OPEN ACCESS

KING'S College LONDON

King's Research Portal

DOI:
10.3233/978-1-61499-672-9-1185
10.3233/978-1-61499-672-9-1185

Document Version
Publisher's PDF, also known as Version of record

Link to publication record in King's Research Portal

Citation for published version (APA):
Savaş, E., Fox, M., Long, D., & Magazzeni, D. (2016). Planning using actions with control parameters. In *Frontiers in Artificial Intelligence and Applications* (Vol. 285, pp. 1185-1193). (Frontiers in Artificial Intelligence and Applications; Vol. 285). IOS Press. https://doi.org/10.3233/978-1-61499-672-9-1185, https://doi.org/10.3233/978-1-61499-672-9-1185

Citing this paper
Please note that where the full-text provided on King's Research Portal is the Author Accepted Manuscript or Post-Print version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version for pagination, volume/issue, and date of publication details. And where the final published version is provided on the Research Portal, if citing you are again advised to check the publisher's website for any subsequent corrections.

General rights
Copyright and moral rights for the publications made accessible in the Research Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

•Users may download and print one copy of any publication from the Research Portal for the purpose of private study or research.
•You may not further distribute the material or use it for any profit-making activity or commercial gain
•You may freely distribute the URL identifying the publication in the Research Portal

Take down policy
If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 26. Feb. 2026

ECAI 2016
G.A. Kaminka et al. (Eds.)

© 2016 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/978-1-61499-672-9-1185

Planning Using Actions with Control Parameters

Emre Savaş and **Maria Fox** and **Derek Long** and **Daniele Magazzeni**[1]

Abstract. Although PDDL is an expressive modelling language, a significant limitation is imposed on the structure of actions: the parameters of actions are restricted to values from finite (in fact, explicitly enumerated) domains. There is one exception to this, introduced in PDDL2.1, which is that durative actions may have durations that are chosen (possibly subject to explicit constraints in the action models) by the planner. A motivation for this limitation is that it ensures that the set of grounded actions is finite and, ignoring duration, the branching factor of action choices at a state is therefore finite. Although the duration parameter can make this choice infinite, very few planners support this possibility, but restrict themselves to durative actions with fixed durations. In this paper we motivate a proposed extension to PDDL to allow actions with infinite domain parameters, which we call *control parameters*. We illustrate reasons for using this modelling feature and then describe a planning approach that can handle domains that exploit it, implemented in a new planner, POPCORN (Partial-Order Planning with Constrained Real Numerics). We show that this approach scales to solve interesting problems.

1 INTRODUCTION

PDDL is a powerful modelling language, but one limitation of the models it supports is that action parameters are only allowed to be drawn from finite domains. These domains are enumerated in the problem descriptions and are typically relatively small (a few tens of objects would already be unusual). A consequence of this constraint is that action schemas can be grounded, by substitution of parameters with values in all possible ways, to yield a finite set of actions for each problem instance. There is one parameter that is an exception to this: the duration of a durative action may be left flexible, so that the planner can choose its value (possibly subject to constraints). In fact, this is a far-reaching choice, since in combination with duration-dependent effects, it can make it possible to model domains in which there is an infinite branching choice of actions applicable in a state. Although the addition of numeric state variables to the classical propositional language allows construction of an infinite state space, without the flexible duration parameter, the state space is always *locally finite*, which is to say that only finitely many states are reachable from a given initial state, using plans bounded by a given finite length. The introduction of the flexible duration parameter changes this, so that infinitely many states may be reachable by plans even bounded to length one. This observation makes clear that the introduction of a flexible duration parameter fundamentally changes the possible structure of state spaces.

In this paper, we consider a generalisation of the duration parameter, allowing actions to take multiple parameters from infinite domains. We call these parameters *control parameters*, since they often represent physical control parameters that a planner might select in order to make an action have a desired effect (or intensity of effect). Examples include volume settings, velocities, power use, volumes or sizes. As a concrete example, illustrating that duration is not always an appropriate surrogate for control parameters, consider the action of withdrawing cash from an ATM (or cashpoint). Execution of this action involves choosing the value of the withdrawal. Although there is probably some small element of the duration of a withdrawal that is affected by the amount withdrawn, the transaction is, for all practical purposes, best modelled as a constant duration action that has an effect on cash in pocket determined by the selected value of the control parameter: the size of the withdrawal. Current standard PDDL dialects require that either the action be modelled artificially, using duration to substitute for the control, or else that the choice of withdrawal amounts be restricted to some fixed and finite menu of choices.

A slightly more complex example, generalising an example drawn from related work on Kongming [13], is a navigation action for an autonomous underwater vehicle (AUV), equipped with independent motors to drive horizontal movement (the x-direction) and vertical movement (the y-direction). In choosing a single leg of movement, the action requires selection of an x-velocity, a y-velocity and a duration. It is clearly not possible to represent choices in this 3-dimensional choice space using a duration alone.

In this paper, we explore the cashpoint example in more detail to illustrate the problems introduced by the use of control parameters. We then propose a way in which planning can be performed in domains containing actions with these parameters, implemented as our planner, POPCORN (Partial-Order Planning with Continuous Real Numerics), based on the POPF planner [4], extending and expanding the mechanisms in that planner to support control parameters beyond duration. In doing so, we retain the ability in POPF to use temporal actions, including with flexible durations and duration-dependent effects. We demonstrate that our approach scales to tackle interesting problems, considering several domains with various characteristics. Our implementation works with linear constraints, relying on an LP support in similar style to POPF, but we observe that our strategy can be generalised to different constraints, indicating how this could be achieved.

The paper is structured as follows. We survey the related work in the next section. Section 2 gives a more detailed description of the cash point motivating example. In Section 4 we provide technical background on the state representation in POPF, and in Section 5 we describe how we extend it to implement our approach for planning with control parameters in POPCORN. In Section 6 we show how POPCORN scales to solve interesting problems. Section 7 concludes the paper.

