Cartesian Abstraction Refinement for Simple Numeric Planning

Tanja Schindler, David Speck, Malte Helmert

University of Basel
{tanja.schindler, davidjakob.speck, malte.helmert}@unibas.ch

Abstract

Cartesian abstractions are a successful approach for obtaining admissible heuristics in optimal classical planning. In this paper, we generalize the counterexample-guided abstraction refinement (CEGAR) algorithm to compute Cartesian abstractions for simple numeric planning. Specifically, our CEGAR algorithm operates on a special form of Cartesian states consisting of a single interval per numeric variable. We prove that, in the absence of zero-cost actions, this algorithm is semi-complete and optimal. Our experimental evaluation shows that our approach performs similarly to other abstraction heuristics and provides better heuristic estimates in multiple domains.

Introduction

Heuristic search (Pearl 1984) is a prominent approach to numeric planning. Several heuristic approaches from classical planning have been generalized to support numeric variables. A large body of work has investigated interval-based relaxations for computing heuristics for simple to rich numeric planning formalisms (e. g., Hoffmann 2003; Scala et al. 2016; Aldinger and Nebel 2017). Furthermore, landmark heuristics (e. g., Helmert and Domshlak 2009) have been generalized to numeric planning and shown to have strong empirical performance (Scala et al. 2017; Kuroiwa, Shleyfman, and Beck 2022). Recently, Gnad et al. (2025) presented numeric versions of pattern database heuristics, which fall into the class of abstraction heuristics.

In this paper, we consider a more general type of abstraction for simple numeric planning: Cartesian abstractions, which due to their expressiveness can yield more informed heuristics with the same abstract state space size. To do so, we generalize the Counterexample-Guided Abstraction Refinement approach (Seipp and Helmert 2018) from classical to simple numeric planning. Specifically, we present a semi-complete and optimal algorithm that operates on Cartesian states, each consisting of a single interval per numeric variable representing its possible values.

Background

As Gnad et al. (2025), we consider a simple fragment of numeric planning: integer-restricted planning tasks (IRTs).

An *IRT* is a tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ with a set $\mathcal{V} = \mathcal{V}_\mathrm{n} \cup \mathcal{V}_\mathrm{p}$ of numeric variables $\mathcal{V}_\mathrm{n}$ and finite-domain variables $\mathcal{V}_\mathrm{p}$, a set of operators $\mathcal{O}$, the initial state $\mathcal{I}$, and a goal $\mathcal{G}$.

A *state* s of the planning task Π is a complete assignment that maps each variable $v \in \mathcal{V}_\mathrm{p}$ to an element of its finite domain $\mathit{dom}(v)$, and each variable $v \in \mathcal{V}_\mathrm{n}$ to an integer value. We denote the value of a variable v in state s by $s[v]$. The *initial state* $\mathcal{I}$ of Π is a state. The set of states of a planning task Π is denoted by $S(\Pi)$.

A *propositional condition* is an atom $(v = d)$ with $v \in \mathcal{V}_\mathrm{p}$ and $d \in \mathit{dom}(v)$. A *numeric condition* is an atom $(v \bowtie c)$ where $v \in \mathcal{V}_\mathrm{n}$ and $c \in \mathbb{Z}$, and $\bowtie \in \{\geq, =, \leq\}$. Numeric conditions $(v > a)$ and $(v < a)$ can be expressed as $v \geq a+1$ and $v \leq a-1$. A state s *satisfies* a condition c, denoted by $s \models c$, if c evaluates to true under the variable assignment s, and it satisfies a set of conditions $\mathcal{C}$, denoted by $s \models \mathcal{C}$, if $s \models c$ for all $c \in \mathcal{C}$.

The *goal* $\mathcal{G}$ is a set of propositional and numeric conditions. We assume that it is consistent, i. e., there are no two contradictory conditions on a variable.

