

Planning and Optimization

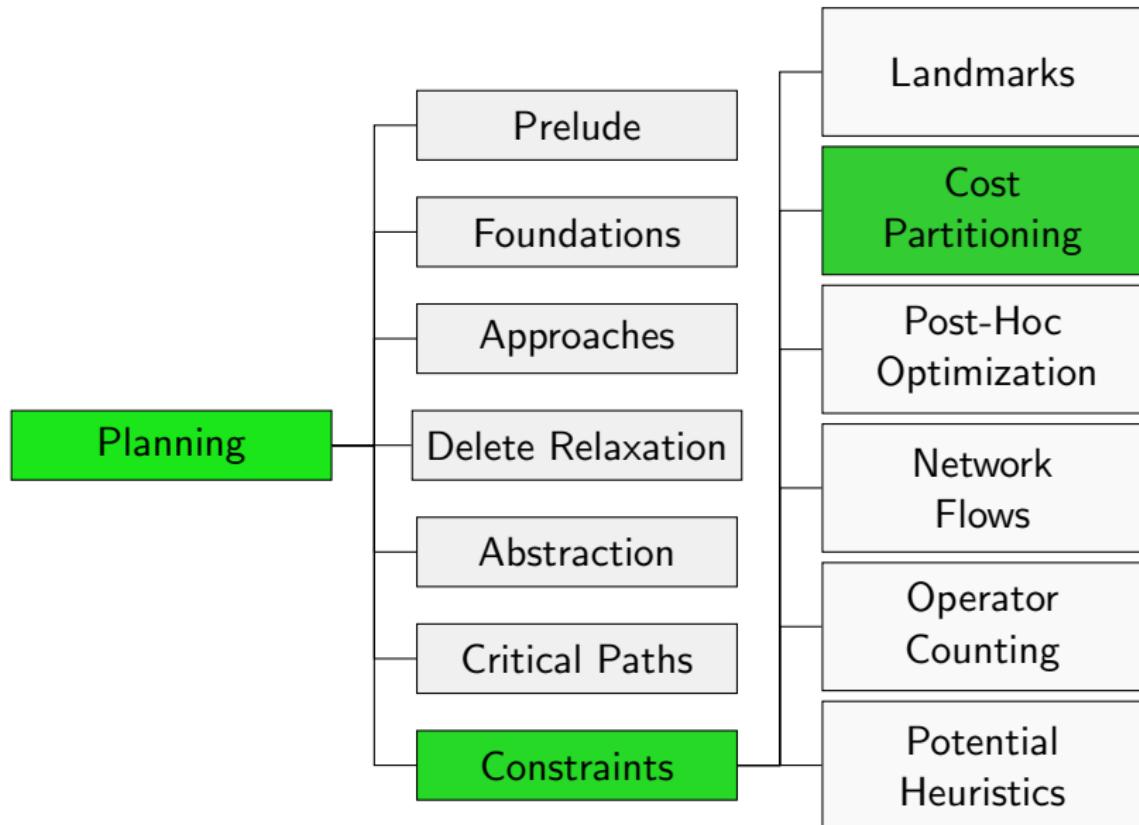
G7. Cost Partitioning

Malte Helmert and Gabriele Röger

Universität Basel

December 11, 2023

Content of this Course



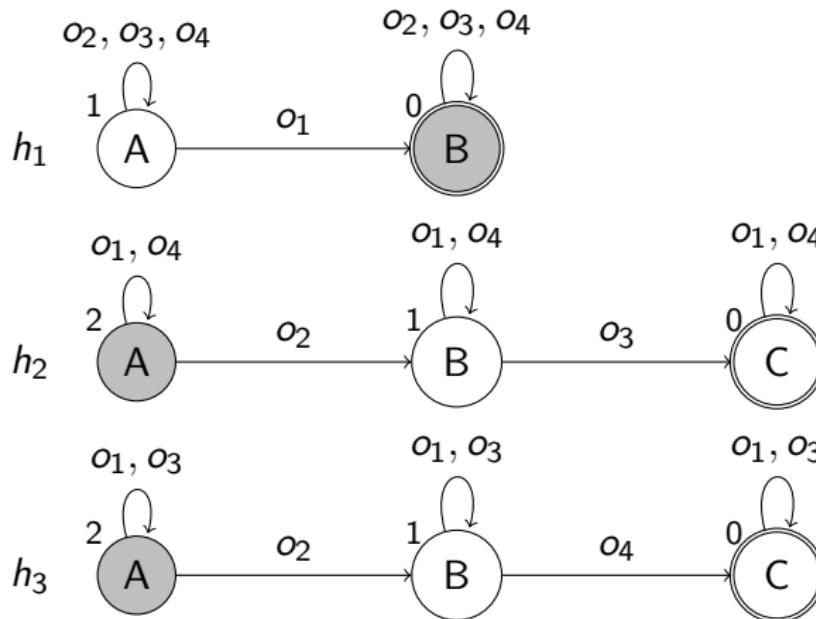
Introduction

Exploiting Additivity

- Additivity allows to add up heuristic estimates admissibly. This gives better heuristic estimates than the maximum.
- For example, the canonical heuristic for PDBs sums up where addition is admissible (by an additivity criterion) and takes the maximum otherwise.
- **Cost partitioning** provides a more general additivity criterion, based on an adaption of the operator costs.

Combining Heuristics (In)admissibly: Example

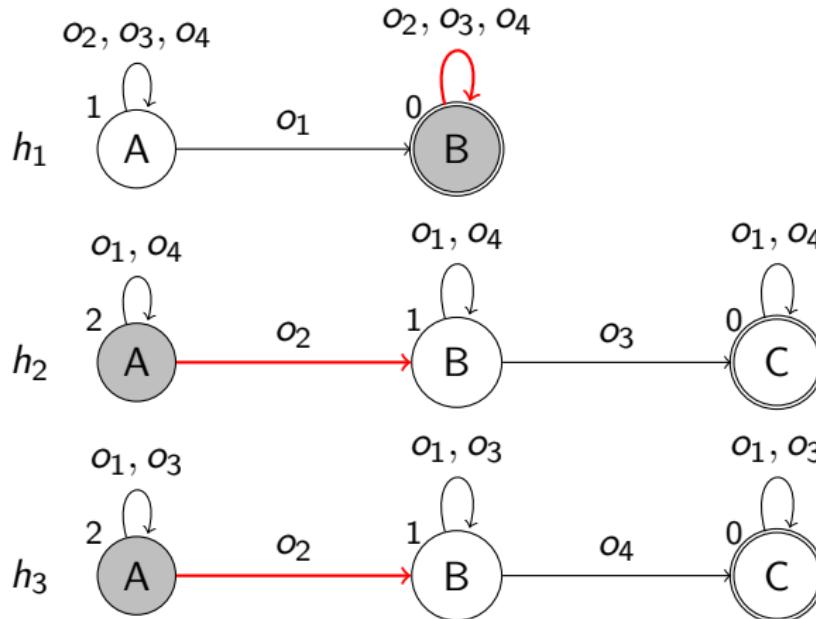
Let $h = h_1 + h_2 + h_3$.



