Representing Perfect Saturated Cost Partitioning Heuristics in Classical Planning

Paul Höft[1], David Speck[2], Jendrik Seipp[1]
Acknowledgments: Thomas Keller
[1]Linköping University, [2]University of Basel

November 15, 2025

Background – Cost Partitioning

Problem

Given set of abstraction heuristics $\mathcal{H} = \{h_1, \ldots, h_n\}$
Find strong *admissible* combination of heuristics

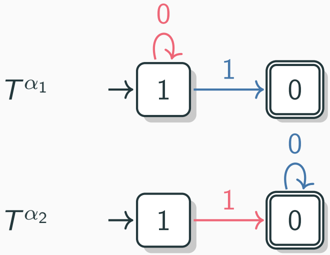
One of the answers is **Cost Partitioning**

Background – Cost Partitioning

Cost function: $cost : L \to \mathbb{R} \cup \{-\infty, \infty\}$

Cost partition for $\mathcal{H}$: $\mathcal{C} = \langle cost_1, \ldots, cost_n \rangle$
with $\sum_{i=1}^{n} h_i(cost_i(\ell) \leq cost(\ell)$ for all $\ell \in L$

$h^{\mathcal{C}} = \sum_{i=1}^{n} h_i(cost_i, s)$ is *admissible*



$cost = \langle 1, 1 \rangle$

Background – Saturated Cost Partitioning

Greedy cost partitioning strategy

Given order $\omega = \langle h_1, \ldots, h_n \rangle$, compute $\langle cost_1, \ldots, cost_n \rangle$

$$
\begin{aligned}
remain_0 &= cost \\
cost_i &= saturate(remain_{i-1}) && \text{for all } 1 \leq i \leq n \\
remain_i &= remain_{i-1} - cost_i && \text{for all } 1 \leq i \leq n
\end{aligned}
$$

Saturated Cost Partitioning

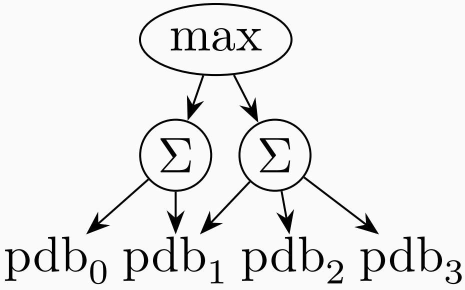
- Best order might differ for every state
- All orders are too many to compute (factorial)
- Best approach optimizes a few orders for sampled states

Saturated Cost Partitioning

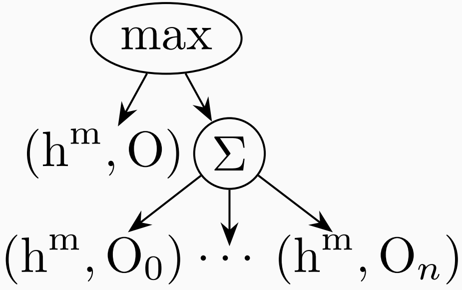
- Best order might differ for every state
- All orders are too many to compute (factorial)
- Best approach optimizes a few orders for sampled states

How can we compute h_*^{SCP} more efficient?
How strong is it?

Additive-Disjunctive Heuristic Graphs (ADHG)[1]



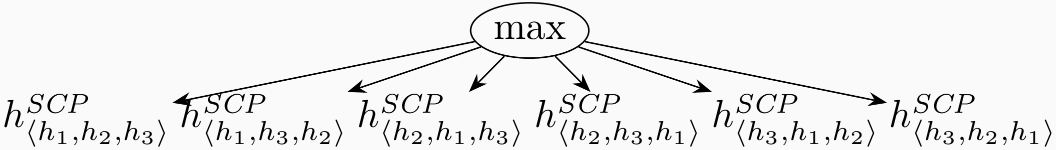
Canonical Heuristic



Additive h^m

[1] Coles et al., "Additive-Disjunctive Heuristics for Optimal Planning".

