Effect-Abstraction Based Relaxation for Linear Numeric Planning

Dongxu Li[1], **Enrico Scala**[2], **Patrik Haslum**[1,3], **Sergiy Bogomolov**[1]
[1] The Australian National University
[2] Fondazione Bruno Kessler (Italy)
[3] CSIRO Data61
dongxu.li@anu.edu.au, enricos83@gmail.com,
patrik.haslum@anu.edu.au, sergiy.bogomolov@anu.edu.au

Abstract

This paper studies an effect abstraction-based relaxation for reasoning about linear numeric planning problems. The effect abstraction decomposes non-constant linear numeric effects into actions with conditional, additive constant numeric effects. With little effort, on this abstracted version, it is possible to use known subgoaling-based relaxations and related heuristics. The combination of these two steps leads to a novel relaxation-based heuristic. Theoretically, the relaxation is proved tighter than the previous interval-based relaxation and leading to pruning-safe heuristics. Empirically, a heuristic developed on this relaxation leads to substantial improvements for a class of problems that are currently out of reach of state-of-the-art numeric planners.

1 Introduction

Numeric planning extends classical planning with numeric state variables. In numeric planning problems, an action's applicability and effects may depend on expressions over several numeric variables. With this expressiveness many problems find compact and closer-to-reality representation [Fox and Long, 2003]. Examples of its application span a variety of problems, from traffic control [Vallati *et al.*, 2016] to robotics [Kiam and Schulte, 2017].

However, the introduction of numeric structures makes the planning task more challenging. Solvability of the numeric planning problem is undecidable in general ([Helmert, 2002]). Therefore, since its introduction efforts have been made to derive meaningful relaxations, and heuristics, for numeric planning [Hoffmann, 2003; Coles and Coles, 2011]. Because plan validation is still decidable, such heuristics can make informed systematic forward search algorithms (such as A* or Greedy Best First Search) effective, and can be used to guide the solution of even complex control-like problems [Scala *et al.*, 2016b; Piotrowski *et al.*, 2016]. Relaxations have been proposed for a severely restricted fragment, known as "simple" numeric planning, in which actions may only increase/decrease numeric variables by a constant [Scala *et al.*, 2016a; Piacentini *et al.*, 2018], and for a very general form allowing arbitrary expressions using algebraic and trascendental functions in conditions and effects of the problem [Scala *et al.*, 2016b; Aldinger *et al.*, 2015]. However, the support of such a general class of planning problems comes at the price of looser relaxations, and consequently weaker heuristics.

In this paper we propose an effect abstraction-based relaxation that is particularly suited for a subclass of numeric planning problems where both conditions and action effects are made up of linear expressions over numeric state variables. This class is less expressive than general numeric planning, but more expressive than simple numeric planning. Our key contribution is to show that with the right abstraction, it is possible to transform a linear numeric planning problem into a relaxed, simple numeric planning problem; by doing so it becomes possible to apply previous relaxations that are suited only for the simple numeric case also to defining heuristics for linear numeric planning.

We show a condition ensuring that the numeric effect abstraction combined with one relaxation, namely Π^1, is pruning-safe and leads to the development of a heuristic that proves to be well informed versus other heuristics presented in literature. We also revisit previous relaxations for simple numeric planning and analyse their relative pruning power.

2 Numeric Planning

We assume a ground planning formalism equivalent to PDDL 2.1 level 2 [Fox and Long, 2003]. A planning problem consists of propositional (F) and numeric (X) state variables, actions, an initial state (s_0) and a goal (G). A state assigns a truth value to each proposition in F and a rational number to each state variable in X. A numeric condition takes the form $\xi \trianglerighteq k$, where ξ is an arithmetic expression over X, $\trianglerighteq \in \{\leq, <, =, >, \geq\}$ and $k \in \mathbb{Q}$ is a constant. Action preconditions and the goal are sets (conjunctions) of numeric conditions and propositions. Action effects can assign boolean variables and/or increase/decrease/assign the value of numeric variables by a numeric expression. A conditional effect is a pair $\langle c, e\rangle$, where c is a set of (numeric or propositional) conditions and e is an effect. e is applied only if c is true in the state where the action is applied. Action a is applicable in a state s iff its preconditions are satisfied in s and

the activated numeric or propositional effects do not induce conflicting assignments. A sequence of actions $\langle a_0, ..., a_{n-1}\rangle$ starting from the initial state s_0 brings (deterministically) the system to a state (s_n). If each action is applicable in the state resulting from the prefix before it and the final state satisfies the goal (i.e., $s_n \models G$), the sequence is a *valid plan*. Each action a_i has a cost $\lambda(a_i) \in \mathbb{R}^{\geq 0}$ and the cost of a plan $\pi = \langle a_0, ..., a_{n-1}\rangle$ is $\sum_{0\leq i<|\pi|} \lambda(a_i)$; the objective is to minimize plan cost.

Notation. Let s be a state and ξ a numeric expression: $[\xi]^s$ is the value of ξ in s. Given action a, $\mathrm{pre}(a)$ and $\mathrm{eff}(a)$ are its preconditions and effects. A numeric effect e is the triple $\langle \mathrm{lhs}(e), \mathrm{op(e)}, \mathrm{rhs}(e)\rangle$ where $\mathrm{lhs}(e)$ is the affected variable, $\mathrm{rhs}(e)$ is a numeric expression and $\mathrm{op(e)}$ is an increase, decrease, or assign. $\mathcal{S}(\Pi)$ denotes the state space induced by the planning problem, and $Sol(\Pi)$ the set of all valid plans.

