Compiling Optimal Numeric Planning to Mixed Integer Linear Programming

Chiara Piacentini[†], **Margarita P. Castro**[†], **Andre A. Cire**[‡] and **J. Christopher Beck**[†]
[†]Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada, ON M5S 3G8
[‡]Department of Management, University of Toronto Scarborough, Toronto, Canada, ON M1C 1A4

Abstract

Compilation techniques in planning reformulate a problem into an alternative encoding for which efficient, off-the-shelf solvers are available. In this work, we present a novel mixed-integer linear programming (MILP) compilation for cost-optimal numeric planning with instantaneous actions. While recent works on the problem are restricted to actions that modify variables present in simple numeric conditions, our MILP formulation, in addition, handles linear conditions and linear action effects on numeric state variables. Such problems are particularly challenging due to the state-dependency of the action effects. Experiments show that our approach, in addition to being the state of the art for the more general problem class, is competitive with heuristic search-based planners on domains with only simple numeric conditions.

1 Introduction

While heuristic search is the most widely used approach for classical planning, there are several techniques based on compilation to other approaches such as Boolean Satisfiability (SAT) (Rintanen 2012), Constraint Satisfaction (Vidal 2011), and Integer Programming (IP) (Vossen et al. 1999; Kautz and Walser 1999; van den Briel, Vossen, and Kambhampati 2005). Compilation techniques have been proven particularly useful for extensions of classical planning, such as SAT Modulo Theory (SMT) approaches to numeric and hybrid planning (Scala et al. 2016b; Cashmore et al. 2016).

This work considers cost-optimal numeric planning with instantaneous actions. Numeric planning is an extension of classical planning where state variables can assume numeric values, action preconditions can be numeric expressions over variables, and action effects can modify the values of such variables. We propose a mixed-integer linear programming (MILP) compilation procedure for cost-optimal numeric planning. Our model builds on the IP model for classical planning proposed by Vossen et al. (1999) and generalizes the work by Kautz and Walser (1999) to consider a richer form of numeric preconditions and effects.

To our knowledge, none of the admissible heuristics for numeric planning address linear effects on numeric variables due to their state-dependency. The empirical evaluation shows that our MILP model outperforms heuristic-based planners that naively ignore state-dependent effects, while remaining competitive to the state of the art in domains with simple numeric conditions.

2 Notation and Preliminaries

We consider a fragment of numeric planning expressible with PDDL2.1, level 2 (Fox and Long 2003).

A numeric planning *task* is a tuple $\Pi = \langle V_p, V_n, A, I, G \rangle$, where V_p and V_n are the sets of *propositional* and *numeric* variables, with domains $\{true, false\}$ and $\mathbb{Q}$, respectively. A is the set of actions, I the initial state, and G the set of goal conditions. A state s is a mapping of each variable to a value in its domain, where $s(v)$ indicates the value of v in s.

Conditions can be *propositional* or *numeric*. The former correspond to $v_p \in V_p$ being *true*, while the latter are of the form $c : \xi \trianglerighteq 0$, where $\trianglerighteq \in \{\geq, >, =\}$ and $\xi = \sum_{v \in V_n} w_v^c v + w_0^c$ is a linear expression over V_n with $w_v^c, w_0^c \in \mathbb{Q}$. C is the set of all numeric conditions. Goals are propositional, G_p, or numeric, G_n, conditions.

An action $a \in A$ is a tuple $\langle pre(a), \mathit{eff}(a), cost(a) \rangle$: $pre(a)$ and $\mathit{eff}(a)$ are sets of preconditions and effects, and $cost(a) > 0$ is the action cost. Preconditions can be propositional, $pre_p(a)$, or numeric, $pre_n(a)$. Effects are defined as $\mathit{eff}(a) = \langle add(a), del(a), num(a) \rangle$, with $add(a), del(a) \subseteq pre_p(a)$ as the added and deleted propositions, and $num(a)$ the set of numeric effects. Numeric effects are assignments $v := \xi$, where $\xi = \sum_{w \in V_n} k_w^{v,a} w + k^{v,a}$ is a linear expression over V_n with $k_w^{v,a}, k^{v,a} \in \mathbb{Q}$. Each action has at most one numeric effect on each numeric state variable.

