Heuristics for Numeric Planning via Subgoaling

Enrico Scala,[1] Patrik Haslum,[1,2] Sylvie Thiébaux,[1,2]
[1]Research School of Computer Science, The Australian National University
[2]NICTA
Canberra, ACT, Australia
firstname.lastname@anu.edu.au

Abstract

The paper presents a new relaxation for hybrid planning with continuous numeric and propositional state variables based on subgoaling, generalising the subgoaling principle underlying the h^{max} and h^{add} heuristics to such problems. Our relaxation improves on existing interval-based relaxations by taking into account some negative interactions between effects when achieving a subgoal, resulting in better estimates. We show conditions on the planning model ensuring that this new relaxation leads to tractable, and, for the h^{max} version, admissible, heuristics. The new relaxation can be combined with the interval-based relaxation, to derive heuristics applicable to general numeric planning, while still providing more informed estimates for the subgoals that meet these conditions. Experiments show the effectiveness of its inadmissible and admissible version on satisficing and optimal numeric planning, respectively. As far as we know, this is the first admissible heuristic enabling cost-optimal numeric planning.

Introduction

A lesson from classical planning is that one source of complexity is the requirement to satisfy conditions simultaneously. Relaxing this requirement, that is, assuming that subgoals can be (recursively) considered in isolation, is the basic idea of both the inadmissible h^{add} and the admissible h^{max} heuristics (Bonet and Geffner 2001; Haslum and Geffner 2000). The objective of this paper is to bring this perspective to the design of new domain-independent heuristics for hybrid planning with propositional and continuous numeric state variables.

Planning with integrated continuous and discrete representations is crucial to applying planning to many real world problems, where interactions between qualitative and quantitative constraints cannot be ignored (Dornhege et al. 2009). The numeric extension to classical planning made in PDDL 2.1 (Fox and Long 2003) provides a unified action model, where qualitative conditions are modelled with propositional variables and quantitative conditions are modelled with expressions over numeric state variables.

Heuristic search, using domain-independent heuristics automatically obtained from relaxations of the planning model, is a widely used and often effective approach to planning. Most heuristics for numeric planning are based on extending the delete-free relaxation to the numeric case, yielding the so called interval-based relaxation (Hoffmann 2003; Gerevini, Saetti, and Serina 2008; Coles et al. 2010; Aldinger, Mattmüller, and Göbelbecker 2015). While this yields general and tractable heuristics, it fails to provide effective guidance for some simple yet important cases. Previous work on improving this heuristic has targeted a different weakness, but still reasoning about reachable values, for the restricted class of numeric variables modelling resources (Coles et al. 2013). Eyerich et al. (2009) extended the context-enhanced additive heuristic to numeric planning; however, they consider the numeric part only qualitatively. To the best of our knowledge, no previous work has looked at numeric planning relaxations from the subgoaling decomposition viewpoint.

Our main contribution is a generalisation of the subgoaling principle to hybrid propositional and numeric planning problems. This enables using different relaxations for different *types* of subgoals/conditions: we distinguish between propositional, simple numeric and hard numeric conditions. For simple numeric conditions our heuristics are tractable, and we are also able to formulate a version that is admissible. Combining this with the interval-based relaxation, we obtain a (inadmissible) heuristic that applies to general numeric planning, but benefits from more accurate estimates of the cost of achieving simple subgoals. The only remaining restriction, inherited from the interval-based relaxation, is that the dependency relation over numeric action effects is acyclic (Aldinger, Mattmüller, and Göbelbecker 2015). As a spin-off of this characterisation, we also show how to generate *redundant constraints* to mitigate the harmful effects of considering subgoals separately. Experiments evaluate two inadmissible and one admissible heuristic, on a variety of numeric domains.

Motivating Example

To illustrate the limits of interval relaxation-based heuristics, let us take as an example the following sailing domain.

There is a sailing boat whose task is to rescue people in an unbounded area of the ocean. The positions of the boat and people to be rescued are described by their coordinates $(x, y) \in \mathbb{Q}^2$. To save a person, the boat must reach an area described by one or more linear inequalities. A (sequential) numeric planning formulation of the problem encodes each of the 7 possible movements of the boat (see Figure 1; the boat cannot sail straight into the wind) as a translation in

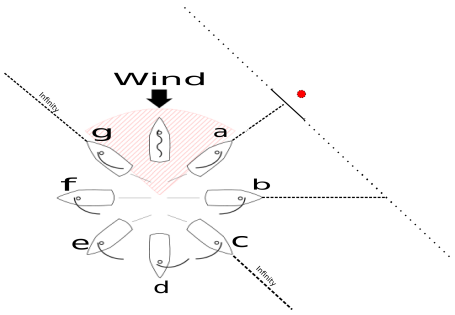


Figure 1: Points of sail (https://en.wikipedia.org); red dot depicts the goal, from action *a* to *g* the 7 possible movements.

the geometrical space defined by $\mathbb{Q}^2$. Speed is implicit: each action changes each of the two variables by a given amount.