[1] Department of Informatics, King's College London, UK, WC2R 2LS, email: {okkes.savas, firstname.lastname}@kcl.ac.uk

2 A MOTIVATING EXAMPLE

We now develop the simple cashpoint example to illustrate some aspects of the use of control parameters. Suppose that we are planning to go to a pub. Initially, we are at home and have only £2 in our pocket. We aim to be at the pub with £20 in our pocket and to have already bought snacks on the way to the pub. The plan for this problem is quite simple: we must go to the cashpoint and collect cash, go to the shop to buy snacks and go to the pub. However, the action to withdraw cash should allow us (as planner) to decide how much to withdraw.

We propose to extend PDDL2.1 [10] so that actions may include an additional field: control parameters. These parameters are typed. We currently assume that they are numeric parameters, although our syntax supports other types. The description of the cash withdrawal action is shown in Figure 1. We list all control parameters except `?duration` in a new field, `:control`. Our language has the expressive power to state the numeric types of the control parameters. A control parameter can defined as an `integer`, `number`, or as a `boolean`. In the example, the `?cash` control parameter is defined as a number in the `WithdrawCash` action. Each instantiation of the action has its own instance of this parameter and each can be given a value independently of the others. The parameter is tied into the state through action preconditions: in this case, the parameter is restricted to lie between a minimum withdrawal and the limit of the machine. It also appears in the effects, increasing what is in the pocket and decreasing what is left in the machine.

```
(:durative-action WithdrawCash
 :parameters (?p - person ?a - location
?m - machine)
 :control (?cash - number)
 :duration (= ?duration 2)
 :condition (and (at start (at ?p ?a))
  (over all (at ?p ?a)) (at start (located ?m ?a))
  (at start (>= ?cash 5))
  (at start (<= ?cash (balance ?m)))
  (at start (canWithdraw ?p ?m)))
 :effect (and
  (at start (decrease (balance ?m) ?cash))
  (at end (increase (inpocket ?p) ?cash))))

(:durative-action BuySnacks
 :parameters (?p - person ?a - location)
 :duration (= ?duration 1)
 :condition (and  (at start (at ?p ?a))
  (over all (at ?p ?a)) (at start (snacksAt ?a))
  (at start (>= (inPocket ?p) 3)))
 :effect (and (at end (decrease (inPocket ?p) 3))
  (at end (gotSnacks ?p))))
```

Figure 1. Main actions of the cash point domain.

An example problem for this domain is shown in Figure 2. In this example there are two ATMs, each with a limited balance, and one shop at which we can buy snacks. In addition, Figure 2 shows the (optional) metric objective of the problem that can play an important role in the valuation of the control parameter `?cash`. The domain also includes a move action to allow movement between locations. Intuitively, to optimise the plan metric, we should withdraw sufficient cash to buy snacks and to have £20 at the pub. We would not want to withdraw more or less cash than required when at the cash point.

This problem could be modelled without control parameters, by discretising the amount that can be withdrawn into a finite set of

```
(:init (at Joe home) (snacksAt store)
 (= (inPocket Joe) 2)
 (canWithdraw Joe atm1) (canWithdraw Joe atm2)
 (located atm1 bank) (located atm2 bank)
 (= (balance atm1) 50) (= (balance atm2) 100))
(:goal (and (>= (inPocket Joe) 20)
 (gotSnacks Joe) (at Joe pub)))
(:metric minimize (inPocket Joe))
```

Figure 2. The initial state of the cash point problem.

possible values (say £5, £10, £20, £50). If we consider a forward searching planner, relying on a standard relaxation-based heuristic, the planner will choose a withdrawal action to meet the demands of the largest goal (the £20 goal), but the relaxation will ignore the effect of the future snacks purchase. This means, the planner will not see that it actually needs to withdraw £25, and it is quite likely that the final plan will involve returning to the cash point to make a second withdrawal (the precise behaviour will depend on the search strategy, but this is what happens in POPF, for example). In the discretised model, the optimal plan (according to the metric) is to withdraw £25 using two withdrawals before moving away, while the *shortest* plan withdraws £50. When using the control parameter model, the planner can decide to withdraw exactly £23 in a single action, yielding both the shortest and optimal plan.

The particular challenge we consider in this paper, is how the value of the control parameter can be chosen by the planner, in particular when performing a forward state space search. In this example, the amount of cash we want to withdraw depends on which actions we apply after visiting the cash point. Early assignment of the value of a control parameter may lead to generation of poor plans. For instance, assigning a value to `?cash` before buying snacks would result in visiting the cash point twice. Therefore, the determination of the amount to withdraw should be made at a later stage in the plan construction (possibly after the entire action sequence has been constructed).

Our planner, POPCORN, builds up all the linear constraints acting on the control parameter `?cash` and (checking numeric consistency of every state visited using the linear program) until the goal state is reached. Then, the planner calls the final linear program to optimise all variables, i.e. `?cash`, subject to the metric objective of the problem. Since the metric objective is to minimise the `inpocket` state variable in this example, the planner chooses the minimum bound of this variable as its value. We describe this process in detail in the remainder of the paper.

3 RELATED WORK

Work exploring the use of control parameters in domain independent planning is limited. The duration parameter was introduced in PDDL2.1 [10], but very few planners can plan with actions that have both flexible durations and duration-dependent effects, such as POPF [4] and its close cousins, COLIN [3] and OPTIC [1]. One way to understand the behaviour of these planners with respect to the choice of durations will be presented below, since it is relevant to the treatment of control parameters in POPCORN.

Williams and co-authors have explored the management of control parameters in several different ways. For example, the Kongming planner [13] uses an extended PDDL representation, also adding a new field to actions in order to capture the control parameters. It captures the interaction of the dynamic continuous variables with *flow*

tubes produced at each action layer. The planner is based on a Graphplan [2] core and the flow tubes contain control trajectories of the variables as the graph expands over time. The graph is translated to a mixed integer program and solved using CPLEX, rather than by Graphplan search, allowing the planner to handle problems with linear effects. The use of the Graphplan core restricts the representation of time, which is discretised, while the rates of change of process effects are taken as control variables. This approach contrasts with COLIN [3] and POPF planners, where the duration is taken as a variable, while rates of change remain constant. Both approaches use linear programming to solve the numeric constraints and this means that it is not possible to allow both variable durations and variable rates of change, since this would lead to quadratic constraints. Kongming suffers from severe limitation of the number of happenings in the plan, due to the cost of iterated deepening search in the Graphplan-CPLEX solution strategy.