Action $a \in A$ is applicable in a state s iff $s(v_p) = \mathit{true}$ $\forall v_p \in pre_p(a)$ and $s(\xi) \trianglerighteq 0$ for all $c : \xi \trianglerighteq 0 \in pre_n(a)$, 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: $\forall v_p \in V_p$, $s'(v_p) = \mathit{true}$ if $v_p \in add(a)$, $s'(v_p) = \mathit{false}$ if $v_p \in del(a) \backslash add(a)$, 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 applicable actions $a_0, \ldots, a_n$, such that all goals are satisfied in the final state. A cost-optimal plan, $\hat{\pi}$, is a plan with minimum cost.

3 Related Work

Recent work on cost-optimal numeric planning has extended admissible heuristics for classical planning to handle actions

with constant effects on numeric variables and linear conditions, known as *simple numeric conditions* (Scala, Haslum, and Thiébaux 2016). Admissible heuristics for these problems are the sub-goaling heuristic $\hat{h}^{rmax}$ (Scala, Haslum, and Thiébaux 2016), cost-optimal partitioning based on landmarks h^{lma+} (Scala et al. 2017), and delete relaxation and network flow heuristics (Piacentini et al. 2018a). While non-admissible heuristics (Hoffmann 2003; Coles et al. 2008; Scala et al. 2016a; Illanes and McIlraith 2017) and SMT compilations (Scala et al. 2016b) for linear effects have been proposed, none have considered the optimal setting.

In classical planning, two IP models (Vossen et al. 1999) have been used to find feasible plans: a *state-base* model, which encodes proposition values in each state using binary variables, and a *state-change* model, which uses four variables to represent the change of value of each proposition in every state. van den Briel, Vossen, and Kambhampati (2005) extended the latter model to SAS+ planning (Bäckström and Nebel 1995). Here we extend the *state-change* model.

State-change IP Model for Classical Planning. Consider $\Pi = \langle V_p, V_n, A, I, G\rangle$ with $V_n = \emptyset$ and $T \in \mathbb{Z}^+$. Let $\mathcal{T} = \{0, ..., T-1\}$ and $\tilde{\mathcal{T}} = \mathcal{T} \cup \{T\}$ be sets of time-steps. For each $p \in V_p$, let $pnd(p) = \{a \in A : p \in pre_p(a), p \notin del(a)\}$ be the set of actions that require and do not delete p, $anp(p) = \{a \in A : p \notin pre_p(a), p \in add(a)\}$ the set of actions that add and do not require p, and $pd(p) = \{a \in A : p \in pre_p(a), p \in del(a) \backslash add(a)\}$ the set of actions that require and delete p.

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. Consider variables $u^a_{p,t}$, $u^{pa}_{p,t}$, $u^{pd}_{p,t}$ and $u^m_{p,t} \in \{0,1\}, \forall p \in V_p, \forall t \in \tilde{\mathcal{T}}$. Variable $u^a_{p,t}$ indicates whether p is added at time-step t but not required before, while $u^{pa}_{p,t}$ indicates whether p is required and not deleted by any action at time-step t. Variable $u^{pd}_{p,t} = 1$ if p is deleted and not added at time-step t but is required before, and $u^m_{p,t} = 1$ if p is true at time-step t and is not required nor deleted.

The *state-change* model $\mathcal{SC}(\Pi, T)$ (Vossen et al. 1999) is shown in Figure 1. Constraints (1) and (2) represent the initial state and goal conditions, respectively. Constraints (3)-(5) update the value of the state change variables. It should be noted that only one action at each time-step with a negative effect on the same proposition is allowed (5). Constraints (6)-(7) enforce actions preconditions and effects. Constraints (8)-(10) avoid the simultaneous application of conflicting actions. Constraint (11) propagates the value of the state change variables from one time-step to the next.

