Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

Search-Guidance Mechanisms for Numeric Planning Through Subgoaling Relaxation

Enrico Scala,[*] Alessandro Saetti, Ivan Serina, Alfonso E. Gerevini
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia, Italy

Abstract

Recently, a new decomposition based relaxation has been proposed for numeric planning problems. Roughly, this relaxation is grounded on the identification of regression-based necessary conditions for the satisfaction of sets of numeric subgoals. So far, it has been used to define novel heuristics that are able to provide great guidance in problems exhibiting a pronounced numeric structure. This paper investigates how to further exploit this relaxation; it does so by introducing the notion of the multi-repetition relaxed plan. The multi-repetition plan annotates actions with the number of times such actions need to be executed. We use this structure for different purposes: extraction of a concrete relaxed plan based heuristic, definition of subgoaling based helpful actions, and definition of what we call up-to-jumping actions. Up-to-jumping actions allow us to deeply leverage from the metric structure of the problem and devise an informed search strategy that can collapse several decision steps. We experimentally analyze a forward state space planner equipped with these novel mechanisms across several planning benchmarks, showing the benefit of the ideas presented in the paper.

Introduction

Automated planning studies how an agent can organize its actions to achieve some desired objective. This task can be described under different environment assumptions, and different languages can be employed to represent it in a more or less concise manner (Geffner and Bonet 2013). One of such languages is numeric planning. Numeric planning is the extension of classical planning where a state also contains numeric information. In numeric planning, actions can change both propositional and numeric variables. Both pieces of information can then participate in the definition of the conditions that determine when actions are executable in a given state, or when the goal is reached. Numeric planning has been formalized in PDDL2.1 (Fox and Long 2003) through the notion of numeric fluents. The result is a language that can be used to model the peculiarity of a domain of interest in a hybrid way, using both purely propositional and numeric structures. Planning with numeric state variables is convenient when the domain features metric information that is difficult to compile away using propositional logic. For instance, consider an underwater or aerial vehicle. The position of the vehicle can be modelled through propositions by discretising the position into classes and encoding each class with a different proposition. However, when the discretisation has to be fine-grained and the area inside which the vehicle moves is large, the number of propositions is huge and this compromises the efficacy and the efficiency of planners especially for those grounding action parameters before search. Although numeric planning is in general undecidable (Helmert 2002), it is still important to investigate how planning systems supporting numeric variables can practically scale up over instances where solution plans are not limited to just a few actions, especially when these planners are used in robotic contexts (Löhr et al. 2012).

Numeric planning has seen an increasing attention in the literature over the last years (e.g., (Gerevini, Saetti, and Serina 2012; Aldinger, Mattmüller, and Göbelbecker 2015; Scala et al. 2016; Piacentini et al. 2018; Bofill, Espasa, and Villaret 2017)), and novel relaxation based heuristics targeting a fragment of it have been defined (Scala, Haslum, and Thiébaux 2016; Piacentini et al. 2018). Following this line of research, and with the long-term goal of closing the performance gap between purely classical/propositional and numeric planners, we focus on the subgoaling-based relaxation and explore it for the design of novel search guidance mechanisms. We start by showing that the subgoaling relaxation and relative heuristics can be used for the extraction of a richer relaxed plan representation that we call *multi-repetition relaxed plan*. We construct such a representation by reworking Keyder and Geffner (2007)'s results through the lens of numeric planning. Our procedure intertwines numeric and propositional justifications for repeated action executions by exploiting the necessary conditions underlying the subgoaling based decomposition and the notion of possible achievers. Then, we show how to use the constructed multi-repetition relaxed plan to devise i) a novel cost-sensitive relaxed plan based heuristic ii) a novel mechanism for helpful actions selection, and iii) the notion of *up-to-jumping actions*, which is yet another way to focus the search towards what the relaxation has provided. It does so by dynamically creating shortcuts in the search space.

After a detailed presentation of the proposed techniques, we empirically evaluate our proposal over a large set of

[*]Corresponding author. Email: enrico.scala@unibs.it

benchmarks. Our analysis reveals that, on a state-space search planner, in particular the combination of subgoaling helpful actions and up-to-jumping actions provides great search-guidance across the majority of the problem instances under evaluation.

Background

We assume a ground planning formalism equivalent to PDDL 2.1 level 2 of which we only summarize the relevant aspects here. For a more thorough description of the syntax and semantics of the language, have a look at the seminal paper by Fox and Long (2003). A planning problem $\Pi \doteq \langle F, X, A, s_0, G\rangle$ consists of propositional and numeric state variables (F and X respectively), actions (A), an initial state (s_0) and a goal (G). A state assigns a truth value to each proposition in F and a rational number to each state variable in X. A numeric condition takes the form $\xi \trianglerighteq k$, where ξ is an arithmetic expression over X, $\trianglerighteq \in \{\leq, <, =, >, \geq\}$ and $k \in \mathbb{Q}$ is a constant. Action preconditions and the goal G are sets (conjunctions) of numeric conditions and propositions. Action effects assign boolean (propositional) variables and/or increase/decrease/assign the value of numeric variables by an arithmetic expression; $\mathrm{pre}(a)$ and $\mathrm{eff}(a)$ indicate preconditions and effects of action a, respectively.

An action a is applicable in a state s iff $\mathrm{pre}(a)$ is satisfied in s and both numeric and propositional effects do not induce conflicting assignments. The application of an action a in a state s yields a state s' where the values of the propositional and numeric variables are changed according to $\mathrm{eff}(a)$. All state variables not touched by the action remain unchanged (frame axiom). A sequence of actions $\langle a_0, ..., a_{n-1}\rangle$ is said to be a *valid plan* if, when iteratively applied in s_0, it is such that each action is applicable in the state resulting from the prefix before it ($s_i \models \mathrm{pre}(a_i)$), and the last state s_n satisfies the goal, $s_n \models G$. Each action a_i has a cost $\lambda(a_i) \in \mathbb{Q}^{\geq 0}$ and the cost of a plan $\pi = \langle a_0, ..., a_{n-1}\rangle$ is $\sum_{0 \leq i < |\pi|} \lambda(a_i)$; a plan is said to be optimal if its cost is minimal over all valid plans. Finally, with $\mathcal{S}(\Pi)$ we indicate the state space induced by planning problem Π, and with $Sol(\Pi)$ the set of all valid plans for Π. When clear from the context, we will omit the variable sets in the problem tuple Π (e.g, $\Pi \doteq \langle s_0, A, G\rangle$).

