On the Optimality of Numeric Planning with Control Variables

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

Planning with actions parameterized by numeric values, also called control variables, raises fundamental challenges to cost-optimal decision making, as such values induce infinite decision spaces. Existing approaches focus on feasibility and reachability, usually assuming unitary action costs, and provide no principled notion of plan optimality beyond plan length. In this paper, we address this gap by introducing a cost-aware formulation in which action costs depend on numeric parameters. To reason about optimality, we build on the Delayed Partial Expansions (DPEX) framework, which performs sampling-based partial expansions while preserving probabilistic completeness. We identify a key limitation of DPEX that precludes explicit control over solution quality. To overcome this limitation, we propose an iterative deepening strategy that also ensures completeness. This yields a new algorithm, ID-DPEX, which provides strong and explicit optimality guarantees when combined with admissible heuristics. Empirical results show that ID-DPEX efficiently produces near-optimal solutions.

Introduction

Planning with control variables, originally introduced as control parameters, has recently emerged as a powerful framework for modeling decision-making problems in which actions are parameterized by values drawn from infinite numeric domains. In this setting, a single action schema gives rise to infinitely many decisions, each corresponding to a particular valuation of numeric parameters.

Early planning frameworks based on control parameters have primarily focused on feasibility and state reachability, largely abstracting away execution cost and plan-level optimality. In planners such as POPCORN (Savaş et al. 2016) and NextFLAP (Sapena, Onaindia, and Marzal 2024), control parameters are not involved in the search process and do not contribute to any notion of plan optimality. In both approaches, plans defer their valuation until a concretization step, where a weak notion of "optimality" is introduced, restricted to the instantiation of control parameters for a fixed plan. This concretization neither guides the search process nor discriminates between competing plans; instead, it is a necessary step to obtain a valid plan instantiation, typically by optimizing a numeric objective.

This paper addresses the lack of a plan-level notion of cost-optimality in planning frameworks with control parameters by introducing a cost-aware formulation that explicitly accounts for control valuations. Our contribution situates planning with control variables within a broader taxonomy of action cost models and provides the first explicit notion of cost-optimality for this class of problems. Reasoning about optimality in this setting poses significant algorithmic challenges. Control variables induce infinite decision spaces, rendering full successor expansion infeasible and preventing the direct application of classical cost-optimal search algorithms such as A*, Weighted-A*, or IDA* (Hart, Nilsson, and Raphael 1968; Pohl 1970; Korf 1985). Other partial-expansion techniques such as PEA and EPEA (Felner et al. 2012; Goldenberg et al. 2014) mitigate large branching factors by expanding only subsets of successors, but they are not designed to handle infinite branching, nor cost-aware metrics.

To address this issue, we build upon the Delayed Partial Expansions (DPEX) framework (Aso-Mollar et al. 2025b), which replaces full successor expansions with sampling-based partial expansions and relies on rectification functions to delay subsequent node expansions. We identify a fundamental limitation of these rectification functions that prevents DPEX from guaranteeing optimal solutions. To overcome this limitation, we impose a finite upper bound on the rectification functions while decoupling the heuristic estimation from the rectification growth. This results in a new algorithm called ID-DPEX, which is inspired by threshold-based search methods such as Iterative Deepening.

Our contributions are fourfold. First, we formalize numeric planning with control variables under control-dependent action costs, filling a gap in the existing taxonomy of planning cost models. Second, we introduce ID-DPEX, a cost-aware iterative search algorithm for infinite decision spaces, and prove that it is probabilistically complete under the same assumptions as DPEX. Third, we show that, when equipped with admissible heuristics, ID-DPEX returns solutions with explicit optimality guarantees. Finally, we propose an admissible heuristic tailored to numeric planning with control variables and prove its admissibility within the proposed framework.

Problem Formulation

We adopt the numeric planning with control variables formalization of (Aso-Mollar et al. 2025a). Let F be a finite

set of propositional state variables, X a finite set of numeric state variables, and U a finite set of numeric control variables. All numeric variables range over subsets of $\mathbb{Q}$, and the control variables domains are bounded. We denote by $\text{Expr}(X \cup U)$ the set of arithmetic expressions over state and control variables. A numeric planning problem with control variables is defined as follows.

Definition 1 (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 $a = (\text{Pre}(a), \text{Eff}(a))$. Preconditions are given by $\text{Pre}(a) = (\text{Pre}_{\mathbb{B}}(a), \text{Pre}_{\mathbb{Q}}(a))$, where $\text{Pre}_{\mathbb{B}}(a)$ is a Boolean formula over F, and $\text{Pre}_{\mathbb{Q}}(a)$ is a conjunction of numeric constraints of the form $(\xi \bowtie 0)$ with $\xi \in \text{Expr}(X \cup U)$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. Effects are given by $\text{Eff}(a) = (\text{Eff}_{\mathbb{B}}(a), \text{Eff}_{\mathbb{Q}}(a))$, where $\text{Eff}_{\mathbb{B}}(a)$ is a set of propositional assignments, and $\text{Eff}_{\mathbb{Q}}(a)$ is a consistent set of numeric assignments of the form $(x := \xi)$, with $x \in X$ and $\xi \in \text{Expr}(X \cup U)$;*
- *s_0 is the initial state, assigning truth values to variables in F and numeric values to variables in X;*
- *$G = (G_{\mathbb{B}}, G_{\mathbb{Q}})$ are the goal conditions, where $G_{\mathbb{B}}$ is a Boolean formula over F and $G_{\mathbb{Q}}$ is a formula of numeric constraints over X.*

A **plan** in this setting is a finite sequence $\pi = (\langle a_i, \mu_i \rangle)_{i=1}^{k}$, where each μ_i assigns values to the control variables in U. We define μ_i as **control valuations**, and we denote the set of all control valuations as $\mathcal{U}$. A plan is **valid** if all action preconditions are satisfied after substituting control variables according to the control valuation μ_i, and it is a **solution** if the resulting final state satisfies the goal conditions.

