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

Emre Savaș (KCL)   Chiara Piacentini (UToronto)

Limitations in PDDL

Action parameters (declared in `:parameters()`) are restricted to take their values from **finite** domains.
Planners do not have freedom to choose the values of *state variables*:

the effect of `(increase (v1) (v2))` increases (v1) precisely by the value of (v2).

Control parameters (declared in `:control()`):

- can have **infinite** domains (:: $[\mathbb{R}]$ infinite, :: $[\mathbb{Z}]$ finite),
- the planner can choose any value in their feasible domains.

```
(:action drive
 :parameters(?t1 - truck ?from ?to - waypoint)
 :control(?speed - number)
 ...)
```

$?t1 :: [truck1, truck2]$; $?to, ?from :: [London, Glasgow, Oxford, Manchester]$;
$?speed :: [\mathbb{R}]$

Control Parameter Examples

- Production planning and inventory control
  - **?batchsize, ?quantity**
- Controlling dynamics in robot manipulation
  - **?torque, ?acceleration, ?lights**
- Power management in unmanned vehicles (i.e. AUVs and UAVs)
  - **?recharge_amount**
- Refinery operations. i.e. *thermal equilibrium, mixing liquids and chemical reactions*
  - **?reactant, ?heat_transfer**
- Management of dynamics in space applications i.e. *controlled landing, objects stay in orbit*
  - **?thrust_force**

Control Parameters in Forwards Search

Previous work handles control parameters in forwards search:

- Ivankovic et al. 2014: Optimal classical planning
- Pantke et al. 2015: Production planning and jobshop scheduling application
- Fernandez-Gonzalez et al. 2015: Scotty planner
- Savaș et al. 2016: POPCORN planner

Pros:

- Provides numeric flexibility to planners

Cons:

- Complex branching factor in state-space search based frameworks
- Common heuristic problems: lack of *basic-informedness* due to *helpful action distortion*
- Dead-ends are highly probable!

Work in MILP Compilation

- Classical Planning:
  - Bylander 1997: An LP heuristic for optimal planning
  - Vossen et al. 1999: the state-change model
  - van den Briel et al. 2005: the SAS+ state change model
- Temporal-Numeric Planning:
  - Kautz and Waiser 1999: State-space Planning by Integer Optimization
  - Piacentini et al. 2018: Compiling Optimal Numeric Planning to MILP
  - LP-SAT and TM-LPSAT (temporal)

Contributions

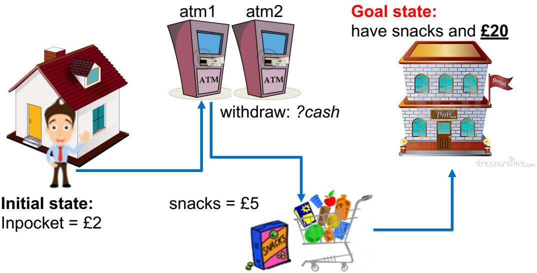
1. Investigating cost-optimal numeric planning with control parameters.
   - The problem is solved by satisficing planners, but the plan quality can be highly *unsatisficing*
2. **Compiling** the entire planning problem in MILP and **iteratively increasing** the time-horizon and **solving** each model.
   - We will extend the model to solve **cost-optimal temporal** version of this problem with flexible durations

Methodology

- Compiling the entire planning problem in MILP with time-index model, with a fixed horizon T.
- Solve by **iteratively increasing** the fixed time-horizon T
- Given a feasible solution π, check optimality at time:

$$T = \frac{cost(\pi)}{min_{a \in A} cost(a)}$$

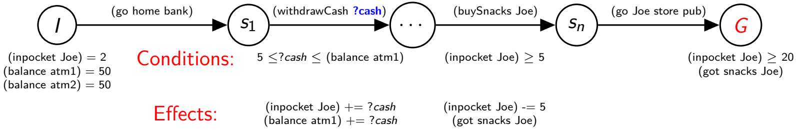
The Cashpoint Example



We do not know the value of ?cash until we decide which actions to apply next.

Early valuation can lead the planner to a dead-end.

Action	Control Parameters
(Go Joe home bank)	
(WithdrawCash Joe bank atm1)	?cash::[23, 50]
(Go Joe bank store)	
(buySnacks Joe store)	
(Go Joe store pub)	



The Cashpoint Example

Domain model and the problem instance modelled using our PDDL extension:

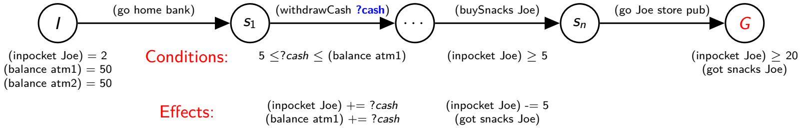
```
(:action WithdrawCash
 :parameters (?p - person ?a - location ?m - machine)
 :control (?cash - number)
 :preconditions (and (at ?p ?a) (at ?p ?a)
    (located ?m ?a) (canWithdraw ?p ?m)
    (>= ?cash 5) (<= ?cash (balance ?m)))
 :effect(and (decrease (balance ?m) ?cash)
    (increase (inpocket ?p) ?cash)))

(:action BuySnacks
 :parameters (?p - person ?a - location)
 :preconditions (and (at ?p ?a) (at ?p ?a)
    (snacksAt ?a) (>= (inPocket ?p) 5))
 :effect (and (decrease (inPocket ?p) 5)
    (gotSnacks ?p)))
```

Initial state:

```
(: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))
```

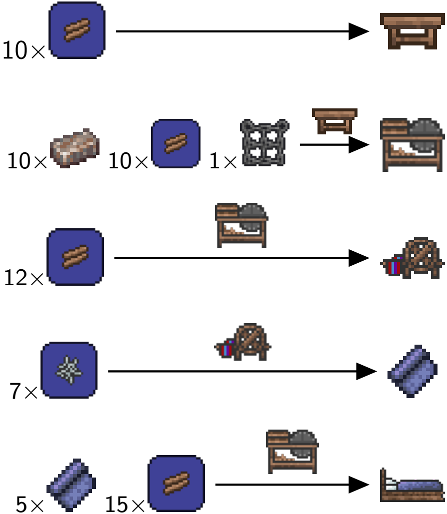
Goal state:

```
(:goal (and (>= (inPocket Joe) 20)
 (gotSnacks Joe) (at Joe pub)))
(:metric minimize (inPocket Joe))
```



The Terraria Example

Steps for making a bed:



Terraria is a 2D adventure video game that involves crafting, exploration and combat.

Items are either procured from the environment or crafted.

Crafting stations are assembled and can be re-used

The Terraria Example

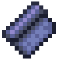
Initial state:

crafting stations:

 = N/A

items:

 = 7  = 4  = 0

 = 6  = 3  = 0

The items are available at different locations:

Forest: 

Underworld: 

Goal:

 ≥ 4

The Terraria Example

 **?wood**

 **?web**  **?iron**

 **?chain**

Desired plan:

Time	Action	Duration	Control Parameters		
0.01	(walk home forest)	2			
2.01	(cut_down_trees)	10	?wood::[5,100]		
12.02	(dig forest underworld)	20			
32.03	(find_resources)	20	?iron::[10,40]	?web::[5,75]	?chain::[0,10]
52.04	(dig underworld home)	20			
72.05	(assemble workbench)	10			
82.06	(assemble sawmill)	10			
92.07	(assemble loom)	10			
102.08	(weave silk)	5	?silk::[5,50]		
107.09	(assemble beds)	10	?bed::[1,10]		

- The value of control parameters depend on which actions we apply next (in forwards search).
- Early valuation leads to poor plan generation.

The Terraria Domain (Temporal Version)

The example is modelled using our PDDL extension:

Actions:

```
(:durative-action cut_down_trees
:parameters(?a - wood ?l - location)
:control (?w - number)
:duration (= ?duration 10)
:condition(and (at start (forest ?l))
(at start (at ?l))
(at start (>= ?w 5))
(at start (<= ?w 100)))
:effect (and (at end (increase (stock ?a) ?w))))

(:durative-action weave_silk
:parameters(?s1-silk ?cob-cobweb ?l - location)
:control (?silk - number)
:duration (= ?duration 5)
:condition (and (at start (loom_ready))
(at start (>= ?silk 5))  (at start (<= ?silk 50))
(at start (>= (stock ?cob) (* ?silk 7))))
:effect((at start (increase (stock ?s1) ?silk))
(at start(decrease (stock ?cob) (* ?silk 7)))))
```

The bounds of control parameters are defined in actions.

Initial state:

```
(:init
(= (stock wood) 7)  (= (stock iron) 4)
(= (stock chain) 0) (= (stock web) 6)
(= (stock silk) 3)  (= (stock bed) 0))
```

Goal:

```
(:goal (>= (stock bed) 4))
```

Numeric Planning with Control Parameters Model

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:

$$a = \langle cparam(a),\ pre(a),\ eff(a),\ cost(a)\rangle$$

  - *cparam(a)* is control params (of a) declaration. Each $d^a \in cparam(a) :: \mathbb{Q}$ or $\mathbb{Z}$.
  - $\text{pre}_p(a)/\text{pre}_n(a)$: propositional/numeric pre(a). Each $\text{pre}_n(a) = \xi \trianglerighteq 0$:

$$\xi = \sum_{v \in V_n \cup \text{ cparam(a)}} w_v^c v + w_0^c$$

  - $eff(a) = \langle add(a), del(a), num(a)\rangle$. $num(a) : v := \xi$:

$$\xi = \sum_{w \in V_n \cup \text{ cparam(a)}} k_w^{v,a} w + k^{v,a}, \text{ with } k_w^{v,a}, k_d^{v,a}, k^{v,a} \in \mathbb{Q}:$$

  - *cost(a)* is the cost of applying action a.

