The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Linear and Integer Programming-Based Heuristics for Cost-Optimal Numeric Planning

Chiara Piacentini,[†] Margarita P. Castro,[†] Andre A. Cire,[‡] 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

Linear programming has been successfully used to compute admissible heuristics for cost-optimal classical planning. Although one of the strengths of linear programming is the ability to express and reason about numeric variables and constraints, their use in numeric planning is limited. In this work, we extend linear programming-based heuristics for classical planning to support numeric state variables. In particular, we propose a model for the interval relaxation, coupled with landmarks and state equation constraints. We consider both linear programming models and their harder-to-solve, yet more informative, integer programming versions. Our experimental analysis shows that considering an NP-Hard heuristic often pays off and that A* search using our integer programming heuristics establishes a new state of the art in cost-optimal numeric planning.

1 Introduction

Cost-optimal planning is the problem of selecting a minimal-cost sequence of actions to achieve a set of goals. Several works (e.g. (Karpas and Domshlak 2009; Pommerening, Röger, and Helmert 2013)) focus on its classical version, in which states are represented by propositional variables. This classical representation limits the expressive power of the formalism, as problems with unbounded domain variables cannot be expressed. The encoding of variables with discrete and finite domains is possible, but arithmetic operations require at least a number of propositions that is quadratic in the variables' domain.

Numeric planning is an extension of classical planning in which state variables can assume numeric values and conditions can be represented as numeric expressions over such variables. Its higher expressive power makes numeric planning more attractive for real-world applications. However, this expressivity comes at a price: numeric planning problems are, in the general case, *undecidable*, while classical planning is in P-SPACE (Helmert 2002).

Integer and linear programming (IP and LP, respectively) are optimization techniques for problems that can be formulated over a set of integer or continuous variables subject to linear constraints and a linear objective function. A number of works in classical planning use LP models to calculate heuristic values. Given the ability of LP to reason with numeric constraints, its use in numeric planning has shown to be promising. For example, the planner LPRPG integrates LP models with Relaxed Planning Graphs (RPG) to calculate upper and lower bounds of numeric resources when applying actions (Coles et al. 2008). The hybrid landmark heuristic (Scala et al. 2017) exploits LP models to compute the minimal number of actions required to achieve numeric conditions identified as landmarks. The result of the LP model is then used as an admissible estimate to solve cost-optimal numeric planning problems.

In this paper, we explore the use of IP and LP models to calculate admissible heuristics for numeric planning problems, where actions modify numeric state variables by adding constant quantities. We adapt the IP model of the delete relaxation in classical planning (Imai and Fukunaga 2015) to the interval relaxation for numeric planning. In addition, we extend this model to include constraints from the classical operator counting framework (Pommerening et al. 2015). We experiment with both IP models and their LP relaxations on a set of benchmark domains. In contrast to classical planning, we find that our IP heuristics are preferable to their LP counterparts as they outperform the state-of-the-art numeric planners.

2 Notation and Preliminaries

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

A numeric planning *task* is a 5-tuple $\Pi = \langle V_p, V_n, A, I, G\rangle$, where V_p is the set of *propositional* variables, V_n is the set *numeric* variables, A is the set of action operators, I is the initial state, and G is a set of goal conditions. Propositional variables $v_p \in V_P$ assume binary domains $\{true, false\}$, while numeric variables $v_n \in V_n$ have numeric domains $\mathbb{Q}$. A state s is a mapping of all the variables to a value in their respective domains. We denote $s(v)$ to be the value of a variable v in state s. In particular, I represents the initial state of the planning task and so $I(v)$ is the value of v in the initial state.

In numeric planning, conditions can be of two types: *propositional* or *numeric*. A propositional condition corresponds to a variable $v_p \in V_p$ being *true*. A numeric condition c is defined as $c : \xi \trianglelefteq 0$, where $\trianglelefteq \in \{\leq, <, =\}$ and ξ is

an arithmetic expression over V_n and $\mathbb{Q}$.

Each action $a \in A$ is defined by a 3-tuple $a = \langle pre, \mathit{eff}, cost \rangle$, where pre is the set of preconditions, eff the set of effects, and $cost \geq 0$ is the action cost. Preconditions are defined as $\langle pre_p, pre_n \rangle$, propositional and numeric conditions, respectively. The effects of an action $a \in A$ correspond to a 3-tuple $\mathit{eff} = \langle add, del, num \rangle$, where add and del are subsets of V_p representing added and deleted propositions, and num is the set of numeric effect. A numeric effect is the assignment of a numeric variable $v_n \circ \xi$ with $\circ \in \{=, +=, -=, *=, /=\}$ and ξ a numeric expression. We assume that each action has at most one numeric effect on each numeric variable.

Action $a \in A$ is applicable in a state s iff all its preconditions are satisfied in s. Formally, $v_p \in pre_p(a)$ is satisfied if $s(v_p) = true$; $(c : \xi \trianglelefteq 0) \in pre_n(a)$ is satisfied if $s(\xi) \trianglelefteq 0$, 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 obtained as follows. For each $v_p \in V_p$, $s'(v_p) = \textit{true}$ if $v_p \in add(a)$, $s'(v_p) = \textit{false}$ if $v_p \in del(a)/add(a)$, and $s'(v_p) = s(v_p)$ otherwise. For each $v_n \in V_n$, $s'(v_n) \circ s(\xi)$ if $(v_n \circ \xi) \in num(a)$, and $s'(v_n) = s(v_n)$ otherwise.

Similar to action preconditions, goal conditions can be either propositional (G_p) or numeric (G_n). Propositional goal conditions are of the form $v_p \in G_p \subseteq V_p$ and numeric goal conditions are $c : \xi \trianglelefteq 0$. We call C the set of numeric conditions that appear as preconditions of actions or as goals; therefore a goal condition is $c \in G_n \subseteq C$.

A sequential plan π^s is a sequence of applicable actions $a_0, \ldots, a_n$, such that all the propositional and numeric conditions are satisfied in $s_G = \pi^s(I)$, where $\pi^s(I) := a_n(a_{n-1}(\ldots(a_0(I))))$. A cost-optimal plan is a sequential plan $\hat{\pi}^s$ such that $cost(\hat{\pi}^s) = \sum_{a \in \hat{\pi}^s} cost(a) \leq cost(\pi^s)$ for any sequential plan π^s of Π.