In this paper we are interested in the fragment of numeric planning where all preconditions and goals are either propositional or simple numeric conditions, that is:

Definition 1 (Simple Numeric Planning (Scala, Haslum, and Thiébaux 2016))**.** *Let Π be a numeric planning problem. A numeric condition $\xi \trianglerighteq k$ is* simple *iff ξ is a linear arithmetic expression and all variables in ξ are affected only by action effects that increase/decrease them by constant values. Problem Π is simple iff all numeric preconditions of its actions and goals are simple.*

The Subgoaling-Based Relaxation

The subgoaling-based relaxation targets the structure of simple numeric problem by reasoning over actions that contribute positively to the achievement of linear numeric conditions using the notion of m-times-regressor (Scala, Haslum, and Thiébaux 2016):

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

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

where $m \in \mathbb{N}$, and $k_{x,a}$ is the constant additive effect of a on x (i.e., $\langle x, +=, k_{x,a}\rangle \in \mathrm{eff}(a)$).

Definition 3 (Possible Achiever)**.** *We say that action a is a* possible achiever *of c in a state s if there exists an $m \in \mathbb{N}$ such that $s \models c^{r(a,m)}$.*

In the case of a simple numeric condition, all elements of Eq. 1 except m are constant values extracted from the action description. This results in the regression being a linear function of the action repetitions, thus having a constant derivative. An important aspect is that, thanks to this property, the actions that possibly achieve a given condition can be detected statically (independently from any state), and this is a necessary condition for satisfying the linear numeric condition. Moreover, even though the number m of repetitions needed to actually achieve the condition is a state-dependent computation, this can be performed by solving a simple mathematical expression. In particular, we denote by $m(s, a, c)$ the number of times an action needs to be executed to make condition c satisfied from s through a.

The numeric subgoaling relaxation (Scala, Haslum, and Thiébaux 2016) uses the notion of possible achiever to over-approximate reachable numeric conditions recursively. This principle can be used to derive admissible or inadmissible heuristic estimates. In this paper we make mainly use of the inadmissible formulation:

$$\hat{h}^{add}_{hbd}(s,c) \doteq \begin{cases} 0 & \text{if } s \models c \\ \min\limits_{a\in ach(c)} \left(\lambda(a) + \hat{h}^{add}_{hbd}(s, \mathrm{pre}(a))\right) & \text{if } c \text{ is PC} \\ \min\limits_{\substack{a\in A\ ,\hat{m}\in\mathbb{Q}^{\geq 0} \\ s\models c^{r(a,\hat{m})}}} \left(\hat{m}\lambda(a) + \hat{h}^{add}_{hbd}(s, \mathrm{pre}(a))\right) & \text{if } c \text{ is SC} \\ \sum\limits_{c'\subset c: |c'|=1} \hat{h}^{add}_{hbd}(s, c') & \text{if } |c| > 1 \end{cases} \qquad (2)$$

where, when c is a propositional condition, $ach(c)$ denotes the set of all actions that achieve the condition. PC denotes when a condition is propositional, and SC when it is a simple numeric condition. Let us recall that subscript hbd stands for $HyBriD$, and the little hat on the h stands for the integrality constraint relaxation (i.e., $m \in \mathbb{N}^{\geq 0} \rightsquigarrow \hat{m} \in \mathbb{Q}^{\geq 0}$). With a slight abuse of notation, we use $\hat{h}^{add}_{hbd}$ to denote both the estimate and the underlying relaxation.

Example 1. *Consider the instance of the* SAILING *domain (Scala, Haslum, and Thiébaux 2016) shown in Figure 1.* SAILING *models the problem of controlling a sailing boat in*

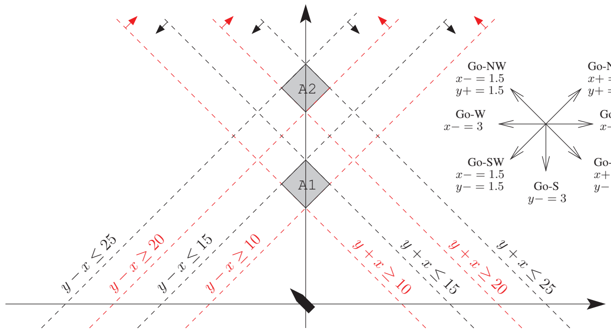


Figure 1: An instance of the SAILING domain.

an unbounded area. We assume the wind is fixed and blows from the north. Propositional variables model the presence of the boat in one of the two grayed areas; numerical variables model the Cartesian coordinates of the boat. Initially the boat is at the origin and is tasked to reach the two grayed areas; the numeric effects of the actions are sketched in the figure (there is no precondition for these actions); actions `a1` *and* `a2`*, both with $\lambda($a1$) = \lambda($a2$) = 1$, mark whether the boat has achieved the two grayed areas; the inequalities denote the numeric conditions of* `a1` *and* `a2` *– in red those not satisfied in the initial state.*

Under $\hat{h}^{add}_{hbd}$ relaxation, the possible achievers of conditions $y - x \geq 10$ and $y - x \geq 20$ are Go-W and Go-NW; these conditions require $3.\bar{3}$ and $6.\bar{6}$ fractional executions of either one of these actions. Similarly, $3.\bar{3}$ and $6.\bar{6}$ reps of either Go-E or Go-NE achieve conditions $y + x \geq 10$ and $y + x \geq 20$. Thus, for $\hat{h}^{add}_{hbd}$, the cost of reaching the goals from the initial state is the sum of the costs of the following actions: the 2 actions `a1` *and* `a2`*; the $3.\bar{3}$ reps of an action achieving $y - x \geq 10$; the $3.\bar{3}$ reps of an action achieving $y + x \geq 10$; the $6.\bar{6}$ reps of an action achieving $y - x \geq 20$; and the $6.\bar{6}$ reps of an action achieving $y + x \geq 20$. So, $\hat{h}^{add}_{hbd}$ estimates 22. In contrast, the optimal (integer) cost is only 16. The estimate of $\hat{h}^{add}_{hbd}$ is greatly pessimistic essentially because it ignores positive interactions among actions altogether, i.e., it ignores that moving the sailing boat to the nearest grayed area also contributes to get close to the farthest grayed one.*