An *operator* $o \in \mathcal{O}$ is a tuple $o = \langle \mathit{pre}(o), \mathit{eff}(o), \mathit{cost}(o) \rangle$. The *precondition* $\mathit{pre}(o)$ of operator o is a set of propositional and numeric conditions. We assume that it is consistent. The *effect* $\mathit{eff}(o)$ of operator o is a set of *propositional effects* of the form $(v := d)$ with $v \in \mathcal{V}_\mathrm{p}$ and $d \in \mathit{dom}(v)$, and *numeric effects* of the form $(v := v + c)$ where $v \in \mathcal{V}_\mathrm{n}$ and $c \in \mathbb{Z} \setminus \{0\}$. In contrast to Gnad et al. (2025), we also allow effects $(v := c)$ for $v \in \mathcal{V}_\mathrm{n}$ and $c \in \mathbb{Z}$. We assume that an operator contains at most one effect per variable. The *cost* of an operator is a positive integer $\mathit{cost}(o) \in \mathbb{N}^+$.

An operator o is *applicable* in state s if $s \models \mathit{pre}(o)$. The successor of s under o is the state $s[\![o]\!]$ where $s[\![o]\!][v] = d$ for propositional effects $(v := d) \in \mathit{eff}(o)$, $s[\![o]\!][v] = s[v] + c$ or $s[\![o]\!][v] = c$ for numeric effects $(v := v + c) \in \mathit{eff}(o)$ or $(v := c) \in \mathit{eff}(o)$, and $s[\![o]\!][v] = s[v]$ otherwise. The regression of a set of states S with respect to an operator o is $\mathit{regr}(S, o) = \{s \in S(\Pi) \mid s \models \mathit{pre}(o), s[\![o]\!] \in S\}$.

A *plan* for a state s is a sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ such that, starting in s, the operators are sequentially applicable, and $s[\![\pi]\!] = s[\![o_1]\!] \cdots [\![o_n]\!]$ is a goal state. A plan for the initial state $\mathcal{I}$ is a plan for Π. The cost $\mathit{cost}(\pi)$ of a plan π is the sum of the costs of its operators, and π is optimal if $\mathit{cost}(\pi) \leq \mathit{cost}(\pi')$ for all other plans π'.

The semantics of a planning task can be described in

terms of transition systems. A *transition system* is a tuple $\mathcal{T} = \langle S, L, c, T, s_\mathrm{I}, S_\mathrm{G} \rangle$ with a set of states S, a finite set of labels L, a cost function $c : L \mapsto \mathbb{N}_0$, a transition relation $T \subseteq S \times L \times S$, an initial state s_I, and a set $S_\mathrm{G} \subseteq S$ of goal states. We write $s \xrightarrow{\ell} s'$ if $\langle s, \ell, s' \rangle \in T$. A *trace* in $\mathcal{T}$ is a sequence of transitions $s_0 \xrightarrow{\ell_1} s_1 \cdots s_{n-1} \xrightarrow{\ell_n} s_n$. A *goal trace* for $\mathcal{T}$ is a trace with $s_0 = s_\mathrm{I}$ and $s_n \in S_\mathrm{G}$.

The transition system *induced by* Π is $\mathcal{T}(\Pi) = \langle S, L, c, T, s_\mathrm{I}, S_\mathrm{G} \rangle$ with states $S = S(\Pi)$, labels $L = \mathcal{O}$, cost function defined by $c(o) = \mathit{cost}(o)$ for all $o \in \mathcal{O}$, transition relation $T = \{\langle s, o, s' \rangle \mid s \in S, o \in \mathcal{O}, s' = s[\![o]\!]\}$, initial state $s_\mathrm{I} = \mathcal{I}$, and goal states $S_\mathrm{G} = \{s \in S \mid s \models \mathcal{G}\}$. The label sequence of a goal trace for $\mathcal{T}(\Pi)$ is a plan for Π.

Given a transition system $\mathcal{T} = \langle S, L, c, T, s_\mathrm{I}, S_\mathrm{G} \rangle$ and an *abstraction* function $\alpha : S \to S^\alpha$, the *induced abstract transition system* is $\mathcal{T}^\alpha = \langle S^\alpha, L, c, T^\alpha, \alpha(s_\mathrm{I}), S^\alpha_\mathrm{G} \rangle$ where $T^\alpha = \{\langle \alpha(s), o, \alpha(s') \rangle \mid \langle s, o, s' \rangle \in T\}$ and $S^\alpha_\mathrm{G} = \{\alpha(s) \mid s \in S_\mathrm{G}\}$. This definition ensures that all plans in $\mathcal{T}$ are also plans in $\mathcal{T}^\alpha$. We use the notation $[s]_\alpha$ to denote the abstract state $\alpha(s)$, and we write $[s]$ when α is clear from the context.