A partial order plan $\pi^p = \pi_0, \ldots, \pi_n$ is a sequence of plan-steps π_i, where π_i is a set of actions applied at time step i. A partial order plan is valid if every sequential plan derived by sequencing the actions in the plan-steps is valid. Similarly, a cost-optimal partial order plan is a plan $\hat{\pi}^p$ such that $cost(\hat{\pi}^p) = \sum_{\pi_i \in \hat{\pi}^p} \sum_{a \in \pi_i} cost(a) \leq cost(\pi^p)$ for any partial order plan π^p of Π.

This work considers the restricted form of numeric planning, where planning tasks only have *simple numeric conditions* (Scala, Haslum, and Thiébaux 2016). Specifically, numeric effects of an action a assume the form $v \mathrel{+}= k_v^a$, with $k_v^a \in \mathbb{Q}$, and all the numeric conditions are linear expressions of the numeric variables, i.e., $c : \sum_{v \in V_n} w_v^c v + w_0^c \trianglelefteq 0$, where w_v^c and w_0^c are in $\mathbb{Q}$.

2.1 Delete and Interval Relaxations

The *delete relaxation* is a well-known relaxation of classical planning tasks which ignores the delete effects of the actions. Similarly, numeric planning defines the *interval relaxation* (Hoffmann 2003) which accumulates the values of the numeric variables as intervals.

Formally, given a planning task $\Pi = \langle V_p, V_n, A, I, G \rangle$, its interval relaxed task is defined as $\Pi^+ = \langle V_p, V_n^+, A^+, I, G \rangle$. Numeric variables $v_n \in V_n^+$ take interval values $v_n = [v_n^{min}, v_n^{max}]$ representing the range of possible values of the variable. Numeric expressions are defined recursively (Aldinger, Mattmüller, and Göbelbecker 2015) and action effects augment the range of the intervals. States satisfy numeric conditions if $\forall v_n \in V_n^+$ there exist a value $\mathtt{v}_n \in [v_n^{min}, v_n^{max}]$ for which the conditions are true. Lastly, actions ignore delete effects, i.e., $del(a) = \emptyset$ for each $a \in A^+$.

3 Related Work

Since the introduction of numeric state variables in the PDDL language, few works have addressed cost-optimal numeric planning with simple conditions. The first set of admissible heuristics (Scala, Haslum, and Thiébaux 2016) extends the classical planning sub-goaling heuristic h^{max} (Geffner and Haslum 2000) to consider numeric effects and conditions. Follow-on work extends the concept of landmarks to consider numeric conditions (Scala et al. 2017) and proposes an *AND/OR* graph landmark extraction algorithm based on a classical planning approach (Keyder, Richter, and Helmert 2010). The work uses the extracted landmarks together with an LP model to compute admissible heuristic estimations.

In contrast to cost-optimal numeric planning, a larger body of work deals with satisficing numeric planning. Earlier works focus on valid plan extraction with respect to the interval relaxation with the aim of getting informative inadmissible heuristics (Hoffmann 2003; Coles et al. 2008; 2013). Later, a random-walk approach was proposed for consumer-only numeric planning problems (Nakhost, Hoffmann, and Müller 2012). A recent work transforms a numeric task into a classical one via abstractions and a plan in the abstract space is used to guide the search in the numeric state space (Illanes and McIlraith 2017). Most of these works are valid for problems beyond *simple numeric conditions*, as they consider linear or polynomial effects of actions. The only heuristic dealing with more general numeric effects is the one proposed by Scala et al. (2016).

The use of mathematical programming is widely studied in classical planning. Bylander (1997) represents classical planning problems as IP models, which can be linearly relaxed to calculate admissible heuristics. LP models are also used to combine abstraction-based heuristics and to derive the minimal cost set of actions that achieves landmark conditions (Katz and Domshlak 2010; Pommerening, Röger, and Helmert 2013). Recently, network-flow heuristics are based on LP models that balance the number of actions that add and delete propositions (van den Briel et al. 2007; Bonet 2013). As shown by Imai and Fukunaga (2015), the delete relaxation task can be compactly encoded into an IP model, and its linear relaxation can be used as an admissible heuristic, achieving comparable performance with state-of-the-art heuristics in cost-optimal classical planning. All these heuristics are unified in the *operator counting* framework (Pommerening et al. 2015).

4 An IP Model for the Interval Relaxation

This section presents an IP model for the interval relaxation, obtained by extending the IP model for the delete relaxation (Imai and Fukunaga 2015) to consider numeric state variables. Our model reifies numeric conditions into boolean

variables, thus many of the constraints for classical planning tasks can be adopted to the numeric case. However, unlike the classical case, actions are not idempotent operators, meaning that their application does not always result in the satisfaction of determined conditions. Instead, actions can potentially achieve a numeric condition, depending on the values of the variables appearing in the condition. Scala, Haslum, and Thiébaux (2016) introduce the concept of *possible achiever* to identify actions that can potentially make a numeric condition true.

Definition 4.1. Given a numeric planning task Π, an action a is a *possible achiever* of a numeric condition c in state s if there exists an integer number m such that $\sum_{v \in V} w_v^c(k_{v,a}m + s(v)) + w_0^c \trianglelefteq 0$.

As a consequence, unlike classical planning, actions can appear multiple times in an optimal relaxed plan.

4.1 IP Model

Consider a planning task $\Pi = \langle V_p, V_n, A, I, G\rangle$ and its interval relaxation Π^+. We define model $IP(\Pi^+)$ as follows.

