Subgoaling Relaxation-based Heuristics for Numeric Planning with Infinite Actions

Ángel Aso-Mollar[1], Diego Aineto[1], Enrico Scala[2], Eva Onaindia[1]

[1]Valencian Research Institute for Artificial Intelligence (VRAIN), Universitat Politècnica de València
[2]Università degli Studi di Brescia
{aaso,dieaigar,onaindia}@vrain.upv.es, {enrico.scala}@unibs.it

Abstract

Numeric planning with control parameters extends the standard numeric planning model by introducing action parameters as free numeric variables that must be instantiated during planning. This results in a potentially infinite number of applicable actions in a state. In this setting, off-the-shelf numeric heuristics that leverage the action structure are not feasible. In this paper, we identify a tractable subset of these problems—namely, controllable simple numeric problems—and propose an optimistic compilation approach that transforms them into simple numeric tasks. To do so, we abstract control-dependent expressions into bounded constant effects and relaxed preconditions. The proposed compilation makes it possible to effectively use subgoaling heuristics to estimate goal distance in numeric planning problems involving control parameters. Our results demonstrate that this approach is an effective and computationally feasible way of applying traditional numeric heuristics to settings with an infinite number of possible actions, pushing the boundaries of the current state of the art.

Code — github.com/hstairs/jpddlplus/tree/enhsp25-dpex

Introduction

In numeric planning, states include numeric variables, which are updated by actions through arithmetic effects (Fox and Long 2003). This extension of classical planning enables quantitative information to directly influence the evolution of the system, representing phenomena such as resource consumption, cumulative change, or behavior governed by numeric computations.

Support for numeric state variables is insufficient for many problems in which the control is subject to numerically parameterized actions. Consider, for example, a `turn-right` action with a certain degree of rotation or a `pour-water` action involving an arbitrary number of liters. Research conducted in this line focuses on extending numeric planning to include *control parameters*; i.e., action parameters that extend over infinite numeric domains. Different approaches have explored this idea from complementary perspectives. The TM-LPSAT planner (Shin and Davis 2005) integrates the control parameters into a hybrid SAT and linear programming framework. POPCORN (Savaş et al. 2016) and NextFLAP (Sapena, Onaindia, and Marzal 2024), in contrast, embed control parameters within a forward partial-order planning search. More recently, the approach of (Heesch et al. 2024) delegates the selection of control parameter values to a neural model.

The aforementioned works treat control parameters as constraints that narrow the search space, ruling out the modeling of these parameters as decision variables, as this would lead to an infinite action space. For example, a robot may be allowed to turn right through an angle between 20° and 45°, yet the consequences of each specific value can differ significantly. The S-BFS approach (Aso-Mollar et al. 2025b), however, introduces sampling into a forward state-space search algorithm to explicitly handle numeric decisions during planning.

S-BFS studies the problem of numeric planning with *control variables*, a reformulation of actions with infinite domain numeric parameters. It is a principled framework for reasoning in such settings under full state information, which allows for the use of heuristic functions as estimators. S-BFS provides a systematic way of searching when there are infinitely many action instantiations, using a sampling function to iteratively generate subsets of successors. Although it shows to be competitive with respect to SOTA methods, even when simple heuristics are used (Aso-Mollar et al. 2025a), the core difficulty with this approach lies in that standard, informative numeric heuristics (Scala et al. 2016, 2020a) cannot be used in infinite action spaces.

In this paper, we identify a tractable fragment of numeric planning with control variables and introduce an optimistic compilation that transforms such problems into simple numeric tasks, a common approach in the literature of numeric planning (Li et al. 2018) or HTN planning (Höller et al. 2019) for defining novel estimators. To do so, we abstract control-dependent expressions and convert them into bounded constant effects and relaxed preconditions. When combined with a relaxation of numeric planning tasks that decomposes the problem into numeric subgoals, hereafter referred to as the subgoaling relaxation (Scala et al. 2020a), the resulting compiled problems preserve the relaxation structure required to provide meaningful estimations. This enables standard numeric subgoaling heuristics to be applied directly to estimate the distance to the goal in the

original infinite tasks. Our results show that this compilation provides an effective and computationally feasible mechanism for extending traditional numeric heuristics based on subgoaling to problems with infinite actions.

Background

In this section, we summarize the control variables formalism of S-BFS, its search scheme and the subgoaling relaxation for numeric planning, on which the heuristic estimators defined by our approach are based.

Numeric Planning with Control Variables

We adopt the numeric planning with control variables formalization by Aso-Mollar et al. (2025a) and adapt it to our needs. In this setting, F is a set of propositional state variables, X, a set of numeric state variables, and U a set of numeric control variables. We assume for every variable $v \in X \cup U$ that a valuation function maps v to its domain $Dom(v)$, and that a numeric domain is a subset of the rational numbers; that is, $Dom(v) \subseteq \mathbb{Q}$ for every $v \in X \cup U$. We denote the set of arithmetic expressions over X and U as $\text{Expr}(X \cup U)$. First, we define numeric conditions that involve control variables.

Definition 1 (Numeric condition with control variables)**.** *A **numeric condition with control variables** is an inequality $(\xi \bowtie 0)$, where $\xi \in \text{Expr}(X \cup U)$ and $\bowtie \in \{<, \leq, \doteq, \geq, >\}$, combined with logical operators. We denote the set of all such conditions over X and U as $\text{Constr}_{\mathbb{Q}}(X \cup U)$.*

Controllable numeric assignments are defined as numeric assignments that involve control variables in the right part of the assignment.

Definition 2 (Numeric assignments with control variables)**.** *A **numeric assignment with control variables** is an atomic update of the form $(x := \xi)$, where $x \in X$ and $\xi \in \text{Expr}(X \cup U)$. We denote assignments of the form $(x := x + \xi)$ as $(x \mathrel{+}= \xi)$. We also denote the set of such assignments from $X \cup U$ to X as $\text{Assign}_{\mathbb{Q}}(X, U)$. A set of assignments $\mathcal{A} \subseteq \text{Assign}_{\mathbb{Q}}(X, U)$ is **consistent** if, for every $x \in X$, there is at most one assignment $(x := \xi)$ in $\mathcal{A}$, as defined for propositional assignments.*

Additionally, we denote as $\text{Constr}_{\mathbb{B}}(F)$ and $\text{Assign}_{\mathbb{B}}(F)$ the sets of propositional conditions and assignments, respectively.