A valid plan prefix $\pi_{1:i}$ induces a unique reached state s_i obtained by sequentially applying the effects of the decisions $\langle a_j, \mu_j \rangle$ for $j = 1, \ldots, i$, after substituting control variables according to μ_j. Each decision $\langle a_i, \mu_i \rangle$ that yields a valid plan given a prefix and applied to its corresponding reached state s_i is a **decision** for s_i and $D(s_i)$ is the set of all decisions for s_i. The set of all states is defined as S.

As discussed in the introduction, existing planning frameworks with control variables have primarily focused on feasibility and state reachability, abstracting away execution cost and plan-level optimality (Savaş et al. 2016; Sapena, Onaindia, and Marzal 2024; Aso-Mollar et al. 2025b). These approaches assume unitary action costs and treat all applicable decisions as equally expensive, which limits their ability to capture domains in which control choices influence the execution cost of actions. To overcome this limitation, we introduce a **control-aware cost function**, associating a positive cost to each action-control pair. This allows modeling scenarios where control decisions affect both the dynamics and the accumulated cost of plans, yielding a more expressive and realistic framework for numeric planning with control variables.

Definition 2 (Control-aware cost function)**.** *Given a numeric planning problem with control variables $\mathcal{P} = (F, X, U, A, s_0, G)$, a **control-aware cost function** is a cost function associated to $\mathcal{P}$, i.e., $\lambda : A \times \mathcal{U} \to \mathbb{R}^+$ that maps a pair $\langle a, \mu \rangle$ to a positive number denoting its cost.*

With this, we have defined a cost function associated to a numeric planning problem with control variables. Given a control-aware cost function λ, the **accumulated cost** of a reached state s_i induced by a plan prefix $\pi_{1:i} = (\langle a_j, \mu_j \rangle)_{j=1}^{i}$ is defined as

$$g(s_i) = \sum_{j=1}^{i} \lambda(a_j, \mu_j)$$

For the initial state s_0, we define $g(s_0) = 0$. A solution plan π is said to be **optimal** if the final state it reaches has minimum accumulated cost g among all solution plans.

We can situate the proposed cost model within the landscape of action cost formulations in automated planning by distinguishing four increasingly levels of expressivity.

1. **Unitary action costs.** In the simplest setting, all actions are assumed to have unitary cost. As a result, the accumulated cost of a plan coincides with its length. This assumption is standard in classical planning and underlies optimality notions such as shortest plans.
2. **Constant action costs.** A more general model assigns a fixed numeric cost to each action. The cost of an action is independent of the state in which it is applied and of any additional parameters.
3. **Controllable action costs.** This work introduces a third level, where the cost of an action depends on associated control variables. The cost is no longer fixed per action but does not depend on a state variable.
4. **State-dependent action costs.** The most expressive setting allows action costs to depend on the current state, typically through numeric fluents. This model can be further extended to incorporate control variables, resulting in costs of the form $\lambda(s, a, \mu)$.

Example 1. *Control-aware cost functions allow modelling levels 1 through 3. Consider a numeric planning problem with control variables summarized as follows. We have a numeric state variable x representing the position of an agent on a line. The initial state is $s_0[x] = 0$, and the goal is $x \geq 10$. We have a control variable u with domain $[0, 5] \cap \mathbb{Q}$. The action set contains two actions modeling movement to the right and to the left: $a_r = move\_right, a_\ell = move\_left$. Both actions have no preconditions. Their numeric effects differ only in the sign of the displacement: $\text{Eff}_{\mathbb{Q}}(a_r) = \{x := x + u\}$ and $\text{Eff}_{\mathbb{Q}}(a_\ell) = \{x := x - u\}$. We now illustrate how different action cost models can be expressed using control-aware cost functions.*

- ***Unitary action costs.** The cost function is defined as $\lambda(a, \mu) = 1$ for all $a \in A$ and all $\mu \in \mathcal{U}$.*
- ***Constant action costs.** The cost function assigns fixed costs to actions, for example $\lambda(a_r, \mu) = 2$ and $\lambda(a_\ell, \mu) = 1$ for all $\mu \in \mathcal{U}$.*
- ***Controllable action costs.** The cost function depends on the control valuation, for instance $\lambda(a, \mu) = \mu[u]^2$ for both $a \in \{a_r, a_\ell\}$. In this setting, the cost of a decision is quadratic w.r.t. the magnitude of the displacement, i.e., the value of the control variable u, $\mu[u]$.*

Algorithm 1: DPEX$_{\phi,r_h}$

```
1: Input: Decision space function D(·), sampling function
   φ, rectification function r_h, NEC f, number of samples
   K, initial state s_0, goal condition G
2: Output: Goal 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 satisfies G then
7:       return true
8:    else
9:       for i = 1 to K do
10:         Sample a new decision in D(s) using φ(s)
11:         Apply sampled decision to s and generate s'
12:         Insert (f(s'), s') into Open
13:      end for
14:      Rectify f(s) using r_h
15:      Insert (f(s), s) into Open
16:   end if
17: end while
18: return false
```

In the remainder of this paper, we focus on the problem of finding optimal plans under the third level of expressiveness, namely controllable action costs.