$$\min \sum_{a \in A, t \in \mathcal{T}} cost_a u_{a,t} \qquad (\mathcal{SC}(\Pi, T))$$

$$\text{s.t. } u^a_{p,0} = I(p) \qquad \forall p \in V_p \quad (1)$$

$$u^a_{p,T} + u^{pa}_{p,T} + u^m_{p,T} \geq 1 \qquad \forall p \in G_p \quad (2)$$

$$\sum_{a \in pnd(p)} u_{a,t} \geq u^{pa}_{p,t+1} \qquad \forall p \in V_p, \forall t \in \mathcal{T} \quad (3)$$

$$\sum_{a \in anp(p)} u_{a,t} \geq u^{a}_{p,t+1} \qquad \forall p \in V_p, \forall t \in \mathcal{T} \quad (4)$$

$$\sum_{a \in pd(p)} u_{a,t} = u^{pd}_{p,t+1} \qquad \forall p \in V_p, \forall t \in \mathcal{T} \quad (5)$$

$$u_{a,t} \leq u^{pa}_{p,t+1} \quad \forall p \in V_p, \forall a \in pnd(p), \forall t \in \mathcal{T} \quad (6)$$

$$u_{a,t} \leq u^{a}_{p,t+1} \quad \forall p \in V_p, \forall a \in anp(p), \forall t \in \mathcal{T} \quad (7)$$

$$u^a_{p,t} + u^m_{p,t} + u^{pd}_{p,t} \leq 1 \qquad \forall p \in V_p, \forall t \in \tilde{\mathcal{T}} \quad (8)$$

$$u^{pa}_{p,t} + u^m_{p,t} + u^{pd}_{p,t} \leq 1 \qquad \forall p \in V_p, \forall t \in \tilde{\mathcal{T}} \quad (9)$$

$$u_{a,t} + u_{a',t} \leq 1 \qquad \forall a, a' \in \tilde{A} \text{ s.t. } a \neq a' \wedge del(a) \cap (add(a') \cup pre(a')) \neq \emptyset \forall t \in \tilde{\mathcal{T}} \quad (10)$$

$$u^{pa}_{p,t+1} + u^m_{p,1} + u^{pd}_{p,t+1} \leq u^a_{p,t} + u^{pa}_{p,t} + u^m_{p,t} \qquad \forall p \in V_p \forall t \in \mathcal{T} \quad (11)$$

Figure 1: The state-change model for classical planning (Vossen et al. 1999).

4 MILP Model of Numeric Planning Tasks

We extend $\mathcal{SC}(\Pi, T)$ to $\mathcal{SCN}(\Pi, T)$. All constraints of $\mathcal{SC}(\Pi, T)$ are included in $\mathcal{SCN}(\Pi, T)$. For modeling purposes, we partition the set of actions affecting a numeric variable $v \in V_n$ into: $se(v) = \{a \in A : (v \coloneqq v + k^{v,a}) \in num(a)\}$ the set of actions that change v via constant effects, and $le(v) = \{a \in A : (v \coloneqq \xi) \in num(a), a \notin se(v)\}$ the set of actions that change v via linear effects.

Given an action $a \in A$, we call $nmutex(a)$ (*numeric mutex of* a) the set of mutex actions of a due to an interference of some numeric variables.

Definition 4.1. Given actions $a, a' \in A$, a' is *numeric mutex* to a if there exists a variable $v \in V_n$ such that $(v \coloneqq \xi) \in num(a)$ and either: (i) v is used in one of the numeric effects of a', i.e., $\exists v' \in V_n$ such that $(v' \coloneqq \xi') \in num(a')$ and $v \in \xi'$, or (ii) v is part of a precondition of a', i.e., $\exists\, (c : \sum_{v \in V_n} w^c_v v + w^c_0 \trianglerighteq 0) \in pre_n(a')$ with $w^c_v \neq 0$.

Our definition of mutex is more restrictive than that allowed by PDDL2.1 (Fox and Long 2003). Nonetheless, since optimal plans minimize the total action cost, the model does not rule out any solutions, provided that the maximum time horizon is increased adequately. This is because there will be a corresponding solution with the same set of actions and some extra time-steps.

4.1 MILP Formulation

Consider parameters $m_{c,t} \in \mathbb{Q}$, $\forall c \in C, \forall t \in \tilde{\mathcal{T}}$, $M^{step}_{v,t}, m^{step}_{v,t}, M^a_{v,t}, m^a_{v,t} \in \mathbb{Q}$, $\forall v \in V_n, \forall t \in \tilde{\mathcal{T}}$. Let $y_{v,t} \in \mathbb{Q}$ $\forall v \in V_n, \forall t \in \tilde{\mathcal{T}}$ represent the value of the numeric variable v at time-step t. The constraints modeling numeric effects and conditions are given in Figure 2. Constraint (12) sets the variables to their initial state values, while constraint (13) enforces the numeric goal conditions. Constraint (14) ensures the satisfaction of numeric preconditions. Constraints (15)-(18) update the values of the numeric variables according to the action effects. Constraint (19) enforces the mutex action relation.

