Automated Planning

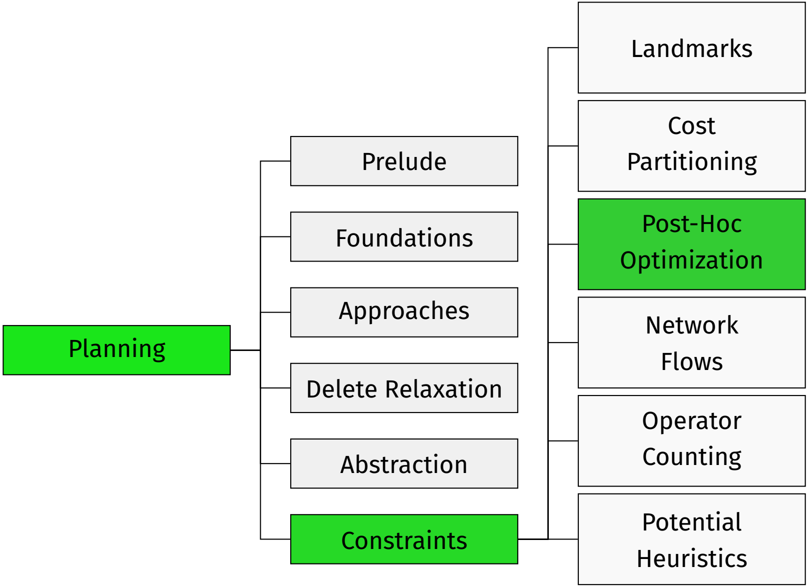
F8. Post-hoc Optimization

Jendrik Seipp

Linköping University

based on slides from the AI group at the University of Basel

Content of this Course



Introduction

Example Task (1)

Example (Example Task)

SAS$^+$ task $\Pi = \langle V, I, O, \gamma \rangle$ with

- $V = \{A, B, C\}$ with $\mathrm{dom}(v) = \{0, 1, 2, 3, 4\}$ for all $v \in V$
- $I = \{A \mapsto 0, B \mapsto 0, C \mapsto 0\}$
- $O = \{inc_x^v \mid v \in V, x \in \{0, 1, 2\}\} \cup \{jump^v \mid v \in V\}$
  - $inc_x^v = \langle v = x, v := x + 1, 1 \rangle$
  - $jump^v = \langle \bigwedge_{v' \in V : v' \neq v} v' = 4, v := 3, 1 \rangle$
- $\gamma = A = 3 \wedge B = 3 \wedge C = 3$

- Each optimal plan consists of three increment operators for each variable $\rightsquigarrow h^*(I) = 9$
- Each operator affects only one variable.

Example Task (2)

- In projections to single variables we can reach the goal with a *jump* operator: $h^{\{A\}}(I) = h^{\{B\}}(I) = h^{\{C\}}(I) = 1$.
- In projections to more variables, we need for each variable three applications of increment operators to reach the abstract goal from the abstract initial state: $h^{\{A,B\}}(I) = h^{\{A,C\}}(I) = h^{\{B,C\}}(I) = 6$

Example (Canonical Heuristic, using orthogonality)

$\mathcal{C} = \{\{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}, \{B, C\}\}$

$$h^{\mathcal{C}}(s) = \max\{h^{\{A\}}(s) + h^{\{B\}}(s) + h^{\{C\}}(s), h^{\{A\}}(s) + h^{\{B,C\}}(s), h^{\{B\}}(s) + h^{\{A,C\}}(s), h^{\{C\}}(s) + h^{\{A,B\}}(s)\}$$

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

Post-hoc Optimization Heuristic: Idea

Consider the example task:

- *type-v* operator: operator modifying variable v

Post-hoc Optimization Heuristic: Idea

Consider the example task:

- *type-v* operator: operator modifying variable v
- $h^{\{A,B\}} = 6$

  $\Longrightarrow$ in any plan operators of type A or B incur at least cost 6.

Post-hoc Optimization Heuristic: Idea

Consider the example task:

- *type-v* operator: operator modifying variable *v*
- $h^{\{A,B\}} = 6$
  $\Rightarrow$ in any plan operators of type A or B incur at least cost 6.
- $h^{\{A,C\}} = 6$
  $\Rightarrow$ in any plan operators of type A or C incur at least cost 6.
- $h^{\{B,C\}} = 6$
  $\Rightarrow$ in any plan operators of type B or C incur at least cost 6.