Multi-Repetition Relaxed Plan

This section describes how to extract what we call a *multi-repetition relaxed plan* (MRP hereinafter) from the subgoaling relaxation. Then we see how the MRP is used for heuristic estimation, and for eliciting further problem structure, such as helpful actions and the new *up-to-jumping actions*. This method is based on the observation that, despite the intrinsic complexity of managing numeric state variables, the formulation presented by Keyder and Geffner (2007) can be extended to support propositional and numeric reasoning at the same time. We do so by studying the problem from the subgoaling relaxation standpoint. Intuitively, subgoals lead to reasoning on estimating the cost of achieving formulae rather than variable values, therefore by-passing the problem of dealing with an infinite state space. The following recursive formulation synthesizes the MRP extraction for numeric planning problems under the subgoaling relaxation $\hat{h}^{add}_{hbd}$:

$$\pi(s,c) \doteq \begin{cases} \{\} & \text{if } s \models c \\ \{(a, m(s,a,c))\} \cup \pi(s, \text{pre}(a)) & \text{if } |c| = 1 \\ \text{ with } a = best(s,c) & \\ \bigcup_{c' \in c} \pi(s,c') & \text{if } |c| > 1 \end{cases} \tag{3}$$

$$best(s,c) \doteq \underset{\substack{a \in A, \hat{m} \in \mathbb{Q}^{\geq 0} \\ s \models c^{r(a,\hat{m})}}}{\arg\min} (\hat{m}\lambda(a) + \hat{h}^{add}_{hbd}(s, \text{pre}(a)))$$

The recursive definition of π in Eq. 3 incrementally constructs a set of pairs, each involving an action instance and a natural number. The construction exploits the commitment established in the $\hat{h}^{add}_{hbd}(s,c)$ computation. This is performed using function *best* as specified in Eq. 3. Once the action is established through such a function, the regression conjoins the MRP for its action preconditions with a pair $\langle \cdot, \cdot \rangle \in A \times \mathbb{N}$; the pair links the established achiever with the ceiling of the action counter $\lceil \hat{m} \rceil$ that is necessary to enable the numeric subgoal through that action. As explained by Scala, Haslum, and Thiébaux (2016), in simple numeric planning this operation amounts at looking at the rate of influence of the action on the targeted condition. Here, since we want to construct an explicit relaxed plan, we round up the fractional repetitions; this gets the action effects to satisfy the constraint.

Note that, the above best function uses the inadmissible estimate $\hat{h}^{add}_{hbd}$, in contrast to the original classical planning formulation (Keyder and Geffner 2007), that instead uses h^{max}. However, the numeric analogous of h^{max} as defined by Scala, Haslum, and Thiébaux (2016), $\hat{h}^{max}_{hbd}$, introduces a further decomposition in the establishment of the achiever for a single numeric condition. This has the effect of decoupling the established achiever from its precondition. That is why we started from $\hat{h}^{add}_{hbd}$.

To obtain a formulation that is more aligned with the pure classical version of the heuristic, we use a slightly different variant of $\hat{h}^{add}_{hbd}$ where we treat the conjunctive case the same way this would be dealt with in the classical case, i.e., maximising instead of summing, and leave everything else unchanged. More precisely, we substitute in Eq. 2 the case where $|c| > 1$ with the following:

$$\max_{c' \subset c : |c'| = 1} \hat{h}^{add}_{hbd}(s,c')$$

Note that, since the operator maximizes among a heterogeneous set made of numeric and propositional conditions, when there are only propositional variables, the formulation nicely mimics its purely propositional version. We also tested experimentally the effect of the former version, but did not observe any significant difference to justify a different treatment for the numeric case.

Since there is an infinite set of MRPs, an obvious question is whether the addition of numeric variables in the plan extraction mechanism makes its construction possible and effective in general. It is easy to see that, not only this is possible, but it also is polynomial in the size of the problem.

Proposition 1. *If the function $\hat{h}^{add}_{hbd}(s,G) \neq \infty$, the computation of the MRP as defined in Eq. 3 terminates in a finite number of steps, and can be accomplished in polynomial time in the size of the input problem.*

Proof Sketch. The proof relies on the following facts: i) the recursion in Eq. 3 is only formulated on the subgoals of the problem, not on the numeric values of the variables; ii) the computation of the best achiever can be computed beforehand for all relevant subgoals using a label-correcting algorithm (Scala, Haslum, and Thiébaux 2016); iii) the recursion reaches always the base case where the precondition of the selected action is satisfied. Once the best achievers are calculated (which takes polynomial time), the backward procedure is a greedy process that only appends the already established achiever to the working relaxed plan. □

From a high level perspective, the main difference with the popular relaxed-plan extraction used by METRIC-FF (Hoffmann 2003) is in the fact that the MRP keeps intact and more explicit the numeric relationships among actions and subgoals. Indeed, METRIC-FF relaxed-plan extraction procedure backchains from the goals, too. However, each numeric condition in METRIC-FF extraction is *deconstructed* in a number of simpler conditions, each of which consists of one numeric variable and a constant. These simpler conditions identify a (further) numeric state variable based decomposition that is then used by METRIC-FF to find justifications for the addition of new actions in the relaxed plan. Also, in simple numeric planning, METRIC-FF's relaxed-plan extraction for the achievement of a deconstructed condition c at a given level l is simplified by adding *all* the actions at any level $l' \leq l$ that contribute to the satisfaction of c. Differently, each element in the MRP approximates the action choice(s) using a subgoal oriented justification, and is guided by the minimisation of $\hat{h}^{add}_{hbd}$; therefore, it does not lose the relationship among numeric variables belonging to the same condition, and provides a way to restrict the action selection. As we will see, this is not only more precise, but also provides an easy access to heuristic computation as well as to the formulation of novel helpful actions and of what we call *up-to-jumping actions*.

A MRP based heuristic

A straightforward way to use the MRP to devise a heuristic function is by summing the costs of each action that occurs in all pairs in the MRP, multiplied by the number of repetitions associated to that action in the pair. However, this may easily result in an inaccurate estimate. In fact, as it is possible to observe, the MRP can contain pairs sharing the same action but having different counters; this arises in problems where the same action can achieve different numeric subgoals; e.g., in our SAILING problem, multiple executions of Go-E or Go-NE can achieve both conditions $y + x \geq 10$ and $y + x \geq 20$. As we have seen before, $\hat{h}^{add}_{hbd}$ ignores this problem and yields a highly pessimistic estimate.

To overcome this issue we devise an MRP based heuristic that merges pairs containing the same action, and maximizes the relative counters; more formally:

$$h^{MRP}_{max}(s) \doteq \sum_{a \in \{a^\star | (a^\star, m^\star) \in \pi\}} c(a) \cdot \max_{(a,m') \in \pi(s)} \{m'\} \qquad (4)$$

An obvious alternative formulation is keeping the minimum counter instead of the maximum one. Notice, however, that such a formulation would lead to severe under-approximation of the actual cost. A smarter formulation would be combining the sum and the max of the repetitions; this could be beneficial in some situations where there is a pronounced presence of negative interactions between the actions. It is not clear how to capture this negative implications in a principled manner, and therefore we leave a more profound investigation of this aspect as a future work. Let us keep in mind though that an optimal selection is NP-hard, since computing an optimal solution of the delete-free relaxation of a classical planning problem is NP-hard (Bylander 1994) and this is obviously a subcase of our task.

