Pattern Database Heuristics for Fully Observable Nondeterministic Planning

Robert Mattmüller[1], Manuela Ortlieb[1],
Malte Helmert[1], and Pascal Bercher[2]

[1]University of Freiburg
[2]Ulm University

May 14, 2010

ICAPS 2010
Toronto

Motivation

Successful techniques for classical planning:

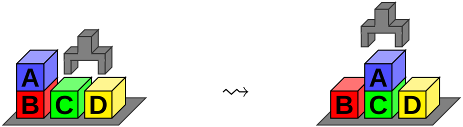
- Heuristic search,
- Various heuristics: abstraction, delete-relaxation, . . .

Classical planning too restricted for many applications.

⇒ Extend applicability of techniques to more expressive models.

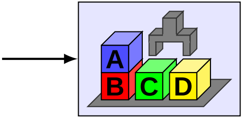
Problem

- Problem: nondeterministic planning
- Environment: fully observable, static, discrete
- Solutions: strong cyclic plans
- Solution Technique: progression search with PDB heuristic
- Example: blocksworld with slippery gripper
  (blocks can fall down)

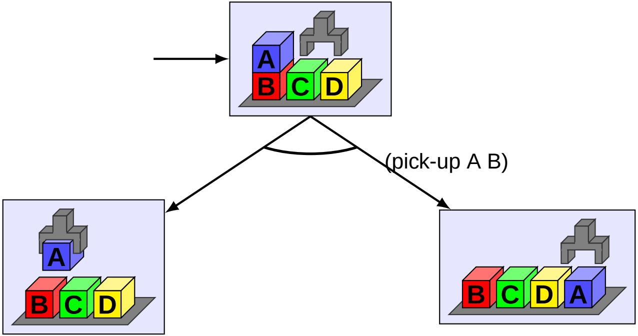


init goal

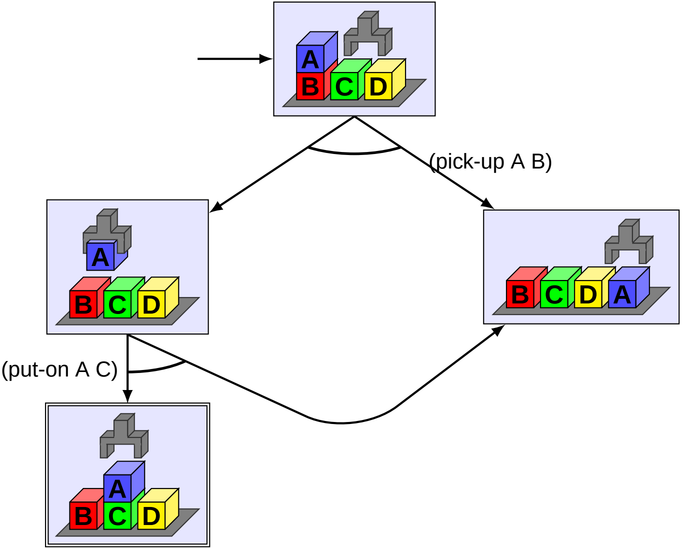
Example



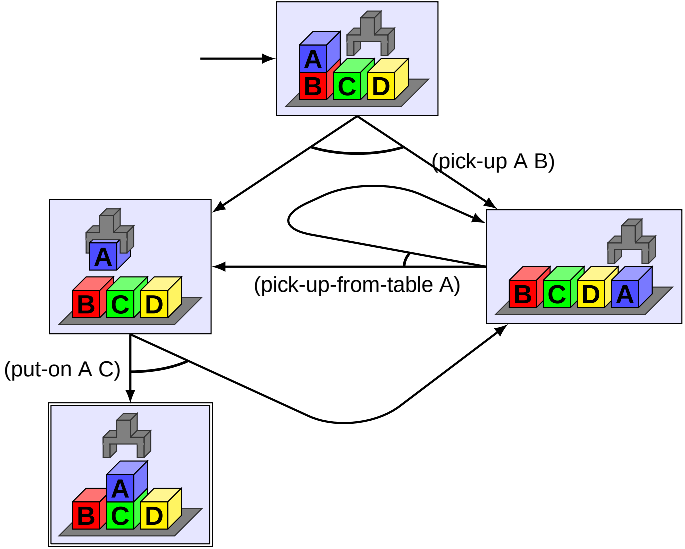
Example



Example



Example



Strong Cyclic Planning

Question: How to compute a strong cyclic plan?