CEGAR Algorithm

A popular approach to compute admissible heuristics for planning is based on abstractions. Cartesian abstractions have been successfully applied in optimal classical planning (Seipp and Helmert 2018) to SAS$^+$ tasks (Bäckström and Nebel 1995) and factored tasks (Büchner et al. 2024), along with many improvements (e. g., Seipp, von Allmen, and Helmert 2020; Pozo, Torralba, and Linares López 2024).

A *Cartesian set* is a Cartesian product $A = A_1 \times \ldots \times A_n$ of sets. A *Cartesian abstraction* of a transition system maps each concrete state to a Cartesian set of states. In the following, we outline the counterexample-guided abstraction refinement (CEGAR) algorithm (Seipp and Helmert 2018) that computes Cartesian abstractions for SAS$^+$ planning tasks.

The initial abstraction of a transition system over variables $\mathcal{V} = \langle v_1, \ldots, v_n \rangle$ consists of the single abstract state $\mathit{dom}(v_1) \times \cdots \times \mathit{dom}(v_n)$. In each iteration, the algorithm tries to find an optimal abstract goal trace. If none can be found, the original task is unsolvable. Otherwise, the algorithm tries to construct a concrete trace from the abstract trace, and if it succeeds, returns the corresponding plan for Π. If it fails, the algorithm identifies a flaw and uses it to refine the abstraction by splitting an abstract state such that the discovered flaw can no longer occur, and continues with the loop until some termination condition is met. At any point, the abstract transition system can be used to compute a consistent and admissible heuristic for the original task.

The following three types of flaws can occur in SAS$^+$ tasks when trying to convert an abstract goal trace $a_0 \xrightarrow{o_1} a_1 \xrightarrow{o_2} \ldots \xrightarrow{o_n} a_n$ into a concrete goal trace: (1) a *precondition flaw*, i. e., operator o_{i+1} is not applicable in the concrete state $s_i = \mathcal{I}[\![\langle o_1, \ldots, o_i \rangle]\!]$, (2) a *goal flaw*, i. e., the concrete state $s_n = \mathcal{I}[\![\langle o_1, \ldots, o_n \rangle]\!]$ is not a goal state, and (3) a *deviation flaw*, i. e., the abstract state corresponding to the concrete state $s_i = \mathcal{I}[\![\langle o_1, \ldots, o_i \rangle]\!]$ is different from the abstract state a_i reached in the abstract trace. For abstraction refinement, the algorithm uses the set of states satisfying an operator precondition, the regression of a set of states, or the goal states to split the abstract state in which the flaw occurs. In SAS$^+$ and factored tasks, all state sets relevant for the CEGAR algorithm can be represented as Cartesian sets and be efficiently computed and stored.

Intervals

We will use intervals to abstract numeric variables. An *integer interval* $I \subseteq \mathbb{Z}$ is of the form $[a, b]$, $(-\infty, a]$ or $[a, +\infty)$. We use $\mathit{lb}(I), \mathit{ub}(I) \in \mathbb{Z} \cup \{-\infty, +\infty\}$ to denote the lower and upper bounds of interval I. For an interval I and a constant $k \in \mathbb{Z}$, we write $I + k$ for the interval obtained by adding k to both $\mathit{lb}(I)$ and $\mathit{ub}(I)$ (with $\pm\infty + k = \pm\infty$). An *$n$-dimensional interval* is a product of intervals $\boldsymbol{I} = I_1 \times \cdots \times I_n$. For two n-dimensional intervals $\boldsymbol{I} = I_1 \times \cdots \times I_n$ and $\boldsymbol{I'} = I'_1 \times \cdots \times I'_n$, we write $\boldsymbol{I} \cap \boldsymbol{I'} = (I_1 \cap I'_1) \times \cdots \times (I_n \cap I'_n)$. Note that the intersection is again an n-dimension interval.