Definition 3 (Numeric planning problem with control variables)**.** *A **numeric planning problem with control variables** is a tuple $\mathcal{P} = (F, X, U, A, s_0, G)$, where:*

- *A is a finite set of actions $a = (\text{Pre}(a), \text{Eff}(a))$, where $\text{Pre}(a) = \langle \text{Pre}_{\mathbb{B}}(a), \text{Pre}_{\mathbb{Q}}(a) \rangle$ and $\text{Eff}(a) = \langle \text{Eff}_{\mathbb{B}}(a), \text{Eff}_{\mathbb{Q}}(a) \rangle$, such that*
  1. *$\text{Pre}_{\mathbb{B}}(a) \in \text{Constr}_{\mathbb{B}}(F)$ and $\text{Pre}_{\mathbb{Q}}(a) \in \text{Constr}_{\mathbb{Q}}(X \cup U)$ are sets of conditions;*
  2. *$\text{Eff}_{\mathbb{B}}(a) \subseteq \text{Assign}_{\mathbb{B}}(F)$ and $\text{Eff}_{\mathbb{Q}}(a) \subseteq \text{Assign}_{\mathbb{Q}}(X, U)$ are consistent sets of assignments;*
- *s_0 is the initial state with valuations over F and X;*
- *$G = \langle G_{\mathbb{B}}, G_{\mathbb{Q}} \rangle$, where $G_{\mathbb{B}} \in \text{Constr}_{\mathbb{B}}(F)$ and $G_{\mathbb{Q}} \in \text{Constr}_{\mathbb{Q}}(X)$, are the goal conditions.*

Algorithm 1: DPEX_{ϕ, r_h}

```
1: Input: Sampling function φ, rectification function r_h,
   num. samples K, initial state s_0, goal conditions G
2: Output: true iff a goal state is reached
3: Open ← {(f(s_0), s_0)}
4: while Open ≠ ∅ do
5:     Extract node s with lowest f-value from Open
6:     if s ⊨ G then
7:         return true
8:     else
9:         for i = 1 to K do
10:            Sample a successor s' using φ
11:            Insert (f(s'), s') into Open
12:        if s is not fully expanded then
13:            Rectify f(s) using r_h
14:            Insert (f(s), s) into Open
15: return false
```

A **plan** in this setting, $\pi = (\langle a_i, \mu_i \rangle)_{i=1}^{k}$, is a sequence of pairs consisting of an action and a control valuation. A plan is **valid** if there exists a sequence of states $(s_i)_{i=1}^{k}$ such that $s_i \models subs_{\mu_i}(\text{Pre}(a_i))$, where $subs_{\mu_i}$ denotes the substitution of control variables in numeric expressions according to valuation μ_i. A plan is a **solution** if it leads to a goal state, i.e., if $s_k \models G$.

Searching with Delayed Partial EXpansions

Hereinafter, we approach the problem of searching for a valid plan using the search scheme of S-BFS, which is based on Delayed Partial Expansions (DPEX) of search nodes. To cope with the infinite branching factor, DPEX follows a BFS-like scheme that relies on a sampling function ϕ to iteratively generate finite subsets of successors, referred to as partial expansions, and on a rectification function r_h to update the parent's f-value after each partial expansion. The rectification function r_h can be seen as an abstraction of a heuristic function h that adjusts the evaluation of h accordingly to the number of partial expansions performed in a given state. Algorithm 1 describes how DPEX works.

The algorithm maintains a frontier of states ordered by an evaluation function f (line 3). At each iteration, the state with lowest f is selected (line 5) and K successors are sampled using a sampling function ϕ (lines 9–10) and inserted in the frontier (line 11). If the selected node is not fully expanded, its f-value is updated via a rectification function r_h (lines 12–13) and reinserted (line 14). Through this interplay of sampling and rectification, DPEX can handle infinite branching factors efficiently while remaining probabilistically complete under mild assumptions on ϕ and r_h. It becomes critical to have a heuristic function h that accurately estimates the cost to the goal. Hereinafter, we will attempt to address this issue by leveraging the subgoaling relaxation.

Subgoaling Relaxation for Numeric Planning

In numeric planning **without** control variables, that is, problems with $U = \emptyset$, the subgoaling relaxation is a method that

estimates the distance to the goal by decomposing complex numeric conditions into simpler subgoals that can be analyzed independently (Scala et al. 2020a). Rather than reasoning about the simultaneous satisfaction of all goal constraints, the relaxation evaluates whether each atomic condition can be achieved individually.

The subgoaling relaxation extends its classical counterpart (Bonet and Geffner 2001) by exploiting the numeric structure of the problem. In the propositional case, identifying potential achievers of a condition is straightforward. When computing the relaxed heuristic h for a propositional condition ψ, it suffices to consider all actions that achieve it, i.e., actions whose effects make ψ true. This set, denoted by $ach(\psi)$, can be determined syntactically by inspecting the action effects, independently of the current state. The heuristic value is then computed as

$$h(\psi) = \min_{a \in ach(\psi)} (\lambda(a) + h(s, \mathrm{Pre}_{\mathbb{B}}(a)))$$

where $\lambda(a)$ denotes the cost of action a (which is assumed to be unitary in this work).

For numeric conditions, however, this syntactic test is not sufficient. Whether an action can contribute to achieving a numeric condition depends on the current values of the involved variables in the state s. Consequently, the identification of possible achievers becomes state dependent. To address this issue, the subgoaling relaxation relies on a **regressor operator** (Scala et al. 2020a), which rewrites goal conditions backward through action effects. The regressed condition characterizes the states from which repeated applications of the action can make progress towards achieving the original condition.

Given a state s, an action is considered a **possible achiever** of a numeric condition if s satisfies the m-times regressed condition (m-regressor) for some $m \in \mathbb{N}$. Let ψ be a numeric condition and $\psi^{r(a,m)}$ the m-regressor of ψ through a; heuristic calculations of ψ are based on the following formula:

$$h(s, \psi) = \min_{\substack{a \in A, \\ s \models \psi^{r(a,m)}, \\ \forall m \in \mathbb{N}}} (m\,\lambda(a) + h(s, \mathrm{Pre}_{\mathbb{Q}}(a)))$$

For example, suppose $x = 8$ in a state s, an action a with effect $x+ = 1$ and a numeric condition $\psi \equiv x > 10$. Then, s satisfies the 3-regressor of ψ through a.