Let M and B be large constants, while variable $m_a \in \{0, ..., M\}$, $\forall a \in A$ represents the number of times an action a appears in a plan; $u_a \in \{0, 1\}$, $\forall a \in A$ indicates if action a appears at least once in the plan; $u_p \in \{0, 1\}$, $\forall p \in V_p$ represents whether proposition p is achieved in the plan; $u_c \in \{0, 1\}$, $\forall c \in C$ indicates whether a numeric condition c is satisfied in the last state; $e_{a,p} \in \{0, 1\}$, $\forall a \in A, \forall p \in add(a)$ indicates that action a is the first achiever of proposition p; $m_{a,c} \in \{0, ..., M\}$, $\forall a \in A, \forall c \in C$ represents the number of times action a is required to satisfy condition c; $e_{a,c} \in \{0, 1\}$, $\forall a \in A, \forall c \in C$ indicates if action a is used to achieve condition c; t_p, t_c and $t_a \in \{0, ..., |A|\}$, represent the time step at which $p \in V_p$, $c \in C$, and $a \in A$ are first added to the plan, respectively.

$$\min \sum_{a \in A} cost_a m_a$$

$$\text{s.t. } u_x = 1 \qquad \forall x \in G_p \cup G_n \quad (1)$$

$$u_x \geq u_a \qquad \forall a \in A, \forall x \in pre(a) \quad (2)$$

$$u_a \geq e_{a,x} \qquad \forall a \in A, \forall x \in C \cup add(a) \quad (3)$$

$$I(p) + \sum_{a \in A \text{ s.t. } p \in add(a)} e_{a,p} = u_p \qquad \forall p \in V_P \quad (4)$$

$$\sum_{v \in V_n} w_v^c \left[\sum_{a \in A \text{ s.t. } k_{v,a} \cdot w_v^c \leq 0} m_{a,c} k_{v,a} + I(v) \right] + w_0^c + B u_c \trianglelefteq B \qquad \forall c \in C \quad (5)$$

$$m_{a,c} - M e_{a,c} \leq 0 \qquad \forall a \in A, \forall c \in C \quad (6)$$

$$m_a \geq m_{a,c} \qquad \forall a \in A, \forall c \in C \quad (7)$$

$$t_x \leq t_a \qquad \forall a \in A, \forall x \in pre(a) \quad (8)$$

$$t_a + 1 \leq t_x + (|A| + 1)(1 - e_{a,x}) \qquad \forall a \in A, \forall x \in C \cup add(a) \quad (9)$$

Constraint (1) ensures that all goal conditions are achieved by a plan, while constraint (2) indicates that an action is applicable only if its preconditions are satisfied. Constraint (3) enforces the inclusion of actions that achieve a condition (propositional or numeric). Constraint (4) indicates that propositions achieved by the plan are either true in the initial state or added by some action. Constraint (5) enforces condition $c : \xi \trianglelefteq 0$ to be true only if variable u_c is equal to 1. Numeric expression ξ is given by the aggregate effects of the actions that contribute to the change of values of the variables in ξ and B is an upper bound on ξ. Constraint (6) sets the first achievers of all the numeric conditions. Constraint (7) sets variables m_a to the number of times action a is used. The last two constraints model sequencing requirement: constraint (8) indicates that an action cannot occur before its preconditions are satisfied and constraint (9) forces a condition to be true after its first achievers are applied.

Constraints (5), (6), and (9) represent logical implication using the so-called *big-M* constraints, a well known approach in IP to model $y = 1 \Rightarrow Ax \leq b$, for a binary variable y and any variable x. This can be equivalently written as $Ax \leq b + M(1 - y)$, which becomes redundant if $y = 0$ and M is sufficiently large. The value of M can impact the IP solver efficiency, as large M values can result in poor linear relaxations. It is therefore desirable to set M as the minimal upper bound of expression $Ax - b$. In constraint (9), this value is $|A| + 1$ since variables t_a and t_x are bound by the number of actions in the problem. However, it is not always possible to compute upper bounds for numeric conditions and for the number of times actions are applied (m_a), which are needed for constraints (5) and (6). In such cases we use arbitrarily large values for M and B to represent infinity.

4.2 Relation between IP Model and Delete Relaxation

We now show that plans of Π^+ respect constraints in $IP(\Pi^+)$, and any feasible solution S of $IP(\Pi^+)$ is equivalent to a partial order plan π of Π^+.

Definition 4.2. Given an interval relaxed planning task Π^+, we define a mapping $\mathcal{M}$ from a plan π of Π^+ to a solution S of $IP(\Pi^+)$ as follows:

- $\forall a \in A$, $u_a = 1$ if a appears at least once in π, 0 otherwise; m_a is equal to the number of times a is in π; t_a is the time-step at which a first appears in π or $|A|$ if $a \notin \pi$.
- $\forall p \in V_p$, if p is in the goal state $\pi(I)$ then $u_p = 1$ and $t_p = \min_{a \in \pi | p \in pre_p(a)} t_a$; otherwise $u_p = 0$ and $t_p = |A|$.
- $\forall c \in C$, if there exists an action $a \in \pi$ s.t. $c \in pre_n(a)$ or $c \in G_n$, $u_c = 1$ and $t_c = \min_{a \in \pi | c \in pre_n(a)} t_a$; otherwise $u_c = 0$ and $t_c = |A|$.
- $\forall a \in A, \forall p \in V_p$, $e_{a,p} = 1$ if $a \in \pi$ is the first action that adds p, 0 otherwise.
- $\forall c \in C, \forall a \in A$, if c appears in some preconditions of an action in π, $m_{a,c}$ is the number of times an action a appears in the plan π before the first time c is required and $e_{a,c} = 1$ if $m_{a,c} > 0$; $m_{a,c} = 0$ and $e_{a,c} = 0$ otherwise.

Proposition 4.1. Given an interval relaxed planning task Π^+ and a plan π of Π^+, a solution $S = \mathcal{M}(\pi)$ satisfies all the constraints of $IP(\Pi^+)$.