Example 2 (Continuing on SAILING)**.** *Let us contrast the relaxed plan computed by* METRIC-FF *heuristic and the estimate h^{MRP}_{max} from the initial state.* METRIC-FF *constructs a relaxed planning graph encompassing 4 (action) levels, numbered from 0 to 3. Condition $y - x \geq 20$ at level 3 is deconstructed into $y \geq 9$ and $(-x) \geq 18$, i.e., the maximum value of y and $(-x)$ reachable after three time steps is required to be achieved by relaxed plan π. The same is applied to $y + x \geq 20$ at level 3, and to $y - x \geq 10$ and $x + y \geq 10$ at level 2. The plan extraction will include action* `a2` *at level 3, and action* `a1` *at level 2; then, in the first three levels we find* all *the actions supporting the deconstructed conditions: Go-NW and Go-NE increasing y, Go-SW and Go-W increasing $(-x)$, and Go-SE and Go-E increasing x. Under unitary action costs,* METRIC-FF*'s estimate is $1 + 1 + 3 \cdot (2 + 2 + 2) = 20$.*

Under the assumption that our best function over $\hat{h}^{add}_{hbd}$ always breaks ties the same way, a concrete MRP that can be generated is the following: pairs $(\texttt{a1}, 1)$, $(\texttt{a2}, 1)$, (Go-NW, 4), (Go-NE, 4), (Go-NW, 7), (Go-NE, 7). According to Eq. 4, the heuristic cost derived from h^{MRP}_{max} is 16, which is also the cost of the optimal plan solving the actual problem.

Subgoaling Helpful Actions

One of the powerful aspects of having a concrete plan representation solving a problem relaxation is the fact that it makes more explicit which set of subgoals is useful to achieve in order to have a solution that is (at least) "relaxed applicable"; these subgoals are extracted by looking at all actions' preconditions selected to form a relaxed plan to achieve a goal, and the goal itself.

An idea that exploits such subgoals to tighten the relation between the actual search and the relaxation is that of using helpful actions (Hoffmann 2003), also referred to as preferred operators (Richter and Helmert 2009). An action is said to be helpful if it achieves at least one precondition of the relaxed plan. The idea is to use helpful actions as a means to eagerly guide the search towards those successor states with shorter relaxed plan length. The use of helpful actions in search has hugely extended the reach of classical and numeric planners in the past (Hoffmann and Nebel 2001; Hoffmann 2003), and is also exploited to speed up satisficing planner in many ways, for instance in limiting the search of enforced-hill-climbing, or in systems using multiple priority queues, one of which is only dedicated to the search space entailed by the use of preferred operators (Richter and Westphal 2010; Richter and Helmert 2009).

Motivated by the success of helpful actions, this section shows that there is quite a simple mechanism for getting helpful actions that are based on the subgoaling relaxation, too. The idea is about exploiting the decomposition-based principle of the subgoaling relaxation for the case of simple numeric planning by reasoning about the *multi-repetition relaxed plan*, and the notion of possible achiever.

Let $\pi(s)$ be the MRP computed for a problem Π on a state s following Eq. 3, the set of subgoaling helpful actions for s are those that are applicable in s and are possible achievers (in the classical or numeric sense) of some other action precondition or goals in π. More formally:

Definition 4 (Subgoaling Helpful Actions)**.** *An action a is said to be helpful in a state s w.r.t. relaxed plan π iff: i) a is applicable in s and ii) there exists a $g \in \bigcup_{(a',m') \in \pi} \mathrm{pre}(a') \cup G$ where $s \not\models g$, and $g \in \mathrm{eff}(a)^+$ or $\exists \hat{m} \in \mathbb{Q}^{\geq 0} \cdot s \models g^{r(a,\hat{m})}$ (acc. Def. 2). We denote with $H(s)$ the set of all helpful actions in s*

Example 3 (Continuing on SAILING)**.** *The subgoaling helpful actions are those achieving the numeric conditions $y - x \geq 10$, $x + y \geq 10$, $y - x \geq 20$, $x + y \geq 20$ of actions a1 and a2 in the MRP, i.e., Go-NW, Go-W, Go-NE, Go-E. In contrast, the classical helpful actions defined by* METRIC-FF *also include Go-SW and Go-SE that increase variables $(-x)$ and x, respectively, but that do not contribute at all to reach the grayed areas. Substantially, the deconstruction of numeric subgoals used by* METRIC-FF *during the relaxed plan extraction results in a poor selection of helpful actions.*

Observing the example one may be inclined to conjecture that, in general, tighter relaxations (as it is the case when comparing the subgoaling relaxation with the interval based relaxation (Li et al. 2018)) lead to less helpful actions because the conditions under which actions are considered helpful are harder to be satisfied. However, note that, tighter relaxations may lead to more subgoals to be achieved, too, and therefore potentially more actions to be considered as helpful. Thereby, the use of helpful actions from different relaxations can lead to quite different explorations of the state space, and it is difficult to provide some general argument about which state space is better to be explored from a theoretical standpoint. In our experiments we will use helpful actions as a pruning technique in a forward state space planner, i.e., we will generate only successors reachable by helpful actions. Note that, as for classical planning (Hoffmann and Nebel 2001), this makes the search incomplete; our experiments evaluates whether this is useful anyway.

Up-To-Jumping Actions

The *multi-repetition relaxed plan* is not only useful because it gives a way to devise subgoaling based helpful actions, but also because it makes explicit when, at least from a relaxation point of view, the same action needs to be repeated several times to make a condition satisfied. Consider the SAILING example. It is easy to see that the sailing boat can achieve a position where $y - x \geq 10$ by executing at least 4 times either *Go-NW* or *Go-W*; every other sequencing of actions will only make the plan unnecessarily longer.

Motivated by situations as in this example, we introduce the notion of *up-to-jumping action*.