Answer: Possible approaches are . . .

- Symbolic regression search [Cimatti et al. 2003, Kissmann and Edelkamp 2009],
  - Advantage: good data structure (BDDs)
  - Disadvantage: uninformed
- Iteratively apply classical planner [Kuter et al. 2008], or
  - Advantage: informed
  - Disadvantage: detour via classical planning
- Informed explicit-state progression search.
  - Advantage: informed, no classical planner needed
  - Disadvantage: explicit state representation

Computing Strong Cyclic Plans

Variant of LAO* search [Hansen and Zilberstein 2001]

- start with initial node
- while initial node unsolved:
  - trace most promising partial solution
  - expand unexpanded nongoal node(s)
  - initialize heuristics for new nodes
  - update heuristics of ancestors
  - run solve-labeling procedure
- return solution graph

Computing Strong Cyclic Plans

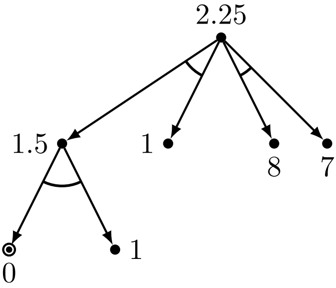
Variant of LAO* search [Hansen and Zilberstein 2001]

- start with initial node
- while initial node unsolved:
  - trace most promising partial solution
  - expand unexpanded nongoal node(s)
  - initialize heuristics for new nodes
  - update heuristics of ancestors
  - run solve-labeling procedure
- return solution graph

Computing Strong Cyclic Plans

Variant of LAO* search [Hansen and Zilberstein 2001]

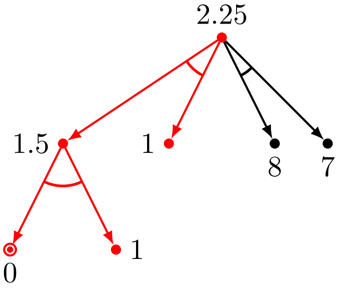
- start with initial node
- while initial node unsolved:
    - trace most promising partial solution
    - expand unexpanded nongoal node(s)
    - initialize heuristics for new nodes
    - update heuristics of ancestors
    - run solve-labeling procedure
- return solution graph



Computing Strong Cyclic Plans

Variant of LAO* search [Hansen and Zilberstein 2001]

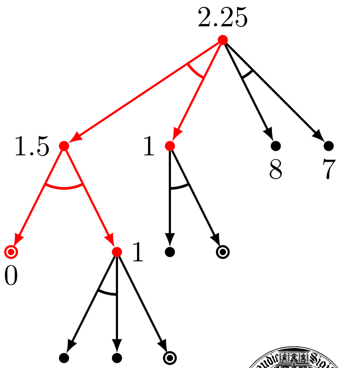
- start with initial node
- while initial node unsolved:
  - trace most promising partial solution
  - expand unexpanded nongoal node(s)
  - initialize heuristics for new nodes
  - update heuristics of ancestors
  - run solve-labeling procedure
- return solution graph



Computing Strong Cyclic Plans

Variant of LAO* search [Hansen and Zilberstein 2001]

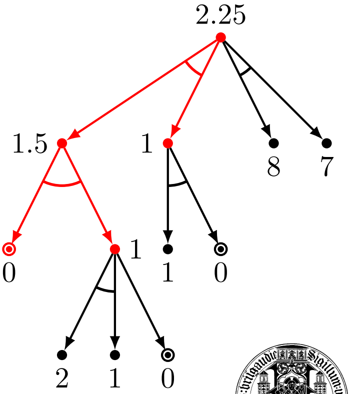
- start with initial node
- while initial node unsolved:
  - trace most promising partial solution
  - expand unexpanded nongoal node(s)
  - initialize heuristics for new nodes
  - update heuristics of ancestors
  - run solve-labeling procedure
- return solution graph