Iterative Deepening Delayed Partial Expansions

This section develops a search strategy tailored to decision spaces induced by control variables, building on the objective of reasoning about optimality under controllable action costs. We build upon the S-BFS algorithm from (Aso-Mollar et al. 2025a), which enables forward search in infinite decision spaces via Delayed Partial Expansions (DPEX), and identify a limitation that prevents any meaningful control over solution quality without compromising the completeness of the algorithm. To reconcile these conflicting objectives, we introduce Iterative Deepening DPEX (ID-DPEX), an algorithm that progressively relaxes this limitation across successive runs, and prove its probabilistic completeness. Finally, we show that, when admissible heuristics are used, ID-DPEX provides explicit suboptimality guarantees.

Preliminaries: Delayed Partial EXpansions

The S-BFS algorithm (Aso-Mollar et al. 2025a) provides a principled approach to search with infinite decisions by combining selective exploration with mechanisms that ensure theoretically sound search behavior via the Delayed Partial EXpansions (DPEX) scheme.

Nodes generated during the search are prioritized according to a Node Evaluation Criterion (NEC). Given a *sampling function* ϕ that samples decisions over the decision space of a state s, $D(s)$, and a *rectification function* $r_h : \mathbb{Z}_0^+ \times S \to \mathbb{R}$ defined over a heuristic $h : S \to \mathbb{R}$ such that $r_h(0, s) = h(s)$, specific instances of this family are denoted as DPEXϕ, r_h.

Algorithm 1 presents the pseudocode of DPEX$_{\phi,r_h}$. The algorithm maintains a priority queue $Open$ ordered by the NEC value f, which is initialized with the initial state s_0 and its corresponding f-value (line 3). At each iteration, the algorithm extracts the most promising state (i.e., the one with the lowest f-value) (line 5). If the extracted state is a goal, the search terminates successfully (lines 6-7). If the queue becomes empty without reaching a goal, the algorithm reports failure (line 18).

Partial expansion. Instead of fully expanding all applicable decisions of a state s, K decisions are sampled from the decision space $D(s)$ using the sampling function $\phi(s)$ (lines 9-10). Each sampled decision is applied to s to generate a successor state s', which is then inserted into $Open$ with its corresponding f-value (lines 11-12).

Rectification and reinsertion. After a partial expansion, DPEX applies the rectification function r_h to update the NEC value (line 14). The state is then reinserted into $Open$ for further consideration in subsequent iterations (line 15).

Probabilistic completeness Rectification plays a central role in balancing exploration and exploitation, while maintaining search completeness. In (Aso-Mollar et al. 2025a), the authors proved that DPEX is probabilistically complete if the rectification functions are defined as monotonically increasing and tending to infinity, and if the sampling functions assign a non-zero probability to every possible decision of $D(s)$. That is, the probability of finding a solution is 1 as the number of iterations of the algorithm (line 4) tends to infinity.

Bounding the Rectification in DPEX

DPEX$_{\phi,r_h}$ equipped with an NEC $f = g + r_h$ that incorporates the accumulated cost g provides mild guarantees on the cost of the solutions it returns, as established by the following proposition, which follows from the properties of $Open$ as a priority-queue:

Proposition 1. *Let ϕ be a sampling function that, given $s \in S$, assigns a non-zero probability to every $s' \in D(s)$, and r_h a monotonic rectification function that tends to infinity when $n \to \infty$. Consider an open node s in the current search tree. Whenever a node s' in the subtree rooted at s is selected for expansion, its f-value satisfies $f(m, s') \leq f(n, s)$ where m and n denote the current numbers of partial expansions of s' and s, respectively.*

Proof. When s' is selected for expansion at some iteration, it must be the minimum f-value node in $Open$ at that time. In particular, since s is in $Open$ at that iteration with current value $f(n, s)$, it follows that

$$f(m, s') \leq f(n, s).$$

□

Particularly, this means that, if the heuristic function is goal-aware, i.e., $h(s_g) = 0$ for every goal state, the cost of the plan $g(s_g)$ is bounded by the f-value of any of the s_g's ancestors from the root node s_0 to s_g's parent node. In

practice, however, this notion is insufficient to guarantee any optimal solution **since the rectification function must tend to infinity**. Improving the theoretical guarantees of the algorithm would appear to require bounding the rectification functions in some way; however, doing so compromises the probabilistic completeness of DPEX. Therefore, it is necessary to devise a mechanism that balances both objectives.

For this reason, we propose a subset of rectification functions, termed bounded additive rectification functions. This subset is characterized by having a rectification term that is independent of the heuristic term and by being monotonic *with a finite bound*.

Definition 3 (Bounded additive rectification function)**.** *Given a heuristic function $h : S \to \mathbb{R}$, a **bounded additive rectification function** is a function $r_h : \mathbb{Z}_0^+ \times S \to \mathbb{R}$ such that $r_h(n, s) = h(s) + \rho(n)$ and:*

- $\rho(0) = 0$*;*
- $\rho(n') > \rho(n)$ *for* $n' > n$*, i.e., ρ is monotonically increasing;*
- *There exists some $C \in \mathbb{R}$ such that $\rho(n) \leq C \; \forall n \in \mathbb{Z}^+$.*

Given an unbounded additive rectification function $r_h(n, s) = h(s) + \rho(n)$, we can construct its boundification, r_h^C, by collapsing every $\rho(n) > C$ to $\rho(n) = C$. When a bounded additive rectification function is used, the rectification term has a finite upper limit. Consequently, solutions in certain regions of the search space may never become sufficiently competitive to be explored, as their combined heuristic and rectification values remain less favorable than those of alternative branches. When considering unitary costs, for instance, this would mean that the search cannot reach nodes of depth more than $h(s_0) + C$. More generally:

Proposition 2. *Given a bounded additive rectification function r_h with bound C, every state selected for expansion by DPEX has an f-value bounded by $h(s_0) + C$, provided that the sampling function ϕ satisfies the conditions of Proposition 1.*

Proof. Follows directly from Proposition 1 by considering the fact that $r_h(n, s_0) = h(s_0) + \rho(n) \leq h(s_0) + C$. □

This means that, if no solution state has cost of less than $h(s_0) + C$, DPEX will never find a solution to the problem. What can be done, however, is to make this bound dynamic over time. This approach allows, on the one hand, certain guarantees on the rectification and, on the other hand, an iterative increase of the heuristic. Building on these ideas, we introduce the Iterative Deepening DPEX (ID-DPEX) algorithm.