Definition 5 (Up-to-jumping Action, Syntax and Semantics)**.** *An up-to-jumping action is a pair (a, m) where $a \in A$ and $m \in \mathbb{N}$. An up-to-jumping action (a, m) is executable in a state s if the precondition of a are satisfied in s (as a regular action); the state resulting form executing (a, m) in s is obtained by repeating the execution of a until either it is executed m times or a state where the preconditions of a are not satisfied is generated by $m' < m$ executions of a.*

We extract up-to-jumping actions from the MRP, considering only those actions that are required to be repeated more than once. Moreover, among those pairs in MRP corresponding to the same action we only keep the pair with the minimum number of repetitions. More formally, let π be an MRP, the set $J(s)$ of *up-to-jumping actions* is defined as:

$$J(s) = \{(a, m) \mid (a, m) \in \pi, m > 1, \forall (a, m') \in \pi \; m \leq m'\}$$

Example 4 (Continuing on SAILING)**.** *According to the set of actions in the MRP mentioned in Example 2, our set J of up-to-jumping actions consists of pairs (Go-NW, 4) and (Go-NE, 4). Note that the prefix of any optimal plan is either (Go-NW, 4) or (Go-NE, 4), and such a prefix is exactly one of the up-to-jumping actions in J. Note that, if the definition of J kept the pairs which maximise the number of repetitions for a given action, instead of those that minimise such a number, the (first) computed plan would firstly reach the farthest grayed area and only subsequently the nearest one, i.e., its quality would be much worse.*

Proposition 2 (Soundness and Incompleteness with up–to-jumping actions)**.** *Let s be a state and S be the set of states obtained using applicable actions in the problem actions set A; let S' be the successor states obtained by applying actions from the heterogeneous set $A' \doteq \{a \mid a \in A \cdot \nexists (a, m) \in J(s)\} \cup J(s)$. It follows that:*

- *Soundness:*
$$\bigcup_{s' \in S'} Sol(\langle s', A', G \rangle) \neq \emptyset \Rightarrow \bigcup_{s'' \in S} Sol(\langle s'', A, G \rangle) \neq \emptyset$$
- *Incompleteness:*
$$\bigcup_{s' \in S'} Sol(\langle s', A', G \rangle) = \emptyset \not\Rightarrow \bigcup_{s'' \in S} Sol(\langle s'', A, G \rangle) = \emptyset$$

Proof Sketch. (Soundness) Let s be a state; every state s' obtained by using an action from $J(s)$ can be reached with regular actions only, too.
(Incompleteness) Intuitively, an up-to-jumping action can hide a state that needs to be traversed to reach the goal.

Consider a planning domain with propositional variables $F \doteq \{p, q, r\}$, numeric variables $X \doteq \{x\}$, and actions $A \doteq \{a, b, c\}$. Consider an initial state s_0 assigning true to p, false to both q and r, and 0 to x. Take as a goal G a formula that requires p true and $x \geq 2$. Let $\text{pre}(a) \doteq \{r, x \leq 1\}$, $\text{eff}(a) \doteq \{q = \top\}$, $\text{pre}(b) \doteq \emptyset$ and $\text{eff}(b) \doteq \{r = \top, p = \bot, x += 1\}$, $\text{pre}(c) \doteq \{q\}, \text{eff}(c) \doteq \{p = \top\}$. The MRP constructed starting from s_0 contains pair $(b, 2)$, which is also the only up-to-jumping action at hand. The state s' obtained from s_0 by applying b twice makes variable x equal to 2, and hence from s' action a becomes not applicable anymore. Since there is no way of recovering from this situation, s' becomes a dead-end state. Indeed, a valid plan for the problem is $\langle b, a, b, c\rangle$. The prefix $\langle b, a, b\rangle$ cannot be explored if we do not split the up-to-jumping action $(b, 2)$. $\square$

In order to deal with this incompleteness, in our experiments, we use up-to-jumping actions only *in addition to* the regular actions, which, in turn, may be affected by helpful action pruning or not; although this increases the branching factor, the depth of the search tree can be reduced, for some situations. On the other hand, it also introduces a degree of sub-optimality if used in a greedy setting. There are so two trade-offs: one is trading branching-factor for depth of the search tree; the other one applies to greedy search and concerns sacrificing quality of the solution for runtime. Both will be evaluated experimentally.

Related Work

Our work builds on relaxed plan heuristics and helpful actions developed for classical propositional planning. In particular, Keyder and Geffner (2007) observed that the adoption of a relaxed planning graph structure is not necessary for the construction of a relaxed plan; this can indeed be obtained directly using the recursive formulation of h^{max}, yielding a crisper definition of the procedure, and, more importantly, making it possible to use the relaxed plan in a cost-sensitive manner. We show that a similar mechanism can be devised in numeric planning under the condition of using a subgoaling based relaxation over a simple numeric planning problem. Besides the obvious extension to support numeric reasoning, another difference w.r.t. the work of Keyder and Geffner (2007) is that we obtain best achievers using a version of the subgoaling heuristic that provides a compromise between $\hat{h}^{add}_{hbd}$ and $\hat{h}^{max}_{hbd}$; this is motivated by the fact that the admissible version of the subgoaling relaxation heuristic $\hat{h}^{max}_{hbd}$ adds a relaxation estimating single conditions, too.

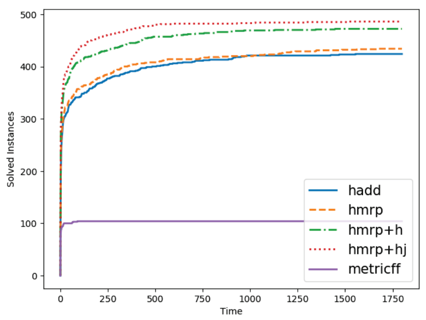
Relaxed plan heuristics and helpful actions in numeric planning have already been studied, but only using the interval-based relaxation (Hoffmann 2003; Gerevini, Saetti, and Serina 2004; Aldinger, Mattmüller, and Göbelbecker 2015). In particular, METRIC-FF's heuristic and helpful actions have thoroughly influenced the literature on numeric planning, e.g., (Gerevini, Saetti, and Serina 2008; Coles et al. 2010; 2014). Our techniques differ from those grounded on interval-based relaxation: we reason about subgoals and actions rather than about variables and actions. We expect that the mechanisms developed from this interpretation are more powerful, e.g., *up-to-jumping actions*; as we will see, our experiments confirm this expectation.