In Figure 1, the rescue requires satisfying the inequality $x + y > 10$ starting from position $\langle x = 0, y = 0\rangle$. In the interval based relaxation, all actions that contribute positively to *at least* to one of the variables are considered *useful* for satisfying this condition. This includes the two actions **a** and **b**, but also the misleading actions **c** and **g**. The latter two actions increase either x or y but decrease the other, and thus fail to bring the state any closer to satisfying $x + y > 10$. Ignoring this internal negative effect inevitably leads to searching large parts of the space that are not relevant (i.e, neither **c** or **g** are useful to reach the goal).

This problem is representative of a variety of domains featuring multi-variable numeric conditions. Our approach contrasts with the value reachability view, by performing a relevance analysis for the subgoal at hand. Analogously to the propositional case, this analysis focuses on determining why certain actions are required, but also additionally here in the numeric case, how many of them are necessary.

Notation and Background Material

This paper focuses on sequential numeric planning with grounded actions; see PDDL 2.1 level 2 (Fox and Long 2003) for a deeper understanding of the semantics.

A state of the system is defined as a total assignment of propositional X^p and continuous numeric variables X^n. A propositional condition is a positive literal, while a numeric condition is a tuple $\langle \xi, \trianglerighteq, 0\rangle$ where ξ is an arithmetical expression and $\trianglerighteq \in \{>, \geq, =\}$ a relational operator. We write $[x]^s$ and $[\xi]^s$ for the value of variable x and expression ξ in s. With slight abuse of notation we write $x \in \xi$ to mean that variable x appears in expression ξ.

Let C be a set of (propositional and numeric) conditions. $s \models C$ denotes that state s satisfies all conditions in C.

Definition 1 (Numeric Action)**.** *An action is a pair* $\langle \mathrm{pre}(a), \mathrm{eff}(a)\rangle$*, where* $\mathrm{pre}(a)$ *is a set of propositional and numeric conditions and* $\mathrm{eff}(a)$ *is a set of assignments. A classic assignment is* $p = \top$ *or* $p = \bot$*; a numeric assignment is* $x = \xi$*, where* ξ *is an arithmetical expression over variables in* X^n*. We require that* $\mathrm{eff}(a)$ *does not contain multiple assignments to the same variable.*

We use subscripts to distinguish the propositional and numeric parts (e.g., $\mathrm{eff}_{\mathrm{num}}(a)$ is the set of numeric effects of a). Moreover, let e be an effect of a, then $\mathrm{lhs}(e)$ and $\mathrm{rhs}(e)$ denote the left- and right-hand sides of the assignment. $\mathrm{lhs}(e)$ is the variable affected by e, while $\mathrm{rhs}(e)$ is, in the numeric case, an expression, or, in the classical case, either $\top$ or $\bot$.

An action a is applicable in s iff $s \models \mathrm{pre}(a)$, and its execution results in the state $s' = s[a]$ such that $\forall x \in X$:
– $[x]^{s'} = [\mathrm{rhs}(e)]^s$ if $\exists e \in \mathrm{eff}(a) : \mathrm{lhs}(e) = x$
– $[x]^{s'} = [x]^s$ otherwise (Frame Axiom)
Increase/decrease effects can be formulated as assignments of the form $x = x \pm e$, where e is an arithmetical expression.

Definition 2 (Numeric Planning Problem)**.** *A numeric planning problem is the tuple* $\Pi = \langle s_0, A, G, X, \gamma\rangle$*, where* $X = X^p \cup X^n$ *is the set of variables,* s_0 *is a state over X, A is a set of actions, and G is a set of propositional and numeric conditions.* $\gamma : A \to \mathbb{Q}^{\geq \nvdash}$ *is a function assigning a non-negative, rational cost to each action.*

Definition 3 (Plan)**.** *A plan* π *for* $\Pi = \langle s_0, A, G, X, \gamma\rangle$*is a finite sequence of actions* $a_0, ..., a_{n-1}$ *from* A *such that every action in* π *is applicable in the state resulting from the application of its predecessors:* $s_0 \models pre(a_0)$*,* $s_0[a_0] \models pre(a_1)$*, etc, and* $s_{|\pi|} = s_0[a_0, .., a_{n-1}] \models G$*. The cost of plan* π *is cost*$(\pi) = \sum_{a \in \pi} \gamma(a)$*.*

As usual, the minimum cost of a plan achieving condition c from state s is denoted $h^\star(s, c)$.

Numeric Planning via Interval-Based Relaxation

Pioneered by Hoffman (2003), the interval-based relaxation (Aldinger, Mattmüller, and Göbelbecker 2015; Gregory et al. 2012) is the principle most used to do numeric reasoning in planning. In different flavors (e.g., building metric extension of the relaxed planning graph), the majority of numeric planning systems exploit this theoretical framework to devise heuristics (Koehler 1998; Hoffmann 2003; Gerevini, Saetti, and Serina 2008; Coles et al. 2012; 2013).