Algorithm 2 presents the pseudocode of ID-DPEX$_{\phi,r_h}$. The algorithm starts with an initial rectification bound C_0 (line 3), constructs the corresponding boundification r_h^C (line 5), and then executes DPEX$_{\phi,r_h^C}$ for at most M iterations (line 6). This iteration limit corresponds to terminating the main **while** loop of Algorithm 1-line 4 once M iterations have been reached. If a goal state is found, the algorithm returns success (line 8); otherwise, the rectification bound is increased by Δ (line 10) and a new run is initiated (line 4).

Algorithm 2: ID-DPEX$_{\phi,r_h}$

```
1:  Input: Rectification function r_h, sampling function φ,
    number of iterations per run M, initial rectification
    bound C_0, bound increment Δ
2:  Output: Goal reached
3:  C ← C_0
4:  while true do
5:     Construct r_h^C boundification from r_h
6:     goal_reached ← run DPEX_{φ,r_h^C} for M iterations
7:     if goal_reached then
8:        return true
9:     else
10:       Increase bound C by Δ
11:    end if
12: end while
```

We refer to this method as Iterative Deepening DPEX because it follows the classical iterative deepening paradigm. The algorithm executes a sequence of forward searches, each constrained by a fixed rectification bound C, and progressively increases this bound across runs. The probabilistic completeness of ID-DPEX relies on the convergence of the probability of failure across independent runs.

Theorem 1 (Probabilistic completeness of ID-DPEX)**.** *Under the assumptions of Proposition 1, and for a sufficiently large per-run iteration budget M, ID-DPEX is probabilistically complete.*

Proof. Let $\mathcal{P}$ be a numeric planning problem with control variables. Assume that the planning problem is solvable. Let π be any solution plan reaching a goal state s_g, and consider a sequence of states

$$s_0, s_1, \ldots, s_k = s_g$$

explored by ID-DPEX during an iteration, where each state s_{i+1} is obtained from s_i by applying the decision $\langle a_i, \mu_i \rangle$. Observe that, if m denotes the length of π, then $k \geq m$, and that a state s_i can be equal to some other state s_j if that state is resampled.

(1) We first show that there exists a finite clipping bound under which the entire solution path remains reachable during a suitable execution of DPEX. For each state s_i of the sequence, Proposition 2 states that a node can be expanded during a run with clipping bound C only if $f(s_i) \leq h(s_0)+C$. By definition of the clipped rectification function,

$$f(s_i) = g(s_i) + h(s_i) + \min(\rho(n_i), C) \leq h(s_0) + C$$

Therefore, a necessary condition for s_i to remain expandable is

$$g(s_i) + h(s_i) - h(s_0) \leq C - \min(\rho(n_i), C).$$

Consider now a *favorable execution* of DPEX in which π is generated with the minimum number of states. The best case would be that $k = m$, i.e., that every state of the plan is sampled immediately after expanding its predecessor, before repeated rectifications accumulate. In this case, $\rho(n_i) = 0$

for every node on the solution path, and the condition would reduce to $g(s_i) + h(s_i) - h(s_0) \leq C$. In general, define

$$C_\pi := \max_{s_i = s_0, \ldots, s_k} \{g(s_i) + h(s_i) - h(s_0) + \rho(n_i)\}, .$$

where n_i is the number of partial expansions of state s_i. Since the explored sequence is always finite and all costs, heuristic values, and rectification counts along it are finite, C_π is finite as well.

Hence, for every run such that $C \geq C_\pi$, all states along π remain expandable during the favorable execution described above. In other words, the clipping bound is sufficiently large to allow the complete generation of π before accumulated rectifications can exclude some prefix node from further expansion.

(2) Next, because π contains a finite number m of decisions, there exists a finite number of iterations M_π sufficient for ID-DPEX to generate the entire path and reach s_g. In fact, $M_\pi = k$. Therefore, any run with $M \geq M_\pi$ allows enough node expansions to potentially generate π completely.

(3) By the assumptions on the sampling function ϕ, every decision in $D(s_i)$ has strictly positive sampling probability whenever s_i is expanded. Hence,

$$\phi(s_i)(\langle a_i, \mu_i \rangle) > 0.$$

Considering the favorable execution, DPEX samples every state of the sequence whenever the corresponding prefix state is expanded, and thus:

$$p_\pi := \prod_{i=0}^{k-1} \phi(s_i)(\langle a_i, \mu_i \rangle) > 0.$$

This quantity is strictly positive because it is a finite product of strictly positive probabilities.

(4) Let $p_\pi(C, M)$ denote the probability that a run of DPEX with clipping bound C and iteration budget M generates the solution plan π. The favorable execution described above is one particular realization leading to success, therefore

$$p_\pi(C, M) \geq p_\pi > 0$$

for every $C \geq C_\pi$ and $M \geq M_\pi$, because $p_\pi(C, M)$ is obtained by summing the probabilities of all execution sequences that generate π, including the favorable execution.

(5) ID-DPEX performs infinitely many independent runs while monotonically increasing the clipping bound C. Consequently, there exists some iteration after which every run satisfies $C \geq C_\pi$. Assuming $M \geq M_\pi$, every such run succeeds with probability at least $p_\pi > 0$.