Fernández-González, Karpas and Williams have recently studied planning with continuous control parameters [8]. Their work has considered the main stages in the development of the Scotty planner. It combines the flow tube representation of Kongming with the forward-chaining search and the linear programming used in the COLIN planner. The flow tubes are used to capture continuous effects with control parameters. It uses forward search to overcome the limitation on numbers of happenings in Kongming. The purpose of control parameters in this work (and, to an extent in Kongming, and earlier work using flow tubes [12]) is to convey flexibility of choice to the *executive* [7]. Therefore, Scotty finds *a flexible plan*, in which it leaves the decision of the values of control parameters to an executive during plan execution in order not to invalidate the plan (whereas it finds the timestamps of actions that can also invalidate the plan). In addition, Scotty does not support actions performing discrete numeric change [9]. Also, it is not possible to define duration-independent control parameters: the control parameters all represent rates of change, so they all combine with the duration to determine the final effects of actions.

Pantke, Edelkamp and Herzog have recently presented a PDDL-based multi-agent planning system that reasons about the control parameters in production planning and control [14, 15]. They propose a similar extension to PDDL language, in which the control parameters are listed in the `:parameters` field. They adapt two-stage planning strategy in their work. First, the planner finds partially grounded total-order plans with *lifted* control parameters. Then, the planner find the fully grounded plans by determining the bounds of the variables of the first *n* partially grounded plans (using a mathematical optimisation tool) based on their potential quality. The optimisation tool is not used to check the consistency of a state. Instead, they use the *interval arithmetic* relaxation, which gives the possible values (not the optimal values) at a state. Additionally, the planning horizon is fixed and they do not use any relaxation heuristic during search.

Our motivation in introducing control parameters is slightly different to that of Williams et al. and Fernández-González et al.: we are interested in the planner choosing values to construct a specific plan and to make choices that lead to efficient plans, rather than in conveying flexibility for execution to the executive. Our control parameters do not necessarily represent rates of change (such as the cash withdrawal), and effects of actions can be independent of duration, or not. The work of Pantke et al. focusses only on the production control domain application, whereas we consider the domain-independent planning applications in forwards search framework using relaxation-based heuristics.

4 TECHNICAL BACKGROUND

As we have noted, duration can be seen as a special control parameter. POPF and its variants handle this parameter, so it forms a natural starting point from which to generalise to manage other control parameters. We therefore briefly review how the management of duration is achieved in POPF and COLIN.

A central innovation in POPF is the extended representation of a state. In classical forward search planners, the states are valuations over the state variables (which may be boolean values in a propositional planner and boolean or numeric in a metric planner). PDDL2.1 planners must extend this representation to include record of which durative actions are executing and when they started: we call this a *temporal state*. POPF further extends this representation, so that it searches in a space in which its states (P-states) represent *sets* of temporal states.

Definition 1 *A P-state, S, is a tuple $\langle F, V, B, Q, P, C, T\rangle$, where:*

- F *is a partial valuation over boolean state variables.*
- V *is a partial valuation over numeric state variables.*
- B *maps the numeric state variables to upper and lower bounds (which might be infinite). Variables with a value defined in V have upper and lower bounds equal to this value.*
- Q *is a set of actions which are started but not yet finished.*
- P *is the collection of steps added to the plan to reach state S.*
- C *is a collection of temporal and metric constraints accumulated over the steps in P.*
- T *is the upper and lower bound on the time at which this state must end.*

POPF only imposes a partial ordering on the actions in its plan (through temporal constraints included in C). POPF postpones commitment to ordering of actions when all orderings consistent with C are executable. This means that state variables whose values are not required to satisfy specific precondition constraints might be undetermined, subject to the resolution of the partial ordering of actions that affect them, which is why F and V are partial valuations. Where a continuous or duration-dependent effect is active in a P-state, the constraints in C tie together the bounds on the time in the current state (T) with the bounds on the value of the affected variable (B).

A P-state represents a set of temporal states because there can be many feasible solutions to C. The constraints in C can be purely temporal constraints (managed as a simple temporal network) if there are no continuous or duration-dependent effects in the plan, or a linear program if time and numeric state variables interact [5]. From C, values of B and T are found by solving C for upper and lower bounds on each variable.

POPF manages search through P-states by implementing both an appropriate P-state progression algorithm and also a variant of the relaxed plan heuristic, based on the Temporal Relaxed Planning Graph [6, 5]. The search choices faced by POPF are whether to add a new action start to extend a P-state, or else to complete an executing action (whose end is in Q). In both cases, constraints can be added to C to ensure preconditions are satisfied. In this way, POPF avoids branching on the infinite choice of values of duration, but manages constraints that restrict the possible values of durations, ensuring feasibility of the constraints at each P-state progression.

In order to implement temporal-numeric planning with control parameters POPCORN is built on the POPF planner. The partial-order mechanism in POPF minimises the addition of ordering constraints to avoid early commitment during forward search, in order to achieve

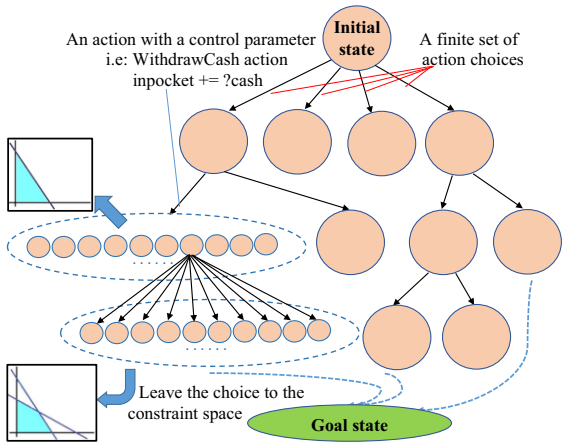


Figure 3. Schematic representation of the search space where there is a control parameter effect. The nodes represent the state reached, and the edges represent the action applied to reach the next state. The graphs in black boxes represent the LP constraint space, which is used to avoid complex branching choice.

flexible plans. The temporal constraints are added as they are needed to meet the preconditions of actions in a possible plan. The existing partial-order mechanism of POPF helps POPCORN to avoid early-commitment in assigning values to the control parameters. As discussed in Section 2, the early commitment in the valuation may lead to poor plans. The constraints implied by the pre- and post-conditions of actions added to the plan govern the possible values of control parameters, as illustrated in Figure 3. The details of this process are described in the rest of this paper.