The subgoaling relaxation is particularly effective for **simple numeric planning problems**, that is, problems in which all conditions are linear and all assignments follow the form $x \mathrel{+}= k$ for some constant k. In these cases, the m-regressor can be computed in closed form. This relaxation allows the derivation of both admissible and inadmissible heuristic estimates, following the principles of the propositional h^{max} and h^{add} heuristics (Bonet and Geffner 2001), respectively. Such heuristics can be computed by considering only possible achievers.

Ultimately, the subgoaling relaxation can be seen as a compilation that transforms a numeric planning problem $\mathcal{P}$ into another problem $\mathcal{P}^1$ that reasons over independent subgoal reachability through possible achievers. The resulting problem $\mathcal{P}^1$ *over-approximates* the reachability of the original problem: any state or goal achievable in $\mathcal{P}$ is also achievable in $\mathcal{P}^1$, although the converse does not necessarily hold due to the decomposition into independent subgoals. Any problem transformation with this property is known as a **relaxation**. As a consequence, states that are detected as unsolvable in the relaxation are also unsolvable in the original problem and can therefore be safely pruned during search, a property known as **safe pruning** (Li et al. 2018).

Controllable Simple Numeric Problems

Extending numeric planning problems with control variables makes the decision space of each state potentially infinite. In particular, a single action containing a control variable can generate infinitely many grounded instantiations, i.e., infinitely many pairs of an action and a control assignment. For instance, an action `heat-up` with a control variable specifying the target temperature can be instantiated with arbitrarily many numeric values (e.g., $\langle$`heat-up` 20.1$\rangle$, $\langle$`heat-up` 20.2$\rangle$, $\dots$).

The infinitude of the action space directly impacts the heuristic computation. Recall that, in the subgoaling relaxation, an action is considered a possible achiever of a condition when the regressed condition holds in the current state. With control variables, a single condition may therefore have infinitely many possible achievers, making the explicit computation of the heuristic derived from the relaxation infeasible.

A common strategy in related work facing similarly prohibitive costs is to compile the original controllable simple numeric problem into a simpler numeric model that relaxes certain aspects of the dynamics. Heuristic estimates are then computed over this compiled problem, which is more amenable to explicit reasoning and efficiently captures an informative approximation of the original task. This compilation-based approach is recurrent in the literature and can be seen, for instance, in work such as the effect-abstraction based relaxation in (Li et al. 2018) for numeric planning problems with linear effects or the HTN planning abstraction for guiding search (Höller et al. 2019).

Following this strategy, we seek to identify a subset of numeric planning problems with control variables whose compilation yields simple numeric planning problems. Since the subgoaling relaxation in standard numeric planning (*without* control variables) admits a closed-form computation only for the class of simple numeric planning problems (Scala et al. 2020a), our goal is to characterize an analogous class in the control-variable setting, which we will call *controllable simple numeric problems*, enabling the effective use of the subgoaling relaxation in this setting through a compilation.

Definition 4 (Simple numeric planning problem with control variables)**.** *A **simple numeric planning problem with control variables**, or a **controllable simple numeric planning problem**, is a numeric planning problem with control variables $\mathcal{P} = (F, X, U, A, s_0, G)$ such that*

- *Every numeric condition of the problem $(\xi \bowtie 0)$, i.e., every condition in $\mathrm{Pre}_{\mathbb{Q}}(a)$ $\forall a \in A$ and in $G_{\mathbb{Q}}$, is of the*

form:
  - $\xi = \xi_X - \xi_U$ *such that* $\xi_X \in \text{Expr}(X)$ *is linear and* $\xi_U \in \text{Expr}(U)$*;*
  - $\bowtie \in \{>, \geq\}$*;*

  We will denote such conditions as $(\xi_X \rhd \xi_U)$*, or as* $\sum_{x \in X} w_x \cdot x \rhd \xi_U$*;*
- *Every assisgnment of the problem, i.e., every assignment in* $\text{Eff}_{\mathbb{Q}}(a)$ $\forall a \in A$*, is of the form* $x \mathrel{+}= \xi$*, where* $\xi \in \text{Expr}(U)$.

The set of all controllable simple numeric problems is a subset of the set of all numeric planning problems with control variables.

Example 1. *Consider a controllable simple numeric problem defined as follows:*

- *Fluents:* $F = \emptyset$*,*
- *Numeric state variables:* $X = \{x, y\}$*,*
- *Control variables:* $U = \{u_1, u_2\}$*,*
- *Initial state:* $x = 0$*,* $y = 0$*,*
- *Numeric goal:* $x > 20$*,*
- *Action a: preconditions* $x > u_1$*,* $y > u_2$ *and effects* $x \mathrel{+}= 2u_1 + u_2$*,* $y \mathrel{+}= u_1 - 3u_2$.

This problem is a controllable simple numeric problem because the goal and action preconditions are linear with respect to X*, and the effects of the action are of the desired form, i.e., additive and linear only with respect to* X.

Controllable simple numeric problems are those where conditions are linear in the state variables and effects depend only on control variables. Since control variables are *free and bounded* variables that are not part of the state, they can be viewed as "lifted numbers". Instantiating their values therefore yields a simple numeric planning problem.

Optimistic Compilation

In this section, we present a compilation that transforms controllable simple numeric planning problems by concretizing the control variable expressions into specific numeric values. To lay the groundwork, we first briefly review the closed arithmetic of intervals:

Definition 5 (Closed arithmetic of intervals)**.** *Given* $x = [\underline{x}, \overline{x}]$ *and* $y = [\underline{y}, \overline{y}]$*, then:*

- $x + y = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$
- $x - y = [\underline{x} - \overline{y}, \overline{x} - \underline{y}]$
- $x \cdot y = [\min(\underline{x} \cdot \underline{y}, \underline{x} \cdot \overline{y}, \overline{x} \cdot \underline{y}, \overline{x} \cdot \overline{y}), \max(\underline{x} \cdot \underline{y}, \underline{x} \cdot \overline{y}, \overline{x} \cdot \underline{y}, \overline{x} \cdot \overline{y})]$

Given an arithmetic expression over U, $\xi \in \text{Expr}(U)$, we denote by $Dom(\xi)$ the interval resultant from applying the arithmetic operation to the domain intervals of all variables. For example, if $Dom(u_1) = [1, 2]$ and $Dom(u_2) = [0, 3]$, then $Dom(3u_1 + u_2) = 3 \cdot [1, 2] + [0, 3] = [3, 6] + [0, 3] = [3, 9]$. We will refer to the lower and upper bound of $Dom(\xi)$ as its **extreme valuations**, i.e., $\underline{Dom(\xi)}$ and $\overline{Dom(\xi)}$. We define the proposed compilation by considering the extreme valuations of each expression in the original problem that depends on control variables.