Since runs are independent, the probability that none of the first k runs satisfying $C \geq C_\pi$ succeeds is at most

$$(1 - p_\pi)^k.$$

Because $0 < p_\pi \leq 1$,we obtain

$$\lim_{k \to \infty} (1 - p_\pi)^k = 0.$$

Equivalently, the probability that at least one run succeeds converges to 1 as the number of runs tends to infinity. Since the argument holds for any solvable problem and any solution plan π, ID-DPEX is probabilistically complete. □

Optimality Guarantees of ID-DPEX

Now that rectification functions are bounded and we have a probabilistically complete algorithm that allows us to use them, some interesting properties emerge when considering admissible heuristic functions. Recall that a heuristic function h is admissible if it never overestimates the optimal cost h^* to the path, i.e., $h(s) \leq h^*(s)$. If we consider bounded additive rectification functions over an admissible heuristic, then the rectification function has an interesting property:

Proposition 3. *If h is an admissible heuristic and r_h is a bounded additive rectification function over h with bound C, then $r_h(n, s) \leq h^*(s) + C$ for every state s and $n \in \mathbb{Z}_0^+$, where h^* represents the optimal cost.*

Proof. Since h is admissible, for every state s, $h(s) \leq h^*(s)$. And since r_h is bounded additive, $r_h(n, s) = h(s) + \rho(n)$, and then $r_h(n, s) \leq h^*(s) + C$ since r_h is bounded by C. □

With admissible functions, bounded additive rectification functions satisfy that they never overestimate the optimal cost by more than C. This also causes that ID-DPEX always returns a solution that never overestimates the optimal cost by more than C.

Theorem 2 (ID-DPEX optimality guarantees)**.** *Under the assumptions of Proposition 1, let h be an admissible heuristic. Given that a solution plan exists, if ID-DPEX$_{\phi, r_h}$ returns a solution during a run using the clipped rectification r_h^C, then the cost $g(s_g)$ of the returned solution s_g satisfies*

$$g(s_g) \leq h^*(s_0) + C$$

where $h^*(s_0)$ denotes the optimal cost from the initial state s_0 to a goal state.

Proof. Consider the particular run of ID-DPEX in which a goal state s_g is returned; denote by C the rectification bound used in that run (so the algorithm employed r_h^C). We reason within this run, which is exactly an execution of DPEX with NEC $f(s) = g(s) + r_h^C(s)$.

Since DPEX returned a solution, its cost must be bounded by the f-value of the root node, and thus $g(s_g) \leq f(s_0) = g(s_0) + r_h^C(n, s_0) = r_h^C(n, s_0)$ for some n. But since r_h^C is bounded additive and h is admissible, then $r_h^C(n, s_0) \leq h^*(s_0) + C$ and thus $g(s_g) \leq h^*(s_0) + C$. □

Overall, the theorem guarantees that, for any given run of ID-DPEX, one can inspect the corresponding clipping constant C and directly infer the quality threshold of the solution that the algorithm may return at that stage. If one wants to give more effort to a certain quality cost of solution, one must increase the number of iterations of that stage M.

On Admissible Heuristics for Numeric Planning with Control Variables

In order to formulate an admissible heuristic for numeric planning with control variables, we leverage a recent compilation-based approach that enables the use of standard informed heuristics despite the presence of infinite decision

spaces. In numeric planning with control variables, classical heuristics based on subgoaling relaxations (Scala et al. 2020a, 2016) cannot be directly applied. These heuristics rely on reasoning over finite sets of possible achievers for each condition, whereas with control variables even a single condition may admit infinitely many achievers, since each valuation of the control variables induces a distinct decision.

A recent paper (Aso-Mollar et al. 2025c) addresses this limitation by proposing the signature compilation (Σ), which identifies a subset of numeric planning problems with control variables, termed controllable-simple numeric planning problems, and transforms control-dependent expressions into simple numeric planning tasks. A simple numeric planning task is a numeric planning problem with simple (linear) conditions and additive effects. The compilation abstracts control-dependent expressions into constant effects and relaxed preconditions, yielding a finite over-approximation of the original problem. In particular, control-dependent effects are replaced by constant effects obtained by taking either the lower or the upper bound of the control expression, depending on the condition affected by the effect. As a consequence, action costs $\lambda(a, \mu)$ are similarly abstracted into one of their extreme values, depending on the optimization metric, so that $\lambda(a_\Sigma) \leq \lambda(a, \mu)$, being a_Σ one of the compiled versions of a.

The compilation is safe under the subgoaling relaxation, in the sense that any solution to the original problem implies the existence of a solution in the compiled version. This property allows existing numeric heuristics such as h^{add} and h^{max} to be applied directly, via the compilation, to problems with infinite decision spaces. For completeness, h^{max} is recursively defined as follows. Let ψ be a constraint and $\psi^{r(a,m)}$ its m-times regressor through action a (Scala et al. 2020a). Intuitively, the regressor captures the weakest constraint that must hold before applying a (possibly multiple times) in order to ensure that ψ holds afterwards.

$$\hat{h}^{max}(s, \psi) = \begin{cases} 0 & \text{if } s \models \psi \\ \min\limits_{a \in ach(\psi)} \left(\lambda(a) + \hat{h}^{max}(s, \text{Pre}_{\mathbb{B}}(a)) \right) & \psi \in \text{PCs} \\ \min\limits_{\substack{a \in A, \\ s \models \psi^{r(a,\hat{m})}, \\ \forall \hat{m} \in \mathbb{Q}^+}} \left(\hat{m}\, \lambda(a) + \hat{h}^{max}(s, \text{Pre}_{\mathbb{Q}}(a)) \right) & \psi \in \text{SCs} \\ \max\limits_{\substack{c \subset \psi \\ |c|=1}} \hat{h}^{max}(s, c) & |\psi| > 1 \end{cases}$$

Where $\lambda(a)$ denotes the cost of action a, PC is the set of propositional conditions ($ach(\psi)$ denotes the set of actions that achieve ψ) and SC is the set of simple numeric conditions. $\hat{m}$ represents the continuous relaxation of the number of repetitions. Interesting properties on admissibility can be directly derived from the nature of the signature compilation and the h^{max} definition.

