Extending a MILP Compilation for Numeric Planning Problems to Include Control Parameters

Emre Savaş[‡] and **Chiara Piacentini**[†]
[‡]Department of Informatics, King's College London, London, UK, WC2B 4BG
[†]Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada, ON M5S 3G8

Although PDDL is an expressive modelling language for planning problems, 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. In real-world, there are parameters whose values have infinite (or highly large-sized) domains, they are called *control parameters*. Thus, modelling and reasoning with these parameters is indeed a requirement. Recent work investigated planning with these parameters (Ivankovic et al. 2014; Fernández-González, Karpas, and Williams 2015; Savaş et al. 2016).

While for classical planning, heuristic search is the *state-of-the-art* approach, the introduction of control parameters might introduce a high number of branching points. In addition, the inclusion of control parameters in the heuristic evaluation is still an open issue (Savaş et al. 2016). Alternatively to heuristic search, several compilations to Boolean Satisfiability (SAT) (Rintanen 2012), Constraint Programming (CP) (Vidal and Geffner 2006) and Mixed Integer Linear Programming (MILP) have been proposed (van den Briel et al. 2007). In this work, we are interested in MILP compilations, as control parameters can be easily modelled as additional variables of the model, whose values are only constrained by the actions preconditions, but not by the actions effects. We present here an extension of the MILP compilation of numeric planning problem with instantaneous actions (Piacentini et al. 2018) to include control parameters.

1 Problem Definition

Numeric planning introduces numeric state variables, extending planning tasks beyond the propositional formalism. We further extend the definition of numeric planning tasks by adding control parameters as follows.

Definition 1 *A* numeric planning task with control parameters *is a tuple* $\langle V_p, V_n, I, A, G\rangle$*, where:*

- V_p *is a finite set of propositional variables,*
- V_n *is a finite set of numeric variables,*
- I *is the initial state,*
- A *is a set of actions. Each action,* $a \in A$*, is a tuple:*
  $$a = \langle \textit{cparam(a), pre(a), eff(a), cost(a)}\rangle$$
  - *cparam(a) is a declaration of a finite set of numeric control parameters of action* a*, where each* $d^a \in$ *cparam(a) has a domain* ($dom(d^a)$) *of* $\mathbb{Q}$ *or* $\mathbb{Z}$*.*
  - *pre(a) is a set of preconditions. The preconditions can be propositional,* $pre_p(a)$*, or numeric,* $pre_n(a)$*.* $\forall p \in pre_p(a)$ *corresponds to* $v_p \in V_p$ *being true and* $\forall np \in pre_n(a)$ *is of the form:* $\xi \trianglerighteq 0$*, where* $\trianglerighteq \in \{\geq, >, =\}$ *and* ξ *is a linear expression over* V_n *and cparam(a) with* $w^c_v, w^c_d, w^c_0 \in \mathbb{Q}$*:*
  $$\xi = \sum_{\substack{v \in V_n \\ \cup\ \textit{cparam(a)}}} w^c_v v + w^c_0$$
  - *eff(a) is a set of effects of the action* a*, where it is defined as eff(a)* $= \langle add(a), del(a), num(a)\rangle$*, with* $add(a)$ *and* $del(a)$ *are sets of added and deleted propositions, respectively.* $num(a)$ *is a set of numeric effects that are assignments* $v := \xi$*, where* ξ *is a linear expression over* V_n *and cparam(a) with* $k^{v,a}_w, k^{v,a}_d, k^{v,a} \in \mathbb{Q}$*:*
  $$\xi = \sum_{\substack{w \in V_n \\ \cup\ \textit{cparam(a)}}} k^{v,a}_w w + k^{v,a}$$
  - *cost(a) is the cost of applying action* a*.*
- G *is the goal described by a set of propositional,* G_p*, and a set of numeric conditions,* G_n*, over numeric variables,*

A state is a mapping of each variable to its domain, where $s(v)$ is the value of v in s. An action $a \in A$ is applicable in s iff $s(v_p) = true$, $\forall v_p \in pre_p(a)$ and $s(\xi) \trianglerighteq 0$ for all numeric conditions of a, where $s(\xi)$ is the evaluation of ξ in s. Given a state s and an applicable action a, the successor state $s' = a(s)$ is: $\forall v_p \in V_p$, $s'(v_p) = \textit{true}$ if $v_p \in add(a)$, $s'(v_p) = \textit{false}$ if $v_p \in del(a) \setminus add(a)$, and $s'(v_p) = s(v_p)$ otherwise. Each $v_n \in V_n$ takes value $s'(v_n) := s(\xi)$ if $(v_n := \xi) \in num(a)$, and $s'(v_n) = s(v_n)$ otherwise.