$$y_{v,0} = I(v) \qquad \forall v \in V_n \tag{12}$$

$$\sum_{v \in V_n} w_v^c y_{v,T} + w_0^c \trianglerighteq 0 \qquad \forall c \in G_n \tag{13}$$

$$\sum_{v \in V} w_v^c y_{v,t} + w_0^c \trianglerighteq m_{c,t}(1 - u_{a,t}) \qquad \forall a \in A, \forall c \in pre_n(a), \forall t \in \mathcal{T} \tag{14}$$

$$y_{v,t+1} \leq y_{v,t} + \sum_{a \in se(v)} k^{v,a} u_{a,t} + M^{step}_{v,t+1} \sum_{a \in le(v)} u_{a,t} \qquad \forall v \in V_n, \forall t \in \mathcal{T} \tag{15}$$

$$y_{v,t+1} \geq y_{v,t} + \sum_{a \in se(v)} k^{v,a} u_{a,t} + m^{step}_{v,t+1} \sum_{a \in le(v)} u_{a,t} \qquad \forall v \in V_n, \forall t \in \mathcal{T} \tag{16}$$

$$y_{v,t+1} - \sum_{w \in V_n} k_w^{v,a} y_{w,t} \leq k^{v,a} + M^a_{v,t+1}(1 - u_{a,t}) \qquad \forall v \in V_n, \forall a \in le(v), \forall t \in \mathcal{T} \tag{17}$$

$$y_{v,t+1} - \sum_{w \in V_n} k_w^{v,a} y_{w,t} \geq k^{v,a} + m^a_{v,t+1}(1 - u_{a,t}) \qquad \forall v \in V_n, \forall a \in le(v), \forall t \in \mathcal{T} \tag{18}$$

$$u_{a,t} + u_{a',t} \leq 1 \;\; \forall a \in A, \forall a' \in nmutex(a) \forall t \in \mathcal{T} \tag{19}$$

Figure 2: Constraints for numeric effects and conditions.

The model considers two ways to update the values of a numeric variable. Constraints (15)-(16) update $v \in V_n$ when using actions with constant effects, and also propagate the value of v from time-step t to $t+1$, if no action affecting v is applied. When a variable is modified via linear effects, constraints (17)-(18) force the terms containing $M^a_{v,t}$ (and $m^a_{v,t}$) to 0, updating the variable accordingly, while constraints (15)-(16) become redundant. Although constant effects could be encoded with constraints (17)-(18), constraints (15)-(16) provide a tighter linear (LP) relaxation.[1]

4.2 Big-M Values

Big-M constraints are an MILP technique to express logical implications. E.g., $y = 1 \Rightarrow a^\top x \leq b$, with binary variable y, can be expressed as $a^\top x + My \leq b + M$, provided that $M \geq a^\top x - b$. Smaller M values (which are still upper bounds on $a^\top x - b$) compute tighter LP relaxations.

We use big-M constraints to model numeric preconditions and effects (14)-(18). Given a time-step t we compute $M_{v,t}$ and $m_{v,t}$ propagating the bounds of $v \in V_n$ as shown in Figure 3. The bounds are calculated starting from the values of the variables in the initial state and iteratively adding the maximum (minimum) effects that can be obtained by applying all the actions.

4.3 Correctness of the Encoding

To show the correctness of our model, we provide a mapping of any feasible plan π of a numeric planning problem Π to a feasible solution of $\mathcal{SCN}(\Pi, |\pi|+1)$ and inversely given a feasible solution of the $\mathcal{SCN}$ model, we provide a mapping to a valid plan π of Π.

