Dissertation Abstract

Emre Ökkeş Savaş
Supervisors: Maria Fox, Derek Long
Department of Informatics,
King's College London, London, WC2R 2LS, UK
e-mail: okkes.savas@kcl.ac.uk

Abstract

This dissertation outlines the work I have done since the beginning of my research degree. My research interest is in constrained resource planning, where I am particularly interested in the applications of operations research techniques in the task planning. Planning community has started to use operations research tools in their work in recent years. I aim to introduce new techniques to the task planning, which are widely practiced in operations research. The contribution of this paper is to present a generalisation of variables in the planning domain. We consider all types of predefined variables and the duration to a new type, which we call *control parameters*. We also describe the development of our new planner POPCORN (Partial-Order Planning with Constrained Real Numerics) that can reason with control parameters. We present an example of how existing task planning benchmark domains can be extended to develop enriched plans. We also provide an example to demonstrate the robustness and applicability of our approach.

1 Introduction

Integration of the temporal and metric fluents has become a popular field of research in the task planning. Many off-the-shelf planners (Della Penna et al. 2009; Fernández-González, Karpas, and Williams 2015a; Bajada, Fox, and Long 2015; Bryce et al. 2015; Piacentini et al. 2015) have overcome challenges posed in hybrid systems with the help of some optimisation tools. Finding the timestamps and the durations of actions have been the major interest of such planners, while the remainder dynamics of the real-world problems are neglected. The duration of an action is the only variable in PDDL domains, for which the planner has the freedom to assign a value. However, there are numerous time-independent dynamics in real-world problems. For instance, the driver decides on the initial velocity of the vehicle, the refuel amount before or during the journey. These dynamics are assigned with a fixed value at the initial state in PDDL problem instances, but the planner should not be constrained with such discretised values. In this paper, we present an approach to include variables other than the duration of the action in planning domains. We consider generalising all sorts of variables into a new type, which we call *control parameters*. We consider the duration of an action as a special type of control parameter, where it plays an important role in the plan ordering.

```
(:durative-action refuel
  :parameters (?v - vehicle ?l - location)
  :control (?fuel - number)
  :duration (= ?duration 10)
  :condition (and
    (at start (>= ?fuel 0))
    (at start (<= ?fuel (fuel-max ?v)))
    (at start (at ?v ?l))
    (over all (at ?v ?l))
    (at start (has-petrol-station ?l)))
  :effect (at end
    (increase (fuel-left ?v) ?fuel)))
```

Figure 1: Updated refuel action in Transport-numeric domain

Many state-of-the-art planners only find a sequence of the time-stamped actions to reach a goal state. Additionally, the use of control parameters enables a planner to make a decision about the values of the variables predefined in the planning domain. The planner can constrain the feasible region of a control parameter during the plan construction. We can present a simple example here, based on the *transport-numeric* domain, which is used as a benchmark domain in the International Planning Competition in 2008[1]. In this domain, trucks deliver packages from their initial locations to the goal locations. The fuel level of trucks decreases as trucks move from one location to another. The `refuel` durative action fills the tanks of trucks to the full tank (a fixed value), even if the truck only needs a small amount of petrol to reach its goal location. It would be more realistic if the planner does not assign a fixed numeric value to the fuel level, but assigns a value that is sufficient to reach its goal. This value can be constrained by numeric constraints that are *dynamically developed* during planning. Figure 1 shows the updated version of the `refuel` action where the refuel amount is taken as a control parameter: `?fuel`, where it is constrained within the real numeric region of `[0, (fuel-max ?v)]` (`(fuel-max ?v)` state variable is fixed with a constant value at the initial state).

In this paper we present our new planning system, POPCORN, that can reason with control parameters with the

[1]Original domain used in the competition can be obtained from *http://ipc08.icaps-conference.org/deterministic/domains.html*

help of linear programming (LP). The implementation we have so far focussed on linear constraints, however we will extend this concept to a non-linear case in the future. POPCORN is built on POPF (Coles et al. 2010), so that the existing sophistication of a temporal planner is preserved. The main objective of this paper is to describe the major steps taken to develop our new planning system. The structure of this paper is in the following order. We consider the related work in the field in Section 2. We then provide a simple example in Section 3 to use throughout the paper. We carry on with the description of POPF planner, which constitutes the basis of our implementation, in Section 2. We then describe the required modifications to the existing problem formulation, constructing linear program (LP), the heuristic guidance, and the forward state space search. Finally, we provide the future work and the preliminary evaluation of our approach.