4.1 LP temporal and numeric scheduling

The POPF planner inherits the use of linear programming from the COLIN planner. It uses the LP to check the temporal and the numeric consistency of a P-state. The P-state variables that capture evolution of the numeric state variables along the trajectory of a plan are defined as follows:
A sequence of pairs of vectors of variables, V_i and V_i', is constructed, one for each step i in the plan. Each $v_i \in V_i$ records the value of the state variable v *just before* the step i. Similarly, each $v_i' \in V_i'$ records the value of a state variable v *immediately after* the step i. For instance, a state variable v can have a discrete instantaneous numeric change at a step i. In this case, the constraint $v_i' = v_i + c$ constraint, where v_i is increased by the numeric value of c at the step i, is added to the LP to record this change. Table 1 shows the constraints and variables created to record numeric changes that are connected to the `?cash` control parameter. bal_i and inp_i represents the $(balance\ ?m)$ and $(inPocket\ ?p)$ state variables at step i, respectively.

5 PLANNING WITH CONTROL PARAMETERS

The difference between the POPF and POPCORN planners is that POPCORN can reason with control parameters in addition to duration. Control parameters are, as with duration, variables that appear only as part of the structure of an action in the plan, not as part of the state. This means that variables can be added to the constraint collection in a P-state as new actions are applied, to represent not

Table 1. Variables and constraints acting upon `?cash` parameter, that are collected from the initial state to reach the goal state. [*lb, ub*] represents the upper and lower bound limits of the variables at a state.

Plan Action	LP Variable	[lb, ub]	Constraints
Withdraw (start)	$cash$	$[5, \infty]$	$cash \geq 5$
		[0,50]	$cash \leq bal_0$
	bal_0	[50,50]	bal_0
	bal_0'	[0,45]	$= bal_0 - cash$
Withdraw (end)	$cash$	[5,50]	$cash$
	inp_1	[2,2]	inp_1
	inp_1'	[7,52]	$inp_1 + cash$
BuySnacks (start)	inp_2		$= inp_1'$
	inp_2'	[7,52]	≥ 5
BuySnacks (end)	inp_3	[7,52]	$= inp_2'$
	inp_3'	[2,47]	$= inp_3 - 5$

only the values of the state variables, but also newly introduced control parameters. This is achieved by extending the existing machinery in POPF. We consider the details of each component in their related subsections. We extend the existing problem definition to capture the control parameters defined in actions. We define the additional constraints and variables added to the LP based on the numeric preconditions and effects of actions added during state progression. We provide details of the modifications made to the existing heuristic state evaluation of POPF and analyse the effects of the control parameters on the search space. We use the cash point example to illustrate elements discussed in the related subsections.

5.1 State definition

The P-state representation used in POPF is further extended for temporal-numeric planning with control parameters. We call this extended representation a C-state (Control state).

Definition 2 *C-state, S, is a tuple $\langle F, V, B, Q, P, C, T, D\rangle$, where: F, V, B, Q, P and T are as defined in a P-state. D maps all control parameters associated with actions in P to pairs recording the upper and lower bounds on their values in S.*

When a new action, A, is started, the state progression for POPCORN adds to D newly created variables corresponding to the control parameters in A, with upper and lower bounds derived from the preconditions of A (or +infinity or -infinity where no bounds appear in A). Constraints are added to C that represent the pre- and post-condition requirements on these control parameters.

The ordering of the duration and control fields in action descriptions is carefully chosen to allow control parameters to appear in duration constraints. This makes it possible to tie together these parameters in constraints. For example, we can capture a more general version of the Kongming AUV descend action, shown in Figure 4, allowing a variable duration descent that is also able to descend at different gradients by separately selecting the x- and y-velocities. The assumption modelled in the action is that the execution of this action involves moving at a constant velocity along a straight vector with the chosen x- and y- displacements. To avoid the non-linear interaction between rates and durations, we model this by choosing the distances travelled in the x- and y-directions, subject to the constraints that these distances must not imply violation of the velocity limits (see Figure 5). We can also combine multiple control parameters in a single constraint, as illustrated in the `descend` action in Figure 5, where the combined power consumption of the motors cannot exceed the available power output. This makes the control space for this problem multi-dimensional.

Scotty [8] offers a syntax for representing control parameters that is very similar to our own. In Scotty, control parameters are intended to represent the margins of control of processes during execution, so these parameters are always intended to be interpreted as rates, as can be seen in their use in the continuous effects in Figure 6. However, this model combines duration and control parameters in a quadratic expression. Scotty overcomes this difficulty by focussing on upper and lower bounds defining the range of possible values for the control parameters. This use of control parameters cannot be exploited in problems such as the cashpoint domain, where the parameter is not a rate.

```
(:action descend
 :duration (d)
 :precondition (and (y <= 200))
 :effect ()
 :dynamics (4 <= vx <= 8, 3 <= vy <= 6))
```

Figure 4. The Kongming action model for AUV descent. Note that this is not PDDL, but a variant in which a new `:duration` field is added to classical actions to associate a fixed duration to the action, and `:dynamics` which defines the control parameters and their bounds.

```
(:durative-action descend
 :parameters (?a - auv)
 :control (?dx ?dy - number)
 :duration (and
  (<= ?duration (/ ?dx (minVx ?a)))
  (<= ?duration (/ ?dy (minVy ?a)))
  (>= ?duration (/ ?dx (maxVx ?a)))
  (>= ?duration (/ ?dy (maxVy ?a))))
 :condition (and
  (over all (<= (+ (* (powerX ?a) ?dx)
  (* (powerY ?a) ?dy))(* ?duration (power ?a)))))
 :effect (and (at end (increase (posX ?a) ?dx))
  (at end (increase (posY ?a) ?dy))))
```