[1]The complete proofs of this claim and the following propositions can be found in the technical report (Piacentini et al. 2018b).

$$m_{c,t} = \sum_{v \in V_n : w_v^c < 0} w_v^c M_{v,t} + \sum_{v \in V_n : w_v^c > 0} w_v^c m_{v,t} + w_0^c$$

$$M^{step}_{v,t} = M_{v,t} - m_{v,t-1}$$

$$m^{step}_{v,t} = m_{v,t} - M_{v,t-1}$$

$$M^a_{v,t} = M_{v,t} - \sum_{\substack{w \in V_n: \\ k_w^{v,a} < 0}} k_w^{v,a} M_{v,t-1} + \sum_{\substack{w \in V_n: \\ k_w^{v,a} > 0}} k_w^{v,a} m_{v,t-1} - k^{v,a}$$

$$m^a_{v,t} = m_{v,t} - \sum_{\substack{w \in V_n: \\ k_w^{v,a} > 0}} k_w^{v,a} M_{v,t-1} + \sum_{\substack{w \in V_n: \\ k_w^{v,a} < 0}} k_w^{v,a} m_{v,t-1} - k^{v,a}$$

where:

$$M_{v,0} = m_{v,0} = I(v),$$

$$M_{v,t} = \max \left\{ M_{v,t-1} + \sum_{a \in se(v): k_{v,a} > 0} k_{v,a}, \max_{a \in le(v)} \bar{a}(v) \right\}$$

$$m_{v,t} = \min \left\{ m_{v,t-1} + \sum_{a \in se(v): k_{v,a} < 0} k_{v,a}, \min_{a \in le(v)} \underline{a}(v) \right\}$$

$$\bar{a}(v) = \sum_{w \in V_n : k_w^{v,a} > 0} k_w^{v,a} M_{w,t-1} + \sum_{w \in V_n : k_w^{v,a} < 0} k_w^{v,a} m_{w,t-1}$$

$$\underline{a}(v) = \sum_{w \in V_n : k_w^{v,a} > 0} k_w^{v,a} m_{w,t-1} + \sum_{w \in V_n : k_w^{v,a} < 0} k_w^{v,a} M_{w,t-1}$$

Figure 3: Calculating the big-M constants.

Definition 4.2. Given a numeric planning task Π, we define a mapping $\mathcal{M}$ from a plan π of length $|\pi|$ of Π to a solution S of $\mathcal{SCN}(\Pi, |\pi|+1)$ as follows:

- $\forall a \in A$, $\forall t \in \mathcal{T}$: $u_{a,t} = 1$ if a appears at time-step t of π, 0 otherwise;
- $\forall p \in V_p$, $\forall t \in \mathcal{T}$: the values $u^{pd}_{p,t}$, $u^{pa}_{p,t}$, $u^a_{p,t}$ and $u^m_{p,t}$ are assigned according to their definitions.
- $\forall v \in V_n$: $y_{v,0} = I(v)$ and $y_{v,t} = a_{t-1}(...(a_0(I(v))))$, $\forall t \in \tilde{T} \backslash \{0\}$.

Proposition 4.1. Given a numeric planning task Π and a plan π, a solution $S = \mathcal{M}(\pi)$ satisfies all the constraints of $\mathcal{SCN}(\Pi, |\pi|+1)$.

Definition 4.3. Given a numeric planning task Π, we define a mapping $\tilde{\mathcal{M}}$ from a solution S of $\mathcal{SCN}(\Pi, T)$ to a sequential plan π of Π by taking the sequence of the actions $a \in A$ for which $u_{a,t} = 1$ in ascending order of $t \in \mathcal{T}$. If for a time-step t more than one action has $u_{a,t} = 1$, then any arbitrary ordering of the actions can be chosen.

Proposition 4.2. Given a feasible solution S of $\mathcal{SCN}(\Pi, T)$, a plan $\tilde{\mathcal{M}}(S)$ is a feasible plan for Π.

The proofs of these propositions are based on showing that the constraints of $\mathcal{SCN}(\Pi, T)$ correctly capture the semantics of numeric planning (Piacentini et al. 2018b).

4.4 Valid Inequalities

While $\mathcal{SCN}(\Pi, T)$ is sufficient to model the problem, we can add valid (redundant) constraints to tighten the model.

Landmark Constraints. Consider F_L the set of *fact landmarks*, i.e., the set of propositions that have to be achieved at least once in every feasible plan, and A_L the set of *action landmarks*, i.e., actions that must be present in every feasible plan. Then, constraints (20) and (21) are valid.

$$\sum_{t \in \hat{\mathcal{T}}} u^a_{p,t} + u^{pa}_{p,t} + u^m_{p,t} \geq 1 \qquad \forall p \in F_L \qquad (20)$$

$$\sum_{t \in \mathcal{T}} u_{a,t} \geq 1 \qquad \forall a \in A_L \qquad (21)$$

Landmarks are extracted using an extension of the algorithm proposed by Imai and Fukunaga (2015).