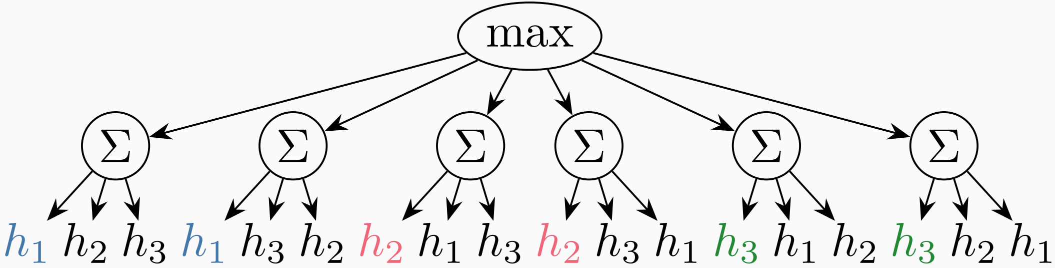
Saturated Cost Partitioning as ADHG



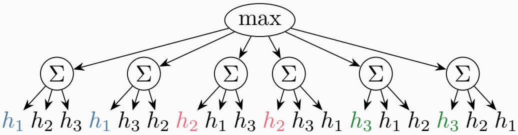
Naive h^{SCP}_{*}

$n \times n!$ Lookup tables
$\rightarrow$ Out of Memory (8G) for **10** Abstractions

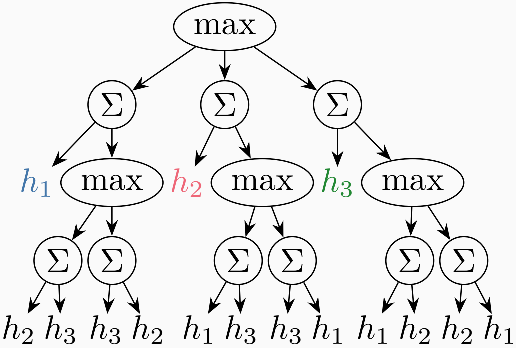
Saturated Cost Partitioning as ADHG



Saturated Cost Partitioning as ADHG

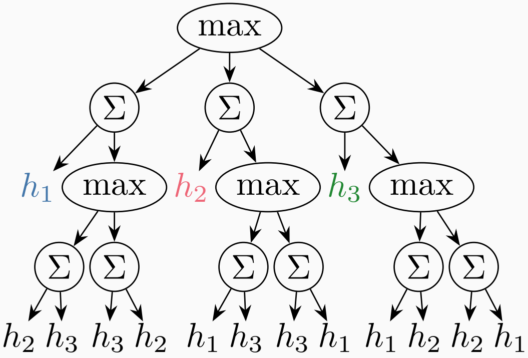


Naive computation



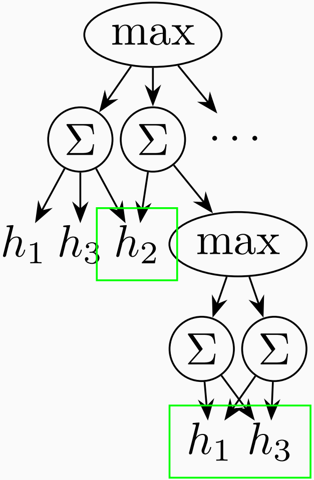
Optimized computation graph

Saturated Cost Partitioning as ADHG



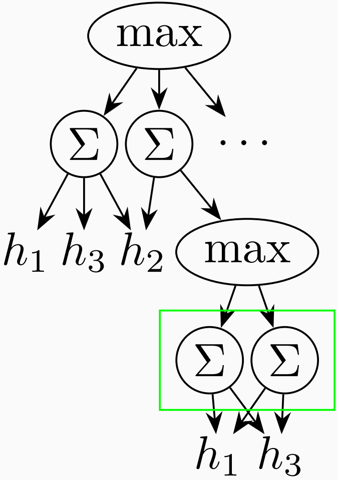
Lookup tables: ~~$n \times n!$~~ $\sum_{k=0}^{n-1} \frac{n!}{k!} \sim e \times n!$

ADHG Reduction Rules



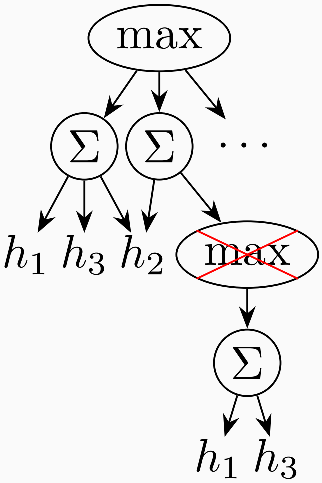
Store only unique **lookup tables**

ADHG Reduction Rules



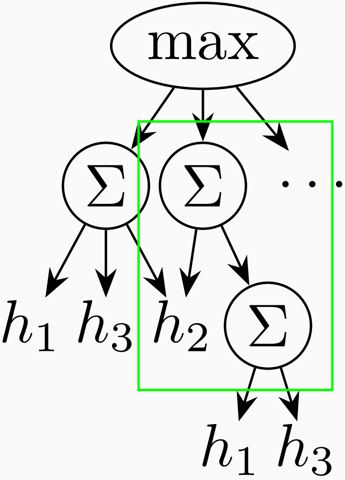
Store only unique **nodes**

ADHG Reduction Rules



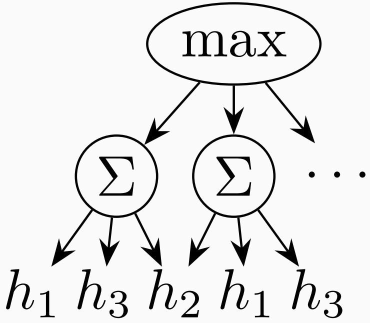
Nodes with a single child can be removed