Definition 1 (Simple Numeric Planning)**.** *Let Π be a numeric planning problem. A numeric condition $\xi \trianglerighteq k$ is* simple *iff ξ is a linear expression and all variables in ξ are affected only by constant increase/decrease effects in Π. Π is simple iff all action preconditions and the goal are simple.*

2.1 The (Additive) Interval-Based Relaxation

The interval-based relaxation is a natural generalisation of the monotonic relaxation of propositional planning to the numeric setting ([Hoffmann, 2003; Aldinger *et al.*, 2015; Scala *et al.*, 2016b]). It has three main elements: (i) a *relaxed state* assigns each numeric variable an interval of possible values; (ii) for a numeric condition to be satisfied in a relaxed state it suffices that the condition holds for some choice of value for each appearance of numeric variables out of their respective intervals, independently of other conditions; and (iii) applying a numeric effect to a variable monotonically extends its interval to the convex union of its current interval and the possible values of the effect expression.

The additive interval-based relaxation (AIBR) transforms assignment effects into equivalent additive effect by a simple reformulation ($x := \xi$ to $x \mathrel{+}= (\xi - x)$), and then applies the interval-based relaxation to the transformed problem. We denote this relaxation of a problem Π with Π^{++}. This simple expedient, combined with the notion of asymptotic supporters, allows reachability under AIBR to be computed in polynomial time also for problems with cyclic numeric dependencies [Scala *et al.*, 2016b]. Reachability can be decided by constructing the so called asymptotic relaxed planning graph, which applies asymptotic supporters until a fixpoint is reached (see [Scala *et al.*, 2016b] for details). As a side-effect, this process also computes for each numeric variable an interval that is guaranteed to contain all reachable values of the variable.

A relaxation, Π^R is said to be *pruning-safe* iff $Sol(\Pi) \neq \emptyset \implies Sol(\Pi^R) \neq \emptyset$, i.e., the relaxation is unsolvable only if the original problem is unsolvable. Likewise, we say that a heuristic is pruning-safe if it reports an infinite value only for unsolvable states. Π^{++}, and heuristics based on it, are pruning-safe.

Heuristics built on the principle of interval-based relaxation abound in the literature [Hoffmann, 2003; Coles and Coles, 2011; Coles *et al.*, 2012; 2010]. A simple heuristic based on AIBR is h^{aibr} obtained by first using the asymptotic relaxed planning graph construction to decide if the goal is relaxed reachable, and, in case it is, repeatedly applying actions until goal is reached; the number of actions applied is the estimate. The h^{FF}_{LNF} heuristic, from the Metric-FF planner [Hoffmann, 2003], translates numeric effects into the so called Linear Normal Form, then combines the construction of a relaxed planning graph and a greedy algorithm to compute a relaxed plan solving the interval-based relaxation of the problem. This heuristic is the basis of many state of the art temporal and numeric planners [Coles *et al.*, 2010; 2012]. However, h^{FF}_{LNF} only supports planning with linear numeric effects with non-cyclic dependencies.

2.2 The Subgoaling-Based Relaxation

The generality of the interval-based relaxation comes at the price of not being able to fully exploit the numeric structure of the problem at hand. The subgoaling-based relaxation targets one such structure, namely which actions contribute positively to the achievement of linear numeric conditions, using the so called m-times-regressor [Scala *et al.*, 2016a]:

Definition 2. *Let $c = \left(\sum_{x\in X} w_{x,c}x\right) + w_{n,c} \trianglerighteq 0$, with $\trianglerighteq \in \{\leq, <, >, \geq\}$, be a simple numeric condition (SC). The m-times regression $c^{r(a,m)}$ of c through action a is:*

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

where $k_{x,a}$ is the constant additive effect of a on x.

We say that a is a *possible achiever* of c in a state s if there exists an m such that $s \models c^{r(a,m)}$. In the case of a simple numeric condition, all elements of (1) except m are coefficients extracted from the action model. This results in the regression being a linear function of time, having a constant derivative. Thus, which actions are possible achievers can be detected statically (independent of s). Yet, the number m of repetitions needed to actually achieve the condition is state-dependent.

The numeric subgoaling relaxation [Scala *et al.*, 2016a] uses this notion of possible achiever to over-approximate reachable numeric conditions recursively. The principle can be used to derive admissible or inadmissible heuristic estimates. The inadmissible formulation is as follows[1]:

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

[1] HCs is the set of non-simple (Hard) numeric Conditions.

We use Π^1 to denote the subgoaling relaxation of Π. Π^1 has a solution iff $\hat{h}^{add}_{hbd+} \neq \infty$.[2] A solution to this relaxation is a set of multi-set of achievers for each condition to be used to reach the goal. Π^1 is a pruning-safe relaxation.

