Domain-Abstraction Heuristics for Simple Numeric Planning

CEGAR over infinite integer domains

Markus Fritzsche Mikhail Gruntov Alexander Shleyfman
Daniel Gnad

ICAPS 2026

Simple numeric planning

A task is $\Pi = \langle V, A, s_0, G \rangle$ with $V = V_p \cup V_n$.

State variables

$$v \in V_p : D_v \text{ finite}, \qquad v \in V_n : D_v = \mathbb{Z}.$$

A state is a complete assignment to all variables.

Conditions

$$\langle v, d \rangle \quad \text{or} \quad v \bowtie k, \quad \bowtie \in \{\leq, =, \geq\}.$$

Actions

$$a = \langle \mathrm{pre}(a), \mathrm{eff}(a), \mathrm{cost}(a) \rangle.$$

Effects are finite-domain assignments or numeric updates:

$$v := d, \qquad v+ = c, \quad c \in \mathbb{Z} \setminus \{0\}.$$

Abstraction heuristics expand abstract state spaces; one integer variable already makes the state space infinite.

Example: Sailing with one person to save

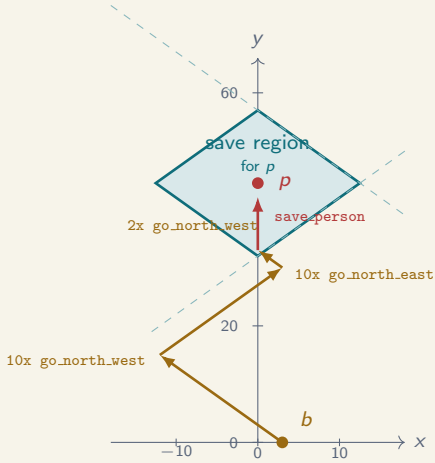
One boat, one person

$$x(b) = 3$$
$$y(b) = 0$$
$$d(p) = 32$$
$$saved(p) = false$$

goal: $saved(p)$

save iff

$$x(b) + y(b) \in [d(p), d(p) + 25]$$
$$y(b) - x(b) \in [d(p), d(p) + 25]$$



A common abstraction view

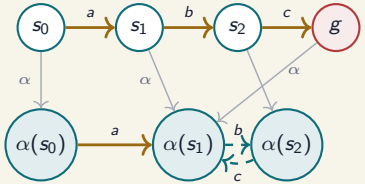
A state abstraction $\alpha : S \to S^\alpha$ preserves transitions:

$$s \xrightarrow{a} t \quad \Rightarrow \quad \alpha(s) \xrightarrow{a} \alpha(t).$$

Admissible abstraction heuristic

$$h^\alpha(s) = h^*_{\mathcal{T}^\alpha}(\alpha(s)).$$

Homomorphism: every concrete plan maps to an abstract plan. Thus abstract optimal cost is a lower bound: $h^\alpha(s) \leq h^*(s)$.

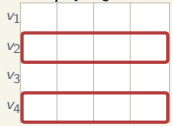


Hence $h^*(\alpha(s_0)) = 1 \leq 3 = h^*(s_0)$.

PDBs, domain abstractions, and Cartesian sets

Picture: FDR with $D(v_1) = \cdots = D(v_4) = \{0, 1, 2, 3\}$. Left/middle: rows are variables, columns are values. Right: a two-variable state-space slice.

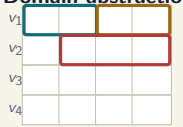
PDB / projection



omit rows

Pattern $P = \{v_1, v_3\}$. Omitted variables map to one trivial value.

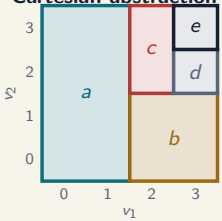
Domain abstraction



group columns per row

One domain partition per variable.

Cartesian abstraction



product rectangles

Product states; local splits.

Domain abstractions sit between PDB projections and Cartesian abstractions: more expressive than omitting variables, less general than arbitrary Cartesian sets.

Domain abstractions: finite and numeric variables

Same abstraction idea

For each variable v, partition its domain; each value maps to its block.