Cartesian Abstraction for Numeric Planning

We show how we can extend the Cartesian CEGAR algorithm to integer-restricted tasks. The core idea is to use integer intervals as abstract values for the numeric variables while treating finite-domain variables as Seipp and Helmert (2018). The following example illustrates this idea.

Example 1. *Consider the IRT $\Pi = \langle \mathcal{V}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ with variables $\mathcal{V} = \mathcal{V}_\mathrm{n} = \{x\}$, initial state $\mathcal{I} = \{x \mapsto 0\}$, goal $\mathcal{G} = (x \geq 6)$, and operators $\mathcal{O} = \{o_1, o_2\}$ where $o_1 = \langle x \geq 4, x := x + 1 \rangle$ and $o_2 = \langle x = 0, x := x + 4 \rangle$.*

We start with the trivial abstraction $S^{\alpha_0} = \{(-\infty, +\infty)\}$. An abstract goal trace for S^{α_0} consists of just $a_0 = (-\infty, +\infty)$. In the concrete transition system, $\pi = \langle\rangle$ is not a plan as the goal is not satisfied in the initial state $s_0 = \mathcal{I}$ (a goal flaw occurs). We refine the abstraction by splitting the abstract state $[s_0]_{\alpha_0} = (-\infty, +\infty)$ on the goal condition, and obtain $S^{\alpha_1} = \{(-\infty, 5], [6, +\infty)\}$. An abstract goal trace for $\mathcal{T}^{\alpha_1}$ is $(-\infty, 5] \xrightarrow{o_1} [6, +\infty)$. In the concrete transition system, we start with $s_0 = \mathcal{I}$, but o_1 is not applicable in s_0 (a precondition flaw occurs). We refine the abstraction again, by splitting $[s_0]_{\alpha_1} = (-\infty, 5]$ on the precondition $\mathit{pre}(o_1)$, and obtain $S^{\alpha_2} = \{(-\infty, 3], [4, 5], [6, +\infty)\}$. An abstract goal trace for $\mathcal{T}^{\alpha_2}$ is $(-\infty, 3] \xrightarrow{o_2} [4, 5] \xrightarrow{o_1} [6, +\infty)$. In the concrete transition system, we start again with $s_0 = \mathcal{I}$, apply o_2 resulting in $s_1 = \{x \mapsto 4\}$, and then apply o_1, resulting in $s_2 = \{x \mapsto 5\}$. The abstract trace ends in the abstract state $a_2 = [6, +\infty)$, but the abstract state corresponding to s_2 is $[s_2]_{\alpha_2} = [4, 5]$ (a deviation flaw occurs). We thus split the abstract state a_1 from which we started, here also $[4, 5]$, into an abstract state from which a_2 can be reached with o_1 and one from which it cannot, resulting in $S^{\alpha_3} = \{(-\infty, 3], [4, 4], [5, 5], [6, +\infty)\}$. An abstract goal trace is $(-\infty, 3] \xrightarrow{o_2} [4, 4] \xrightarrow{o_1} [5, 5] \xrightarrow{o_1} [6, +\infty)$, and $\pi = \langle o_2, o_1, o_1 \rangle$ is also a concrete plan for the problem.

We present the theoretical foundations of our approach. For simplicity, we assume that we are given an IRT over

Algorithm 1: FindFlaw for interval-restricted planning tasks (adapted from Seipp and Helmert, 2018).

```
1  Function FindFlaw(τ : abstract goal trace):
2      s ← I
3      foreach a --o--> b ∈ τ do
4          if o is not applicable in s then
5              return ⟨s, [s] ∩ I(pre(o))⟩
6          if b does not include s⟦o⟧ then
7              return ⟨s, [s] ∩ regr(b, o)⟩
8          s ← s⟦o⟧
9      if s is not a goal state then
10         return ⟨s, [s] ∩ I(G)⟩
11     return "no flaw"
```

n numeric variables and no finite-domain variables. Finite-domain variables can be handled in the same way as in previous work (Seipp and Helmert 2018; Büchner et al. 2024).