Theorem 1. $Sol(\Pi) \neq \emptyset \implies Sol(\Pi^1) \neq \emptyset$

Proof Sketch. Let π be a solution for Π. Each (sub)goal c in Π is either satisfied in the initial state s_0, or there is a sequence of actions whose application from s_0 makes it true. From this sequence we can construct possible achievers such that the regression of c through them is satisfied in s_0. Therefore $\hat{h}^{add}_{hbd+} < \infty$. □

2.3 Handling Conditional Effects

Π^1-based heuristics can support conditional effects by transforming the actions set in a way similar to that presented by Röger et al. [2014]. For each action $a \in A$ and each conditional effect $ce : \langle c, e\rangle$ in $\mathrm{eff}(a)$ we (i) create a new action a' where $\mathrm{pre}(a') := \mathrm{pre}(a) \cup \{c\}$ and $\mathrm{eff}(a') := \{e\}$ and (ii) remove ce from a. Note that the transformed problem is used only for computing the heuristic. For simplicity, $\hat{h}^{add}_{hbd+}$ and $\hat{h}^{max}_{hbd+}$ are used also for the heuristic applied to numeric planning problems with conditional effects. The transformation does not necessarily preserve admissibility, unless the cost of actions corresponding to conditional effects is set to zero.

2.4 Towards Linear Numeric Planning Problems

In the following, we focus our attention on a subclass of general numeric conditions that are not simple, but still linear.

Definition 3 (Linear Numeric Planning)**.** *A* linear numeric condition *is a condition $\xi \trianglerighteq k$ where ξ is a linear expression such that all effects on variables appearing in ξ are increase or decrease effects whose right-hand side is a linear expression. A numeric planning problem is called linear iff all numeric action preconditions and the goal are linear conditions.*

A numeric effect featuring a linear but non-constant right-hand side is said to be *non-constant linear*.

The class of linear numeric planning problems as we have defined it is similar to the linear numeric planning tasks defined by Hoffman [2003]. However, it does not include explicit assignment effects, which can be supported via the additivity transformation [Scala *et al.*, 2016b] without the acyclicity requirement as it is instead the case for Hoffman's linear numeric planning fragment.

3 The Effect-Abstraction Compilation

The relaxation-based heuristics developed so far for numeric planning are either very widely applicable, or restricted to a limited problem class such as simple numeric planning. Our objective is to find a middle ground between these two. We do so by applying an abstraction mechanism to numeric action effects, which we call *effect abstraction*. In this section we present the general effect abstraction schema, while in Section 4 we study the Π^1 relaxation applied to the effect abstracted problem.

Different from previous abstraction-based heuristics, e.g., [Helmert *et al.*, 2007; Illanes and McIlraith, 2017], the effect abstraction does not shrink the state space of the problem, but rather it coarsens (non-simple) numeric effects of actions.

Definition 4 (Decomposition and Tags of a Numeric effect)**.** *Let e be an additive numeric effect, i.e., an effect with operator $+=$ or $-=$. A* decomposition *of e, $D(e)$ is a set of disjoint non-empty intervals whose union contains all reachable values of $\mathrm{rhs}(e)$ except 0. $T(e) : D(e) \to \mathbb{Q}$ is a function such that for all $l \in D(e)$, $T(e)(l) \in D(e)$, i.e., that selects for each interval in the decomposition a specific value in that interval. $T(e)(l)$ is called the* tag *of the interval l.*

A pair of functions $\langle D, T\rangle$ that provide the decomposition and tags for each non-constant numeric effect in a planning problem Π is an effect decomposition *of Π.*

A decomposition $D(e)$ is a partitioning of the set of values that the right-hand side of the numeric effect can take. For each partition, its tag $T(e)(l)$ is a commitment to one of these values. The abstraction that we will apply is to treat the effect as having the value $T(e)(l)$ whenever $\mathrm{rhs}(e) \in l$. Excluding 0 from the decomposition is natural since an additive effect has no effect when its right-hand side is zero, and it simplifies the statement of Theorem 2 below. Apart from this exclusion, $D(e), T(e)$ forms a *tagged partition* as defined by Gordon [1998]. Note that $D(e)$ only needs to contain (not equal) the reachable non-zero values of $\mathrm{rhs}(e)$; thus, any partitioning of $\mathbb{Q} - \{0\}$ into intervals is permissible.

Starting from the actions of the original problem, we express the effect abstraction with conditional effects. Given an effect decomposition $\langle D, T\rangle$, from each action a we generate a new action a' which has each non-constant additive effect $\langle \mathrm{lhs}(e), \mathrm{op(e)}, \mathrm{rhs}(e)\rangle$ of a replaced with a set of conditional effects of the form $\langle \mathrm{rhs}(e) \in l, \langle \mathrm{lhs}(e), \mathrm{op(e)}, T(e)(l)\rangle\rangle$, one for each interval $l \in D(e)$, but is otherwise identical to a. We call the *Piecewise Conditional Effects Abstraction* of the action, and denote it $\Gamma_{\langle D,T\rangle}(a)$. Formally:

Definition 5 (Piecewise Conditional Effects Abstraction)**.** *Let $\langle D, T\rangle$ be an effect decomposition of Π and a an action in Π. Partition $\mathrm{eff}(a)$ into a set $\mathrm{eff}^{nca}(a)$ containing all additive numeric effects of a with a non-constant right-hand side and $\mathrm{eff}^{other}(a) = \mathrm{eff}(a) - \mathrm{eff}^{nca}(a)$. $\Gamma_{\langle D,T\rangle}(a)$ is an action a' such that $\mathrm{pre}(a') = \mathrm{pre}(a)$ and*

$$\mathrm{eff}(a') = \mathrm{eff}^{other}(a) \cup \bigcup_{e \in \mathrm{eff}^{nca}(a)} \{\langle \mathrm{rhs}(e) \in l, \langle \mathrm{lhs}(e), \mathrm{op(e)}, T(e)(l)\rangle\rangle : l \in D(e)\}.$$