Proof. We can show that π satisfies constraints (1)-(4) and (8)-(9) using the same arguments as Imai and Fukunaga (2015) (Proposition 1). If condition $c : \xi \trianglelefteq 0$ never appears as a precondition of any action in π or is not part of the goals, then $u_c = 0$ and constraint (5) is satisfied if B is an upper bound of the expression ξ. Whenever c is a precondition of an action in π or a goal condition, $u_c = 1$ and constraint (5) becomes $\sum_{v \in V} w_v^c \mathtt{v} + w_0^c \trianglelefteq 0$, where $\mathtt{v}$ is the maximum (minimum) value of the interval of variable $v \in V_n$ if w_v^c is a positive (negative) value, at time-step t_c. Since c is valid in the interval relaxation, the constraint is also satisfied. Constraints (6) and (7) are satisfied by definition. □

Definition 4.3. Given an interval relaxed planning task Π^+, we define a mapping $\tilde{\mathcal{M}}$ from a solution S of $IP(\Pi^+)$ to a partial order plan π of Π^+ by sorting in ascending order of t_a all actions a for which $u_a = 1$ and inserting them m_a times at plan-step t_a.

Proposition 4.2. Given a feasible solution S of $IP(\Pi^+)$, a plan $\pi = \tilde{\mathcal{M}}(S)$ is a feasible plan for Π^+.

Proof. We need to prove that (i) the sequence of actions produces the goal conditions, (ii) every actions in plan-step π_i is applicable in the state $\pi_{i-1}(\dots(\pi_0(I)))$, and (iii) the ordering of actions at the same plan-step does not matter. The propositional part of conditions (i) and (ii) holds due to Proposition 2 by Imai and Fukunaga (2015).

We prove the numeric part of condition (i) by contradiction. Consider a goal condition $c \in G_n$ such that there is no combination of actions in π that satisfies c. Since S is a solution of $IP(\Pi^+)$, $u_c = 1$ and since the c is not satisfied in the initial state, there exist one or more actions a_j for which $m_{a_j,c} \geq 1$. Given constraints (3), (6) and (7) we get $e_{a_j,c} = 1$ and $u_{a_j} = 1$. Thus $a_j \in \pi$, contradicting the hypothesis.

We prove condition (ii) by induction. For the base case, assume that there exists an action in the first plan-step $a \in \pi_0$ and condition $c \in pre_n(a)$ such that c is not satisfied in the initial state. Since $u_a = 1$, constraint (2) forces $u_c = 1$. Then, given constraint (5), there exists a combination of actions, a_j, that makes condition c true, so $e_{a_j,c} = 1$ due to constraint (6). We have $t_{a_j} < t_a$ given (8)-(9), but a is in the first plan-step, contradicting the initial hypothesis. Hence, we prove the base case. Now consider that condition (ii) holds for $\pi_0, \dots, \pi_{i-1}$. Let $a \in \pi_i$ and $c \in pre_n(a)$ such that c is not satisfied in $\pi_{i-1}(\dots(\pi_0(I)))$. Then, there must exist an action $a^* \notin (\pi_0, \dots, \pi_{i-1})$ such that $u_{a^*} = 1$ and $t_{a^*} < t_{a_i}$. Then, c must be satisfied in $\pi_{i-1}(\dots(\pi_0(I)))$.

Condition (iii) is trivially satisfied in the interval relaxation because once an action is applicable, it remains applicable for the rest of the plan. □

Given an optimal solution $\hat{S}$ of $IP(\Pi^+)$, a plan $\pi = \tilde{\mathcal{M}}(\hat{S})$ is an optimal plan for Π^+.

5 Additional Constraints

In this section we present two classes of additional constraints that enhance the model. The first class corresponds to valid inequalities, i.e., a set of redundant constraints for the IP model that can provide tighter LP relaxations. The second class of constraints provides complementary information to the interval relaxation. In both cases, optimal solutions obtained when using these constraints correspond to lower bounds on the original planning task.

5.1 Valid Inequalities

Landmark Constraints In classical planning, a *fact landmark* is a proposition that is achieved at least once in every feasible plan. Similarly, action landmarks are those actions that must be present in every feasible solution of the planning task. Scala et al. (2017) extend the concept of fact landmarks to include numeric conditions and propose an algorithm to identify them. In the literature, algorithms to extract landmarks are based on Planning Graph propagation (Zhu and Givan 2003) or *AND/OR* Graphs (Keyder, Richter, and Helmert 2010). We compute numeric landmarks by extending the polynomial algorithm proposed by Imai and Fukunaga (2015). The algorithm extracts fact landmarks for each proposition $p \in V_p$ and numeric condition $c \in C$. The set of fact landmarks of the planning task (F_L) is the union of the fact landmarks of the goal conditions. Actions landmarks (A_L) are actions that are the sole achievers of a fact landmark. Although the algorithm is not complete, it generates sound landmarks. The exact relation between this algorithm and the one presented by Scala et al. (2017) is a subject of future investigation.

Given a set of fact (F_L) and action (A_L) landmarks, the following constraints can be added to $IP(\Pi^+)$:

$$u_f = 1 \qquad \forall f \in F_L \tag{10}$$

$$u_a = 1 \qquad \forall a \in A_L \tag{11}$$

Relevance Analysis The model can be enhanced via backchaining relevance analysis. We extend the definition of *relevant* actions (Imai and Fukunaga 2015) to numeric planning using the concept of *possible achiever* in Definition 4.1. Given an action a with some numeric effects, we call $\text{ADD}(a)$ the union of propositions $p \in add(a)$ and the set of numeric conditions for which a is a possible achiever. Let $\text{FADD}(a) \subseteq \text{ADD}(a)$ be the set of propositions and numeric conditions for which action a is a possible achiever and no $f \in \text{FADD}(a)$ is a fact landmark of a.

Definition 5.1. Given a planning task Π, an action a is *relevant* if: (i) $\text{FADD}(a) \cap G \neq \emptyset$, or (ii) there exists a relevant action a' satisfying $\text{FADD}(a) \cap pre(a') \neq \emptyset$.

Definition 5.2. Given a planning task Π, a condition $x \in V_p \cup C$ is relevant if: (i) $x \in G$, or (ii) there exists a relevant action a, such that $x \in pre(a)$.