$\langle o_2, o_3, o_4 \rangle$ is a plan for $s = \langle B, A, A \rangle$ but $h(s) = 4$.

Combining Heuristics (In)admissibly: Example

Let $h = h_1 + h_2 + h_3$.



$\langle o_2, o_3, o_4 \rangle$ is a plan for $s = \langle B, A, A \rangle$ but $h(s) = 4$.

Heuristics h_2 and h_3 both account for the single application of o_2 .

Solution: Cost Partitioning

The reason that h_2 and h_3 are not additive is because the cost of o_2 is considered in both.

Solution: Cost Partitioning

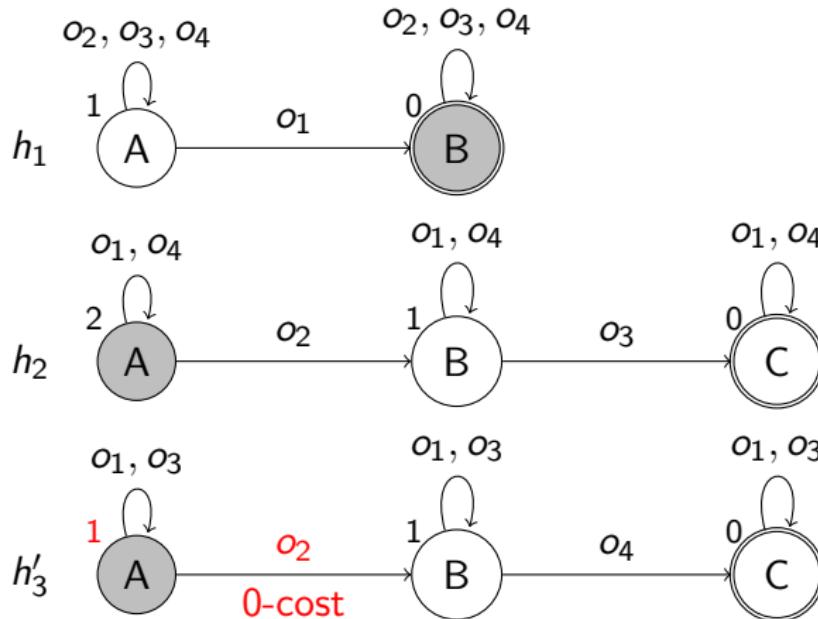
The reason that h_2 and h_3 are not additive is because the cost of σ_2 is considered in both.

Solution 1: We can ignore the cost of o_2 in all but one heuristic by setting its cost to 0 (e.g., $cost_3(o_2) = 0$).

This is a Zero-One cost partitioning.

Combining Heuristics Admissibly: Example

Let $h' = h_1 + h_2 + h'_3$, where $h'_3 = h^{v_3}$ assuming $\text{cost}_3(o_2) = 0$.



$\langle o_2, o_3, o_4 \rangle$ is an optimal plan for $s = \langle B, A, A \rangle$ and $h'(s) = 3$ an admissible estimate.

Solution: Cost Partitioning

The reason that h_2 and h_3 are not additive is because the cost of o_2 is considered in both.

Solution 1: We can ignore the cost of o_2 in all but one heuristic by setting its cost to 0 (e.g., $cost_3(o_2) = 0$).

This is a **Zero-One cost partitioning**.

Solution: Cost Partitioning

The reason that h_2 and h_3 are not additive is because the cost of o_2 is considered in both.

Solution 1: We can ignore the cost of o_2 in all but one heuristic by setting its cost to 0 (e.g., $cost_3(o_2) = 0$).

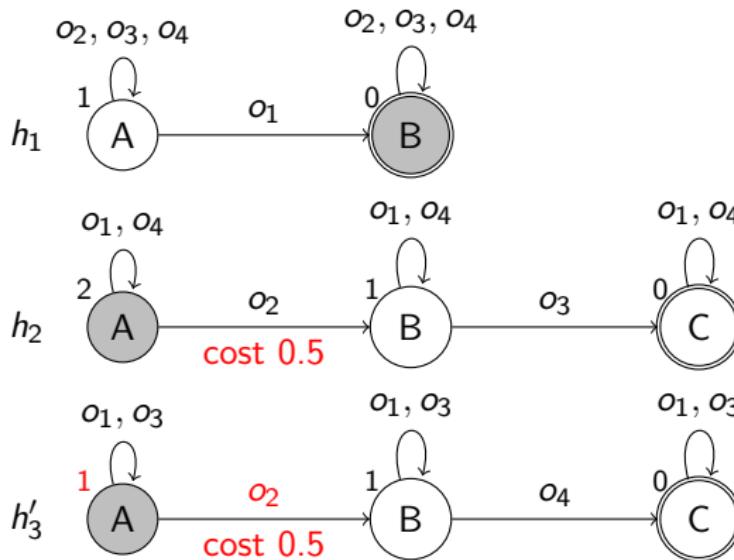
This is a **Zero-One cost partitioning**.

Solution 2: We can equally distribute the cost of o_2 between the abstractions that use it (i.e. $cost_1(o_2) = 0$, $cost_2(o_2) = cost_3(o_2) = 0.5$).

This is a **uniform cost partitioning**.

Combining Heuristics Admissibly: Example

Let $h' = h'_1 + h'_2 + h'_3$, where $h'_i = h^{v_i}$ assuming $cost_1(o_2) = 0$, $cost_2(o_2) = cost_3(o_2) = 0.5$.



$\langle o_2, o_3, o_4 \rangle$ is an optimal plan for $s = \langle B, A, A \rangle$ and $h'(s) = 0 + 1.5 + 1.5 = 3$ an admissible estimate.

Solution: Cost Partitioning

The reason that h_2 and h_3 are not additive is because the cost of o_2 is considered in both.

Solution 1: We can ignore the cost of o_2 in all but one heuristic by setting its cost to 0 (e.g., $cost_3(o_2) = 0$).

This is a **Zero-One cost partitioning**.

Solution 2: We can equally distribute the cost of o_2 between the abstractions that use it (i.e. $cost_1(o_2) = 0$, $cost_2(o_2) = cost_3(o_2) = 0.5$).

This is a **uniform cost partitioning**.

General solution: satisfy **cost partitioning constraint**

$$\sum_{i=1}^n cost_i(o) \leq cost(o) \text{ for all } o \in O$$

Solution: Cost Partitioning

The reason that h_2 and h_3 are not additive is because the cost of o_2 is considered in both.