The interval-based relaxation approximates each numeric state variable using an interval that contains its *possible* values. The interval of an expression is calculated by Interval Analysis (Moore, Kearfott, and Cloud 2009), and a numeric condition is satisfied if at least one value in the expression's interval satisfies it; applicable action effects monotonically widen the bounds of the intervals, so a *relaxed* solution is computed by extending intervals until the goal is reached. Aldinger et al. (2015) defined a numeric dependency relation over actions and showed that termination and safe pruning can be ensured when the planning task is acyclic w.r.t. this dependency relation. For acyclic tasks, the set of reachable states (denoted by the larger intervals for each variable) can be computed by executing applicable actions as many times as necessary, in the order defined by the dependency relation. Given $\Pi = \langle s_0, A, G, X, \gamma\rangle$, $\Pi^+ = \langle s_0^+, A^+, G, X\rangle$ denotes the interval-based relaxation of Π (Aldinger, Mattmüller, and Göbelbecker 2015).

Subgoaling Relaxation in the Propositional Case

The idea underlying the classical h^{max} heuristic (Bonet and Geffner 2001; Haslum and Geffner 2000) is to estimate the

state–goal distance of atomic subgoals independently. h^{max} is defined as the point-wise maximal fixpoint of the equation

$$h^{max}(s,C) = \begin{cases} 0 & \text{if } s \models C \\ \min\limits_{a \in ach(C)} (h^{max}(s, \mathrm{pre}(a)) + \gamma(a)) & |C| = 1 \\ \max\limits_{C' \subset C : |C'|=1} h^{max}(s,c') & |C| > 1 \end{cases}$$

where $ach(C)$ denotes the set of achievers for C. The maximal fixpoint is unique (Haslum 2009). h^{max} is admissible; an inadmissible, but usually more informative, heuristic h^{add} is obtained by summing, instead of maxing, over the estimated costs of subgoals in a non-singleton set.

Numeric Regression

Regression can be extended to numeric planning by building the so called weakest precondition (Scala 2013). As in Hoare's (1969) calculus, the weakest precondition is computed by substitution ('[]') of the affected variables:

Definition 4 (Effect Regressor)**.** *Given a numeric condition $c : \langle \xi, \trianglerighteq, 0 \rangle$ and an action a, the effect regressor $c^{r(a)}$ transforms* c *into $\langle \xi[x_1/\tau(a,x_1), \ldots, x_k/\tau(a,x_k)], \trianglerighteq, 0 \rangle$ where*

$$\tau(a,x_i) = \begin{cases} \mathrm{rhs}(e) & \text{if } \exists e \in \mathrm{eff}_{\mathrm{num}}(a) : \mathrm{lhs}(e) = x_i \\ x_i & \text{Otherwise} \end{cases}$$

and $x_1, \ldots, x_k$ are the numeric variables assigned by a.

$c^{r(a)}$ is the necessary and sufficient condition for c to hold after applying the effects of a. Numeric regression is defined by additionally incorporating the action precondition:

Proposition 1 (Numeric Regression)**.** *Let a be an action and c a numeric condition. $s[a] \models c$ iff $s \models \{c^{r(a)}\} \cup \mathrm{pre}(a)$. $\{c^{r(a)}\} \cup \mathrm{pre}(a)$ denotes the regression of c through a.*

Regression and Action Repetition

Unlike in the propositional case, numeric actions are not idempotent operations, since the repeated action application in a state does not generally retain its initial effect – the one obtained after its first application. Dually, also the numeric regression operation, and in particular the *effect regressor* is not idempotent: $s \not\models c^{r(a)} \not\Rightarrow s \not\models c^{r(a)^{r(a)}} \ldots \not\Rightarrow s \not\models c^{\overbrace{r(a)^{\cdots^{r(a)}}}^{\text{m times}}}$. While there could be idempotent numeric actions (e.g., assigning a specific value for a variable), this does not apply in general. Several real world problems feature the accumulation of resources or translation in geometrical spaces (e.g., the sailing problem). While a state of the system might not be consistent with a condition arising from a single step of regression, it is possible that it will eventually satisfy the condition if a given number of action *repetitions* is allowed to be executed just before; or dually, if the repeated regression is satisfied.

Some restrictions on the form of action effects allows us to formulate repeated regression in closed form.

Definition 5 (Self-interfering effects)**.** *An action a is said to have* self-interfering *effects whenever $\exists\ e,\ e' \in \mathrm{eff}_{\mathrm{num}}(a)$: $e \neq e'$ and $\mathrm{lhs}(e)$ occurs in $\mathrm{rhs}(e)$.*

Definition 6 (Linear effects action)**.** *An action has linear effects if $\forall e \in \mathrm{eff}_{\mathrm{num}}(a)$, e can be written as $x = \sum_{y \in X^n} w_{y,a,x} y + k_{x,a}$, with $w_{y,a,x}, k_{x,a} \in \mathbb{Q}$, where x is the variable modified by e.*

An action is said to be *LSF* (Linear and Self-interference Free) if it complies with Def. 6 and does not have self-interfering effects. The closed form is as follows:

Theorem 2 (*m-times effect regressor*)**.** *Given a linear numeric condition $c \equiv \sum_{x \in X^n} w_{x,c} x + k_c \trianglerighteq 0$, where $w_{x,c}, k_c \in \mathbb{Q}$, an LSF action a, and a state s, it follows:*

$$s \models c^{\overbrace{r(a)^{\cdots^{r(a)}}}^{\text{m times}}} \Leftrightarrow s \models c^{r(a,m)}$$

where $c^{r(a,m)} = \left(\sum_{x \in X^n} w_{x,c}(f_{x,a}(m)) + k_c\right) \trianglerighteq 0$

$$f_{x,a}(m) = \sum_{i=0}^{m-1} w_{x,a,x}^i \left(\sum_{y \neq x, y \in X^n} w_{y,a,x} y + k_{x,a} \right) + w_{x,a,x}^m x$$

Proof sketch. By induction on m. □

$f_{x,a}(m)$ is the value of x after applying action a m times in the current state (which is implicitly an argument). Note that m appears in the exponent of the last term; thus, the function is non-linear if $w_{x,a,e}$ is not 0 or 1.

Proposition 3 (Sufficient and Necessary Condition)**.** *A numeric condition c is reachable using an LSF action* a *from state s only if there exists an $m \in \mathbb{N}^+$ such that $s \models c^{r(a,m)}$ and $s \models \mathrm{pre}(a)$ (necessary part). The condition is also sufficient if the action does not assign any variable occurring in its precondition.*

Proof sketch. Direct consequence of Theorem 2, and Proposition 1. $s \models c^{r(a,m)}$ effectively captures *m* steps of regression, so the condition can be effectively reached using action *a*, repeating it *m* times. □

Definition 7 (Possible Achiever)**.** *An LSF action a is said to be a (possible)* achiever *of a numeric condition c from s whenever there exists an $m \in \mathbb{N}^+$ such that $s \models c^{r(a,m)}$.*

From Theorem 2 follows that a is a possible achiever of $c \equiv \sum_{x \in X^n} w_{x,c} x + k_c \trianglerighteq 0$ iff there exists m such that

$$\sum_{x \in X^n} w_{x,c} f_{x,a}(m) + k_c \trianglerighteq 0 \qquad (1)$$

In the following, we use $ach(c)$ to denote the set of *possible achievers* for c.

Subgoaling in the Numeric Case

Extending the subgoaling relaxation to both propositional (PCs) and numeric conditions (NCs) is easy from a declarative perspective, since different regression mechanisms can be recursively interleaved depending on the condition's type (PC or NC):

$$h^{max}_{hbd}(s,C) = \begin{cases} 0 & \text{if } s \models C \\ \min_{a \in ach(C)} (h^{max}_{hbd}(s, \text{pre}(a)) + \gamma(a)) & C \in \text{PCs} \\ \min_{a \in ach(C)} (h^{max}_{hbd}(s, \text{pre}(a) \cup \{c^{r(a)}\}) + \gamma(a)) & C \in \text{NCs} \\ \max_{C' \subset C : |C'|=1} h^{max}_{hbd}(s, C') & |C| > 1 \end{cases}$$

The hbd subscript indicates the new *HyBriD* formulation. By substituting sum for max, we obtain an inadmissible variant h^{add}_{hbd}. As in the propositional case, h^{max}_{hbd} and h^{add}_{hbd} are defined as any point-wise maximal fixpoint of these equations. Here, however, we do not have a proof that this fixpoint is unique. On the other hand, uniqueness is not essential: all properties of the heuristics that we establish in the following hold for any maximal fixpoint.

Two practical problems arise with this formulation: First, deciding if an action is a possible achiever ($a \in ach(c)$) is not easy because (i) it is state dependent, and (ii) it may require solving of a non-linear optimization problem, even if we limit ourselves to *LSF* actions (Eq. 1 depends on m, which occurs in the exponent in $f_{x,a}(m)$). Second, regression can generate an unbounded number of different numeric subgoals. For instance, in the motivating example, regressing the goal of crossing a line generates an infinite set of areas from which that goal can be reached. This is not surprising since, in general, even a single numeric condition planning problem can be semi-decidable (Helmert 2002). Thus, the regression tree is infinite, and the minimization done in the heuristic is not guaranteed to terminate. The next section studies sufficient conditions that make variations of this formulation tractable.

The Simple Numeric Condition Case

We define simple numeric conditions as those only affected by actions having constant increase and decrease effects:

Definition 8 (Simple Numeric Condition)**.** *Let c: $\langle \xi, \rhd, 0 \rangle$. c is said to be a* simple numeric condition *(SC) if: (i) $\forall e \in \{e' | e' \in \text{eff}_{\text{num}}(a), a \in A, \xi \cap \text{lhs}(e) \neq \emptyset\}$, e is of the form $x = x + k_{x,a}$ with $k_{x,a}$ a constant in $\mathbb{Q}$; (ii) $\rhd \in \{>, \geq\}$, i.e., c is not an equality condition*[1]*; and (iii) ξ is linear.*

Proposition 4. *If all the numeric conditions in $\Pi = \langle s_0, A, G, X, \gamma \rangle$ are SCs:*

a. reachability of numeric conditions can be computed in closed form using the m-times effect regressor *(Th. 2);*

b. the m-times effect regressor *is commutative, that is $c^{r(a,n)^{r(b,m)}} \equiv c^{r(b,m)^{r(a,n)}}$ for any SC c, actions a, b and $n, m \in \mathbb{N}$;*

c. the effect of each action on each condition is state-independent.