2 Related Work

Early work exploring the use of the control parameters in planning domain is considerably limited. Kongming planner (Li and Williams 2008) captures the interaction of the dynamic continuous variables with *flow tubes* produced at each action layer. The flow tubes contain control trajectories of the variables as the graph expands over time. It can only handle problems with linear effects. In order to capture the continuous dynamics of a problem, time is discretised while the rate of change is taken as a variable. This concept contrasts with COLIN (Coles et al. 2009) and POPF planners, where the duration is taken as a variable, while the rate of change remains constant. Kongming suffers from the limitation of the number of happenings in the plan, so it fails generating plans requiring long time horizons.

Enrique Fernández-González, Erez Karpas and Brian Williams have recently studied the planning with continuous control parameters (2015a). Their work has considered the main stages in the development of the Scotty planning system. The Scotty planner combines the flow tube representation of the 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 the forward search to overcome the happening limitation of Kongming. The planner finds *a fixed plan*, in which the planner assigns a value to the control parameters at an early stage during planning, which makes the plan invalid due to early-commitment. 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. On the other hand, the planner assigns values to the timestamps of actions without any interaction with an executive. In addition, Scotty does not support discrete numeric change (Fernández-González, Karpas, and Williams 2015b). Our planner, however, makes a decision for the values of control parameters without any human interaction, and it *delays* the valuation of these parameters, including `?duration` and the timestamps of actions, until a decision is forced by the planner.

3 A Motivating Example

We now present a simple example to introduce the 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. Intuitively we would 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. There are three ATM machines at the cash point. Each machine has a limited balance available that can be withdrawn (`(balance ?m)`), and minimum withdrawal amount is £3. Actions of this domain and the initial/goal states are given in Figure 3 and 2, respectively. In addition, Figure 2 shows the metric objective of the problem that plays an important role in the valuation of the control parameter `?cash` . The language we use is a modified version of PDDL 2.1 to encode control parameters. We list all control parameters except `?duration` in a new line, `:control()`, in a durative action.

```
 (:init (at person1 home)
  (canbuy person1 store)
  (canwithdraw person1 cashpoint)
  (available)
  (located atm1 cashpoint)
  (located atm2 cashpoint)
  (= (inpocket person1)  2)
  (= (balance atm1) 50)
  (= (balance atm2) 100)
  (= (balance atm3) 150))

 (:goal (and (>= (inpocket person1) 20)
(gotsnacks person1)  (at person1 pub))
 (:metric minimize (inpocket person1)))
```

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

In this example, the amount of cash we want to withdraw, `?cash`, depends on which actions we apply after visiting the cash point. Early assignment of the value of a control parameter may lead to generate poor plans. For instance, assigning a value to `?cash` before buying snacks would result in visiting the cash point twice. Therefore, the decision of withdrawal amount should be made at a later stage in the plan (or eventually, at the end of the plan). POPCORN builds up all the linear constraints acting upon the control parameter `?cash` until the end of the plan. Then, the planner calls the linear program to optimize all variables, i.e. `?cash`, subject to the metric objective of the problem. Since the metric objective is to minimize `inpocket` state variable in this example, the planner chooses the minimum bound of this variable as its value.

4 Background

Temporal and numeric planning has been emerged together with the help of linear programming. Numeric and temporal constraints are handled separately in the early instances of temporal planners (Coles et al. 2008). Integration of the

```
(:durative-action WithdrawCash
:parameters (?p - person ?a - location
?m - machine)
:control (?cash - number)
:duration (= ?duration 2)
:condition (and (over all (at ?p ?a))
   (at start (>= ?cash  3))
   (at start (<= ?cash (balance ?m)))
   (at start (canwithdraw ?p ?a))
   (at start (located ?m ?a) ) )
: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 (>= (inpocket ?p) 5))
  (at start (canbuy ?p ?a)))
:effect (and (at start (not (available)))
  (at end (decrease (inpocket ?p) 3))
  (at end (gotsnacks ?p))))
```

Figure 3: Main actions of the cash point domain.

temporal and numeric constraints together made it possible to handle continuous numeric change, in which the value of a state variable can depend on the timestamp and the duration of the action (Coles et al. 2012). In order to implement temporal-numeric planning with control parameters we built our planning system on the POPF planner, which can already handle this integration. In general, the state representation of a temporal-numeric planning problem can be shown by a tuple S = $\langle F, V, Q, P, C\rangle$, where:

F is the set of propositions that are true in the current state S.
V is the vector of values of the numeric state variables. Depending on the length of a state S, the state variable V varies within V^{min} and V^{max} due to linear continuous numeric effects.
Q is a list of actions, which are started but not yet finished.
P is the plan to reach the current state S.
C is a list of temporal constraints accumulated over the steps in P.