For an interval $l \in D(e)$, $\underline{l}$ and $\overline{l}$ denote the lower and upper bounds of l, respectively. The condition $\mathrm{rhs}(e) \in l$ is expressed by the conjunction of $\mathrm{rhs}(e) \rhd \underline{l}$ and $\overline{l} \rhd \mathrm{rhs}(e)$, where each operator $\rhd$ is either $>$ or $\geq$, depending on if l is open or closed at that end.

[2] An admissible variant $\hat{h}^{max}_{hbd+}$ can be constructed by using only reachability testing for hard numeric conditions. This preserves admissibility, and has the same unreachability detection power as $\hat{h}^{add}_{hbd+}$.

For an example, consider an action with the linear effect $x \mathrel{+}= y + z$. A possible decomposition could be into the intervals $[-\infty, 0)$, $(0, 5]$ and $(5, \infty]$, with the tag function $T([-\infty, 0)) = -1$, $T((0, 5]) = 5$, and $T((5, \infty]) = 10$. In the piece-wise conditional effect abstraction of the action, the linear effect is replaced with the three conditional effects $\langle y + z < 0, x \mathrel{+}= -1\rangle$, $\langle 0 < y + z \wedge y + z \leq 5, x \mathrel{+}= 5\rangle$ and $\langle 5 < y + z, x \mathrel{+}= 10\rangle$.

The effect abstraction transformation extends naturally to planning problems:

Definition 6 (Numeric Effect Abstraction)**.** *The Numeric Effect Abstraction of* $\Pi = \langle F, X, s_0, A, G\rangle$ *is the planning problem* $\Gamma_{\langle D,T\rangle}(\Pi) = \langle F, X, s_0, \{\Gamma_{\langle D,T\rangle}(a) : a \in A\}, G\rangle$.

The effect abstraction has similarities with the construction of asymptotic supporters defined for the AIBR relaxation. The AIBR asymptotic supporter construction can be interpreted as an effect abstraction function assigning two open-ended intervals $(-\infty, 0)$, $(0, \infty)$ to each numeric effect, and taking as tags $-\infty$ and ∞.

It is easy to see that when every expression in the original problem is linear (or can be made linear by simple invariant analysis), then applying $\Gamma_{\langle D,T\rangle}$ results in a simple numeric planning problem with conditional effects.

Observation 1. *If* Π *is a linear numeric planning problem, then* $\Gamma_{\langle D,T\rangle}(\Pi)$ *is a simple numeric planning problem for any effect decomposition* $\langle D, T\rangle$.

Although the effect abstraction makes the problem literally simpler, the concrete choice of the decomposition $\langle D, T\rangle$ is critical to making the transformation useful. In the next section we show several properties of the subgoaling relaxation of the abstraction, $\Gamma_{\langle D,T\rangle}(\Pi)^1$. In Section 5 we propose a pragmatic choice of decompositions, which we then evaluate empirically in Section 6.

Towards Non-Linear Numeric Effects. Abstracting a linear numeric effect makes the piece-wise effect conditions linear (and hence simple in the abstracted problem). However, a non-linear effect expression can also be partitioned with linear (simple) conditions on the variables that appear in it. For example, the restriction of the effect $x \mathrel{+}= y \times z$ to the interval $(0, \infty]$ can be expressed with the two conditions $y > 0 \wedge z > 0$ and $y < 0 \wedge z < 0$, resulting in two conditional effects. Unlike the linear effect case, however, there is no general, automatic procedure to compute the conditions of the decomposition for general non-linear effects.

4 Theoretical Analysis

4.1 A Safe Effect-Abstraction Relaxation

We say that an effect abstraction is *safe* with respect to a relaxation iff the relaxation applied to the effect-abstracted problem is unsolvable only when the original problem is actually unsolvable.

Theorem 2 (Numeric Effect Abstraction Safeness w.r.t. Π^1)**.** *Let* $\Pi = \langle F, X, A, G, I\rangle$ *be a linear numeric planning problem and* $\Gamma_{\langle D,T\rangle}(\cdot)$ *a numeric effect abstraction. If for every non-constant linear effect* e *in* Π *and for every* $l \in D(e)$ *it holds that* $\underline{l} \cdot \bar{l} > 0$*, then* $Sol(\Pi) \neq \emptyset \implies Sol(\Gamma_{\langle D,T\rangle}(\Pi)^1) \neq \emptyset$.