Theorem 3. *Let $\mathcal{P}$ be a controllable-simple numeric planning problem and let $\mathcal{P}_\Sigma$ be its signature compilation. For any state s, the heuristic $\hat{h}^{max}$ computed over $\mathcal{P}_\Sigma$, h^{max}_Σ, is admissible for $\mathcal{P}$, that is, $h^{max}_\Sigma(s, \psi) \leq h^*(s, \psi)$, where $h^*(s, \psi)$ denotes the cost of an optimal plan from s to satisfy condition ψ.*

Proof. Assume an arbitrary condition ψ. Let s be an arbitrary state and let π^* be an optimal plan from s that fulfills ψ, with optimal cost $h^*(s, \psi)$. We proceed by induction on the structure of goal conditions.

Base case. If $s \models \psi$, then by definition $h^{max}_\Sigma(s, \psi) = 0 \leq h^*(s, \psi) = 0$. The optimal plan is the empty plan.

Inductive hypothesis. The inductive hypothesis states that for any atomic condition ϕ such that $h^*(s, \phi) < h^*(s, \psi)$, the inequality $h^{max}_\Sigma(s, \phi) \leq h^*(s, \phi)$ holds. We will apply this hypothesis both for propositional and numeric conditions. Concretely:

- If ψ is a propositional condition (PC), let $\langle a, \mu \rangle$ be the decision that fulfills ψ in the optimal plan. Since control variables do not affect propositional achievement, any of the compiled actions of a, namely a_Σ, also fulfills ψ in the relaxed problem. Since $\langle a, \mu \rangle$ is the decision that fulfills ψ optimally, this means that it is the best achiever for that condition in the original problem. But since a_Σ also fulfills ψ in the compiled problem and its cost is, by definition of the compilation, $\lambda(a_\Sigma) \leq \lambda(a, \mu)$, then we know that $\lambda(a_\Sigma) + h^{max}_\Sigma(s, \text{Pre}_{\mathbb{B}}(a_\Sigma)) \leq \lambda(a, \mu) + h^*(s, \text{Pre}_{\mathbb{B}}(a_\Sigma))$, using the inductive hypothesis. Since the compilation relaxes every condition, achieving $\text{Pre}_{\mathbb{B}}(a_\Sigma)$ can only be "easier" than achieving $\text{Pre}_{\mathbb{B}}(a)$. For this reason, $\lambda(a_\Sigma) + h^{max}_\Sigma(s, \text{Pre}_{\mathbb{B}}(a_\Sigma)) \leq \lambda(a, \mu) + h^*(s, \text{Pre}_{\mathbb{B}}(a)) = h^*(\psi)$. Since a_Σ is an achiever of ψ, the minimum of the second equation of h^{max}_Σ (see $\hat{h}^{max}$ definition) is, at worst, that value, and at best even lower if there is a compiled achiever with lower overall cost.
- If ψ is a simple numeric condition (SC), let $\langle a, \mu \rangle$ be the decision that fulfills ψ. We can distinguish two cases:
  1. The decision $\langle a, \mu \rangle$ fulfills ψ with a control valuation μ that assigns an extreme value of the control variables' domain. In this case, the compiled action a_Σ also fulfills ψ in the relaxed problem and we can apply the same rationale as in the propositional condition case. It already works for $\hat{m} = 1$, i.e., there is no need for repetition on that compiled numeric achiever.
  2. The decision $\langle a, \mu \rangle$ fulfills ψ with μ being an intermediate value of the control variables' domain. This is analogous to the previous case but now we can adjust $\hat{m}$ to fulfill the precondition.

  Both of these cases state that, at least, there exists an element in the minimum of the third equation of h^{max}_Σ that already lowers the cost of the compiled plan, but there could exist another different plan in the compilation that (1) reduces the overall cost and (2) fulfills ψ maximizing the net value of achieving that condition.
- If ψ is a conjunction, since each atomic condition ϕ satisfies $h^{max}_\Sigma(s, \phi) \leq h^*(s, \phi)$, it follows that $h^{max}_\Sigma(s, \psi) = \max\limits_{\substack{\phi \subset \psi \\ |\phi|=1}} h^{max}_\Sigma(s, \phi) \leq h^*(s, \psi)$. □

As a direct consequence of the previous theorem, ID-DPEX retains its optimality guarantees when using h^{max}_{Σ}.

Experiments

In this section, we analyze the capability of ID-DPEX to generate near-optimal plans for numeric planning problems with control variables. To this end, we conduct a two-fold experimentation: (1) an ablation study of the performance of ID-DPEX under different heuristics, including non-admissible ones (2) an evaluation against state of the art methods, namely NextFLAP and DPEX, in domains with unitary cost definitions; and (3) an assessment against theoretical optimal bounds for domains with controllable action costs.

Setup Experiments were conducted with a 30-minute timeout and an 8 GB memory limit, using uniform sampling, logarithmic rectification, and $K = 10$ samples per ID-DPEX and DPEX run. Node Evaluation Criterion is defined to consider both heuristic information and cost, naturally: $f = g + r_h$. For ID-DPEX, we set the initial cost bound to $C_0 = 1$ and increased it additively by 1 at each iteration, $\Delta = 1$. The maximum number of iterations was set to $M = 10^6$. ID-DPEX and DPEX are both implemented in the ENHSP planner[1].