Figure 5. Descend action with control parameters. This model is in PDDL2.1 extended with our proposed syntax for control parameters. Note that the linear power constraint restricts the total power use across the two motors according to the consumption rates of the motors.

```
(:durative-action navigate
 :control-variables ((velX) (velY))
 :duration (and (<= ?duration 5000))
 :condition (and
  (over all (>= (velX) -4))(over all (<= (velX) 4))
  (over all (>= (velY) -4))(over all (<= (velY) 4))
  (over all (<= (x) 700))(over all (>= (x) 0))
  (over all (<= (y) 700))(over all (>= (y) 0))
 :effect (and
  (increase (x) (* 1.0 (velX) #t))
  (increase (y) (* 1.0 (velY) #t))))
```

Figure 6. The `navigate` action for Scotty [8]. The syntax is very similar to our proposal, but control parameters are always rates of change.

State progression of C-states is similar to P-state progression: new actions can be started, or current actions can be completed. In both cases, the appropriate constraints are added to C, introducing control parameters where necessary and new state variables, and new bound are then computed for B, T and D.

5.2 Checking C-state consistency

It is worth emphasising the main characteristic of control parameters within an action instance: each control parameter is a *local* variable, whose scope is limited to the action within which it is defined. They are carried out through the plan with the help of *global* variables (e.g. *(inPocket ?p)* in the cashpoint domain). Each control parameter is introduced in constraints only at the point when an action is applied. Subsequent constraints can impact on the values of control values by constraining variables that also appear in constraints with the control parameters.

In addition to the existing temporal and numeric constraints added to the LP in POPF, our approach inserts the following constraints to the LP during C-state progression:

- Any numeric precondition that is given in the form:
  $\langle v, \{\leq, <, \geq, >\}, \mathbf{w}\cdot\mathbf{v}+f(d_{i,0}, ..., d_{i,m-1}) + c\rangle$
  $\langle f(d_{i,0}, ..., d_{i,m-1}), \{\leq, <, \geq, >\}, v\rangle$
  $\langle f(d_{i,0}, ..., d_{i,m-1}), \{\leq, <, \geq, >\}, c\rangle$
- Any numeric effect that is in the form:
  $\langle v, \{\mathbf{+=,-=,=}\}, \mathbf{w}\cdot\mathbf{v}+f(d_{i,0}, ..., d_{i,m-1}) + c\rangle$

where c is a constant (appearing in the domain model), $\mathbf{v}$ is the vector of metric fluents in the problem, $\mathbf{w}$ is a vector of constants (the coefficients of state variables in the domain model) and $f(d_{i,0}, ..., d_{i,m-1})$ is a linear function applied to a set of m control parameters $\{d_{i,0}, d_{i,1}..., d_{i,m-1}\}$ appearing in action i.

Following extension of C as part of C-state progression, C is tested for feasibility. If it is infeasible, then the C-state corresponds to an empty set of temporal states, so it is pruned and search reverts to an earlier C-state. In POPCORN, C is solved for minimum and maximum bounds T, B and D. However, an alternative possibility is to tighten bounds selectively. Furthermore, a solution to C can act as a *witness* that can be carried forward in the C-state, rather like a watched literal in modern SAT-solvers, checking it against new constraints as they are added and only resolving C when the current witness fails.

The requirement for state checking is that it is possible to check C for feasibility: this does not necessarily require C to be a linear program. In principle, we can use mixed collections of constraints, possibly separating C into disconnected sets of constrained variables, using constraint solvers appropriate to each set of constraints according to type. As a simple example, we experiment with integer control variables, making C a mixed integer program (MIP). We envisage the possibility to extend this idea to different types, using specialised constraint solvers to handle them.

5.3 Temporal and numeric state-space search

The duration of an action might not be fixed. It might be determined by either the values of metric fluents, such as `(<= ?duration (v ?p))`, or constrained within a range of values [3], for example, `(and (>= ?duration 10) (< ?duration 50))`. Likewise, the value of a control parameter defined in an action is usually not fixed, but is constrained within an interval and, possibly, constrained by multiple constraints. The values of state variables subject to continuous or duration dependent effects are similarly constrained within ranges, $[v^{min}, v^{max}]$. The planner is free to choose the value of a variable within its bounds. The choices might not be

independent, so each choice must be propagated through C to determine how bounds of other variables are affected.

The existence of control parameters generates a complex branching choice in the search space, just as is the case for variable durations, but possibly multi-dimensional. Figure 3 illustrates this effect for our cash point example. When the planner branches over this set of infinitely many states, it avoids exploring each state by leaving the choice to the *LP constraint space*.

5.4 Modifications to the temporal RPG heuristic

The Metric Relaxed Planning Graph (RPG) [11] heuristic has been widely used in numeric planning over the last decade. POPF planner uses an extended variant, based on a Temporal RPG (TRPG), to guide the planner in the search space towards the goal. The difference between the two heuristics is that the TRPG associates timestamps with each action and fact layer, using rules on relaxed temporal progression to increase the time as the reachability analysis is developed [3].

Table 2. An LP relaxation over a control parameter that does not have a constant upper bound value.

Maximise: $?cash$
Subject to:
$bal_0 = 50$
$?cash \geq 3$
$?cash - bal_0 \geq -inf$
$?cash - bal_0 \leq 0$
$?cash + bal'_0 - bal_0 = 0$
$inp_0 = 2$
$inp'_0 - inp_0 - ?cash = 0$

Our modification to the TRPG heuristic of POPF is to make an optimistic assumption: if an action a has a control parameter effect on a variable v, then the control parameter is relaxed to whichever bound in D for the corresponding control variable gives the largest effect (increasing the upper bound or decreasing the lower bound on the reachable values for v). In case the bounds on the control variable depend on a the value of state variables (for example, `(<= ?cash (balance ?m))`), then the heuristic constructs an LP, using only the time-independent numeric constraints of the action, to precompute the bounds for the heuristic before extracting a relaxed plan.