Proof Sketch. We only need to prove that a plan for Π implies the existence of a plan for $\Gamma_{\langle D,T\rangle}(\Pi)^1$. Let $\pi \in Sol(\Pi)$; the difficult part is to show that we can find possible achievers that are reachable in $\Gamma_{\langle D,T\rangle}(\Pi)^1$ for those conditions that are reached by π (at some stage) and whose reachability depends on using at least one non-constant linear effect. We proceed by induction over π. Any condition reached by an empty plan is satisfied in the initial state, and thus trivially reachable also in $\Gamma_{\langle D,T\rangle}(\Pi)^1$. A condition reached by π that is not true in the initial state implies the existence of some action a_i in π making c true from the state s_{i-1} reached by the prefix of π up to a_i. If a_i has a non-constant linear effect e acting positively on c in the state s, then the value $[\mathrm{rhs}(e)]^s$ makes a_i a *possible achiever* of c (under Π^1 semantics). From the combination of the effect abstraction and the subgoaling over conditional effects (Section 2.3), we know there is an action a' in $\Gamma_{\langle D,T\rangle}(\Pi)^1$ whose precondition combines the precondition of a_i with an interval l that includes $\mathrm{rhs}(e)$. Because of the condition $\underline{l} \cdot \bar{l} > 0$, we also know that the effect of this action on $\mathrm{lhs}(e)$, which is the tag $T(e)(l)$, is in the same direction as $[\mathrm{rhs}(e)]^s$; in fact, using the tag, or any value from l, in place of $[\mathrm{rhs}(e)]^s$ does not affect the asymptotic behavior of the effect, so a' is a possible achiever of c. The preconditions of a' are the preconditions of a_i, and $\mathrm{rhs}(e) \rhd \underline{l}$ and $\bar{l} \rhd \mathrm{rhs}(e)$ (with $\rhd \in \{>, \geq\}$) which, due to the choice of l, are each achieved by the prefix of π up to a_i, and thus reachable in $\Gamma_{\langle D,T\rangle}(\Pi)^1$ by inductive assumption. If a_i achieves c with a constant numeric or propositional effect, it does so also in $\Gamma_{\langle D,T\rangle}(\Pi)^1$. Finally, observe that the subgoaling relaxation considers reachability of conditions separately; thus, since all conditions achieved by the prefix up to and including a_i are reachable, in some way, in $\Gamma_{\langle D,T\rangle}(\Pi)^1$, so is any conjunction of them. □

Next, we study the relationship between $\Gamma_{\langle D,T\rangle}(\Pi)^1$ and the additive interval-based relaxation. Unless otherwise stated, in the following we consider only effect decompositions that satisfy the condition of Theorem 2.

4.2 Comparing Different Relaxations/Heuristics

Any effect-abstraction subgoaling relaxation, regardless of the choice of D and T, is tighter than the AIBR. To prove this, we first need to prove that subgoaling is tighter than AIBR.

Theorem 3. *For any simple numeric planning problem* Π*:*

1. $Sol(\Pi^1) \neq \emptyset \implies Sol(\Pi^{++}) \neq \emptyset$
2. $Sol(\Pi^{++}) \neq \emptyset \not\Longrightarrow Sol(\Pi^1) \neq \emptyset$

Proof Sketch. (1). If c is reachable in Π^1 there is some reachable action a that contributes positively to achieving it. From any interval I, there exists a finite number m of times that a applied on I under Π^{++} semantics makes c true, because Π^{++} grows intervals by convex union with action effects. Moreover, given that the preconditions of a are individually reachable in Π^1, they are also reachable individually in Π^{++} by inductive assumption; the monotonicity of Π^{++} means that the conjunction of all the action's preconditions is reachable in Π^{++}, by the concatenation of the plans for

the individual conditions. (2). Consider a numeric condition $c' : x > y$, a state $s : \{x = 0, y = 0\}$ and an action $a : \langle \emptyset, \{(x \mathrel{+}= 1), (y \mathrel{+}= 1)\}\rangle$. c can be reached under Π^{++} semantics, but not under Π^1 semantics because a is not a possible achiever for c. □

Theorem 4. *Let Π be a linear numeric planning problem, and $\Gamma_{\langle D,T\rangle}\Pi^1$ a safe effect abstraction relaxation:*

1. $Sol(\Gamma_{\langle D,T\rangle}\Pi^1) \neq \emptyset \implies Sol(\Pi^{++}) \neq \emptyset$
2. $Sol(\Pi^{++}) \neq \emptyset \not\Longrightarrow Sol(\Gamma_{\langle D,T\rangle}\Pi^1) \neq \emptyset$

Proof Sketch. (2) follows since $\Gamma_{\langle D,T\rangle}\Pi^1$ inherits the power of the reasoning over simple numeric condition from the subgoaling relaxation, as shown in Theorem 3. (1) can be shown by an argument similar to that developed for the previous theorem. In particular, from a plan found in $Sol(\Gamma_{\langle D,T\rangle}\Pi^1)$ it is possible to extract asymptotic supporters that are inductively reachable, and can be used to make the asymptotic relaxed planning graph procedure return true. □

Note that the effect-abstraction subgoaling relaxation dominates AIBR with respect to detecting unreachability (result 2 of Th. 4) even if the enclosing interval of the decomposition D is the whole set of rational numbers.

5 Practical Decompositions

To implement an actual heuristic based of the effect abstraction we need to (i) (over-)approximate reachable values for the right-hand side of each non-constant linear numeric effect, (ii) select a decomposition satisfying the condition of Th. 2, and (iii) determine an actual tag function. Point (ii) introduces a trade-off between informativeness and computational cost of the heuristic. A decomposition into many intervals captures more *contingencies* and results in higher accuracy of the representation; on the other hand, many intervals for each numeric effect means higher computational cost for the Π^1 relaxation, which can outweigh the information gain it might bring during search.