Classical finite domain

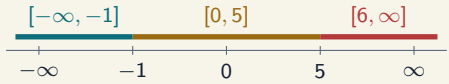
$$D_v = \{0, 1, 2, 3, 4\}, \qquad \{0, 1\} \mid \{2\} \mid \{3, 4\}$$

Value 4 maps to block $\{3, 4\}$.

Numeric domain (our contribution)

For $v \in V_n$, blocks are integer intervals covering $\mathbb{Z}$:

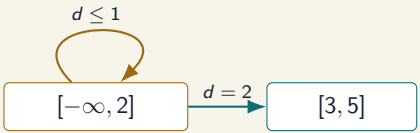
$$[-\infty, -1] \mid [0, 5] \mid [6, \infty].$$



Interval abstractions: conditions and effects

Abstract effects may branch

For $\alpha(v) = \{[-\infty, 2], [3, 5], [6, \infty]\}$, $v+ = 1$ can branch.



Validate the abstract trace concretely; deviations refine.

Relaxed conditions

In an abstract state, $s^\alpha[v]$ is an interval.

$$s^\alpha \vDash v \bowtie k \quad \text{iff} \quad \exists d \in s^\alpha[v] : d \bowtie k.$$

Ambiguity is true: the abstraction stays optimistic.

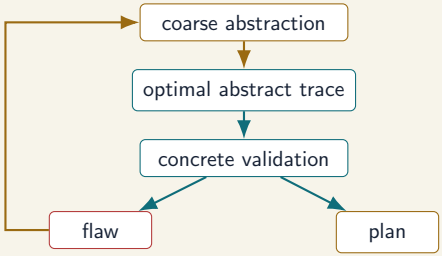
$v \geq 3$ on $[0, 5]$

CEGAR loop for numeric domain abstractions

Counterexample-guided abstraction refinement (CEGAR): standard solution in the literature.

1. Build a coarse domain abstraction.
2. Compute a cheapest abstract plan trace.
3. Execute the action sequence from the concrete state.
4. If it is concrete and reaches the goal: done.
5. Otherwise split the responsible domain block.

In practice: many abstractions, combined canonically.



Flaws: goal, precondition, deviation.

Deviation flaws are the numeric-specific addition (our contribution).

Experimental setup

- Numeric Fast Downward implementation.
- A* with admissible heuristics.
- 30 min / 8 GiB IPC 2023 limits.
- Downward Lab 8.2.

- Baselines: blind, numeric LM-cut[1], Vanilla PDBs[2], LMc-enhanced PDBs[3]
- Domain abstractions: sDA and cDA.
- Domains from Gnad et al., AAAI 2025.

[1] Kuroiwa et al., JAIR 2022: The LM-Cut Heuristic Family for Optimal Numeric Planning with Simple Conditions.

[2] Gnad et al., AAAI 2025: PDBs Go Numeric: Pattern-Database Heuristics for Simple Numeric Planning.

[3] Fritzsche et al., AAAI 2026: Managing Infinite Abstractions in Numeric Pattern Database Heuristics.

Coverage summary

Benchmark group	Blind	LMc	Vanilla PDBs	LMc-enhanced PDBs	sDA (ours)	cDA (ours)
RT domains	86	106	110	109	109	**119**
Compiled to RT	117	**204**	144	163	123	136
Total	203	**310**	254	272	232	255

Native RT: cDA solves most instances and beats numeric LM-cut.

Compiled to RT: cDA trails LM-cut and LLB; compilation remains future work.

What to remember

- Numeric variables make abstract state spaces infinite.
- Domain abstractions generalize via integer intervals.
- Relaxed conditions and branching effects preserve admissibility.
- Classical CEGAR generalizes with deviation flaws.
- Canonical collections are competitive on native RT domains.

Why not saturated cost partitioning?

Saturated cost partitioning is the natural next question.

So far, we barely see improvements over the canonical combination.

We are identifying the problems there; this is future work.