Proof sketch. (a) is a consequence of Theorem 2. For SCs, $f_{x,a}(m)$ simplifies to $mk_{x,a} + x$. Hence (c), the change in x caused by a is an additive constant. This implies (b). □

An immediate consequence of Prop. 4 is that the number of relevant numeric conditions that can be generated by h^{max}_{hbd} (h^{add}_{hbd}) is finite. Relevant numeric conditions are those generated by regressing through possible achievers only, and the simple closed form $f_{x,a} = mk_{x,a} + x$ enables us to find the possible achievers efficiently.

Theorem 5 (Termination with SCs)**.** *If numeric conditions are regressed only through possible achievers, then computation of h^{max}_{hbd} and h^{add}_{hbd} with only SCs terminates in a finite number of steps.*

Explicit Regression for h^{add}_{hbd}

Although finite, a naive implementation of h_{hbd} is not tractable. Fortunately, the SC properties allow us to minimize cost over just the numbers of action repetitions rather than regressing through all relevant action sequences.

Theorem 6 (Explicit Regression for h^{add}_{hbd})**.**
$$h^{add}_{hbd}(s,c) = \min_{m_1,..,m_{n=|A|} \in \mathbb{N}} \sum_{j=1..n} m_j [h^{add}_{hbd}(\text{pre}(a_j)) + \gamma(a_j)]$$
$$s \models c^{r(a_1,m_1)^{..^{r(a_n,m_n)}}}$$

where c is a single simple numeric condition.

Proof sketch. Like h^{max}_{hbd}, h^{add}_{hbd} is defined by recursively minimizing over the regression tree, but adding instead of maxing $h^{add}_{hbd}(s, \text{pre}(a))$ and $h^{add}_{hbd}(s, c^{r(a)})$. Thus the precondition cost $h^{add}_{hbd}(s, \text{pre}(a))$ is counted for every repetition of a. Because of this, choosing m_i as the number of times action a_i is repeated in the path that minimizes $h^{add}_{hbd}(s,c)$ gives the same cost. □

Because solving the above integer optimization problem is intractable, we replace it with its continuous relaxation[2]. Moreover, to reduce over-estimation we count the precondition cost for each action used in the solution only once, i.e, replace $\sum_{j=1..n} m_j [h^{add}_{hbd}(\text{pre}(a_j)) + \gamma(a_j)]$ with the more optimistic estimate $\sum_{\substack{j=1..n \\ \text{s.t. } m_j > 0}} [h^{add}_{hbd}(\text{pre}(a_j)) + m_j \gamma(a_j)]$.

We denote the resulting heuristic $\hat{h}^{add}_{hbd}$.

An Admissible Estimate: $\hat{h}^{max}_{hbd}$

To get a practical and admissible version of h^{max}_{hbd}, we will go through three main steps. First, observe that a lower bound on the cost of achieving condition c is given by the optimal solution to the following Mixed Integer Problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{a \in A} [\gamma(a) m_a] \\ \text{subject to} \quad & \sum_{x \in X} w_{x,c} [\sum_{a \in A} m_a k_{x,a} + x] + w_{n,c} \rhd 0 \\ & m_a \in \mathbb{N} \ \ \forall a \in A \end{aligned}$$

This problem accounts only for the cost of the actions needed, ignoring their precondition costs. Second, we take the continuous relaxation of the above problem, which is not only tractable but whose optimal solution is also a lower bound on the cost of the optimal solution of the integer version. We can show there always exists an optimal solution to

[1] An equality $\langle \xi, =, 0 \rangle$ can however be expressed as the conjuction of $\{\langle \xi, \geq, 0 \rangle, \langle \xi, \leq, 0 \rangle\}$.

[2] The continuous relaxation of an integer problem is obtained by allowing integer variables to take rational values. This makes it an LP that can be solved in polynomial time.

the continuous relaxation where just one action is used (i.e., $m_a > 0$ for just one $a \in A$). This allows us to solve this problem by evaluating each action independently. Third, we can improve this estimate by considering only reachable actions ($A^+ \subseteq A$) and their precondition costs. However, to maintain admissibility we can only add the smallest precondition cost among the possible achievers of the condition. Combining these steps and letting s and c be a state and a *single* numeric condition, we obtain the following estimate:

$$\hat{h}^{max}_{hbd}(s,c) = \min_{\substack{a\in A^+, \hat{m}\in\mathbb{Q}^{\geq 0} \\ s\models c^{r(a,\hat{m})}}} (\hat{m}\cdot\gamma(a)) + \min_{\substack{a\in A, \\ s\models c^{r(a,\hat{m})}}} \hat{h}^{max}_{hbd}(s,\mathrm{pre}(a))$$

where A^+ is the set of actions with $\hat{h}^{max}_{hbd}(s,\mathrm{pre}(a)) \neq \infty$.