We start by arguing that intervals are a suitable abstraction for numeric variables in IRTs, since sets of states satisfying a set of numeric conditions can be represented by n-dimensional intervals.

Lemma 1. *Let $\Pi = \langle \mathcal{V}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ be an IRT with $\mathcal{V} = \mathcal{V}_\mathrm{n} = \{v_1, \dots v_n\}$ and let $\mathcal{C}$ be a consistent set of numeric conditions. Then the set of states $S_\mathcal{C}$ that satisfy $\mathcal{C}$ can be represented by a non-empty n-dimensional integer interval $\boldsymbol{I}(C)$.*

Proof. Let $\mathcal{C}_i \subseteq \mathcal{C}$ denote the set of conditions on variable $v_i \in \mathcal{V}_\mathrm{n}$. The values for v_i that satisfy $\mathcal{C}_i$ are exactly the values in the interval $I_i(\mathcal{C}) = \bigcap_{c \in \mathcal{C}_i} I(c)$ where

$$I(c) = \begin{cases} [a, a] & \text{if } c = (v_i = a) \\ (-\infty, a] & \text{if } c = (v_i \leq a) \\ [a, +\infty) & \text{if } c = (v_i \geq a). \end{cases}$$

As $\mathcal{C}$ is consistent, the intersection $I_i(\mathcal{C})$ is non-empty. If there is no condition on v_i, the above simplifies to $I_i(\mathcal{C}) = \bigcap_{c \in \emptyset} I(c) = (-\infty, +\infty)$. Overall, the set of states $S_\mathcal{C}$ satisfying $\mathcal{C}$ can be represented by the non-empty n-dimensional interval $\boldsymbol{I}(\mathcal{C}) = I_1(\mathcal{C}) \times \cdots \times I_n(\mathcal{C})$. □

In the following, we use the notations $\boldsymbol{I}(\mathcal{C})$ and $I_i(\mathcal{C})$ as in the proof of Lemma 1 to refer to the n-dimensional interval representing the set of states that satisfy a set of conditions $\mathcal{C}$ and to its i-th component.

Next, we show that all relevant state sets used in the CEGAR algorithm can be represented as n-dimensional intervals, and that we can split an n-dimensional interval $\boldsymbol{I}$ into two n-dimensional intervals to separate a state $s \in \boldsymbol{I}$ from an interval $\boldsymbol{I}' \subset \boldsymbol{I}$. These properties ensure that the CEGAR algorithm operates exclusively on n-dimensional intervals.

Theorem 1 (adapted from Seipp and Helmert, 2018)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ be an IRT with $\mathcal{V} = \mathcal{V}_\mathrm{n} = \{v_1, \dots, v_n\}$. Then the following properties hold.*

(P1) *The set of goal states of Π is an n-dimensional interval.*

(P2) *The set of states where an operator $o \in \mathcal{O}$ is applicable is an n-dimensional interval.*

(P3) *The regression $\mathrm{regr}(B, o)$ of an n-dimensional interval B with respect to $o \in \mathcal{O}$ is an n-dimensional interval.*

(P4) *If C and A are n-dimensional intervals with $C \subseteq A$, and $s \in A \setminus C$, then A can be partitioned into two n-dimensional intervals D and E with $s \in D$ and $C \subseteq E$.*

Proof. **(P1), (P2)** follow from Lemma 1 with $\mathcal{C} = \mathcal{G}$ and $\mathcal{C} = \mathit{pre}(o)$.

(P3) The regression of an n-dimensional interval $B = B_1 \times \cdots \times B_n$ with respect to operator $o \in \mathcal{O}$ is $\mathit{regr}(B, o) = A_1 \times \cdots \times A_n$ with

$$A_i = \begin{cases} B_i \cap I_i(\mathit{pre}(o)) & \text{if } v_i \notin \mathit{vars}(\mathit{eff}(o)) \\ I_i(\mathit{pre}(o)) \cap (B_i - k) & \text{if there is an effect } v_i := v_i + k \\ I_i(\mathit{pre}(o)) & \text{if there is an effect } v_i := k \text{ with } k \in B_i \\ \emptyset & \text{if there is an effect } v_i := k \text{ with } k \notin B_i \end{cases}$$