ADHG Reduction Rules

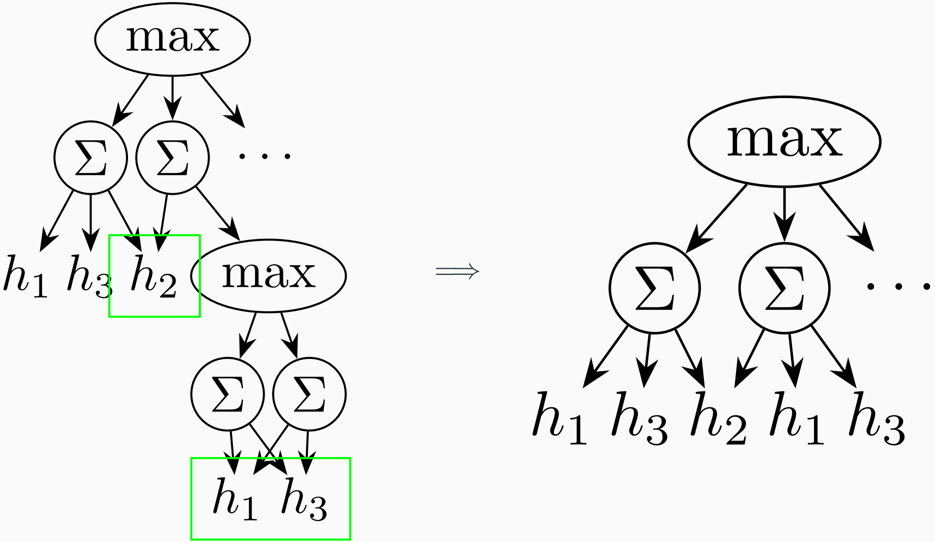


Consecutive same type nodes can be merged

ADHG Reduction Rules



ADHG Reduction Rules



All Reductions: Computable for up to **50** Abstractions

SCP Order Independence

Two orders can produce the same **heuristic**

SCP Heuristic Equivalence

$$h^{\mathrm{SCP}}_{\omega}(s) = h^{\mathrm{SCP}}_{\omega'}(s) \text{ for all } s \in S$$

Hard to test

SCP Order Independence

Two orders can produce the same **cost partition**

SCP Cost Partition Equivalence

$$\mathcal{C}_{\omega}^{\mathsf{SCP}}(h_i) = \mathcal{C}_{\omega'}^{\mathsf{SCP}}(h_i) \text{ for all } 1 \leq i \leq n$$

Easy to test

sufficient for heuristics equivalence

SCP Order Independence

For two heuristics h_1, h_2

$$\mathcal{C}^{\text{SCP}}_{\langle h_1, h_2 \rangle}(h_i) = \mathcal{C}^{\text{SCP}}_{\langle h_2, h_1 \rangle}(h_i) \text{ for all } 1 \leq i \leq n$$

$\Rightarrow$

$$cost - mscf_2 \leq cost$$
$$cost - mscf_2 \geq mscf_1$$

SCP Order Independence

For two heuristics h_1, h_2

$$\mathcal{C}^{\text{SCP}}_{\langle h_1, h_2 \rangle}(h_i) = \mathcal{C}^{\text{SCP}}_{\langle h_2, h_1 \rangle}(h_i) \text{ for all } 1 \leq i \leq n$$

$\Rightarrow$

$cost - mscf_2 \leq cost$ **non-negative** minimum saturated costs

$cost - mscf_2 \geq mscf_1$

SCP Order Independence

For two heuristics h_1, h_2

$$\mathcal{C}^{\mathsf{SCP}}_{\langle h_1,h_2\rangle}(h_i) = \mathcal{C}^{\mathsf{SCP}}_{\langle h_2,h_1\rangle}(h_i) \text{ for all } 1 \leq i \leq n$$

$\Rightarrow$

$cost - mscf_2 \leq cost$	**non-negative** minimum saturated costs
$cost - mscf_2 \geq mscf_1$	minimum saturated costs form a **cost partition**