Theorem 7 (Admissibility)**.** *Let c be a SC: $\hat{h}^{max}_{hbd}(s,c) \leq h^\star(s,c)$.*

Theorems 6 and 7 provide tractable ways of computing best achievers, without generating new numeric conditions that do not appear in the problem (either as preconditions or goals). As in the case of propositional h^{max}, this enables us to use a label correcting algorithm (such as the Bellman-Ford algorithm (Bellman 1958)) to compute these heuristics in polynomial time.

Theorem 8 (Tractability)**.** *$\hat{h}^{max}_{hbd}$ and $\hat{h}^{add}_{hbd}$ can be computed in time polynomial in the size of the problem.*

Limitations. Dealing with a larger fragment of numeric planning (e.g., involving only *LSF* actions and linear numeric condition) is possible in principle; however, when action effects depend on some other actions, ignoring indirect effects makes the heuristic inadmissible and even not safe to use for pruning. Take as an example a domain with actions $a : \langle \emptyset, x = y\rangle$ and $b : \langle \emptyset, y = y + 5\rangle$, and a condition $c : \langle x, \geq, 10\rangle$. Assume that the current state is $s_0 : \langle x = 0, y = 0\rangle$. c can be achieved only by the *ordered* sequence of actions b, b, a.

Integration with the Interval Based Relaxation

To handle more general numeric conditions, including indirect effects of actions, the following formulation integrates $\hat{h}^{add}_{hbd}$ with the interval-based relaxation, where numeric conditions are categorized in SCs and HCs (Hard Numeric Conditions):

$$\hat{h}^{add}_{hbd+}(s,C) = \begin{cases} 0 & \text{if } s \models C \\ \min\limits_{a\in ach(C)} (\hat{h}^{add}_{hbd+}(s,\mathrm{pre}(a)) + \gamma(a)) & C \in \mathsf{PCs} \\ \min\limits_{\substack{a\in A, \\ s\models c^{r(a,\hat{m})} \\ \forall \hat{m}\in\mathbb{Q}^{\geq 0}}} (\hat{m}\gamma(a) + \hat{h}^{add}_{hbd+}(s,\mathrm{pre}(a))) & C \in \mathsf{SCs} \\ \sum\limits_{\substack{a\in\pi', \\ \pi'\in sol(<s^+,A^+,G,X>)}} (\hat{h}^{add}_{hbd+}(s,\mathrm{pre}(a)) + \gamma(a)) & C \in \mathsf{HCs} \\ \sum\limits_{c'\subset C:|c'|=1} \hat{h}^{add}_{hbd+}(s,c') & |C| > 1 \end{cases}$$

HCs denotes the set of the numeric conditions that are not simple (Def. 8). The subscript $hbd+$ denotes the support for HCs. Each case (PCs, SCs, HCs) is treated with a specialized reasoning, but just for that particular subgoal. The implementation of the interval-based relaxation is similar to Metric-FF in the spirit, but does not require transforming the task into Linear Normal Form (Hoffmann 2003). Instead it follows a fixpoint analysis using actions reachable for $\hat{h}^{add}_{hbd+}$ (A^+) where termination and safe pruning are guaranteed by the transitive closure of the action effects dependency relation. Fixpoint analysis returns the actions *sufficient* to reach each complex condition, then their preconditions are recursively evaluated using $\hat{h}^{add}_{hbd+}$; if those actions have any simple numeric precondition, its evaluation is done with the more accurate estimate. Thus, the heuristic interleaves both relaxations. There is no extraction of a relaxed plan so the heuristic can be less accurate than Metric-FF-like heuristics (Hoffmann 2003; Coles et al. 2012). An admissible formulation ($\hat{h}^{max}_{hbd+}$) can be obtained by using any admissible approximation of the interval-based relaxation for HCs – the simplest such being to just assign them an estimate of zero – while using the lower bound $\hat{h}^{max}_{hbd}$ for SCs and maxing over subgoals. All the above heuristic variants provide *safe* pruning, giving infinite values only for unsolvable problems.

More Information via Redundant Constraints

The subgoaling relaxation fails to capture negative interactions arising between actions used to achieve parts of a conjunctive condition simultaneously. To alleviate this, we use redundant constraints to tighten the relaxation. This is possible because even simple numeric conditions can express some interaction between variables, which is considered in our heuristic, leading to better accuracy.

We automatically add, to each numeric precondition and goal set, redundant constraints that are implied by the simultaneous satisfaction of each pair of conditions in that set. Let $c_1 : \langle \xi, \geq, 0\rangle$ and $c_2 : \langle \xi', \geq, 0\rangle$ be two SCs: the implied redundant constraint is $c_3 : \langle \xi + \xi', \geq, 0\rangle$.