Stay tuned.

markus.fritzsche@liu.se

appendix

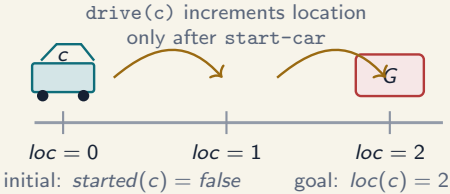
Appendix: CEGAR example: car task

Task

Initial state: $loc(c) = 0$, $started(c) = false$
Goal: $G = \{loc(c) = 2\}$

Actions

`start-car`	pre $\top$; eff $started(c) := true$
`drive(c)`	pre $started(c)$; eff $loc(c) := loc(c) + 1$



Initial abstraction

Start with one block for each variable:

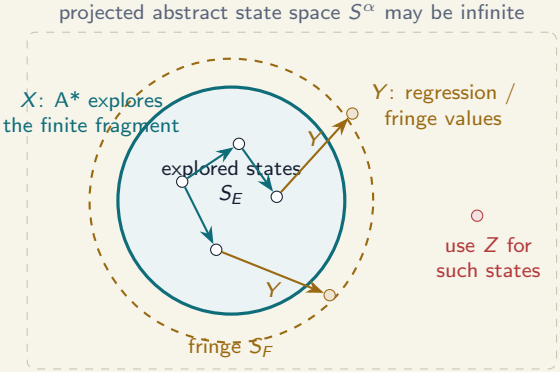
$$\alpha(loc(c)) = \{[-\infty, \infty]\}$$
$$\alpha(started(c)) = \{\{false, true\}\}$$

Appendix: CEGAR example: abstraction evolution

Current abstract plan	Flaw and refinement	$\alpha(\mathit{loc}(c))$	$\alpha(\mathit{started}(c))$
$\langle\rangle$	**Goal flaw.** Empty plan reaches the abstract goal, but concrete $\mathit{loc}(c) = 0$. Split at the goal boundary.	$\{[-\infty, 1], [2, \infty]\}$	$\{\{\mathit{false}, \mathit{true}\}\}$
$\langle \mathit{drive} \rangle$	**Precondition flaw.** `drive` needs concrete $\mathit{started}(c) = \mathit{true}$. Split the Boolean block.	$\{[-\infty, 1], [2, \infty]\}$	$\{\{\mathit{false}\}, \{\mathit{true}\}\}$
$\langle \mathit{start}, \mathit{drive} \rangle$	**Deviation flaw.** The abstract successor is $[2, \infty]$, but concrete execution reaches $\mathit{loc}(c) = 1$. Split at the witness.	$\{[-\infty, 0], [1, 1], [2, \infty]\}$	$\{\{\mathit{false}\}, \{\mathit{true}\}\}$
$\langle \mathit{start}, \mathit{drive}, \mathit{drive} \rangle$	**Validated.** The abstract plan is executable concretely and reaches the goal.	$\{[-\infty, 0], [1, 1], [2, \infty]\}$	$\{\{\mathit{false}\}, \{\mathit{true}\}\}$

Initially, $\alpha(\mathit{loc}(c)) = \{[-\infty, \infty]\}$ and $\alpha(\mathit{started}(c)) = \{\{\mathit{false}, \mathit{true}\}\}$.

Appendix: Numeric PDBs: partial exploration



XYZ naming scheme

- X: heuristic for abstract A* exploration;
- Y: heuristic for boundary regression;
- Z: heuristic for failed lookup.

B = blind, L = numeric LM-cut

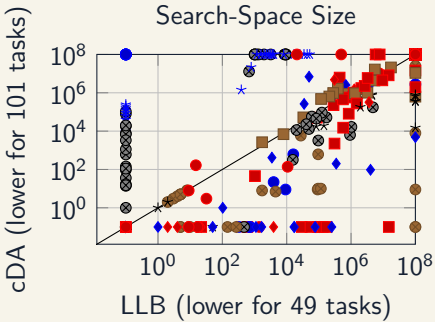
BBB: blind / blind / blind
LLB: LM-cut / LM-cut / blind

Appendix: search guidance against numeric PDBs

Empirical pattern

Domain abstractions are complementary to numeric PDBs and often provide better search guidance.

- BBB/LLB explore selected abstract states.
- Failed lookups need a backup heuristic.
- Domain abstractions are total: every concrete state has an abstract state.