Computing Strong Cyclic Plans

Variant of LAO* search [Hansen and Zilberstein 2001]

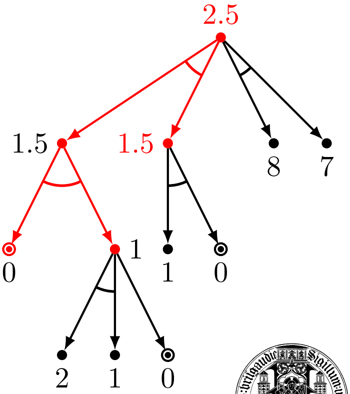
- start with initial node
- while initial node unsolved:
    - trace most promising partial solution
    - expand unexpanded nongoal node(s)
    - initialize heuristics for new nodes
    - update heuristics of ancestors
    - run solve-labeling procedure
- return solution graph



Computing Strong Cyclic Plans

Variant of LAO* search [Hansen and Zilberstein 2001]

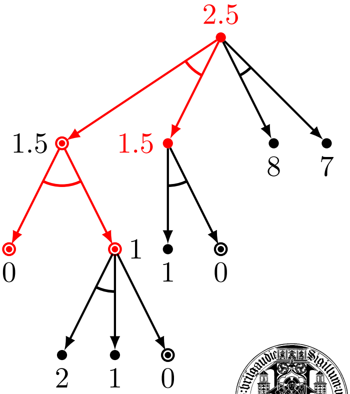
- start with initial node
- while initial node unsolved:
  - trace most promising partial solution
  - expand unexpanded nongoal node(s)
  - initialize heuristics for new nodes
  - update heuristics of ancestors
  - run solve-labeling procedure
- return solution graph



Computing Strong Cyclic Plans

Variant of LAO* search [Hansen and Zilberstein 2001]

- start with initial node
- while initial node unsolved:
  - trace most promising partial solution
  - expand unexpanded nongoal node(s)
  - initialize heuristics for new nodes
  - update heuristics of ancestors
  - run solve-labeling procedure
- return solution graph



Computing Strong Cyclic Plans

Variant of LAO* search [Hansen and Zilberstein 2001]

Details (for cyclic graphs and solutions):

- Solve labeling?
  - Nested fixpoint iteration.
- Updating heuristic estimates?
  - Value iteration (use discounting to ensure termination).
- Initializing heuristic estimates?
  - PDB heuristic. Following slides.

Pattern Database Heuristics

[Culberson and Schaeffer 1998, Edelkamp 2001]

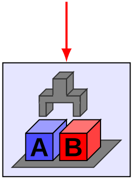
Basic Idea:

- Create abstract problem by ignoring some state variables.
- Use abstract costs as heuristic in original problem.
- Precompute abstract costs and store them in PDB.

Additive Pattern Databases:

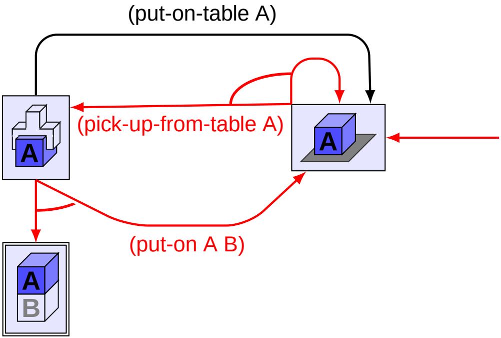
- Compute several abstractions.
- Use sum of abstract costs as heuristic.

Example: Stack **A** on **B**



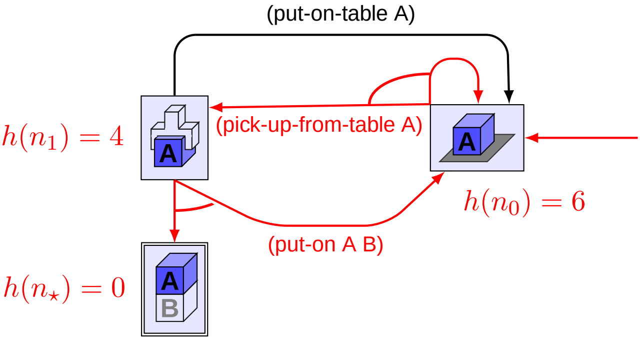
Example: Stack **A** on **B**

Abstraction to pattern $\left\{\ pos(\text{A})\ \right\}$. No-ops ignored.



