Journal of Artificial Intelligence Research 68 (2020) 691-752 Submitted 12/2019; published 08/2020

Subgoaling Techniques for Satisficing and Optimal Numeric Planning

Enrico Scala ENRICOS83@GMAIL.COM
Universitá di Brescia (Italy)
The Australian National University (Australia)

Patrik Haslum PATRIK.HASLUM@ANU.EDU.AU
Sylvie Thiébaux SYLVIE.THIEBAUX@ANU.EDU.AU
The Australian National University (Australia)

Miquel Ramirez MIQUEL.RAMIREZ@UNIMELB.EDU.AU
University of Melbourne (Australia)

Abstract

This paper studies novel subgoaling relaxations for automated planning with propositional and numeric state variables. Subgoaling relaxations address one source of complexity of the planning problem: the requirement to satisfy conditions simultaneously. The core idea is to relax this requirement by recursively decomposing conditions into atomic subgoals that are considered in isolation. Such relaxations are typically used for pruning, or as the basis for computing admissible or inadmissible heuristic estimates to guide optimal or satisficing heuristic search planners. In the last decade or so, the subgoaling principle has underpinned the design of an abundance of relaxation-based heuristics whose formulations have greatly extended the reach of classical planning. This paper extends subgoaling relaxations to support numeric state variables and numeric conditions. We provide both theoretical and practical results, with the aim of reaching a good trade-off between accuracy and computation costs within a heuristic state-space search planner. Our experimental results validate the theoretical assumptions, and indicate that subgoaling substantially improves on the state of the art in optimal and satisficing numeric planning via forward state-space search.

1. Introduction

Automated planning is the field of Artificial Intelligence that studies the development of intelligent agents capable of choosing and organising a sequence of actions to achieve a given set of objectives, ideally at minimal execution cost. Automated planning is model based; the planning model describes the conditions under which each action can be applied, and the effects such an action has on the world if executed. Given a description of the initial state and the goals that need to be achieved, a typical planning question asks whether there exists a plan that is executable and brings the agent to a state where the goal is satisfied.

Over the past decades, research has shown that the high worst-case computational complexity of classical (propositional) planning (Bylander, 1994) can be effectively managed, by exploiting the structure of the problem, as captured by its representation (Geffner, 2011). The most prominent approach aimed at exploiting such a structure in a systematic way is to *relax* the target problem into an easier problem to solve and use the relaxed solution cost as a heuristic (Bonet & Geffner, 2001; Geffner, 2018; Pearl, 1984).

In the last two decades, several relaxations for propositional representations have been proposed and explored (e.g., Bonet & Geffner, 2001; Haslum & Geffner, 2000; Helmert & Geffner, 2008; Helmert & Domshlak, 2009; Pommerening, Röger, Helmert, & Bonet, 2015; Domshlak, Hoffmann, & Katz, 2015), leading to planners that scale to instances of realistic size. Less has been done to address more expressive problem classes, such as planning with numeric information (Fox & Long, 2003). Since planning agents often have to deal with limited resources or geometrical constraints, the ability to deal with this information is essential to address a wider class of real-world problems.

Numeric state variables were introduced into the de-facto standard planning language (PDDL) through the notion of numeric fluents (Fox & Long, 2003). A numeric fluent is a function mapping a tuple of objects to a rational number. Numeric fluents are a convenient device to model quantitative properties of object(s), such as their position, their distance, their cost. Our main intuition is that, even though the introduction of numeric variables complicates planning substantially (numeric planning is undecidable in general (Helmert, 2002) while classical planning is "only" PSPACE-complete), numeric variables and numeric conditions allow for more compact descriptions of certain planning problems, which, if exploited properly, can lead to more powerful reasoning. For instance, imagine we wish to encode a domain involving a state variable x that models the position of the agent through time. Let us assume that x ranges from 0 to 10 and we only wish to model its integer representation. In order to encode such knowledge using propositional variables we need to enumerate each of its possible values from 0 to 10, so having a total of 11 propositional variables $\{P_0, P_1, ..., P_{10}\}$. This not only causes a rapid explosion of the number of variables to consider, but also makes them *orthogonal* and incomparable. In particular, in a purely propositional representation, we would need to introduce further propositions such as $\{less_{P_1P_2}, less_{P_1P_3}, ..., less_{P_8P_9}\}$ to re-axiomatize the simple ordering relation; therefore, any kind of reasoning that involves such information would require to rebuild that structure. It is therefore only natural to study how to capture numeric aspects directly, possibly using some kind of relaxation and heuristic approach. Our aim in this paper is therefore to focus on the reasoning and algorithmic aspects of planning languages supporting a numeric representation, so as to provide corresponding planners with novel relaxation-based heuristics capable of delivering the same benefits that they have achieved in classical planning. Although, in principle, we could exclude propositional variables and conditions by compiling them into suitable numeric variables and numeric conditions, our approach will still consider propositional structures as such, alongside numeric ones. This is because, when certain aspects of the problem at hand are more conveniently modelled using purely qualitative conditions, compilation into (infinite) numeric variables obscures their underlying structure, and reconstructing it requires additional reasoning effort.

Other researchers have investigated the solution of numeric planning problems through relaxation-based heuristic search before (e.g., Hoffmann, 2003; Coles, Coles, Fox, & Long, 2012, 2013). However, in all of these approaches, the well-known interval-based relaxation (Hoffmann, 2003) underlies the heuristic. An exception is the work by Eyerich (2009), which is founded on the context-enhanced additive heuristic (Helmert & Geffner, 2008). In contrast, we take inspiration from the h^1 relaxation presented for classical planning (Haslum

& Geffner, 2000), and use its underlying decomposition principle to capture the numeric structure of the problem. Rather than looking at sets of values that numeric variables may attain (as in the interval-based relaxation), we start from the numeric subgoals, meaning elementary numeric conditions in formulae representing the goal and action preconditions, that need to be reached, and consider the operators that affect them. Differently from classical planning where atomic subgoals are propositional literals, numeric subgoals can constrain sets of numeric state variables together; therefore even classical notions such as that of an achiever (i.e. an action that contributes positively towards achieving a subgoal) need to be revised.

The techniques we investigate support mixed propositional–numeric reasoning: where propositional and numeric variables co-occur in the definition of the problem, the justification for including actions in a relaxed solution may derive from both "quantitative" (numeric) and "qualitative" (propositional) subgoals. Numeric conditions in particular may justify the presence of some action, which takes the role of a *possible achiever*. How to define and compute such possible achievers is not obvious, and we present a method to do so using a closed-form arithmetic computation that is obtained in a number of steps, including a Mixed Integer Program formulation and its continuous relaxation. This formulation applies to what we call *simple* numeric conditions. Through possible achievers, we show how to devise a variety of heuristics for optimal and satisficing planning. Interestingly, even the most basic one proves superior to previous interval-based relaxations. Moving from atomic subgoals to conjunctive subgoals, we also provide an extended version of possible achievers that we call *conjunctive achievers*. Conjunctive achievers account for the simultaneous achievements of sets of subgoals, including mixed propositional and numeric conditions. They can be computed using a novel Mixed Integer Program formulation. The paper shows how to use this characterisation to get a novel lower bound that can be used as an admissible heuristic.

We study these relaxations and heuristics both from a theoretical and practical viewpoint. Our theoretical analysis shows that the computational issues arising from the formulation can be tamed without loosing too much information nor compromising admissibility. From a practical perspective, our results over a number of numeric planning problems, taken from previous work on numeric planning and the from International Planning Competition suite, indicate that the presented heuristics often translate to great guidance and substantially extend the reach of forward search in numeric planning.

1.1 Contribution

This work extends our previous conference paper on the topic (Scala, Haslum, & Thiébaux, 2016a) in a number of respects:

- We provide a more in-depth analysis of the subgoaling theoretical framework. Not only proofs are extended, but also new theorems are reported to better analyse the complexities and the benefits of the new formulations.

- We report a novel, revised admissible estimate, which accounts for a limited form of assignment operations and is more informed because it considers a narrow set of reachable actions.

- We provide a novel relaxation schema based on the idea of subgoaling. Such a relaxation makes heavy use of a Linear Program formulation to capture conjunctive reasoning. This leads to an expensive but quite informed admissible estimate that is evaluated on optimal planning tasks.
- We conduct an extended experimental analysis, with more insights and discussions encompassing the implications of the novel conjunctive relaxation, with comparisons not only against numeric planning, but also against classical planning reformulations of a bounded numeric planning problem.
- We extend our related work analysis, and consider in greater depth previous work on heuristic search, other approaches to numeric planning via Satisfiability Modulo Theory, and links to other research fields such as that of linear time systems.

1.2 Outline

In Section 2, we start by describing the language we use to model numeric planning problems. Then our discussion proceeds through the following steps. First, in Section 3, we provide the first formulation of a numeric h^1 relaxation, and the computational consequences of a direct encoding via standard regression. In Section 4, we explain how this can be made computable and then tractable in the case of what we call *simple numeric conditions*, and in Section 5, we use this novel formulation to obtain admissible and inadmissible estimates. In Section 6 we compare the novel relaxation with the well-known interval-based relaxation in terms of reachability accuracy. In Section 7, we describe a transformation aimed at tightening the presented relaxation through redundant constraints. In Sections 8 and 9, we then present a more systematic approach to tightening the relaxation using a much more sophisticated (yet expensive) Linear Program based schema. The novel methodological contribution of the paper ends with Sections 10 and 11, which show how to exploit decomposition to also consider more general numeric constructs in the subgoaling schema; this requires a slight extension of the notion of possible achiever. Sections 12 and 13 report on related work and on the experimental evaluation of our contributions, respectively.

2. Numeric Planning Problems

We are concerned with sequential numeric planning problems described in PDDL2.1 – level 2; see (Fox & Long, 2003; Haslum, Lipovetzky, Magazzeni, & Muise, 2019) for a more detailed presentation of the language and modelling examples.

2.1 Syntax

We work with the grounded problem representation, where a numeric planning problem is described as a tuple $\Pi \doteq \langle F, X, I, G, C, A, \gamma \rangle$[1] consisting of the following elements:

F is a set of Boolean variables (the *propositional variables*). X is a set of rational-valued variables (the *numeric variables*). A *state* is a total mapping from F to $\{\top, \bot\}$ and from

1. We will in the remaining sections use the symbol '$\doteq$' to differentiate the equality by definition with the standard equality '$=$'.

X to $\mathbb{Q} \cup \{\bot\}$ [2] Propositional variables assigned to $\bot$ in a state are false in that state whilst numeric variables assigned to $\bot$ are referred to as *undefined.* I is the initial state. G describes the *goal* that the final state of any solution plan must satisfy, and C the *global state constraints* that must be satisfied by all states reached at any step of a valid plan. Both G and C are formally represented as *formulae* in Negation Normal Form (NNF), inductively defined as follows:

(i) a *propositional condition* $v = \top$, such that $v \in F$, is a (atomic) formula;

(ii) a *numeric condition* $\langle \xi, \trianglerighteq, 0\rangle$, where ξ is an arithmetical expression over X and $\trianglerighteq \in \{>, \geq, =\}$ is a comparison operator, is a (atomic) formula;

(iii) if ψ is a formula as defined by items (i) or (ii) above, $\neg\psi$ is a formula;

(iv) if ψ and ψ' are formulae, so are $\psi \wedge \psi'$ and $\psi \vee \psi'$.

We use Greek letters (generally ψ) to indicate formulae, whether atomic or compound. When no confusion arises, we use the term literal in place of a propositional condition or its negation, and use v and $\neg v$ as a shorthand for $v = \top$ and $v = \bot$, respectively. Moreover, for convenience, we write $c \in \psi$ when the formula ψ is either a conjunction or a disjunction, to indicate that the sub-formula c appears as a conjunct/disjunct of ψ.

A is the set of actions and γ is a function that associates each action with a non-negative rational *cost.* Each *action* $a \in A$ is described by a tuple $\langle \text{name}(a), \text{pre}(a), \text{eff}(a)\rangle$,[3] where name$(a)$ is the (unique) name of the action, pre(a) is the formula (as defined above) representing the *precondition* that must be true of the state for the action to be executable, and eff(a) is a set of assignments over F and X representing the *effects* of the action in the resulting state. *Assignments* are defined as follows: an assignment to a propositional variable $v \in F$ is of the form $v{:=}\top$ or $v{:=}\bot$; an assignment of a numeric variable $x \in X$ is of the form $\langle x, op(e), \xi\rangle$ where $x \in X$ is the affected variable, ξ is a numeric expression, and $op(e)$ is one of $+=$ (increase), $-=$ (decrease) or $:=$ (assign).[4] Given an effect e, we use $lhs(e)$ to refer to the affected variable and $rhs(e)$ to the expression used to update that variable. We disallow the occurrence of conflicting effects, i.e. of two effects $\{e, e'\} \subseteq \text{eff}(a)$ with $e \neq e'$ and $lhs(e) = lhs(e')$.[5] In the following, we use subscripts to distinguish the

2. Other works (Fox & Long, 2003; Scala, Haslum, Thiébaux, & Ramírez, 2016b; Cashmore, Magazzeni, & Zehtabi, 2020) consider numeric variables that take real values. Note that, even though some conditions can only be satisfied by real values (e.g. $x * x = 2$), our formulation and PDDL2.1 preclude reaching states satisfying such conditions: PDDL2.1 does not allow initial states with non-rational values (initial state assignments must have final decimal expansions) nor does it allow writing action effects that will yield non-rational values in a finite number of steps. The situation would be different if we allowed non-rational value assignments in action effects, such as $x{:=}\sqrt{2}$.

3. Later in the text we will often use actions as simple pair of precondition and effects when the name of the action is not important.

4. Note that, in principle, an increase or decrease operation can be defined in terms of assignment, e.g., $x{+}{=}\ \xi$ is equivalent to $x{:=}x + \xi$. However, this kind of transformation hides some structure, namely additivity, which we exploit to a great extent in this paper.

5. PDDL allows actions to have conflicting propositional effects, taking an interpretation called Delete Before Adding (DBA) where the *positive* statement ($v{:=}\top$) prevails over the negative one ($v{:=}\bot$). We disallow the occurrence of such conflicting effects. Note that PDDL2.1 does not allow actions with conflicting effects on numeric variables.

propositional and numeric parts of action effects, writing $\text{eff}_{\text{prop}}(a)$ for the propositional effects and $\text{eff}_{\text{num}}(a)$ for the numeric effects of a.

2.2 Semantics

The semantics of numeric planning is defined as follows. We write $[v]^s$ for the value of variable $v \in F \cup X$ in state s and $[\xi]^s$ for the value of numeric expression ξ in state s. Any numeric expression containing an undefined numeric variable is undefined. A numeric expression of the form ξ/o is also undefined in s if $[o]^s = 0$, i.e., division by zero is undefined. For a numeric condition, $s \models \langle \xi, \trianglerighteq, 0\rangle$ iff $[\xi]^s \trianglerighteq 0$ evaluates to true. A numeric condition, and the negation of a numeric condition, containing a numeric expression that is undefined in state s is not satisfied in s. Given these building blocks, the evaluation of the rest of the language describing formulae follows the usual semantics of propositional logic.

An action a is applicable in state s iff $s \models \text{pre}(a)$ and all numeric expressions in $\text{eff}(a)$ are defined in s. The state $s' = s[a]$ resulting from applying an (applicable) action a in s is defined as follows. For all $v \in F \cup X$:

- $[v]^{s'} = rhs(e)$ if $\exists e \in \text{eff}_{\text{prop}}(a)$ with $lhs(e) = v$ (Propositional Effect)
- $[v]^{s'} \circ\!= [rhs(e)]^s$ if $\exists e \in \text{eff}_{\text{num}}(a)$ with $lhs(e) = v$ and $op(e) = \circ\!=$ (Numeric Effect)
- $[v]^{s'} = [v]^s$ otherwise (Frame Axiom)

A plan is a finite sequence of actions $\langle a_0, a_1, ..., a_{n-1}\rangle$ such that for all $i \in \{0, \dots, n-1\}$ a_i is applicable in s_i, where $s_0 = I$ and $s_{i+1} = s_i[a_i]$. A valid plan is one that satisfies the global constraints C throughout its execution, i.e., $\forall i \in \{0, \dots, n\} s_i \models C$; a solution plan is a valid plan that also reaches the goal, i.e., such that $s_n \models G$. We say that a state s is reachable if there exists some valid plan that leads from the initial state to s. We say that a solution plan π is optimal if it minimises the overall cost $\sum_{i=0}^{|\pi|-1} \gamma(a_i)$.

2.3 Small Example

To illustrate these definitions and as a running example for the rest of the paper, let us consider a very simple Sailing navigation domain, which is however representative of a variety of domains featuring multi-variable numeric conditions.

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 x-y coordinates in $\mathbb{Q}^2$. Note that these coordinates are not bounded a priori, making a propositional representation impossible. The boat can sail in 7 possible directions $Dir \doteq \{\text{NE}, \text{E}, \text{SE}, \text{S}, \text{SW}, \text{W}, \text{NW}\}$ (see Figure 1; the boat cannot sail straight into the wind). To save a person, the boat must reach an area described by one or more linear inequalities. Once the boat reaches any point within this area, the boat can rescue the castaway.

A numeric planning formulation of the problem uses two numeric variables $X \doteq \{x, y\}$ to model the position of the sailing boat and actions $A \doteq \{a_d \mid d \in Dir\}$ for each of the 7 possible movements of the boat. These actions have an empty precondition, and numeric effects defined as a function of the direction that the action is modelling, that is:

$$\text{eff}(a_d) \doteq \begin{cases} \langle x, :=, x + k_{d,x}\rangle \\ \langle y, :=, y + k_{d,y}\rangle \end{cases}$$

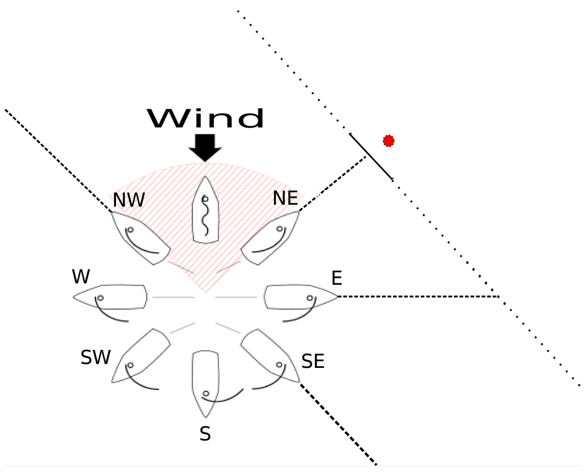

Figure 1: Points of sail (https://en.wikipedia.org). There are 7 possible directions for the boat to move with respect to a stable wind. The goal is to cross the dotted line to the side indicated by the red dot.

where $k_{d,x}, k_{d,y}$ are constants associated with moving a certain distance in the particular direction (d). Actions therefore model translations in $\mathbb{Q}^2$.

The initial state is an assignment to the numeric variables x, y, say $\langle x = 1, y = 1\rangle$, while the goal is defined as a target region, specified by a conjunction of linear constraints. For instance, the goal region depicted in Figure 1 may be defined by $G \doteq \langle x + y, \geq, 10\rangle$. If we assume that the translation coefficients of all actions ($k_{d,x}$ and $k_{d,y}$ for all d) are equal to 1, an optimal plan for this problem is to apply action a_{NE} five times.

The other structures of our numeric planning problem can be left empty for the task of modelling the movement of the sailboat. The modelling of multiple rescues in different regions $O \doteq r_0, \cdots, r_m$ can be done by introducing an action a_i for each such region, whose precondition is the numeric constraints defining the area of interest, i.e., $\text{pre}(a_i) \doteq \bigwedge_{c \in r_i} \langle \xi_c, \geq, 0\rangle$, and whose effect records the fact that the rescue in r_i has been accomplished, i.e., $\text{eff}_{\text{prop}}(a_i) \doteq (\text{rescued } r_i)$. The goal is then the conjunction of rescued propositions for all regions in O, that is $G \doteq \bigwedge_{r \in O}(\text{rescued r})$.

Throughout this paper, we will investigate the structure of the computational problem made explicit by the language of numeric planning just introduced in order to study conditions for the development of informed but tractable heuristics. Analogously to the propositional case, our analysis focuses on determining why certain actions are required, but also, additionally here in the numeric case, counting how many times they need to be executed. Later, we explain how the techniques we develop for a restricted class of problems combines with more general cases, and how these can be used in well known search algorithms.

3. From Classical to Numeric Subgoaling Relaxation

We start our discussion with a generalised formulation of the h^{max} and h^{add} heuristics (Bonet & Geffner, 2001) for classical planning; in this formulation, action preconditions and goals are not restricted to sets of propositions, but can instead be arbitrary formulae in Negation Normal Form (NNF). Following this, we extend this reformulation to account for numeric information.

3.1 h^{max} for NNF formulae in Classical Planning

h^{max} can be extended to handle NNF subgoals directly, without rewriting by splitting actions (in the case of disjunctions) or introducing artificial propositions to capture negative literals via positive ones. Let ψ be a propositional condition, that is, either v or $\neg v$ for some $v \in F$: $ach(s, \psi)$ denotes the set of *achievers of* ψ *in state* s. In the classical case, these are simply the actions that make ψ true, i.e., that add the variable if the literal is positive or delete it if the literal is negative. This is independent of the state s. However, because numeric effects can be state-dependent, the set of achievers of a numeric condition – defined below – can depend on s, and for this reason we incorporate the state into $ach(s, \psi)$ to make the similarity between the two settings clearer. The h^{max} estimate for an arbitrary NNF ψ in s can now be defined as follows: [6]

$$h^{max}(s,\psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(s,\psi)} (h^{max}(s, \text{pre}(a)) + \gamma(a)) & \text{if } \psi \text{ is a literal} \\ \min\limits_{\psi' \in \psi} (h^{max}(s,\psi')) & \text{if } \psi \text{ is } \vee \\ \max\limits_{\psi' \in \psi} (h^{max}(s,\psi')) & \text{if } \psi \text{ is } \wedge \end{cases} \tag{1}$$

Besides the base case which gives an estimate of zero when ψ is already satisfied in s, the reasoning implemented by Eq. 1 depends on whether ψ is a simple literal or a more complex conjunction or disjunction. In the former case, Eq. 1 recursively calls itself against the regression of ψ through each $a \in ach(s, \psi)$, and as for the original h^{max}, selects the option that leads to minimising the cost of executing the action plus the estimated cost (via h^{max}) of achieving its precondition. When instead the condition is complex, h^{max} calls itself recursively on each subgoal of ψ in isolation and either minimises or maximises over the results, depending on whether ψ was a disjunction or a conjunction.