Once the relevant facts and actions are identified, the following constraints can be added to $IP(\Pi^+)$ without pruning any optimal solutions (Imai and Fukunaga 2015):

$$u_x = 0 \qquad \forall x \in V_p \cup C \qquad \text{s.t. } x \text{ is not relevant} \tag{12}$$

$$u_a = 0 \qquad \forall a \in A \qquad \text{s.t. } a \text{ is not relevant} \tag{13}$$

Dominance Analysis Further variable pruning can be obtained using *dominated* actions. Dominated actions of a classical delete free task can be easily identified as the actions

whose add effects are a subset of another action's add effects. We extend the definition of dominance for numeric planning, adding a further condition for numeric effects.

Definition 5.3. Given an interval relaxed planning task Π^+, an action $a \in A$ is said to be *dominated* by an action $a' \in A$, if (i) $\text{FADD}(a) \subseteq \text{FADD}(a')$, (ii) $\forall f \in pre(a')$, f is a fact landmark for a or $f \in I$, (iii) $cost(a) \geq cost(a')$, and (iv) $\forall(v \mathrel{+}= k_{v,a}) \in num(a)$, there exists a $(v \mathrel{+}= k_{v,a'}) \in num(a')$ such that $k_{v,a} \cdot k_{v,a'} \geq 0$ and $|k_{v,a}| \leq |k_{v,a'}|$.

Condition (iv) indicates that an action is dominated by another if for every positive increase (decrease) of a variable of the dominated action, the dominating action positively increases (decreases) the same variables by a larger quantity.

Proposition 5.1. Consider an interval relaxed planning task Π^+ and an action $a \in A$. If there exists an action a' that dominates a and a plan π that contains a, then there is a plan π^* such that $a \notin \pi^*$ and $cost(\pi^*) \leq cost(\pi)$.

Proof. Following the arguments of Proposition 4 by Imai and Fukunaga, we can construct π^* by replacing a with a' in π. Given that a' dominates a, condition (ii) in Definition 5.3 guarantees that if a is applicable in a state, then a' is also applicable. In addition, conditions (i) and (iv) guarantee that all propositional and numeric conditions achieved by a are also achieved by a'. Lastly, $cost(\pi^*) \leq cost(\pi)$ follows from condition (iii). □

We can identify dominated actions (A_{dom}) in polynomial time with respect to the number of actions. Starting with $A_{dom} = \emptyset$, we iterate over all the actions a in A/A_{dom} and add to A_{dom} all the actions a' dominated by a, such that $a \neq a'$. The following constraint ensures that actions in A_{dom} are not part of any solution.

$$u_a = 0 \qquad \forall a \in A_{dom} \tag{14}$$

Inverse Actions Pruning We also consider inverse action pruning (Imai and Fukunaga 2015) for actions with only propositional effects.

Definition 5.4. Given an interval relaxed planning task Π^+, an action $a_1 \in A$ is said to be inverse of action $a_2 \in A$ if (i) $add(a_1) \subseteq pre_p(a_2)$, (ii) $add(a_2) \subset pre_p(a_1)$, (iii) $num(a_1) = \emptyset, num(a_2) = \emptyset$.

Constraint (15) is valid for $IP(\Pi^+)$, where $inv(a, p)$ is the set of inverse actions of a which have p as an *add* effect.

$$u_p - \sum_{a' \in inv(a,p)} e_{a',p} \geq u_a \quad \forall a \in A, \forall p \in pre_p(a) \tag{15}$$

5.2 Strengthening Constraints

A Simple Strengthening Constraint Model $IP(\Pi^+)$ represents the interval relaxed planning task Π^+. Since we are interested in strong admissible heuristics, we can strengthen some of the constraints to improve the estimation. Specifically, we modify constraint (5) to partially include negative effects.

Constraint (5) models the satisfaction of a numeric condition in the interval relaxation. A condition $\sum_{v \in V_n} k_v v + k_0 \trianglelefteq 0$ is satisfied if its variables assume their minimum values when $k_v \geq 0$ and their maximum values when $k_v < 0$. This is modeled in constraint (5) by considering only the increasing effects on the variables appearing with negative coefficients, and the decreasing effects on the variables with positive coefficients. To strengthen the constraint, we can replace (5) with the following constraint:

$$\sum_{v \in V} w_v^c \left[\sum_{a \in A} m_{a,c} k_{v,a} + I(v) \right] + w_0^c + B u_c \trianglelefteq B, \quad \forall c \in C \tag{5'}$$

Constraint (5′) considers the net effect of all the actions that are possible achievers of the numeric condition. The new model, $IP(\Pi^+)$ with constraint (5′), computes plans with greater or equal cost, while still providing a lower bound to the original planning task Π. This can be trivially proved by observing that $IP(\Pi^+)$ with constraint (5′) corresponds to the interval relaxation of a planning task where we augment the set of numeric state variables with an additional variable for each numeric condition $c : \xi \trianglelefteq 0$, equivalent to ξ. We call the extended interval relaxed planning task $\Pi'^+ = \langle V_p, V_n'^+, A^+, I, G \rangle$.

When considering constraint (5′) in conjunction with dominance analysis, Definition 5.3 has to include the additional variables introduced in Π'^+ in order to guarantee the validity of Proposition 5.1.

State Equation Constraints Model $IP(\Pi^+)$ can be coupled with constraints from the network flow model (van den Briel et al. 2007; Bonet 2013). In classical planning, this set of constraints restricts the number of times a proposition is deleted by considering the number of times it is added in the plan. Since such constraints take into account the delete effects of actions, the optimal solution of the resulting model does not represent the optimal relaxed plan of the planning task. Nevertheless, the optimal solution remains a lower bound on the original planning task (Pommerening et al. 2015).

In the propositional case, the state equation constraint takes the form (Imai and Fukunaga 2015):

$$g_p + \sum_{a | p \in predel(a)} m_a \leq I(p) + \sum_{a | p \in add(a)} m_a \tag{16}$$

where g_p is a parameter equal to 1 if $p \in G_p$, and 0 otherwise and $predel(a) = pre_p(a) \cap del(a)$.