Solution 1: We can ignore the cost of o_2 in all but one heuristic by setting its cost to 0 (e.g., $cost_3(o_2) = 0$).

This is a **Zero-One cost partitioning**.

Solution 2: We can equally distribute the cost of o_2 between the abstractions that use it (i.e. $cost_1(o_2) = 0$, $cost_2(o_2) = cost_3(o_2) = 0.5$).

This is a **uniform cost partitioning**.

General solution: satisfy **cost partitioning constraint**

$$\sum_{i=1}^n cost_i(o) \leq cost(o) \text{ for all } o \in O$$

What about o_1 , o_3 and o_4 ?

Introduction
oooooooo

Cost Partitioning
●oooooooo

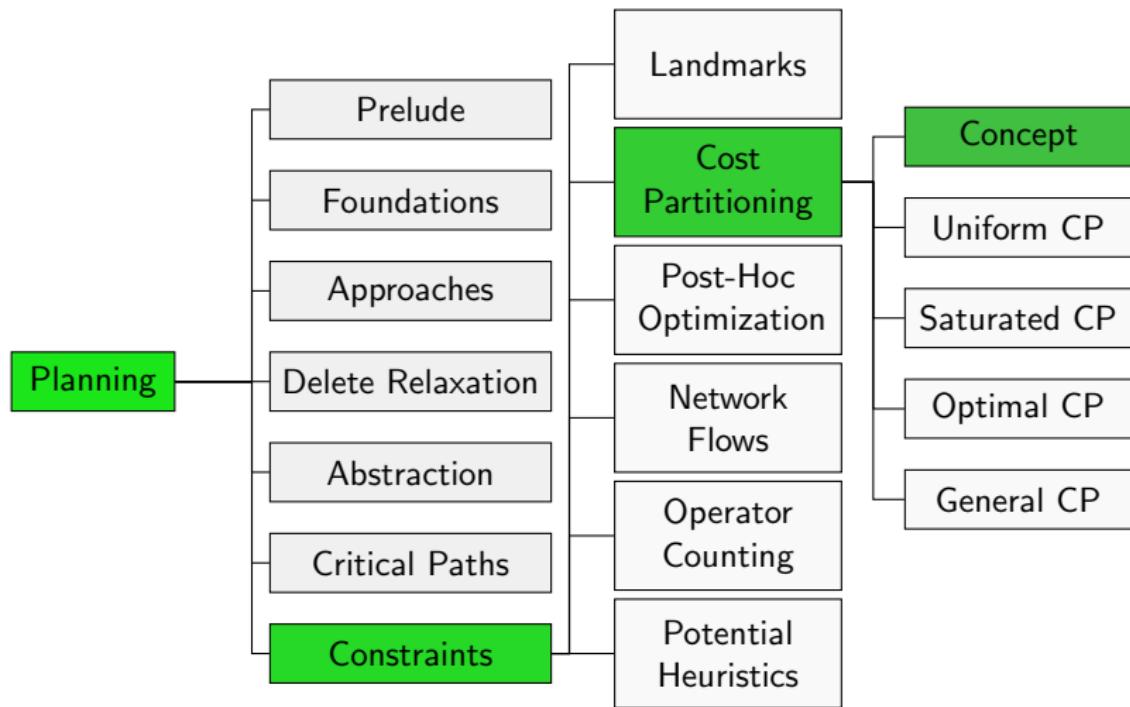
Uniform Cost Partitioning
ooooo

Saturated Cost Partitioning
oooooooooooo

Summary
oo

Cost Partitioning

Content of this Course



Cost Partitioning

Definition (Cost Partitioning)

Let Π be a planning task with operators O .

A **cost partitioning** for Π is a tuple $\langle cost_1, \dots, cost_n \rangle$, where

- $cost_i : O \rightarrow \mathbb{R}_0^+$ for $1 \leq i \leq n$ and
- $\sum_{i=1}^n cost_i(o) \leq cost(o)$ for all $o \in O$.

The cost partitioning induces a tuple $\langle \Pi_1, \dots, \Pi_n \rangle$ of planning tasks, where each Π_i is identical to Π except that the cost of each operator o is $cost_i(o)$.

Cost Partitioning: Admissibility (1)

Theorem (Sum of Solution Costs is Admissible)

Let Π be a planning task, $\langle cost_1, \dots, cost_n \rangle$ be a cost partitioning and $\langle \Pi_1, \dots, \Pi_n \rangle$ be the tuple of induced tasks.

Then the sum of the solution costs of the induced tasks is an admissible heuristic for Π , i.e., $\sum_{i=1}^n h_{\Pi_i}^* \leq h_{\Pi}^*$.

Cost Partitioning: Admissibility (2)

Proof of Theorem.

If there is no plan for state s of Π , both sides are ∞ . Otherwise, let $\pi = \langle o_1, \dots, o_m \rangle$ be an optimal plan for s . Then

$$\begin{aligned} \sum_{i=1}^n h_{\Pi_i}^*(s) &\leq \sum_{i=1}^n \sum_{j=1}^m \text{cost}_i(o_j) && (\pi \text{ plan in each } \Pi_i) \\ &= \sum_{j=1}^m \sum_{i=1}^n \text{cost}_i(o_j) && (\text{comm./ass. of sum}) \\ &\leq \sum_{j=1}^m \text{cost}(o_j) && (\text{cost partitioning}) \\ &= h_{\Pi}^*(s) && (\pi \text{ optimal plan in } \Pi) \end{aligned}$$



Cost Partitioning Preserves Admissibility

In the rest of the chapter, we write h_{Π} to denote heuristic h evaluated on task Π .

Corollary (Sum of Admissible Estimates is Admissible)

Let Π be a planning task and let $\langle \Pi_1, \dots, \Pi_n \rangle$ be induced by a cost partitioning.

For admissible heuristics h_1, \dots, h_n , the sum $h(s) = \sum_{i=1}^n h_{i,\Pi_i}(s)$ is an admissible estimate for s in Π .

Cost Partitioning Preserves Consistency

Theorem (Cost Partitioning Preserves Consistency)

Let Π be a planning task and let $\langle \Pi_1, \dots, \Pi_n \rangle$ be induced by a cost partitioning $\langle cost_1, \dots, cost_n \rangle$.

If h_1, \dots, h_n are consistent heuristics then $h = \sum_{i=1}^n h_{i,\Pi_i}$ is a consistent heuristic for Π .

Proof.