SCP Order Independence

For two heuristics h_1, h_2

$$\mathcal{C}^{\text{SCP}}_{\langle h_1, h_2 \rangle}(h_i) = \mathcal{C}^{\text{SCP}}_{\langle h_2, h_1 \rangle}(h_i) \text{ for all } 1 \leq i \leq n$$

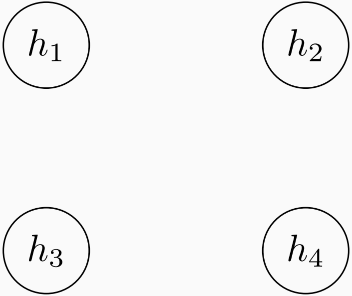
$\Rightarrow$

$cost - mscf_2 \leq cost$	**non-negative** minimum saturated costs
$cost - mscf_2 \geq mscf_1$	minimum saturated costs form a **cost partition**

Conditions need saturated costs $\rightarrow$ approximations necessary

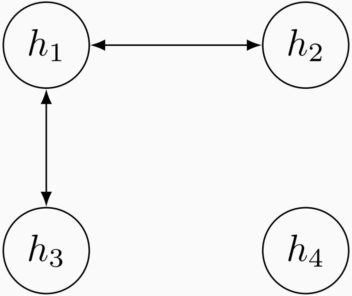
SCP Order Independence

Order-independent Sets are computed through Connected Components:



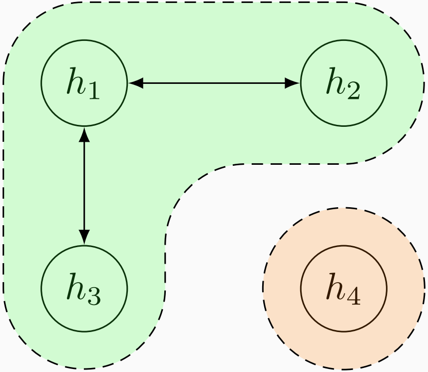
SCP Order Independence

Order-independent Sets are computed through Connected Components:



SCP Order Independence

Order-independent Sets are computed through Connected Components:



SCP Order Independence

Order-Independent Sets

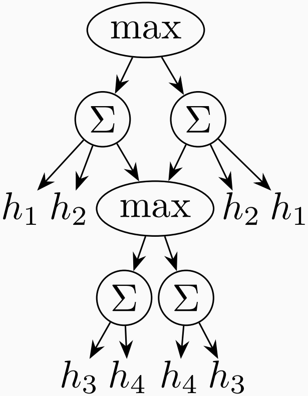
A partition of $\mathcal{H} = \{\mathcal{H}_1, \dots, \mathcal{H}_n\}$ forms *order-independent sets* if any order with the same relative ordering for elements inside each $\mathcal{H}_n$, results in the same saturated cost partitioning heuristic.

Example: $\mathcal{H}_1 = \{h_1, h_2\}, \mathcal{H}_2 = \{h_3\}$

$$h^{SCP}_{\langle h_1,h_2,h_3 \rangle} = h^{SCP}_{\langle h_1,h_3,h_2 \rangle} = h^{SCP}_{\langle h_3,h_1,h_2 \rangle}$$

$$h^{SCP}_{\langle h_2,h_1,h_3 \rangle} = h^{SCP}_{\langle h_2,h_3,h_1 \rangle} = h^{SCP}_{\langle h_3,h_2,h_1 \rangle}$$