(P4) Let $A = A_1 \times \cdots \times A_n$ and $C = C_1 \times \cdots \times C_n$. Let $v_i \in \mathcal{V}_\mathrm{n}$ be a variable with $s[v_i] \notin C_i$. If $s[v_i] < \mathit{lb}(C_i)$, we define $D_i = A_i \cap (-\infty, \mathit{lb}(C_i) - 1]$ and $E_i = A_i \setminus D_i = A_i \cap [\mathit{lb}(C_i), +\infty)$. Otherwise, $s[v_i] > \mathit{ub}(C_i)$, and we define $D_i = A_i \cap [\mathit{ub}(C_i) + 1, +\infty)$ and $E_i = A_i \setminus D_i = A_i \cap (-\infty, \mathit{ub}(C_i)]$. Then $D = A_1 \times \cdots \times A_{i-1} \times D_i \times A_{i+1} \times \cdots \times A_n$ and $E = A_1 \times \cdots \times A_{i-1} \times E_i \times A_{i+1} \times \cdots \times A_n$. □

The overall CEGAR algorithm follows the description in the background. The FindFlaw function (Alg. 1) converts an abstract goal trace to a concrete trace. The same three flaws as in Seipp and Helmert (2018) can occur here for IRTs. FindFlaw returns a pair $\varphi = \langle s, \boldsymbol{I} \rangle$ consisting of a concrete state s and an n-dimensional interval $\boldsymbol{I} \subseteq [s]$ such that $s \notin \boldsymbol{I}$ but $s \in \boldsymbol{I}$ would be required to continue constructing the concrete trace. The computations of the n-dimensional intervals for the precondition, deviation, and goal flaws in lines 5, 7, and 10 use (P1), (P3) and (P2) from Theorem 1. Once a flaw $\varphi = \langle s, \boldsymbol{I} \rangle$ has been detected, it is used to refine the abstract transition system by splitting the abstract state $[s]$, as described in the proof of (P4) in Theorem 1, and recomputing the incoming and outgoing transitions of the two new states. We now show that the obtained algorithm is semi-complete and optimal.

Theorem 2. *The interval-based CEGAR algorithm for integer-restricted tasks is* semi-complete *and* optimal*, i. e., it returns a plan if one exists, and this plan is optimal.*

Proof. We assume that there exists an optimal plan for Π with cost c^*. The abstract plans considered by the CEGAR algorithm are therefore operator sequences with cost at most c^*. There are only finitely many such operator sequences, and thus it suffices to show that each abstract plan triggers a finite number of refinement steps.

Consider an abstract goal trace with operator sequence $o_1, \dots, o_m$ and assume that FindFlaw detects the first flaw

Domain	h^{blind}	h^{pdbs}	h^{cpdbs}	h^{ipdbs}	h^{lmcut}	$h^{\text{cegar}}_{\text{rnd}}$	$h^{\text{cegar}}_{\text{max}}$	$h^{\text{cegar}}_{\text{min}}$
Counters (20)	3	3	4	4	**5**	**5**	4	4
CountersS (11)	2	2	5	8	**11**	2	2	2
Delivery (20)	2	2	2	2	**3**	2	2	2
Depots (20)	3	5	**7**	**7**	**7**	3	4	3
DepotsS (20)	4	4	4	6	**7**	4	4	4
Drone (20)	3	3	3	3	3	3	**4**	3
FnCounters (8)	6	6	**7**	**7**	**7**	**7**	**7**	**7**
Forestfire (20)	7	7	7	7	**10**	8	8	8
MinecraftP (20)	13	18	17	18	0	**19**	**19**	**19**
MinecraftS (20)	**20**	**20**	**20**	**20**	5	**20**	**20**	**20**
Mprime (20)	5	9	8	9	**15**	**15**	14	**15**
PetriNet (20)	2	2	6	7	**8**	6	6	7
Rover-unit (20)	4	4	4	5	**7**	4	4	5
Sugar (20)	2	2	2	2	**9**	8	**9**	**9**
Zeno (23)	5	7	9	10	**12**	8	8	8
Zeno-IPC (20)	6	6	6	6	**8**	**8**	6	**8**
Others (163)	**93**	**93**	**93**	**93**	**93**	**93**	**93**	**93**
Sum (465)	160	173	184	194	190	195	194	**197**

Table 1: Number of solved tasks of A* with different heuristics for simple numeric planning.