Table 2 shows the LP constructed to optimistically approximate the upper bound of the `?cash` variable.

6 EVALUATION

The cashpoint domain demonstrates that there are examples of actions in which it is natural to model an action with an infinite domain parameter, other than duration. Furthermore, the `descend` action shown in Figure 5 illustrates that there are also natural reasons to transcend the use of a single control parameter, making it impossible to express such models by some compilation of control parameters into flexible durations. This demonstrates that the extended expressive power of the control parameters we propose offers a way to capture important and intuitively significant behaviours that cannot be modelled in existing PDDL formulations.

Although it is clear that the extended expressive power of control parameters has a valuable role to play in modelling realistic domains, there remains the question of whether it is possible to plan with this extended expressiveness. In this section, we consider several different domains in order to explore the scalability of the performance of POPF when confronting a variety of challenges in the use of control parameters.

Our planner, POPCORN, is a temporal planner that handles discrete numeric change in state variables that is controlled with continuous-valued variables selected by the planner. There are no other PDDL2.1 planners available with similar expressive power to compare on problems using control parameters, including Scotty. Scotty supports control parameters for setting *rates of change* for continuous change, but not control parameters used for discrete change, as in the cashpoint example. Therefore, we compare the performance of POPCORN with POPF (the code base on which POPCORN is built). This comparison is carried out by discretising the control parameter choices available to POPF. The purpose of the comparison is to explore the scaling behaviour of POPCORN, since POPF is solving a related and similar problem to the control parameters choice problem, but the discretisation imposes both additional constraints (the possible values of the control parameters are limited) and a simplification (the branching factor is made finite). A comparison with UPMurphi [16] (or the most recent version, DiNo [17]) is also possible, using discretised control parameters (which could be managed automatically in UPMurphi within its existing machinery). However, its performance does not scale well in domains with long combinatorial chains, such as the Procurement and Terraria domains.

In addition to these experiments, we make an approximate performance comparison between POPCORN and Scotty on continuous control parameter change problems (using problems from [8]). Since POPCORN does not aim to handle continuous control parameter change, the duration of actions in these problems are fixed[2].

6.1 The cash point domain

This domain is based on the motivating example presented in Section 2. The cashpoint domain has a relatively flat search space, with short plans.

The purpose of using this domain is to confirm that POPCORN effectively generates plans with control parameters. As the amount of cash required in the goal is increased relative to cash point limits, the plan has to extend to include additional withdrawal actions. It is important the planner realises that the plan cannot achieve the goals with a single withdrawal (or, more generally, insufficient withdrawals) early enough to avoid wasted search effort.

We extend the basic actions of the original domain to include multiple currencies, allowing new currencies to be obtained using an exchange bureau. This complicates problems by constructing chains of constraints connecting the values of control parameters across multiple actions.

In order to make a comparison with POPF, we add a set of `WithdrawCash` actions with fixed withdrawal values, namely `?cash` $\in \{1, 5, 10, 20\}$. This assumption allows us to make a fair comparison in these domains with POPF, which does not aim to handle control parameters. Regardless of the values we fix for the withdrawal values, the POPF will usually generate longer plans. The POPCORN does not require any fixed value, so it is able to solve the problem for any value of `(inpocket ?p)` within the bounds defined.

In all domains (including this one), we use a set of 12 problem instances. The required goal amounts and the types of currencies are increased, and in some problem instances, the planner is forced to ex-

[2] All domain and problem files used for our experiments can be found here: https://github.com/Emresav/ECAI16Domains

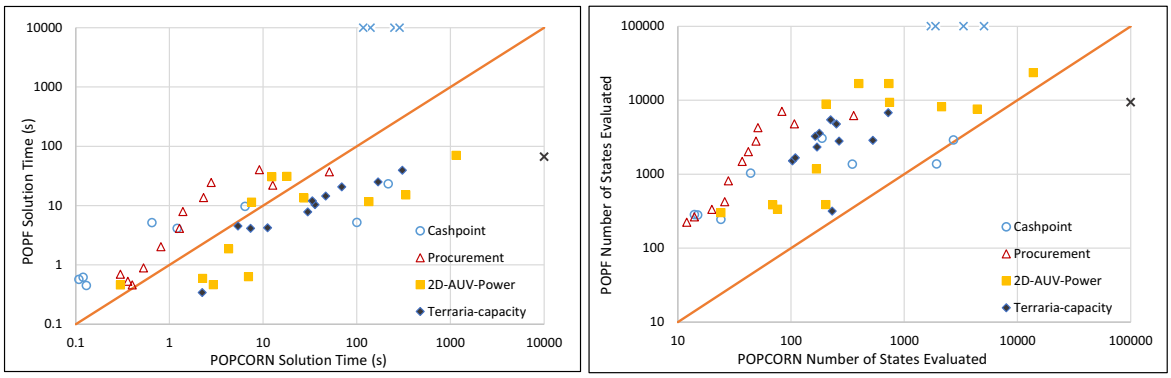


Figure 7. Comparison of time taken by POPF and POPCORN to solve each problem instance, and the numbers of states evaluated, in four domains. The crossed points indicate that only one of the planners found a solution.

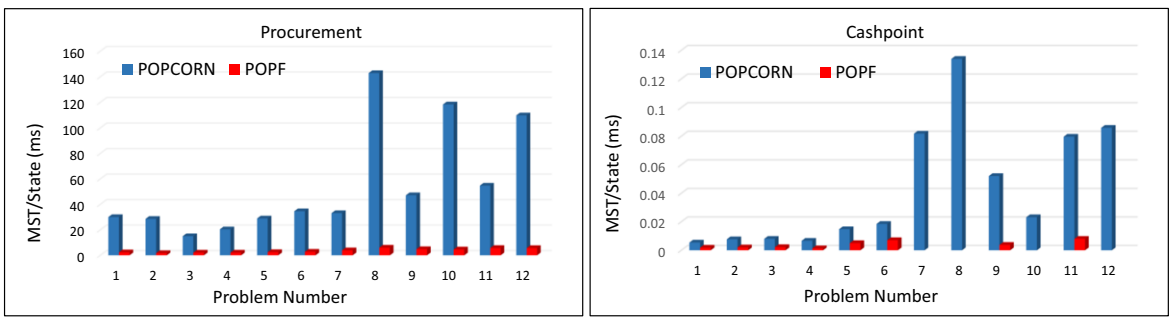


Figure 8. Mean Scheduling Time (MST) per state in procurement and cashpoint instances

change currencies at the exchange bureau. In addition, several items are added that need to be obtained to reach the goal.

