Pattern Database Heuristics for Lifted Planning

Mika Skjelnes Dominik Drexler Jordan Thayer
Daniel Gnad Jendrik Seipp

Linköping University · Heidelberg University

HSDIP 2026

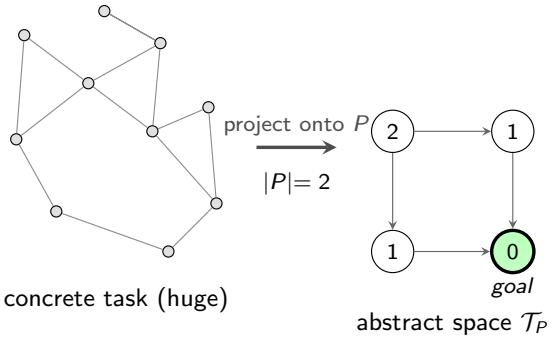
Grounding is the Bottleneck

- Classical planning almost always **grounds first**.
- But grounding can **blow up**: organic synthesis, large-scale logistics.
- Lifted *admissible* heuristics are often **weaker**, or far **harder to compute** without the grounded task.

⋆ **pattern databases**, has **no** lifted version.

Patterns and Pattern Databases

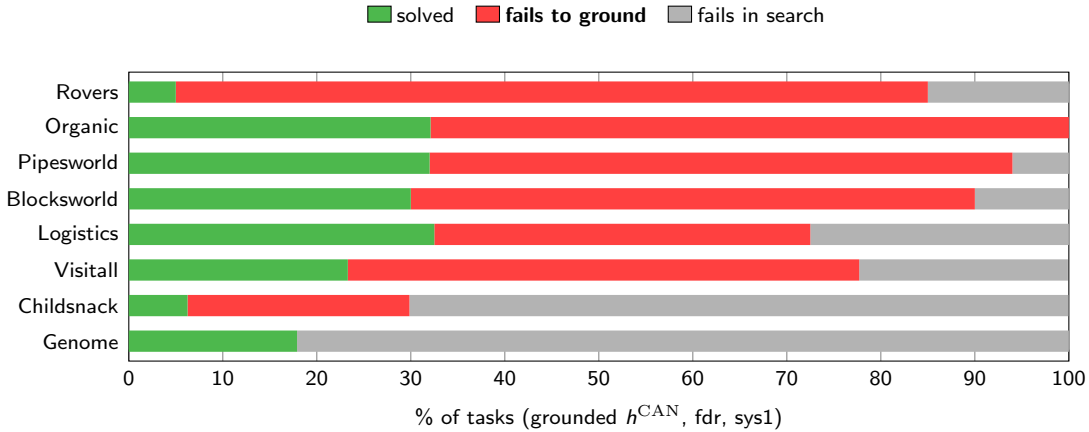
- A **pattern** P projects the task onto a subset of its atoms.
- Precompute optimal goal distances over the small abstract space $\mathcal{T}_P$; just **look them up**.
- Admissible, consistent, **very cheap**.
- (Natural extension) Combine a *collection* via *cost partitioning*.



We don't get PDBs for free

Building a PDB the classical way needs the **grounded task**, exactly what we cannot afford.

Grounding Blows Up Before Search Even Starts



Outcome of the strongest grounded PDB planner (h^{CAN}, FDR, sys1) on 862 HTG tasks. Limits: 8 GiB memory, 30 min.

32 % of tasks (273/862) fail *during grounding*; ground tasks reach **millions** of operators (Logistics 3.1M).

Translating PDBs to the Lifted Setting (the heart of the work)

Idea: build PDBs *directly* on the lifted task, lazily.

Without the grounded task, *three* steps must be rethought:

- **PDBs without grounding the task.**
  Consider only substitutions σ whose effect σa *alters* the abstract state s.
- **Detecting conflicting PDBs.**
  Relaxed estimation catching a *superset* of all conflicts.
- **Generating the pattern collection.**
  Grow the causal graph **lazily, backwards** from each goal atom.

Preliminary Results

	Lifted baselines			Grounded PDB (h^{CAN})					Lifted PDB (ours)	
Domain	$h^{\max}$	LMC-f	LMC-l	bin_1	bin_2	fdr_1	fdr_2	blind	sys1	sys2
Blocksworld (40)	0	0	1	2	4	**12**	**12**	0	5	7
Childsnack (144)	2	0	2	8	6	**9**	6	4	4	4
Genome (312)	44	30	40	**56**	30	**56**	44	44	50	24
Logistics (40)	5	0	0	0	0	13	**18**	5	5	10
Organic (56)	**44**	35	42	18	18	18	18	**44**	1	0
Pipesworld (50)	8	2	10	16	16	16	**17**	13	14	2
Rovers (40)	1	0	0	2	2	2	3	3	3	**4**
Visitall (180)	**67**	26	36	15	15	42	42	36	46	46
Total (862)	171	93	131	117	91	168	160	149	128	97

Coverage on HTG benchmarks (unit cost). **Bold** = best per row. $h^{\max}$/LMC = lifted $h^{\max}$ / LM-cut (full, least); GR = grounded canonical PDBs (binary / FDR encoding); sys$k$ = systematic patterns up to size k. Genome = GED + GED-split (312).