A plan π is a sequence of actions and specified values for control parameters of actions in π (i.e. $\pi = \{\langle a_0, cval(a_0)\rangle, \ldots, \langle a_n, cval(a_n)\rangle\}$), where n is the plan length, $cval(a) \in \mathbb{Q}^{m_a} \vee \mathbb{Z}^{m_a}$ is the vector of values of the control parameters of a, and m_a is the number of control parameters declared in action a. In addition, all conditions of actions are met and the goals are satisfied in the final state.

The scope of each control parameter is restricted to its action. We use the PDDL language proposed by Savaş et al. (2016) to model our task, as it allows the declaration of *multiple* and *typed* (i.e. they can have integer or rational number domains) control parameters in the action schema. An example PDDL action encoding using this language is shown in Figure 1. The control parameters are declared in the `:control()` field associated with their types.

```
(:action bake_a_cake
 :parameters (?c - spongecake)
 :control (?milk ?flour - number, ?cake - int)
 :precondition (and
  (>= ?milk (* 200 ?cake)) (>= ?flour (* 100 ?cake)))
 :effect (and (increase (stock ?c) ?cake)))
```

Figure 1: An example PDDL action schema

2 MILP Formulation

In this section, we present the extension of the MILP formulation for numeric planning problem to handle control parameters. Due to space limitation, we report only the constraints necessary to model the numerical part of the problem. These constraints can be added to any of the MILP models designed for classical planning problems: the state-based model, the state-change model (Vossen et al. 1999) and the SAS+ state-change model (van den Briel, Vossen, and Kambhampati 2005).

Let $\mathcal{T} = \{0, ..., T-1\}$ and $\tilde{\mathcal{T}} = \mathcal{T} \cup \{T\}$ be sets of time-steps. Consider parameters $m_{c,t} \in \mathbb{Q}$, $\forall c \in C, \forall t \in \tilde{\mathcal{T}}$, $M^{step}_{v,t}, m^{step}_{v,t}, M^{a}_{v,t}, m^{a}_{v,t} \in \mathbb{Q}$, $\forall v \in V_n \bigcup_{a \in A} \text{cparam(a)}, \forall t \in \tilde{\mathcal{T}}$, define as in previous work (Piacentini et al. 2018). Let $y_{v,t} \in \mathbb{Q}\ \forall v \in V_n \bigcup_{a \in A} \text{cparam(a)}, \forall t \in \tilde{\mathcal{T}}$ represent the value of the numeric variable v or the control parameter at time-step t. Variable $u_{a,t} \in \{0,1\}$, $\forall a \in A, \forall t \in \mathcal{T}$ indicates whether a is applied at time-step t. The constraints modelling numeric effects and conditions are given in Figure 2. Constraint (1) sets the variables to their initial state values, while constraint (2) enforces the numeric goal conditions. Constraint (3) ensures the satisfaction of numeric preconditions. Constraints (4)-(7) update the values of the numeric variables according to the action effects (simple or linear). Constraints (8)-(9) model the effects of the actions on their control parameters. They become redundant if an action is not applied. Constraint (10) is added to model the type of the control parameters. Constraint (11) enforces the mutex relation, according to the numeric mutex relation presented in previous work (Piacentini et al. 2018).

3 Conclusion

Although most planning problems are efficiently solved using state space heuristic search approaches, they become highly cumbersome with the introduction of numeric parameters with large-sized domains. Constraint programming and operations research techniques are considerably powerful for these problems. We investigated only a small subset of the product of this *cross-fertilisation* between these fields, but the recent interest shows that it is ample.