Domains We used the same benchmark domains as in (Aso-Mollar et al. 2025a): BLOCKGROUPING, CASHPOINT, COUNTERS, DRONE, PROCUREMENT, SAILING, and TERRARIA. Problem instances are provided in the supplementary material. For each of these seven domains, we considered both continuous (C) and discrete (D) variants, corresponding to control variables with and without decimal values, respectively. Although our approach is not limited in terms of precision, in the continuous variants we deliberately restricted control variables to three decimal places (0.001), matching NextFLAP's representation for fairness. All domains assume unitary action costs. Additionally, we designed two domains with controllable action costs: CASHPOINT-CAC and COUNTERS-CAC. All details about these additional domains can be consulted in the supplementary material.

Analysis of the impact of heuristic choice

In this set of experiments, we consider ID-DPEX performance under admissible heuristics: h^0 and h^{max}_{Σ}. We also evaluate ID-DPEX using non-admissible heuristics that provide more informative cost estimates, namely the goal-counting heuristic h^{gc} and the heuristic h^{mrp}_{Σ} from (Aso-Mollar et al. 2025c), obtained via the signature compilation of h^{mrp} (Scala et al. 2020b).

By contrasting admissible and non-admissible heuristics, we aim to assess how ID-DPEX behaves when admissibility is sacrificed in favor of more informative guidance and whether this trade-off leads to improved performance. Table 1 reports the results. Non-admissible heuristics consistently yield higher coverage. We also observe an improvement of h^{max}_{Σ} over the blind heuristic.

[1]https://github.com/hstairs/jpddlplus/tree/enhsp25-dpex

Domains (20)	h^0	h^{max}_{Σ}	h^{gc}	h^{mrp}_{Σ}
BLOCKGROUPING-C	0	0	**20**	13
BLOCKGROUPING-D	0	0	**20**	13
CASHPOINT-C	0	0	0	**20**
CASHPOINT-D	0	0	0	**20**
COUNTERS-C	3	3	**11**	**11**
COUNTERS-D	0	0	**10**	**10**
DRONE-C	0	0	5	10
DRONE-D	**20**	**20**	**20**	**20**
PROCUREMENT-C	4	8	6	**11**
PROCUREMENT-D	4	7	6	**10**
SAILING-C	1	2	4	**14**
SAILING-D	0	1	1	**16**
TERRARIA-C	4	6	5	**20**
TERRARIA-D	2	9	8	**20**
CASHPOINT-CAC	10	**20**	11	**20**
COUNTERS-CAC	15	14	12	**20**
Total (320)	63	90	139	**249**

Table 1: Coverage of ID-DPEX per heuristic. In bold, global best; in grey background, setting best (admissible and inadmissible).

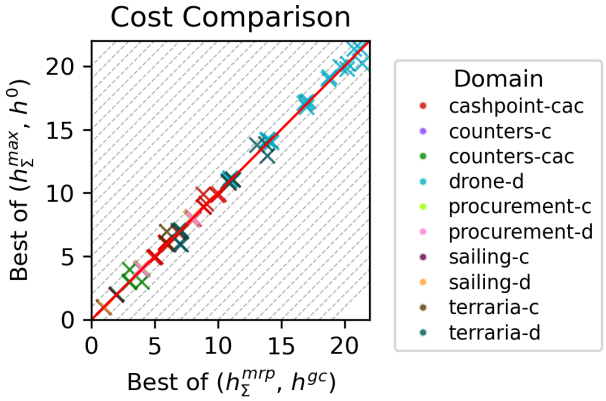

Figure 1: Cost comparison between admissible and inadmissible heuristics.

Although an increase in coverage is observed when using non-admissible heuristics, it is important to assess whether this improvement comes at the expense of solution cost quality. To this end, Figure 1 presents a comparison of the best solutions obtained using admissible versus non-admissible heuristics. It is worth noting that the theoretical bounds imposed by C correspond to upper bounds on the solution cost; consequently, depending on the sampling strategy and on how informative the heuristic is, the search may reach solutions with lower cost of than the bound.

Notably, the solution costs when using admissible and inadmissible heuristics in the search are nearly identical, differing by at most a single unit. This behavior can be explained by the cost bound C, which effectively limits the depth of the search. As a result, even when using non-admissible heuristics solutions remain nearly optimal.

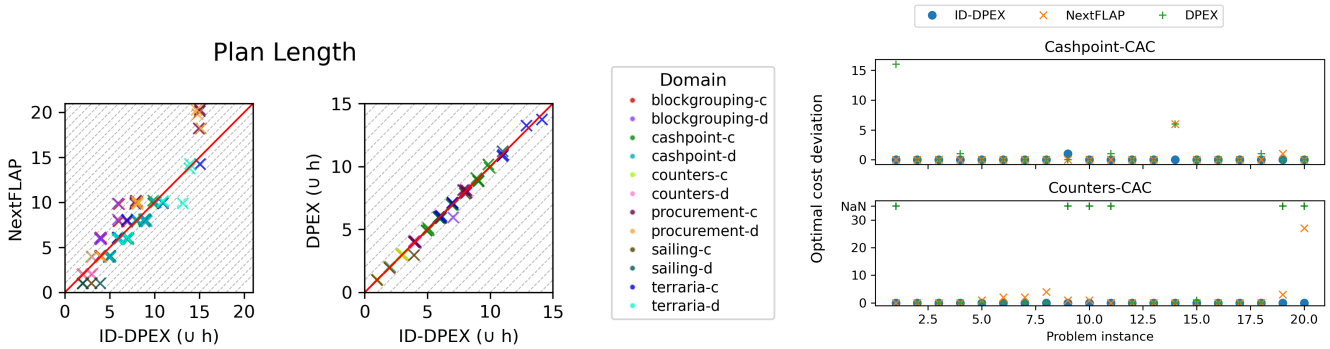

Figure 2: Cost of ID-DPEX versus NextFLAP and DPEX in unitary (left) and controllable (right) action costs.

