Getting the Most Out of Pattern Databases for Classical Planning

Florian Pommerening Gabriele Röger Malte Helmert

University of Basel, Switzerland

UNI BASEL

IJCAI 2013

Structure of this talk

Getting the Most Out of Pattern Databases for Classical Planning

Structure of this talk

Getting the Most Out of Pattern Databases for Classical Planning

Structure of this talk

Getting the Most Out of Pattern Databases for Classical Planning

Structure of this talk

Getting the Most Out of Pattern Databases for Classical Planning

Planning tasks

Planning task

- Variables
  - Variable assignments are states
- Operators
  - Allow to manipulate states
  - Transitions in implicitly defined transition system
- Initial state
- Goal description
  - Find (shortest) path

Solving planning tasks

Common approach

- Informed search algorithm + heuristic

Optimal planning

- A^* + admissible heuristic

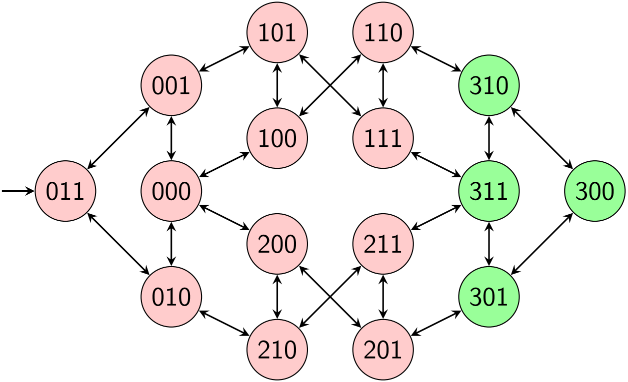
One type of admissible heuristics

- Pattern database (PDB) heuristics

Pattern database heuristics by example

Pattern database

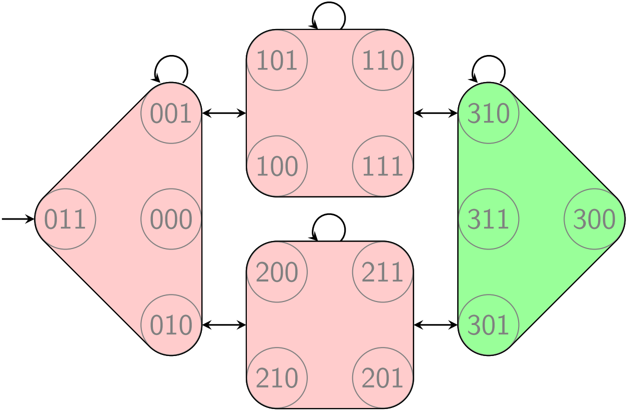
- Projection to subset of variables
- Abstract distance as heuristic value



Pattern database heuristics by example

Pattern database

- Projection to subset of variables
- Abstract distance as heuristic value



Running example

Example task

- Three variables $\{A, B, C\}$
- Each operator affects only one variable
- Pattern databases

$$
\begin{aligned}
h^{\{A\}}(s) &= h^{\{B\}}(s) &= h^{\{C\}}(s) &= 1 \\
h^{\{A,B\}}(s) &= h^{\{A,C\}}(s) &= h^{\{B,C\}}(s) &= 6
\end{aligned}
$$

What is the best heuristic value we can get from this information?

Using multiple PDBs

Getting Much Out of Pattern Databases for Classical Planning

Key idea: Use multiple PDBs

Two aspects

1. Pattern selection
2. Heuristic combination

Using multiple PDBs

Getting Much Out of Pattern Databases for Classical Planning

Key idea: Use multiple PDBs

iPDB procedure [Haslum et al.]

1. Pattern selection → hill-climbing search
2. Heuristic combination → canonical heuristic

Canonical heuristic

Additivity of a pattern collection $\mathcal{C}$

- No operator affects variables in two patterns
- Sum of heuristic values is admissible

Canonical heuristic

- Sum where possible, maximize where necessary
- $\mathit{MAS}(\mathcal{C})$: set of maximal additive subsets of $\mathcal{C}$

Definition (Canonical heuristic)

$$h^{\mathcal{C}}(s) = \max_{\mathcal{A} \in \mathit{MAS}(\mathcal{C})} \sum_{P \in \mathcal{A}} h^P(s).$$

Canonical heuristic (example task)

Example task

Each operator affects only one variable

$\Rightarrow$ Disjoint patterns are additive

- For example $h^{\{A\}}(s) + h^{\{B,C\}}(s) = 1 + 6$

$$h^{\mathcal{C}}(s) = 7$$

Post-hoc optimization heuristic: idea

Getting the Most Out of Pattern Databases for Classical Planning

$\rightarrow$ Can we do better than the canonical heuristic?