Definition 6 (Optimistic compilation)**.** *Given a controllable simple numeric planning problem* $\mathcal{P} = (F, X, U, A, s_0, G)$*, we define the* ***optimistic compilation*** *of* $\mathcal{P}$ *as a new problem,* $\mathcal{P}_O = (F, X, A_O, s_0, G)$*, which results from substituting every expression with its extreme valuations. That is,* A_O *is a set* $A_O = \bigcup_{a \in A} \Lambda(a)$ *such that:*

- *Given* $a \in A$ *with* $\mathit{Eff}_{\mathbb{Q}}(a) = \{(x_1 \mathrel{+}= \xi_{x_1}), \ldots, (x_n \mathrel{+}= \xi_{x_n})\}$*, we define* $\Lambda(a) = \{a_{\lambda_1, \ldots, \lambda_n} \mid \lambda_i \in \left\{\underline{Dom(\xi_{x_i})}, \overline{Dom(\xi_{x_i})}\right\} \ \forall i \in \{1, \ldots, n\}\}$*;*
- $\mathit{Eff}_{\mathbb{Q}}(a_{\lambda_1, \ldots, \lambda_n}) = \{(x_1 \mathrel{+}= \lambda_1), \ldots, (x_n \mathrel{+}= \lambda_n)\}$*;*
- *For every numeric condition* $\psi \equiv (\xi_X \rhd \xi_U)$ *appearing in* $\mathit{Pre}_{\mathbb{Q}}(a)$*, it is transformed to* $\psi_O \equiv (\xi_X \rhd \underline{Dom(\xi_U)})$ *in* $\mathit{Pre}_{\mathbb{Q}}(a_{\lambda_1, \ldots, \lambda_n}))$ $\forall \lambda_1, \ldots, \lambda_n$.

This compilation produces a finite problem by exhaustively enumerating all extreme valuations of the control-dependent effects of each action. In parallel, all preconditions are relaxed to their lower bounds, intuitively ensuring that any action that is executable in the original problem under some valuation of the control variables remains executable in the compiled task. For this reason, this transformation is referred to as an **optimistic compilation**.

Example 2. *Consider Example 1 from the previous section, with control variable domains* $Dom(u_1) = [0, 4]$ *and* $Dom(u_2) = [3, 5]$*. First, we compute the ranges of the linear expressions:*

$$Dom(2u_1 + u_2) = [3, 13], \quad Dom(u_1 - 3u_2) = [-15, -5].$$

The optimistic compilation generates four actions corresponding to the extreme bounds of these expressions. All actions share the same preconditions, coming from the original preconditions $x > u_1$*,* $y > u_2$*:* $x > 0, y > 3$*. The actions and their effects are:*

Action	*Effect on* x	*Effect on* y
$a_{3,-15}$	$x \mathrel{+}= 3$	$y \mathrel{+}= -15$
$a_{13,-15}$	$x \mathrel{+}= 13$	$y \mathrel{+}= -15$
$a_{3,-5}$	$x \mathrel{+}= 3$	$y \mathrel{+}= -5$
$a_{13,-5}$	$x \mathrel{+}= 13$	$y \mathrel{+}= -5$

Each action represents one combination of extreme values for the expressions involved in the control variables.

The first thing we need to prove is that, in fact, this compilation induces a simple numeric planning problem.

Theorem 1. *Given a controllable simple numeric planning problem* $\mathcal{P} = (F, X, U, A, s_0, G)$*, the optimistic compiled task* $\mathcal{P}_O$ *is a simple numeric planning problem.*

Proof. Since $\mathcal{P}$ is controllable simple, conditions are linear with respect to X, so in $\mathcal{P}_O$ conditions are linear given that the control variable expressions become constants. Effects also become additive with respect to a constant for the same reason. □

After establishing that the optimistic compilation $\mathcal{P}_O$ is indeed a simple numeric planning problem (Theorem 1), we now focus on its subgoaling relaxation, denoted as $\mathcal{P}_O^1$. We know that $\mathcal{P}_O^1$ is a relaxation of $\mathcal{P}_O$. Nevertheless, $\mathcal{P}_O$ **is not a relaxation** of $\mathcal{P}$.

Example 3. *Consider the following example: $\mathcal{P}$ defines a control variable $u \in [0, 10]$, an initial state $x = 0$, and a single action a with no preconditions and effect $x := x + u$. The goal condition is $13 \leq x \leq 14$. In this setting, the sequence $\langle a, 10\rangle, \langle a, 3.5\rangle$ constitutes a valid plan for the original problem $\mathcal{P}$. Under the optimistic compilation $\mathcal{P}_O$, the problem contains only two discrete actions, which correspond to the two extreme valuations: a_0 with effect $x := x + 0$ and a_{10} with effect $x := x + 10$. In $\mathcal{P}_O$, the original goal $13 \leq x \leq 14$ is unreachable, because no combination of these actions can satisfy both bounds simultaneously.*

In Example 3, applying the subgoaling relaxation $\mathcal{P}^1_O$ decomposes the goal into the independent subgoals $x \geq 13$ and $x \leq 14$. Each subgoal is individually achievable in $\mathcal{P}_O$, and therefore $\mathcal{P}^1_O$ is solvable, while $\mathcal{P}_O$ is not. Following this example, the following theorem formally shows that, for any $\mathcal{P}$, the relaxation property is recovered when considering the subgoaling relaxation $\mathcal{P}^1_O$.

Theorem 2. *Given $\mathcal{P} = (F, X, U, A, s_0, G)$ a controllable simple numeric planning problem, let $\Pi_{\mathcal{P}}$ be the set of solutions for $\mathcal{P}$ and $\Pi_{\mathcal{P}^1_O}$ be the set of solutions for $\mathcal{P}^1_O$. Then $\Pi_{\mathcal{P}} \neq \emptyset \implies \Pi_{\mathcal{P}^1_O} \neq \emptyset$, i.e., the relaxed compiled task $\mathcal{P}^1_O$ recovers the relaxation property with respect to the original problem $\mathcal{P}$.*

Proof. If $\Pi_{\mathcal{P}} \neq \emptyset$, let $\pi \in \Pi_{\mathcal{P}}$ be a solution for $\mathcal{P}$. We proceed by induction over π.

Base case. A condition reached by an empty plan is trivially satisfied in the initial state s_0, which is the same in both compilations.