Setup: compute cluster, 8 GiB memory, 30 min time limit.

Question 1: Better Patterns

- Too many interesting patterns ⇒ don't build them *all*.

sys2: keep only the first k patterns

Domain	all	10	50	100
Blocksworld (40)	7	5	7	**8**
Genome (312)	24	**48**	**48**	40
Organic (56)	0	**2**	0	0
Pipesworld (50)	2	**14**	3	3
Rovers (40)	**4**	3	**4**	**4**
Total (862)	97	**132**	122	115

Fewer patterns ⇒ *more* solved (k=10: **+35**). Flat domains omitted.

Directions

- **Meta-ordering:** which patterns to generate *first*?
- **Multi-valued patterns** via lifted mutex groups (Fišer 2023).
- **iPDB-style** hill-climbing selection, lifted.

What selection strategy pays off *without* grounding?

Two More Threads

Projection is *not* the bottleneck

Building $\mathcal{T}_P$ is a median $< 1\%$ of runtime, the memory sink is combining *many* patterns, which loops back to selection.

Cost partitioning

The canonical heuristic is just *one* way to combine PDBs. Cost partitioning (post-hoc optimization, saturated cost partitioning) generally produces **stronger** admissible estimates from the *same* collection.

Which cost-partitioning method ports cleanly to lifted PDBs?

Takeaways

PDBs were **missing** in the lifted setting.

We build patterns *and* PDBs **without grounding the whole task**.

Already competitive on speed for tasks that we do solve, but we must improve the overall performance. Next: **better patterns, cost partitioning**.

Any thoughts or suggestions?

mika.skjelnes@liu.se

Backup: Pattern Explosion Seems to be a Bottleneck

	sys1		sys2	
Domain	median	max	median	max
Blocksworld	4	5	18	25
Childsnack	5	7	273	10000
GED	33	52	2923	7072
GED-split	33	52	990	2054
Logistics	2	4	8	12
Organic	45	100	208	636
Pipesworld	5	8	1283	4327
Rovers	5	8	63	216
Visitall	2	3	14	30

Patterns collected per domain (tasks where pattern generation completed).

Size 1 $\rightarrow$ 2 explodes the collection (GED **33**$\rightarrow$**2900**, max 10 000) $\Rightarrow$ combining them all **exhausts memory**.

Backup: Where Does It Fail?

sys1

		Pattern gen.		Projection		Heur. init		Search	
Domain	Solved	T/O	M/O	T/O	M/O	T/O	M/O	T/O	M/O
Blocksworld	5	–	–	–	–	–	–	**24**	11
Childsnack	4	–	–	–	–	–	–	35	**105**
GED	26	–	–	–	–	–	–	**130**	–
GED-split	24	–	–	–	–	–	4	**128**	–
Logistics	5	–	–	–	–	–	–	**32**	3
Organic	1	20	–	**35**	–	–	–	–	–
Pipesworld	14	–	–	–	–	–	–	12	**24**
Rovers	3	–	–	–	–	–	–	**37**	–
Visitall	46	–	36	–	–	–	–	–	**98**
Total	128	20	36	35	0	0	4	398	241

sys2

		Pattern gen.		Projection		Heur. init		Search	
Domain	Solved	T/O	M/O	T/O	M/O	T/O	M/O	T/O	M/O
Blocksworld	7	–	–	–	11	**22**	–	–	–
Childsnack	4	–	–	–	8	11	48	**73**	–
GED	16	–	–	–	4	10	**124**	2	–
GED-split	8	–	–	–	–	–	**144**	4	–
Logistics	10	–	–	–	–	–	–	**28**	2
Organic	0	**38**	–	18	–	–	–	–	–
Pipesworld	2	4	–	–	**42**	1	1	–	–
Rovers	4	–	–	–	–	–	4	**32**	–
Visitall	46	–	36	–	–	–	–	18	**80**
Total	97	42	36	18	65	44	321	157	82

862 HTG tasks. **Bold** = dominant failure mode per row.

Backup: Enumerating Meaningful Substitutions

Both steps enumerate substitutions σ that bind a schema's *effect*

Build the PDB: transitions $s \xrightarrow{a} s'$

$\delta^+ = s' \setminus s, \quad \delta^- = s \setminus s'$

return all σ with
$\mathrm{eff}^+(a)\sigma = \delta^+$
$\mathrm{eff}^-(a)\sigma = \delta^-$
$\mathrm{pre}(a)\sigma \subseteq s$

effect *pins down* $\sigma \Rightarrow$ no self-loops

Find patterns: causal-graph growth

$Q \leftarrow [g], \quad G \leftarrow \{g\}$
while Q nonempty:
  $v \leftarrow Q.\mathrm{pop}()$
  for (a, σ) with $v \in \mathrm{eff}(a)\sigma$:
    for non-static $v' \in \mathrm{pre}(a)\sigma$:
      link $v' \to v$; push new v'

backward + lazy $\Rightarrow$ only the relevant fragment