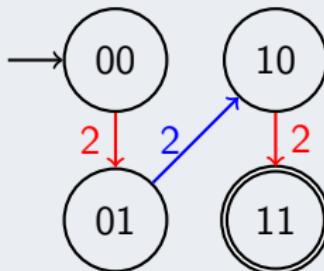
Let o be an operator that is applicable in state s .

$$\begin{aligned} h(s) &= \sum_{i=1}^n h_{i,\Pi_i}(s) \leq \sum_{i=1}^n (cost_i(o) + h_{i,\Pi_i}(s[o])) \\ &= \sum_{i=1}^n cost_i(o) + \sum_{i=1}^n h_{i,\Pi_i}(s[o]) \leq cost(o) + h(s[o]) \end{aligned}$$



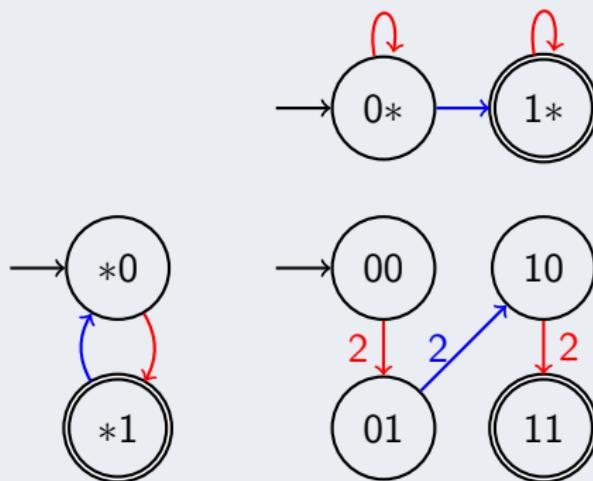
Cost Partitioning: Example

Example



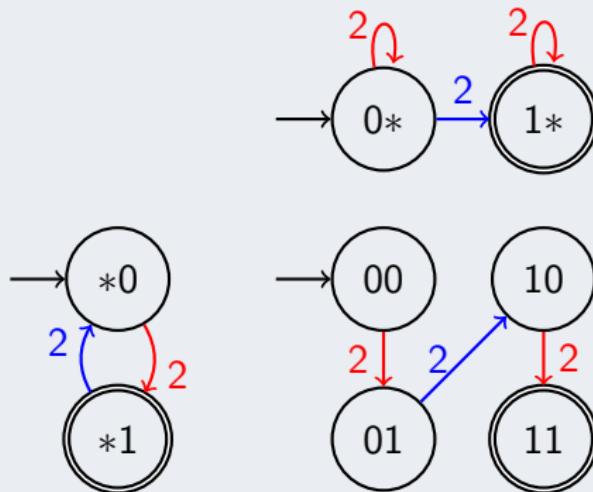
Cost Partitioning: Example

Example



Cost Partitioning: Example

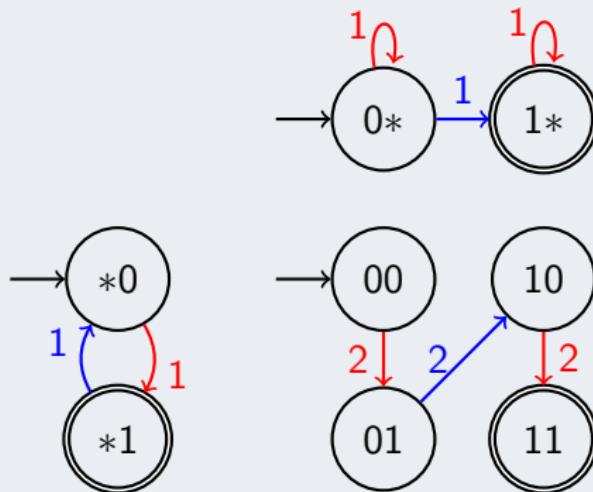
Example (No Cost Partitioning)



Heuristic value: $\max\{2, 2\} = 2$

Cost Partitioning: Example

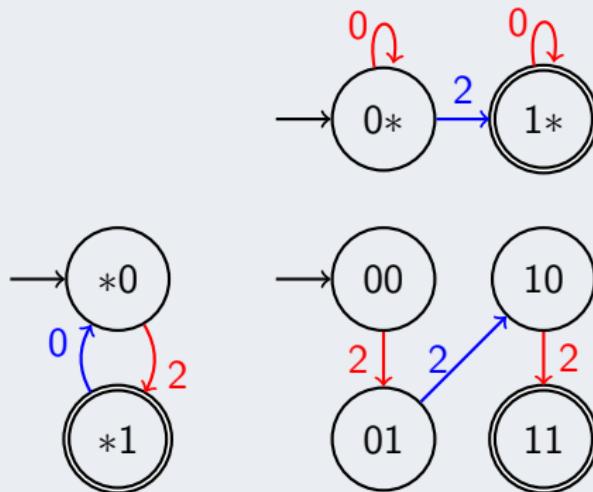
Example (Cost Partitioning 1)



Heuristic value: $1 + 1 = 2$

Cost Partitioning: Example

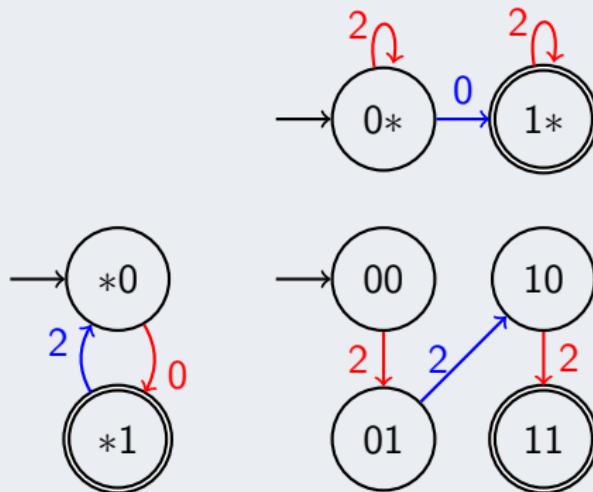
Example (Cost Partitioning 2)



Heuristic value: $2 + 2 = 4$

Cost Partitioning: Example

Example (Cost Partitioning 3)