$$y_{v,0} = I(v) \qquad \forall v \in V_n \tag{1}$$

$$\sum_{v \in V_n} w^c_v y_{v,T} + w^c_0 \qquad \forall c \in G_n \tag{2}$$

$$\sum_{v \in V_n \cup \text{cparam(a)}} w^c_v y_{v,t} + w^c_0 \geq m_{c,t}(1 - u_{a,t}) \qquad \forall a \in A, \forall c \in pre_n(a), \forall t \in \mathcal{T} \tag{3}$$

$$y_{v,t+1} \leq y_{v,t} + \sum_{a \in se(v)} k^{v,a} u_{a,t} + M^{step}_{v,t+1} \sum_{a \in le(v)} u_{a,t} \qquad \forall v \in V_n, \forall t \in \mathcal{T} \tag{4}$$

$$y_{v,t+1} \geq y_{v,t} + \sum_{a \in se(v)} k^{v,a} u_{a,t} + m^{step}_{v,t+1} \sum_{a \in le(v)} u_{a,t} \qquad \forall v \in V_n, \forall t \in \mathcal{T} \tag{5}$$

$$y_{v,t+1} - \sum_{\substack{w \in V_n \\ \cup \text{cparam(a)}}} k^{v,a}_w y_{w,t} \leq k^{v,a} + M^a_{v,t+1}(1 - u_{a,t}) \qquad \forall v \in V_n, \forall a \in le(v), \forall t \in \mathcal{T} \tag{6}$$

$$y_{v,t+1} - \sum_{\substack{w \in V_n \\ \cup \text{cparam(a)}}} k^{v,a}_w y_{w,t} \geq k^{v,a} + m^a_{v,t+1}(1 - u_{a,t}) \qquad \forall v \in V_n, \forall a \in le(v), \forall t \in \mathcal{T} \tag{7}$$

$$y_{v,t} \leq M_{v,t} u_{a,t} \qquad \forall a \in A, \forall v \in \text{cparam(a)}, \forall t \in \mathcal{T} \tag{8}$$

$$y_{v,t} \geq m_{v,t} u_{a,t} \qquad \forall a \in A, \forall v \in \text{cparam(a)}, \forall t \in \mathcal{T} \tag{9}$$

$$y_{v,t} \in \mathbb{Z} \qquad \forall a \in A, \forall v \in \text{cparam(a) s.t.} dom(v) = \mathbb{Z}, \forall t \in \mathcal{T} \tag{10}$$

$$u_{a,t} + u_{a',t} \leq 1 \qquad \forall a \in A, \forall a' \in nmutex(a) \forall t \in \mathcal{T} \tag{11}$$

Figure 2: Constraints for numeric effects and conditions.

References

[Fernández-González, Karpas, and Williams 2015] Fernández-González, E.; Karpas, E.; and Williams, B. C. 2015. Mixed Discrete-Continuous Heuristic Generative Planning Based on Flow Tubes. In *IJCAI-2015*.

[Ivankovic et al. 2014] Ivankovic, F.; Haslum, P.; Thiébaux, S.; Shivashankar, V.; and Nau, D. S. 2014. Optimal planning with global numerical state constraints. In *ICAPS-2014*.

[Piacentini et al. 2018] Piacentini, C.; Castro, M. P.; Cire, A. A.; and Beck, J. C. 2018. Compiling Optimal Numeric Planning to Mixed Integer Linear Programming. In *ICAPS-2018*, 383–387.

[Rintanen 2012] Rintanen, J. 2012. Engineering efficient planners with SAT. *ECAI-2012* 242:684–689.

[Savaş et al. 2016] Savaş, E.; Fox, M.; Long, D.; and Magazzeni, D. 2016. Planning Using Actions with Control Parameters. In *ECAI-2016*, 1185–1193. IOS Press.

[van den Briel et al. 2007] van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An LP-Based Heuristic for Optimal Planning. In *CP-2007*, 651–665.

[van den Briel, Vossen, and Kambhampati 2005] van den Briel, M.; Vossen, T.; and Kambhampati, S. 2005. Reviving integer programming approaches for AI planning: A branch-and-cut framework. In *ICAPS-2005*, 310–319.

[Vidal and Geffner 2006] Vidal, V., and Geffner, H. 2006. Branching and Pruning: An Optimal Temporal POCL Planner Based on Constraint Programming. *Artificial Intelligence* 170:298–335.

[Vossen et al. 1999] Vossen, T.; Ball, M. O.; Lotem, A.; and Nau, D. S. 1999. On the Use of Integer Programming Models in AI Planning. In *IJCAI-1999*, 304–309.