3.1.1 Properties

It is well known that the h^{max} formulation induces a relaxation of the problem being solved. Such a relaxation, which is known as the *subgoaling relaxation* (Haslum, Bonet, & Geffner, 2005), is based on the following decomposition principle: *given a conjunction or a disjunction of literals* ψ*, the achievement of a literal* $l \in \psi$ *is considered as completely independent from the achievement of any other literal* $l' \in \psi$ *such that* $l' \neq l$. Because this is indeed a proper relaxation, it is safely pruning, meaning that the relaxed problem has no solution only if the original planning problem has no solution. This also means

6. Rintanen (2006) also extends h^{max} to arbitrary preconditions and, additionally, to conditional effects. Our formulation and motivation are different: we build directly on the original recursive formulation proposed by Bonet and Geffner (2001) with the aim of focusing on the underlying decomposition power of this framework and transferring it to the numeric context.

that heuristics based on the relaxation, such as h^{max} and h^{add}, have infinite values only for goals that are unreachable from the given state. Moreover, it is known that h^{max}, which estimates the cost of a conjunction by that of the most costly subgoal, is an admissible heuristic, meaning that $h^{max}(s, \psi) \leq h^{*}(s, \psi)$, where $h^{*}(s, \psi)$ denotes the actual optimal cost of achieving ψ from s, for any s and ψ.

3.1.2 Computational Aspects

The solution to Eq. 1 can be computed in time polynomial in the number of conditions and actions using an iterative procedure that updates the cost estimate associated with each formula ψ involved in the problem, until a fix-point is reached (this can be seen easily extending (Haslum & Geffner, 2000)). Such a fix-point is guaranteed to exist by the simple observation that each action needs to be considered at most once, when the estimated cost of its precondition becomes known. A way to do this is by using a label-correcting algorithm (for example, the Bellman-Ford Algorithm (Bellman, 1958)), or, more efficiently since there are no negative costs, organising the updating process using a priority queue in a Uniform Cost Search (UCS). The priority queue keeps track of the cheapest condition and the causal chain of actions needed to achieve it, following the update rules given by Eq. 1. The procedure terminates when either the goal condition is popped from the priority queue or when the priority queue becomes empty. An obvious benefit of UCS over a pure label correcting algorithm is that it restricts the iterations to those conditions belonging to actions that are relaxed reachable and relevant for the estimation of the final optimal cost associated with the goal. Because UCS guarantees optimality on expansion, the cost associated to the goal is guaranteed to be the minimal one when the goal is popped out of the queue.

3.2 h^{max}_{hbd} and h^{add}_{hbd}: h^{max} and h^{add} with numeric conditions

The recursion $\min_{a \in ach(s,\psi)} h^{max}(s, \text{pre}(a)) + \gamma(a)$ in the definition of h^{max} can be explained as the result of regressing the condition ψ through possible actions and taking the least cost of the resulting formulae (Haslum et al., 2005). The restriction of the minimisation to achievers only simply excludes regression through actions that have either no or a detrimental effect on ψ and therefore will never attain the minimum. Thus, the first step towards extending Eq. 1 to planning with numeric information is a definition of regression for numeric conditions and action effects. Numeric regression can be formulated as the computation of the so called weakest precondition, and relies on a simple manipulation of the expressions involved in the numeric condition against the effects of the action (see (Scala, 2013) or more generally (Hoare, 1969)). Let $\psi \doteq \langle \xi, \trianglerighteq, 0 \rangle$ be a numeric condition and a an action; the regression of ψ through a is computed by substitution ('[]') of the affected variables:

Definition 1 (Effect Regressor). *Given a numeric condition $\psi \doteq \langle \xi, \trianglerighteq, 0 \rangle$ and an action a, the effect regressor $\psi^{r(a)}$ transforms ψ into $\langle \xi[x_1/\tau(a, x_1), \ldots, x_k/\tau(a, x_k)], \trianglerighteq, 0 \rangle$ where*

$$\tau(a, x_i) \doteq \begin{cases} \xi' & \textit{if } \exists \langle x_i, :=, \xi' \rangle \in \text{eff}_{\text{num}}(a) \\ x_i + \xi' & \textit{if } \exists \langle x_i, +=, \xi' \rangle \in \text{eff}_{\text{num}}(a) \\ x_i - \xi' & \textit{if } \exists \langle x_i, -=, \xi' \rangle \in \text{eff}_{\text{num}}(a) \\ x_i & \textit{Otherwise} \end{cases}$$

and $x_1, \dots, x_k$ are the numeric variables affected by a. The substitution (denoted by '/' symbol) of $x_1, \dots, x_k$ in $\psi^{r(a)}$ is simultaneous. Variables among $x_1, \dots, x_k$, indeed even x_i, can appear in the right-hand side of the effect of a on x_i, and in this case will appear in $\tau(a, x_i)$, but these occurrences are not replaced.

$\psi^{r(a)}$ represents the necessary and sufficient condition for ψ to be true in the state resulting from applying the effects of a. To complete the notion of regression requires only that this state is defined, i.e., that the actions precondition is also true:

Proposition 3.1 (Numeric Regression)**.** *Let a be an action and ψ a numeric condition. $s[a] \models \psi$ iff $s \models \psi^{r(a)} \land \text{pre}(a)$. $\psi^{r(a)} \land \text{pre}(a)$ denotes the regression of ψ through a.*

Proof. (Sufficiency). Because $s \models \text{pre}(a)$, the successor state $s[a]$ is well defined. $s[a]$ is by definition created by substituting for each variable affected by a the right hand-side of the effect modifying it, if any – remember here that we cannot have conflicting effects, i.e. there is at most one effect affecting each variable. $\psi^{r(a)}$ is by definition constructed by substitution: it contains, for each affected variable, the right-hand side of the effect of a on that variable. Now, because $s \models \psi^{r(a)}$ we can substitute the variable modified by a on both sides yielding $s[a] \models \psi$.
(Necessity). Because $s[a] \models \psi$, $s[a]$ must be a well defined state which implies that $s \models \text{pre}(a)$. That $s \models \psi^{r(a)}$ follows from $s[a] \models \psi$ follows an argument analogous to that used for the sufficiency part. □

Having defined regression for numeric conditions, the introduction of numeric information into the subgoaling relaxation becomes straightforward: it suffices to distinguish propositional conditions from numeric conditions in the recurrence relation of Eq. 1, and treat each of them with the proper regression. Eq. 2 and Eq. 3 formalise this hybrid case for the admissible (analogous of h^{max}) and inadmissible and additive (analogous of h^{add}) settings. NCs stands for the set of numeric conditions and PCs for the set of propositional conditions.

$$h^{max}_{hbd}(s, \psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(s,\psi)} (h^{max}_{hbd}(s, \text{pre}(a)) + \gamma(a)) & \text{if } \psi \in \mathsf{PC}s \\ \min\limits_{a \in A} (h^{max}_{hbd}(s, \text{pre}(a) \land \psi^{r(a)}) + \gamma(a)) & \text{if } \psi \in \mathsf{NC}s \\ \min\limits_{\psi' \in \psi} h^{max}_{hbd}(s, \psi') & \text{if } \psi \text{ is } \lor \\ \max\limits_{\psi' \in \psi} h^{max}_{hbd}(s, \psi') & \text{if } \psi \text{ is } \land \end{cases} \tag{2}$$

$$h^{add}_{hbd}(s, \psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(s,\psi)} \left(h^{add}_{hbd}(s, \text{pre}(a)) + \gamma(a)\right) & \text{if } \psi \in \mathsf{PC}s \\ \min\limits_{a \in A} (h^{add}_{hbd}(s, \text{pre}(a) \land \psi^{r(a)}) + \gamma(a)) & \text{if } \psi \in \mathsf{NC}s \\ \min\limits_{\psi' \in \psi} h^{add}_{hbd}(s, \psi') & \text{if } \psi \text{ is } \lor \\ \sum\limits_{\psi' \in \psi} h^{add}_{hbd}(s, \psi') & \text{if } \psi \text{ is } \land \end{cases} \tag{3}$$

Unfortunately, the introduction of numeric conditions makes the recurrence relation much more complicated from a computational standpoint. While there is an obvious fix-point for tasks involving only propositional variables, the existence of such a fix-point is not clear in the numeric setting. This is due to the following facts:

First, numeric action effects may depend on the state in which the action is applied, which complicates the definition of which actions are achievers of a numeric condition. In general, we cannot look at actions in isolation to determine whether they contribute to achieving a numeric condition. For example, consider the question whether the condition $\psi \doteq \langle x-10, \geq, 0\rangle$ can be achieved from from state $s \doteq \{x = 0, y = 0\}$ using actions $\langle a_0, \emptyset, y \mathrel{+}= 10\rangle$ and $\langle a_1, \emptyset, x \mathrel{+}= y\rangle$. It is obvious that neither action, when applied in s, can make ψ satisfied; indeed neither action changes the value of x. Yet the condition is achieved by the sequence $\langle a_0, a_1\rangle$ of both actions.

In certain numeric planning domains, such as our Sailing domain for example, this situation does not arise. Even though the numeric effects of actions are still state-dependent, in the sense that the value of a numeric variable after the application of an action's effects can depend on the value it had before, whether the action makes a positive or negative contribution with respect to the numeric conditions they are affecting is not. Considering again the numeric condition $x \geq 10$, it is clear that an action effect of the form $\langle x, \mathrel{+}=, 1\rangle$ makes a positive contribution towards achieving the condition, regardless of the current state, as long as it is not already satisfied. This is the type of property that we will exploit in the next section.

In difference to the propositional case, the regression of a numeric condition through an action is not limited to the action's precondition only, but carries the remaining part of the regressed condition into the recursive evaluation of h^{max}_{hbd} (and h^{add}_{hbd}). For example, regressing $\psi \doteq \langle x-10, \geq, 0\rangle$ through the effect $\langle x, \mathrel{+}=, 1\rangle$ results in $\psi^{r(a)} = \langle (x+1)-10, \geq, 0\rangle$, and thus considering an action a with this effect as a possible achiever in h^{max}_{hbd} leads to the recursive estimation $h^{max}_{hbd}(s, \text{pre}(a) \wedge \langle (x+1) - 10, \geq, 0\rangle)$. In contrast, in the propositional case, the regression of a single literal through an action achieving it results in considering only the action precondition. This also means that any negative effect that the action may have on a conjunction that the literal is part of is ignored – that is the subgoaling relaxation – which is not always true in the numeric case.

The main implication of these two observations is computational; this is not completely surprising, since numeric planning is undecidable even with planning problems involving just one goal (Helmert, 2002). To overcome this computational barrier, we will first analyse regression of linear conditions and effects in greater depth, and then focus our attention on a further restricted class of numeric planning problems.

4. Regression and Action Repetition for Simple Numeric Conditions

Unlike in classical planning, actions with numeric effects are not necessarily idempotent, meaning that the repeated application of an action in a state can, in general, result in a state different from that reached by a single application of the action.[7] Dually, this is also true of numeric regression: that $s \not\models \psi^{r(a)}$ does not imply that $s \not\models \psi^{r(a)^{r(a)}}$, or indeed

7. Though note that the same is true of actions with *conditional effects* in classical planning.

that $s \not\models \psi^{\overbrace{r(a)^{\cdots r(a)}}^{\text{m times}}}$ for some m. This is particularly relevant since many numeric planning problems feature accumulation of resources or translation in geometrical spaces (e.g., the Sailing domain seen in Section 1). While the current state may not satisfy the result of a regressing a condition once, it is possible that it will eventually satisfy the condition after a number of action *repetitions*.

Restricting the form of action effects and their relations to the conditions they are interacting with allows us to formulate repeated regression in closed form. In this section we present the relevant restriction, and conclude with a notion of *possible achiever* that exploits the closed form to infer necessary conditions for the reachability of a numeric condition.

4.1 Intuition

To start with a specific example, consider the fully instantiated action a_{NE} from the Sailing domain. Say a_{NE}'s precondition remains empty and that its effects are:

$$\text{eff}(a_{NE}) \doteq \begin{cases} \langle x, +=, 4\rangle \\ \langle y, +=, 3\rangle \end{cases}$$

Assume that the condition $\psi \doteq (x - y) - 3 \geq 0$ needs to be satisfied, and let us assess the conditions that must hold for achieving ψ via regression through a_{NE}. Regressing ψ through a_{NE} once gives us:

$$\psi^{r(a_{\text{NE}})} = ((x + 4) - (y + 3)) - 3 \geq 0$$

After a second regression we obtain:

$$\psi^{r(a_{\text{NE}})^{r(a_{\text{NE}})}} = ((x + 8) - (y + 6)) - 3 \geq 0 \tag{4}$$

For a more general example, consider an action $\langle a, \emptyset, x_k = \sum_{x\in X} w_{a,x_k,x}x + k_{a,x_k}\rangle$ and the numeric condition $\psi \doteq \langle \sum_{x\in X} w_{\psi,x}x + k_\psi, \trianglerighteq, 0\rangle$. Note that expressions in the action effect and the left-hand side of the condition are both linear. All coefficients $w_{(\cdot,\cdot,\cdot)}, w_{(\cdot,\cdot)}, k_{(\cdot,\cdot)}$ and $k_{(\cdot)}$ are rational constants and their subscripts refer to the objects the coefficients depend on. In the action effect coefficients $w_{a,x_k,x}$, the subscript includes the action a containing the effect, the variable x_k affected, and the variable x the coefficient relates to in the right-hand side of the effect. Similarly k_{a,x_k} denotes the constant contribution of the effect of a on x_k. For each condition coefficient $w_{\psi,x}$ the subscript indicates the condition ψ the coefficient relates to, and the variable the coefficient is associated with, and similarly k_ψ is the constant in the condition expression. Note that, in the following, we intend indexed summation ("$\sum$") to take precedence over simple addition ("+"); that is, when we write $\sum_{i=1}^n a_i + b$ it should be understood as $(\sum_{i=1}^n a_i) + b$. We explicitly parenthesise the summed term when it is itself a sum, as in $\sum_{i=1}^n (a_i + b)$.

Having defined precisely what these components are, we can see that regressing ψ through a equates to:

$$\psi^{r(a)} = \langle \sum_{x\in X\setminus\{x_k\}} w_{\psi,x}x + k_\psi + w_{\psi,x_k}\left(\sum_{x\in X} w_{a,x_k,x}x + k_{a,x_k}\right), \trianglerighteq, 0\rangle$$

and that if we regress the regressed condition $\psi^{r(a)}$ through the action again we obtain:

$$\begin{aligned}\psi^{r(a)^{r(a)}} = \langle & \sum_{x\in X\setminus\{x_k\}} w_{\psi,x}x + k_\psi \\ & + w_{\psi,x_k}\left[\sum_{x\in X\setminus\{x_k\}} w_{a,x_k,x}x + k_{a,x_k} + w_{a,x_k,x_k}\left(\sum_{x\in X} w_{a,x_k,x}x + k_{a,x_k}\right)\right], \trianglerighteq, 0\rangle\end{aligned}$$

which is equivalent to:

$$\begin{aligned}\psi^{r(a)^{r(a)}} = \langle & \sum_{x\in X\setminus\{x_k\}} w_{\psi,x}x + k_\psi \\ & + w_{\psi,x_k}\left[(1+w_{a,x_k,x_k})\left(\sum_{x\in X\setminus\{x_k\}} w_{a,x_k,x}x + k_{a,x_k}\right) + w^2_{a,x_k,x_k}x_k\right], \trianglerighteq, 0\rangle \qquad (5)\end{aligned}$$

The conditions obtained in equations 4 and 5 can be used to determine whether ψ is achieved by two repetitions of the same action (a_{NE} or a) from some state s. For instance, in the Sailing example, if ψ is the condition $(x-y)-3 \geq 0$ and s is the state $\langle x=0, y=0\rangle$, then $[\psi^{r(a_{\text{NE}})^{r(a_{\text{NE}})}}]^s$ evaluates to $-1 \geq 0$ which is false. If, instead, the state s is $\langle x=2, y=0\rangle$ it evaluates to $1 \geq 0$, meaning that applying a_{NE} twice in this state will satisfy ψ.

4.2 Possible Achievers of Numeric Conditions

The intuition underlying the above examples can be generalised to a variable number m of action repetitions. For the class of so called *linear and self-interference free* (LSF) actions, which we define now, we give a formula for the generalised repeated regression.

Definition 2 (Self-interfering effects)**.** *An action a is said to have* self-interfering *effects whenever* $\exists\, e,\ e' \in \text{eff}_{\text{num}}(a) : e \neq e'$ *and* $\text{lhs}(e)$ *occurs in* $\text{rhs}(e')$.

Definition 3 (Linear effects)**.** *An action has linear effects if* $\forall e \in \text{eff}_{\text{num}}(a)$*, e can be written as* $\langle x, :=, \sum_{y\in X} w_{a,x,y}y + k_{a,x}\rangle$*, with* $w_{a,x,y}, k_{a,x} \in \mathbb{Q}$*, where x is the variable modified by e.*

An action is said to be *LSF* (*Linear and Self-interference Free*) if it has linear effects and does not have self-interfering effects.

We are now in a position to define the *m-times effect regressor* as the closed form expression for the regression of a numeric condition through an indeterminate number of repetitions of the effects of an *LSF* action:

Definition 4 (*m-times effect regressor*)**.** *Given a linear numeric condition* $\psi \doteq \sum_{x\in X} w_{\psi,x}x + k_\psi \trianglerighteq 0$*, where* $w_{\psi,x}, k_\psi \in \mathbb{Q}$*, and an LSF action a, the* m-times effect regressor *for a and ψ, denoted by* $\psi^{r(a,m)}$ *is*

$$\psi^{r(a,m)} \doteq \left(\sum_{x\in X} w_{\psi,x}(f_{x,a}(m)) + k_\psi\right) \trianglerighteq 0$$

where

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

is the value of x after applying action a m times.

Note that m appears in the exponent of the last term of $f_{x,a}(m)$; thus, the function is non-linear if $w_{a,x,x}$ is different from 0 and 1.

Theorem 4.1 (Correctness of *m-times effect regressor*)**.** *The* m-times effect regressor *represents m repetitions of the* effect regressor *. That is, for all states s:*

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

Proof. To prove the theorem it is sufficient to show that the two conditions are equivalent, i.e., that $\psi^{\overbrace{r(a)^{\cdots^{r(a)}}}^{\text{m times}}} \equiv \psi^{r(a,m)}$. The proof is by induction on m. For the base case ($m = 1$), note that $\psi^{r(a,1)}$ is the result of substituting $f_{x,a}(1)$ for x in ψ, and that $f_{x,a}(1)$ is simply the right-hand side of the effect of a on x. Thus, $\psi^{r(a,1)} \equiv \psi^{r(a)}$. For the inductive case let us assume that the equivalence holds for some $m = z - 1$ with $z \geq 2$; by inductive hypothesis we have that $\psi^{r(a,z-1)} \equiv \psi^{\overbrace{r(a)^{\cdots^{r(a)}}}^{\text{(z-1)times}}}$. The proof reduces to showing that $\psi^{r(a,z-1)^{r(a)}} \equiv \psi^{r(a,z)}$. Since the right-hand side and comparison of the condition is not changed by regression, we only need to focus on the expression on left-hand side. Consider $f_{x,a}(z) = \overbrace{\sum_{k=0}^{z-1} w_{a,x,x}^{k} \left(\sum_{y \in X \setminus \{x\}} w_{a,x,y} y + k_{a,x} \right)}^{l_{x,a}(z)} + w_{a,x,x}^{z} x$; for convenience, we introduce the abbreviation $l_{x,a}(z)$ for the left term of the formula. The left hand side of $\psi^{r(a,z-1)}$ is

$$lhs(\psi^{r(a,z-1)}) = \sum_{x \in X} w_{\psi,x} \left(l_{x,a}(z-1) + w_{a,x,x}^{z-1} x \right)$$

Since the action does not have self-interfering effects, any variable y with a non-zero coefficient in the sum $\sum_{y \in X \setminus \{x\}} w_{a,x,y} y$ is unchanged by regression through a; hence,

$$\begin{aligned}
lhs(\psi^{r(a,z-1)^{r(a)}}) &= \sum_{x \in X} w_{\psi,x} \left(l_{x,a}(z-1) + w_{a,x,x}^{z-1} (\sum_{y \in X} w_{a,x,y} y + k_{a,x}) \right) + k_\psi \\
&= \sum_{x \in X} w_{\psi,x} \left(l_{x,a}(z-1) + w_{a,x,x}^{z-1} (\sum_{y \in X \setminus \{x\}} w_{a,x,y} y + k_{a,x}) + w_{a,x,x}^{z} x \right) + k_\psi \\
&= \sum_{x \in X} w_{\psi,x} \left(l_{x,a}(z) + w_{a,x,x}^{z} x \right) + k_\psi \\
&= \sum_{x \in X} w_{\psi,x} (f_{x,a}(z)) + k_\psi = lhs(\psi^{r(a,z)})
\end{aligned}$$

□

Proposition 4.1 (Sufficient and Necessary Condition for Reachability of a Condition)**.** *A numeric condition ψ is reachable from a state s by repeated applications of an LSF action* a *only if there exists an $m \in \mathbb{N} \cup \{0\}$ such that $s \models \psi^{r(a,m)}$ and $s \models$* pre(a) *(necessary part). This condition is also sufficient if the action does not modify any variable occurring in its precondition.*

Proof. This is proved quite directly as a consequence of Theorem 4.1, and Proposition 3.1.
Necessity. If the condition ψ is reachable through a from s this implies the existence of a finite t for which $s \overbrace{[a] \cdots [a]}^{t} \models \psi$. This, in turn, implies that:

(i) $s \models \text{pre}(a)$

(ii) $s \overbrace{[a] \cdots [a]}^{t-1} \models \psi^{r(a)}$ (using Proposition 3.1) till

(iii) $s \models \overbrace{\psi^{r(a)^{\cdots r(a)}}}^{t} \equiv \psi^{r(a,t)}$ (using Theorem 4.1)

In particular (iii) implies the existence of a $m \in \mathbb{N} \cup \{0\}$ such that $s \models \psi^{r(a,m)}$.
Sufficiency under hypothesis that action a does not affect any variable in its precondition. $s \models \psi^{r(a,m)}$ and $s \models \text{pre}(a)$ implies that $s[a] \models \psi^{r(a,m-1)}$; now, since $s[a] \models \text{pre}(a)$ we have that $s[a][a] \models \psi^{r(a,k-2)}$ till $s \overbrace{[a] \cdots [a]}^{m} \models \psi^{r(a,0)} \equiv \psi$. □

Definition 5 (Possible Achiever)**.** *An LSF action a is said to be a* possible achiever *of a numeric condition ψ from s whenever there exists an $m \in \mathbb{N}^+$ such that $s \models \psi^{r(a,m)}$.*

From Theorem 4.1 it follows that a is a possible achiever of $\psi \doteq \sum_{x \in X} w_{\psi,x} x + k_\psi \trianglerighteq 0$ from state s iff there exists m such that

$$\left[\sum_{x \in X} w_{\psi,x} f_{x,a}(m) + k_\psi \right]^s \trianglerighteq 0 \tag{6}$$

In the following, we use $ach(s, \psi)$ to denote the set of *possible achievers* of ψ from s.

Even with the restriction to linear numeric effects, using the closed form requires reasoning over an exponential function that can be non-monotonic in m (if the coefficient of the affected variable is negative). To overcome this complication, in the next section we introduce *simple numeric conditions*, and in Section 5 show that, for this class, efficient, admissible but still informed heuristics can be devised.