AIBR-based Decomposition (ABD). To overcome issue (i) and (ii) we extract information to predict $\mathcal{S}(\Pi)$ and its structure from Π^{++}. More precisely, we use asymptotic reachability to over-approximate the reachable values for each variable in form of an interval, and, using interval analysis, the reachable values of each non-constant linear effect. Returning to the example effect $x \mathrel{+}= y + z$, if the asymptotically reachable values of y and z are contained in $[0, 5]$ and $[2, \infty)$, respectively, the interval of possibly reachable values of the expression $y + z$ is $[2, \infty)$. Next, we use h^{aibr} to predict *how* these intervals change. Precisely, we take the sequence of relaxed states obtained to reach the goal (under Π^{++} semantics), and manipulate this to get a view of "relevant" (w.r.t. the goal) changes for each non-constant linear effect.

For each right hand side expression ξ, we generate an "increment" interval sequence (IIS), which partitions the reachable values of ξ. The IIS is a sequence of intervals providing a differential view of the values of ξ. Suppose from h^{aibr} we obtain the intervals $I : [[2, 2], [2, 5], [2, 20]]$ for $\xi = y + z$; then $IIS(I) : [[2, 2], (2, 5], (5, 20]]$. Alg. 1 shows how the IIS is generated; besides $\mathrm{rhs}(e)$, it takes as input s_∞ and S_c, i.e., the relaxed goal state after asymptotic reachability and the sequence of relaxed states provided by h^{aibr}. The algorithm iterates over $S_c \oplus [s_\infty]$ incrementally populating the IIS associated with $\mathrm{rhs}(e)$. It differentiates when the interval is extending the largest or the smallest value enclosed by IIS ($min(\cdot)$ and $max(\cdot)$). To meet condition of Th. 2, `addElement` makes sure none of the intervals includes 0. Alg. 1 is called for each effect in the Piecewise Conditional Effects Abstraction.

Algorithm 1: AIBR-based-Decomposition

Input: $rhs(e)$ – right-hand side expression of effect e
s_∞ – relaxed goal state from AIBR reachability analysis
S_c – ordered relaxed states from AIBR counting
Output: IIS - Increment Interval Sequence

```
1  S_c := S_c ⊕ [s_∞]
2  IIS := [[rhs(e)]^{S_c[0]}];
3  for i from 1 to |S_c| − 1 do
4  |  addElement(IIS, [rhs(e)]^{S_c[i]})
5  end
6  return IIS
7  Procedure addElement (L – List, I – Interval)
8  |  if I̲ < min(L) then
9  |  |  if I̲ · min(L) < 0 then
   |  |  |  L := [[I̲, 0), (0, min(L)]] ⊕ L
10 |  |  else L := [[I̲, min(L))] ⊕ L
11 |  end
12 |  if Ī > max(L) then
13 |  |  if Ī · max(L) < 0 then
   |  |  |  L := L ⊕ [[max(L), 0), (0, Ī]]
14 |  |  else L := L ⊕ [(max(L), Ī]]
15 |  end
```

Midpoint Tag Function (MTF). Starting from the IIS generated for a non-constant numeric effect, we need to select a tag, i.e., a representative, for each of the intervals in IIS. While this does not affect the safety of the effect abstraction, it does affect the heuristic estimates for reachable conditions. Because the abstraction is done offline (i.e., prior to search), we take a compromise between the two extremes of the interval. For non-diverging intervals (where neither end is open to ∞ or $-\infty$) we select as the tag its midpoint; for diverging intervals we choose a value at the finite end. We call this the Midpoint Tag Function, and it is defined as follows:

$$MTF(I) = \begin{cases} \underline{I} + \epsilon & \text{if } \overline{I} = \infty \\ \overline{I} - \epsilon & \text{if } \underline{I} = -\infty \\ \frac{\underline{I}+\overline{I}}{2} & \text{otherwise} \end{cases}$$

with $\epsilon \in \mathbb{Q}^{>0}$ to ensure the tag lies within the interval.

We call h^{add}_{abs} the heuristic obtained by using the effect abstraction through ABD and MTF, and using the estimate provided by h^{add}_{hbd+} on $\Gamma_{ABD,MTF}(\Pi)$.

Other tag functions are of course possible. For instance, selecting always an extreme of the interval may result in a more "conservative" evaluation, and thus a heuristic that favours cheaper plans. Exploring the impact of tag functions on the heuristic is a question for future work.

Theorem 5. *h^{add}_{abs} is (i) polynomial in the size of the problem and $|S_c|$, and (ii) returns infinity only for unsolvable instances*

Proof Sketch. (i) The number of actions after decomposition is bounded by the number of intervals given by ABD, which is linear in the size of the relaxed plan obtained by h^{aibr}, and so $|S_c|$. h^{add}_{hbd+} applied to the abstract problem, i.e. $\Gamma_{ABD,MTF}(\Pi)$, is polynomial in the instance size. (ii) Direct consequence of Th. 2. ABD over-approximates reachable values, and splits intervals so as not to include 0. □

6 Experiments

In this section, we study the practical implications of the effect abstraction implemented by the h^{add}_{abs} schema. We implemented h^{add}_{abs} in the ENHSP planner[3], and compared it to other heuristics in a greedy best-first search ($f(n) = h(n)$ with ties broken in favor of lower g-values). The other heuristics are h^{aibr} [Scala *et al.*, 2016b], $\hat{h}^{add}_{hbd+}$ [Scala *et al.*, 2016a], and the Metric-FF heuristic, herein called h^{FF}_{LNF} [Hoffmann, 2003]. h^{FF}_{LNF} is obtained by running Metric-FF in greedy best-first search with no helpful actions. We also compare with Metric-FF in its default configuration.