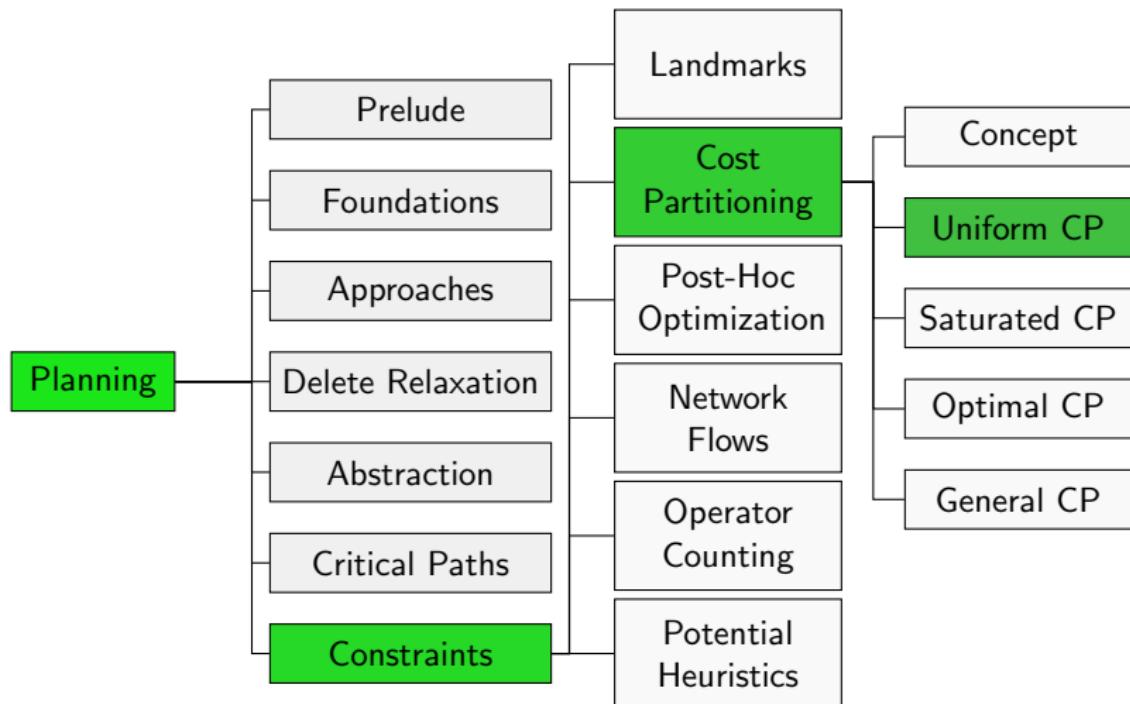
Heuristic value: $0 + 0 = 0$

Cost Partitioning: Quality

- $h(s) = h_{1,\Pi_1}(s) + \cdots + h_{n,\Pi_n}(s)$
can be **better or worse** than any $h_{i,\Pi}(s)$
→ depending on cost partitioning
- strategies for defining cost-functions
 - uniform (now)
 - zero-one
 - saturated (afterwards)
 - optimal (next chapter)

Uniform Cost Partitioning

Content of this Course



Idea

- Principal idea: Distribute the cost of each operator equally (= uniformly) among all heuristics.
- But: Some heuristics do only account for the cost of certain operators and the cost of other operators does not affect the heuristic estimate. For example:
 - a disjunctive action landmark accounts for the contained operators,
 - a PDB heuristic accounts for all operators affecting the variables in the pattern.

Idea

- Principal idea: Distribute the cost of each operator equally (= uniformly) among all heuristics.
- But: Some heuristics do only account for the cost of certain operators and the cost of other operators does not affect the heuristic estimate. For example:
 - a disjunctive action landmark accounts for the contained operators,
 - a PDB heuristic accounts for all operators affecting the variables in the pattern.

⇒ Distribute the cost of each operator uniformly among all heuristics that account for this operator.

Example: Uniform Cost Partitioning for Landmarks

- For disjunctive action landmark L of state s in task Π' , let $h_{L,\Pi'}(s)$ be the cost of L in Π' .
- Then $h_{L,\Pi'}(s)$ is admissible (in Π').
- Consider set $\mathcal{L} = \{L_1, \dots, L_n\}$ of disjunctive action landmarks for state s of task Π .
- Use cost partitioning $\langle cost_{L_1}, \dots, cost_{L_n} \rangle$, where

$$cost_{L_i}(o) = \begin{cases} cost(o)/|\{L \in \mathcal{L} \mid o \in L\}| & \text{if } o \in L_i \\ 0 & \text{otherwise} \end{cases}$$

- Let $\langle \Pi_{L_1}, \dots, \Pi_{L_n} \rangle$ be the tuple of induced tasks.
- $h(s) = \sum_{i=1}^n h_{L_i, \Pi_{L_i}}(s)$ is an admissible estimate for s in Π .
- h is the uniform cost partitioning heuristic for landmarks.

Example: Uniform Cost Partitioning for Landmarks

Definition (Uniform Cost Partitioning Heuristic for Landmarks)

Let \mathcal{L} be a set of disjunctive action landmarks.

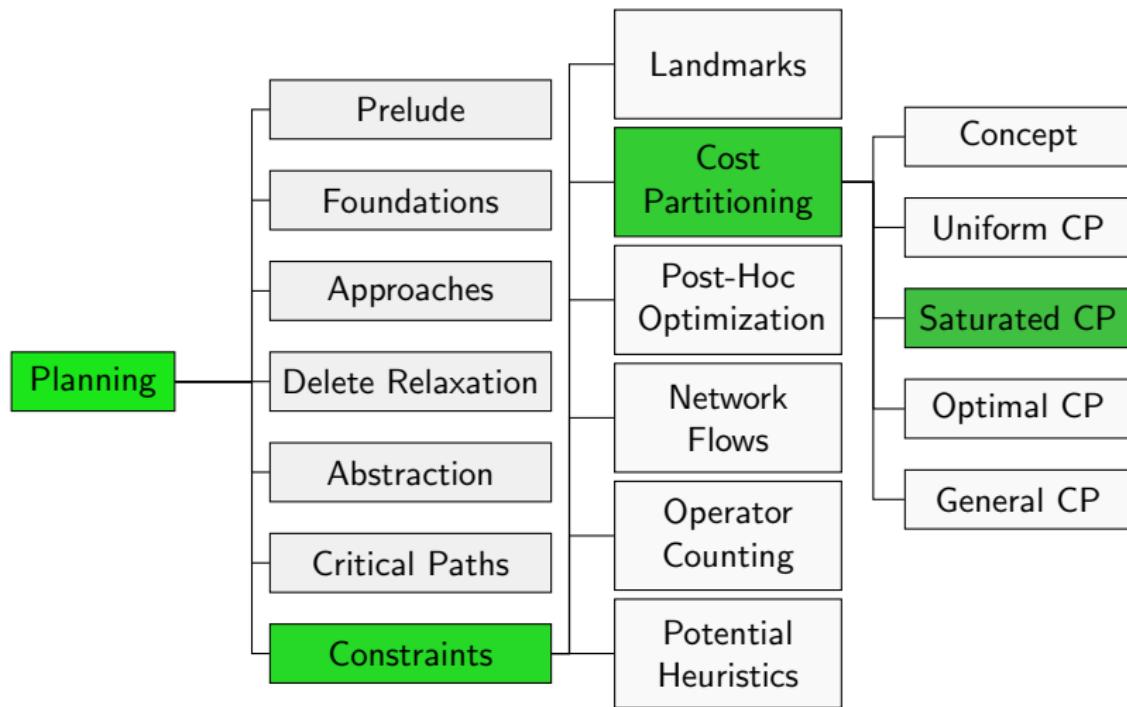
The **uniform cost partitioning heuristic** $h^{\text{UCP}}(\mathcal{L})$ is defined as

$$h^{\text{UCP}}(\mathcal{L}) = \sum_{L \in \mathcal{L}} \min_{o \in L} c'(o) \text{ with}$$

$$c'(o) = \text{cost}(o)/|\{L \in \mathcal{L} \mid o \in L\}|.$$

Saturated Cost Partitioning

Content of this Course



Idea

Heuristics do not always “need” all operator costs

- Pick a heuristic and use minimum costs **preserving all estimates**
- Continue with **remaining cost** until all heuristics were picked