A similar constraint can be written for variables u_c, but the extension to numeric state variables is not trivial as we generally do not know the target value of the numeric state variables. We can use, instead, the lower and upper bound information whenever it can be determined (Coles et al. 2008). We can calculate an upper bound of a numeric state variable $v \in V_n$ if $\forall a \in A$: (i) a has an effect $v \mathrel{+}= k_{v,a}$, $k_{v,a} \geq 0$; (ii) a has a precondition of the type $v \leq w_a$. Then, the upper bound of v is $ub_v = \max_{a \in A}(w_a + k_{v,a})$[1]. The resulting

[1]The calculation of the lower bound lb_v of a numeric state variable v is analogous.

numeric state equation constraints are:

$$\sum_{v \in V} w_v^c \left[\sum_{a \in A} m_a k_{v,a} + I(v) \right] + w_0^c \trianglelefteq 0 \quad \forall c \in G_n \quad (17)$$

$$I(v) + \sum_{a \in A} k_{a,v} m_a \leq ub_v \quad \forall v \in V_n, \text{ s.t. } ub_v \text{ exists} \quad (18)$$

$$I(v) + \sum_{a \in A} k_{a,v} m_a \geq lb_v \quad \forall v \in V_n, \text{ s.t. } lb_v \text{ exists} \quad (19)$$

The state equation constraints are not compatible with the dominated action constraint (14), because dominated actions are identified considering the interval relaxation and they do not consider negative effects, which are accounted for in constraints (16)-(19).

6 LP and IP-based Models as Heuristics

The IP model defined in the previous sections can be used to calculate admissible heuristic values for heuristic search algorithms. In every expanded state s, fact and action landmarks are extracted, and relevant and dominated actions are identified by relevance and dominance analyses, respectively. An IP or LP model is then solved for the task defined by $\Pi'^{+} = \langle V_p, V_n'^{+}, A^{+}, s, G \rangle$.

We observe that the constraint set defined by (1)-(19) can be categorized as operator counting constraints (Pommerening et al. 2015), as the only common variables are m_a.

We define different heuristics by taking various constraint sets. In our analysis, we consider the combinations of constraints reported in Table 1. It should be noted that all the models use constraint (5′) instead of (5). While omitted in Table 1 for clarity, elsewhere we use subscripts h_{IP} or h_{LP} to indicate if we solve the IP model or its LP relaxation.

Constraints	(1)-(7)	(8)-(9)	(10)-(11)	(12)-(13), (15)	(14)	(16)-(19)
h^{ir}	✓	✓				
h^{l}	✓	✓	✓			
h^{e}	✓	✓	✓	✓	✓	
h^{c}	✓	✓	✓	✓		✓
$h^{c,tr}$	✓		✓	✓		✓

Table 1: Constraints used in different heuristic variations.

7 Empirical Evaluation

In this section we empirically evaluate the performance of our heuristics. In particular, we present: (i) the performance difference between heuristics derived by our IP model and their relaxed LP versions; (ii) the impact of valid and strengthening constraints; and (iii) a comparison against the state-of-the-art cost-optimal numeric planners.

7.1 Experimental Setup

We consider the set of eight benchmark domains used by Scala et al. (2017):

- **Counters:** a set of integer variables $X_1, ..., X_n$ can be increased or decreased by actions. The goal is to achieve $X_i < X_{i+1}$ for $i \in \{1, ..., n\}$ variables (Francès and Geffner 2015);
- **Gardening:** an agent needs to water plants located on a grid. To do so, the agent needs to reach a tap and carry water, subject to a capacity constraint (Francès and Geffner 2015);
- **Sailing:** boats can navigate in an unbounded grid space to rescue people. In order for a boat to rescue a person, it has to reach a region defined by a set of inequalities (Scala, Haslum, and Thiébaux 2016);
- **Farmland:** people can be transported between farms and need to be allocated to given locations (Scala, Haslum, and Thiébaux 2016);
- **Rover, Satellite, Depots, Zeno Travel:** these IPC domains are characterized by resources (energy, fuel, loads) that are produced and consumed, and actions are constrained by these quantities. The Zeno Travel domain has an action (`refuel`) whose effect is not a constant increase or decrease of the numeric state variable representing `fuel`, rather it is an assignment of the variable to a maximum value. This domain does not fall in the *simple numeric condition* case, since action `refuel` is the only achiever of all the numeric conditions, but the heuristics proposed by Scala, Haslum, and Thiébaux (2016) and Scala et al. (2017) can be adapted to consider this case, preserving admissibility. Although our model can also be adapted, we chose to modify the effect of action `refuel` to be a constant increase of the numeric state variable `fuel`, instead.

Previous works in cost-optimal numeric planning experimented only with problems with unary action costs. Since the original IPC domains Rover, Satellite and Depots have non-uniform action costs, for them, we present results for both variants.

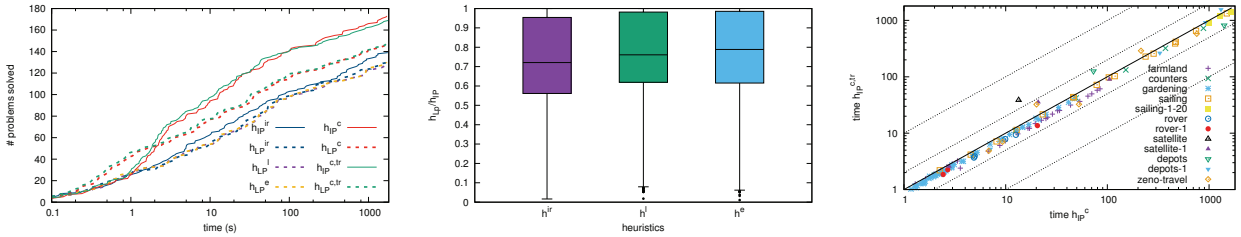
We implement the heuristic evaluation in the planner LPRPG (Coles et al. 2008), modified to handle cost-optimal planning by using an A* algorithm, where ties are broken preferring states with lower heuristic value. The LP and IP models are solved using CPLEX v12.7. We run all the experiments on a Xeon 3.5GHz processor machine running OS Sierra. Every problem is limited to 4 GB and 30 minutes.

7.2 Evaluation of Heuristics

The following analysis considers the set of heuristics shown in Table 1, considering both their IP and LP variants.