Domains. We extended simple numeric planning domains from the literature [Scala *et al.*, 2016a; Francès and Geffner, 2015; Piacentini *et al.*, 2018] by changing constant numeric effects to be dependent on "second-order" variables, which are in turn changed by constant effects. This emphasises reasoning over linear but non-simple numeric conditions. More precisely: **FO-COUNT** extends the COUNTERS domain [Francès and Geffner, 2015]. Like in the original formulation, there are N numeric variables, $X_1, X_2, ..., X_N$, each of which can be increased or decreased by a variable delta, $\Delta X_1, \Delta X_2, ..., \Delta X_N$, which in turn can be changed by constant steps in the range from 0 to a constant maximum. The goal is to set the values of the counters in ascending order. Instances are split in three groups, with the initial values all 0, ordered inversely to the goal, or randomly chosen (three for each size). Instances scale on the number of counters. **FO-SAILING** extends the SAILING domain [Scala *et al.*, 2016a]. The boat speed (along the 7 compass directions) is here a function of the direction of the boat and of a variable factor that can be increased/decreased by a second-order set of actions, again within the range from 0 to a constant maximum. Instances scale on the number of boats and people to be rescued. **FO-FARMLAND**. This domain extends FARMLAND [Scala *et al.*, 2016a] by the introduction of a new action (move by car) with which it is possible to move more workers per time from place to place. The more cars are hired, the more people can be moved per action, but at a higher cost. Instances scale on the number of farms and workers. We also add **TPP-METRIC**, which is the only IPC domain featuring non-simple linear numeric conditions.

Results. h^{add}_{abs} dominates all the other heuristics across the tested domains regarding coverage (Table 1). Except for FO-COUNT, h^{add}_{abs} expands less nodes, improving run-time and coverage. In FO-COUNT optimal solutions require firstly increasing the rate of change of the variables, and then increase the respective counters. The h^{aibr} relaxation favors more committed states, which are those where less actions can be done. These states correspond to states where the rate is set to maximum. h^{aibr} and $\hat{h}^{add}_{hbd+}$ perform better than h^{add}_{abs} as they capture this necessity, yet yield unnecessary increases of the delta variables and thus resulting in longer plans. In FO-COUNT-INV and FO-COUNT-RND, h^{add}_{abs} is more informed as decompositions result in finer granularities (the goal is not immediately satisfied after one step of h^{aibr}). In the remaining domains, h^{add}_{abs} converges faster but with longer plans than h^{aibr}, especially for FO-FARMLAND. Loss of plan quality is explained by h^{add}_{abs} over-estimation when subgoals are not completely independent. (This happens to some extent in $\hat{h}^{add}_{hbd+}$too, yet we do not observe a general trend when comparing it with h^{aibr}.) Surprisingly, h^{add}_{abs} turns out to be competitive even with the full Metric-FF system on the majority of the domains. This result is quite significant as h^{add}_{abs} has a large handicap: 1) it does not use local search techniques such as Enforced Hill Climbing (EHC), and 2) – probably more important – does not make any use of helpful actions. Metric-FF outperforms the other techniques on the TPP-METRIC domain by far. Note that all instances of TPP-METRIC can be solved in seconds with depth-first search, which is what Metric-FF's tie-breaking plus EHC exploits. h^{add}_{abs} internally reasons with a larger number of (heuristic) actions. In the FO-COUNT domain, the number of heuristic actions are on average 3.5 times the number of original actions; in FO-SAILING it is 12.87, in FO-FARMLAND it is 5.31 and in TPP-METRIC it is 8.97 times. It has been observed that for large instances of TPP-METRIC, the heuristic becomes very slow, and is probably the bottleneck of the approach. A possible solution to this problem is to bound the number of intervals in the decomposition, leading to fewer heuristic actions.

[3]https://gitlab.com/enricos83/ENHSP-Public

7 Conclusion and Future Work

We presented a novel relaxation for a subclass of numeric planning inbetween the simple case and the fully general problem. The relaxation rests on a novel effect abstraction whose main objective is making the problem numerically simple. We presented a schema for an inadmissible heuristic based on this relaxation. We provided (i) a theoretical analysis of effect abstraction, in particular safeness and tightness w.r.t. other numeric planning relaxations, and (ii) empirical evidence of the usefulness of this approach. Future work is twofold: first, to study whether other relaxations/heuristics (e.g., [Piacentini *et al.*, 2018]) can be combined with effect abstraction whilst preserving safeness and informativeness; second, to extend the automatic construction of effect abstractions to fragments of numeric planning beyond linear.

Acknowledgments

This work was funded by ARC project DP140104219 (Robust AI Planning for Hybrid Systems), and partly supported by the Air Force Office of Scientific Research award no.