4.3 The Simple Numeric Condition Case

We define simple numeric conditions as linear inequalities that are only affected by actions having constant increase and decrease effects:

Definition 6 (Simple Numeric Condition). *Let $\psi \doteq \langle \xi, \rhd, 0 \rangle$. ψ 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 can be equivalently written on the form $\langle x, {+}{=}, k_{a,x} \rangle$ with $k_{a,x}$ a constant in $\mathbb{Q}$; (ii) $\rhd \in \{>, \geq\}$, i.e., ψ is not an equality condition; and (iii) ξ is linear.*

In the following a *simple numeric planning problem* is one in which every numeric condition is simple.

Note that the restriction on the effects of actions affecting variables appearing in a simple numeric condition is that they are equivalent to an increase by a constant (which may be negative); effects of the form $\langle x, {-}{=}, k_{a,x} \rangle$ and $\langle x, {:=}, x + k_{a,x} \rangle$ obviously satisfy this condition. Also note that while atomic simple numeric conditions are restricted to inequalities, an equality can be expressed as the conjunction $\langle \xi, \geq, 0 \rangle \wedge \langle \xi, \leq, 0 \rangle$.

Despite the restrictions, the class of simple numeric planning problem is as hard as general numeric planning, viz. undecidable. Helmert (2002) proved the undecidability of several restricted classes of numeric planning by reductions from known undecidable problems, including for the case of planning with numeric preconditions of the forms $\langle x, \{<, =, >\}, 0 \rangle$ and effects of the forms $\langle x, {+}{=}, 1 \rangle$ or $\langle x, {-}{=}, 1 \rangle$ (Helmert, 2002, Theorem 12). With the reformulation of equality comparisons into conjunctions of two inequalities, his reduction results in a simple numeric planning problem. The restriction to simple numeric conditions is motivated by several useful properties that they have:

First, every action that affects a variable in ψ is LSF. Effects are clearly linear, and no variable other than the affected variable appears in the right-hand side of an effect. This implies that the m-times regression of ψ through the effects of any action a can be computed in closed form: If a affects any variable appearing in ψ, then a is LSF and $\psi^{r(a,m)}$ is given by Definition 4. Moreover, the functions $f_{x,a}(m)$ in $\psi^{r(a,m)}$ all have the simple form $f_{x,a}(m) = mk_{a,x} + x$; since ψ is linear, the m-times regression $\psi^{r(a,m)}$ is also linear. If, on the other hand, a has no effect on any variable appearing in ψ, then clearly $\psi^{r(a,m)} \equiv \psi$.

Second, to determine if a is a possible achiever of an SC $\psi = \langle \sum_{x \in X} w_{\psi,x} x + k_\psi, \rhd, 0 \rangle$ it suffices to see whether the combined effect of all numeric effects of a will increase, decrease, or leave unchanged the left-hand side of ψ. The m-times regression of ψ is

$$\psi^{r(a,m)} \equiv \sum_{x \in X} w_{\psi,x}(mk_{a,x} + x) + k_\psi \rhd 0$$

Rearranging the terms, we have

$$\psi^{r(a,m)} \equiv m \overbrace{\left(\sum_{x=\text{lhs}(e), e \in \text{eff}_{\text{num}}(a)} w_{\psi,x} k_{a,x} \right)}^{N_{\psi,a}} + \overbrace{\left(\sum_{x \in X} w_{\psi,x} x \right) + k_{\psi}}^{\xi_{\psi}} \rhd 0 \tag{7}$$

where the first term ($N_{\psi,a}$) represents the net effect of each application of action a on the left-hand side of ψ, and the second term (ξ_{ψ}) is simply the left-hand side of ψ. Note that $N_{\psi,a}$ is a constant, and that $\psi^{r(a,m)}$ is also a simple numeric condition.

If $[\xi_{\psi}]^s \rhd 0$, then ψ is already satisfied in state s. If that is not the case, since the number of repetitions m is non-negative it follows that Eq. (7) has a solution – and, in other words, that a is a possible achiever of ψ, in the sense of Definition 5 – if and only if $N_{\psi,a} > 0$. Recall, however, Proposition 4.1: this is a necessary, but not sufficient, condition for the reachability of ψ from s since it does not take into account whether a is in fact applicable (once or m times).

The number of repetitions that are needed depends on the value of ξ_{ψ} in the current state (and the rate of change), but that a finite number exists does not. In this sense, the effect of actions on SCs is state-independent. In fact, we can write a closed-form expression for the number of repetitions of the effects of a required to achieve ψ from state s:

$$\text{rep}(a, \psi, s) = \begin{cases} 0 & \text{if } s \models \psi \\ \left\lceil \frac{-[\xi_{\psi}]^s}{N_{\psi,a}} \right\rceil & \text{else if } a \in ach(\psi) \\ \infty & \text{otherwise} \end{cases} \tag{8}$$

A consequence of the above properties is that the number of relevant numeric conditions that can be generated by the h^{max}_{hbd} (and h^{add}_{hbd}) recurrence is finite. Relevant numeric conditions are those generated by regressing through possible achievers only.

Theorem 4.2 (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.*

Proof. The number of numeric conditions that can be generated in the recursive evaluation of the heuristics is finite because it starts only from the finite number of conditions present in the problem (action preconditions and goal) and only actions having a positive effect on them are considered. Each regression through a possible achiever is in fact *closing the gap* w.r.t. the target of the condition, and, since the contribution is constant (we are dealing only with simple numeric conditions), it does not get infinitesimal. Thus, if at least one possible achiever exists, the base case in which the condition is satisfied will be reached (Eq. 2, line 1) at some point. Otherwise the cost for that condition is infinity, and this can only be true if there is no possible achiever for the condition. □

It also follows that ψ is relaxed reachable, i.e., that $h^{max}_{hbd}(s, \psi)$ is finite, iff there exists some action a that has a positive effect on ψ such that $h^{max}_{hbd}(s, \text{pre}(a))$ is also finite. In other words, the relaxed reachability of a SC can be demonstrated by considering only a single achiever. Finding the minimum relaxed cost, i.e., the value of $h^{max}_{hbd}(s, \psi)$ (or $h^{add}_{hbd}(s, \psi)$), may however require considering a combination of achievers. To see why, consider two actions $\langle a_0, \emptyset, x = x + 0.9\rangle$ and $\langle a_1, \emptyset, x = x + 1.1\rangle$, with costs $\gamma(a_0) = 1$ and $\gamma(a_1) = 1.5$, and the cost of achieving the condition $\psi = \langle x - 2, \geq, 0\rangle$ from a state where $x = 0$. The minimum cost using only repetitions of one action is 3 (either three times a_0 or two times a_1), while the minimum cost of 2.5 is achieved using each action once. In general, an exponential number of regressions may be required to find the minimum cost combination of achievers. Thus, the fact that h^{max}_{hbd} with SCs can be computed in finite time does not imply that the heuristic can be computed efficiently.

The next section shows how to obtain approximations of h^{max}_{hbd} and h^{add}_{hbd} that support efficient reasoning over different combinations of actions. To do this, we exploit that the m-times effect regression of an SC through different actions is commutative; that is, $\psi^{r(a,n)^{r(b,m)}} \equiv \psi^{r(b,m)^{r(a,n)}}$ for any SC ψ and actions a and b affecting it. To see why, note that if a variable x is affected by both a and b, then the value of x after applying first b m times and then a n times is $mk_{b,x} + f_{x,a}(n) = mk_{b,x} + nk_{a,x} + x$, which is clearly the same as $nk_{a,x} + f_{x,b}(m)$, or the value of x after first applying a n times and then b m times.

We characterise the above optimisation task using a Mixed Integer Program, and then elaborate upon its continuous relaxation to devise tractable inadmissible and admissible estimates, which are numeric analogues of h^{add} and h^{max}.

5. Tractable Heuristics via Explicit Regression

Although finite, a naive implementation of both h^{max}_{hbd} and h^{add}_{hbd} is not tractable. In this section, we will derive approximations of both heuristics that can be computed in time polynomial in the problem representation, i.e., the number of actions, propositions and numeric conditions that appears in it. Our tractable approximation $\hat{h}^{max}_{hbd}$ of h^{max}_{hbd} is still admissible, and our tractable approximation $\hat{h}^{add}_{hbd}$ of h^{add}_{hbd} is inadmissible but more informed.

5.1 An Inadmissible Estimate: $\hat{h}^{add}_{hbd}$

The core of the intractability of h^{add}_{hbd} lies in the recursive minimisation over combinations of possible achievers of numeric conditions. To address it, we will first turn it into an explicit minimisation problem, and then relax that problem until it can be solved efficiently.

The commutativity of the m-times regressor makes it possible to formulate h^{add}_{hbd} using a linear combination of each action's contribution[8]:

Theorem 5.1 (Explicit Regression for h^{add}_{hbd}). *Let ψ be a simple numeric condition.*

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

Proof. Let $\mathcal{P} \doteq \langle a_0, \cdots, a_z\rangle$ be a sequence of actions whose combined effects achieve ψ, and let $c(\mathcal{P}) \doteq \sum_{a \in \mathcal{P}}(h^{add}_{hbd}(s, \text{pre}(a)) + \gamma(a))$ be the associated cost. $h^{add}_{hbd}(s, \psi)$ finds, among all such finite sequences the one with minimum cost, i.e., $\mathcal{P}^*$. Thus, $h^{add}_{hbd}(s, \psi) = c(\mathcal{P}^*)$. Note that $\mathcal{P}^*$ is such that:

8. This can be seen as lifting up the sequencing problem to action counting, drawing this way a nice relation with the line of research on operator counting (Pommerening, Röger, Helmert, & Bonet, 2014)

(i) the action effects on ψ are commutative with each other; and

(ii) the same action can be repeated a number $m_a \geq 0$ of times.

Therefore, let $MS(\mathcal{P}^*)$ be the set of pairs (a, m_a) where a is an action in $\mathcal{P}^*$ and m_a the number of occurrences of this action in $\mathcal{P}^*$; we can rewrite $c(\mathcal{P}^*)$ as follows:

$$c(\mathcal{P}^*) \doteq \sum_{a \in \mathcal{P}} (h^{add}_{hbd}(s, \text{pre}(a)) + \gamma(a)) = \sum_{(a,m_a) \in MS(\mathcal{P}^*)} \Big(m_a \cdot \Big(h^{add}_{hbd}(s, \text{pre}(a)) + \gamma(a) \Big) \Big)$$

It is easy to see that the cost of the optimal sequence found by the explicit regression of the theorem is equivalent to the cost of the optimal sequence (i.e., $c(\mathcal{P}^*)$) found solving the recursive regression. □

Note that what Theorem 5.1 gives is a formula for calculating $h^{add}_{hbd}(s, \psi)$ *given* the estimated costs $h^{add}_{hbd}(s, \text{pre}(a_j))$ for all relevant actions. This formula can be used to perform a single update-step in the computation of h^{add}_{hbd}, whether in a label-correcting algorithm or a uniform-cost search, as described in Subsection 5.3. However, this formula has two problems: First, the hybrid additive heuristic h^{add}_{hbd} suffers from over-counting, in the same way as the classical h^{add} heuristic does, but in the numeric case this is exacerbated because the estimated cost of an action's preconditions is added for every repetition of the action. Second, because it involves a minimisation problem over $n = |A|$ integer variables, it cannot be solved efficiently. As we shall now show, the resolution of both problems is linked, through a process of two approximations to the original formula.

First, to reduce the overly pessimistic estimate that this results in, we consider a more optimistic variant where the estimated precondition cost of actions used in the solution are counted only once. That is, we replace $\sum_{j=1..n} m_j[h^{add}_{hbd}(s, \text{pre}(a_j)) + \gamma(a_j)]$ 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 this modified update formula with update$'(\psi)$:

$$\text{update}'(\psi) \doteq \min_{\substack{m_1,..,m_{n=|A|} \in \mathbb{N} \\ s \models \psi^{r(a_1,m_1)^{\cdot^{\cdot^{r(a_n,m_n)}}}}}} \sum_{\substack{j=1..n \\ \text{s.t. } m_j > 0}} (h^{add}_{hbd}(s, \text{pre}(a_j)) + \gamma(a_j) m_j)$$

The solution to update$'(\psi)$ can be computed by solving the following Mixed Integer Program:

$$\begin{aligned} \text{minimise} \quad & \sum_{a \in A} \Big(h^{add}_{hbd}(s, \text{pre}(a)) \cdot act_a + \gamma(a) m_a \Big) \\ \text{subject to} \quad & \sum_{x \in X} w_{\psi,x} \left(\sum_{a \in A} m_a k_{a,x} + [x]^s \right) + k_\psi \rhd 0 \\ & m_a \leq act_a \cdot M_a, \forall a \in A \\ & act_a \in \{0, 1\}, m_a \in \mathbb{N}, \forall a \in A \end{aligned} \tag{9}$$

The decision (integer) variable m_a, as before, represents the number of times that action a is used in the optimal solution, while act_a is a 0/1 variable that takes the value 1 iff

$m_a > 0$, i.e., iff a is used at all. The constants are taken from the condition ($w_{\psi,x}$, k_ψ), the actions ($\gamma(a)$, $k_{a,x}$) and the state s from which the heuristic is computed ($[x]^s$). For the formulation to be correct, we have to choose the constant M_a to be a number greater or equal to the maximum number of repetitions of a that may be used in any optimal solution; as detailed in Section 4.3, we can simply find for each possible achiever a the number of repetitions needed to satisfy ψ from s using only a, $\text{rep}(a, \psi, s)$, which is an upper bound on the number of times that a may be used in an optimal solution. If there is no achiever of ψ from s the MIP is infeasible; in this case $\text{update}'(\psi) = \infty$.

However, solving the optimisation problem in Eq. 9 is NP-complete:

Theorem 5.2. *Computing* $\text{update}'(\psi)$ *is NP-complete.*

Proof. The problem is in NP, as it can be formulated using Eq 9. The problem is also NP-hard as its decisional version is equivalent to the decisional variant of the famous unbounded-knapsack problem (Martello & Toth, 1990), as the following shows. Let us first formulate our optimisation problem as a decision problem. In the decision version we set an upper-bound K to the cost and use a lower-bound T to determine valid solutions. The decision problem asks whether there is a number of repetitions for each action in A such that the cost is within K, and the lower bound T still in check. Formally:

$$\forall a \in A \;\; \exists m_a \geq 0 \;\mid\; \sum_{a \in A} \gamma(a) m_a \leq K \text{ and } \sum_{a \in A} N_{\psi,a} m_a \geq T$$

where $T \doteq -\,[\xi_\psi]^s \doteq -\,[\left(\sum_{x \in X} w_{\psi,x} x\right) + k_\psi]^s$, and $N_{\psi,a} \doteq \sum_{x \in X} w_{\psi,x} k_{a,x}$. ($T$ is the *target value*, obtained by separating the evaluation of the left-hand side expression of ψ on the state s from the contribution of the action effects $N_{\psi,a}$, and bringing the resulting value on the right-hand side of the inequality.)

An unbounded-knapsack problem (which is a well known NP-complete problem) consists in determining a subsets of non-negative integer variables $S' \subseteq S = \{e_0, ..., e_n\}$ such that each can be added an arbitrary number of times so as to maximise benefit ($\sum v_i(e_i)$) keeping the overall weight in check ($\sum w_i(e_i) \leq W$). It is straightforward to see that its decision version can be solved using our machinery: it suffices to use one action to model the adding of a particular element. The benefit of this element corresponds to the action contribution ($N_{\psi,a}$), and the weight to its cost ($\gamma(a)$). This proves the NP-hardness of our problem. □

5.1.1 Relaxation of the MIP and its Implications

Because computing $\text{update}'(\psi)$ is intractable, we further approximate it by allowing the m_a variables to take rational rather than integral values, i.e., replacing the constraint $m_a \in \mathbb{N}$ with $m_a \geq 0$ for all a in MIP 9. We denote this relaxation with $\text{update}''(\psi)$, and the heuristic that results from replacing the right-hand side of the case for (simple) numeric conditions in the h^{add}_{hbd} recurrence (3) with $\text{update}''(\psi)$ by $\hat{h}^{add}_{hbd}$.

The resulting problem is still a MIP, because the act_a variables remain integral. However, as we will show in Theorem 5.3 below, there always exists an optimal solution to the continuous relaxation of the MIP that uses only one achiever; in other words: there is no need to look for combinations of actions to satisfy a given condition. This means that the optimum can be found in polynomial time, without invoking a MIP solver. Our implementation of $\hat{h}^{add}_{hbd}$ makes use of this.

Theorem 5.3. *An optimal solution to the relaxation of MIP 9 with fractional m_a variables can be found by minimising over each possible achiever in isolation:*

$$\text{update}''(\psi) = \begin{cases} 0 & \textit{if } s \models \psi \\ \min\limits_{\substack{a \in ach(s,\psi), \hat{m} \in \mathbb{Q}^{\geq 0} \\ s \models \psi^{r(a,\hat{m})}}} (\hat{m} \cdot \gamma(a) + \hat{h}^{add}_{hbd}(\text{pre}(a))) & \textit{otherwise} \end{cases}$$

Proof. If the condition is already satisfied in state s, the minimum in both the formula above and the relaxation of MIP 9 is zero. If there is no achiever of ψ in s, update$''(\psi) = \infty$ by definition. Thus, suppose $s \not\models \psi$ and $ach(s, \psi) \neq \emptyset$, and let f^* be the minimum cost of achieving the condition using a single achiever, that is:

$$f^* = min_{a \in A}\{\frac{T}{N_{\psi,a}} \cdot \gamma(a) + \hat{h}^{add}_{hbd}(pre(a))\}$$

where $T \doteq -[\xi]^s$, i.e., the gap that needs to be closed to achieve the condition. A possible achiever requires $N_{\psi,a} > 0$; in that case, $\frac{T}{N_{\psi,a}} \cdot \gamma(a) + \hat{h}^{add}_{hbd}(pre(a))$ represents the relaxed minimum cost of achieving ψ with a alone. Let a^* be the action that attains the minimum.

Assume for contradiction that f^* is not the optimal solution to MIP 9. This implies the existence of some other action a' contributing to the satisfaction of ψ. Let us refer to this cheaper cost with f^+. Suppose, for simplicity, that the solution with cost f^+ uses two actions, a^* and a'. This means that

$$f^+ = \frac{T - T'}{N_{\psi,a^*}} \cdot \gamma(a^*) + \frac{T'}{N_{\psi,a'}} \cdot \gamma(a') + \hat{h}^{add}_{hbd}(pre(a')) + \hat{h}^{add}_{hbd}(pre(a^*))$$

where T' is the part of the gap that is closed by some fractional number of repetitions of action a', and $T - T'$ the part closed by a^*. Since the solution is optimal, we must have $T' \leq T$ and $T' > 0$. Since $f^+ < f^*$, we can deduce that:

$$\frac{T - T'}{N_{\psi,a^*}} \cdot \gamma(a^*) + \frac{T'}{N_{\psi,a'}} \cdot \gamma(a') + \hat{h}^{add}_{hbd}(pre(a')) + \hat{h}^{add}_{hbd}(pre(a^*)) < \frac{T}{N_{\psi,a^*}} \cdot \gamma(a^*) + \hat{h}^{add}_{hbd}(pre(a^*))$$

which yields

$$\frac{T - T'}{N_{\psi,a^*}} \cdot \gamma(a^*) + \frac{T'}{N_{\psi,a'}} \cdot \gamma(a') + \hat{h}^{add}_{hbd}(pre(a')) < \frac{T}{N_{\psi,a^*}} \cdot \gamma(a^*)$$

Subtracting the first term from both sides and simplifying the right-hand side, we can deduce that

$$\frac{T'}{N_{\psi,a'}}\gamma(a') + \hat{h}^{add}_{hbd}(pre(a')) < \frac{T'}{N_{\psi,a^*}}\gamma(a^*)$$

which, observing that the precondition cost of an action is always non-negative, implies

$$\frac{T}{N_{\psi,a'}}\gamma(a') + \hat{h}^{add}_{hbd}(pre(a')) < \frac{T}{N_{\psi,a^*}}\gamma(a^*) + \hat{h}^{add}_{hbd}(pre(a^*))$$

for any non-negative T. This contradicts the assumption that f^* is the minimum single-achiever solution cost. If the optimal solution uses more than two actions, we can simply apply the same argument to show that all but one can be eliminated, since actions' contributions are additive. □

Let us return to the the example of Section 4.3 with actions $\langle a_0, \emptyset, x = x + 0.9\rangle$ and $\langle a_1, \emptyset, x = x+1.1\rangle$, with costs $\gamma(a_0) = 1$ and $\gamma(a_1) = 1.5$, and the condition $\psi = \langle x-2, \geq, 0\rangle$. The optimal cost of achieving ψ from a state with $x = 0$ is 2.5, by applying each action once, while using either action alone results in a cost of 3 (either three times a_0 or two times a_1). However, if we allow for fractional action repetitions, the optimal solution is to apply action a_0 $2 + 2/9$ times, for a cost of $2 + 2/9$. Using action a_1 requires $1 + 9/11$ repetitions, for a cost of $2 + 8/11$.

From now on, we will use $\widehat{\mathrm{rep}}(a, \psi, s)$ to denote the minimum *fractional* number of repetitions of the effects of an action a that are needed to satisfy ψ, starting from s. That is:

$$\widehat{\mathrm{rep}}(a, \psi, s) = \begin{cases} 0 & \text{if } s \models \psi \\ \frac{-[\xi_\psi]^s}{N_{\psi,a}} & \text{else if } a \in ach(\psi) \\ \infty & \text{otherwise} \end{cases} \tag{10}$$

where ξ_ψ is the expression on the left hand side of the condition. Note that this is the same as Eq. 8 except that the fraction is not rounded up. After having gone through these steps, let us report the resulting $\hat{h}^{add}_{hbd}$ heuristic:

$$\hat{h}^{add}_{hbd}(s, \psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(s,\psi)} \left(\hat{h}^{add}_{hbd}(s, \mathrm{pre}(a)) + \gamma(a)\right) & \text{if } \psi \in \mathsf{PC} \\ \boldsymbol{\min\limits_{a \in ach(s,\psi)} \left(\widehat{\mathrm{rep}}(a, \psi, s) \cdot \gamma(a) + \hat{h}^{add}_{hbd}(\mathrm{pre}(a))\right)} & \text{if } \psi \in \mathsf{SC} \\ \min\limits_{\psi' \in \psi} \hat{h}^{add}_{hbd}(s, \psi') & \text{if } \psi \text{ is } \vee \\ \sum\limits_{\psi' \in \psi} \hat{h}^{add}_{hbd}(s, \psi') & \text{if } \psi \text{ is } \wedge \end{cases} \tag{11}$$

Note that, $\hat{h}^{add}_{hbd}(s, \psi)$ will never give a greater estimate than that computed by its integral and more pessimistic variant $h^{add}_{hbd}(s, \psi)$. This is because $\hat{h}^{add}_{hbd}(s, \psi)$ only counts preconditions of an action once (regardless of whether multiple repetitions of the same action are required), and uses the continuous relaxation of $\mathrm{update}'(\psi)$, i.e., $\mathrm{update}''(\psi)$.