Figure 1a shows the cumulative number of problem solved as a function of the execution time, for each heuristic variation. The figure omits the results for h^{l}_{IP} and h^{e}_{IP}, which give the same heuristic values as h^{ir}_{IP}. As shown in the figure, the IP heuristics give a better overall coverage than their respective LP version.

Adding landmark, relevance, dominance and inverse actions constraints slightly improves the coverage of the LP-based heuristics, as their value is closer to the IP optimal one. This is also highlighted in Figure 1b, that shows a box plot of the mean ratio of the heuristic values of every state expanded using h_{IP} and the respective h_{LP} for every problem. The box plot shows that dominance and relevance constraints marginally reduce the integrality gap when coupled with landmark constraints. The difference between IP and



(a) Cumulative number of problem solved as function of execution time.

(b) Boxplot h_{LP}/h_{IP} for different heuristic versions.

(c) Runtime comparison of $h_{IP}^{c,tr}$ and h_{IP}^{c} on the benchmark problems.

Figure 1: Experimental Evaluation of h_{IP} and h_{LP}.

Domain	#	coverage							time						# states expanded					
		$\hat{h}^{rmax}$	h^{lma+}	h_{IP}^{lma+}	h_{IP}^{c}	$h_{IP}^{c,tr}$	$h_{LP}^{c,tr}$	VBS	$\hat{h}^{rmax}$	h^{lma+}	h_{IP}^{lma+}	h_{IP}^{c}	$h_{IP}^{c,tr}$	$h_{LP}^{c,tr}$	$\hat{h}^{rmax}$	h^{lma+}	h_{IP}^{lma+}	h_{IP}^{c}	$h_{IP}^{c,tr}$	$h_{LP}^{c,tr}$
Counters	15	6	7	6	12	**13**	**13**	13	2.5	2.3	15.0	0.3	0.3	**0.2**	12664.7	7052.2	7052.2	**10.3**	**10.3**	**10.3**
Gardening	63	**63**	**63**	**63**	**63**	**63**	**63**	63	**3.5**	4.8	51.7	6.1	5.3	52.8	34442.3	19845.7	19845.7	**178.5**	**178.5**	6693.8
Sailing	25	14	5	5	**22**	**22**	5	22	**1.3**	2.6	10.8	32.1	29.7	44.9	12549.0	3431.4	**1252.4**	**1252.4**	**1252.4**	3432.6
Sailing (1-20)	20	12	**20**	19	12	12	9	20	**5.2**	8.5	39.1	169.0	152.2	254.6	78409.1	8855.8	**3671.0**	3974.2	3974.2	8855.8
Farmland	30	**30**	**30**	28	**30**	**30**	**30**	30	**7.8**	19.1	238.4	18.3	13.4	225.0	27929.2	27949.3	14319.8	**417.7**	**417.7**	7252.8
Rover	20	4	**5**	3	**5**	4	4	6	15.6	9.6	41.7	7.5	5.6	**4.3**	40430.3	6175.0	6175.0	51.7	51.7	**48.3**
Depots	20	2	**4**	2	3	3	**4**	5	**1.1**	3.3	5.5	37.0	63.4	49.6	1957.5	752.0	752.0	**46.5**	482.5	475.0
Satellite	20	1	1	1	**3**	1	1	3	**0.7**	2.0	6.5	13.4	38.7	35.3	1525.0	1965.0	1965.0	**126.0**	731.0	676.0
Zeno Travel	20	6	7	6	**8**	**8**	7	8	17.7	30.7	86.9	14.9	**12.9**	17.2	35595.0	18414.7	18638.2	**79.8**	87.3	148.3
Rover-1	20	4	4	4	**7**	6	4	8	1.4	**1.2**	3.2	1.8	1.4	2.2	3078.2	296.0	296.0	**10.2**	**10.2**	18.0
Depots-1	20	2	4	3	**5**	**5**	4	5	**1.1**	3.2	5.4	3.9	2.9	4.2	1957.5	752.0	752.0	**17.0**	**17.0**	36.0
Satellite-1	20	2	**3**	**3**	**3**	**3**	**3**	3	75.6	**1.7**	4.1	11.8	19.1	29.2	387929.5	522.5	522.5	**35.5**	73.0	101.5
Total	293	146	153	143	**173**	170	147	186	**6.3**	9.0	84.9	21.5	19.4	95.2	37623.5	17620.2	14277.9	**504.1**	516.3	5532.2

Table 2: Coverage, average execution time and average number of expanded nodes by domain. With "-1" we identify the domain variants with unary action costs. VBS indicates the number of problems that are solved by at least one heuristic.

LP heuristics varies with the domain. For Counters we observe that the LP relaxation produces the same values of the IP solution, while solving it in significantly less time. Sailing shows the opposite behavior: all the numeric state variables are unbounded and the presence of the big-M constraints makes the LP relaxation very poor.

Not surprisingly, adding state equation constraints leads to a more informative heuristic value, which is also reflected in the total number of problems solved. Similar to classical planning (Imai and Fukunaga 2015), sequencing constraints (8)-(9) make the problems slower to solve without resulting in a much more accurate heuristic value for most of the domains (Figure 1c). The graph shows that $h_{IP}^{c,tr}$ finds solutions faster than h_{IP}^{c} in most of the instances. However, there are some instances where $h_{IP}^{c,tr}$ performs worse than h_{IP}^{c}.

Overall, the heuristic with best coverage is h_{IP}^{c}, which obtains a total coverage of 173 problems solved (Table 2).

7.3 Comparison with State of the Art

We now compare the coverage of our heuristics with the state of the art in cost-optimal numeric planning. The two other admissible heuristics for the fragment of numeric planning that we consider are $\hat{h}^{rmax}$ (Scala, Haslum, and Thiébaux 2016) and h^{lma+} (Scala et al. 2017). In addition we consider a variation of the original h^{lma+}, where the cost-partitioning problem is solved as IP, h_{IP}^{lma+}.

Table 2 reports the total coverage, the average execution time, and the average number of nodes expanded by domain for $\hat{h}^{rmax}$, h^{lma+}, h_{IP}^{lma+}, our two our best IP heuristics, h_{IP}^{c} and $h_{IP}^{c,tr}$, and our best LP heuristic $h_{LP}^{c,tr}$.