Inductive step. A condition reached by π that is not true in the initial state implies the existence of some pair $\langle a_i, \mu_i\rangle$ in π making ψ true from some reachable state s_{i-1}. We want to discover a possible achiever in the relaxation that works towards ψ. Since $\mathcal{P}$ is controllable simple, every condition ψ is of the form $\xi_X \rhd \xi_U$, with ξ_X being a linear expression over X, which is equivalent to $\psi \equiv \sum_{x_i} w_{x_i} \cdot x_i \rhd \xi_U$, where $\xi_U \in \text{Expr}(U)$. In the optimistic compilation $\mathcal{P}_O$, this condition is abstracted by replacing the control-dependent expression with its lower bound, yielding the compiled condition $\psi_O \equiv \sum_{x_i \in X} w_{x_i} \cdot x_i \rhd \underline{Dom(\xi_U)}$. Hence, the set of states that satisfy the compiled condition in $\mathcal{P}_O$ is a superset of those satisfying the original condition in $\mathcal{P}$. In particular, since $s_{i-1} \models_{\mu_i} \text{Pre}(a_i)$, by the induction hypothesis it follows that $s_{i-1} \models \text{Pre}(a_{i\lambda_1,\ldots,\lambda_n})\ \forall \lambda_1, \ldots, \lambda_n$.

Now, it is just a matter of choosing the right lambdas in order for the effects to work towards fulfilling ψ. This can be computed by calculating the net effect $N_{\psi,a}$ of each action $a_{i\lambda_1,\ldots,\lambda_n}$ and choosing the right extreme values for it to be positive. For every action a, the net effect before the compilation, or the contribution of the action towards fulfilling the condition, can be computed as a trivial generalization of the net effect for standard numeric planning (Scala et al. 2020a):

$$N_{\psi,a} = \sum_{\substack{x_i \in lhs(e) \\ e \in \text{Eff(a)}}} w_{x_i} \cdot \xi_{x_i}$$

Where each ξ_{x_i} correspond to the right side of the assignment $x_i \mathrel{+}= \xi_{x_i}$. But $N_{\psi,a}$ in this setting is an expression that depends on control variables; concretely, its domain can be calculated as an interval regarding the close arithmetic:

$$Dom(N_{\psi,a}) = \sum_{\substack{x_i \in lhs(e) \\ e \in \text{Eff(a)}}} w_{x_i} \cdot Dom(\xi_{x_i})$$

If the domain of the net effect is greater than zero, then the action is a possible achiever for the condition because it has a positive contribution on the condition. $Dom(N_{\psi,a})$ is an interval that **we actually know has a positive side**, because of the inductive hypothesis, since there exists some control valuation μ_i that actually makes ψ fulfillable, which also means that the substitution of the expression using the control valuation, $subs_\mu(N_{\psi,a})$, follows $subs_\mu(N_{\psi,a}) \in Dom(N_{\psi,a})$ and $subs_\mu(N_{\psi,a}) > 0$.

We choose lambdas taking into account the sign of each w_{x_i}. If $w_{x_i} > 0$, we define $\lambda_i := \overline{Dom(\xi_{x_i})}$, and if $w_{x_i} < 0$, we define $\lambda_i := \underline{Dom(\xi_{x_i})}$. With this choice of lambdas, $a_{\lambda_1,\ldots,\lambda_n}$ is a possible achiever of ψ_O, because we are taking the extreme values for the expressions and we know that the net value has a positive side by induction.

Finally, observe that the subgoaling relaxation considers reachability of conditions separately, and thus, since all conditions are reachable, so is any conjunction of them. □

Reducing the Complexity of the Optimistic Compilation

The optimistic compilation $\mathcal{P}_O$ is exponential in the number of action effects, since we generate $2^{|\text{Eff}_{\mathbb{Q}}(a)|}$ compiled actions for every action a in $\mathcal{P}$; which makes it impractical. Nevertheless, following the proof of Theorem 2 reveals that the set of optimistic actions can actually be reduced. The objective of this section is thus to reduce the compilation size by leveraging theoretical properties from the proof of Theorem 2. We first define the *sign choice function* as follows:

Definition 7 (Sign choice function)**.** *Let $\mathcal{P} = (F, X, U, A, s_0, G)$ be a controllable simple numeric planning problem and let $a \in A$. Let $\text{Eff}_{\mathbb{Q}}(a) = \{(x_1 \mathrel{+}= \xi_{x_1}), \ldots, (x_n \mathrel{+}= \xi_{x_n})\}$ be the set of numeric effects of a. For a given condition $\psi \equiv \sum_{x_i \in X} w_{x_i} x_i \rhd \xi_U$, we define the **sign choice function of a given effect** $x_i \mathrel{+}= \xi_{x_i}$ as*

$$\chi_{\psi,a}(x_i \mathrel{+}= \xi_{x_i}) = \begin{cases} \overline{Dom(\xi_{x_i})}, & \text{if } w_{x_i} > 0 \\ \underline{Dom(\xi_{x_i})}, & \text{if } w_{x_i} < 0 \\ \overline{0}, & \text{if } w_{x_i} = 0 \end{cases}$$

The signature of an action uses the sign choice function to select only those actions from the optimistic compilation that have a positive net effect on the condition, i.e., the possible achievers.

Definition 8 (Signature compilation)**.** *Let $\mathcal{P}$ be a controllable simple numeric planning problem. Let Ψ_a be the set of all relevant conditions for a, i.e., every condition that contains a variable affected by some effect of a. For each action*

$a \in A$ with effects $\{(x_1 \mathrel{+}= \xi_{x_1}), \ldots, (x_n \mathrel{+}= \xi_{x_n})\}$ and each atomic condition $\psi \in \Psi_a$, we define a compiled action a_ψ whose numeric effects are $Eff_{\mathbb{Q}}(a_\psi) = \{(x_1 \mathrel{+}= \chi_{\psi,a}(x_i \mathrel{+}= \xi_{x_1})), \ldots, (x_n \mathrel{+}= \chi_{\psi,a}(x_i \mathrel{+}= \xi_{x_n}))\}$. Let

$$A_\Sigma := \{a_\psi \mid a \in A,\ \psi \in \Psi_a\}.$$

The signature compilation of $\mathcal{P}$ is $\mathcal{P}_\Sigma = (F, X, A_\Sigma, s_0, G)$.

We can further reduce the set of actions by collapsing non-zero components, that is, by removing effect entries that do not participate in any relevant condition. This allows multiple action signatures to be merged when they differ only in components that are irrelevant for the conditions under consideration. The following example illustrates this situation.