5.2 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 ψ is given by the optimal solution to a slight reformulation of the MIP in Eq. 9 where we drop precondition cost:

$$\begin{aligned} \text{minimise} \quad & \sum_{a \in ach(s,\psi)} \gamma(a) m_a \\ \text{subject to} \quad & \sum_{x \in X} w_{\psi,x} \left(\sum_{a \in A} m_a k_{a,x} + [x]^s \right) + k_\psi \rhd 0 \\ & m_a \in \mathbb{N} \;\; \forall a \in A \end{aligned} \tag{12}$$

Second, we take the continuous relaxation of the above problem. The relaxation is tractable and, as observed previously, lower bounds the integer version. Also in this case optimal solutions to the continuous relaxation can be found by looking at actions in isolation (i.e., $m_a > 0$ for just one $a \in ach(s, \psi)$).

Third, we improve this estimate by considering only relaxed reachable actions and their precondition costs. Let $A^+ \subseteq A$ denote this set. 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 ψ be a state and a simple numeric condition, we obtain the following expression for the cost update:

$$\hat{h}^{max}_{hbd}(s, \psi) \doteq \left(\min_{a \in A^+ \cap ach(s,\psi)} \widehat{\text{rep}}(a, \psi, s) \cdot \gamma(a) \right) + \left(\min_{a \in A^+ \cap ach(s,\psi)} \hat{h}^{max}_{hbd}(s, \text{pre}(a)) \right) \tag{13}$$

where $A^+ = \{a \in A \mid \hat{h}^{max}_{hbd}(s, \text{pre}(a)) < \infty\}$ is the set of relaxed reachable actions.

Theorem 5.4 (Admissibility). *For simple numeric planning problems, $\hat{h}^{max}_{hbd}(s, \psi) \leq h^*(s, \psi)$.*

Proof. To prove the statement we consider the different cases that can arise:

1. ψ is already satisfied in s;
2. ψ is a single simple numeric condition;
3. ψ is a single propositional condition;
4. ψ is a conjunction.

When ψ is satisfied in s (case 1), $\hat{h}^{max}_{hbd}(s, \psi) = 0$, which is a valid lower bound. Case 3 follows directly from the original formulation of h^{max}. Assuming that cases 2 and 3 are valid lower bounds, maximising over a set of valid lower bounds (case 4) will also be a lower bound. The only slightly tricky case is 2. Let π^* be an optimal plan to achieve ψ from s. We can decompose π^* into several sub-sequences of actions. π^* must contain a multi-set of achievers of ψ, whose combined effects are sufficient to make ψ true from s; call this sub-sequence π^*_A, and let A be its total cost. Let a_0 be the action in π^*_A that appears first in π^*, let π^*_B be the sub-sequence of actions in π^* that are necessary for achieving the preconditions of a_0, and let B be its total cost. Since every action in π^*_B appears before every action π^*_A in the plan, the two are disjoint and the total cost of π^* is at least $A + B$. The $\hat{h}^{max}_{hbd}$ estimate is likewise made up of two terms. The first term in Eq. 13 lower-bounds the minimum cost of any combination of actions that can achieve ψ from s, i.e., it lower-bounds A. (That the optimal solution to the linear relaxation of MIP 12 is attained by considering at most one achiever follows by the same reasoning as in the proof of Theorem 5.3.) The second term in Eq. 13 lower-bounds the cost of achieving the preconditions of the first achiever of ψ in an optimal plan, i.e., it lower-bounds B. Hence, $\hat{h}^{max}_{hbd}(s, \psi) \leq A + B \leq \gamma(\pi^*)$. $\square$

However, Eq. 13 is too optimistic, in that it allows the minimum in the first term to use any possible achiever a whose current precondition cost estimate is finite, even when the estimated precondition cost $\hat{h}^{max}_{hbd}(s, \text{pre}(a))$ is greater than what the minimum of the first term over actions excluding a would be. This is caused by the decoupling of the two minimisation problems.

To overcome this issue we restrict the set of actions considered in both minimisation problems to only those ones whose precondition cost does not exceed the cost of achieving the numeric condition at hand without using them. Let $\hat{h}^{max}_{hbd}(A', s, \psi)$ denote the heuristic estimate taken using only the set of actions $A' \subseteq A$. We define the set

$$R(A, s, \psi) \doteq \{a \mid a \in A^+ \cap ach(s, \psi), \hat{h}^{max}_{hbd}(A \setminus \{a\}, s, pre(a)) < \hat{h}^{max}_{hbd}(A \setminus \{a\}, s, \psi)\}$$

and redefine the heuristic update equation as follows:

$$\hat{h}^{max}_{hbd}(s, \psi) \doteq \left(\min_{a \in R(A,s,\psi)} \widehat{\text{rep}}(a, \psi, s) \cdot \gamma(a) \right) + \left(\min_{a \in R(A,s,\psi)} \hat{h}^{max}_{hbd}(s, \text{pre}(a)) \right) \tag{14}$$

It is important to observe that this restriction does not violate admissibility of the heuristic. The revised formulation only makes sure that we do not consider actions that make the estimate unnecessary low. The only actions discarded from the minimisation are those having an estimated precondition cost which is at least as high as the cost of achieving the target condition without them. Of course, this restriction of the set of actions makes the new estimate more accurate than its original formulation. Now we are ready to summarise the $\hat{h}^{max}_{hbd}$ heuristic:

$$\hat{h}^{max}_{hbd}(s, \psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(s,\psi)} \left(\hat{h}^{max}_{hbd}(s, \text{pre}(a)) + \gamma(a) \right) & \text{if } \psi \in \mathsf{PC} \\ \boldsymbol{\min\limits_{a \in R(A,s,\psi)} \widehat{\text{rep}}(a, \psi, s) \cdot \gamma(a) + \min\limits_{a \in R(A,s,\psi)} \hat{h}^{max}_{hbd}(s, \text{pre}(a))} & \text{if } \psi \in \mathsf{SC} \\ \min\limits_{\psi' \in \psi} \hat{h}^{max}_{hbd}(s, \psi') & \text{if } \psi \text{ is } \vee \\ \max\limits_{\psi' \in \psi} \hat{h}^{max}_{hbd}(s, \psi') & \text{if } \psi \text{ is } \wedge \end{cases} \tag{15}$$

5.3 Computational Aspects of $\hat{h}^{max}_{hbd}$ and $\hat{h}^{add}_{hbd}$

The approximations introduced in the previous two sections 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 the propositional h^{max} and h^{add} heuristics, this enables us to use a Uniform Cost Search procedure to compute these heuristics in polynomial time.

Theorem 5.5 (Tractability). *$\hat{h}^{max}_{hbd}$ and $\hat{h}^{add}_{hbd}$ can be computed in time polynomial in the size of the problem.*

Alg. 1 shows a polynomial algorithm proving the previous. The procedure is basically a blind search over the relaxed reachable actions of the problem. $\eta(a)$ denotes the current estimate for satisfying precondition of action a. $\omega(\psi)$ denotes the cost of achieving a single terminal ψ. Each condition in ψ gets an estimate of 0 or ∞ depending on whether it is satisfied or not in the state. The priority queue, Q, serves the purpose of organising the expansion of the reachable actions according to their heuristic cost given by function η.

Algorithm 1: $\hat{h}^{max/add}_{hbd}$ computation

```
1  computeCost(ψ- Formula):
2  if $\psi$ is literal then
3      return $\omega(\psi)$
4  if $\psi$ is $\vee$ then
5      return $\min_{c\in\psi}$ computeCost($c$)
6  if $\psi$ is $\wedge$ then
7      if max setting then
8          return $\max_{c\in\psi}$ computeCost($c$)
9      return $\sum_{c\in\psi}$ computeCost($c$)
10 Main(A - Actions, G - Goals):
11 $\omega, \eta = init\_costs(A, G)$
12 $Q$ = PriorityQueue($\{(a, 0) | a \in A, \eta(a) = 0\}$);
13 $A = A \cup \{G_a\}$ ;
14 $closed = \emptyset$
15 $poss = \emptyset$;
16 while $Q \neq \emptyset$ do
17     $a$ = pull($Q$);
18     if $G_a == a$ then
19         return $\eta(G_a)$
20     $closed = closed \cup \{a\}$
21     foreach $\psi \mid \widehat{\text{rep}}(a, \psi, s) < \infty, \psi \in NCs$ do
22         k = $\widehat{\text{rep}}(a, \psi, s)$;
23         if Admissible then
24             $poss = poss \cup (a, \psi)$;
25             $\omega(\psi) = \min(\omega(\psi), k \cdot \gamma(a) + \min_{(a',\psi)\in poss}(\eta(a')))$;
26         else
27             $\omega(\psi) = \min(\omega(\psi), k \cdot \gamma(a) + \eta(a))$
28     foreach $\psi \mid a \in ach(\psi), \psi \in PCs$ do
29         $\omega(\psi) = \min(\omega(\psi), \gamma(a) + \eta(a))$
30     foreach $a \in A$ do
31         $\eta(a)$ = computeCost (pre($a$));
32     $A' = \{a \in A \mid \eta(a) \neq \infty, a \notin closed\}$;
33     update_queue($Q$,$A'$,$\eta$);
34     return $\infty$
```

Alg. 1 makes use of a dummy action G_a having the goal as precondition and empty effects. This is a simple expedient to have homogeneous elements in the priority queue. Every time a new action is pulled from Q, unless it is G_a (in which case the algorithm can terminate with the cost associated to it), the algorithm updates the η cost for every action of the problem. This is triggered in case the action a just pulled from Q has indeed affected at least a meaningful condition. This can be made efficient by carefully triggering only those actions whose preconditions have been touched by the action. Then, routine *update_queue* updates the queue for all those reachable actions that have not being closed yet. The procedure eventually terminates as the queue will empty (actions are never re-added). However, the procedure may also end earlier, as soon as G_a is pulled from Q [9] . Note that the update of the cost of a condition needs to take into account actions already expanded that are though achievers of the condition object of the update (line 25). The additive version and the admissible version are obtained switching appropriately between maximising and summing in the computeCost routine, which explores the action precondition formula using the cases defined for the two cases as for Equations 11 and 15, but considering the recursive calls as updates instead of actual recursions. The recursion within the computeCost function only serves the purpose of properly exploring the formula, which for simplicity is hereby represented as a tree.

Because the algorithm is an optimal search over the heuristic cost of each action, whenever an action is popped out from the priority queue, its associated cost will be the minimum possible cost according to the update rule defined by the specific setting (admissible or additive).

The computational complexity of the procedure is bounded polynomially by the number of actions in A, and the number of conditions in C. The intuition is that the main loop (the priority queue) needs to iterate at most $|A|$ times; once an action is popped out from the priority queue, it will never be re-inserted again. This is ensured by using the "closed" set in the algorithm; then, at most there are $|C|$ updates to be performed, and relative actions to be added in the priority queue if it is necessary. The small minimisation task performed for action preconditions of possible achievers of numeric condition (line 25) can also be optimised by using another map that keeps track of the precondition contribution.

6. Over-Approximation Guarantees and Relation with IBR

We conclude this first part by observing that all the heuristics presented so far are safe-pruning heuristics, meaning that they give infinite values only for problems that are actually unsolvable. This holds under the condition that all the subgoals of the problems are simple numeric conditions.

Theorem 6.1. *For simple numeric planning problems, $\hat{h}^{max}_{hbd}$ and $\hat{h}^{add}_{hbd}$ are safe-pruning heuristics, i.e. $\hat{h}^{max/add}_{hbd} = \infty \Rightarrow h^* = \infty$*

Proof. First, note that both h^{max}_{hbd} and h^{add}_{hbd} are based on the same 1-subgoaling relaxation h^1_{hbd}, so have the same pruning power. Let Π be a numeric planning task, with initial state I and goal G, and π a solution plan. Regressing the goal G through π yields a condition that is satisfied in the initial state I (cf. Proposition 3.1). Likewise each precondition of each action appearing in π regressed through the preceding segment of π yields a condition satisfied in I. Thus $h^{max}_{hbd}(I, G)$ and $h^{add}_{hbd}(I, G)$ are both finite. □

9. The algorithm is similar to Liu's implementation (2002), yet in this version we also include the numeric effects and their implication on the numeric conditions of the problem, and we iterate over actions instead of iterating over conditions.

We now turn our attention to the relationship between h^1_{hbd} (the relaxation underlying both $\hat{h}^{max}_{hbd}$ and $\hat{h}^{add}_{hbd}$) and the interval-based relaxation of Π (Scala et al., 2016b; Aldinger, Mattmüller, & Göbelbecker, 2015), denoted Π^+. It is easy to see that problems that are solvable in h^1_{hbd} are also solvable in Π^+: The set of achievers that is used in the solution to the h^1_{hbd} equation can be used to solve Π^+ in an obvious way: achievers can be applied to extend state intervals up to the point where all the conditions they can possibly achieve become satisfied. Thus, any problem detected as unsolvable by Π^+ is also detected as unsolvable by h^1_{hbd}. However, the converse is not true: there are problems solvable in Π^+ that are detected as unsolvable by h^1_{hbd}. This is because internal interactions within an action are relaxed in Π^+, but some are considered by h^1_{hbd}. Consider the following example: the initial state is $I \doteq \langle x = 0, y = 0\rangle$, the goal is $G \doteq x + y > 0$, and there is a single action $a \doteq \langle \emptyset, \{x = x+1, y = y-1\}\rangle$. The interval-based relaxation Π^+ of this problem is solvable. The initial state in Π^+ is $I \doteq \langle x = [0, 0], y = [0, 0]\rangle$. Applying action a (trivially executable in the initial state) generates the relaxed state $I[a] = \langle x = [0, 1], y = [-1, 0]\rangle$. Following terminology from Scala et al. (2016b), we have that $[lhs(G)]^{I[a]} : [-1, 1]$, which implies that $I[a] \models^+ G$. With h^1_{hbd}, the problem is instead trivially unsolvable because *the action a is not a possible achiever of G.*

Our Sailing domain instance shown in Figure 1 (Section 2.3), presents another example: the rescue requires satisfying $x + y \geq 10$ starting from position $\langle x = 1, y = 1\rangle$. In the interval-based relaxation, any action that increases *at least* one of the variables x or y can be used to achieve the goal, and therefore is considered *helpful* by heuristics based on this relaxation. This includes the two actions a_{NE} and a_{W}, but also the two misleading actions a_{NW} and a_{SE}. The latter two actions increase either x or y but decrease the other by the same amount, and thus fail to bring the state any closer to satisfying $x + y \geq 10$. Thus, ignoring the interaction of action effects not only leads to weaker heuristic estimates but also can make irrelevant actions appear helpful.

However, even though the h^1_{hbd} subgoaling relaxation captures some of the interactions among actions' effects, it still considers satisfaction of multiple conditions in conjunction independently. In the next three sections of the paper we describe two ways of remedying this issue. The first is a simple technique based on the adoption of redundant constraints (Section 7). The second is a more sophisticated approach that revises the notion of possible achievers to better account for conjunctive constraints (Sections 8–9).

7. Redundant Constraints

The subgoaling relaxation, as presented thus far, assumes and exploits strong independence among subgoals appearing in conjunctions. Depending on whether we are interested in admissibility or informativeness, we maximise or sum (respectively) the contribution of each conjunct. It is by reasoning over those basic elements that, for the admissible case, we obtain a numeric equivalent of the so called critical-paths *necessary* to achieve each goal. However, even though this form of reasoning gives us a way to devise admissible and tractable estimates, it ignores interactions between subgoals.

To illustrate the consequences, imagine a planning problem modelling an agent tasked with pouring a given amount L of water into an empty bucket in less than T seconds. The agent has a single action, *pour*: each time it is executed, it increases the level of water by one unit and increases the time spent by one second. Consider an instance with goal $(T \leq 10) \wedge (L \geq 11)$, and initial state $\langle T = 0, L = 0\rangle$. Clearly, the problem is unsolvable: there is no way of filling the bucket in less than 11 seconds. The pouring action has a positive effect on the second goal (it is one of its possible achievers), but has an irreversible negative side effect on the time constraint. However, the subgoaling relaxation of the task is solvable. A relaxed plan for this problem is to execute 11 times the refilling action. The relaxation monotonically assumes that once a subgoal is satisfied, as $T \leq 10$ is in the initial state, then this subgoal remains satisfied at all future steps.

In order to overcome this problem in some situations, we propose a simple problem reformulation technique whose aim is to make some of the hidden numeric structure of the problem explicit. More precisely, let c be a precondition or a goal, we modify c by introducing redundant constraints that are implied by the simultaneous satisfaction of each pair of (numeric) conditions in any conjunction set in c. Let $c_1 \doteq \langle \xi, \rhd, 0\rangle$ and $c_2 \doteq \langle \xi', \rhd', 0\rangle$ be two SCs appearing in a conjunction. The implied redundant constraint that we add to the conjunction is $\langle \xi + \xi', \mathit{weaker}(\rhd, \rhd'), 0\rangle$, where $\mathit{weaker}(\rhd, \rhd')$ denotes the less strict of the two relations $\rhd$ and $\rhd'$; that is, if one of them is strict inequality and the other a non-strict inequality, the redundant constraint uses the non-strict relation. This is to ensure that the implied constraints remain a necessary condition for the satisfiability of the conjunction of its implicants. We denote the redundant constraint implied by two SCs ψ and ψ' with $\mathrm{red}(\psi, \psi')$. Note that the implied constraint is also an SC.

In the example above, the conjunction of the two inequalities $-T+10 \geq 0$ and $L-11 \geq 0$ imply the constraint $L - T - 1 \geq 0$. Because the pouring action has no net effect on $L - T$ (it increases both variables by 1), applying the *m-times effect regressor* to this constraint results in $m - m - 1 \geq 0$, which is not satisfiable. Thus, the pouring action is not a possible achiever of the implied constraint, and therefore also not of the original conjunctive goal. Situations like this are quite frequent in numeric planning problems.

We define variants of our heuristics by reformulating the problem with the addition of redundant constraints for each pair of numeric conditions in conjunctions: we denote with $\hat{h}^{rmax}_{hbd}$ the extended admissible estimate (the 'r' stands for redundant constraints), and with $\hat{h}^{radd}_{hbd}$ the inadmissible version. Eq. 16 and 17 below formalise $\hat{h}^{rmax}_{hbd}$ and $\hat{h}^{hradd}_{hbd}$. $\mathrm{red}(\psi', \psi'')$ denotes the redundant condition implied by ψ' and ψ'', as defined above.

$$\hat{h}^{rmax}_{hbd}(s,\psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(s,\psi)} \left(\hat{h}^{rmax}_{hbd}(s, \mathrm{pre}(a)) + \gamma(a)\right) & \text{if } \psi \in \mathsf{PC} \\ \min\limits_{a \in R(A,s,\psi)} \widehat{\mathrm{rep}}(a,\psi,s) \cdot \gamma(a) + \min\limits_{a \in R(A,s,\psi)} \hat{h}^{rmax}_{hbd}(s, \mathrm{pre}(a)) & \text{if } \psi \in \mathsf{SC} \\ \min\limits_{\psi' \in \psi} \hat{h}^{rmax}_{hbd}(s,\psi') & \text{if } \psi \text{ is } \vee \\ \boldsymbol{\max\left(\max\limits_{\psi' \in \psi} \hat{h}^{rmax}_{hbd}(s,\psi'), \max\limits_{\substack{\psi',\psi'' \in \psi \cap \mathsf{NC}, \\ \psi' \neq \psi''}} \hat{h}^{rmax}_{hbd}(s, \mathrm{red}(\psi',\psi''))\right)} & \text{if } \psi \text{ is } \wedge \end{cases} \tag{16}$$

$$\hat{h}^{radd}_{hbd}(s,\psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(s,\psi)} \left(\hat{h}^{radd}_{hbd}(s, \text{pre}(a)) + \gamma(a) \right) & \text{if } \psi \in \mathsf{PC} \\ \min\limits_{a \in ach(s,\psi)} (\widehat{\text{rep}}(a,\psi,s) \cdot \gamma(a) + \hat{h}^{radd}_{hbd}(\text{pre}(a))) & \text{if } \psi \in \mathsf{SC} \\ \min\limits_{\psi' \in \psi} \hat{h}^{radd}_{hbd}(s,\psi') & \text{if } \psi \text{ is } \vee \\ \boldsymbol{\sum\limits_{\psi' \in \psi} \hat{h}^{radd}_{hbd}(s,\psi') + \sum\limits_{\substack{\psi',\psi'' \in \psi \cap \mathsf{NC}, \\ \psi' \neq \psi''}} \hat{h}^{radd}_{hbd}(s, \textbf{red}(\psi',\psi''))} & \text{if } \psi \text{ is } \wedge \end{cases} \tag{17}$$

The reformulation could be further strengthened in certain cases by including redundant constraints over conjunctions that are implied by more complex formulae. For example, $(\neg b \vee d) \vee (b \wedge c)$ implies $c \wedge d$ and therefore also $\text{red}(c, d)$, if c and d are both SC. We could also create redundant constraints from conjunctions of numeric and propositional conditions, by augmenting the problem with a numeric encoding of propositions and propositional action effects that parallels the original formulation, similar to van den Briel et al. (2007), Coles et al. (2013) and others.

From a practical perspective, Alg. 1 can be modified to account for redundant constraints. In our implementation, we compute the redundant constraints for each action's preconditions upfront, then treat them as regular numeric conditions.

It is very easy to see that $\hat{h}^{rmax}_{hbd}$ dominates $\hat{h}^{max}_{hbd}$, as the redundant constraints simply extend the set of conditions that are maximised over in Eq. 15. There is, however, a computational overhead: the number of conditions to evaluate, and therefore the number of updates to consider in Alg. 1, is potentially much larger. Finally, it is also easy to see that $\hat{h}^{rmax}_{hbd}$ remains admissible: since the redundant constraints are implied by the satisfaction of the original conjunction, making them explicit does not reduce the space of possible solutions. However, in the inadmissible heuristic $\hat{h}^{radd}_{hbd}$, the addition of redundant constraints may again worsen the problem of over-counting, and unnecessarily inflate the heuristic estimate.

Redundant constraints have proven useful to prune the search space of invalid partial solutions in areas such as constraint-based scheduling (Getoor, Ottosson, Fromherz, & Carlson, 1997). Our experimental analysis will show that their use in numeric planning can be beneficial as well.

In the following two sections we introduce a different way of tackling conjunctive goals. This new method uses a different formulation of our notion of a possible achiever, considering sets of actions as achievers of conjunctions of (numeric) conditions. In Section 9 we further extend this notion to hybrid achievers of conjunctions of numeric and propositional conditions.

8. Generalised Subgoaling: From Atoms to Conjunctions

To show the limits of what can be accomplished with redundant constraints, we extend the "filling the bucket" example from the previous section. Suppose that the agent is not only constrained by a time limit, but also needs to satisfy a constraint on total cost, $C \leq 10$. The slow pouring action has no cost (i.e., does not increase C). There is also a quicker but more expensive action *fill*, fills the whole bucket (i.e., $L = L + 11$) in just one second ($T = T + 1$) but at a cost of 11 ($C = C + 11$). The problem remains unsolvable, but the introduction of pair-wise redundant constraints shown in the previous section fails to capture this situation: The filling action can achieve the constraint $\text{red}(L \geq 11, T \leq 10) = L - T - 1 \geq 0$, and the pouring action is an achiever of $\text{red}(L \geq 11, C \leq 10) = L - C - 1 \geq 0$. Even combing all three subgoals yields the redundant constraint $L - T - C + 9 \geq 0$, which is satisfied in the initial state. Therefore, even equipped with redundant constraints, the relaxation fails to detect that the problem is unsolvable. Again, the problem lies in the difficulty of predicting whether and how a set of conditions can all be simultaneously satisfied in some future state.

