# Planning and Optimization

## G9. Post-hoc Optimization

Malte Helmert and Gabriele Röger

Universität Basel

December 13, 2023



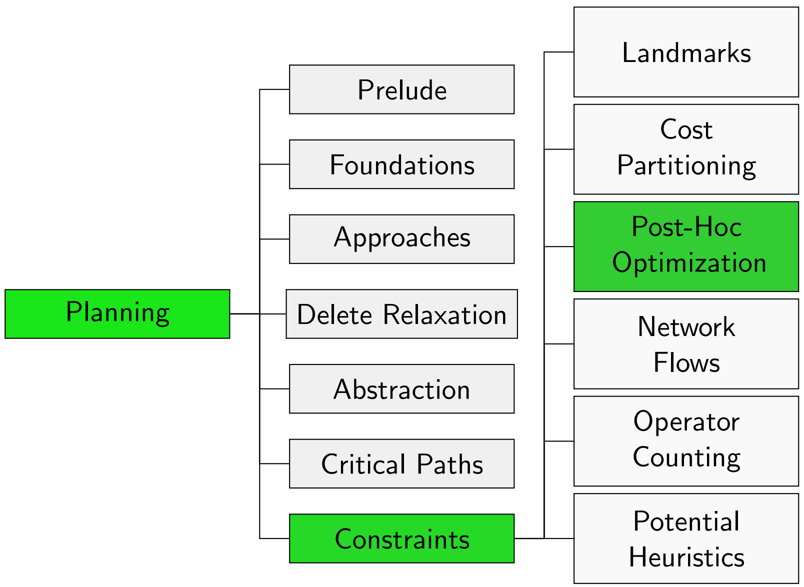
# Planning and Optimization

December 13, 2023 — G9. Post-hoc Optimization





# Content of this Course








# G9.1 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 on single variables we can reach the goal with a *jump* operator: $h^{\{A\}}(I) = h^{\{B\}}(I) = h^{\{C\}}(I) = 1$.
- In projections on 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)**

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

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

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





## Post-hoc Optimization Heuristic: Idea

Consider the example task:

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





# G9.2 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 $\ell$ be one of its labels.

We say that $\ell$ 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 $\alpha$ if $o$ affects $\mathcal{T}^\alpha$.

We can efficiently determine operator relevance for abstractions.





## Linear Program (1)

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

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

- can tighten these constraints to

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





## Linear Program (2)

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

**Variables**

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

**Objective**

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

**Subject to**

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

## Simplifying the LP

- Reduce the size of the LP by aggregating variables which always occur together in constraints.
- Happens if several operators are relevant for exactly the same heuristics.
- Partitioning $O/\sim$ induced by this equivalence relation
- One variable $X_{[o]}$ for each $[o] \in O/\sim$





## Example

**Example**

- only operators $o_1, o_2, o_3$ and $o_4$ are relevant for $h_1$ and $h_1(s_0) = 11$
- only operators $o_3, o_4, o_5$ and $o_6$ are relevant for $h_2$ and $h_2(s_0) = 11$
- only operators $o_1, o_2$ and $o_6$ are relevant for $h_3$ and $h_3(s_0) = 8$

Which operators are relevant for exactly the same heuristics?
What is the resulting partitioning?

Answer: $o_1 \sim o_2$ and $o_3 \sim o_4$
$\Rightarrow O/\sim = \{[o_1], [o_3], [o_5], [o_6]\}$





## Simplifying the LP: Example

LP before aggregation

**Variables**
Non-negative variable $X_1, \dots, X_6$
for operators $o_1, \dots, o_6$

Minimize $X_1 + X_2 + X_3 + X_4 + X_5 + X_6$ subject to

$$
\begin{aligned}
X_1 + X_2 + X_3 + X_4 &\geq 11 \\
X_3 + X_4 + X_5 + X_6 &\geq 11 \\
X_1 + X_2 + X_6 &\geq 8 \\
X_i &\geq 0 \quad \text{for } i \in \{1, \dots, 6\}
\end{aligned}
$$





## Simplifying the LP: Example

LP after aggregation

**Variables**
Non-negative variable $X_{[1]}, X_{[3]}, X_{[5]}, X_{[6]}$
for equivalence classes $[o_1], [o_3], [o_5], [o_6]$