In addition to the state representation given above, POPF includes further elements[2] to support partial-order planning. The partial-order mechanism simply minimises the ordering constraints to avoid early-commitment during forward search in order to achieve 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 of assigning values to the control parameters. As discussed in Section 3, the early-commitment in the valuation may lead to generate poor plans.

[2]Full list of partial ordering extensions to state representation for propositional and numeric case can be found in (Coles et al. 2010)

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 state. The state variables that capture discrete/continuous numeric change along the trajectory of the plan are defined as follows:
Each $v_i \in V_i$ records the value of each 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, $v'_i = v_i + c$ constraint, where v_i is increased by the numeric value of c at the step i, is added to the LP in order to record this change. Table 1 shows the constraints and variables created to record numeric changes over the control parameter. bal_i and inp_i represents the $(balance\ ?m)$ and $(inpocket\ ?p)$ state variables at step i, respectively.

Plan Action	LP Variable	[lb, ub]	Constraints
Withdraw (start)	$cash$	[3, inf]	$cash \geq 3$
		[0,50]	$cash \leq bal_0$
	bal_0	[50,50]	bal_0
	bal'_0	[0,47]	$= bal_0 - cash$
Withdraw (end)	$cash$	[3,50]	$cash$
	inp_1	[2,2]	inp_1
	inp'_1	[5,52]	$inp_1 + cash$
BuySnacks (start)	inp_2		$= inp'_1$
	inp'_2	[5,52]	≥ 5
BuySnacks (end)	inp_3	[5,52]	$= inp'_2$
	inp'_3	[2,49]	$= inp_3 - 3$

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.

The use of LP makes it possible to record numeric change *between* steps. This change can be considered as a continuous change, because the time elapsed between steps is a variable. In this case, $v'_{i+1} = v'_i + \delta v_i(step_{i+1} - step_i)$ is added to the LP that records the continuous linear numeric change between consecutive steps. $step_i$ is the timestamp LP variable of step i, while δv_i represents the gradient of continuous change on v at step i. The value of a state variable v depends on time elapsed before the next action is executed. We can then say the value of state variable v varies between lower ($v^{min} \in V^{min}$) and upper ($v^{max} \in V^{max}$) bounds, which are required to check state validity when the action is still executing. In order to compute these bounds, additional variables, v_{now} and $step_{now}$, are added to the LP to check the state validity. Figure 4 shows the relationship discussed between temporal, numeric variables encoded into the LP.

5 Planning with Control Parameters

The main distinction between POPF and POPCORN planners is that POPCORN can reason with variables other than the duration in planning. We consider all variables available in the planning domain as control parameters. This is

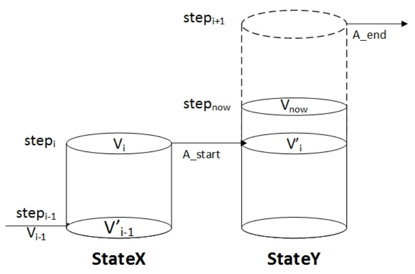

Figure 4: Schematic representation of the relationship between numeric state variable, V, and timestamp variable that are encoded in LP. V_{now} is used to compute upper-lower bounds of V during continuous linear numeric change.

achieved by extending the existing machineries of the POPF. We consider the details of each component in their related subsections. In summary, 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. We provide the modifications made to the existing heuristic approach of POPF and analyse the effects of the control parameters to the search space.We use the cash point example to enumerate elements discussed in the related subsections.

Problem Definition

Many state-of-the-art temporal planners make a decision only about *which* actions to apply, and *when* to apply these actions. Our new planner can additionally make a decision about the values of the predefined numeric variables, which are constrained with linear constraints in the linear program. Slightly different than the existing state representation of POPF, the new state representation for temporal-numeric planning with control parameters problem can be shown by a tuple S = $\langle F, V, Q, P, D, L\rangle$, where:

F is the set of propositions that are true in the current state S.