Previous work in propositional planning proposed to deal with conjunctive reasoning through mutex relations (Blum & Furst, 1997), and through higher-order versions of the subgoaling relaxation, such as the h^m family (Haslum, 2009). In this section, we introduce a method that shares the objective of strengthening the relaxed reasoning over conjunctions, but focusing on the numeric components of numeric planning problems and exploiting, as much as possible, the numeric properties underlying the formulation. We propose a formulation that can be implemented in a tractable algorithm. Intuitively, we achieve these objectives by capturing simultaneous validity within a constraint-based representation that still adopts the decomposition principle given by the subgoaling relaxation. The key idea is to lift the subgoaling principle from the single atomic subgoals to conjunction of subgoals. To achieve that, rather than seeking optimal values by minimising over different regressions action by action (Haslum, 2009), we propose to bundle several *m-times effect regressor*s together into a single mathematical formulation that can be delegated to an LP solver. Intuitively, our formulation approximates the cost of achieving a conjunction, such as an action's preconditions, and it is used in an incremental fashion. Each time a new action is made reachable because its preconditions become relaxed satisfied, this action will be used, together with the actions already reached, in order to reach new preconditions at potentially lesser cost. As we will see, this yields a tighter, but computationally more expensive, relaxation than those we have presented so far. In the following subsections, we develop this idea through the notion of conjunctive achiever, which we then use to design a new admissible heuristic.

For ease of presentation, in the rest of this section we consider only conjunctions of simple numeric conditions. We extend this to conjunctions of mixed numeric and propositional conditions in Section 9. In the same way as we have shown earlier in the paper, the more general case of arbitrary NNF formulae can be dealt with recursively, by minimising each formula in a disjunction separately.

8.1 Conjunctive Achievers

Intuitively, a *conjunctive achiever* is a set of actions that can be used to achieve a conjunction of numeric conditions. Formally:

Definition 7 (Conjunctive Achiever)**.** *Given a conjunction $\psi \doteq \psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_n$ and a state s, the set $A \doteq \{a_1, .., a_k\}$ is said to be a conjunctive achiever of ψ iff there exists $m_1, ..., m_k \in \mathbb{N}$ such that for all i, $s \models \psi_i^{r(a_1,m_1)^{\cdots^{r(a_k,m_k)}}}$*

The crucial difference w.r.t. the notion of a possible achiever (Def. 5) is that a conjunctive achiever set considers not only what (and how many) actions are needed to achieve each condition, but also how these actions affect the other conjuncts via negative and/or positive side-effects. As it can be seen from the definition, the statement asks for values for the action counters such that the *m-times effect regressor* of each conjunct through all actions, with the given repetitions, is satisfied. This means each conjunct is regressed through not only those actions that are needed for that specific condition, but also those that are necessary to achieve some other conjunct. However, the existence of a conjunctive achiever is still a relaxation of the reachability of the conjunction, since the regression does not consider whether there is any order that allows the multi-set of actions that make up the conjunctive achiever to be executed without violating any action's precondition. We will use $\Xi(A', s, \psi)$ to denote the set of possible conjunctive achievers of ψ from a given state s using only the set of actions A'. (We will use conjunctive achievers obtained from a restricted set of actions in the definition of the heuristic and incrementally add to this set new actions found to be relaxed reachable - see later Algorithm 2.)

Because we are still focusing on the simple numeric condition case and are not considering action preconditions, the order in which the regressions are applied is irrelevant:

Proposition 8.1. *Let $A \doteq \{a_1, .., a_k\}$ be a conjunctive achiever of $\psi \doteq \psi_1 \wedge \psi_2 \wedge, ..., \wedge \psi_n$. The order in which actions are regressed does not matter, i.e., given $m_1, ..., m_k \in \mathbb{N}$:*

$$\psi_i^{r(a_1,m_1)^{r(a_2,m_2)^{\cdots^{r(a_k,m_k)}}}} \equiv \psi_i^{r(a_2,m_2)^{r(a_1,m_1)^{\cdots^{r(a_k,m_k)}}}} \equiv \psi_i^{r(a_k,m_k)^{r(a_{k-1},m_{k-1})^{\cdots^{r(a_1,m_1)}}}} \equiv ...$$

Proof. This is a direct consequence of pair-wise action commutativity shown in Section 4.3. □

Our interest is not only in determining the existence of conjunctive achievers, but also in determining *optimal* conjunctive achievers, that is those minimising the sum of action costs; the computation of this cost is crucial for getting valid lower bounds. For this task also, we formulate a Mixed Integer Program, Eq. 18 below. Input to the MIP formulation is the set of conditions, ψ_i, in the conjunction, the set of actions A' that may be part of the conjunctive achiever (this may be the entire set of actions, or a subset), and the state s from which it is computed. The solution to the MIP determines the action counters $\{m_a\}_{a\in A'}$, and implicitly also the subset of actions that make up the conjunctive achiever (those for which $m_a > 0$).

$$\begin{aligned} \text{minimise} \quad & \sum_{a\in A'} \gamma(a) m_a \\ \text{subject to} \quad & \sum_{x\in X} w_{\psi_1,x} \left(\sum_{a\in A'} m_a k_{a,x} + [x]^s \right) + k_{\psi_1} \rhd 0 \\ & ... \\ & \sum_{x\in X} w_{\psi_n,x} \left(\sum_{a\in A'} m_a k_{a,x} + [x]^s \right) + k_{\psi_n} \rhd 0 \\ & m_a \in \mathbb{N} \;\; \forall a \in A' \end{aligned} \tag{18}$$

The MIP in Eq. 18 has one constraint for each conjunct ψ_i of the conjunction, corresponding to the condition that $s \models \psi_i^{r(a_1,m_1)\cdots r(a_k,m_k)}$. The decision variables of the MIP are m_a, for each $a \in A'$, and the remaining constants in the inequalities are taken from the conjunct ($w_{\psi_i,x}$, k_{ψ_i}), actions ($k_{a,x}$, $\gamma(a)$), and the state s ($[x]^s$). The sum over actions in each constraint links each action's contribution with the number of repetitions required to achieve the corresponding conjunct. Note that in this sum, actions are considered regardless of whether they are possible achievers of the conjunct or not. This way, if an action is required to achieve some conjunct ψ_j, its (possibly negative) effects are also considered in the condition for satisfying every other conjunct ψ_k. Therefore, even conditions already true in the state where the heuristic is calculated, if negated by some action in the conjunctive achiever may require further actions to bring them in check again. The objective function simply sums the repetitions of each action that is used, weighted by the action's cost.

To illustrate, let us continue the "filling the bucket" problem from the beginning of this section. The conjunction for which we need to compute the optimal conjunctive achievers is $(T \leq 10) \wedge (L \geq 11) \wedge (C \leq 10)$. Formulating the MIP for this conjunction with actions $\{\mathit{pour}, \mathit{fill}\}$ (both having a cost of 1) and the initial state $\langle L=0, T=0, C=0\rangle$ results in the following:

$$\begin{array}{lll}
\text{minimise} & m_{\mathit{pour}} + m_{\mathit{fill}} & \\
\text{subject to} & (-1 \cdot m_{\mathit{pour}} + -1 \cdot m_{\mathit{fill}} + 0) + 10 \geq 0 & (\text{to satisfy } T \leq 10) \\
 & (1 \cdot m_{\mathit{pour}} + 11 \cdot m_{\mathit{fill}} + 0) - 11 \geq 0 & (\text{to satisfy } L \geq 11) \\
 & (-1 \cdot m_{\mathit{pour}} + -11 \cdot m_{\mathit{fill}} + 0) + 10 \geq 0 & (\text{to satisfy } C \leq 10)
\end{array}$$

This MIP does not admit any solution, showing that there is no conjunctive achiever using only these two actions.

Given a state s, conjunctive condition ψ and set of actions A', we denote by $\Xi^*(A', s, \psi)$ the cost of the minimum-cost conjunctive achiever of ψ from s using only actions in A'; in other words $\Xi^*(A', s, \psi)$ is the optimal objective value of the MIP in Eq. 18. As in the case of achievers of a single condition, the optimal objective value of the MIP can be lower-bounded by its continuous relaxation, allowing fractional action repetitions, which is a linear program and thus can be solved in polynomial time. We denote the optimal value of the continuous relaxation with $\widehat{\Xi}^*(A', s, \psi)$. Unfortunately, it is not clear whether there are more practical ways of computing $\widehat{\Xi}^*(A', s, \psi)$, or any other reasonable lower bound on $\Xi^*(A', s, \psi)$, than by solving this LP (e.g., using actions in isolation). We leave this aspect for future investigation. The next subsection shows how conjunctive achievers are incorporated into the $h_{hbd}^{\max}$ formulation to devise a concrete admissible heuristic.

8.2 The h^{gen} Formulation

Given a planning problem $\Pi \doteq \langle I, A, G, C, F, X, \gamma\rangle$ only involving simple numeric conditions, we define an admissible estimate which exploits the notion of conjunctive achievers and which we call the *Generalised Subgoaling heuristic* or h^{gen}. This estimate is defined by substituting the conjunctive case within Eq. 2, leading to the following equation:

$$h^{gen}(s,\psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \widehat{\Xi}^*(A^+,s,\psi) + \min\limits_{\substack{a\in A \\ \exists\psi'\in\psi | a\in ach(s,\psi')}} h^{gen}(s,\text{pre}(a)) & \text{if } \psi \text{ is } \wedge \end{cases} \tag{19}$$

The first term lower-bounds the cost of satisfying formula ψ, and is the optimal objective value of the continuous relaxation of the MIP in Eq. 18. Same as in the definition of $h^{\max}_{hbd}$, the action set A^+ is the set of actions that have a finite estimated precondition cost, i.e., $A^+ = \{a \in A \mid h^{gen}(s,\text{pre}(a)) < \infty\}$. The second term in Eq 19 strengthens the bound by including the lowest precondition cost among all achiever actions for the conditions in ψ.

Theorem 8.1 (Admissibility)**.** *For simple numeric planning problems, h^{gen} is an admissible heuristic. Let s be a state and ψ an NNF formula. Then $h^{gen}(s,\psi) \leq h^*(s,\psi)$.*

Proof.

We first consider the case when ψ is a conjunction of SCs, by induction on cost. (We assume for simplicity that action costs are strictly positive; to account for zero-cost actions, the induction order can be modified by tie-breaking on plan size.) Consider an optimal plan for achieving ψ from s, and let M be the multi-set of actions appearing in this plan. Let $M' \subseteq M$ be the multi-set of actions whose effects are necessary and sufficient to achieve ψ, disregarding the achievement of actions' preconditions. Unless $s \models \psi$ (in which case $h^{gen}(s,\psi) = h^*(s,\psi) = 0$), M' is non-empty. The set of actions in M', with action counters set to the number of repetitions of each action in the plan, is a conjunctive achiever of ψ, and therefore its total cost, C' is greater than or equal to the optimal solution, C, to the continuous relaxation of the corresponding MIP 18. Every action a that has a non-zero count m_a in the optimal solution to MIP 18 is an achiever of some $\psi_i \in \psi$; if it were not, the action count could be removed from the conjunctive achiever (i.e., m_a set to 0) without violating any of the constraints, resulting in a conjunctive achiever with lower cost. Let $B = \min_{a\in A:\exists\psi'\in\psi, a\in ach(s,\psi')} h^{gen}(s,\text{pre}(a))$. If $M' = M$, i.e., there are no actions in the optimal plan other than those in the conjunctive achiever M', then at least one of these is applicable in s; hence $B = 0$. Otherwise, there is some action a' in M' such that one or more of the actions in $M \setminus M'$ are necessary to achieve $\text{pre}(a')$; since $h^*(s,\text{pre}(a')) < h^*(s,\psi)$ we have $h^{gen}(s,\text{pre}(a')) \leq h^*(s,\text{pre}(a'))$ by inductive assumption, and since a' is an achiever of some $\psi_i \in \psi$, $B \leq h^{gen}(s,\text{pre}(a'))$. Thus, $h^{gen}(s,\psi) = C + B \leq h^*(s,\psi)$.

The case of an NNF formula with SCs as atomic conditions follows simply from the fact that h^{gen} (like $h^{\max}_{hbd}$) minimises cost over disjunctions. □

We can strengthen the h^{gen} estimate in the same way as we did $h^{\max}_{hbd}$, by further restricting the action set A^+ used in $\widehat{\Xi}^*(A^+,s,\psi)$ to include only those actions a whose precondition cost does not exceed the cost of achieving ψ without using a. This is obtained by updating the cost of each conjunction in an incremental fashion.

8.3 Computing h^{gen}

Algorithm 2 presents a modified version of the blind search schema in Algorithm 1 that computes h^{gen}. Again, we use a priority queue to explore actions according to their estimated cost η. The search starts with the actions applicable in the state; these are initially pushed in the queue with an associated cost of 0. Each action pulled from the priority queue Q enlarges the set of relaxed reachable actions A'. For each non-expanded action a' whose (conjunctive) precondition can be achieved using actions in A', the algorithm computes $\widehat{\Xi}^*(A', s, \text{pre}(a))$, the lower bound on the cost of a conjunctive achiever pf pre(a) obtained from the LP-relaxation of MIP 18. Then the action with this associated cost is pushed onto Q and a new iteration starts. The algorithm terminates when the priority queue empties (the problem is not solvable) or when the goal action g_a is pulled out from the queue.

Algorithm 2: Computing $h^{gen}(s, G)$

```
1  $Closed = \emptyset$
2  $A' = \emptyset$
3  foreach $a \in A \cup \{g_a\}$ do
4      if $s \models pre(a)$ then
5          $\eta(a) = 0$
6      else
7          $\eta(a) = \infty$
8  $Q$ = PriorityQueue($\{(a, 0) | a \in A, \eta(a) = 0\}$);
9  while $Q \neq \emptyset$ do
10     $a$ = pull($Q$);
11     if $a == g_a$ then
12         return $\eta(a)$
13     $A' = A' \cup \{a\}$;
14     $Closed = Closed \cup \{a\}$;
15     foreach $a' : a' \notin Closed$ do
16         $\eta(a') = \widehat{\Xi}^*(A', s, pre(a')) + \min_{a'' \in A}(\eta(a''))$;
17         if $a' \in Q$ then
18             update_queue($Q$,$a'$,$\eta(a')$);
19         else
20             push($Q$,$a'$,$\eta(a')$);
21 return $\infty$
```

The algorithm is guaranteed to terminate as the number of actions is finite. The computational cost is polynomially bounded by the number of actions and by the resolution of the LP, which is also polynomial. More formally:

Theorem 8.2. *The computational complexity of h^{gen} is polynomial in the problem size.*

Proof. Note that there will be at most $|A|$ iterations (each action is popped out from the queue at most once), and each iteration i solves an LP at most $(|A| - i)$ times, hence the computational complexity is bounded by $O(|A|^2 \cdot LP)$. Moreover, because the generated LP contains a number of variables that is linear in the number of actions of the problem and a number of constraints that is linear in the size of the conjunctive formula, the overall complexity is polynomial in the size of the problem. □

The next section shows how the heuristic can be combined with propositional variables and global constraints.

9. Hybridisation with Propositional Conditions: Conjunctive Hybrid Achievers

In this section, we extend the notion of conjunctive achievers and the generalised subgoaling heuristic to the hybrid case, where the subgoals in a conjunction can be both simple numeric and propositional conditions. Furthermore, we show how the heuristic integrates in a beneficial way with *global constraints*, which are conditions that must be satisfied in every state traversed by a plan.

9.1 Conjunctive Hybrid Achievers

The extension of conjunctive achievers to hybrid propositional conditions and numeric conditions is straightforward:

Definition 8 (Conjunctive Hybrid Achiever)**.** *Given a conjunction $\psi \doteq \psi_1 \wedge \psi_2 \wedge, ..., \wedge \psi_n$ of literals and simple numeric conditions, and a state s, the set $A \doteq \{a_1, .., a_k\}$ is said to be a conjunctive hybrid achiever, or simply a* hybrid achiever *of ψ iff there exists $m_1, ..., m_k \in \mathbb{N}$ such that for every $\psi_i \in \psi$*

- *if ψ_i is a numeric condition, then $s \models \psi_i^{r(a_1,m_1)^{\cdots^{r(a_k,m_k)}}}$*
- *if ψ_i is a literal, and $s \not\models \psi_i$ then $\sum_{j|a_j \in ach(s,\psi_i)} m_j \geq 1$*

Intuitively, a hybrid achiever must contain actions whose effects, if executed a sufficient number of times, are sufficient to satisfy each conjunct, numeric and propositional. For propositional conditions we only require that they are achieved by at least one action, if not already satisfied in the current state. We use $\Xi_{hbd}(A', s, \psi)$ to represent the set of hybrid achievers for ψ from a given state s, using actions from set A'.

The existence of a conjunctive hybrid achiever is a necessary, but not sufficient, condition for ψ being reachable: It does not take into account whether the actions' preconditions are reachable, or if some actions in the achiever have negative effects on the precondition of others. There is a certain asymmetry, in that negative side-effects on numeric subgoals are considered, but not on propositional conditions. For example, consider the conjunction $p \wedge (x \geq 0)$: if the action achieving p also decreases x (below 0), a hybrid achiever for the conjunction will be forced to also include actions that restore $x \geq 0$. This way, some of the negative interference will be taken into account. On the other hand, if the action required to increase x deletes p, this is not considered. The reason for this is that effects on simple numeric conditions commute, but propositional effects, which are assignments, do not.

Given the conjunction $\psi \doteq \psi_1 \wedge \ldots \wedge \psi_n \wedge \ldots \wedge l_1 \wedge \ldots \wedge l_t$ where ψ_i are simple numeric conditions and l_j propositional literals, the least-cost hybrid achiever can be computed by the following slight revision of the MIP formulation from Eq. 18:

$$\begin{aligned}
\text{minimise} \quad & \sum_{a \in A'} \gamma(a) \cdot m_a \\
\text{subject to} \quad & \sum_{x \in X} w_{\psi_1,x} \left(\sum_{a \in A'} m_a k_{a,x} + [x]^s \right) + k_{\psi_1} \rhd 0 \\
& \ldots \\
& \sum_{x \in X} w_{\psi_n,x} \left(\sum_{a \in A'} m_a k_{a,x} + [x]^s \right) + k_{\psi_n} \rhd 0 \\
& \sum_{a \in A' \cap ach(s,l_1)} m_a \geq \begin{cases} 1 & \text{if } [l_1]^s = \bot \\ 0 & \text{otherwise} \end{cases} \\
& \ldots \\
& \sum_{a \in A' \cap ach(s,l_t)} m_a \geq \begin{cases} 1 & \text{if } [l_t]^s = \bot \\ 0 & \text{otherwise} \end{cases} \\
& m_a \in \mathbb{N} \ \ \forall a \in A'
\end{aligned} \tag{20}$$

The only difference between this MIP and the earlier is the presence of the constraints ensuring that for each propositional literal l_i that is not already satisfied in state s, at least one achiever is executed at least once (i.e., $m_a \geq 1$). Synergies between achievers are accounted for automatically, since the decision variables (action counters) are shared between all constraints. That is, if some action contributes to achieving multiple conjuncts, whether propositional or numeric, the optimal solution to the MIP still determines the minimum number of repetitions of the action that is needed.

Again, we denote the cost of a minimum-cost hybrid achiever, i.e., the optimal objective value of the MIP in Eq. 20, with $\Xi^*_{hbd}(A', s, \psi)$, and the optimal objective value of the continuous relaxation of the MIP with $\widehat{\Xi_{hbd}}^*(A', s, \psi)$. With this in hand, the hybrid version of h^{gen}, denoted h^{gen}_{hbd}, is defined analogously:

$$h^{gen}_{hbd}(s, \psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \widehat{\Xi_{hbd}}^*(A^+, s, \psi) + \min\limits_{\substack{a \in A \\ \exists \psi' \in \psi | a \in ach(s,\psi')}} h^{gen}_{hbd}(s, \text{pre}(a)) & \text{if } \psi \text{ is } \wedge \end{cases} \tag{21}$$

This equation is identical to Eq. 19 but for the fact that we replace $\widehat{\Xi}^*(A^+, s, \psi)$ with $\widehat{\Xi_{hbd}}^*(s, \psi)$; in practice, this means that the minimum-cost hybrid achiever is computed using the LP-relaxation of MIP 20. Note also that the minimum precondition cost in the second term is over all achievers of any subgoal in ψ, propositional as well as numeric. It is easy to see that h^{gen}_{hbd} is also admissible, and that its computation is polynomial in the problem size.

9.2 Enforcing Global Constraints

Global constraints are conditions that must be satisfied in every state traversed by the plan. In other words, they act as additional preconditions of every action, and as a conjunct to the goal. Global constraints are a convenient way to express, for example, bounds on variables that model resources, or plan safety constraints.

Global constraints are neglected in the standard h^1 relaxation because they are already satisfied in each state where the heuristic is computed. However, because conjunctive hybrid achievers support conjunctions of numeric conditions, and take into account some negative interactions, it suffices to adjoin the global constraints (assuming these also are conjunctions of simple numeric conditions) with any conjunctive condition that is evaluated. The estimated cost of achieving the condition will then also account for any additional cost of keeping the global constraint satisfied. Formally, only a small change to the h^{gen}_{hbd} equation is needed:

$$h^{gen,C}_{hbd}(s,\psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \widehat{\Xi_{hbd}}^{*}(A^+, s, \psi \wedge C) + \min\limits_{\substack{a \in A \\ \exists \psi' \in \psi | a \in ach(s,\psi')}} h^{gen,C}_{hbd}(s, \text{pre}(a)) & \text{if } \psi \text{ is } \wedge \end{cases} \tag{22}$$

where C is the global constraint, a conjunction of simple numeric constraints. The only change from Eq. 21 is that instead of an estimate of the cost of a hybrid achiever of ψ, we use the estimated cost of a hybrid achiever of $\psi \wedge C$, the conjunction of the condition and the global constraint. The heuristic remains admissible: The addition of global constraints is equivalent to a problem reformulation where each action's precondition and the goal are replaced with their conjunction with C.

10. Handling Assignment Operations in Subgoaling

The restriction to simple numeric conditions does not allow for action effects that assign a numeric variable a constant value. To be simple, a condition can only be affected by constant increases and decreases. This can be a limitation of the heuristic for domains having even very simple assignments. For instance, consider the case in which one wants to model the action of replenishing a water tank to its full capacity T. This can be modelled in numeric planning with an action, "refill", that sets the amount v of water in the tank with the constant numeric effect $\langle v, :=, T\rangle$.