Minimize $X_{[1]} + X_{[3]} + X_{[5]} + X_{[6]}$ subject to

$$
\begin{aligned}
X_{[1]} + X_{[3]} &\geq 11 \\
X_{[3]} + X_{[5]} + X_{[6]} &\geq 11 \\
X_{[1]} + X_{[6]} &\geq 8 \\
X_i &\geq 0 \quad \text{for } i \in \{[1], [3], [5], [6]\}
\end{aligned}
$$

# PhO Heuristic

**Definition (Post-hoc Optimization Heuristic)**

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

$$\text{Minimize} \sum_{[o] \in O/\sim} X_{[o]} \text{ subject to}$$

$$\sum_{[o] \in O/\sim : o \text{ relevant for } \alpha} X_{[o]} \geq h^{\alpha}(s) \quad \text{for all } \alpha \in \{\alpha_1, \dots, \alpha_n\}$$

$$X_{[o]} \geq 0 \quad \text{for all } [o] \in O/\sim,$$

where $o \sim o'$ iff $o$ and $o'$ are relevant for exactly the same abstractions in $\alpha_1, \dots, \alpha_n$.





# PhO Heuristic

**$h^{\mathrm{PhO}}$**

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





# Post-hoc Optimization Heuristic: Admissibility

**Theorem (Admissibility)**

*The post-hoc optimization heuristic is admissible.*

**Proof.**

Let $\Pi$ be a planning task and $\{\alpha_1, \dots, \alpha_n\}$ be a set of abstractions. We show that there is a feasible variable assignment with objective value equal to the cost of an optimal plan.
Let $\pi$ be an optimal plan for state $s$ and let $cost_\pi(O')$ be the cost incurred by operators from $O' \subseteq O$ in $\pi$.

Setting each $X_{[o]}$ to $cost_\pi([o])$ is a feasible variable assignment: Constraints $X_{[o]} \geq 0$ are satisfied. ...





# Post-hoc Optimization Heuristic: Admissibility

**Theorem (Admissibility)**

*The post-hoc optimization heuristic is admissible.*

**Proof (continued).**

For each $\alpha \in \{\alpha_1, \dots, \alpha_n\}$, $\pi$ is a solution in the abstract transition system and the sum in the corresponding constraint equals the cost of the state-changing abstract state transitions (i.e.. not accounting for self-loops). As $h^{\alpha}(s)$ corresponds to the cost of an optimal solution in the abstraction, the inequality holds.

For this assignment, the objective function has value $h^*(s)$ (cost of $\pi$), so the objective value of the LP is admissible. □

# G9.3 Comparison





## 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,
  - the canonical heuristic (for PDBs), and
  - optimal cost partitioning (not covered in detail).
- How does PhO compare to these?





## Reminder: The Canonical Heuristic Function

If for a set of patterns no operator affects more than one pattern, the sum of the heuristic estimates is admissible.

**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

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

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

For a given pattern collection, the canonical heuristic is the best possible admissible heuristic not using cost partitioning.





## 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 $h^{\mathcal{C}}$.
- ...is very expensive to compute (recomputing all abstract goal distances in every state).

## PhO: Linear Program

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

**Variables**

$X_{[o]}$ for all equivalence classes $[o] \in O/\sim$

**Objective**

Minimize $\sum_{[o] \in O/\sim} X_{[o]}$

**Subject to**

$$\sum_{[o] \in O/\sim : o \text{ relevant for } \alpha} X_{[o]} \geq h^\alpha(s) \quad \text{for all } \alpha \in \{\alpha_1, \dots, \alpha_n\}$$

$$X_{[o]} \geq 0 \quad \text{for all } [o] \in O/\sim$$





## PhO: Dual Linear Program

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

**Variables**

$Y_\alpha$ 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**

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

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

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 $\langle Y_{\alpha_1}, \dots, Y_{\alpha_n} \rangle$ 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 $\langle Y_{\alpha_1} cost, \dots, Y_{\alpha_n} cost \rangle$.





## 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 $h^{\mathcal{C}}(s)$.*

**Corollary**

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

## $h^{\mathrm{PhO}}$ vs $h^{\mathcal{C}}$

- 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.





# G9.4 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.