Relevance Analysis. Irrelevant actions are actions that never contribute to achieving a goal condition or a precondition of another relevant action. Given a set of irrelevant actions, constraint (22) is valid (Imai and Fukunaga 2015).

$$u_{a,t} = 0 \qquad \forall a \in A \text{ s.t. } a \text{ is irrelevant}, \forall t \in \mathcal{T} \qquad (22)$$

5 Iterative Time Horizon Allocation

Model $\mathcal{SCN}(\Pi, T)$ assumes a time horizon T. An initial T can be estimated from an admissible heuristic in the initial state, either using the heuristic value or, when the heuristic calculates a relaxed plan, from the relaxed plan length. We then solve the MILP model, and iteratively increase the value of T by one, until a solution is found. Let π^*_T be the optimal plan for the smallest T that admits a feasible solution. Plan π^*_T is not necessarily optimal for the planning task Π, as there might be lower cost solutions with a longer horizon. If the cost of π^*_T is equal to the admissible heuristic of the initial state or is less than $T \cdot \min_{a \in A} cost(a)$, then the solution is optimal. Otherwise, to guarantee optimality, we solve the MILP with time-horizon $\hat{T} = cost(\pi^*_T) / \min_{a \in A} cost(a)$, with the following additional constraints.

$$\sum_{a \in A} u_{a,t} \leq 1 \qquad \forall t \in \mathcal{T} \qquad (23)$$

$$\sum_{a \in A} u_{a,t+1} \leq \sum_{a \in A} u_{a,t} \qquad \forall a \in A, \forall t \in \mathcal{T} \qquad (24)$$

Constraint (23) forces a plan to be sequential, while constraint (24) eliminates symmetries caused by time-steps without any actions. Optimality can be guaranteed only when the problem does not have zero-cost actions.

6 Empirical Evaluation

We evaluate $\mathcal{SCN}(\Pi, T)$ with the iterative time horizon allocation (C^{SC}) on domains featuring linear effects and simple numeric conditions. In both cases, we start with a T equal to the number of time-steps of the relaxed plan calculated by h^c_{IP} in the initial state (Piacentini et al. 2018a). In domains with linear effects, h^c_{IP} ignores the numeric conditions. We run every instance with 30 minutes time and 4 GB memory limits. All models are solved using CPLEX 12.7.

Domain	#	h^{blind}		$h^{c,prop}_{IP}$		C^{SC}		VBS
		C	T	C	T	C	T	
Counters-le	15	6	13.6	6	117.0	**7**	0.5	7
Sailing-le	25	1	0.3	1	4.0	**4**	0.1	4
Rover-le	20	3	229.1	**4**	0.8	**4**	1.3	4
Total	60	10	76.9	11	70.8	**15**	**0.7**	15

Table 1: Coverage (C) and average execution time in seconds (T) of domains with linear and simple effects. VBS is the number of problems solved by at least one method.

Domain	#	$\hat{h}^{rmax}$		h^{lma+}		h^c_{IP}		C^{SC}		VBS
		C	T	C	T	C	T	C	T	
Counters	15	6	2.5	7	2.3	12	0.3	**15**	0.0	15
Gardening	63	**63**	3.5	**63**	4.8	**63**	6.1	**63**	11.1	63
Sailing	25	14	1.3	5	2.6	**22**	32.1	21	1.4	23
Sailing (1-20)	20	12	24.6	**20**	18.7	12	454.4	19	9.3	20
Farmland	30	**30**	11.1	**30**	27.0	**30**	22.6	**30**	11.0	30
Rover	20	4	1.4	4	1.2	**7**	1.8	4	2.7	7
Depots	20	2	0.4	4	0.7	**5**	0.7	1	4.0	5
Satellite	20	2	75.6	3	1.7	3	11.8	**4**	197.0	4
Zeno Travel	20	6	7.0	7	12.9	**8**	13.9	5	144.5	8
Total	233	139	**8.3**	143	11.2	**162**	53.0	**162**	17.8	175

Table 2: Coverage (C) and average execution time in seconds (T) of domains with simple condition effects only. VBS is the number of problems solved by at least one method.