Evaluation against the state of the art

Once the inner workings of the approach have been analyzed, we proceed to compare the system against the state of the art. The two existing baselines are NextFLAP and DPEX; however, NextFLAP does not support controllable action costs. Therefore, in the following we restrict our evaluation to domains with unitary cost actions. For both ID-DPEX and DPEX, we select the run that produces the best plan length, using the four heuristics defined in the previous section.

	BLOCK.-C	BLOCK.-D	CASHP.-C	CASHP.-D	COUNT.-C	COUNT.-D	DRONE-C	DRONE-D	PROCU.-C	PROCU.-D	SAILI.-C	SAILI.-D	TERRA.-C	TERRA.-D	**Total**
NextFLAP	20	19	20	18	20	12	0	0	18	13	20	18	8	10	196
DPEX	20	20	0	20	10	20	5	20	8	8	4	14	7	9	165
ID-DPEX	20	20	20	20	11	20	10	20	11	10	14	20	20	20	223

Table 2: Coverage of NextFLAP, DPEX and ID-DPEX.

Table 2 reports the number of problems solved by each system across proposed domains. Overall, ID-DPEX outperforms both NextFLAP and DPEX. When examining individual domains, we observe a decrease in performance in PROCUREMENT, as well as in some (C) domains. This behavior is expected, as the search effort increases with the cardinality of the control variable domains, making these instances inherently more computationally demanding. In contrast, NextFLAP is independent of the size of the control domains and is therefore unaffected by this factor. The comparison between DPEX and ID-DPEX further indicates that bounding the rectification with the iterative scheme favors an increase in coverage.

In Figure 2, we compare the plan lengths obtained by ID-DPEX against those from NextFLAP and DPEX. Plans generated by ID-DPEX are either strictly shorter or differ from those of NextFLAP by at most three actions (mostly one), which aligns with the theoretical suboptimality guarantees. NextFLAP is a best-effort planner with no guarantees, which sometimes leads to significantly longer plans as observed in the figure. Notably, ID-DPEX and DPEX runs achieve identical plan lengths along most instances.

For the two domains with controllable action costs (CASHPOINT-CAC and COUNTERS-CAC), we additionally compared the solution costs obtained by ID-DPEX against analytically computable optimal costs. In CASHPOINT-CAC, the optimal cost corresponds to the minimum amount of withdrawn money together with the required navigation and purchase actions, whereas in COUNTERS-CAC the optimum can be formulated as a constrained optimization problem over the increment and decrement control values. Figure 2 reports the deviation from the optimal cost for both ID-DPEX and NextFLAP.

ID-DPEX reaches the optimum in 39 out of the 40 analyzed instances, with a deviation of one unit in the remaining case. In contrast, although NextFLAP does not natively support controllable action costs, we nevertheless evaluated the costs induced by the plans it generates. Under this setting, only 30 out of 40 cases achieve optimal cost, with some instances deviating substantially from the optimum. This behavior is expected, since NextFLAP optimizes plan length before concretizing control values, whereas ID-DPEX directly incorporates control valuations into the search cost function.

Conclusions and Future Work

In this work, we propose an approach for computing near-optimal plans in planning with control variables. We extend the notion of action cost to account for costs that depend on control valuations, and we formally define the resulting problem setting. Building on this formulation, we introduce a novel algorithm, ID-DPEX, together with admissible heuristics tailored to this framework. We show that ID-DPEX is probabilistically complete and that it provides explicit optimality guarantees when combined with admissible heuristics.

We empirically evaluate the algorithm on a diverse set of problems using both admissible and non-admissible heuristics. The results demonstrate not only the feasibility of the proposed approach, but also its advantages over existing methods in the literature. As future work, we plan to investigate more informative heuristics to further improve search efficiency, as well as additional algorithmic optimizations.

Acknowledgements

This work has been partially supported 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 and by the AI4WATER AEI PCI2025-163189 project.

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.

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.

Aso-Mollar, Á.; Aineto, D.; Scala, E.; and Onaindia, E. 2025c. Subgoaling Relaxation-based Heuristics for Numeric Planning with Infinite Actions. *Proceedings of the 36th International Conference on Automated Planning and Scheduling, ICAPS-26 (arXiv:2512.22367)*.

Felner, A.; Goldenberg, M.; Sharon, G.; Stern, R.; Beja, T.; Sturtevant, N.; Schaeffer, J.; and Holte, R. 2012. Partial-Expansion A* with Selective Node Generation. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI-12*, 471–477.

Goldenberg, M.; Felner, A.; Stern, R.; Sharon, G.; Sturtevant, N.; Holte, R. C.; and Schaeffer, J. 2014. Enhanced Partial Expansion A*. *Journal of Artificial Intelligence Research*, 50(1): 141–187.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Korf, R. E. 1985. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, 27(1): 97–109.

Pohl, I. 1970. Heuristic Search Viewed as Path Finding in a Graph. *Artificial Intelligence*, 1(3): 193–204.

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 *Proceedings of the Twenty-Second European Conference on Artificial Intelligence, ECAI-16*, volume 285, 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, ECAI-16*, 655–663.

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. In *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, ICAPS-20*, 226–234.