Let us focus on the case where each action has at most one effect that interacts with a linear numeric condition, and this effect is an assignment with a constant right-hand side. In this situation it is possible to use the Effect-Regressor $\psi^{r(a)}$ to infer whether action a achieves ψ or not. More precisely, let $e \doteq \langle x_k, :=, k_{a,x_k}\rangle$ be the numeric effect of a. The regression of the linear condition $\psi \doteq \langle \sum_{x \in X} w_{\psi,x}x + k_\psi, \rhd, 0\rangle$ through a yields a condition:

$$\psi^{r(a)} = \langle \sum_{x \in X \setminus \{x_k\}} w_{\psi,x}x + k_\psi + w_{\psi,x_k}k_{a,x_k}, \rhd, 0\rangle \tag{23}$$

Since in this case the action a is idempotent, we can say that a is an achiever of ψ in a state s iff $s \models \psi^{r(a)}$. Moreover, noting that $\rhd \in \{\geq, >\}$, we can also observe that a necessary (but insufficient) state-independent condition for a to be an achiever for ψ is that $w_{\psi,x_k}k_{a,x_k} \geq 0$. When this holds we will say, with a slight abuse of terminology, that a is an *assignment achiever* for ψ. Similarly to what we did for Simple Numeric Conditions, this notion can be used in a relaxation schema, extending the subgoaling heuristics seen so far to also account for what we call Direct Assignment Conditions:

Definition 9 (Direct Assignment Condition.). *Let $\psi \doteq \langle \sum_{x \in X} w_{\psi,x} x + k_\psi, \rhd, 0 \rangle$. ψ is said to be a* Direct Assignment Condition *(DC) if there is only one achiever of it, and this achiever has a single numeric effect which is an assignment with a constant value.*

Direct Assignment Conditions can be used in the recursive schema of $\hat{h}^{max}_{hbd}$ and $\hat{h}^{add}_{hbd}$, by devising a specific treatment that differs from that used for Simple Numeric Conditions. More precisely, when a Direct Assignment Condition ψ is not satisfied in a state s, we check the satisfaction of the regressed condition through its assignment achiever $ach(\psi)$ (which is unique given Def. 9) and use the recursive call to get an estimate of the achiever's precondition cost. For convenience, we report the extended formulation in Equation 24, where bold is used for highlighting the new supported case. The additive heuristic is extended in the analogous manner.

$$\hat{h}^{max}_{hbd}(s,\psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(s,\psi)} \left(\hat{h}^{max}_{hbd}(s, \text{pre}(a)) + \gamma(a) \right) & \text{if } \psi \in \mathsf{PC}s \\ \min\limits_{a \in R(A,s,\psi)} \widehat{\text{rep}}(a,\psi,s) \cdot \gamma(a) + \min\limits_{a \in R(A,s,\psi)} \hat{h}^{max}_{hbd}(s, \text{pre}(a)) & \text{if } \psi \in \mathsf{SC}s \\ \boldsymbol{\hat{h}^{max}_{hbd}(s, \text{pre}(ach(\psi))) + \gamma'(ach(\psi), \psi, s)} & \text{if } \psi \in \mathsf{DC}s \\ \min\limits_{\psi' \in \psi} \hat{h}^{max}_{hbd}(s,\psi') & \text{if } \psi \text{ is } \vee \\ \max\limits_{\psi' \in \psi} \hat{h}^{max}_{hbd}(s,\psi') & \text{if } \psi \text{ is } \wedge \end{cases} \tag{24}$$

In this formulation, $\gamma'(a,\psi,s)$ is a function that given the condition of interest and a state, returns the cost $\gamma(a)$ of the action if $s \models \psi^{r(a)}$, and ∞ otherwise. In other words, the actual achievement of a direct condition is determined when the recurrence relation is solved.

Proposition 10.1. *If all numeric conditions of the problem are either Direct Assignment Conditions or Simple Numeric Conditions, $\hat{h}^{max}_{hbd}$ provides an admissible estimate. Likewise, $\hat{h}^{add}_{hbd}$ is safe-pruning.*

Proof. The optimal plan to achieve a Direct Assignment Condition is lower-bounded by the cost of its achiever (which is unique) plus the lower bound on the cost of its precondition. As matter of fact, if we take as inductive hypothesis in $\hat{h}^{max}_{hbd}$ that the precondition cost of each action is indeed a lower bound, it is easy to prove the lower bound for the Direct Assignment Condition. This can in fact be either (i) supported by the state where the heuristic is computed, or is supported (ii) by at least one execution of its unique achiever. For case (i) we get the obvious lower bound of 0; for case (ii) we know that since the precondition cost of each action computed by $\hat{h}^{max}_{hbd}$ is a valid lower bound to get to this precondition, the sum of this cost plus the cost of executing the achiever for a Direct Assignment Condition is a lower bound for achieving the Direct Assignment Condition too. There is in fact no alternative path to get to it. Finally, as $\hat{h}^{add}_{hbd}$ is equivalent to $\hat{h}^{max}_{hbd}$ in reachability, it will always be safe-pruning. $\square$

It is very likely that more general conditions could be handled by extending the theory proposed in this section. For instance, one can think of generalising to the case in which the achiever set is not a singleton, but where each action in this set is restricted to affect the same variable (but may have quite different preconditions and effects). We leave this and other extensions as future work.

11. Integration with AIBR for More General Action Models

By integrating our results with other relaxation schema, it is possible to handle more general numeric conditions, even when these are affected by actions with non-constant additive effects. In the following definition, we further partition the numeric condition set to differentiate between simple, direct assignment and Hard Numeric Conditions (HC). HCs denotes the set of numeric conditions that are neither Simple Numeric Conditions (Def. 6) nor Direct Assignment Conditions (Def. 9). We handle HCs using the relaxed solution given by the interval-based relaxation (Scala et al., 2016b).

$$\hat{h}^{add}_{hbd+}(s,\psi) \doteq \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(\psi)} (\hat{h}^{add}_{hbd+}(s,\text{pre}(a)) + \gamma(a)) & \text{if } \psi \in \mathsf{PC}\text{s} \\ \min\limits_{a \in A^+} (\widehat{\text{rep}}(a,\psi,s) \cdot \gamma(a) + \hat{h}^{add}_{hbd+}(s,\text{pre}(a))) & \text{if } \psi \in \mathsf{SC}\text{s} \\ \hat{h}^{add}_{hbd}(s,\text{pre}(ach(\psi))) + \gamma'(ach(\psi),\psi,s) & \text{if } \psi \in \mathsf{DC}s \\ \sum\limits_{\substack{a \in \pi' \\ \pi' \in sol(<s^+,A^+,G,X>)}} (\hat{h}^{add}_{hbd+}(s,\text{pre}(a)) + \gamma(a)) & \text{if } \psi \in \mathsf{HC}\text{s} \\ \sum\limits_{\psi' \subset \psi : |\psi'|=1} \hat{h}^{add}_{hbd+}(s,\psi') & \text{if } |\psi| > 1 \end{cases} \tag{25}$$

The subscript *hbd+* denotes the support for HCs. Each subgoal type (PCs, SCs, DCs, HCs) is treated using the appropriate specialised reasoning, independently of other subgoals.

The implementation of the interval-based relaxation follows our previous work (Scala et al., 2016b). The procedure is split into a reachability and a plan computation phase. The former performs a fix-point analysis using actions reachable for $\hat{h}^{add}_{hbd+}$ (A^+), which returns the actions *sufficient* to reach each complex condition, and then preconditions are recursively evaluated using $\hat{h}^{add}_{hbd+}$.[10] 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). Note that, in this formulation different relaxations can interleave in computing the cost of a subgoal; if in fact an action includes some simple numeric precondition, the evaluation is done with the more accurate estimate.

From a computational standpoint, the integration of the AIBR is achieved by calling the relaxation and its estimate whenever an action contributing (indirectly or directly) to the condition is popped out from the queue in Alg. 1. This can be expensive and we leave more efficient implementation as object of study for future work.

10. As a difference w.r.t. previous work in this context (Aldinger et al., 2015; Hoffmann, 2003), the reachability analysis in (Scala et al., 2016b) does not require the transitive closure of the action effects dependency relation.

Also note here that the formulation hereby presented does not consider the notion of conjunctive achievers. Although possible from a theoretical standpoint, we have not implemented this aspect yet and it is not clear how efficient and practical it could be.

An admissible formulation ($\hat{h}^{max}_{hbd+}$) can be obtained by using any admissible approximation of the interval-based relaxation for HCs – the simplest one 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.

12. Related Work

Planning with numeric information has been tackled by others using different approaches, two of which have gained the majority of the attention: heuristic search, and compilation to other models. Our related work discusses thoroughly the heuristic search approach which is the closest to our method. We then draw connections with compilation approaches that have been used in other contexts, but share similarities with our techniques. We conclude with a discussion of other, less similar approaches.

12.1 Numeric Planning via Heuristic Search

Heuristic search is a well established approach to solving planning problems (Pearl, 1984; Geffner & Bonet, 2013), which underlies the majority of the top performer planners at the International Planning Competitions (Vallati, Chrpa, Grzes, McCluskey, Roberts, & Sanner, 2015). One of the powerful ideas behind previous and current approaches is to devise heuristics from a computationally effective relaxation or abstraction of the targeted problem. In particular, the solution of the relaxed problem can be used as a means to estimate the actual distance or cost to the problem goal

Pioneered by Hoffman (2003), the interval-based relaxation (Aldinger et al., 2015; Gregory, Long, Fox, & Beck, 2012) has been the principle most used to extract such a heuristic information in the numeric extension to classical planning. These heuristics come in different flavours, including as metric extensions of the relaxed planning graph (Koehler, 1998; Hoffmann, 2003; Gerevini, Saetti, & Serina, 2008; Coles et al., 2012, 2013).

The interval-based relaxation approximates the value of each numeric state variable using an interval enclosing all the *possible* values that this variable may eventually reach. Intervals for expressions are calculated using Interval Analysis (Moore, Kearfott, & Cloud, 2009), and a numeric condition is satisfied if at least one value in the expression's interval satisfies the condition. Applicable action effects monotonically widen the bounds of the intervals, hence a *relaxed* solution can be computed by extending intervals until the goal is reached, and then extracting from these intervals a set of relaxed actions. Monotonicity is achieved taking the convex union between the interval representing the possible values of the variable before the action execution, and the interval representing the effect expression. It follows that any condition previously satisfied will also be satisfied in the successor state.

Aldinger et al. (2015) showed that termination and safe pruning can be ensured when the dependency relation between numeric variables in the planning task is acyclic. For such tasks, the set of reachable states are the fixed point intervals computed by executing applicable actions as many times as necessary, in the order defined by the dependency relation between numeric variables. Scala et al. (2016b) subsequently removed this restriction and extended the interval-based relaxation to numeric planning tasks with cyclic dependencies, as long as numeric effects are additive. In addition, they observed that any planning task can be transformed to have only additive effects, at the price of a further relaxation of state dependent assignment operations.

Our approach differs from this relaxation in a very substantial way. The subgoaling technique we present here reasons about *relevant subgoals*, by regression, instead of reachable values. As Keyder and Geffner (2008) point out regarding the classical case, cost-sensitive additive and inadmissible estimates can be obtained this way without a planning graph. By approaching the problem using a decomposition approach, the present work has been the first to provide a heuristic design leading to admissible estimates for numeric planning.

Analogously to the classical delete-relaxation, the interval-based relaxation fails to capture any negative interactions between numeric variables. Coles et al. (2013) proposed the use of a Mixed-Integer Program (MIP) to tighten the bounds on intervals for problems that model *resources* as numeric variables and feature only specific patterns of interaction between preconditions and effects (actions are *consumers* or *producers* of resources). This addresses one important weakness of interval-based heuristics, by forcing intervals to capture changes to resource variables. However, this does not account for other kinds of negative interactions, such as those illustrated by the Sailing domain. The heuristic and the planner developed by Coles et al., namely LPRPG, demonstrates good performance over this specific class of numeric planning problems, which is a subclass of simple numeric planning. It is, however, not applicable to any problem outside of the producer–consumer class. It is an open and interesting question how resource-flow reasoning can be effectively integrated into our subgoaling relaxation; a possible way to do so would be by injecting the resource flow constraints arising from the producer–consumer pattern in the constraints of the h^{gen} formulation. We leave this aspect as a future work.

Eyerich et al. (2009) proposed another approach to devising heuristics for numeric planning, which extends the context-enhanced additive heuristic. In this approach, numeric conditions are justified only qualitatively, completely abstracting away from the counting problem. While this can be powerful in domains where only one repetition of numeric actions is necessary (e.g., in ZenoTravel the refuelling action only needs to be executed once), in many other domains this abstraction is too coarse and looses all numeric information in the problem. Temporal fast downward, which is the heuristic search planner employing this heuristic, was not able to solve any of the instances of our Sailing problem. Illanes and McIlraith (2017) also use a qualitative abstraction of the numeric aspects of the problem. They construct a classical planning problem by replacing numeric conditions with propositions and use a classical planner to devise a partial policy, which is used to guide the search for a plan in the numeric planning problem. Our subgoaling approach captures numeric structures, enabling inferences that would be difficult to do with only propositional variables. However, it should be possible to integrate our heuristics with abstraction approaches so as to accelerate even further the search. This is an aspect definitely worth exploring in future work.

Regression – the ability to find sufficient and necessary conditions under which an action achieves a given goal – has been widely used in many areas of planning (McDermott, 2003; Fikes, Hart, & Nilsson, 1972; Hoare, 1969; Rintanen, 2008; Scala, 2014). The very first heuristic that makes use of regression to estimate goal distance is due to McDermott (1996); this idea has been used extensively in the work by Bonet and Geffner (2001) and in the work by Haslum and Geffner (2000). Our approach builds on the declarative and decomposition-based formulation of heuristics of the latter two.

12.2 SMT Based Approaches

Another successful approach to numeric planning is based on the compilation of the planning problem into the problem of finding a model satisfying a given SMT formula (Scala, Ramírez, Haslum, & Thiébaux, 2016c; Hoffmann, Gomes, Selman, & Kautz, 2007; Shin & Davis, 2005). As for SAT-based planning in the classical planning case (Kautz & Selman, 1999; Rintanen, 2012), SMT-based planning stems from the idea of capturing all the possible trajectories for a bounded version of the planning problem in a quantifier-free formula having numeric and propositional conditions as terms. Such a formula is responsible for capturing all those transitions that are consistent with the dynamics of the system (represented in the actions), the initial state and the goals to be achieved. A challenging aspect for this approach is to find a compact representation of such possible trajectories; this is usually obtained by allowing multiple actions to share the same timepoint, and reconstructing the sequence of operations only when a model for the formula is found. To achieve this objective, previous work has exploited different ways of decomposing the problem, e.g. through the identification of independent actions, or encodings that encapsulate ordered sequences of actions (Wehrle & Rintanen, 2007). Whilst this has been a fruitful line of work in classical planning that has led to efficient SAT-based planning systems (Rintanen, 2012), less has been done, relatively speaking, in the context of numeric planning.

There are two exceptions worth mentioning. In our previous work (Scala et al., 2016c), we have used the same m-times regressor as in this paper to capture unbounded sequences of actions within a single time step, and found conditions under which these sequences of actions represent valid transitions. Devising powerful heuristics can also be beneficial in the framework of compilation into other theories. This has been shown elsewhere (Rintanen, 2012).

Looking into the potential of integrating planning as satisfiability, optimisation and mathematical programming is an old idea, of which two early examples are the LPSAT planner, due to Wolfman and Weld's (2001), and Shin and Davis's (2005) TMLPSAT. Both planners are built around "proto"-SMT solvers, based on a decision procedure, either DPLL (Davis, Logemann, & Loveland, 1962) or CDCL (Zhang, Madigan, Moskewicz, & Malik, 2001), suitably modified so that some clauses are not represented in propositional logic, but in an auxiliary theory, and still able to take advantage of both unit propagation and conflict-directed clause learning. Both planners are strictly more expressive than any of the planners we propose in this paper, as we consciously trade-off expressiveness for performance. A "second wave" of approaches based on this integration followed the formalisation of SMT into *abstract decision procedures* (Nieuwenhuis, Oliveras, & Tinelli, 2006; Kroening & Strichman, 2008) and the availability of powerful, scalable implementations of important fragments of first-order logic such as linear constraints and difference logic (Dutertre & de Moura, 2006). Examples of these later systems, which all use slight reformulations of Shin and Davies' encoding, are dReal (Bryce, Gao, Musliner, & Goldman, 2015) and

SMTPLAN (Cashmore et al., 2020). Parallel to these, other planning systems exist which depart significantly from this line of research, that also support in limited, incidental ways, numeric effects like the ones discussed in this paper. Inspired by well-established practices in Optimisation (Hooker, 2003), there is the work due to Rintanen (2017) on *clock-based* SMT encodings for temporal planning with numeric and propositional resources. Also, following up the notion of *chronicles* (Ghallab, Nau, & Traverso, 2004), Bit-Monnot (2018) puts forward a very efficient SMT encoding that compares very favourably with other SMT-based approaches. Note that most of these systems address temporal, as well as numeric, planning. All, though, ultimately struggle with problems where there is a strong interaction between the subproblem of achieving individual goals separatedly, and that of determining their ordering so as to avoid interference between temporal actions and minimise makespan. This was addressed via the integration of domain-independent heuristics into the process of compiling temporal problems into Constraint Programming (CP) (Vidal & Geffner, 2006). In this paper we address directly the problem of goal ordering, or *subgoaling*, in the context of numeric planning, and propose effective heuristic methods that could be used to enhance the capabilities of existing SMT-based temporal-numeric planners in an analogous way as Vidal & Geffner did.

SMT-based planning approaches suffer from the same limitations as SAT-based planning approaches. First, encodings based on Kautz and Selman's (1999) seminal contribution, have trouble dealing with unsolvable planning problems, as they need to test satisfiability for an upper bound on plan length or makespan. Finding a tight upper bound is as hard as solving the planning task itself (Bylander, 1994), and naive upper bounds lead to SAT encodings with huge numbers of variables and clauses. Recent work on alternative SAT-based approaches (Suda, 2014; Eriksson & Helmert, 2020), that rely on Bradley's (2011) notion of Property Directed Reachability (PDR), overcome the limitations inherent in Kautz & Selman's encoding on completeness with respect to refutation, and propose practical algorithms to produce certificates of unsolvability for propositional planning. Second, and for similar reasons, controlling the quality of solution along arbitrary finite horizons while producing succinct SAT encodings is not trivial, as noted by Rintanen (2012). Our formulation can guarantee admissible estimates to find optimal plans also in the numeric setting. Note that the admissible estimates presented in this paper are the first of their kind.

12.3 Similar Problems and Methods

Analysis of discrete, linear time invariant systems. The notion of planning operators to abstract the behaviour [11] of a dynamical system has a long history in planning. An early example, which is closely related to the methods discussed in this paper, is the idea by E. P. D. Pednault (1987) to associate with action effects an integral equation that describes successor states after a set amount of time. In this work we limit ourselves to linear dynamics, and therefore, established results from the theory of discrete-linear systems (Edward, 1996; Borrelli, Bemporad, & Morari, 2017) are particularly relevant to our work. Given a formulation of how a variable evolves given in terms of a linear function of some other variables, the future of this variable is given by the exponentiation of the matrix formed

11. A behaviour in control is the set of trajectories that are compatible with a set of differential *path* constraints.

by the coefficients in the linear functions given. However, a) computing such a matrix exponentiation is computationally expensive, b) the number of possible configurations, and therefore, behaviours of the system, depend on how actions are ordered in plans. The closed-form expression presented in this paper is valid for simple numeric conditions, and in particular the *m-times effect regressor*, as it exploits the factored representation of numeric planning problems. We see this as a special case of the general setting of discrete-linear systems with variables that only change because of actions, and do not interfere (the self-interference effect free assumption). In terms of the theory developed by Borrelli et al. on Model Predictive Control (MPC) for linear systems, this corresponds to having the *natural response* always set to 1 for each variable. We believe that this special case can still capture very interesting problems with guarantees such as computational cost and admissibility if desired. Once this connection has been drawn, we look forward to further refine our understanding of the relationship between the contributions reported in this paper and well-established results from Systems and Optimal Control theory.

Heuristics for classical planning The problem of capturing the number of times an action has to appear in the solution plan is well known in the purely classical planning literature. State of the art heuristics for optimal classical planning are based on the framework of *operator counting* (Pommerening et al., 2015). Our heuristic can be seen as an extension of this framework to the numeric case. Interestingly, it seems that the combination of our heuristic h^{max}_{hbd} with the generalised form h^{gen} captures information in a similar way the lm-cut and the state equations combine with each other (Helmert & Domshlak, 2009; Bonet & Helmert, 2010).Understanding the expressive power of our formulation against the classical planning version, together with understanding how to benefit from the various characterisations in a comprehensive way are however all object of interesting future work.

13. Experimental Analysis

Our experimental analysis has been conducted to evaluate the usefulness of the heuristics presented in this paper for both satisficing and optimal planning. We employed such heuristics in a forward state space search planner guided by two different informed search algorithms: Greedy Best First Search (GBFS) for the satisficing setting, and A^* for the optimal setting. In the satisficing setting ties are broken to favour nodes with lower g-values; in the optimal setting ties are broken to favour nodes with higher g-values. The implementation is part of the ENHSP planning system (`https://sites.google.com/view/enhsp/`), which is a JAVA implementation containing all the necessary functionalities developed. For solving the linear programs in the h^{gen}_{hbd} formulation we used CPLEX 12.6.3.

The impact of heuristics on the planner is measured along two main dimensions: CPU-time to produce a valid (or optimal) solution, and length of the produced plans (for satisficing planning). Since we assess our heuristics in a state-space planner, we also report the number of node expansions as the number of times a node is popped out from the priority queue. Intuitively, the smaller the number of expansions, the less search was required to find a solution. We seek an understanding of whether and when fewer expansions translate into a faster convergence to the solution. Moreover, we want to understand how the inadmissible heuristics behave in terms of plan length. In our experiments, all actions have unit cost, therefore the optimal plans are those with the shortest length.

For comparison, we ran experiments on heuristics based on the Interval Based Relaxation (IBR) and with a simple (yet much faster) goal sensitive heuristic. In the satisficing setting, we take MetricFF (Hoffmann, 2003) as representative of IBR based heuristic search numeric planners. In the optimal setting, the goal sensitive heuristic giving 0 to goal states and 1 to non-goal states is run with our A^* implementation. In our experiments we do not compare with the LPRPG planner (Coles et al., 2013) because its heuristic is designed only for handling producer–consumer resource problems (a subclass of simple numeric planning, see Section 12), and we assume that it will be more effective than search with any of the more general heuristics that we consider for problems that fall within that specific class. Therefore, since the results of an experimental comparison are quite predictable, we do not see it as necessary to carry out that comparison.

Our experiments were run under Ubuntu on an Intel(R) Xeon(R) CPU E3-1240 v3 @ 3.40GHz and 8Gb of Ram with a timeout of 1800 secs. per problem.

In the following, we first introduce the domains used for our experiments; then we discuss the results obtained in the satisficing and optimal planning settings.

13.1 Domains

Our evaluation is carried out i) on a variety of standard numeric domains from the International Planning Competition (IPC)[12] and the LPRPG benchmark suite (Coles et al., 2013), ii) on numeric planning reformulations of Functional STRIPS problems (Francès & Geffner, 2015) (F-STRIPS), and iii) on two sequential numeric domains called Sailing and Farmland introduced in our previous work (Scala et al., 2016a).