Simple-condition domains are taken from the literature (Scala et al. 2017), while we develop three domains with linear effects by modifying *counters*, *sailing* and *rovers*.

In domains with linear effects, we compare our compilation with an A* search using two simple admissible heuristics: a goal sensitive heuristic (h^{blind}), that returns 0 if the state is a goal state, 1 otherwise, and the h^c_{IP} heuristic that ignores numeric conditions ($h^{c,prop}_{IP}$) (Imai and Fukunaga 2015). Table 1 reports the coverage of the problems solved and the average execution time. C^{SC} solves only 15 of 60 instances and any instance solved by another approach is also solved by C^{SC}. Clearly, these state-of-the-art results leave much room for improvement.

For domains with simple conditions, we compare against three heuristics: $\hat{h}^{rmax}$ (Scala, Haslum, and Thiébaux 2016), h^{lma+} (Scala et al. 2017), and h^c_{IP} (Piacentini et al. 2018a). Table 2 reports the coverage and the average execution time. The results show that C^{SC} achieves the same coverage as the best heuristic-search planner while solving four instances in the simple condition domains that none of heuristic-search planners could solve.

7 Conclusion

We presented a mixed integer linear programming compilation for numeric planning problems with instantaneous actions and linear numeric conditions and effects. Linear effects are particularly challenging due to their state-dependent nature and admissible heuristics that account for them have yet to be developed. Our experimental evaluation showed that while our compilation can deal with these problems, they still remain difficult to consistently solve in a reasonable run-time. When considering domains with only simple numeric conditions, our MILP model is competitive with state-of-the-art heuristic search approaches.

Acknowledgements

We would like to thank the anonymous reviewers whose feedback helped improve the paper. We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada and CONICYT (Becas Chile).

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS+ planning. *Computational Intelligence* 11:625–656.

Cashmore, M.; Fox, M.; Long, D.; and Magazzeni, D. 2016. A Compilation of the Full PDDL + Language into SMT. In *ICAPS 2016*, 79–87.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. A Hybrid Telaxed Planning Graph-LP Heuristic for Numeric Planning Domains. In *ICAPS 2008*, 52–59.

Fox, M., and Long, D. 2003. PDDL2. 1: An Extension to PDDL for Expressing Temporal Planning Domains. *JAIR* 20:61–124.

Hoffmann, J. 2003. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *JAIR* 20:291–341.

Illanes, L., and McIlraith, S. A. 2017. Numeric planning via abstraction and policy guided search. In *IJCAI 2017*, 4338–4345.

Imai, T., and Fukunaga, A. 2015. On a practical, integer-linear programming model for delete-free tasks and its use as a heuristic for cost-optimal planning. *JAIR* 54:631–677.

Kautz, H., and Walser, J. P. 1999. State-space planning by integer optimization. *AAAI 1999* 1:8.

Piacentini, C.; Castro, M.; Cire, A. A.; and Beck, J. C. 2018a. Linear and integer programming-based heuristics for cost-optimal numeric planning. In *AAAI 2018*.

Piacentini, C.; Castro, M.; Cire, A. A.; and Beck, J. C. 2018b. Online Appendix to "Compiling Optimal Numeric Planning to Mixed Integer Linear Programming", published at ICAPS2018. Technical Report 2018A, Toronto Intelligent Decision Engineering Laboratory, University of Toronto. tidel.mie.utoronto.ca/pubs/TechReport2018A.pdf.

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

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2016a. Interval-based relaxation for general numeric planning. In *ECAI*, 655–663.

Scala, E.; Ramirez, M.; Haslum, P.; and Thiebaux, S. 2016b. Numeric Planning with Disjunctive Global Constraints via SMT. In *ICAPS 2016*, 276–284.

Scala, E.; Haslum, P.; Magazzeni, D.; and Thiebaux, S. 2017. Landmarks for numeric planning problems. In *IJCAI 2017*.

Scala, E.; Haslum, P.; and Thiébaux, S. 2016. Heuristics for numeric planning via subgoaling. In *IJCAI 2016*, 3228–3234.

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, V. 2011. CPT4: An Optimal Temporal Planner Lost in a Planning Competition without Optimal Temporal Track. In *The 7th IPC: Description of Participant Planners of the Deterministic Track*. 25–28.

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.