Saturated cost partitioning (SCP) currently offers the **best tradeoff** between computation time and heuristic guidance in practice.

Saturated Cost Function

Definition (Saturated Cost Function)

Let Π be a planning task and h be a heuristic.

A cost function scf is **saturated** for h and $cost$ if

- ① $scf(o) \leq cost(o)$ for all operators o and
- ② $h_{\Pi_{scf}}(s) = h_{\Pi}(s)$ for all states s ,
where Π_{scf} is Π with cost function scf.

Minimal Saturated Cost Function

For **abstractions**, there exists a unique
minimal saturated cost function (MSCF).

Definition (MSCF for Abstractions)

Let Π be a planning task and α be an abstraction heuristic.
The **minimal saturated cost function** for α is

$$\text{mscf}(o) = \max\left(\max_{\alpha(s) \xrightarrow{o} \alpha(t)} h^\alpha(s) - h^\alpha(t), 0\right)$$

Algorithm

Saturated Cost Partitioning: Seipp & Helmert (2014)

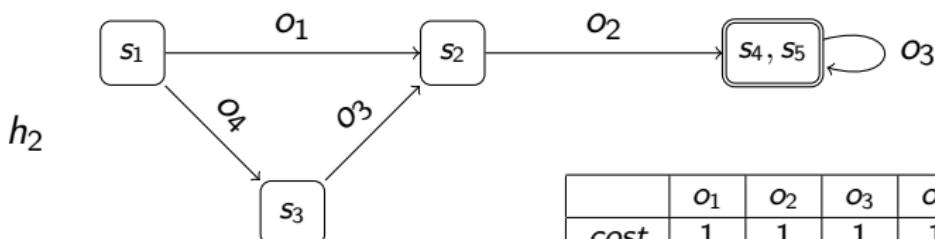
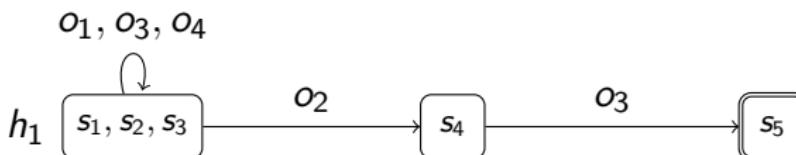
Iterate:

- ① Pick a heuristic h_i that hasn't been picked before.
Terminate if none is left.
- ② Compute h_i given current cost
- ③ Compute an (ideally minimal) saturated cost function scf_i for h_i
- ④ Decrease $cost(o)$ by $scf_i(o)$ for all operators o

$\langle scf_1, \dots, scf_n \rangle$ is **saturated cost partitioning** (SCP)
for $\langle h_1, \dots, h_n \rangle$ (in pick order)

Example

Consider the abstraction heuristics h_1 and h_2

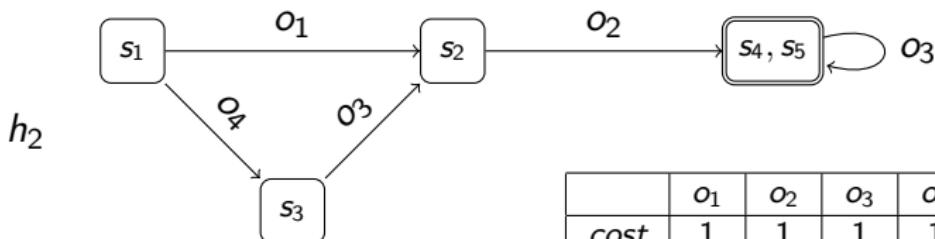
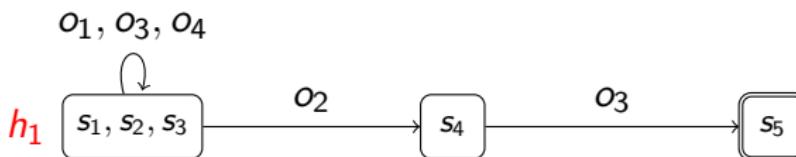


	o_1	o_2	o_3	o_4
<i>cost</i>	1	1	1	1

Example

Consider the abstraction heuristics h_1 and h_2

- 1 Pick a heuristic h_i

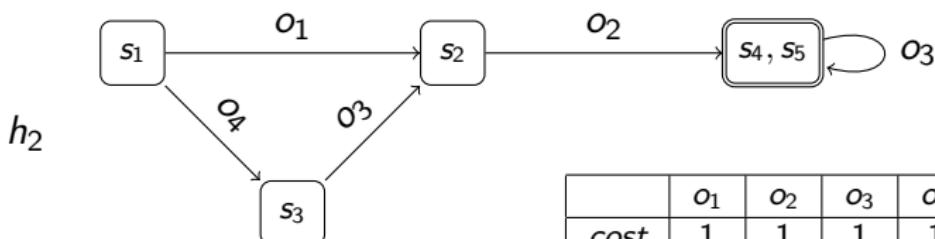
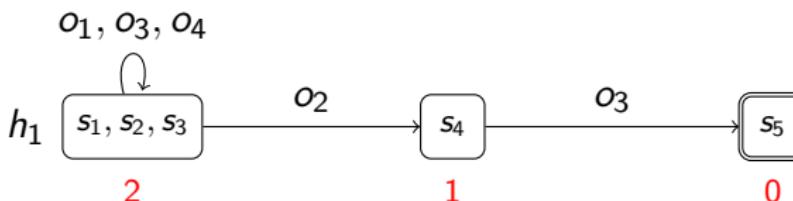