- G is the goal.

Preliminaries

- $y_{v,t}$: the value of state variable or control parameter v at time-step t,
- $u_{a,t}$: is a binary decision variable indicating whether action a applied at t,
- $I(v)$: the initial value of v,

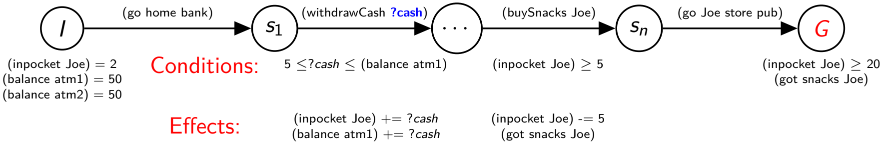
MILP Model – Initial State, Preconditions and Goals

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

$$\sum_{v \in V_n} w_v^c y_{v,T} + w_0^c \qquad \forall c \in G_n \tag{2}$$

$$\sum_{v \in V_n \cup \text{ cparam(a)}} w_v^c y_{v,t} + w_0^c \trianglerighteq m_{c,t}(1 - u_{a,t})$$

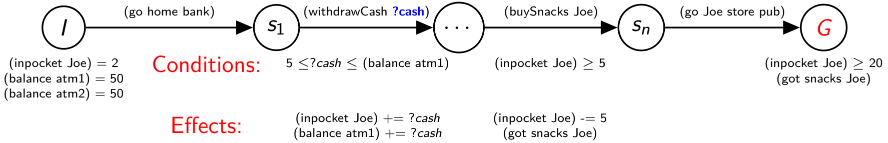
$$\forall a \in A, \forall c \in pre_n(a), \forall t \in \mathcal{T} \tag{3}$$



MILP Model (cont'd) – Effects

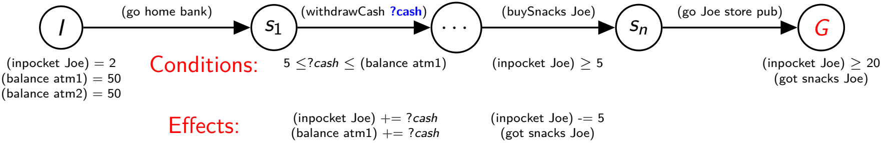
$$y_{v,t} + \sum_{a \in se(v)} k^{v,a} u_{a,t} + m^{step}_{v,t+1} \sum_{a \in le(v)} u_{a,t} \leq y_{v,t+1} \leq y_{v,t} + \sum_{a \in se(v)} k^{v,a} u_{a,t} + M^{step}_{v,t+1} \sum_{a \in le(v)} u_{a,t} \quad (4)$$

$$k^{v,a} + m^{a}_{v,t+1}(1 - u_{a,t}) \leq y_{v,t+1} - \sum_{\substack{w \in V_n \\ \cup \mathrm{cparam(a)}}} k^{v,a}_{w} y_{w,t} \leq k^{v,a} + M^{a}_{v,t+1}(1 - u_{a,t}) \quad (5)$$



MILP Model (cont'd) – Control Parameter Redundancy Constraints

$$m_{v,t} u_{a,t} \leq y_{v,t} \leq M_{v,t} u_{a,t}, \qquad \forall a \in A, \forall v \in \text{cparam(a)}, \forall t \in \mathcal{T} \tag{6}$$



Evaluations

- NA: the planner cannot reason with these domains.
- NT: not tested on this domain
- **bold**: the best, <u>underlined</u>: the worst performing system.

	Cashpoint (20)		Procurement (20)		Terraria (20)		Airplane Cargo (20)		AUV-Fuel (12)		Rover (20)	
	#solved	*T*	*#solved*	*T*	*#solved*	*T*	*#solved*	*T*	*#solved*	*T*	*#solved*	*T*
SATISFICING (temporal)												
POPCORN (TRPG)	8	5.78	6	21.37	11	65.2	12	96.53	7	138.83	<u>5</u>	0.8
POPCORN (RefinedTRPG)	**16**	4.23	**15**	99.4	**18**	100.2	**17**	8.94	**12**	517.58	**11**	2.6
cqScotty (TRPG)	NA	NA	NA	NA	NA	NA	NA	NA	5	95.06	NA	NA
COST-OPTIMAL (non-temporal)												
MILP-Compilation	<u>4</u>	829.42	<u>3</u>	344.3	NT	NT	<u>12</u>	143.35	<u>6</u>	550.68	7	1.01

Conclusions

- Planning with control parameters is more than a modelling choice, but a requirement,
- This compilation does not scale in problems with relatively large time-horizons (i.e. Cashpoint and Terraria),
- An early instance of this paradigm in cost-optimal + domain-independent way.
- Helped us to see how well Compilation-based systems do in these problems