From the table, we can observe that for the IPC domains, the coverage is much lower than Counters, Gardening, Farmland and Sailing. This is caused by the propositional structure of these domains, for which adding the integrality constraints results in an excessive time overhead, that is not compensated by increased informativeness. The number of states expanded by $h_{LP}^{c,tr}$ and h_{IP}^{c} are similar for the IPC domains, while the difference is greater for Gardening, Farmland and Sailing. The major benefit of integrality constraints is gained when the the big-M constraints are more influential. This is also reflected in h^{lma+}, that does not use big-M constraints and, consequently, the IP model does not improve on the LP version.

The table shows that h_{IP}^{c} and $h_{IP}^{c,tr}$ are much more informative than and outperform any other heuristics. This is highlighted by the average number of states expanded by h_{IP}^{c} and $h_{IP}^{c,tr}$, which are almost two orders of magnitude fewer than the existing heuristics. Our best LP heuristic $h_{LP}^{c,tr}$ is also competitive with $\hat{h}^{rmax}$ and h^{lma+}.

8 Conclusion

In this paper, we consider cost-optimal numeric planning with instantaneous actions, where numeric state variables are subject to linear constraints and are modified by constant increases and decreases. Given its success in classical planning, we use mathematical programming to formulate admissible heuristics, taking advantage of its ability to solve problems with linear constraints. In particular, we extend the delete relaxation heuristic model (Imai and Fukunaga 2015) to the interval relaxation and subsequently add

landmark constraints and constraints implied by relevance and dominance analyses. We consider additional constraints inspired by state equation constraints. Our empirical evaluation shows that, for our models, the gain in informativeness of IP-based heuristics often compensates for the computational cost. The interval relaxation heuristic with state equation constraints establishes a new state of the art in cost-optimal numeric planning.

We intend to further investigate the trade-off between informativeness and speed of IP and LP based heuristics, in order to identify the conditions, especially with respect to the size of the problems, that make one favored over the other and to determine if there is a middle ground between the two. The extension of our heuristic to include other operator counting constraints is also subject to future work, as well as the generalization of the presented models to numeric planning with state-dependent effects.

Acknowledgements

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

References

Aldinger, J.; Mattmüller, R.; and Göbelbecker, M. 2015. Complexity of Interval Relaxed Numeric Planning. In *Proceedings of Annual German Conference on AI 2015*, 19–31.

Bonet, B. 2013. An Admissible Heuristic for SAS+ Planning Obtained from the State Equation. In *Proceedings of International Joint Conference on Artificial Intelligence 2013*, 2268–2274.

Bylander, T. 1997. A Linear Programming Heuristic for Optimal Planning. In *Proceedings of AAAI Conference on Artficial Intelligence 97*.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. A hybrid Relaxed Planning Graph-LP Heuristic for Numeric Planning Domains. In *Proceedings of International Conference on Automated Planning and Scheduling 2008*, 52–59.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2013. A Hybrid LP-RPG Heuristic for Modelling Numeric Resource Flows in Planning. *Journal of Artificial Intelligence Research* 46:343–412.

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

Francès, G., and Geffner, H. 2015. Modeling and Computation in Planning: Better Heuristics from More Expressive Languages. In *Proceedings of International Conference on Automated Planning and Scheduling 2015*, 70–78.

Geffner, H., and Haslum, P. 2000. Admissible Heuristics for Optimal Planning. In *Proceedings of International Conference on Automated Planning and Scheduling 2000*, 140–149.

Helmert, M. 2002. Decidability and Undecidability Results for Planning with Numerical State Variables. In *Proceedings of International Conference on Artificial Intelligence Planning and Scheduling 2002*, 44–53.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *Journal of Artificial Intelligence Research* 20:291–341.

Illanes, L., and McIlraith, S. A. 2017. Numeric Planning via Abstraction and Policy Guided Search. In *Proceedings of International Joint Conference on Artificial Intelligence 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. *Journal of Artificial Intelligence Research* 54:631–677.

Karpas, E., and Domshlak, C. 2009. Cost-optimal Planning with Landmarks. In *Proceedings of International Joint Conference on Artificial Intelligence 2009*, 1728–1733.

Katz, M., and Domshlak, C. 2010. Optimal Admissible Composition of Abstraction Heuristics. *Artificial Intelligence* 174(12-13):767–798.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and Complete Landmarks for AND/OR Graphs. In *Proceedings of European Conference on Artificial Intelligence 2010*, 335–340.

Nakhost, H.; Hoffmann, J.; and Müller, M. 2012. Resource-Constrained Planning : A Monte Carlo Random Walk Approach. In *Proceedings of International Conference on Automated Planning and Scheduling 2012*, 181–189.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2015. Heuristics for Cost-Optimal Classical Planning Based on Linear Programming. In *Proceedings of International Joint Conference on Artificial Intelligence 2015*, 4303–4309.

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the Most Out of Pattern Databases for Classical Planning. In *Proceedings of International Joint Conference on Artificial Intelligence 2013*, 2357–2364.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2016. Interval-Based Relaxation for General Numeric Planning. In *Proceedings of European Conference on Artificial Intelligence 2016*, 655–663.

Scala, E.; Haslum, P.; Magazzeni, D.; and Thiebaux, S. 2017. Landmarks for Numeric Planning Problems. In *Proceedings of International Joint Conference on Artificial Intelligence 2017*, 4384–4390.

Scala, E.; Haslum, P.; and Thiébaux, S. 2016. Heuristics for Numeric Planning via Subgoaling. In *Proceedings of International Joint Conference on Artificial Intelligence 2016*, 3228–3234.

van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An LP-based Heuristic for Optimal Planning. In *Proceedings of International Conference on Principles and Practice of Constraint Programming 2007*, 651–665.

Zhu, L., and Givan, R. 2003. Landmark Extraction via Planning Graph Propagation. In *Doctoral Consortium of International Conference on Automated Planning and Scheduling 2003*, 156–160.