Other numeric planners (i.e., (Aldinger and Nebel 2017), (Scala et al. 2016)) exploiting the interval-based relaxation or other abstractions (Illanes and McIlraith 2017) have recently appeared in the literature. However, in contrast to our work, they do not consider the simple numeric planning case, but focus on more general, non-linear numeric planning. As a consequence, when the problem does not involve complex, non-linear constructs, they perform even worse than METRIC-FF. It is important to note that our focus on simple numeric planning is not limiting, but complementary; similarly to the extension of $\hat{h}^{add}_{hbd}$, $\hat{h}^{add}_{hbd+}$ (Scala, Haslum, and Thiébaux 2016), our techniques can still be adopted in more expressive settings using a decomposition/recursive schema. For instance, one can extract a plan ignoring the complex (non-simple) subgoals, and then add the cost given by another heuristic (e.g., based on interval relaxation) for all those subgoals that are required by the relaxed-plan, but are not achieved yet. Alternatively, one can apply our mechanisms to the simple numeric planning effect-abstraction of linear problems presented by Li et al. (2018). Studying the effect of these choices is future work.

Related is the field that studies macro-actions (e.g., (Chrpa, Vallati, and McCluskey 2014; Botea et al. 2005; Korf 1985; Scala 2014; Scala and Torasso 2015)), and all techniques speeding up search by unrolling relaxed plans in state-space planners (e.g., (Vidal 2004)). An up-to-jumping action can be seen as a standard macro action with the difference being that it is computed deductively (instead of inductively), and is dynamically extracted from a relaxation. MARVIN (Coles and Smith 2007) extracts macros during the search, too, but it uses them to escape from previously encountered plateau. The usage of up-to-jumping actions in a forward state-space search poses issues similar to those raised by the usage of macro-actions, and techniques similar to those used by MARVIN for detecting the most promising situations could in principle be used in our context, too.

Implementation and Experiments

Our experimental analysis evaluates the usefulness of the techniques introduced in this paper. In particular, we intended to measure the effectiveness of the *multi-repetition relaxed plan* structure when this is exploited for the extraction of (i) the estimate h^{MRP}_{max}, (ii) the subgoaling helpful actions, and (iii) the up-to-jumping actions. For this reason, we consider four different configurations in a forward state space planner guided by a greedy best-first search ($f(n) = h(n)$). With $\hat{h}^{add}_{hbd}$ we denote a configuration where the planner is guided by the additive heuristic presented in (Scala, Haslum, and Thiébaux 2016); this is the baseline in our experiments. For the new contributions we use h^{MRP}_{max} to indicate the configuration where the same planner is guided by h^{MRP}_{max}; with $h^{MRP}_{max} + H$ and $h^{MRP}_{max} + HJ$ we denote two variants of h^{MRP}_{max} where we branch over only helpful actions ($h^{MRP}_{max} + H$) and helpful actions plus up-to-jumping actions ($h^{MRP}_{max} + HJ$) extracted from the MRP, respectively. Note that up-to-jumping actions are not affected



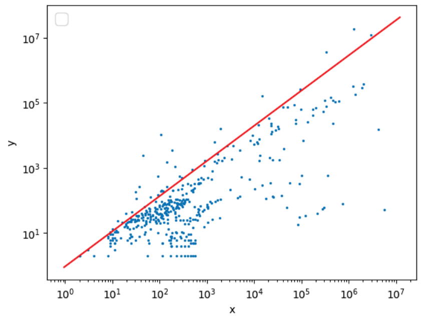


Figure 2: Left: Coverage vs Timeout analysis. x-axis displays number of solved instances against allotted time (y-axis). Right: Scatter plotting node expansions of $h^{MRP}_{max} + HJ$ (x-axis) vs $\hat{h}^{add}_{hbd}$ (y-axis). Unsolved instances have been removed.

Domain	$\hat{h}^{add}_{hbd}$	h^{MRP}_{max}	$h^{MRP}_{max} + H$	$h^{MRP}_{max} + HJ$	$h^{\text{METRIC-FF}}$
Heavily-Numeric					
COUNTERS (11)	6	6	9	**11**	1
COUNTERS-INV (11)	6	6	8	**10**	1
COUNTERS-RND (33)	26	26	**33**	32	8
GARDENING (51)	51	51	51	51	0
FARMLAND (50)	50	50	50	50	0
GROUPING (192)	159	156	168	**178**	17
SAILING (40)	37	**40**	**40**	**40**	3
From IPCs					
DEPOTS (23)	10	16	18	**19**	3
ROVER (20)	11	11	**12**	**12**	10
SATELLITE (20)	7	10	**14**	**14**	7
SETTLERS (20)	4	4	6	6	1
ZENOTRAVEL (23)	22	**23**	**23**	**23**	22
DEPOTS (C)(23)	15	18	**19**	**19**	2
SATELLITE (C)(22)	4	4	4	4	7
ZENOTRAVEL (C)(23)	16	13	**17**	**17**	**22**
Total	424	434	472	**486**	104

Table 1: Coverage of systems across all domains. In parenthesis, the number of instances for a given domain.

by the helpful actions pruning in that they are themselves constructed from helpful actions. We also add to our analysis the interval-based relaxation with the corresponding helpful actions; we call this configuration $h^{\text{METRIC-FF}}$. All subgoaling based mechanisms are built on top of ENHSP (https://sites.google.com/view/enhsp/). For simulating configuration $h^{\text{METRIC-FF}}$, we use METRIC-FF by disabling enforced hill climbing and enabling (interval-based) helpful actions in a greedy best-first search. Every tested planner addressed each problem instance by running up to 1800 seconds with a memory cut of 16GB. Tests are performed on a single core of a 4 Intel(R) Xeon(R) (2.30 GHz).

Benchmarks. We use domains from recent work in numeric planning (Scala, Haslum, and Thiébaux 2016), Functional STRIPS (Francès and Geffner 2016), and previous International Planning Competitions. Our analysis puts particular emphasis on domains presenting a pronounced numeric structure (hereinafter referred as *Heavily-Numeric*): COUNTERS, SAILING, GARDENING, FARMLAND, GROUPING. These problems involve mostly numeric effects and numeric conditions. Their description is omitted here for lack of space; for details have a look at (Scala, Haslum, and Thiébaux 2016) and (Francès and Geffner 2016) papers. The descriptions of the IPCs domains can be found in the paper by Long and Fox (2003). For some IPC domains, we also consider the action costs version. This is indicated with a 'C' appended next to the domain name.