V is the vector of values of the numeric state variables. Depending on the length of a state S, the state variable V varies within V^{min} and V^{max} due to continuous or control parameter numeric effects.

Q is a list of actions, in which actions started but not yet finished.

P is the plan to reach the current state S.

D is a list of all control parameters available, including the `?duration` variable in durative actions, in the planning domain, where each $d \in D$ is a tuple $\langle op, i, num\rangle$:

- op is the identifier of instantiated action,
- i is the index of the step in the plan,
- num is the unique identifier of each $d \in D$,

where the corresponding control parameter(s) are added. The unique identifier, num, of duration variable is identical in every durative action, since there is only one duration variable defined in an action.

L is a list of constraints that encapsulates discrete numeric change with the control parameters over the steps in P, where each $l \in L$ of the form $minControl(d) \leq d \leq maxControl(d)$. The value of d lies within a range of values constrained by $minControl(d)$ and $maxControl(d)$. These bounds on d are determined from the numeric preconditions of the action on d. If there is not any numeric precondition in d defined, the range of d is set to $[0, inf]$. We restrict our definition of d to be a positive real number in order to avoid modeling errors due to sign convention in the domain.

Checking Plan Consistency with LP

In this section we consider additional variables and constraints added to the LP to support control parameters. Before we begin the formulation, it is worthwhile mentioning the main characteristic of control parameters within an action instance. The control parameter is a *local* variable, whose scope is limited to the action it is defined. It can be carried out through the plan with the numeric state variables. The following equation gives the relationship between these state variables v, and control parameters.

In general form, where step k is the current state:

$$v_{i+1} = v_{val} + \sum_{n=0}^{num} \delta w_{i,n} d_{i,n} \tag{1}$$

where,

- $d_{i,n}$ is the n^{th} control parameter defined in the action op applied to the plan at $step_i$.
- v_{val} is the variable that contains the most recent numeric value of v prior to v is affected by the control parameter d. If there is no discrete numeric effect on v at $step_i$, the value of v_{val} is equal to v_i
- $\delta w_{i,n}$ is the total gradient of the n^{th} control parameter acting upon v.
- v_{i+1} is the value of the numeric state variable immediately after the discrete control parameter effect acting on v

Temporal and numeric constraints are added to the LP to confirm that the plan to reach a state S can be scheduled. In our new planner POPCORN, the constraints with control parameters are used to check whether there is a $feasible$ range of values of the control parameter that can satisfy the plan to reach state S. In order to capture these constraints, the following constraints below are added to the LP.

- Any numeric precondition that is given in the form:
  $\langle v,\, sgn,\, \mathbf{w}{\cdot}\mathbf{v}{+}k.(d_{i,n}) + c\rangle$, s.t. $sgn \in \{\leq, <, =, \geq, >\}$
  $\langle d_{i,n},\, sgn,\, v\rangle$, s.t. $sgn \in \{\leq, <, \geq, >\}$
  $\langle d_{i,n},\, sgn,\, c\rangle$, s.t. $sgn \in \{\leq, <, \geq, >\}$

- Any numeric effect that is in the form: $\langle v,\, sgn,\, \mathbf{w}{\cdot}\mathbf{v}{+}k.(d) + c\rangle$, s.t. $sgn \in \{$+=,-=,=$\}$; $c, k \in \mathbb{R}$

are added as constraints over V to the LP. If the constructed LP with these constraints is not solvable, then the state S is pruned from the search space, and the planner backtracks to look for a state, in which the LP has a feasible solution. Our approach isolates the nonlinear interaction between a control parameter and a temporal variable. The LP is inadequate to check state consistency for nonlinear states. We are working on addressing this with the help of an appropriate nonlinear solver.