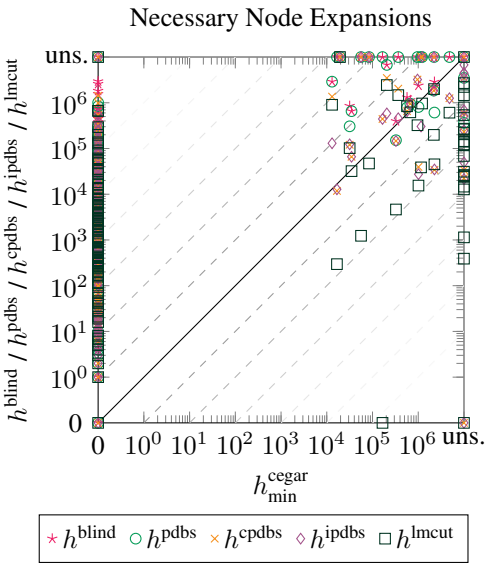

Figure 1: Number of necessary node expansions of A* with different heuristics for simple numeric planning.

at iteration k for the concrete state s. Note that s depends only on $o_1, \ldots, o_k$ and not on the current abstraction. Thus, it suffices to argue that a fixed state s can only trigger a finite number of refinements. We observe that when s triggers an update, its abstract state is refined (and thus changes).

For a given state s, consider the sequence of distinct abstract states (n-dimensional intervals) $[s]_1, [s]_2, \ldots$ during the run of the algorithm. Each $[s]_{j+1}$ is a strict subset of $[s]_j$, which means that there exists a dimension for which the corresponding interval in $[s]_{j+1}$ is a strict subset of the interval in $[s]_j$. But in each dimension i, the interval can only shrink a finite number of times: the upper bound can shrink from $+\infty$ to a finite value b once, and all following updates must set the new bound to a lower value in $\{s[v_i], \ldots, b-1\}$, which can only happen finitely often. The same is true for updates of the lower bound.

If the algorithm returns a plan, it is an optimal plan for an abstraction of Π and must therefore be optimal for Π. □

Experiments

We implemented CEGAR in Numeric Fast Downward (Aldinger and Nebel 2017). We test three variants, $h^{\text{cegar}}_{\text{rnd}}$, $h^{\text{cegar}}_{\text{min}}$, and $h^{\text{cegar}}_{\text{max}}$, which differ in the refinement strategy (Seipp and Helmert 2018; Speck and Seipp 2022), i.e., on which variable to split when multiple splits are possible. The variants either choose a split at random or select a variable with minimal or maximal number of values in the flawed abstract state. Intuitively, $h^{\text{cegar}}_{\text{min}}$ tends to refine finite-domain variables, while $h^{\text{cegar}}_{\text{max}}$ tends to refine numeric variables.

Using A* (Hart, Nilsson, and Raphael 1968), we compare our heuristics against other admissible heuristics for numeric planning: the blind heuristic h^{blind}, pattern database heuristics h^{pdbs}, h^{cpdbs}, and h^{ipdbs} (Gnad et al. 2025), and the LM-Cut heuristic h^{lmcut} (Kuroiwa, Shleyfman, and Beck 2022). We use limits of 30 min and 3.5 GB per run. In CEGAR, we refine for 15 min and, if no concrete solution is found, use the abstraction as heuristic. We use all domains from Gnad et al. (2025), including IPC 2023 domains (Taitler et al. 2024) that lie in the supported fragment. Our code and data are available online (Schindler, Speck, and Helmert 2026).

Tab. 1 shows the number of solved tasks. The CEGAR heuristics solve overall slightly more tasks than the other approaches. While they perform similarly to other abstraction heuristics based on pattern databases, they show complementary strengths to the LM-Cut heuristic. This is particularly evident in the comparatively large tasks of the Minecraft domains, where abstractions excel, whereas in many other domains LM-Cut performs best. The refinement strategy appears to have a minor effect, with a slight preference for refining finite-domain variables first, as in $h^{\text{cegar}}_{\text{min}}$.