Expansions until the last f-layer.

Appendix: sailing action schemas

Movement actions

All movement actions have precondition $\top$.

action	effect
`go_north_east(`b`)`	$x(b)+=1.5,\ y(b)+=1.5$
`go_north_west(`b`)`	$x(b)-=1.5,\ y(b)+=1.5$
`go_est(`b`)`	$x(b)+=3$
`go_west(`b`)`	$x(b)-=3$
`go_south_west(`b`)`	$x(b)+=2,\ y(b)-=2$
`go_south_east(`b`)`	$x(b)-=2,\ y(b)-=2$
`go_south(`b`)`	$y(b)-=2$

Saving a person

`save_person(`b, p`)`

pre: $d(p) \leq x(b) + y(b) \leq d(p) + 25$
$d(p) \leq y(b) - x(b) \leq d(p) + 25$
eff: $\mathit{saved}(p) := \mathit{true}$

The sailing picture shows these preconditions as the diamond-shaped save region.

Appendix: the RT formalism used in the paper

Restricted tasks (RT)

We present the construction for Hoffmann's restricted numeric tasks:

- numeric conditions compare one variable to an integer;
- numeric effects add or subtract an integer constant;
- all variables are fully observable state variables.

Example: preconditions may contain $\mathit{fuel} \geq 1$; effects may contain $\mathit{fuel} += -1$.

Scope comment

In the code, we support the full simple numeric planning fragment, not only the RT fragment used for the clean presentation here.

Why start with RT?

RT isolates the abstraction issue: finite actions, integer variables, and infinite transition systems.

Appendix: how numeric refinement splits intervals

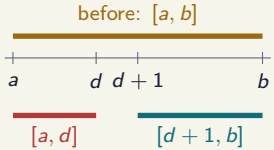
Suppose a condition $v \geq k$ fails for the current concrete value $d < k$, but its abstract interval still made the condition true.

Split at the witness

If $\alpha_v(d) = [a, b]$, replace it by

$$[a, d] \qquad \text{and} \qquad [d + 1, b].$$

The left interval cannot satisfy $v \geq k$; ambiguity is reduced only where the counterexample requires it.



Deviation flaw

For $v+ = c$, split at the concrete source value that caused the wrong abstract successor.

Appendix: semi-completeness statement

Theorem

The proposed CEGAR procedure is semi-complete on RT tasks without zero-cost actions.

Proof idea

If a concrete plan of cost L exists and all action costs are positive, then only finitely many concrete states and numeric values are reachable within cost L.

Why refinement terminates

Splits are made only at values witnessed inside this finite fragment. Repeated flaws are eliminated or the abstract plan cost increases until a valid concrete plan is found.

Appendix: implementation choices

Planner

Implemented in Numeric Fast Downward and used as an admissible heuristic for A*.

Variants

- **sDA**: a single domain abstraction.
- **cDA**: canonical heuristic over a collection.

Refinement controls

- abstraction-size and collection-size limits;
- blacklisting variables to diversify abstractions;
- flaw selection by minimum abstract state-space growth.

Caveat

The best cDA configuration was tuned on the evaluation benchmarks; PDB baselines were not tuned in the same way.

Appendix: abstraction class references

- PDB abstractions: projections onto a subset of variables.
- Cartesian abstractions: abstract states are Cartesian sets over the full variable set; CEGAR refines them locally.
- Domain abstractions: per-variable domain partitions, described as a middle point between PDBs and Cartesian abstractions.

Relevant papers

Seipp and Helmert, "Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning", JAIR 62, 2018.
Kreft, Buchner, Sievers, Helmert, "Computing Domain Abstractions for Optimal Classical Planning with Counterexample-Guided Abstraction Refinement", ICAPS 2023.

Appendix: exact experimental setup

- Numeric Fast Downward implementation.
- A* search with admissible heuristics.
- 30 minute / 8 GiB IPC 2023 limits.
- Downward Lab 8.2.
- Baselines: blind, numeric LM-cut, numeric PDB variants BBB and LLB.
- Domain-abstraction variants: sDA and cDA.
- Benchmarks from recent SNP work.