Post-hoc Optimization Heuristic: Idea

Consider the example task:

- *type-v* operator: operator modifying variable v
- $h^{\{A,B\}} = 6$

  $\Rightarrow$ in any plan operators of type A or B incur at least cost 6.
- $h^{\{A,C\}} = 6$

  $\Rightarrow$ in any plan operators of type A or C incur at least cost 6.
- $h^{\{B,C\}} = 6$

  $\Rightarrow$ in any plan operators of type B or C incur at least cost 6.
- $\Rightarrow$ any plan has at least cost ???.

Post-hoc Optimization Heuristic: Idea

Consider the example task:

- *type-v* operator: operator modifying variable v
- $h^{\{A,B\}} = 6$

  $\Rightarrow$ in any plan operators of type A or B incur at least cost 6.
- $h^{\{A,C\}} = 6$

  $\Rightarrow$ in any plan operators of type A or C incur at least cost 6.
- $h^{\{B,C\}} = 6$

  $\Rightarrow$ in any plan operators of type B or C incur at least cost 6.
- $\Rightarrow$ any plan has at least cost ???.
- (let’s use linear programming…)

Post-hoc Optimization Heuristic: Idea

Consider the example task:

- *type-v* operator: operator modifying variable v
- $h^{\{A,B\}} = 6$
  $\Rightarrow$ in any plan operators of type A or B incur at least cost 6.
- $h^{\{A,C\}} = 6$
  $\Rightarrow$ in any plan operators of type A or C incur at least cost 6.
- $h^{\{B,C\}} = 6$
  $\Rightarrow$ in any plan operators of type B or C incur at least cost 6.
- $\Rightarrow$ any plan has at least cost ???.
- (let's use linear programming...)
- $\Rightarrow$ any plan has at least cost 9.

Post-hoc Optimization Heuristic: Idea

Consider the example task:

- *type-v* operator: operator modifying variable v
- $h^{\{A,B\}} = 6$
  $\Rightarrow$ in any plan operators of type A or B incur at least cost 6.
- $h^{\{A,C\}} = 6$
  $\Rightarrow$ in any plan operators of type A or C incur at least cost 6.
- $h^{\{B,C\}} = 6$
  $\Rightarrow$ in any plan operators of type B or C incur at least cost 6.
- $\Rightarrow$ any plan has at least cost ???.
- (let's use linear programming…)
- $\Rightarrow$ any plan has at least cost 9.

Can we generalize this kind of reasoning?

Post-hoc Optimization

Post-hoc Optimization

The heuristic that generalizes this kind of reasoning is the Post-hoc Optimization Heuristic (PhO)

- can be computed for any kind of heuristic . . .
- . . . as long as we are able to determine relevance of operators
- if in doubt, it's always safe to assume an operator is relevant for a heuristic
- but for PhO to work well, it's important that the set of relevant operators is as small as possible

Operator Relevance in Abstractions

Definition (Reminder: Affecting Transition Labels)

Let $\mathcal{T}$ be a transition system, and let ℓ be one of its labels.

We say that ℓ affects $\mathcal{T}$ if $\mathcal{T}$ has a transition $s \xrightarrow{\ell} t$ with $s \neq t$.

Definition (Operator Relevance in Abstractions)

An operator o is relevant for an abstraction α if o affects $\mathcal{T}^\alpha$.

We can efficiently determine operator relevance for abstractions.

Linear Program (1)

For a given set of abstractions $\{\alpha_1, \ldots, \alpha_n\}$, we construct a linear program:

- variable X_o for each operator $o \in O$
- intuitively, X_o is cost incurred by operator o
- abstraction heuristics are admissible

$$\sum_{o \in O} X_o \geq h^{\alpha}(s) \quad \text{for } \alpha \in \{\alpha_1, \ldots, \alpha_n\}$$

- can tighten these constraints to

$$\sum_{o \in O : o \text{ relevant for } \alpha} X_o \geq h^{\alpha}(s) \quad \text{for } \alpha \in \{\alpha_1, \ldots, \alpha_n\}$$

Linear Program (2)