Post-hoc optimization heuristic: idea

Example task

$h^{\{A,B\}} = 6$

$\Rightarrow$ Any solution spends at least cost 6 on operators modifying A or B.

Post-hoc optimization heuristic: idea

Example task

$h^{\{A,B\}} = 6$

$\Rightarrow$ Any solution spends at least cost 6 on operators modifying A or B.

$$6 \quad = h^{\{A,B\}} \quad \leq \quad \textit{type-}A \quad + \quad \textit{type-}B$$

Post-hoc optimization heuristic: idea

Example task

$h^{\{A,B\}} = 6$

$\Rightarrow$ Any solution spends at least cost 6 on operators modifying A or B.

$$
\begin{array}{llllllll}
6 & = h^{\{A,B\}} & \leq & \textit{type-}A & + & \textit{type-}B & & \\
6 & = h^{\{A,C\}} & \leq & \textit{type-}A & + & & & \textit{type-}C \\
6 & = h^{\{B,C\}} & \leq & & & \textit{type-}B & + & \textit{type-}C \\
\hline
\end{array}
$$

Post-hoc optimization heuristic: idea

Example task

$h^{\{A,B\}} = 6$

$\Rightarrow$ Any solution spends at least cost 6 on operators modifying A or B.

$$
\begin{array}{rlccccccc}
6 & = h^{\{A,B\}} & \leq & \textit{type-}A & + & \textit{type-}B & & \\
6 & = h^{\{A,C\}} & \leq & \textit{type-}A & + & & & \textit{type-}C \\
6 & = h^{\{B,C\}} & \leq & & & \textit{type-}B & + & \textit{type-}C \\
\hline
18 & & \leq & 2\textit{type-}A & + & 2\textit{type-}B & + & 2\textit{type-}C
\end{array}
$$

Post-hoc optimization heuristic: idea

Example task

$h^{\{A,B\}} = 6$

$\Rightarrow$ Any solution spends at least cost 6 on operators modifying A or B.

$$
\begin{array}{lllccccc}
6 & = h^{\{A,B\}} & \leq & \textit{type-}A & + & \textit{type-}B & & \\
6 & = h^{\{A,C\}} & \leq & \textit{type-}A & + & & & \textit{type-}C \\
6 & = h^{\{B,C\}} & \leq & & & \textit{type-}B & + & \textit{type-}C \\
\hline
18 & & \leq & 2\textit{type-}A & + & 2\textit{type-}B & + & 2\textit{type-}C \\
9 & & \leq & \textit{type-}A & + & \textit{type-}B & + & \textit{type-}C
\end{array}
$$

$\Rightarrow$ at least cost 9 in any plan

Post-hoc optimization heuristic: idea

Example task

$h^{\{A,B\}} = 6$

$\Rightarrow$ Any solution spends at least cost 6 on operators modifying A or B.

$$\begin{array}{rlcccccc}
6 & = h^{\{A,B\}} & \leq & \textit{type-}A & + & \textit{type-}B & & \\
6 & = h^{\{A,C\}} & \leq & \textit{type-}A & + & & & \textit{type-}C \\
6 & = h^{\{B,C\}} & \leq & & & \textit{type-}B & + & \textit{type-}C \\
\hline
18 & & \leq & 2\textit{type-}A & + & 2\textit{type-}B & + & 2\textit{type-}C \\
9 & & \leq & \textit{type-}A & + & \textit{type-}B & + & \textit{type-}C
\end{array}$$

$\Rightarrow$ at least cost 9 in any plan

Can we generalize this kind of reasoning?

Post-hoc optimization heuristic: linear program

Construct linear program for pattern collection $\mathcal{C}$:

- Variable X_o for each operator $o \in \mathcal{O}$
    - Cost incurred by operator o in a plan
    - $X_o \geq 0$ for each $o \in \mathcal{O}$

Post-hoc optimization heuristic: linear program

Construct linear program for pattern collection $\mathcal{C}$:

- Variable X_o for each operator $o \in \mathcal{O}$
  - Cost incurred by operator o in a plan
  - $X_o \geq 0$ for each $o \in \mathcal{O}$
- PDB heuristics admissible

$$h^P(s) \leq \sum_{o \in \mathcal{O}} X_o \text{ for each pattern } P \in \mathcal{C}$$

Post-hoc optimization heuristic: linear program

Construct linear program for pattern collection $\mathcal{C}$:

- Variable X_o for each operator $o \in \mathcal{O}$
  - Cost incurred by operator o in a plan
  - $X_o \geq 0$ for each $o \in \mathcal{O}$
- PDB heuristics admissible

$$h^P(s) \leq \sum_{o \in \mathcal{O}} X_o \text{ for each pattern } P \in \mathcal{C}$$

- Can tighten constraints to

$$h^P(s) \leq \sum_{o \in \mathcal{O} : o \text{ affects } P} X_o$$

Post-hoc optimization heuristic: linear program

Construct linear program for pattern collection $\mathcal{C}$:

- Variable X_o for each operator $o \in \mathcal{O}$
  - Cost incurred by operator o in a plan
  - $X_o \geq 0$ for each $o \in \mathcal{O}$
- PDB heuristics admissible

$$h^P(s) \leq \sum_{o \in \mathcal{O}} X_o \text{ for each pattern } P \in \mathcal{C}$$

- Can tighten constraints to

$$h^P(s) \leq \sum_{o \in \mathcal{O}: o \text{ affects } P} X_o$$

- Total cost of the plan is $\sum_{o \in \mathcal{O}} X_o$
- Minimizing total cost leads to admissible estimate

Post-hoc optimization heuristic: definition and admissibility

Definition (Post-hoc optimization heuristic)

The post-hoc optimization heuristic $h^{\mathsf{PhO}}_{\mathcal{C}}$ is the objective value of the linear program for $\mathcal{C}$ as described above.

Post-hoc optimization heuristic: definition and admissibility

Definition (Post-hoc optimization heuristic)

The post-hoc optimization heuristic $h_{\mathcal{C}}^{\mathsf{PhO}}$ is the objective value of the linear program for $\mathcal{C}$ as described above.

Theorem

The post-hoc optimization heuristic is admissible.

Aside: cost partitioning

- Alternative way of using multiple patterns:
  operator cost partitioning
- Account only for a fraction of the actual operator costs
  in each PDB so that estimates can be summed up admissibly

Aside: cost partitioning

- Alternative way of using multiple patterns: operator cost partitioning
- Account only for a fraction of the actual operator costs in each PDB so that estimates can be summed up admissibly

Examples

- Uniform cost partitioning
  - Operator o affects k patterns $\Rightarrow$ each PDB uses cost $c(o)/k$
- Optimal cost partitioning [Katz and Domshlak]
  - Best way to admissibly partition costs
  - State-specific LP

Post-hoc optimization heuristic: insight

Duality Theorem

- Rewrite minimization LP as maximization LP
- Same objective value
- Different view on the same problem

Dual of LP in $h_{\mathcal{C}}^{\text{PhO}}$

- State-specific cost partitioning
- Scales operator costs for heuristic h^P by a factor Y_P
- Much smaller than LP for optimal cost partitioning

Relation to canonical heuristic

Theorem

Consider the dual of the LP solved by $h_{\mathcal{C}}^{\mathsf{PhO}}$ in state s.
If we restrict the variables to integers, the objective value is the canonical heuristic value $h^{\mathcal{C}}(s)$.

Relation to canonical heuristic

Theorem

Consider the dual of the LP solved by $h_{\mathcal{C}}^{\mathsf{PhO}}$ in state s. If we restrict the variables to integers, the objective value is the canonical heuristic value $h^{\mathcal{C}}(s)$.

Theorem

The post-hoc optimization heuristic $h_{\mathcal{C}}^{\mathsf{PhO}}$ dominates the canonical heuristic $h^{\mathcal{C}}$.

Experimental results I

Coverage	iPDB hill-climbing			Systematic (size 2)		
	$h^{\mathcal{C}}$	h^{PhO}	h^{OCP}	$h^{\mathcal{C}}$	h^{PhO}	h^{OCP}
IPC 2011 (280)	133	133	56	126	158	43
IPC 1998–2008 (1116)	456	459	241	446	475	231
Sum (1396)	589	592	297	572	633	274

More detailed results on the poster

Experimental results II

Why is h^{PhO} better than $h^{\mathcal{C}}$?

Additional evaluation on systematic pattern collections:

- Theoretical dominance of h^{PhO}?
- Faster computation?

Experimental results II

Why is h^{PhO} better than $h^{\mathcal{C}}$?

Additional evaluation on systematic pattern collections:

- Theoretical dominance of h^{PhO}?
  - Better guidance on only a few domains
- Faster computation?

Experimental results II

Why is h^{PhO} better than $h^{\mathcal{C}}$?

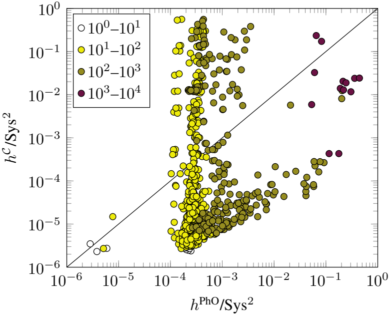
Additional evaluation on systematic pattern collections:

- Theoretical dominance of h^{PhO}?
  - Better guidance on only a few domains
- Faster computation?
  - Considered tasks solved by h^{PhO} but not by $h^{\mathcal{C}}$
  - Most ran out of memory during generation of *MAS*($\mathcal{C}$)
  - On these tasks, $h^{\mathcal{C}}$ would be extremely slow
  - On commonly solved tasks $h^{\mathcal{C}}$ *tends to be faster*