Example 4. *Continuing Example 2, recall that a is the only action with effects $e_1 : x \mathrel{+}= 2u_1 + u_2$ and $e_2 : y \mathrel{+}= u_1 - 3u_2$, and compiled preconditions $x > 0$ and $y > 3$. The numeric goal is $x > 20$. The conditions relevant to the action a are the goal $x > 20$ and the numeric preconditions $x > 0$ and $y > 3$, since they involve variables that are affected by the effects of a (namely, x and y).*

- *The goal $\psi_1 \equiv x > 20$ and the condition $\psi_2 \equiv x > 0$ can be rewritten as $1 \cdot x + 0 \cdot y > 20$ and $1 \cdot x + 0 \cdot y > 0$, respectively, and thus, $w_x = 1$ and $w_y = 0$. Therefore:*
  - *$\chi_{\psi_1,a}(x \mathrel{+}= 2u_1 + u_2) = \overline{2u_1 + u_2} = 13$, since $w_x > 0$*
  - *$\chi_{\psi_1,a}(y \mathrel{+}= u_1 - 3u_2) = 0$, since $w_y = 0$*

  which both generate the same action $a_{\psi_1} = a_{\psi_2} = a_{13,0}$, with effects $\{x \mathrel{+}= 13, y \mathrel{+}= 0\}$;
- *Condition $\psi_3 \equiv y > 3$ can be rewritten as $0 \cdot x + 1 \cdot y > 0$, and thus, $w_x = 0$ and $w_y = 1$. Therefore:*
  - *$\chi_{\psi,a}(x \mathrel{+}= 2u_1 + u_2) = 0$, since $w_x = 0$*
  - *$\chi_{\psi,a}(y \mathrel{+}= u_1 - 3u_2) = \overline{u_1 - 3u_2} = -5$, since $w_y > 0$*

  which generates the action $a_{\psi_3} = a_{0,-5}$ with effects $\{x \mathrel{+}= 0, y \mathrel{+}= -5\}$.

Both cases select extreme effects for either x or y, and leave the other variable unchanged. Consequently, actions $a_{\psi_1} = a_{\psi_2} = a_{13,0}$ and $a_{\psi_3} = a_{0,-5}$ can be collapsed into a single action that increments both variables at the same time: $a_{13,-5}$, with effects $\{x \mathrel{+}= 13, y \mathrel{+}= -5\}$.

The signature compilation can also be used to detect unsolvability of the problem, since it exposes actions whose effects do not contribute positively to any goal-relevant condition. The resulting action set A_Σ is still a subset of A_O and preserves the relaxation property established in the previous theorem.

Proposition 1. *Given a controllable simple numeric planning problem $\mathcal{P}$, then $\mathcal{P}_\Sigma$ is a relaxation under the subgoaling relaxation. In other words, replacing A_O by the reduced set A_Σ preserves the relaxation property.*

Proof. In the compilation to $\mathcal{P}_\Sigma$, we construct the reduced set $A_\Sigma \subseteq A_O$ by selecting only those instantiations that can act as achievers of at least one goal or subgoal. That is, for every subgoal g, if an action $a \in A_O$ can achieve g under some instantiation of its control variables, the corresponding instantiation is included in A_Σ. Conversely, any action that cannot contribute to achieving any subgoal is discarded.

Since A_Σ is a subset of A_O that preserves all possible achievers for every subgoal, the subgoaling relaxation remains valid: any relaxed plan that exists in $\mathcal{P}$ can also be constructed in $\mathcal{P}_\Sigma$. No new constraints are introduced, and no potential achievers are removed from consideration for the purpose of the relaxation. □

The size of the signature compilation can be proven to be linear with respect to the number of conditions of the problem, i.e., linear with respect to the size of the problem.

Proposition 2 (Size bound of the signature compilation)**.** *Let $\mathcal{P}$ be a controllable simple numeric planning problem and let $\mathcal{P}_\Sigma = (F, X, A_\Sigma, s_0, G)$ be its signature compilation. Let Ψ denote the set of all numeric atomic conditions appearing in action preconditions and in the goal. Then the number of compiled actions satisfies*

$$|A_\Sigma| \leq |A| \cdot |\Psi|.$$

Hence, the size of the compilation is polynomial in the size of the original problem with respect to the number of atomic numeric conditions.

Proof. By Definition 8, for each action $a \in A$ and each relevant condition for a, $\psi \in \Psi_a$, the signature compilation generates at most one action a_ψ. □

We define the heuristic h^{add}_Σ as the numeric heuristic h^{add} applied to the compiled problem obtained through the signature compilation of the problem.

Extracting More Information from Subgoaling Heuristics

The additive heuristic h^{add}_Σ computes costs by summing relaxed action contributions for each subgoal. While this provides strong and valuable search guidance, this relaxation ignores positive interactions among actions, causing a systematic overestimation of the real plan cost, especially in domains with overlapping effects. This overestimation arises directly from counting the cost of actions multiple times when they contribute to multiple subgoals, even though a single action instance may suffice for several achievements in the real plan.

To address this limitation, the h^{mrp} heuristic (Scala et al. 2020b), based on multi-repetition relaxed plans in simple numeric planning problems, merges redundant action contributions by tracking the maximum required count for each action rather than simply accumulating all individual contributions. This approach is also safe pruning and allows h^{mrp} to more accurately capture the dependencies and synergies between numeric actions and subgoals, reducing inadmissible overestimation and providing more informed search guidance for simple numeric planning problems. We then define h^{mrp}_Σ as the h^{mrp} heuristic under the signature compilation.

Experiments

In this section, we analyze the practical impact of the signature compilation used in the h^{add}_{Σ} and h^{mrp}_{Σ} heuristics. We implemented both heuristics in the ENHSP planner (Scala et al. 2016) and compared their performance with existing heuristics within the Delayed Partial Expansions (DPEX) algorithm. We also evaluated their performance against the NextFLAP planner.

DPEX instantiation We used DPEX with $f(n) = r_h(n, s)$ as node evaluation criterion, where $r_h(n, s) = h(s) + \log(1 + n)$, i.e., logarithmic rectification, where n is the number of partial expansions of s. For each partial expansion, we sample $K = 5$ successors. We used uniform sampling for the sampling function ϕ. Experiments were conducted with a fixed seed on a 12th Gen Intel(R) Core(TM) i9-12900KF CPU running Ubuntu 22.04 LTS, with a 30-minute timeout and 8 GB memory limit.