From the IPC, we only include domains where numeric variable(s) are included in at least one action precondition or goal.

F-STRIPS's domains, Sailing and Farmland are heavily numeric domains and our evaluation puts a greater emphasis on them. This is to better understand the role played by our heuristics in dealing with the numeric structure of the problem. A detailed description of these domains follow.

13.1.1 F-STRIPS Domains

Counters. This domain revolves around the idea of controlling the values of a set of *counters* $\{X_1, ..., X_n\}$ in order to satisfy a set of constraints. The domain of each counter goes from 0 to a MAX value that changes depending on the number of counters, and is set in order to make the problem solvable. The domain comes with three sets of instances, whose goals always require a total order among the counters i.e.: $\{X_1 < X_2, ..., X_{n-1} < X_n\}$. In the first set of instances (Counters), counters are all initially set to 0, in the second set (Count-Rnd), their values are set randomly, while in the last set of instances (Count-Inv) the values are set in such a way that they all need to be reversed in order for the goal to be satisfied. For the first set we have 11 instances; for the random variant we have 33 instances; for the reversed variant 11 instances. Differently to the original formulation, our counters are represented using numeric fluents, and the domains of such counters are encoded either via appropriate action preconditions, or via global constraints. We will study the two different versions. Even though the domain is very simple, the number of actions in the optimal

12. `http://icaps-conference.org/index.php/main/competitions`

plan grows quadratically with the number of counters, and common heuristics based on the Interval Based Relaxation (IBR) do not capture even the simplest interactions between the actions that are needed to set the counters. In the IBR, the goal is reached after one step of parallel execution. This unfortunately happens also in our subgoaling relaxation, but not in our extended formulations (i.e., using redundant constraints or conjunctive achievers).

Gardening. This domain models an agent in a 2D grid. This agent is tasked to water a number of plants scattered within the grid. The agent can carry water with a bucket; however, as this bucket has limited capacity, the agent may need to refill it multiple times to water all the plants. The goal specifies which plants need to be watered and, for each of them, the amount of water that has to be poured. In our formulation, both the position of the agent in the 2-D space and the amount of water carried are represented by numeric state variables. Optimal plans for this problem are the ones where the agent maximises the use of the bucket while she is moving, and at the same time minimises the number of moves to get to the plants and to the tap position for refilling. A greedy solution would be to consider each plant in isolation, refilling the bucket sufficiently to water a plant, and then going back to refill the bucket for the next plant. A greedy solution would however make the plans unnecessarily longer; it is in fact much better to reason over multiple plants at once, and therefore to refill the bucket with more water each time. In this domain, instances scale up to grids of size 20×20, and three plants to water.

Grouping. Finally, Grouping is the problem of pushing boxes in a grid-based domain which is entirely represented using numeric variables to denote boxes' position. The agent's goal is to group subsets of these boxes together; the goal formula makes heavy use of disjunctions to compactly representing the allowed and disallowed configurations. Instances scale on the number of boxes to be moved (from 2 to 10) and size of the grid (from 5×5 to 100×100).

13.1.2 Sailing and Farmland

These two domains were originally introduced by Scala et al. (2016a).

Sailing. This is the domain presented in Section 2.3, and depicted in Figure 1. Goals in our instances are given as a set of sites to be reached. As we have seen before, we have an action for each person to rescue, and the difficult part of this problem is to get the precondition of these actions satisfied. The satisfaction of a precondition implies that the sailing boat is close enough to rescue the castaway. Instances generated for this domain scale on the number of boats (from 1 to 4) and the number of people to rescue (from 1 to 10). Since the positions to be reached by the boats can be quite far away, plans in this problem can be quite long (we measured plans up to 1000 actions). An optimal plan corresponds to the fastest possible tour reaching all the designed locations; in this regards this problem is similar to the Travelling Salesperson Problem (Flood, 1956).

Farmland. The Farmland domain models the problem of 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. In our instances, each farm requires at least one worker. The number of workers ranges from 100 to 1000, and the number of farms from 2 to 10. In this domain plans can be quite long; each action models the single movement of each worker. The accuracy of the heuristic becomes crucial to deal with such a problem.

13.2 Satisficing Planning

We evaluate three GBFS planners, differing from each other on the particular heuristic employed[13]:

- GBFS($\hat{h}^{add}_{hbd+}$): GBFS with the optimistic variant of the inadmissible heuristic given in Eq. 11.
- GBFS($\hat{h}^{radd}_{hbd+}$): GBFS with the previous heuristic plus redundant constraints. A redundant constraint is added to approximate the conjunctive satisfaction of each pair of numeric conditions in the goal and in any action precondition, following the schema given in Section 7.
- GBFS(IBR). This has been obtained by using the best-first-search implemented in MetricFF with helpful actions disabled. This configuration yields quite a distilled version of the planner against with we can compare the heuristic proposed in this paper. For simplicity, we will use the term IBR to refer to this particular configuration of MetricFF.

Note that the two implementations with the h^1 based heuristic pay some handicap in that: i) unlike MetricFF, they do not transform the task into a LNF (the LNF transformation induces a better analysis of the state spaces by avoiding the exploration of states which are dominated by other states already seen) ii) they are implemented in ENHSP, which is in JAVA, while MetricFF is written in C.

13.2.1 Subgoaling Heuristics Analysis

Table 1 reports the data obtained using the two satisficing heuristics presented in this paper. Redundant constraints help capture action interactions (e.g, Counters) but have two main drawbacks: overestimation and computational cost. In the variants of Counters, heuristic that incorporates redundant constraints largely dominates its rival due to the high number of interacting goals: redundant constraints do capture those interactions to some extent, whereas the pure additive version does not. The result is a substantial reduction of the number of expanded nodes, which overcomes $\hat{h}^{radd}_{hbd+}$'s higher computational cost. Plan length is comparable, although it seems that adding redundant constraints tends to favour deeper solutions. This is not surprising however, as the $\hat{h}^{radd}_{hbd+}$'s estimate could be highly inadmissible, much more than $\hat{h}^{add}_{hbd+}$.

The situation in the Sailing domain is more interesting; here GBFS($\hat{h}^{radd}_{hbd+}$) solves just 29 problems, yet there are instances where it expands fewer nodes. A deeper analysis on the raw data revealed that the adoption of redundant constraint was particular beneficial on the instances involving only a single boat. For these instances, the reduction of expanded node is substantial: the configuration with redundant constraints expands on average 4225.9

13. Note here that other search engines can be used. Yet, our analysis is restricted to GBFS with the purpose of deepening our understanding on the informativeness of our relaxation and derived heuristics.

nodes, while $\hat{h}^{add}_{hbd+}$ expands 20640.8 and still obtains plans of slightly worse quality. Redundant constraints in this domain are generated starting from the convex set of constraints delimiting zones to be reached. In the instances with more than one boat, a large number of the generated constraints turned out to be dominated by the cost of the others. Summing them together results in a higher estimation that misled the planner when navigating the search space in many situations.

In the IPC domains, redundant constraints are only generated for ZenoTravel and TPP. In particular, in ZenoTravel, redundant constraints lead to the generation of non simple numeric conditions, causing a substantial drop in performance of the planner. Although this signals the potential harm of redundant constraints, it has to be noted that it can be easily avoided by not enforcing redundant constraints between constraints that are affected by assignment operations. In that case, the performance of ZenoTravel would equate again the performance obtained by $\hat{h}^{add}_{hbd+}$. In the other domains (Rovers, Satellite, Depots) there are no redundant constraint so the minor differences are only due to small perturbations in the running of the experiments.

Redundant constraints are also generated for most of the resource-flow domains, i.e., HydroPower, MarketTrader, Pathways, Sugar, Pathways. However, apart from a small advantage in node expansion in Settlers and Pathways, their contribution seems to only bring overhead. It is likely that other forms of redundant constraints should be generated in order to better capture the numeric flows arising from these domains.

	Coverage		Planning Time		Expanded Nodes		Plan Length	
Domain	$\hat{h}^{add}_{hbd+}$	$\hat{h}^{radd}_{hbd+}$	$\hat{h}^{add}_{hbd+}$	$\hat{h}^{radd}_{hbd+}$	$\hat{h}^{add}_{hbd+}$	$\hat{h}^{radd}_{hbd+}$	$\hat{h}^{add}_{hbd+}$	$\hat{h}^{radd}_{hbd+}$
Counters (11)	6	8	71.64	15.95	74653.17	2373.00	68.50	80.17
Grouping(192)	179	97	9.09	329.49	83.09	84.73	82.08	82.15
Count-Inv (11)	6	9	101.56	3.60	99094.00	339.67	110.50	120.50
Count-Rnd (33)	25	28	71.73	18.32	37156.24	1175.84	180.84	185.60
Gardening(51)	51	29	2.03	98.70	11080.52	30379.90	329.90	226.79
Sailing (40)	38	29	85.94	69.91	109441.04	115581.89	803.61	847.64
Farmland(50)	50	50	1.25	3.09	308.64	303.98	301.74	302.44
ZenoTravel(23)	18	7	1.15	116.51	14.57	14.57	11.29	11.29
Rovers(20)	10	10	25.94	25.86	14155.80	14155.80	22.60	22.60
Depots(22)	5	5	17.66	13.57	230.00	230.00	18.80	18.80
Satellite(20)	10	11	120.63	102.13	9569.40	9569.40	45.80	45.80
TPP(40)	9	7	128.97	291.56	40.57	46.57	18.71	18.86
Settlers(20)	4	2	39.24	155.91	5706.50	2973.00	40.00	41.50
Mprime(30)	26	26	66.51	99.76	214.38	214.38	7.50	7.50
HydroPower(30)	5	5	566.88	578.02	8602977.00	8602977.00	41.20	41.20
MarketTrader(20)	6	6	872.58	1224.61	457140.33	457120.33	5384.83	5384.83
Sugar(20)	5	4	32.40	131.37	19686.00	19686.67	29.33	30.00
Pathways(30)	3	3	208.41	45.11	488764.67	61423.67	32.67	36.00
Total	456	336						

Table 1: In-Depth comparison of our heuristics along 3 different dimensions: Planning Time, Plan Length and number of Expanded Nodes. The data is averaged over instances solved by all configurations.

13.2.2 Subgoaling VS Interval Based Relaxation

Table 2 compares the $\hat{h}^{add}_{hbd+}$ heuristic with the IBR obtained using MetricFF.

	Coverage		Planning Time		Expanded Nodes		Plan Length	
Domain	$\hat{h}^{add}_{hbd+}$	IBR	$\hat{h}^{add}_{hbd+}$	IBR	$\hat{h}^{add}_{hbd+}$	IBR	$\hat{h}^{add}_{hbd+}$	IBR
Counters (11)	6	11	71.64	0.11	74653.17	87.00	68.50	86.00
Grouping(192)	179	10	0.87	13.42	14.80	19.30	13.80	18.30
Count-Inv (11)	6	2	0.83	0.05	8.50	10.50	7.50	9.50
Count-Rnd (33)	25	4	0.66	0.07	3.75	3.75	2.75	2.75
Gardening(51)	51	0	NA	NA	NA	NA	NA	NA
Sailing (40)	38	0	NA	NA	NA	NA	NA	NA
Farmland(50)	50	0	NA	NA	NA	NA	NA	NA
ZenoTravel(23)	18	6	1.10	0.02	16.00	15.67	13.17	14.67
Rovers(20)	10	2	0.65	299.72	11.00	350337.00	9.00	14.00
Satellite(20)	5	1	1.48	0.03	12.00	12.00	11.00	11.00
Depots(22)	10	2	119.57	1.10	130.00	17.50	34.00	NA
TPP(40)	9	6	50.00	0.02	29.67	12.67	13.83	11.67
Settlers(20)	4	1	2.82	0.01	222.00	1.00	26.00	NA
Mprime(30)	26	11	2.82	0.11	94.09	9.09	6.64	8.09
HydroPower(30)	5	0	NA	NA	NA	NA	NA	NA
MarketTrader(20)	6	0	NA	NA	NA	NA	NA	NA
Sugar(20)	5	0	NA	NA	NA	NA	NA	NA
Pathways(30)	3	0	NA	NA	NA	NA	NA	NA
Total	456	56						

Table 2: Experimental analysis comparing $\hat{h}^{add}_{hbd+}$ with IBR. The data is averaged over instances solved by all configurations. NAs report situations where the intersection of instances solved by all configurations is empty.

Apart from the first set of instances of Counters and instances from TPP, the subgoaling heuristic developed largely outperforms the classical interval based relaxation implemented by MetricFF. There are however a number of interesting findings.

It is in fact not only the different relaxations employed by the two heuristics ($\hat{h}^{add}_{hbd+}$ vs IBR) that affect the performance, but also the way the heuristic estimate is actually calculated. On the one hand, $\hat{h}^{add}_{hbd+}$ pessimistically sums the contribution of each (sub)goal, thus neglecting any positive interactions among the actions needed to achieve them; On the other hand, IBR used in MetricFF is based on a plan-extraction mechanism that, as it happens for the propositional setting, alleviates the additive pessimistic assumption by using sometimes the same action to achieve multiple conditions. We have also observed some technical issues, with MetricFF showing unexpected behaviour over some of the numeric-flow instances. In particular for MarketTrader, MetricFF wrongly claims some of the instances to be unsolvable.

In problems like ZenoTravel, assuming complete independence between subgoals (as for $\hat{h}^{add}_{hbd+}$) pays off, and it is also beneficial in Sailing and Farmland. These two domains are particularly interesting. Indeed, only the subgoaling based heuristics are able to capture some of the internal interferences occurring among the actions. Several conditions in these two domains encompass more than one variable, a situation for which, as we have seen in our motivating example, the subgoaling relaxation is provably tighter than the interval

	Plan Length		Planning Time	
Instance	IBR	$\hat{h}^{add}_{hbd+}$	IBR	$\hat{h}^{add}_{hbd+}$
2	1	1	0.01	0.57
4	11	6	0.01	0.59
8	41	28	0.01	0.84
12	87	66	0.03	2.28
16	149	120	0.10	20.47
20	227	190	0.41	311.64
24	321	NA	1.21	TO
28	431	NA	3.03	TO
32	557	NA	6.91	TO
36	699	NA	15.04	TO
40	857	NA	25.88	TO

Table 3: IBR vs $\hat{h}^{add}_{hbd+}$ on Counters

based one, since it reasons on regressed subgoals rather than on reachable values. Therefore, $\hat{h}^{add}_{hbd+}$ will likely results more accurate than IBR. As it is possible to observe from the table, none of the instances is solved using MetricFF.

The situation in the Gardening domain is more intricate. It seems that MetricFF gets stuck quite immediately. To understand this behaviour, we also tried to run it with the standard configuration (enforced hill climbing plus helpful actions pruning), and it solved all the instances. We ascribe the reason for this behaviour to the very specific way ties are broken in MetricFF. The data for Counters, and in particular the subset of instances where all counters are initially set to zero, is also somewhat difficult to explain. MetricFF seems to dominate in terms of coverage, even if it produces longer plans on average. Table 3 shows the results for individual instances.

To investigate this behaviour more in depth we studied the effect of tie breaking in the search of our planner. More precisely, we changed the tie breaking to favour higher g-values also in the satisficing setting. In the Counters domain there are no dead-ends so being greedy cannot harm too much in terms of the ability to find *some* plan, but it could in terms of the ability to find *short* plans. Table 4 reports the data collected with this setting, and the comparison with MetricFF. This data partly justifies our intuition about the possible causes of the discrepancies between the two systems, but not fully. It seems in fact that MetricFF is not using a deterministic tie-breaking strategy, as sometimes it acts more conservatively resulting in plans slightly longer than those produced by our implementation.

Our experimental analysis for the satisficing setting concludes with another experiment where we measure the performance of our best heuristic against MetricFF with all the functionalities enabled (enforced hill climbing plus helpful actions). This experimental setting was meant to understand to what extent a tighter relaxation influences the overall system. The analysis focuses on the Sailing domain. In Sailing we have evidence that the relaxation is actually tighter. Out of 40 instances submitted, MetricFF was able to solve just 5 of them within a 1800 secs timeout. This shows the great importance of a careful exploitation of numeric structures in the heuristic.

	Plan Length		Planning Time	
# Counters	IBR	$\hat{h}^{add}_{hbd+}$ + LGTB	IBR	$\hat{h}^{add}_{hbd+}$ + LGTB
2	1	1	0.01	0.81
4	11	6	0.01	0.67
8	41	28	0.01	0.98
12	87	66	0.03	1.42
16	149	120	0.10	1.64
20	227	190	0.41	2.17
24	321	276	1.21	2.26
28	431	378	3.03	2.47
32	557	496	6.91	3.12
36	699	630	15.04	4.05
40	857	780	25.88	6.23

Table 4: IBR vs $\hat{h}^{add}_{hbd+}$ with modified tie-breaking on Counters

13.3 Optimal Planning

We evaluate 3 planners based on A^* which differ by the heuristics they are equipped with:

- A^* ($\hat{h}^{max}_{hbd+}$): A^* with the admissible heuristic described in Eq. 15
- A^* ($\hat{h}^{rmax}_{hbd+}$): A^* with the admissible heuristic plus redundant constraints, Eq. 16
- A^* (h^{gen}_{hbd}): A^* with the generalised subgoaling procedure presented in Eq. 22.

As a baseline, we run A^* with a blind goal-sensitive heuristic that gives 0 to goal states, and the value of 1 to non-goal states.

Experiments focus on a selection of domains, namely Gardening, Counters, Farmland and Sailing. For some of these, we also introduce a variant where we make use of global constraints (see next subsection). For this optimal setting, Counters instances consist of 2 up to 9 counters, for a total of 8 instances. This lets us analyse the experiments with a finer level of granularity. We refer to this new set with SmallCounters. For the other domains, we use the same set of instances as Scala et al. (2016a).

	Coverage				Planning Time				Expanded Nodes			
	$\hat{h}^{max}_{hbd+}$	$\hat{h}^{rmax}_{hbd+}$	h^{gen}_{hbd}	Blind	$\hat{h}^{max}_{hbd+}$	$\hat{h}^{rmax}_{hbd+}$	h^{gen}_{hbd}	Blind	$\hat{h}^{max}_{hbd+}$	$\hat{h}^{rmax}_{hbd+}$	h^{gen}_{hbd}	Blind
SmallCounters(8)	7	7	7	7	14.6	19.5	29.7	96.5	582670	242926	18946.7	2685633.7
Count-Inv (11)	2	2	11	2	0.7	0.5	0.5	1	415	160.5	8.5	1373.5
Count-Rnd (33)	6	7	32	6	0.5	0.4	0.5	0.8	13.3	5.5	5.5	280.3
Gardening(63)	63	63	62	63	2.6	2.7	108.8	1.9	35786.8	25959	32620.9	135477.5
Sailing (40)	6	17	7	6	5.5	0.9	152.1	88.3	92349.2	3114.8	94201	4512638
Farmland(50)	30	30	30	12	0.6	0.6	1.9	24.2	430.5	426.8	359.1	1495061.8
Total	114	126	149	96								

Table 5: Coverage for the different heuristics. In parenthesis the number of instances for each domain.

Table 5 gives an overview of the performance of each heuristic when the planner is run on the original domain definition (without global constraints). The coverage of the various heuristics differ for Sailing, Count-Inv and Count-Rnd. In Sailing the formulation using redundant constraints dominates the other heuristics quite considerably, including the sophisticated generalised subgoaling formulation h^{gen}_{hbd}. The issue with h^{gen}_{hbd} is due to an interesting phenomenon occurring in the Sailing domain, which we call the *precondition hiding phenomenon*. In this domain, each goal predicate needs a goal marking action which has a single propositional effect, but quite complex precondition. In fact, its execution is only possible when a set of numeric constraints is satisfied. The generalised form h^{gen}_{hbd} does capture the simultaneous satisfaction of the conditions, but only adds the minimum precondition cost over all the possible achievers of *every* condition present in the goals G. In Sailing, the cost of each goal $g \in G$ will always be one plus the cheapest cost of the precondition of some action that is achieving any goal g' in G, even if $g' \neq g$. Therefore, the action precondition cost that is actually needed may be actually *hidden* by the cost of some less expansive action. This phenomenon is somehow less evident in the $\hat{h}^{max}_{hbd+}$ case. Considering subgoals in isolation enables us to add the lower bound of the minimum precondition cost of the possible achievers of only a single condition (cf. Eq. 14). Looking at the raw data for Sailing, it emerges that for the problem where there is a large number of boats, h^{gen}_{hbd} performs better than $\hat{h}^{max}_{hbd+}$. However, in problems where the number of propositional goals is higher, $\hat{h}^{max}_{hbd+}$, but in particular $\hat{h}^{rmax}_{hbd+}$ estimates much better than h^{gen}_{hbd}. In SmallCounters, h^{gen}_{hbd} expands much fewer nodes than the other heuristics on average; however, this does not translate into an overall speed up. With the exception of one Count-Rnd instance, neither $\hat{h}^{max}_{hbd+}$ nor its extension with redundant constraints (i.e., $\hat{h}^{rmax}_{hbd+}$) outperform the blind heuristic in terms of coverage. Yet, they both reduce the number of expanded nodes.

The performance of h^{gen}_{hbd} on Count-Rnd and Count-Inv is magnificent. Considering the results obtained by both $\hat{h}^{max}_{hbd+}$ and $\hat{h}^{rmax}_{hbd+}$ for all the versions of Counters, it is clear that, although $\hat{h}^{rmax}_{hbd+}$ expands fewer nodes, $\hat{h}^{max}_{hbd+}$ is generally faster.

This suggests us that the redundant constraints that we generate do not bring enough information in this domain. It is possible that more powerful forms of redundant constraints could be constructed, possibly by exploiting more directly the structure of the problem. Studying how to do this systematically is an interesting research direction.

13.3.1 Experiments with Global Constraints

Global constraints have been added in the following two domains:

- Counters. Each counter has de-facto a domain of possible values; global constraints make this explicit.

- Gardening. The agent cannot cross the boundary of the grid. Moreover, the agent cannot carry a negative amount of water nor one that exceeds the actual capacity of the bucket. Both conditions are encoded using conjunctive global constraints.

We did not come up with any interesting global constraint for either the Sailing or the Farmland problem. We leave this as future work, as well as the exploitation of such global constraints automatically (which is what we would need here when such global constraints are not so easy to formulate).

Global constraints can play two roles in our planners. The first is in the actual search as they can prune away portions of the search space which are not interesting. The second is in the heuristic, whenever such a heuristic actually includes them in its reasoning. We implemented constraints that do not prune any reachable state; they just make the structure of our problem more explicit. Note that both $\hat{h}^{max}_{hbd+}$ and $\hat{h}^{rmax}_{hbd+}$ ignore global constraints, thus the only goal of this experimental evaluation is to measure the impact of h^{gen}_{hbd}.

