregated results are presented per each model formulation, and per each search combination. Bold indicates the best aggregated results.

1.0. The total IPC score is the sum of the scores achieved on each considered instance.

As a first observation, it is easy to notice that the use of A⋆ does not allow solving any of the considered instances. On one hand, this is not surprising, considering the complexity and challenges of PDDL+ planning, which require reasoning in terms of both continuous and discrete changes. On the other hand, this seems to suggest that there is no silver bullet when it comes to planning models and language features: improvements can be remarkable but not game-changing.

When it comes to the performance that the models with and without the considered language features allow to achieve, some interesting behaviours can be observed. In general terms, it is apparent that the use of conditional effects and assignments does not harm planning performance. This is a very surprising result, as it contradicts the common knowledge that considers such features detrimental to planning performance. This is highlighted by the poor performance, according to all of the considered metrics, of the model when both conditional effects and assignments are compiled away.

Let us now take a closer look at the variability of performance according to the considered metrics. In terms of coverage, i.e., the ability to solve planning instances, numeric assignments are the features that is mostly beneficial, while conditional effects do not appear to have any remarkable impact on the capabilities of the considered search combinations. A similar figure can be drawn when looking at quality scores and runtime, i.e., the time needed by the approach to generate a solution. However, taking a closer look at the way in which the search space is explored gives us a different insight. In terms of expanded nodes, the use of conditional effects has a positive impact on performance: in other words, conditional effects may allow us to find a solution by exploring a smaller chunk of the search space. In the considered domain, this has a limited impact on runtime performance and on the ability to solve planning instances, but it should be noted that in different applications, where states are perhaps larger and more complex, the impact could be significant.

Turning now our attention to the heuristics, it appears that the one providing the best guidance for GBFS in the considered domain is h^{max}. Intuitively, this appears to be reasonable as it allows the search to focus on dealing with the link that requires more attention to get traffic through, as h^{max} focuses on the hardest to achieve goal. In a sense, it may allow the planning system to focus on the bottleneck of the considered network, hence implicitly improving the overall performance.

Summarising, our extensive experimental analysis disproofs the common belief that the use of conditional effects and numeric assignments has a detrimental impact on PDDL+ search and heuristic techniques. On the contrary, we showed that their use can be beneficial and should not be excluded a priori. While we acknowledge that the analysis considers a single domain model, it is worth highlighting that the model is amongst the most complex benchmarks in PDDL+ planning in terms of dynamics of the environment and size of the models, hence it is the most suitable to emphasise performance differences.

Conclusion

With the aim of supporting the knowledge engineering process of complex planing knowledge models, in this paper we empirically assessed the impact on planning performance of two potentially challenging PDDL language features, namely assignments and conditional effects. First, we described compilations to remove such features from a PDDL+ model. Second, we performed an extensive analysis on a range of search and heuristic approaches, focusing on a challenging real-world application leveraging on historical data. The surprising result is that the features, commonly believed to have a detrimental impact on performance, demonstrate a beneficial impact, and their employment should therefore be

considered when PDDL+ planning problems are concerned. Future work will look into extending the analysis to different benchmarks, where the impact of other language features can be assessed.

Acknowledgements

Francesco Percassi and Mauro Vallati were supported by a UKRI Future Leaders Fellowship [grant number MR/T041196/1].

References

Alaboud, F. K.; and Coles, A. 2019. Personalized Medication and Activity Planning in PDDL+. In *Proc. of ICAPS*, 492–500.

Bhatnagar, S.; Guo, R.; McCabe, K.; McCluskey, T. L.; Percassi, F.; and Vallati, M. 2023. Automated Planning for Generating and Simulating Traffic Signal Strategies. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI*, 7119–7122.

Cardellini, M.; Maratea, M.; Vallati, M.; Boleto, G.; and Oneto, L. 2021. In-Station Train Dispatching: A PDDL+ Planning Approach. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling*, 450–458.

Ceriani, L.; and Gerevini, A. E. 2015. Planning with always preferences by compilation into strips with action costs. In *Eighth Annual Symposium on Combinatorial Search*.

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

El Kouaiti, A.; Percassi, F.; Saetti, A.; McCluskey, L.; and Vallati, M. 2024. PDDL+ Models for Deployable yet Effective Traffic Signal Optimisation. In *34th International Conference on Automated Planning and Scheduling*.

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

Fox, M.; Long, D.; Tamboise, G.; and Isangulov, R. 2018. Creating and executing a well construction/operation plan. US Patent App. 15/541,381.

Gazen, B. C.; and Knoblock, C. A. 1997. Combining the Expressivity of UCPOP with the Efficiency of Graphplan. In *Proc. of ECP*, 221–233.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press. ISBN 978-1-107-03727-4.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artif. Intell.*, 173(5-6): 503–535.

Kiam, J. J.; Scala, E.; Javega, M. R.; and Schulte, A. 2020. An AI-Based Planning Framework for HAPS in a Time-Varying Environment. In *Proc. of ICAPS*, 412–420.

McCluskey, T. L.; and Vallati, M. 2017. Embedding Automated Planning within Urban Traffic Management Operations. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS*, 391–399.

McCluskey, T. L.; Vaquero, T. S.; and Vallati, M. 2017. Engineering Knowledge for Automated Planning: Towards a Notion of Quality. In *Proceedings of the Knowledge Capture Conference, K-CAP*, 14:1–14:8.

Nebel, B. 2000. On the Compilability and Expressive Power of Propositional Planning Formalisms. *J. Artif. Intell. Res.*, 12: 271–315.

Percassi, F.; Bhatnagar, S.; Guo, R.; McCabe, K.; McCluskey, L.; and Vallati, M. 2023. An Efficient Heuristic for AI-based Urban Traffic Control. In *8th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*.

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

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. 2016. Interval-based relaxation for general numeric planning. In *ECAI 2016*, 655–663.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramirez, M. 2020a. Subgoaling Techniques for Satisficing and Optimal NumericPlanning. *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 *Proc. of ICAPS 2020*, 226–234. AAAI Press.

Scala, E.; and Vallati, M. 2021. Effective grounding for hybrid planning problems represented in PDDL+. *Knowl. Eng. Rev.*, 36: e9.

Vallati, M.; Chrpa, L.; and McCluskey, T. L. 2018. What you always wanted to know about the deterministic part of the International Planning Competition (IPC) 2014 (but were too afraid to ask). *Knowledge Eng. Review*, 33: e3.