	Coverage			CPU-Time (s)			Plan Length			Exp. Nodes			Coverage	
	h^{add}_{abs}	h^{aibr}	$\hat{h}^{add}_{hbd+}$	h^{add}_{abs}	h^{aibr}	$\hat{h}^{add}_{hbd+}$	h^{add}_{abs}	h^{aibr}	$\hat{h}^{add}_{hbd+}$	h^{add}_{abs}	h^{aibr}	$\hat{h}^{add}_{hbd+}$	h^{FF}_{LNF}	M-FF
FO-COUNT(20)	**8**	**8**	**8**	39.4	**8.9**	56.3	**17.5**	20.4	21.9	24991.1	5807.8	**2239.6**	1	8
FO-COUNT-INV(20)	**8**	6	6	**1.0**	67.7	13.0	**22.0**	24.0	26.5	**804.0**	70880.8	970.3	1	7
FO-COUNT-RND(60)	**31**	24	21	**9.6**	33.0	123.1	**19.7**	22.3	19.9	**5755.3**	25085.0	9582.7	0	23
FO-SAILING(20)	**17**	4	5	**1.0**	344.0	160.6	91.0	**74.0**	126.3	**92.0**	997881.7	36323.0	0	11
FO-FARMLAND(50)	**50**	**50**	**50**	**0.7**	2.0	64.8	58.1	26.8	**26.3**	**60.4**	638.8	172.4	0	38
TPP-METRIC(40)	**20**	8	10	**2.9**	123.3	107.8	**20.5**	20.8	23.2	**29.6**	91546.9	144.0	6	40
Total	**134**	100	100										8	127

Table 1: Comparison between heuristics h^{add}_{abs}, h^{aibr}, $\hat{h}^{add}_{hbd+}$, h^{FF}_{LNF} and the Metric-FF planning system (M-FF). Time, plan length and expansions are averages over instances solved with the first three heuristics. Bold is for best performer. Timeout is 1800 seconds.

FA2386-17-1-4065. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

References

[Aldinger *et al.*, 2015] Johannes Aldinger, Robert Mattmüller, and Moritz Göbelbecker. Complexity of interval relaxed numeric planning. In *Proc. of KI 2015: Advances in Artificial Intelligence*, pages 19–31, 2015.

[Coles and Coles, 2011] Amanda Jane Coles and Andrew Coles. LPRPG-P: Relaxed plan heuristics for planning with preferences. In *ICAPS*, 2011.

[Coles *et al.*, 2010] Amanda Jane Coles, Andrew Coles, Maria Fox, and Derek Long. Forward-chaining partial-order planning. In *ICAPS*, pages 42–49, 2010.

[Coles *et al.*, 2012] Amanda Jane Coles, Andrew I Coles, Maria Fox, and Derek Long. Colin: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research*, 44:1–96, 2012.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[Francès and Geffner, 2015] Guillem Francès and Hector Geffner. Modeling and computation in planning: Better heuristics from more expressive languages. In *Proc. of the Conference on Automated Planning and Scheduling, (ICAPS 2015)*, pages 70–78, 2015.

[Gordon, 1998] Russell A Gordon. The use of tagged partitions in elementary real analysis. *The American mathematical monthly*, 105(2):107–117, 1998.

[Helmert *et al.*, 2007] Malte Helmert, Patrik Haslum, Jörg Hoffmann, et al. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, pages 176–183, 2007.

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

[Hoffmann, 2003] Jörg Hoffmann. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research (JAIR)*, 20:291–341, 2003.

[Illanes and McIlraith, 2017] León Illanes and Sheila A McIlraith. Numeric planning via abstraction and policy guided search. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 4338–4345, 2017.

[Kiam and Schulte, 2017] Jane Jean Kiam and Axel Schulte. Multilateral quality mission planning for solar-powered long-endurance UAV. In *Aerospace Conference, 2017 IEEE*, pages 1–10. IEEE, 2017.

[Piacentini *et al.*, 2018] Chiara Piacentini, Margarita P. Castro, Andre A. Cire, and J. Christopher Beck. Linear and integer programming-based heuristics for cost-optimal numeric planning. In *to appear in proceedings of AAAI-18*, 2018.

[Piotrowski *et al.*, 2016] Wiktor Mateusz Piotrowski, Maria Fox, Derek Long, Daniele Magazzeni, and Fabio Mercorio. Heuristic planning for hybrid systems. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 4254–4255, 2016.

[Röger *et al.*, 2014] Gabriele Röger, Florian Pommerening, and Malte Helmert. Optimal planning in the presence of conditional effects: Extending LM-cut with context splitting. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, pages 765–770. IOS Press, 2014.

[Scala *et al.*, 2016a] Enrico Scala, Patrik Haslum, and Sylvie Thiébaux. Heuristics for numeric planning via subgoaling. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3228–3234, 2016.

[Scala *et al.*, 2016b] Enrico Scala, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramírez. Interval-based relaxation for general numeric planning. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, pages 655–663, 2016.

[Vallati *et al.*, 2016] Mauro Vallati, Daniele Magazzeni, Bart De Schutter, Lukás Chrpa, and Thomas Leo McCluskey. Efficient macroscopic urban traffic models for reducing congestion: A PDDL+ planning approach. In *AAAI*, pages 3188–3194, 2016.