6.2 The procurement and terraria domains

The procurement domain is about finding a plan to manufacture final products, which require different amount and types of raw materials to be procured from a store and intermediate products to be produced at a workshop. The purpose in defining this domain is to explore the effects of scaling numbers of control parameters in individual actions. Materials can be purchased in a single action, selecting the numbers of each of the materials available at the supplier to be included in the single purchase. This problem also contains a deeper search space, producing longer plans, allowing us to explore the cost of solving increasing numbers of more complex constraint sets.

The complexity of each instance is increased by extending the depth of the bill of materials tree. Figure 9 shows the most challenging problem instance in which two of product A, a product B, and a product L are to be delivered to customers Customer1, Customer2, and Customer3, respectively. The leaf nodes represent raw materials, which can be obtained from the store. The other nodes represent the intermediate and final products. To produce a product A, we need to have already sub-assembled two product B and a product C. These intermediate products require products I, J, K, D, and E. In this domain, the batch sizes of items procured and produced are taken as control parameters, whereas these parameters are discretised with the values of 1, 5, 10, 20 for the test runs in POPF.

The Terraria domain is similar to the procurement domain, but it additionally includes a capacity constraint that limits the total amount of raw material procured. As an extension of the procurement problem set, the complexity of each Terraria problem instance is extended by decreasing the capacity limit of raw materials. The purpose of this test is to introduce a constraint that links multiple control parameters within a single action.

6.3 The 2D-AUV-power domain

This domain is based on the Kongming AUV domain presented in Section 5. The purpose in introducing this domain is to show that duration and other control parameters in an action can be successfully constrained within a single constraint.

In this domain, the AUV travels between different waypoints to collect samples in a two dimensional underwater environment (depth and horizontal distance). Figure 5 shows the `descend` action with our proposed syntax for control parameters. In this experiment, we reinterpret this action as a single `glide` and allow the velocity range to include positive and negative changes in the y-direction (depth), so that the action can be used to ascend or descend. The duration of `glide` is constrained by the maximum and minimum displacement rates of the vehicle in x- and y-directions. The linear power constraint of `(<= (+ (*?dx 3) (*?dy 4)) 90)` limits the total power use of the vehicle across the two motors in gliding. The complexity of this experiment increases by increasing the sample destinations and decreasing the power capacity (per glide action) of the vehicle. The

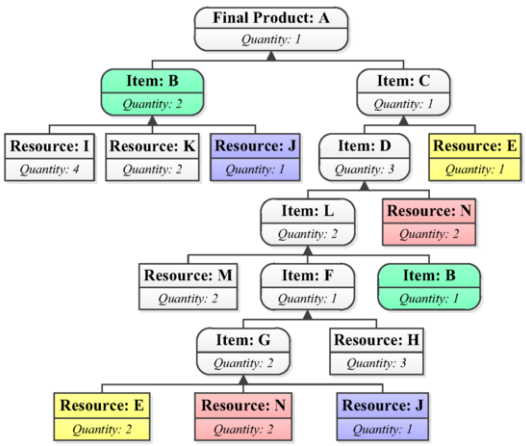


Figure 9. Bill of material tree of product A in procurement domain model. Coloured nodes indicate that those items are already used in the tree

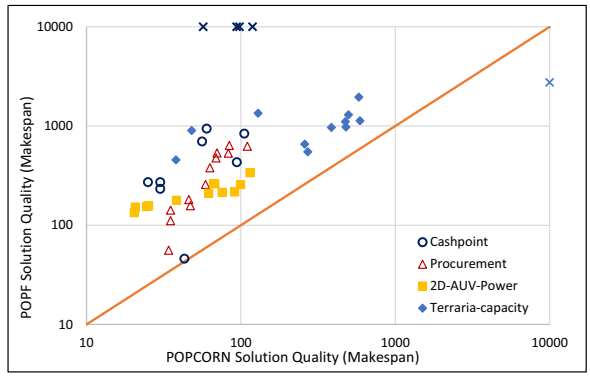


Figure 10. Comparison of plan quality (measured in this case as plan makespan) in four temporal numeric domains. POPCORN is compared with POPF on each problem instance.

displacement rate of the glide action is fixed with 1, 5, and 10 units (`?dx`, `?dy` $\in \{1, 5, 10, 20\}$). The AUV can take sample whenever the x and y positions are in the range the sample destination.

6.4 Experimental results

Figure 7 shows the time taken to solve each problem instance, and the number of states evaluated in the domain models.

As discussed in Section 5.3, the LP avoids early commitment to values of control parameters, but maintains the constraint space in which they must lie. However, a deeper search space might lead to an excessive use of the LP during planning. As mentioned earlier in this section, the procurement domain model creates a deeper search space than the cashpoint domain. We observe that POPCORN takes longer than POPF to generate plans in almost all of the procurement problem instances. The reason behind this behaviour is that POPF realises the use of its Simple Temporal Network (STN) is sufficient in most of the steps. Coles et al. show that using the STN in temporal domains is significantly faster than using the LP [5]. Figure 8 shows the mean time spent solving the LP at each state. These figures indicate that the constraint-solving time per state explored by POPCORN gradually increases, because larger-sized linear programs are generated as the problem complexity increases.

Despite the LP costs it is worth noting that POPF generates poor plans. Repetition of the same action is observed in plans generated by POPF. Even though multiple choices of the same discrete actions are defined, POPF usually chooses the ones with the most minimum increase or decrease effects, which leads it to generate poor plans; POPCORN generates better plans where repetition of the same action is not usually observed. Figure 10 compares the solution quality for each problem instance solved by POPCORN and POPF.