Temporal-Numeric State-Space Search

Duration of an action in a durative action may not be fixed, and it can be determined by either the values of metric fluents: i.e. `(<= ?duration (v ?p))`, or it is constrained within a range of values (Coles et al. 2009), i.e. `(and (>= ?duration 10) (< ?duration 50))`. Likewise, the value of control parameter defined in an action is not fixed, but it can be constrained within some interval. Similar logic applies to the numeric state variables that have ever had a discrete control parameter dependent change. The value of a state variable is constrained within a range of values, $[v^{min}, v^{max}]$, that the planner has the freedom to choose. Figure 5 shows the differences between discrete, continuous, and discrete control parameter changes acting upon state variable v. Suppose that state variable v is affected by discrete numeric changes at $step_1$ and $step_4$, linear continuous change between $step_2$ and $step_3$, and discrete change with control parameter at $step_3$. v can take at any numeric value within dashed area.

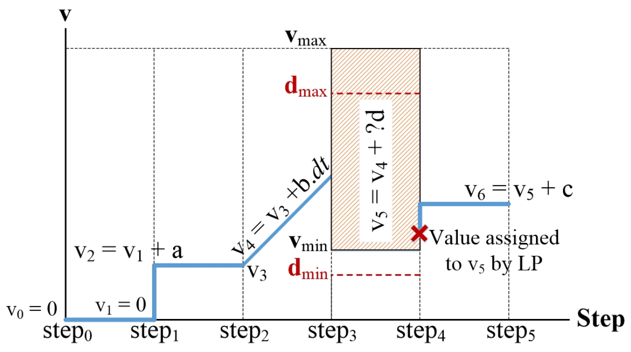

Figure 5: Schematic representation of numeric state variable, v, affected by discrete, continuous, discrete control parameter numeric change over steps 0 to 5. Shaded area shows the feasible region of values of v that can be assigned by calling linear program. $?d$ is a control parameter with a value of d lies within range of $[d_{min}, d_{max}]$. a, b, $c \in \mathbb{R}$.

The existence of control parameters generate a complex branching choice in the search space, while there remains a finite set of action choices available in the search space. Then we can say the width of the search tree remains the same, while the depth of the search tree dramatically increases after a control parameter effect. Figure 6 illustrates this effect in the search space for our cash point example. (*inpocket* += *?cash*) effect produces infinitely many states, because the value of *?cash* is not yet assigned. However, if our implementation is forced to branch over this infinite space, it avoids this by leaving the choice to the LP constraint space. avoids this by leaving the decision to the LP constraint space.

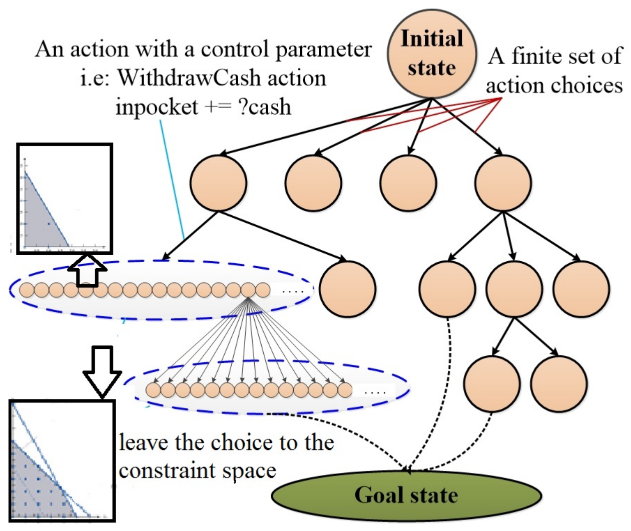
An action with a control parameter i.e: WithdrawCash action inpocket += ?cash
Initial state
A finite set of action choices
leave the choice to the constraint space
Goal state

Figure 6: 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.

Modifications to The Temporal RPG Heuristic

The Metric Relaxed Planning Graph (Hoffmann 2003) heuristic has been widely used in the numeric planning over the last decade. POPF planner uses a heuristic, Temporal RPG, to guide the planner in the search space towards the goal. The Temporal RPG (TRPG) heuristic is a modified version of the Metric RPG. The main difference between two heuristics is the timestamps associated to each action and fact layer in TRPG.

Our modification to the existing temporal RPG 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 $minControl(d)$ or $maxControl(d)$ gives the largest(smallest) effect. In case the $minControl(d)$ and/or $maxControl(d)$ depend on a the value of a state variable (i.e: `(<= ?cash (balance ?m))`), then the heuristic calls the LP, which only contains the time-independent numeric constraints of the action, to precompute the bounds for the heuristic before extracting a relaxed plan. For instance, in the reachability analysis, the following LP constructed to find the upper bound of *?cash*:

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

6 Evaluation

In this section we present the preliminary results of our implementation. Since the existing planners that reason about control parameters are not available online, we compare the capability of our implementation with our base planner. However, POPF can not run at all on the cash point problem if we do not provide a fixed withdrawal value. Therefore, we fix the withdrawal value, `?cash`, at £10 for running the experiments with POPF. Regardless of the value we fix the withdrawal value to, the POPF will always 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.