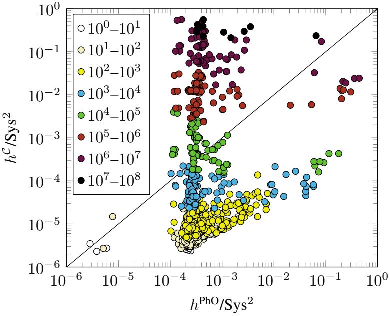
Experimental results III

Time to calculate heuristic value of initial state

Colored by $|\mathcal{C}|$



Colored by $|MAS(\mathcal{C})|$



Conclusion

Two contributions

- Post-hoc optimization heuristic
  - Middle ground between canonical heuristic and optimal cost partitioning
- Systematic generation of interesting patterns
  - Improves over iPDB hill climbing when used with suitable heuristic

Thank you

Thank you for your attention!

Poster presentation

- Friday 8:30 – 9:45

h^{PhO} LP

Minimize $\sum_{o \in \mathcal{O}} X_o$ subject to

$$\sum_{o \in \mathcal{O} : o \text{ affects } P} X_o \geq h^P(s) \qquad \text{for all } P \in \mathcal{C}$$

$$X_o \geq 0 \qquad \text{for all } o \in \mathcal{O}.$$

Corresponding dual program to h^{PhO} LP

Maximize $\sum_{P \in \mathcal{C}} Y_P h^P(s)$ subject to

$$\sum_{P \in \mathcal{C} : o \text{ affects } P} Y_P \leq 1 \qquad \text{for all } o \in \mathcal{O}$$

$$Y_P \geq 0 \qquad \text{for all } P \in \mathcal{C}.$$

Post-hoc optimization heuristic: simplifying the LP

Reduce size of LP

- Aggregate variables which always occur together in constraints
- Happens when several operators are relevant for exactly the same PDBs
- Merge individual variables into one new variable
  - Represents their sum

Example task

Merged all operators modifying A into variable *type*-A

Detailed experimental results I

	$\mathrm{HC}^{\mathcal{C}}$			$\mathrm{HC}^{\mathrm{PhO}}$	Sys^2		
	$h^{\mathcal{C}}$	h^{PhO}	h^{OCP}	h^{PhO}	$h^{\mathcal{C}}$	h^{PhO}	h^{OCP}
barman (20)	4	4	0	0	4	4	0
elevators (20)	16	16	0	16	16	15	0
floortile (20)	2	2	0	2	2	2	0
nomystery (20)	16	16	3	16	18	18	6
openstacks (20)	14	14	5	14	5	14	0
parcprinter (20)	8	8	8	8	7	13	15
parking (20)	5	5	1	5	0	1	0
pegsol (20)	0	0	0	0	5	17	1
scanalyzer (20)	10	10	1	7	10	8	1
sokoban (20)	20	20	18	20	20	20	2
tidybot (20)	14	14	6	11	14	14	6
transport (20)	6	6	2	6	6	6	0
visitall (20)	16	16	10	16	16	16	10
woodworking (20)	2	2	2	1	3	10	2
Sum IPC 2011 (280)	133	133	56	122	126	158	43
IPC 1998–2008 (1116)	456	459	241	426	446	475	231
Sum (1396)	589	592	297	548	572	633	274

Detailed experimental results II

	Sys^1		Sys^2		Sys^3		Sys^*
	$h^{\mathcal{C}}$	h^{PhO}	$h^{\mathcal{C}}$	h^{PhO}	$h^{\mathcal{C}}$	h^{PhO}	h^{PhO}
barman (20)	4	4	4	4	0	0	4 (1–2)
elevators (20)	9	9	16	15	16	14	15 (2)
floortile (20)	2	2	2	2	2	2	2 (1–3)
nomystery (20)	12	12	18	18	19	19	19 (3–4)
openstacks (20)	14	14	5	14	2	9	14 (1–2)
parcprinter (20)	11	11	7	13	5	18	20 (4)
parking (20)	5	5	0	1	0	0	5 (1)
pegsol (20)	17	17	5	17	1	16	17 (1–2)
scanalyzer (20)	10	10	10	8	10	4	10 (1)
sokoban (20)	19	19	20	20	20	13	20 (2)
tidybot (20)	13	13	14	14	8	14	14 (2–3)
transport (20)	6	6	6	6	9	8	8 (3)
visitall (20)	16	16	16	16	16	16	16 (1–3)
woodworking (20)	5	5	3	10	1	9	10 (2)
Sum IPC 2011 (280)	143	143	126	158	109	142	174
IPC 1998–2008 (1116)	449	449	446	475	406	422	501
Sum (1396)	592	592	572	633	515	564	675