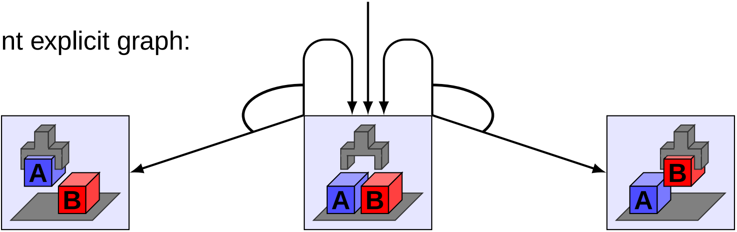
Example: Stack **A** on **B**

Cost: expected number of steps to goal (equal outcome probabilities).



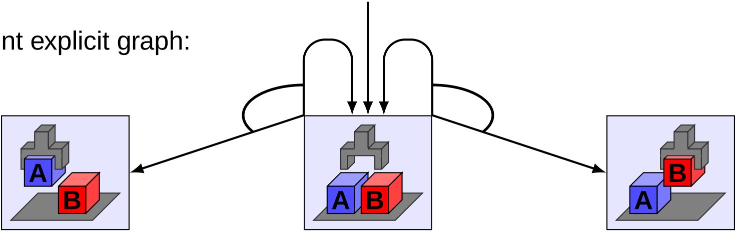
Example: Stack **A** on **B**

Current explicit graph:



Example: Stack **A** on **B**

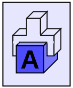
Current explicit graph:



abstraction to

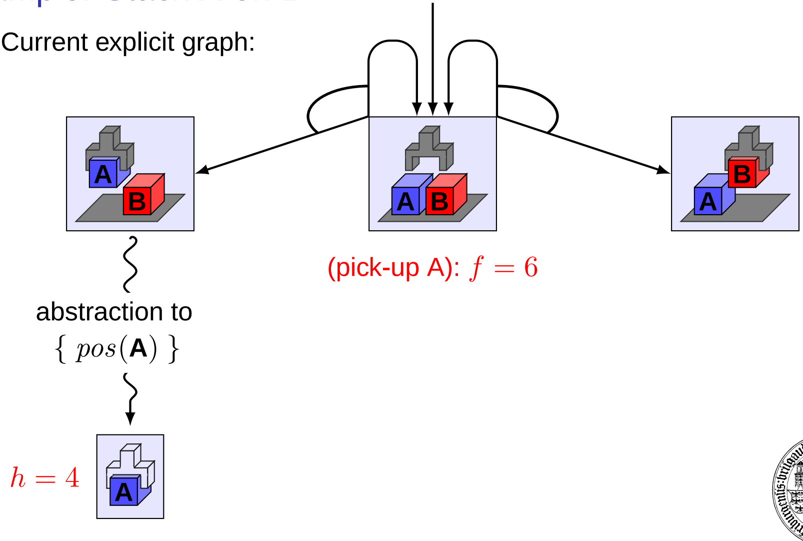
$\{\ pos(\textbf{A})\ \}$

$h = 4$



Example: Stack **A** on **B**

Current explicit graph:



(pick-up A): $f = 6$

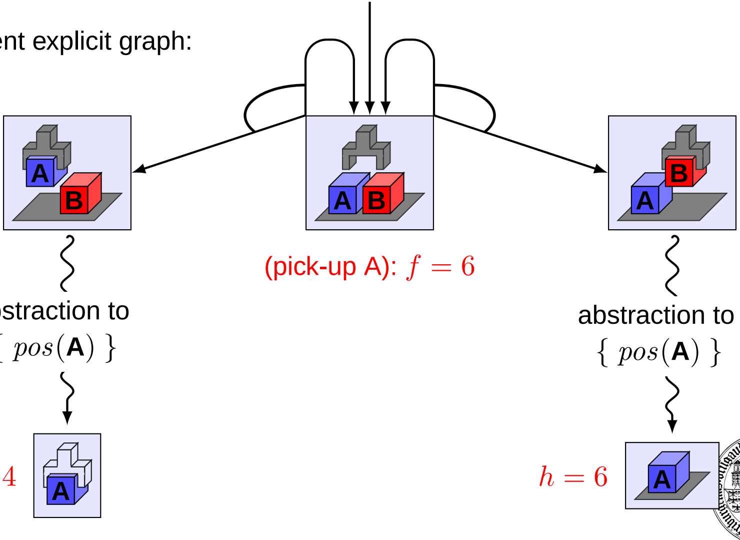
abstraction to

$\{ \; pos(\textbf{A}) \; \}$

$h = 4$

Example: Stack **A** on **B**

Current explicit graph:



(pick-up A): $f = 6$

abstraction to

$\{ \ pos(\textbf{A}) \ \}$

$h = 4$

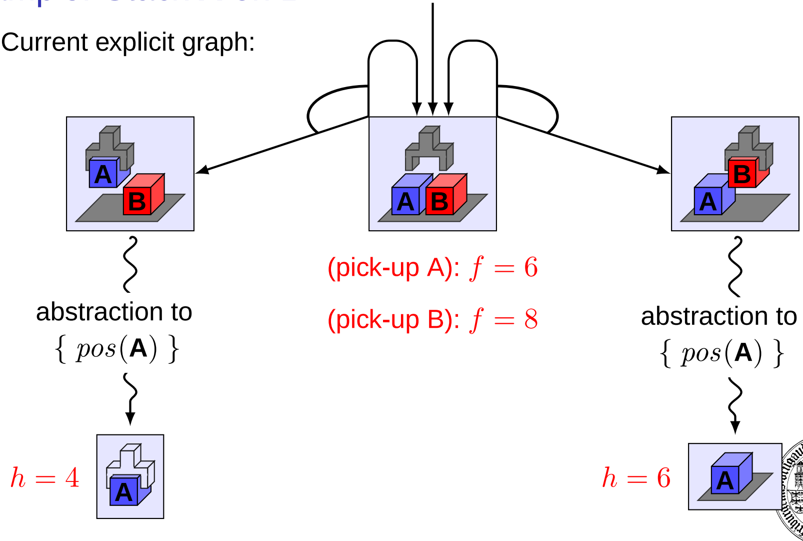
abstraction to

$\{ \ pos(\textbf{A}) \ \}$

$h = 6$

Example: Stack **A** on **B**

Current explicit graph:



(pick-up A): $f = 6$

(pick-up B): $f = 8$

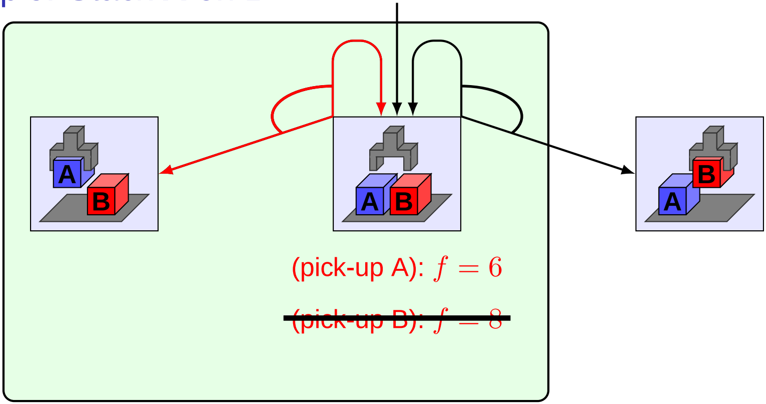
abstraction to $\{ \; pos(\textbf{A}) \; \}$

abstraction to $\{ \; pos(\textbf{A}) \; \}$

$h = 4$

$h = 6$

Example: Stack **A** on **B**



(pick-up A): $f = 6$

~~(pick-up B): $f = 8$~~

Pattern Selection

[Haslum et al. 2007]

Question: Which abstractions to use?

Problem: In general no domain knowledge.

Then how to compute suitable patterns?

- Local search in space of additive pattern collections.
- Quality criterion: minimize expected number of node expansions of IDA* search with pattern collection.