	o_1	o_2	o_3	o_4
<i>cost</i>	1	1	1	1

Example

Consider the abstraction heuristics h_1 and h_2

② Compute h_i

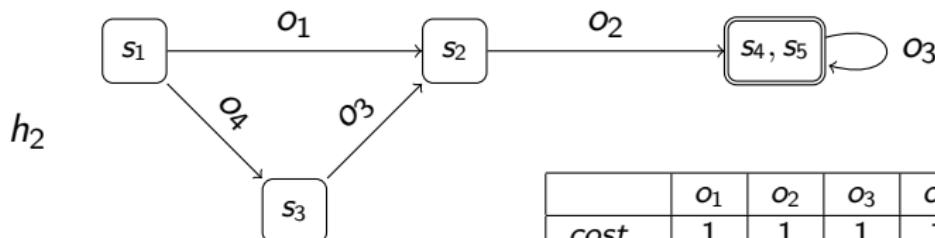
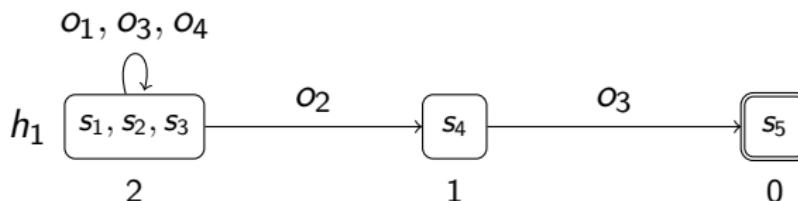


	o_1	o_2	o_3	o_4
<i>cost</i>	1	1	1	1

Example

Consider the abstraction heuristics h_1 and h_2

③ Compute minimal saturated cost function $mscf_i$ for h_i

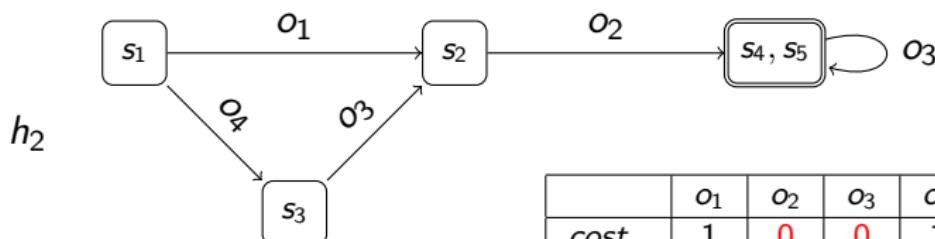
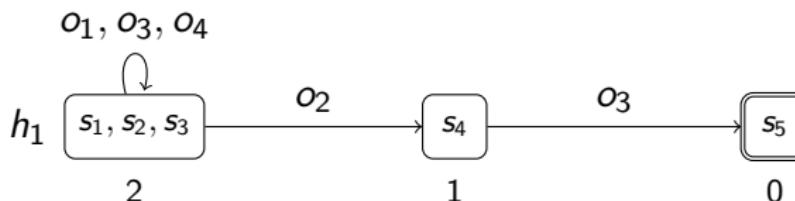


	o_1	o_2	o_3	o_4
<i>cost</i>	1	1	1	1
$mscf_1$	0	1	1	0

Example

Consider the abstraction heuristics h_1 and h_2

- ④ Decrease $cost(o)$ by $mscf_i(o)$ for all operators o

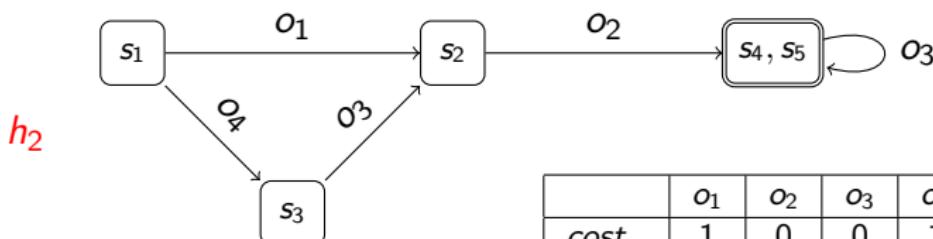
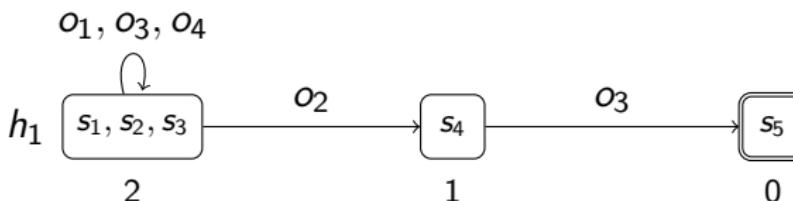


	o_1	o_2	o_3	o_4
$cost$	1	0	0	1
$mscf_1$	0	1	1	0

Example

Consider the abstraction heuristics h_1 and h_2

- 1 Pick a heuristic h_i

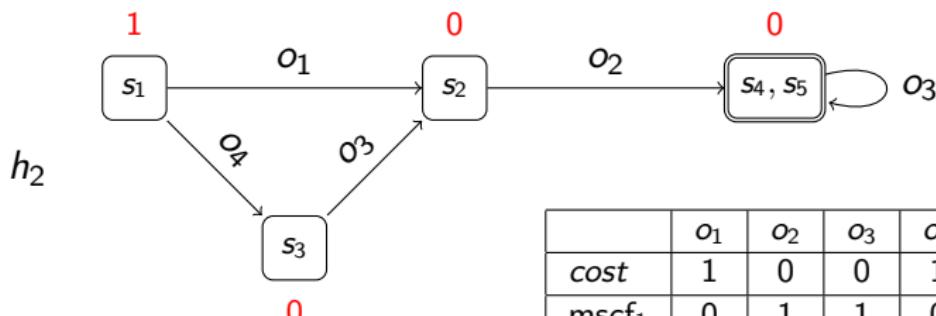
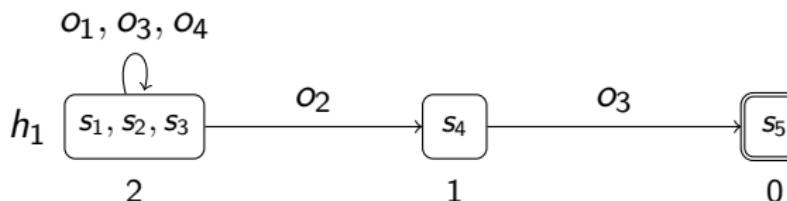