Table 6 reports the results for Gardening (using the same problem instances as for the previous analysis) and SmallCounters. There is nothing surprising here, apart from the exceptional performance of the h^{gen}_{hbd} formulation in SmallCounters. The heuristic perfectly estimates the optimal cost for *all* the instances provided. We have also evaluated this with larger instances, and h^{gen}_{hbd} is indeed perfect with this domain. Table 7 reports the results instance by instance.

h^{gen}_{hbd} performs also quite well in Gardening. The tighter relaxation behind h^{gen}_{hbd}, and in particular the intersection with global constraints is highly beneficial to the overall time, bringing its coverage in line with that of the other heuristics. Table 9 presents raw data for Gardening. Interestingly h^{gen}_{hbd} *almost* always reduces (up to an order of magnitude) the number of expanded nodes. However, this almost never translates into an overall speed up. We only observed h^{gen}_{hbd} outperforming $\hat{h}^{rmax}_{hbd+}$ on 4 out of 63 instances (instances '9 2 3', '5 3 1', '7 1 2', and '7 3 3'). It is only when the reduction of node expansions is prominent that h^{gen}_{hbd} performs better than $\hat{h}^{rmax}_{hbd+}$ also in terms of planning time. Otherwise h^{gen}_{hbd}'s reasoning benefits are nullified by its computational cost. A careful analysis of the raw data reveals that there could even be situations where the less sophisticated $\hat{h}^{rmax}_{hbd+}$ results in more informed estimates (e.g., instance '8 1 2'). The issue here can be tracked down to the fact that h^{gen}_{hbd} can be dominated by $\hat{h}^{rmax}_{hbd+}$.

	Coverage		Planning Time		Expanded Nodes	
	$\hat{h}^{rmax}_{hbd+}$	h^{gen}_{hbd}	$\hat{h}^{rmax}_{hbd+}$	h^{gen}_{hbd}	$\hat{h}^{rmax}_{hbd+}$	h^{gen}_{hbd}
SmallCounters(8)	7	8	21.5	0.7	242926	13
Gardening(63)	63	63	4.2	32.2	34443.7	6379

Table 6: Analysis with Global Constraints

13.3.2 Numeric vs Classical Planning

Our experimental analysis concludes with an experiment where we tried to answer the following question: to what extent a sophisticated heuristic such as h^{gen}_{hbd} can make ENHSP competitive with highly optimised classical planners? To this end we encoded Counters in propositional planning. Each integer number in this propositional encoding has been modelled with an object, and arithmetic axiomatized using actions and further predicates. Table 8 reports a comparison between h^{gen}_{hbd} and Fast-Downward (Helmert, 2006) with the LM-CUT heuristic configuration[14]. Surprisingly, despite the huge number of expanded nodes, LM-CUT is competitive in the first 6 instances. However, while h^{gen}_{hbd}'s run-time

14. The code has been obtained from `http://www.fast-downward.org/IpcPlanners`

	Planning Time			Expanded Nodes		
# Counters	h^{gen}_{hbd}	$\hat{h}^{rmax}_{hbd+}$	Blind	h^{gen}_{hbd}	$\hat{h}^{rmax}_{hbd+}$	Blind
2	0.64	0.62	0.50	2	2	2
3	0.70	0.64	0.52	4	4	6
4	0.73	0.62	0.51	7	19	91
5	0.85	0.91	0.70	11	231	1332
6	0.92	1.32	2.19	16	3752	27963
7	0.97	6.69	13.07	22	71980	670859
8	1.23	141.16	TO	29	1624494	TO
9	1.33	TO	TO	37	TO	TO

Table 7: Focus on smaller instances of the Counters domain

	Planning Time		Expanded Nodes	
	h^{gen}_{hbd}	*LM-Cut*	h^{gen}_{hbd}	*LM-Cut*
2	0.39	5×10^{-5}	2	3
3	0.38	0.001	4	8
4	0.39	0.003	7	30
5	0.45	0.029	11	273
6	0.57	0.836	16	3962
7	1.42	37.494	22	71214
8	0.89	1435.98	29	1×10^{6}
9	1.08	TO	37	TO

Table 8: LM-Cut vs h^{gen}_{hbd} on a selection of instances from the Counters domain. In these instances all Counters are initially set to 0 and should be ordered such that $c_1 \leq c_2 \leq ... \leq c_n$, where n is the number of counters considered.

remain almost constant across the instances, LM-CUT run-time grows exponentially. In fact, LM-CUT only scales up to instance number 8.

14. Conclusion and Future Work

We have presented decomposition techniques based on the subgoaling principle previously developed for classical planning. These decompositions induce safe relaxations of numeric planning problems, which can be used to obtain both admissible and inadmissible yet more informed heuristics estimates. The key ideas behind the notions presented in this paper derive from a more direct exploitation of some of the numeric properties that arise in numeric planning problems, with a particular emphasis on those problems where only simple numeric conditions occur. Even though we investigate them in the context of the development of heuristics, these ideas may have a wider influence on all those reasoning tasks involving a real-world problem modelled at a greater level of detail. For instance, one can think of adopting the same relaxations to speed up SMT-based planning (Scala et al., 2016c; Bofill,

	Planning Time		Expanded Nodes	
S P V	$\hat{h}^{rmax}_{hbd+}$	h^{gen}_{hbd}	$\hat{h}^{rmax}_{hbd+}$	h^{gen}_{hbd}
4 1 1	0.59	1.41	67	40
4 1 2	0.81	1.11	477	66
4 1 3	0.62	0.99	166	30
4 2 1	0.75	0.95	267	26
4 2 2	1.58	3.70	4926	136
4 2 3	1.88	2.99	2675	230
4 3 1	2.66	7.89	9097	620
4 3 2	4.79	16.90	51531	3760
4 3 3	6.81	15.78	61176	735
5 1 1	0.54	0.83	27	30
5 1 2	0.76	1.39	529	224
5 1 3	0.83	1.48	416	199
5 2 1	1.23	8.30	1875	1110
5 2 2	2.42	9.25	8649	1974
5 2 3	1.26	3.66	2250	704
5 3 1	9.19	6.41	84418	515
5 3 2	1.15	4.91	782	730
5 3 3	4.01	7.63	18530	744
6 1 1	0.82	1.37	815	139
6 1 2	0.91	1.55	837	215
6 1 3	0.58	0.84	39	24
6 2 1	0.64	1.00	69	52
6 2 2	1.71	6.19	5055	1587
6 2 3	1.03	3.11	1103	451
6 3 1	17.82	181.01	204185	51124
6 3 2	11.59	40.63	110120	5458
6 3 3	2.42	19.58	5845	4339
7 1 1	0.49	0.82	6	22
7 1 2	1.64	1.43	705	51
7 1 3	0.79	2.01	557	471
7 2 1	1.41	2.98	3134	357
7 2 2	1.94	3.89	5498	287
7 2 3	1.08	6.59	1172	1280
7 3 1	1.86	12.15	3369	1958
7 3 2	1.29	4.79	1120	470
7 3 3	13.81	10.33	168119	1711
8 1 1	0.67	1.05	239	37
8 1 2	0.61	0.78	7	33
8 1 3	0.98	1.01	1131	31
8 2 1	1.38	3.04	1797	336
8 2 2	2.07	16.05	3883	4947
8 2 3	0.79	1.91	244	169
8 3 1	2.81	30.25	8724	3284
8 3 2	10.88	104.25	105328	22047
8 3 3	7.16	124.24	59227	26440
9 1 1	1.08	1.78	264	81
9 1 2	1.00	1.57	16	100
9 1 3	0.54	0.84	12	49
9 2 1	3.08	36.72	21782	11503
9 2 2	1.82	20.91	5540	4656
9 2 3	2.36	1.74	2871	97
9 3 1	1.41	12.29	1997	2494
9 3 2	8.86	194.34	81558	39767
9 3 3	14.67	48.87	142269	5624
10 1 1	0.86	1.01	696	29
10 1 2	0.84	1.32	755	116
10 1 3	0.71	1.28	334	147
10 2 1	2.11	3.37	6672	270
10 2 2	4.26	62.97	37106	18740
10 2 3	3.32	18.98	14474	6553
10 3 1	52.33	334.31	560494	52532
10 3 2	1.65	10.76	2088	1374
10 3 3	32.62	596.37	350839	118552

Table 9: Data on Gardening, optimal planning instances. S, P and V in the table stand for size of the grid, number of plants and specific 'variant'. For each pair S and V we generated 3 random instances where we vary the structure of the problem.

Espasa, & Villaret, 2017), much as delete-free relaxations have helped in providing shorter encodings in SAT-based planning (Rintanen, 2012) through existential semantics.

The paper discusses these novel techniques theoretically, and also practically in combination with informed search strategies such as A^* and GBFS. Results report a substantial advancement on the state of the art. Some of the notions presented here, and in particular the notion of possible achievers of numeric conditions have provided the foundations of other heuristics and relaxations recently discussed in the literature, such as hybrid landmarks (Scala, Haslum, Magazzeni, & Thiébaux, 2017) and LP- and IP-based heuristics

(Piacentini, Castro, Ciré, & Beck, 2018). These new techniques, which extend the reach of numeric planning even further, would not have been possible without the notions presented in this paper.

This work can be extended in many ways. First, the notion of possible achiever can be extended to consider more complex formalisms. That could start a completely new relaxation framework aimed at better exploiting the actual structure of certain mathematical functions natively (differently from compilation-based techniques, such as that by Li, Scala, Haslum, & Bogomolov, 2018). One of this extension could be the support for non-linear functions, such as quadratic functions. Second, the restriction on assignment operations in this paper is also very conservative and can be relaxed to account for multiple assignments to the same variable; we do not know however the implications of this relaxation. The problem of estimating numeric condition cost is already NP-complete with additive effects only, so it is likely to be at least as hard for assignments. Moreover, it is not clear whether a continuous relaxation of the underlying computational problem will be as informed as it is for additive effects. Third, the conjunctive relaxation investigated in this paper opens the way to much more powerful heuristics, however with a more unbalanced trade-off between computational cost and time. New forms of helpful actions and incrementality should be explored in order for such novel heuristics to be informed and effective at the same time.

Acknowledgments

This work was supported by ARC project DP140104219, 'Robust AI Planning for Hybrid Systems'. We also want to thank the anonymous reviewers for their useful feedback and comments.

References

Aldinger, J., Mattmüller, R., & Göbelbecker, M. (2015). Complexity of interval relaxed numeric planning. In Hölldobler, S., Krötzsch, M., Peñaloza, R., & Rudolph, S. (Eds.), *KI 2015: Advances in Artificial Intelligence - 38th Annual German Conference on AI, Dresden, Germany, September 21-25, 2015, Proceedings*, Vol. 9324 of *Lecture Notes in Computer Science*, pp. 19–31. Springer.

Bellman, R. (1958). On a routing problem. *Quarterly of applied mathematics*, *16*(1), 87–90.

Bit-Monnot, A. (2018). A constraint-based encoding for domain-independent temporal planning. In *International Conference on Principles and Practice of Constraint Programming*.

Blum, A., & Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, *90*(1-2), 281–300.

Bofill, M., Espasa, J., & Villaret, M. (2017). Relaxed exists-step plans in planning as SMT. In Sierra, C. (Ed.), *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 563–570. ijcai.org.

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, *129*, 5–33.

Bonet, B., & Helmert, M. (2010). Strengthening landmark heuristics via hitting sets. In Coelho, H., Studer, R., & Wooldridge, M. (Eds.), *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, Vol. 215 of *Frontiers in Artificial Intelligence and Applications*, pp. 329–334. IOS Press.

Borrelli, F., Bemporad, A., & Morari, M. (2017). *Predictive control for linear and hybrid systems*. Cambridge University Press.

Bradley, A. R. (2011). Sat-based model checking without unrolling. In Jhala, R., & Schmidt, D. A. (Eds.), *Verification, Model Checking, and Abstract Interpretation - 12th International Conference, VMCAI 2011, Austin, TX, USA, January 23-25, 2011. Proceedings*, Vol. 6538 of *Lecture Notes in Computer Science*, pp. 70–87. Springer.

Bryce, D., Gao, S., Musliner, D. J., & Goldman, R. P. (2015). Smt-based nonlinear PDDL+ planning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pp. 3247–3253.

Bylander, T. (1994). The computational complexity of propositional strips planning. *Artificial Intelligence*, *69*(1-2), 165–204.

Cashmore, M., Magazzeni, D., & Zehtabi, P. (2020). Planning for hybrid systems via satisfiability modulo theories. *Journal of Artificial Intelligence Research*, *67*, 235–283.

Coles, A. J., Coles, A., Fox, M., & Long, D. (2012). COLIN: planning with continuous linear numeric change. *Journal of Artificial Intelligence Research*, *44*, 1–96.

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

Davis, M., Logemann, M., & Loveland, D. (1962). A machine program for theorem-proving. *Communications of the ACM*, *5*(7), 394–397.

Domshlak, C., Hoffmann, J., & Katz, M. (2015). Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, *221*, 73–114.

Dutertre, B., & de Moura, L. (2006). A fast linear-arithmetic solver for DPLL(T). In *18th International Conference on Computer Aided Verification (CAV)*, pp. 81–94. Springer.

Edward, R. S. (1996). Invitation to dynamical systems. *Prentice Hall, Inc. A Simon & Schuster Company, Upper Saddle River, New Jersey*, *7458*, 231–316.

Eriksson, S., & Helmert, M. (2020). Certified unsolvability for SAT planning with property directed reachability. In Beck, J. C., Buffet, O., Hoffmann, J., Karpas, E., & Sohrabi, S. (Eds.), *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, pp. 90–100. AAAI Press.

Eyerich, P., Mattmüller, R., & Röger, G. (2009). Using the context-enhanced additive heuristic for temporal and numeric planning. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*.

Fikes, R., Hart, P. E., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence*, *3*(1-3), 251–288.

Flood, M. M. (1956). The traveling-salesman problem. *Operations research*, *4*(1), 61–75.

Fox, M., & Long, D. (2003). Pddl2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, *20*, 61–124.

Francès, G., & Geffner, H. (2015). Modeling and computation in planning: Better heuristics from more expressive languages. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, pp. 70–78.

Geffner, H. (2011). The model-based approach to autonomous behavior: Prospects and challenges. *Intelligenza Artificiale*, *5*(2), 163–169.

Geffner, H. (2018). Model-free, model-based, and general intelligence. In Lang, J. (Ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 10–17. ijcai.org.

Geffner, H., & Bonet, B. (2013). *A Concise Introduction to Models and Methods for Automated Planning.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Gerevini, A., Saetti, I., & 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., & 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.*, pp. 302–307.

Ghallab, M., Nau, D. S., & Traverso, P. (2004). *Automated planning - theory and practice.* Elsevier.

Gregory, P., Long, D., Fox, M., & Beck, J. C. (2012). Planning modulo theories: Extending the planning paradigm. In McCluskey, L., Williams, B. C., Silva, J. R., & Bonet, B. (Eds.), *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012.* AAAI.

Haslum, P. (2009). $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In Gerevini, A., Howe, A. E., Cesta, A., & Refanidis, I. (Eds.), *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009.* AAAI.

Haslum, P., Bonet, B., & Geffner, H. (2005). New admissible heuristics for domain-independent planning. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pp. 1163–1168.

Haslum, P., & Geffner, H. (2000). Admissible heuristics for optimal planning. In Chien, S. A., Kambhampati, S., & Knoblock, C. A. (Eds.), *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge, CO, USA, April 14-17, 2000*, pp. 140–149. AAAI.

Haslum, P., Lipovetzky, N., Magazzeni, D., & Muise, C. (2019). *An Introduction to the Planning Domain Definition Language.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Helmert, M. (2002). Decidability and undecidability results for planning with numerical state variables. In Ghallab, M., Hertzberg, J., & Traverso, P. (Eds.), *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems, April 23-27, 2002, Toulouse, France*, pp. 44–53. AAAI.

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research*, *26*, 191–246.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In Gerevini, A., Howe, A. E., Cesta, A., & Refanidis, I. (Eds.), *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009.* AAAI.

Helmert, M., & Geffner, H. (2008). Unifying the causal graph and additive heuristics. In Rintanen, J., Nebel, B., Beck, J. C., & Hansen, E. A. (Eds.), *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, pp. 140–147. AAAI.

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*, *20*, 291–341.

Hoffmann, J., Gomes, C. P., Selman, B., & Kautz, H. A. (2007). SAT encodings of state-space reachability problems in numeric domains. In *IJCAI 2007, Proceedings of the*

20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007, pp. 1918–1923.

Hooker, J. N. (2003). Planning and scheduling by logic-based benders decomposition. *Operations Research*, *55*(3), 588–602.

Illanes, L., & McIlraith, S. A. (2017). Numeric planning via abstraction and policy guided search. In Sierra, C. (Ed.), *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 4338–4345. ijcai.org.

Kautz, H. A., & Selman, B. (1999). Unifying sat-based and graph-based planning. In Dean, T. (Ed.), *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pp. 318–325. Morgan Kaufmann.

Keyder, E., & Geffner, H. (2008). Heuristics for planning with action costs revisited. In Ghallab, M., Spyropoulos, C. D., Fakotakis, N., & Avouris, N. M. (Eds.), *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings*, Vol. 178 of *Frontiers in Artificial Intelligence and Applications*, pp. 588–592. IOS Press.

Koehler, J. (1998). Planning under resource constraints. In Prade, H. (Ed.), *13th European Conference on Artificial Intelligence, Brighton, UK, August 23-28 1998, Proceedings.*, pp. 489–493. John Wiley and Sons.

Kroening, D., & Strichman, O. (2008). *Decision Procedures: An Algorithmic Point of View* (2nd edition). Springer.

Li, D., Scala, E., Haslum, P., & Bogomolov, S. (2018). Effect-abstraction based relaxation for linear numeric planning. In Lang, J. (Ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pp. 4787–4793. ijcai.org.

Liu, Y., Koenig, S., & Furcy, D. (2002). Speeding up the calculation of heuristics for heuristic search-based planning. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada.*, pp. 484–491.

Martello, S., & Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Inc., New York, NY, USA.

McDermott, D. V. (1996). A heuristic estimator for means-ends analysis in planning. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems, Edinburgh, Scotland, May 29-31, 1996*, pp. 142–149.

McDermott, D. V. (2003). Reasoning about autonomous processes in an estimated-regression planner. In Giunchiglia, E., Muscettola, N., & Nau, D. S. (Eds.), *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003), June 9-13, 2003, Trento, Italy*, pp. 143–152. AAAI.

Moore, R. E., Kearfott, R. B., & Cloud, M. J. (2009). *Introduction to Interval Analysis.* SIAM.

Nieuwenhuis, R., Oliveras, A., & Tinelli, C. (2006). Solving SAT and SAT modulo theories: From an abstract davis-putnam-logemann-loveland procedure to dpll(t). *Journal of the ACM*, *53*(6), 937–977.

Pearl, J. (1984). *Heuristics - intelligent search strategies for computer problem solving.* Addison-Wesley series in artificial intelligence. Addison-Wesley.

Pednault, E. P. (1987). Formulating multiagent, dynamic-world problems in the classical planning framework. In *Reasoning about actions & plans*, pp. 47–82. Elsevier.

Piacentini, C., Castro, M. P., Ciré, A. A., & Beck, J. C. (2018). Linear and integer programming-based heuristics for cost-optimal numeric planning. In McIlraith, S. A., & Weinberger, K. Q. (Eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 6254–6261. AAAI Press.

Pommerening, F., Röger, G., Helmert, M., & Bonet, B. (2014). Lp-based heuristics for cost-optimal planning. In Chien, S. A., Do, M. B., Fern, A., & Ruml, W. (Eds.), *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI.

Pommerening, F., Röger, G., Helmert, M., & Bonet, B. (2015). Heuristics for cost-optimal classical planning based on linear programming. In Yang, Q., & Wooldridge, M. (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 4303–4309. AAAI Press.

Rintanen, J. (2006). Unified definition of heuristics for classical planning. In Brewka, G., Coradeschi, S., Perini, A., & Traverso, P. (Eds.), *ECAI 2006, 17th European Conference on Artificial Intelligence, August 29 - September 1, 2006, Riva del Garda, Italy, Including Prestigious Applications of Intelligent Systems (PAIS 2006), Proceedings*, Vol. 141 of *Frontiers in Artificial Intelligence and Applications*, p. 600. IOS Press.

Rintanen, J. (2008). Regression for classical and nondeterministic planning. In *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings*, pp. 568–572.

Rintanen, J. (2012). Planning as satisfiability: Heuristics. *Artificial Intelligence*, *193*, 45–86.

Rintanen, J. (2017). Temporal planning with clock-based SMT encodings. In Sierra, C. (Ed.), *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 743–749. ijcai.org.

Scala, E. (2013). Numeric kernel for reasoning about plans involving numeric fluents. In Baldoni, M., Baroglio, C., Boella, G., & Micalizio, R. (Eds.), *AI*IA 2013: Advances in Artificial Intelligence - XIIIth International Conference of the Italian Association for Artificial Intelligence, Turin, Italy, December 4-6, 2013. Proceedings*, Vol. 8249 of *Lecture Notes in Computer Science*, pp. 263–275. Springer.

Scala, E. (2014). Plan repair for resource constrained tasks via numeric macro actions. In Chien, S. A., Do, M. B., Fern, A., & Ruml, W. (Eds.), *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI.

Scala, E., Haslum, P., Magazzeni, D., & Thiébaux, S. (2017). Landmarks for numeric planning problems. In Sierra, C. (Ed.), *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 4384–4390. ijcai.org.

Scala, E., Haslum, P., & Thiébaux, S. (2016a). 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*, pp. 3228–3234.

Scala, E., Haslum, P., Thiébaux, S., & Ramírez, M. (2016b). Interval-based relaxation for general numeric planning. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, pp. 655–663.

Scala, E., Ramírez, M., Haslum, P., & Thiébaux, S. (2016c). Numeric planning with disjunctive global constraints via SMT. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016, London, UK, June 12-17, 2016.*, pp. 276–284.

Shin, J., & Davis, E. (2005). Processes and continuous change in a sat-based planner. *Artificial Intelligence*, *166*(1-2), 194–253.

Suda, M. (2014). Property directed reachability for automated planning. *Journal of Artificial Intelligence Research*, *50*, 265–319.

Vallati, M., Chrpa, L., Grzes, M., McCluskey, T. L., Roberts, M., & Sanner, S. (2015). The 2014 international planning competition: Progress and trends. *AI Magazine*, *36*(3), 90–98.

van den Briel, M., Benton, J., Kambhampati, S., & Vossen, T. (2007). An LP-based heuristic for optimal planning. In *Proc. 13th Int. Conf. on Principles and Practice of Constraint Programming (CP)*, pp. 651–665.

Vidal, V., & Geffner, H. (2006). Branching and pruning: An optimal temporal POCL planner based on constraint programming. *Artif. Intell.*, *170*(3), 298–335.

Wehrle, M., & Rintanen, J. (2007). Planning as satisfiability with relaxed $-step plans. In *AI 2007: Advances in Artificial Intelligence, 20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia, December 2-6, 2007, Proceedings*, pp. 244–253.

Wolfman, S. A., & Weld, D. S. (2001). Combining linear programming and satisfiability solving for resource planning. *The Knowledge Engineering Review*, *16*(1), 85–99.

Zhang, L., Madigan, C. F., Moskewicz, M. H., & Malik, S. (2001). Efficient conflict driven learning in a boolean satisfiability solver. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 279–285. IEEE.