Baselines. Our experiments are evaluated against baseline search configurations that rely on domain-independent heuristics. Specifically, we compared against a blind heuristic h^0 and the Manhattan goal-counting heuristic h^{mgc}, which computes the Manhattan distance to the numeric subgoals and counts the number of satisfied propositional subgoals. We also compared against a run of the deterministic planner NextFLAP (Sapena, Onaindia, and Marzal 2024), which is capable of handling control parameters. NextFLAP is a planner based on forward partial-order planning (POP) search that integrates SMT solving into the search process. Rather than treating control parameter instantiations as explicit branching choices, the planner incrementally constructs sequences of symbolic actions while accumulating constraints over numeric variables, including control parameters. Periodic calls to an SMT solver check the consistency of these constraints, allowing infeasible branches to be pruned during search.

Domains. We used three domains from the repository of POPCORN, the first planner that originally introduced the notion of control variables[1]: CASHPOINT, PROCUREMENT and TERRARIA. These domains only have propositional goal conditions and are relatively hard, especially the TERRARIA domain, which uses six control variables at the same time, and the PROCUREMENT domain that has a lot of causality between actions. We also used four domains from the numeric IPC[2] introduced in (Aso-Mollar et al. 2025b): BLOCKGROUPING, COUNTERS, DRONE, which have numeric goal conditions, and SAILING, which has only propositional conditions. The DRONE domain is the hardest due to its many dead-ends, while the SAILING domain stands out for its unbounded boat positions. For each of these seven domains, we considered both continuous (C) and discrete (D) versions, i.e., we used control variables with and without decimals, respectively. For every domain, we generated 20 problems of increasing size, which can be found in the GitHub repository.

[1] github.com/Emresav/popcorn

[2] github.com/ipc2023-numeric/ipc2023-dataset

Domain	$\lvert A\rvert$	$\lvert A_O\rvert$	$\lvert A_\Sigma\rvert$
BLOCKGROUPING	114.60	229.20	229.20
CASHPOINT	7379.00	8758.00	7379.00
COUNTERS	15.00	30.00	26.00
DRONE	14.50	25.375	20.50
PROCUREMENT	470.00	1700.50	470.00
SAILING	22.70	53.45	53.45
TERRARIA	40.00	120.00	40.00

Table 1: Average size of action sets A, A_O and A_Σ.

Domain	h^0	h^{mgc}	h^{add}_{Σ}	h^{mrp}_{Σ}	h_Σ	NF
BLOCKG.-C (20)	0	20	20	20	20	18
BLOCKG.-D (20)	0	20	20	20	20	16
CASHP.-C (20)	10	8	20	20	20	10
CASHP.-D (20)	10	17	20	20	20	9
COUNT.-C (20)	12	20	20	19	20	19
COUNT.-D (20)	16	20	20	20	20	17
DRONE-C (20)	12	0	2	13	13	0
DRONE-D (20)	18	18	18	18	18	0
PROCUR.-C (20)	8	10	16	10	16	10
PROCUR.-D (20)	5	9	14	11	14	10
SAIL.-C (20)	0	14	12	20	20	2
SAIL.-D (20)	1	12	20	20	20	0
TERR.-C (20)	6	6	20	4	20	0
TERR.-D (20)	6	7	20	3	20	0
TOTAL (280)	105	181	242	218	261	111

Table 2: Coverage per domain and heuristic. Column h_Σ corresponds to instances solved by either one of h^{add}_{Σ} or h^{mrp}_{Σ}, and NF corresponds to instances solved by NextFLAP.

Heuristic setup. Table 1 compares the average number of actions in the original problem A with the number of actions in the optimistic compilation (A_O) and the signature compilation (A_Σ). The results show that the number of compiled actions of A_Σ is much lower than the optimistic compilation. In some domains such as CASHPOINT, PROCUREMENT, and TERRARIA, the size of A_Σ is identical to that of the original action set A. In contrast, the optimistic compilation A_O introduces a significantly large number of actions. This difference is particularly noticeable in domains such as PROCUREMENT, where each action contains on average three control-dependent effects.

Results. Table 2 reports coverage results for each domain and problem. For DPEX, we performed five runs per instance and counted the problem as solved if at least one run succeeded. The column h_Σ reports problems solved using either h^{add}_{Σ} or h^{mrp}_{Σ}, while the column NF reports problems solved by NextFLAP.

Overall, our compilation-based approach is consistently dominant across all domains. We observe that either DPEX with h^{add}_{Σ} or h^{mrp}_{Σ} heuristic systematically outperforms DPEX with h^{mgc}. As for NextFLAP, we observe that its performance is comparable to DPEX with blind search (h^0) in terms of solved problems.

The baseline heuristics h^0 and h^{mgc} rely exclusively on the current state to compute their estimates; they do not perform any form of subgoaling nor exploit causal relationships in the domain. Instead, they provide a purely numer-

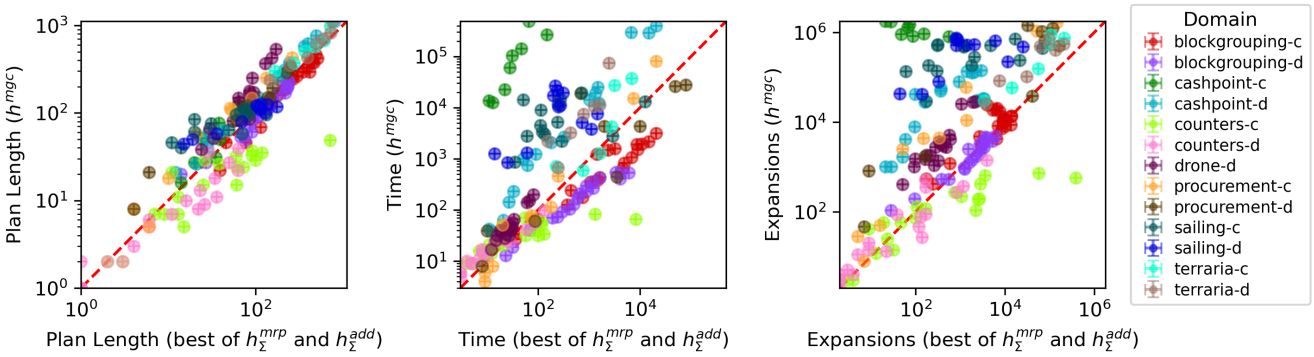

Figure 1: Comparison of the runs of h^{add}_Σ or h^{mrp}_Σ heuristic versus baselines in terms of plan length, time and number of (partial) expansions, for the best case among the 5 runs.

ical notion of distance based only on the current valuation of the state variables and their desired values in the goal. For instance, a state with $x = 0$ and $y = 0$ and a goal condition $x > 10$ and $y > 3$ yields the h^{mgc} estimate $|10 - 0| + |3 - 0| = 13$, which is computed independently of the actions required to achieve such conditions. That is, h^{mgc} does not capture whether there exist actions that can increase x or y, nor how these variables may interact through the problem dynamics. In contrast, h^{add}_Σ and h^{mrp}_Σ explicitly reason about *possible achievers* in the infinite transition system. This is enabled by the compilation, which induces a finite abstraction where control variables are fixed to extreme valuations, allowing the heuristic to capture action instances that can optimistically support goal conditions. This explains the rise in coverage when considering h^{add}_Σ or h^{mrp}_Σ.