Scotty planner introduces various domains with continuous control parameter effects. Table 3 compares the performance of POPCORN and Scotty in these problem instances. We avoid non-linear interaction between control parameters by modelling the problems in the same fashion discussed in Section 5. 2D- and 3D-AUV domains are based on an underwater sampling mission scenario. In 2D-AUV domain, the vehicle collects sampling data in at the same depth, while the vehicle collects data in different depths in the 3D-AUV domain. In this experiment, we slightly modified these domains to challenge the performances of both planners. Seven and fourteen sampling destinations are defined in 2D AUV1 and in 2D AUV2 domains. We observed that Scotty generates longer plans in 2D AUV domains than POPCORN due to taking longer paths in both problems.

Table 3. Comparison between POPCORN and Scotty in the 5 problems in Firefighting (FF), 2D and 3D AUV navigation [8]. The numbers in the brackets are the makespan, while the numbers outside the brackets are the execution times.

Domains:	2D AUV1	2D AUV2	3D AUV	FF1	FF2
Scotty	2.6(139)	8.4(260)	0.8(12)	1.2(20)	2.5(14)
Popcorn	0.9(116)	8.8(153)	0.5(12)	0.1(20)	0.7(15)

7 CONCLUSIONS

Physical and logical properties of real-world examples require multiple numeric variables to create realistic planning models. We have shown examples of problems, in which it is most natural to model a choice of numeric parameter values to control the behaviour of actions, in a similar way, but in addition to, the duration of flexible-duration actions. Furthermore, the opportunity to combine multiple control parameters in a single action, so that the control space is multi-dimensional, is motivated by specific examples.

We have presented a planning approach capable of solving problems in domains with control parameters and we have demonstrated that it is capable of solving interesting problems with a range of characteristics, performing scalably and producing efficient plans.We compare performance with two alternatives: POPF using discretised control parameters and Scotty, which offers a different role for control parameters that does not include examples such as the cashpoint, procurement or Terraria domains.

The approach we have proposed, in which constraints governing control parameters are accumulated into a constraint program that is checked for feasibility as states are progressed, generalises to other types and to non-linear constraints, subject to the capabilities of the solver for the constraint program. We intended to explore this in future work.

ACKNOWLEDGEMENTS

This work is funded by European Commision Seventh Framework Programme for Research and Technological Development (FP7) as a part of SQUIRREL project under grant agreement No 610532.

REFERENCES

[1] J. Benton, Amanda Coles, and Andrew Coles, 'Temporal Planning with Preferences and Time-Dependent Continuous Costs', in *Proceedings of The 21st International Conference on Automated Planning and Scheduling, ICAPS*, (2012).

[2] Avrim Blum and Merrick L. Furst, 'Fast Planning Through Planning Graph Analysis', in *Proceedings of The 14th International Joint Conference on Artificial Intelligence, IJCAI*, pp. 1636–1642, (1995).

[3] Amanda Coles, Andrew Coles, Maria Fox, and Derek Long, 'Temporal Planning in Domains with Linear Processes', in *Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, (July 2009).

[4] Amanda Coles, Andrew Coles, Maria Fox, and Derek Long, 'Forward-Chaining Partial-Order Planning', in *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 42–49, (2010).

[5] Amanda Coles, Andrew Coles, Maria Fox, and Derek Long, 'COLIN: Planning with Continuous Linear Numeric Change', *Journal of Artificial Intelligence Research (JAIR)*, 1–96, (2012).

[6] Andrew Coles, Maria Fox, Derek Long, and Amanda Smith, 'Planning with Problems Requiring Temporal Coordination', in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI)*, (July 2008).

[7] Robert T. Effinger, Brian C. Williams, Gerard Kelly, and Michael Sheehy, 'Dynamic Controllability of Temporally-flexible Reactive Programs', in *Proceedings of The 19th International Conference on Automated Planning and Scheduling, ICAPS*, (2009).

[8] Enrique Fernández-González, Erez Karpas, and Brian C. Williams, 'Mixed Discrete-Continuous Heuristic Generative Planning Based on Flow Tubes', in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, (2015).

[9] Enrique Fernández-González, Erez Karpas, and Brian C. Williams, 'Mixed Discrete-Continuous Heuristic Generative Planning Based on Flow Tubes', in *Proceedings of the 3rd Workshop on Planning and Robotics (PlanRob)*, pp. 106–115, (2015).

[10] Maria Fox and Derek Long, 'PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains', *Journal of AI Research (JAIR)*, **20**, 61–124, (2003).

[11] Jörg Hoffmann, 'The Metric-FF Planning System: Translating 'Ignoring Delete Lists' to Numeric State Variables', *Journal of Artificial Intelligence Research*, 291–341, (2003).

[12] Andreas G. Hofmann and Brian C. Williams, 'Exploiting Spatial and Temporal Flexibility for Plan Execution for Hybrid, Under-actuated Robots', in *Proceedings of The 21st National Conference on Artificial Intelligence (AAAI)*, pp. 948–955, (2006).

[13] Hui X. Li and Brian C. Williams, 'Generative Planning for Hybrid Systems Based on Flow Tubes', in *International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 206–213, (2008).

[14] Florian Pantke, Stefan Edelkamp, and Otthein Herzog, 'Planning with Numeric Key Performance Indicators over Dynamic Organizations of Intelligent Agents', in *German Conference on Multi Agent System Technologies*, pp. 138–155. Springer, (2014).

[15] Florian Pantke, Stefan Edelkamp, and Otthein Herzog, 'Symbolic Discrete-Time Planning with Continuous Numeric Action Parameters for Agent-controlled Processes', *Mechatronics*, **34**, 38–62, (2016).

[16] Giuseppe Della Penna, Daniele Magazzeni, Fabio Mercorio, and Benedetto Intrigila, 'UPMurphi: A Tool for Universal Planning on PDDL+ Problems', in *Nineteenth International Conference on Automated Planning and Scheduling (ICAPS)*, (2009).

[17] Wiktor Piotrowski, Maria Fox, Derek Long, Daniele Magazzeni, and Fabio Mercorio, 'Heuristic Planning for PDDL+ Domains', in *Proceedings of The Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, (July 2016).