Fig. 1 compares the heuristic quality of $h^{\text{cegar}}_{\text{min}}$ with other heuristics from the literature. Overall, the heuristic quality of $h^{\text{cegar}}_{\text{min}}$ is higher in many instances than that of the other approaches. In fact, the number of node expansions before the last f-layer (necessary expansions) is zero in the vast majority of solved instances for $h^{\text{cegar}}_{\text{min}}$. This is because we often find a concrete solution in the abstraction during refinement.

Future Work

We introduced the first, though basic, variant of Cartesian CEGAR for numeric planning. For future work, we plan to investigate saturated cost-partitioning based on abstractions derived from individual goal atoms and landmarks, which has led to significant performance improvements in classical planning (Seipp and Helmert 2018; Seipp, Keller, and Helmert 2020). Additionally, we aim to support a richer numeric planning formalism beyond integer-restricted tasks and to explore predicate abstraction (Graf and Saïdi 1997).

Acknowledgements

This work was funded by the Swiss National Science Foundation (SNSF) as part of the project "Unifying the Theory and Algorithms of Factored State-Space Search" (UTA).

References

Aldinger, J.; and Nebel, B. 2017. Interval Based Relaxation Heuristics for Numeric Planning with Action Costs. In *Proc. KI 2017*, 15–28.

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS^+ Planning. *Computational Intelligence*, 11(4): 625–655.

Büchner, C.; Ferber, P.; Seipp, J.; and Helmert, M. 2024. Abstraction Heuristics for Factored Tasks. In *Proc. ICAPS 2024*, 40–49.

Gnad, D.; Alon, L.; Weiss, E.; and Shleyfman, A. 2025. PDBs Go Numeric: Pattern-Database Heuristics for Simple Numeric Planning. In *Proc. AAAI 2025*, 26507–26515.

Graf, S.; and Saïdi, H. 1997. Construction of Abstract State Graphs with PVS. In *Proc. CAV 1997*, 72–83.

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.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proc. ICAPS 2009*, 162–169.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating 'Ignoring Delete Lists' to Numeric State Variables. *JAIR*, 20: 291–341.

Kuroiwa, R.; Shleyfman, A.; and Beck, J. C. 2022. LM-Cut Heuristics for Optimal Linear Numeric Planning. In *Proc. ICAPS 2022*, 203–212.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Pozo, M.; Torralba, Á.; and Linares López, C. 2024. Gotta Catch 'Em All! Sequence Flaws in CEGAR for Classical Planning. In *Proc. ECAI 2024*, 4287–4294.

Scala, E.; Haslum, P.; Magazzeni, D.; and Thiébaux, S. 2017. Landmarks for Numeric Planning Problems. In *Proc. IJCAI 2017*, 4384–4390.

Scala, E.; Haslum, P.; Thiebaux, S.; and Ramirez, M. 2016. Interval-Based Relaxation for General Numeric Planning. In *Proc. ECAI 2016*, 655–663.

Schindler, T.; Speck, D.; and Helmert, M. 2026. Code, Benchmarks and Data for the ICAPS 2026 Paper "Cartesian Abstraction Refinement for Simple Numeric Planning". https://doi.org/10.5281/zenodo.18999076.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *JAIR*, 62: 535–577.

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated Cost Partitioning for Optimal Classical Planning. *JAIR*, 67: 129–167.

Seipp, J.; von Allmen, S.; and Helmert, M. 2020. Incremental Search for Counterexample-Guided Cartesian Abstraction Refinement. In *Proc. ICAPS 2020*, 244–248.

Speck, D.; and Seipp, J. 2022. New Refinement Strategies for Cartesian Abstractions. In *Proc. ICAPS 2022*, 348–352.

Taitler, A.; Alford, R.; Espasa, J.; Behnke, G.; Fišer, D.; Gimelfarb, M.; Pommerening, F.; Sanner, S.; Scala, E.; Schreiber, D.; Segovia-Aguas, J.; and Seipp, J. 2024. The 2023 International Planning Competition. *AI Magazine*, 45(2): 280–296.