Results. The parameter under evaluation that we consider most important is the coverage, i.e., the number of instances solved by a system over all instances of our benchmark suite. Table 1 and Figure 2 detail such an evaluation. The other parameters we evaluated are run time, number of expanded search nodes, and plan cost. These pieces of information are given in Table 2. $\hat{h}^{add}_{hbd}$ and h^{MRP}_{max} solve roughly the same number of instances, but present strengths and weaknesses in different domains. In all the instances of the three considered variants of the COUNTERS domain, both heuristics lead to the same amount of node expansions. In this domain, conjunctive goals are relaxed in the same manner by $\hat{h}^{add}_{hbd}$ and h^{MRP}_{max}. Moreover, as each of them is dealt with a different achiever, there is no reason why h^{MRP}_{max} should behave differently; there are indeed no positive interactions that can be captured by looking at the MRP. Instead, in GROUPING, heuristic $\hat{h}^{add}_{hbd}$ is much more accurate leading to less search; it seems that h^{MRP}_{max} tends to be too optimistic. As expected, h^{MRP}_{max} is very effective in SAILING. Indeed, we observe not only a significant reduction of node expansions, but also significant improvements on the quality of the obtained plans. This is due to the capability of h^{MRP}_{max} to capture the positive interactions among the sailing actions (see Figure 3 left).

The improvement is global with subgoaling helpful action pruning activated, and there is no domain where our baseline solves more problems than $h^{MRP}_{max} + H$. Also the speed-up is substantial. As it is indicated by Table 2, the reduction of node expansions happens pretty much in every domain, and we get savings of at least one order of magnitude. This situation confirms that using helpful actions is very beneficial in numeric domains as it is in classical planning; this somewhat contrasts to some earlier observations done in the seminal work of METRIC-FF (Hoffmann 2003) about the effectiveness of numeric helpful actions in preserving completeness. Out of all our tested instances, only for one instance of DEPOTS the use of subgoaling helpful action pruning wrongly deemed it unsolvable. We think that this robustness is explained by the use of a tighter relaxation that seems beneficial in strongly pronounced numeric problems, and is less

Domain	# I	Run Time				Expansions				Plan Cost			
		$\hat{h}^{add}_{hbd}$	h^{MRP}_{max}	$h^{MRP}_{max}+H$	$h^{MRP}_{max}+HJ$	$\hat{h}^{add}_{hbd}$	h^{MRP}_{max}	$h^{MRP}_{max}+H$	$h^{MRP}_{max}+HJ$	$\hat{h}^{add}_{hbd}$	h^{MRP}_{max}	$h^{MRP}_{max}+H$	$h^{MRP}_{max}+HJ$
Heavily-Numeric													
COUNTERS	6	30.55	35.05	**1.05**	**0.59**	50780.67	50780.67	3887.67	**80.83**	77.0	77.0	81.5	**69.5**
COUNTERS-INV	6	50.6	62.66	**1.95**	**1.53**	83827.5	83827.5	8668.0	**4610.0**	**119.5**	**119.5**	134.5	136.67
COUNTERS-RND	26	70.06	80.21	**1.08**	50.69	42921.31	42921.31	**2248.58**	140053.23	**197.85**	**197.85**	202.12	247.27
GARDENING	51	21.37	48.45	**9.01**	**9.18**	212281.22	352309.63	**46713.14**	**52180.06**	1181.35	**1151.71**	1418.67	1498.8
FARMLAND	50	0.88	0.88	0.78	0.81	308.96	330.52	304.26	**9.4**	301.72	301.88	301.7	301.76
GROUPING	146	111.62	152.29	96.01	**37.77**	179.75	203.39	199.7	**61.04**	**178.69**	196.64	196.86	233.68
SAILING	37	65.74	**1.13**	**0.98**	53.67	309365.32	**483.16**	**500.19**	859102.08	754.78	**452.51**	**452.73**	560.49
From IPCs													
DEPOTS	10	158.21	38.6	**10.81**	**8.44**	12177.2	11103.5	**4111.0**	**4111.0**	**53.7**	71.4	67.2	67.2
ROVER	9	2.66	1.94	1.7	**1.62**	9421.22	**473.78**	**25.0**	**25.0**	22.67	24.22	**20.0**	**20.0**
SATELLITE	7	61.86	221.09	**1.53**	**1.46**	613349.86	1166805.29	**2210.57**	**2210.57**	23.0	27.86	**22.14**	**22.14**
SETTLERS	4	198.6	202.55	**2.98**	37.22	39151.5	47393.5	**2089.0**	66561.75	**70.5**	75.25	100.25	119.5
ZENOTRAVEL	22	123.96	32.04	**24.3**	**25.07**	171.09	**70.09**	94.05	94.05	46.55	**43.32**	44.73	44.73
DEPOTS (C)	14	112.03	**10.88**	**9.26**	**8.31**	8211.64	**851.64**	1932.71	1932.71	**113.57**	149.57	128.36	128.36
SATELLITE (C)	2	1.06	1.31	0.85	0.93	**263.0**	1976.0	587.5	587.5	**112.34**	**112.34**	150.6	150.6
ZENOTRAVEL (C)	13	8.29	6.14	**1.03**	**1.02**	**174.0**	1174.38	299.62	299.62	37431.54	42418.92	**29215.85**	**29215.85**

Table 2: Average run-time, node expansions and plan cost for all configurations. The evaluation considers instances those solved by all systems (# I is the number of such instances). Bold indicates best values within 5%.

evident in problems from the IPC. In fact, in the IPC domains, also METRIC-FF achieves good performances.

h^{MRP}_{max} with helpful and up-to-jumping actions obtains the highest coverage. This is more pronounced with limited computational time (Figure 2 left). This configuration solves all instances of COUNTERS. Note that there is a big jump of difficulty among small and large instances in COUNTERS. The largest instance (40 counters) has an optimal plan of 780 actions, while the optimal plan of the immediately smaller instance (36 counters) involves 630 actions. Another domain benefiting from this configuration is GROUPING. Here the planner generates quite an interesting sequences of moving actions. Same thing happens in SAILING. The situation for this latter is however more complicated. Figure 3 left shows that, for the majority of the instances, the configuration using up-to-jumping actions allowed the planner to quickly converge towards a solution. However, in a few cases, the up-to-jumping actions make the search more difficult. The best performance of the up-to-jumping actions setting is in FARMLAND (Figure 3 right). Here, we observe a reduction of node expansions going up to one order of magnitude when compared to h^{MRP}_{max} and helpful actions. Figure 2 also shows an overall substantial improvement in terms of number of expanded nodes w.r.t. the baseline $\hat{h}^{add}_{hbd}$.

Regarding plan quality, the use of up-to-jumping actions make the search greedier, pretty much across all the tested domains; e.g., in the GARDENING domain, we observed an average increase of almost fifty percent. Interestingly, there is no clear winner across all domains; configurations seem quite complementary. Remarkable are the results of the MRP based mechanisms in SAILING. In particular h^{MRP}_{max} almost halves (on the average) the size of the plan; as we have seen in our example throughout the paper, this is explained by a more principled handling of the positive interactions between actions.