We compared the performance of POPF and POPCORN in problems, where the goal `(>= (inpocket person1) 500)` incrementally increases to `(>= (inpocket person1) 950)` in every problem instance. We observe lengthy plans produced by POPF due to repetitive `WithdrawCash` actions. Table 2 shows the results of this evaluation. This preliminary evaluation shows that our approach dramatically decreases the number of states evaluated and the plan length produced by our based planner. POPF is not able to produce plans for `(>= (inpocket person1) 900)` problem instances, because it runs out of memory.

	# States Evaluated		Plan Length	
inpocket≥	POPF	POPCORN	POPF	POPCORN
500	508	457	57	10
550	2089	142	62	10
600	2248	484	67	12
650	3892	484	72	12
700	7461	484	77	12
750	14143	1430	82	13
800	8750	1430	87	13
850	32341	1430	92	13
900	–	4366	–	14
950	–	4366	–	14

Table 2: Number of states evaluated and plan length evaluation of POPF and POPCORN planners, where `inpocket` goal is discretised for POPF.

7 Future Work

I consider finalising the implementation of POPCORN presented in this paper, and extend its capability by implementing a nonlinear solver to solve problems requiring nonlinear numeric change. In order to achieve this, I will initially identify which nonlinear solver is sufficient to use in planning. Then, I will explore the required modifications to implement this solver within our existing planning system. Another future work I want to work on is about managing the preferences of objectives defined in the domain. I plan to use Goal-Programming approach to minimise the penalty costs for the multi-objective planning domains. The minimised penalty cost can be used to get guidence in the search space as a tie-breaking factor (where the timespan of plan options are equal). Finally, I consider extending the generalisation of the control parameters with non-numeric object variables. As mentioned in Section 1, the planner is initially constrained with discretised assignments, for which the planner actually should have freedom to choose. This approach can be implemented for objects defined in planning problem.

8 Conclusion

Physical and logical properties of the real-world examples require multiple numeric variables to create realistic planning models. In this paper we provide the preliminary work of our implementation to handle control parameters in the planning domain. We generalise the use of all parameters to a new type to fully integrate temporal and numeric planning. At this stage we identified the necessary modifications to the existing mechanism of our base planning system.

References

Bajada, J.; Fox, M.; and Long, D. 2015. Temporal planning with semantic attachment of non-linear monotonic continuous behaviours. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 1523–1529. AAAI Press.

Bryce, D.; Gao, S.; Musliner, D.; and Goldman, R. 2015. Smt-based nonlinear pddl+ planning. In *Proceedings of the Twenty Nineth Conference on Artificial Intelligence (AAAI-15). AAAI Press.*

Coles, A. I.; Fox, M.; Long, D.; and Smith, A. J. 2008. Planning with problems requiring temporal coordination. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 08).*

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2009. Temporal planning in domains with linear processes. In *Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 42–49.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2012. Colin: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research* 1–96.

Della Penna, G.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. Upmurphi: a tool for universal planning on pddl+ problems. In *Nineteenth International Conference on Automated Planning and Scheduling*.

Fernández-González, E.; Karpas, E.; and Williams, B. C. 2015a. Mixed discrete-continuous heuristic generative planning based on flow tubes. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Fernández-González, E.; Karpas, E.; and Williams, B. C. 2015b. Mixed discrete-continuous heuristic generative planning based on flow tubes. In *Proceedings of the 3rd Workshop on Planning and Robotics (PlanRob-15)*, 106–115.

Hoffmann, J. 2003. The metric-ff planning system: Translating“ignoring delete lists”to numeric state variables. *Journal of Artificial Intelligence Research* 291–341.

Li, H. X., and Williams, B. C. 2008. Generative planning for hybrid systems based on flow tubes. In *Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 206–213.

Piacentini, C.; Alimisis, V.; Fox, M.; and Long, D. 2015. An extension of metric temporal planning with application to ac voltage control. *Artificial Intelligence* 229:210–245.