For set of abstractions $\{\alpha_1, \ldots, \alpha_n\}$:

Variables

Non-negative variables X_o for all operators $o \in O$

Objective

Minimize $\sum_{o \in O} X_o$

Subject to

$$\begin{aligned} \sum_{o \in O : o \text{ relevant for } \alpha} X_o &\geq h^\alpha(s) && \text{for } \alpha \in \{\alpha_1, \ldots, \alpha_n\} \\ X_o &\geq 0 && \text{for all } o \in O \end{aligned}$$

PhO Heuristic

Definition (Post-hoc Optimization Heuristic)

The post-hoc optimization heuristic $h^{\mathrm{PhO}}_{\{\alpha_1,\ldots,\alpha_n\}}$ for abstractions $\alpha_1, \ldots, \alpha_n$ is the objective value of the following linear program:

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

$$\begin{aligned} \sum\nolimits_{o \in O : o \text{ relevant for } \alpha} X_o &\geq h^{\alpha}(s) && \text{for all } \alpha \in \{\alpha_1, \ldots, \alpha_n\} \\ X_o &\geq 0 && \text{for all } o \in O \end{aligned}$$

PhO Heuristic

h^{PhO}

1. Precompute all abstraction heuristics $h^{\alpha_1}, \ldots, h^{\alpha_n}$.
2. Create LP for initial state s_0.
3. For each new state s:
   - Look up $h^{\alpha}(s)$ for all $\alpha \in \{\alpha_1, \ldots, \alpha_n\}$.
   - Adjust LP by replacing bounds with the $h^{\alpha}(s)$ values.

Theorem (Admissibility)

The post-hoc optimization heuristic is admissible.

Combining Estimates from Abstraction Heuristics

- Post-Hoc optimization combines multiple admissible heuristic estimates into one.

Combining Estimates from Abstraction Heuristics

- Post-Hoc optimization combines multiple admissible heuristic estimates into one.
- We have already heard of two other such approaches for abstraction heuristics,
  - optimal cost partitioning and
  - the canonical heuristic for PDBs (both not covered in detail).

Combining Estimates from Abstraction Heuristics

- Post-Hoc optimization combines multiple admissible heuristic estimates into one.
- We have already heard of two other such approaches for abstraction heuristics,
  - optimal cost partitioning and
  - the canonical heuristic for PDBs (both not covered in detail).
- How does PhO compare to these?

PhO vs. OCP

What about Optimal Cost Partitioning for Abstractions?

Optimal cost partitioning for abstractions…

- …uses a state-specific LP to find the best possible cost partitioning, and sums up the heuristic estimates.
- …dominates the canonical heuristic, i.e., for the same pattern collection, it never gives lower estimates than h^C.
- …is very expensive to compute
(recomputing all abstract goal distances in every state).

PhO: Linear Program

For set of abstractions $\{\alpha_1, \ldots, \alpha_n\}$:

Variables

X_o for all equivalence classes $o \in O$

Objective

Minimize $\sum_{o \in O} X_o$

Subject to

$$\sum_{o \in O : o \text{ relevant for } \alpha} X_o \geq h^{\alpha}(s) \quad \text{for all } \alpha \in \{\alpha_1, \ldots, \alpha_n\}$$

$$X_o \geq 0 \quad \text{for all } o \in O$$

PhO: Dual Linear Program

For set of abstractions $\{\alpha_1, \dots, \alpha_n\}$:

Variables

Y_α for each abstraction $\alpha \in \{\alpha_1, \dots, \alpha_n\}$

Objective

Maximize $\sum_{\alpha \in \{\alpha_1,\dots,\alpha_n\}} h^\alpha(s) Y_\alpha$

Subject to

$$\begin{aligned} \sum_{\alpha \in \{\alpha_1,\dots,\alpha_n\} : o \text{ relevant for } \alpha} Y_\alpha &\leq 1 \quad \text{for all } o \in O \\ Y_\alpha &\geq 0 \quad \text{for all } \alpha \in \{\alpha_1, \dots, \alpha_n\} \end{aligned}$$

PhO: Dual Linear Program

For set of abstractions $\{\alpha_1, \dots, \alpha_n\}$:

Variables

Y_α for each abstraction $\alpha \in \{\alpha_1, \dots, \alpha_n\}$

Objective

Maximize $\sum_{\alpha \in \{\alpha_1,\dots,\alpha_n\}} h^\alpha(s) Y_\alpha$

Subject to

$$\sum_{\alpha \in \{\alpha_1,\dots,\alpha_n\} : o \text{ relevant for } \alpha} Y_\alpha \leq 1 \quad \text{for all } o \in O$$

$$Y_\alpha \geq 0 \quad \text{for all } \alpha \in \{\alpha_1, \dots, \alpha_n\}$$

We compute a state-specific cost partitioning that can only scale the operator costs within each heuristic by a factor $0 \leq Y_\alpha \leq 1$.

Relation to Optimal Cost Partitioning

Theorem

Optimal cost partitioning dominates post-hoc optimization.

Proof Sketch.

Consider a feasible assignment $\langle Y_{\alpha_1}, \ldots, Y_{\alpha_n} \rangle$ for the variables of the dual LP for PhO.

Its objective value is equivalent to the cost-partitioning heuristic for the same abstractions with cost partitioning $\langle Y_{\alpha_1} cost, \ldots, Y_{\alpha_n} cost \rangle$.

Canonical Heuristic

Canonical Heuristic: Finding Additive Pattern Sets

Theorem (Additive Pattern Sets)

Let $P_1, \dots, P_k$ be disjoint patterns for an FDR planning task Π.

If there exists no operator that has an effect on a variable $v_i \in P_i$ and on a variable $v_j \in P_j$ for some $i \neq j$, then $\sum_{i=1}^{k} h^{P_i}$ is an admissible and consistent heuristic for Π.

This theorem gives us a simple criterion to decide which pattern heuristics can be admissibly added.

Given a pattern collection $\mathcal{C}$ (i.e., a set of patterns), we can use this information as follows:

1. Build the compatibility graph for $\mathcal{C}$.
   - Vertices correspond to patterns $P \in \mathcal{C}$.
   - There is an edge between two vertices iff no operator affects both incident patterns.
2. Compute all maximal cliques of the graph. These correspond to maximal additive subsets of $\mathcal{C}$.

The Canonical Heuristic Function

Definition (Canonical Heuristic Function)

Let $\mathcal{C}$ be a pattern collection for an FDR planning task.

The **canonical heuristic $h^{\mathcal{C}}$** for pattern collection $\mathcal{C}$ is defined as

$$h^{\mathcal{C}}(s) = \max_{\mathcal{D} \in cliques(\mathcal{C})} \sum_{P \in \mathcal{D}} h^{P}(s),$$

where $cliques(\mathcal{C})$ is the set of all maximal cliques in the compatibility graph for $\mathcal{C}$.

For all choices of $\mathcal{C}$, heuristic $h^{\mathcal{C}}$ is admissible and consistent. It is also the best possible admissible heuristic not using cost partitioning.

Canonical Heuristic: Example

Example

Consider a planning task with state variables $V = \{v_1, \dots, v_5\}$ and the pattern collection $\mathcal{C} = \{P_1, \dots, P_5\}$ with $P_1 = \{v_1, v_2, v_3\}$, $P_2 = \{v_1, v_2\}$, $P_3 = \{v_3\}$, $P_4 = \{v_4\}$ and $P_5 = \{v_5\}$.

There are operators affecting each individual variable, variables v_1 and v_2, variables v_3 and v_4 and variables v_3 and v_5.

What is the compatibility graph for $\mathcal{C}$?

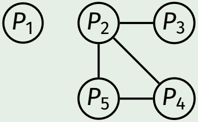
Canonical Heuristic: Example

Example

Consider a planning task with state variables $V = \{v_1, \ldots, v_5\}$ and the pattern collection $\mathcal{C} = \{P_1, \ldots, P_5\}$ with $P_1 = \{v_1, v_2, v_3\}$, $P_2 = \{v_1, v_2\}$, $P_3 = \{v_3\}$, $P_4 = \{v_4\}$ and $P_5 = \{v_5\}$.

There are operators affecting each individual variable, variables v_1 and v_2, variables v_3 and v_4 and variables v_3 and v_5.