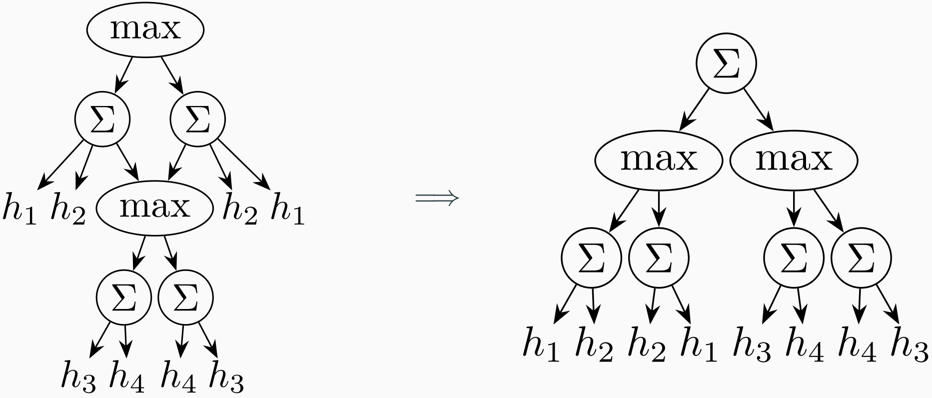
SCP Order Independence



$\Longrightarrow$

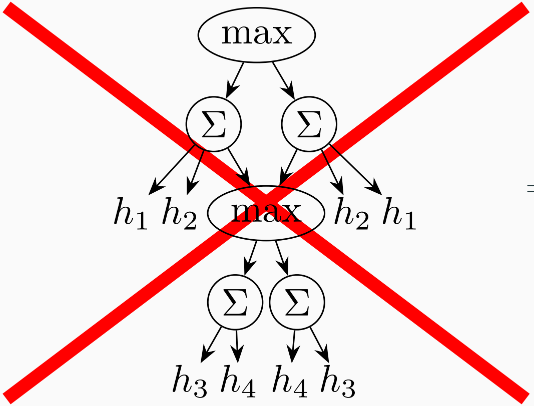
$\{h_1, h_2\}$ do not influence $\{h_3, h_4\}$

SCP Order Independence

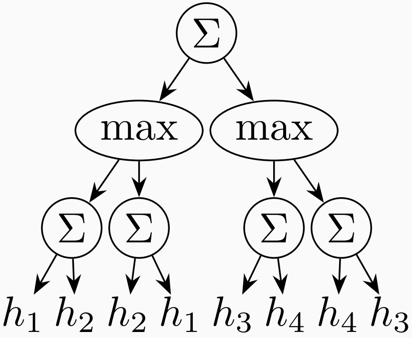


$\{h_1, h_2\}$ do not influence $\{h_3, h_4\}$

SCP Order Independence



$\Longrightarrow$



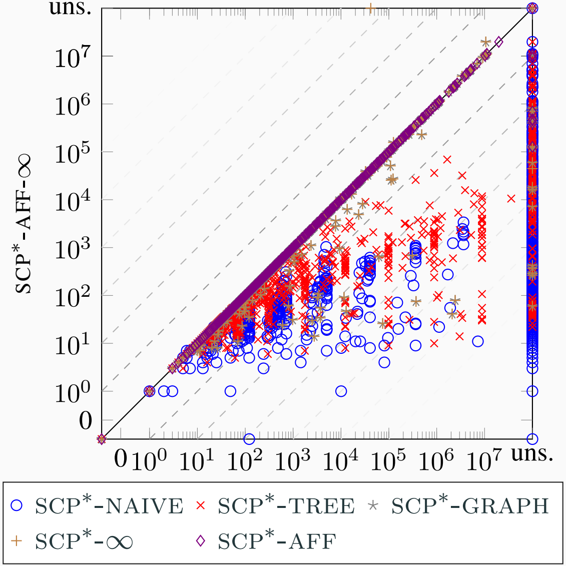
From $O(|\mathcal{H}|!)$ to $O(\max_i^n |\mathcal{H}_i|)$

Computable for up to **160** Abstractions

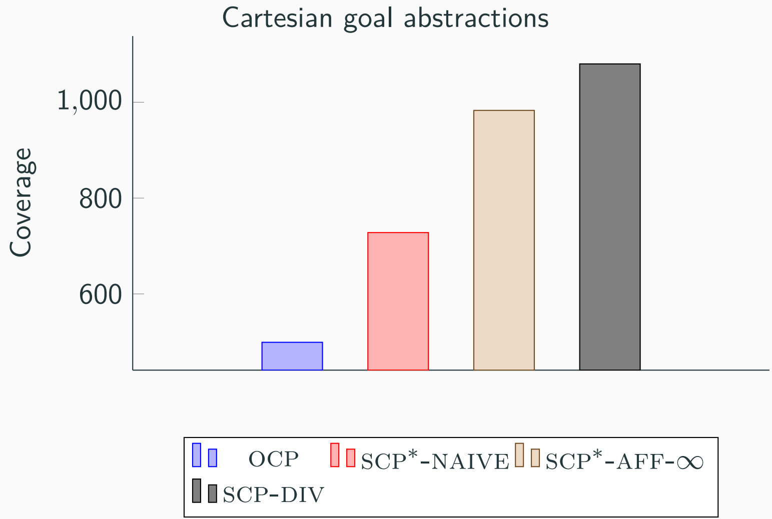
Results

- More efficient representation of SCP heuristics
- First general ADHG reduction rules
- Formalized SCP order independence (approximation)
- First practical computation of h_*^{SCP} on smaller abstraction set sizes

Results – Graph nodes



Results – Coverage



Results – Difference h_*^{SCP}, h^{SCP}, h^{OCP}