We also tested h^{MRP}_{max} with up-to-jumping actions only, but found this configuration rather unsatisfactory. By limiting the branching factor, the helpful actions pruning compensates at best the overhead of adding up-to-jumping actions into the search. This proves to be extremely prolific. Yet, there are situations needing further investigations; as observable from Table 1 and Figure 3 left, the full configuration loses an instance of COUNTERS-RND and took long time for a couple of instances in SAILING. It is likely that the up-to-jumping actions greedy nature hided some action in the search tree that could have made the search shorter; a possible explanation for this can be found in the integral approximation that is performed whilst building the MRP.

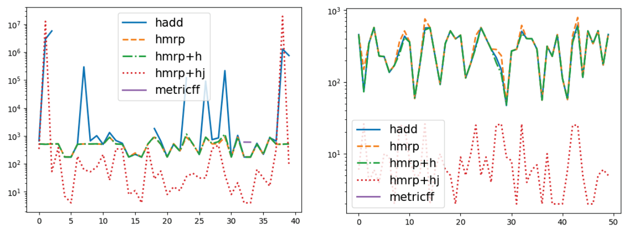


Figure 3: SAILING (left) and FARMLAND (right) number of expanded nodes across all instances.

The addition of helpful actions takes the subgoaling relaxation to be competitive with $h^{\text{METRIC-FF}}$ over IPC domains, too. These domains all present a heavy classical structure and $h^{\text{METRIC-FF}}$ greatly handles them by leveraging from the efficacy of FF's propositional reasoning. The subgoaling helpful actions seems to provide an effective means to put classical and numeric reasoning in synergy.

Conclusion

We have studied how to exploit structural information from the subgoaling relaxation. We did so by exploiting a novel characterization of the relaxed plan, called multi-repetition relaxed plan (MRP), that is aimed at keeping more intact the numeric structure of the problem and the relation between actions and subgoals. We have proposed techniques to extract such an MRP from subgoaling relaxation and, starting from the MRP, novel ways to enhance the search: the former is an extension of well known helpful actions; the latter is a novel mechanism allowing to collapse many decision steps.

An experimental analysis in the paper demonstrates the benefit of these new notions and methods in the context of greedy-best-first search, but it can be the case that they can be powerful tools also in other search schemas (e.g., enforced-hill-climbing, or search schema based on the use of multi priority-queues (Richter and Helmert 2009; Richter and Westphal 2010)). Another important aspect that is worth to explore is finding the right way of combining different heuristics together, an approach that, if done properly, can be quite fruitfully (Röger and Helmert 2010). We look forward to investigate these aspects in our immediate future work.

Acknowledgments. We thank the anonymous reviewers for their useful comments. This work was supported by MIUR "Fondo Dipartimenti di Eccellenza 2018-2022" of the DII Department at the University of Brescia.

References

Aldinger, J., and Nebel, B. 2017. Interval based relaxation heuristics for numeric planning with action costs. In *KI*, volume 10505 of *Lecture Notes in Computer Science*, 15–28. Springer.

Aldinger, J.; Mattmüller, R.; and Göbelbecker, M. 2015. Complexity of interval relaxed numeric planning. In *KI*, volume 9324 of *Lecture Notes in Computer Science*, 19–31. Springer.

Bofill, M.; Espasa, J.; and Villaret, M. 2017. Relaxed exists-step plans in planning as SMT. In *IJCAI*, 563–570. ijcai.org.

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-ff: Improving AI planning with automatically learned macro-operators. *J. Artif. Intell. Res.* 24:581–621.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artif. Intell.* 69(1-2):165–204.

Chrpa, L.; Vallati, M.; and McCluskey, T. L. 2014. MUM: A technique for maximising the utility of macro-operators by constrained generation and use. In *ICAPS*. AAAI.

Coles, A., and Smith, A. 2007. Marvin: A heuristic search planner with online macro-action learning. *J. Artif. Intell. Res.* 28:119–156.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *ICAPS*, 42–49. AAAI.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2014. COLIN: planning with continuous linear numeric change. *CoRR* abs/1401.5857.

Fox, M., and Long, D. 2003. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.* 20:61–124.

Francès, G., and Geffner, H. 2016. Effective planning with more expressive languages. In *IJCAI*, 4155–4159. IJCAI/AAAI Press.

Geffner, H., and 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, A.; and Serina, I. 2004. Planning with numerical expressions in LPG. In *ECAI*, 667–671. IOS Press.

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

Gerevini, A.; Saetti, A.; and Serina, I. 2012. Case-based planning for problems with real-valued fluents: Kernel functions for effective plan retrieval. In *ECAI*, 348–353. IOS Press.

Helmert, M. 2002. Decidability and undecidability results for planning with numerical state variables. In *AIPS*, 44–53. AAAI.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.* 14:253–302.

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

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

Keyder, E., and Geffner, H. 2007. Heuristics for planning with action costs. In *CAEPIA*, volume 4788 of *Lecture Notes in Computer Science*, 140–149. Springer.

Korf, R. E. 1985. Macro-operators: A weak method for learning. *Artif. Intell.* 26(1):35–77.

Li, D.; Scala, E.; Haslum, P.; and Bogomolov, S. 2018. Effect-abstraction based relaxation for linear numeric planning. In *IJCAI*, 4787–4793. ijcai.org.

Löhr, J.; Eyerich, P.; Keller, T.; and Nebel, B. 2012. A planning based framework for controlling hybrid systems. In *ICAPS*. AAAI.

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *J. Artif. Intell. Res.* 20:1–59.

Piacentini, C.; Castro, M. P.; Ciré, A. A.; and Beck, J. C. 2018. Linear and integer programming-based heuristics for cost-optimal numeric planning. In *AAAI*, 6254–6261. AAAI Press.

Richter, S., and Helmert, M. 2009. Preferred operators and deferred evaluation in satisficing planning. In *ICAPS*. AAAI.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res.* 39:127–177.

Röger, G., and Helmert, M. 2010. The more, the merrier: Combining heuristic estimators for satisficing planning. In *ICAPS*, 246–249. AAAI.

Scala, E., and Torasso, P. 2015. Deordering and numeric macro actions for plan repair. In *IJCAI*, 1673–1681. AAAI Press.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2016. Interval-based relaxation for general numeric planning. In *ECAI*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 655–663. IOS Press.

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

Scala, E. 2014. Plan repair for resource constrained tasks via numeric macro actions. In *ICAPS*. AAAI.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In *ICAPS*, 150–160. AAAI.