What is the compatibility graph for $\mathcal{C}$? Answer:



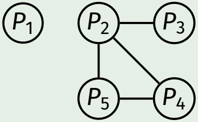
Canonical Heuristic: Example

Example

Consider a planning task with state variables $V = \{v_1, \ldots, v_5\}$ and the pattern collection $\mathcal{C} = \{P_1, \ldots, P_5\}$ with $P_1 = \{v_1, v_2, v_3\}$, $P_2 = \{v_1, v_2\}$, $P_3 = \{v_3\}$, $P_4 = \{v_4\}$ and $P_5 = \{v_5\}$.

There are operators affecting each individual variable, variables v_1 and v_2, variables v_3 and v_4 and variables v_3 and v_5.

What is the compatibility graph for $\mathcal{C}$? Answer:



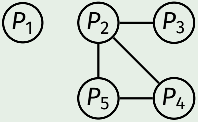
What are the maximal cliques in the compatibility graph for $\mathcal{C}$?

Canonical Heuristic: Example

Example

Consider a planning task with state variables $V = \{v_1, \ldots, v_5\}$ and the pattern collection $\mathcal{C} = \{P_1, \ldots, P_5\}$ with $P_1 = \{v_1, v_2, v_3\}$, $P_2 = \{v_1, v_2\}$, $P_3 = \{v_3\}$, $P_4 = \{v_4\}$ and $P_5 = \{v_5\}$.

There are operators affecting each individual variable, variables v_1 and v_2, variables v_3 and v_4 and variables v_3 and v_5.



What is the compatibility graph for $\mathcal{C}$? Answer:

What are the maximal cliques in the compatibility graph for $\mathcal{C}$?
Answer: $\{P_1\}$, $\{P_2, P_3\}$, $\{P_2, P_4, P_5\}$

What is the canonical heuristic function $h^{\mathcal{C}}$?

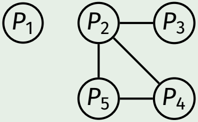
Canonical Heuristic: Example

Example

Consider a planning task with state variables $V = \{v_1, \ldots, v_5\}$ and the pattern collection $\mathcal{C} = \{P_1, \ldots, P_5\}$ with $P_1 = \{v_1, v_2, v_3\}$, $P_2 = \{v_1, v_2\}$, $P_3 = \{v_3\}$, $P_4 = \{v_4\}$ and $P_5 = \{v_5\}$.

There are operators affecting each individual variable, variables v_1 and v_2, variables v_3 and v_4 and variables v_3 and v_5.

What is the compatibility graph for $\mathcal{C}$? Answer:



What are the maximal cliques in the compatibility graph for $\mathcal{C}$?
Answer: $\{P_1\}$, $\{P_2, P_3\}$, $\{P_2, P_4, P_5\}$

What is the canonical heuristic function $h^{\mathcal{C}}$?
Answer: $h^{\mathcal{C}} = \max\{h^{P_1}, h^{P_2} + h^{P_3}, h^{P_2} + h^{P_4} + h^{P_5}\}$

PhO vs. Canonical Heuristic

Relation to Canonical Heuristic

Theorem

Consider the dual D of the LP solved by the post-hoc optimization heuristic in state s for a given set of abstractions. If we restrict the variables in D to integers, the objective value is the canonical heuristic value $h^{C}(s)$.

Relation to Canonical Heuristic

Theorem

Consider the dual D of the LP solved by the post-hoc optimization heuristic in state s for a given set of abstractions. If we restrict the variables in D to integers, the objective value is the canonical heuristic value $h^C(s)$.

Corollary

The post-hoc optimization heuristic dominates the canonical heuristic for the same set of abstractions.

h^{PhO} vs h^{C}

- For the canonical heuristic, we need to find all maximal cliques, which is an NP-hard problem.
- The post-hoc optimization heuristic dominates the canonical heuristic and can be computed in polynomial time.
- The post-hoc optimization heuristic solves an LP in each state.
- With post-hoc optimization, a large number of small patterns works well.

Summary

Summary

- Post-hoc optimization heuristic constraints express admissibility of heuristics
- exploits (ir-)relevance of operators for heuristics
- explores the middle ground between canonical heuristic and optimal cost partitioning.
- For the same set of abstractions, the post-hoc optimization heuristic dominates the canonical heuristic.
- The computation can be done in polynomial time.