Lower coverage is observed for h^{add}_Σ and h^{mrp}_Σ in domains such as PROCUREMENT and DRONE, which can be attributed to the assumptions introduced by the compilation's relaxation. By ignoring the actual values of control variables and fixing them to extreme defaults, the heuristics may commit too aggressively to actions without properly accounting for feasibility conditions that depend on these parameters. This limitation is particularly impactful in these domains, as they involve a larger number of complex conditions over control variables, where feasibility is tightly coupled to specific parameter valuations.

In other domains, such as PROCUREMENT and COUNTERS, empirical results show that h^{add}_Σ often outperforms h^{mrp}_Σ. The additive heuristic h^{add}_Σ is known to overestimate the true cost, as it sums the contributions of actions independently. Interestingly, a nontrivial interaction seems to emerge between the compilation's underestimation (committing too aggressively to actions) and the heuristic's overestimation (its tendency to overestimate the true cost).

Figure 1 presents a more detailed comparison between DPEX with h_Σ (best of h^{add}_Σ and h^{mrp}_Σ) and h^{mgc} in terms of plan length, time and number of (partial) expansions for each domain individually. We evaluated only the instances solved by both of the proposed approaches (h_Σ and h^{mgc}), comparing the best-performing run of either strategy. Extended figures with additional comparisons can be found in the GitHub repository.

We can observe a clear dominance of our approach over Manhattan goal-counting h^{mgc}. The h^{mgc} heuristic is known to perform particularly well in strongly numeric domains, where the problem structure is largely determined by numeric progress toward the goals. In line with this characteristic, h^{mgc} produces slightly shorter plans in purely numerical domains such as BLOCKGROUPING and COUNTERS, when considering the instances solved by both approaches. In contrast, our approach generally requires fewer node expansions than h^{mgc} in most solved instances and, consequently, is generally faster.

Conclusions and Future Work

As a conclusion, we have proposed a compilation that allows the use of heuristics based on the numeric subgoaling relaxation that is applicable to the setting of control variables and achieves competitive results compared to standard domain-independent heuristics. This compilation is based on an optimistic relaxation, which transforms the original problem into a tractable simple numeric planning problem. This compilation, however, is exponential in the number of effects of the problem at hand, so to mitigate this potential combinatorial explosion we introduce the signature compilation, which retains only actions that contribute positively to conditions.

Our approach provides a practical and theoretically sound method for applying standard numeric heuristics based on subgoaling in the context of control variables, supported with experimental evaluation. We observed that the over-approximation inherent in the optimistic compilation, which ignores the control component in the decision-making, can lead to decreased performance in some domains.

This work opens a promising direction for heuristic computation in problems with infinite actions. In particular, identifying informative valuations for possible achievers could help bridge the gap with geometric domains. Although challenging, our approach provides a solid foundation for developing more informed heuristics in this setting.

Acknowledgements

This work has been partially supported by the project I+D+i AEI PID2021-127647NB-C22 funded by MICIU/AEI/10.13039/501100011033 and by FEDER/UE; and by the GENERALITAT VALENCIANA project PROMETEO CIPROM/2023/23. Enrico Scala has been supported by the Italian Ministry of University and Research within the PRIMA 2024 programme project "Optimizing Water Resources in Coastal Areas using Artificial Intelligence" (AI4WATER – D53C25000510006). Ángel Aso-Mollar has been partially supported by the FPU21/04273.

References

Aso-Mollar, Á.; Aineto, D.; Scala, E.; and Onaindia, E. 2025a. Handling Infinite Domain Parameters in Planning Through Best-First Search with Delayed Partial Expansions. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, 8456–8464. International Joint Conferences on Artificial Intelligence Organization.

Aso-Mollar, Á.; Aineto, D.; Scala, E.; and Onaindia, E. 2025b. A Sampling Approach to Planning with Infinite Domain Control Variables. In *Proceedings of the Thirty-Fifth International Conference on Automated Planning and Scheduling, ICAPS-25*, 149–153.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence*, 129(1): 5–33.

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

Heesch, R.; Cimatti, A.; Ehrhardt, J.; Diedrich, A.; and Niggemann, O. 2024. A Lazy Approach to Neural Numerical Planning with Control Parameters. In *European Conference on Artificial Intelligence 2024*, Frontiers in Artificial Intelligence and Applications, 4262–4270. IOS Press.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2019. On Guiding Search in HTN Planning with Classical Planning Heuristics. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 6171–6175. International Joint Conferences on Artificial Intelligence Organization.

Li, D.; Scala, E.; Haslum, P.; and Bogomolov, S. 2018. Effect-Abstraction Based Relaxation for Linear Numeric Planning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 4787–4793. International Joint Conferences on Artificial Intelligence Organization.

Sapena, O.; Onaindia, E.; and Marzal, E. 2024. A Hybrid Approach for Expressive Numeric and Temporal Planning with Control Parameters. *Expert Systems with Applications*, 242: 122820.

Savaş, E.; Fox, M.; Long, D.; and Magazzeni, D. 2016. Planning Using Actions with Control Parameters. In *Procedings of the Twenty-Second European Conference on Artificial Intelligence, ECAI-16*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 1185–1193. IOS Press.

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

Scala, E.; Haslum, P.; Thiebaux, S.; and Ramirez, M. 2020a. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *Journal of Artificial Intelligence Research*, 68: 691–752.

Scala, E.; Saetti, A.; Serina, I.; and Gerevini, A. E. 2020b. Search-Guidance Mechanisms for Numeric Planning Through Subgoaling Relaxation. *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling*, 30(1): 226–234.

Shin, J.; and Davis, E. 2005. Processes and Continuous Change in a SAT-based Planner. *Artificial Intelligence*, 166(1-2): 194–253.