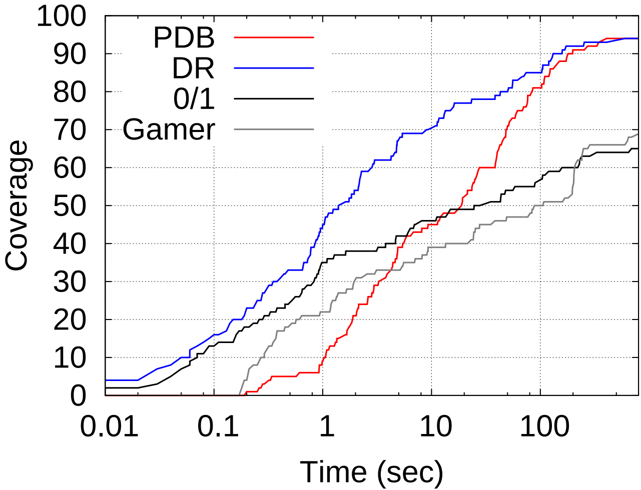
Experiments

- Compared planners:
  - LAO* + PDB heuristic.
  - LAO* + determinization + delete-relaxation heuristic.
  - LAO* without heuristic.
  - Gamer [Kissmann and Edelkamp 2009], which uses BDD-based symbolic reachability analysis and regression search.
- Tasks: IPC 2008 benchmarks (FOND track).

Experiments: Problems Solved (15 Minutes per Problem)

	Heuristic			
Domain (probs)	**PDB**	**DR**	**None**	**Gamer**
blocksworld (30)	10	10	10	10
faults (55)	55	54	33	34
first-responders (100)	23	24	19	19
forest (90)	6	6	3	6
overall (275)	**94**	**94**	**65**	**69**

Experiments: Coverage over Time



Experiments: Runtimes and Guidance (Node Expansions)

	PDBs		Delete Relaxation		None		Gamer
Problem	t	n	t	n	t	n	t
bw-1	25.10	**43**	0.20	50	**0.16**	296	220.73
bw-2	3.91	293	0.28	293	**0.07**	**92**	211.27
bw-3	4.23	**931**	**0.48**	**931**	0.85	2335	206.07
bw-4	4.90	**8515**	**4.84**	23154	6.39	24406	203.46
bw-5	4.88	4899	1.39	5968	**0.92**	**3476**	202.66
bw-6	4.39	2960	0.87	2960	**0.72**	**2710**	196.37
bw-7	5.43	10277	1.95	8549	**1.28**	**5373**	198.17
bw-8	5.90	14515	**1.84**	**8718**	3.22	15754	197.07
bw-9	4.42	**34**	0.35	626	**0.29**	534	203.51
bw-10	4.66	**1988**	0.95	3080	**0.89**	3904	205.38
faults-5-5	26.14	**329**	**0.73**	509	43.75	6138	168.35
faults-6-4	19.26	**5987**	**1.75**	7072	35.03	13157	88.39
faults-7-4	51.92	46964	**3.00**	**15152**	157.93	39895	25.34
faults-8-3	23.54	26311	**4.76**	**26304**	58.39	32351	107.69
faults-9-3	42.62	**23836**	**9.03**	49068	240.82	82410	285.15
faults-10-2	27.54	**882**	**2.13**	15012	48.02	20358	84.25
fr-1-6	2.83	9776	2.27	9776	119.56	**7414**	**1.23**
fr-2-4	**1.74**	**1191**	2.17	7780	8.13	6400	38.31
fr-4-3	2.61	**8060**	**2.19**	**8060**	24.66	20928	631.95
forest-2-5	21.31	6378	13.43	**4138**	229.22	6841	**2.03**

Discussion

- Delete relaxation + determinization surprisingly good.
- High preprocessing cost for PDB heuristics.
- But: PDBs provide good guidance, fast lookup.
- Generally: Informed progression search feasible approach to strong cyclic planning.

Discussion

- Delete relaxation + determinization surprisingly good.
- High preprocessing cost for PDB heuristics.
- But: PDBs provide good guidance, fast lookup.
- Generally: Informed progression search feasible approach to strong cyclic planning.

Thank you!