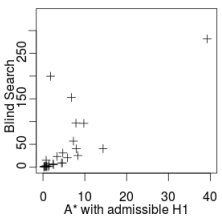
Redundant constraints capture conflicting effects and are known to be a useful technique to prune the search space of invalid partial solutions in areas such as scheduling (Getoor et al. 1997). As an example of their usefulness in planning, consider the goal $\{c_1, c_2\}$ where $c_1 : \langle x - y, \geq, 0\rangle$ and $c_2 : \langle z, \geq, 0\rangle$, an action $a : \langle \emptyset, \{x = x + 1, z = z - 1\}\rangle$ and $s_0 : \langle x = 0, y = 1, z = 0\rangle$. c_1 is achieved by applying action a_1 once; c_2 is already true. The problem is however unsolvable since action a_1 falsifies c_2, and there is no way to recover it. The redundant constraint obtained from c_1 and c_2 is $c_3 : x - y + z \geq 0$. Using the condition of Eq. 1, we can show that action a_1 is not an achiever of c_3. Thus, even though c_1 and c_2 can be achieved, c_3 cannot, making the problem unsolvable also in the relaxed version. Situations like this one may be quite frequent in planning problems. Redundant constraints preserve admissibility and safe pruning, but if used in the inadmissible setting, they may exacerbate the overestimation of the actual distance to goal.

Implementation and Experimental Analysis

We evaluate three planners: two satisficing planners, using the inadmissible additive heuristic $\hat{h}^{add}_{hbd+}$ and its extension with *redundant constraints* $\hat{h}^{radd}_{hbd+}$, and one optimal planner using the admissible heuristic with redundant constraints

		Coverage			Average Time			Plan length			Exp. Nodes		
	I	IBR	$\hat{h}^{add}_{hbd+}$	$\hat{h}^{radd}_{hbd+}$	IBR	$\hat{h}^{add}_{hbd+}$	$\hat{h}^{radd}_{hbd+}$	IBR	$\hat{h}^{add}_{hbd+}$	$\hat{h}^{radd}_{hbd+}$	IBR	$\hat{h}^{add}_{hbd+}$	$\hat{h}^{radd}_{hbd+}$
COUNT	55(10)	17	23	**25**	**0.1**	3.3	2	20.7	13.7	**13**	**21.7**	1811.1	**974.2**
GARD	51(16)	0	16	**26**	NA	21.2	**4.3**	NA	176.7	**123.7**	NA	49393	**5757.7**
SAIL	40(33)	0	**40**	33	NA	**1.9**	2.7	NA	**459.8**	476.3	NA	745.8	**648.4**
FARM	50(50)	0	**50**	**50**	NA	**0.9**	1.8	NA	**400.2**	416	NA	**405.6**	419.2
ZENO	23(6)	6	**21**	19	**0.0**	0.3	0.5	14.7	**13.3**	**13.3**	15.7	15.7	15.7
ROVER	20(2)	2	**7**	**7**	166.3	0.4	0.5	14	**9**	**9**	350	**11**	**11**
SATEL	20(6)	1	**6**	**6**	NA	9.1	**5.2**	NA	20.6	20.6	NA	123.6	123.6
DEPOT	20(3)	1	**3**	**3**	NA	1.5	**1.1**	NA	17.3	17.3	NA	19	19
TPP	40(3)	**6**	3	1	**0.1**	0.5	NA	8	9.7	NA	**9**	14.3	NA

(a) Results of experiments with satisficing planners. Entries are calculated on instances solved by all systems (if appropriate). The size of this set is given in brackets next to the number of instances. Times are given in seconds. The timeout for all planners was $1,800$ seconds.



			Coverage		Exp. Nodes	
	I	P	Blind	A*	Blind	A*
GARD	63	18.6	27	50	$\approx 10^5$	$\approx 10^3$
FARM	30	439	0	20	NA	$\approx 10^3$
SAIL	25	17	1	9	NA	$\approx 10^4$

(b) Results of experiments with optimal planners using Blind Search (*Blind*) and A^* with $\hat{h}^{rmax}_{hbd+}$. Time distribution in GARDENING(left); average expanded nodes (where significant) and coverage for both domains (right). P is the average plan length. Timeout was $1,800$ seconds.

Figure 2: Experimental Results

$\hat{h}^{rmax}_{hbd}$. The latter is used within A*, while all other heuristics are used in a Greedy Best First Search (GBFS). For comparison, we also run GBFS with the interval-based relaxation (IBR) heuristic, obtained from Metric-FF (Hoffmann 2003) by disabling Enforced Hill Climbing. The optimal planner is compared with blind search.

Benchmarks include IPC domains (http://www.icaps-conference.org/), numeric reformulation of the COUNTERS and GARDENING domains by Francés and Geffner (2015), our motivating example (SAILING) and a new domain called FARMLAND. SAILING instances are scaled by the numbers of boats and people to be rescued. The FARMLAND domain models allocating manpower to farms, with a hard constraint on a metric measuring benefit. The contribution of each farm to the benefit is a function of the number of workers assigned. Each farm requires at least one worker. The number of workers ranges from 100 to $1,000$, and the number of farms from 2 to 10. The experimental setting has been conceived to stress numeric reasoning. Most of the domains feature only SCs, except TPP and DEPOT which also have HCs. We expect our heuristics to perform well on domains featuring mostly SCs.

Evaluation Figure 2a reports the performance of the satisficing planners. Both variants of the hybrid heuristic outperform the IBR heuristic in every domain but TPP. IBR is very weak for domains where numeric conditions involve several variables. This is prominent in SAILING and FARMLAND. The h_{hbd^+} heuristics also performs well in classical IPC domains where the propositional part of the problem is important, showing synergy between the numeric and propositional relaxations. $\hat{h}^{radd}_{hbd+}$ is well-informed for problems with many interacting goals, e.g., COUNTERS and GARDENING, producing plans of higher quality. However, the redundant constraints are also misleading in some domains, so they must be added with care.

Figure 2b shows results for the optimal planner. Experiments were run on a smaller set of instances of three domains: GARDENING, with up to 3 plants and $10x10$ grids, FARMLAND, with 4 farms, and SAILING, with smaller metric distance to the goals. As expected, the number of goals is crucial for the effectiveness of the heuristic. In GARDENING, the optimal planner solves all instances with 1 plant, 90.5% with 2 plants and about 50% with 3 plants. Blind search solves 19 instances out of 21 with 1 plant, 8 with 2 plants and none with 3 plants. In FARMLAND, the optimal planner solves 20 out of 30 instances, blind search none at all. In SAILING, performance is poorer because of the large number of subgoals, which leads to more decompositions and lower heuristic accuracy. Average runtime is around 5 seconds in FARMLAND, around 68 seconds in GARDENING, though the largest instances took over $1,000$ seconds, and on average 38 seconds in SAILING.

Conclusion

In the classical setting, considering subgoals in isolation implies a delete-free relaxation; this is not the case in numeric planning, where the analogue of delete relaxation, known as the interval-based relaxation, ignores negative interactions also between numeric variables in a single numeric condition. Our generalisation of the principle underlying the h^{max} and h^{add} heuristics to hybrid planning takes such interactions into account, making it more accurate. For the class of simple numeric conditions, tractable heuristics based on this relaxation can be devised. General numeric conditions can be handled by interleaving it with the interval-based relaxation. For problems with only simple numeric conditions, we formulated an admissible heuristic. As far as we know, it is the first of its kind.

Future work includes lifting the remaining restriction – to acyclic numeric dependencies – posed by the interval-based relaxation, and integrating $\hat{h}^{add}_{hbd+}$ with state-of-the-art search strategies by identifying preferred operators.

Acknowledgements

This work is supported by ARC project DP140104219, "Robust AI Planning for Hybrid Systems". NICTA is funded by the Australian Government through the Department of Com-

munications and the Australian Research Council through the ICT Centre of Excellence Program.

References

Aldinger, J.; Mattmüller, R.; and Göbelbecker, M. 2015. Complexity of interval relaxed numeric planning. In *Proc. of KI 2015: Advances in Artificial Intelligence*, 19–31.

Bellman, R. 1958. On a Routing Problem. *Quarterly of Applied Mathematics* 16:87–90.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129:5–33.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proc. of International Conference on Automated Planning and Scheduling (ICAPS 2010)*.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2012. Colin: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research (JAIR)* 44:1–96.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2013. A hybrid LP-RPG heuristic for modelling numeric resource flows in planning. *Journal of Artifificial Intelligence Research (JAIR)* 46:343–412.

Dornhege, C.; Eyerich, P.; Keller, T.; Trúg, S.; Brenner, M.; and Nebel, B. 2009. Semantic attachments for domain-independent planning systems. In *Proc. of International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2009. Using the context-enhanced additive heuristic for temporal and numeric planning. In *19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

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 *Proc. of the Conference on Automated Planning and Scheduling, (ICAPS 2015)*, 70–78.

Gerevini, A.; Saetti, I.; and Serina, A. 2008. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artificial Intelligence* 172(8-9):899–944.

Getoor, L.; Ottosson, G.; Fromherz, M. P. J.; and Carlson, B. 1997. Effective redundant constraints for online scheduling. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island.*, 302–307.

Gregory, P.; Long, D.; Fox, M.; and Beck, J. C. 2012. Planning modulo theories: Extending the planning paradigm. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, 140–149.

Haslum, P. 2009. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In *Proc. of the International Conference on Automated Planning and Scheduling, (ICAPS 2009)*.

Helmert, M. 2002. Decidability and undecidability results for planning with numerical state variables. In *Proc,. of International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, 44–53.

Hoare, C. A. R. 1969. An axiomatic basis for computer programming. *Commun. ACM* 12(10):576–580.

Hoffmann, J. 2003. The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research (JAIR)* 20:291–341.

Koehler, J. 1998. Planning under resource constraints. In *Proc. 13th European Conference on Artificial Intelligence (ECAI 1998)*, 489–493.

Moore, R. E.; Kearfott, R. B.; and Cloud, M. J. 2009. *Introduction to Interval Analysis*. SIAM.

Scala, E. 2013. Numeric kernel for reasoning about plans involving numeric fluents. In et al., M. B., ed., *AI*IA 2013, LNAI 8249, Springer International Publishing Switzerland*.