	o_1	o_2	o_3	o_4
<i>cost</i>	1	0	0	1
<i>mscf₁</i>	0	1	1	0

Example

Consider the abstraction heuristics h_1 and h_2

② Compute h_i

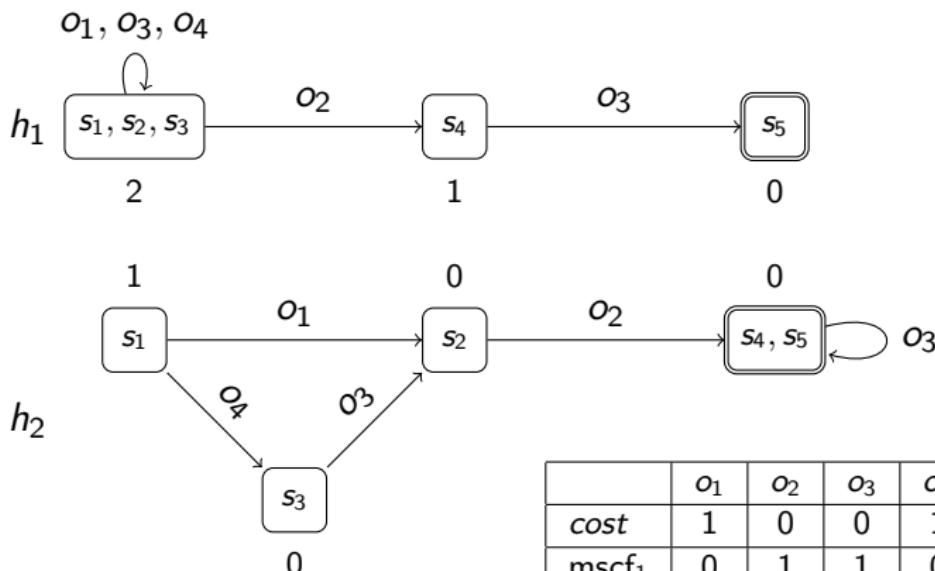


	o_1	o_2	o_3	o_4
<i>cost</i>	1	0	0	1
<i>mscf₁</i>	0	1	1	0

Example

Consider the abstraction heuristics h_1 and h_2

③ Compute minimal saturated cost function $mscf_i$ for h_i

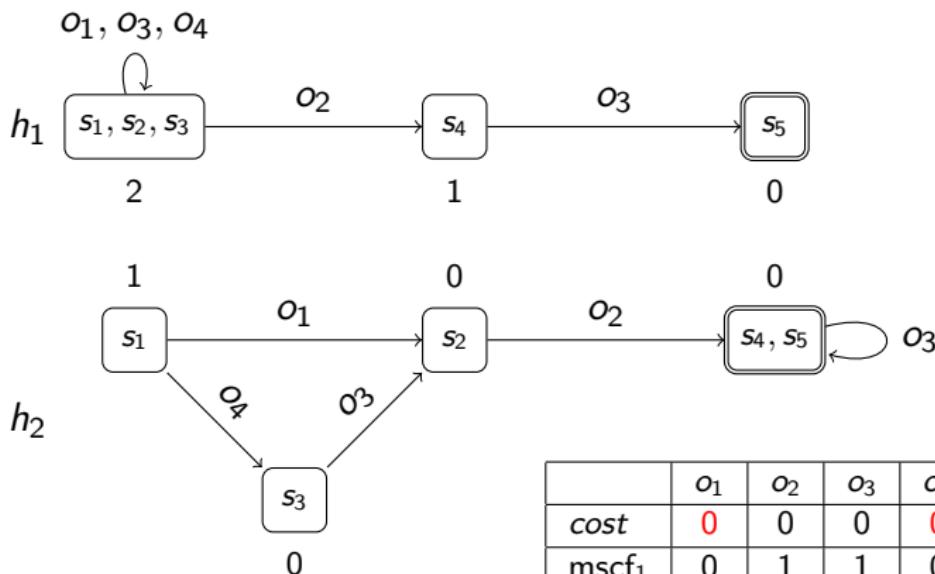


	o_1	o_2	o_3	o_4
<i>cost</i>	1	0	0	1
$mscf_1$	0	1	1	0
$mscf_2$	1	0	0	1

Example

Consider the abstraction heuristics h_1 and h_2

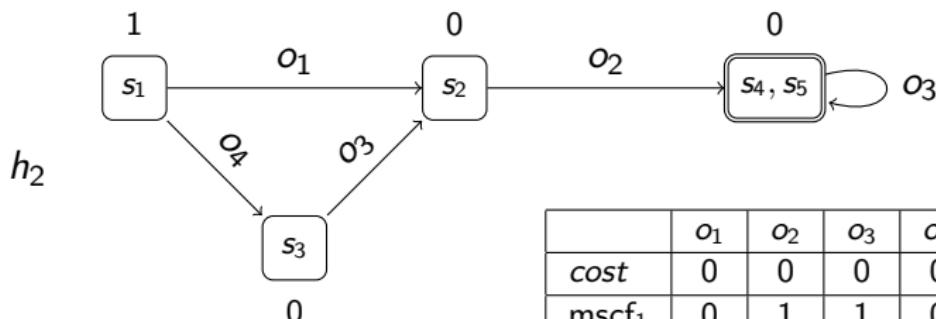
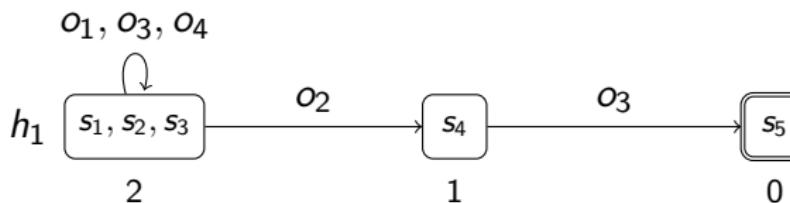
④ Decrease $cost(o)$ by $mscf_i(o)$ for all operators o



Example

Consider the abstraction heuristics h_1 and h_2

- 1 Pick a heuristic h_i . **Terminate if none is left.**



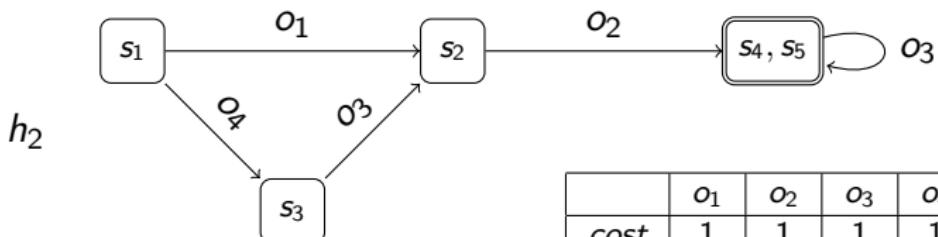
	o_1	o_2	o_3	o_4
<i>cost</i>	0	0	0	0
<i>mscf</i> ₁	0	1	1	0
<i>mscf</i> ₂	1	0	0	1

Influence of Selected Order

- quality **highly susceptible to selected order**
- there are almost always orders where SCP performs much better than uniform or zero-one cost partitioning
- but there are also often orders where SCP performs worse

Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

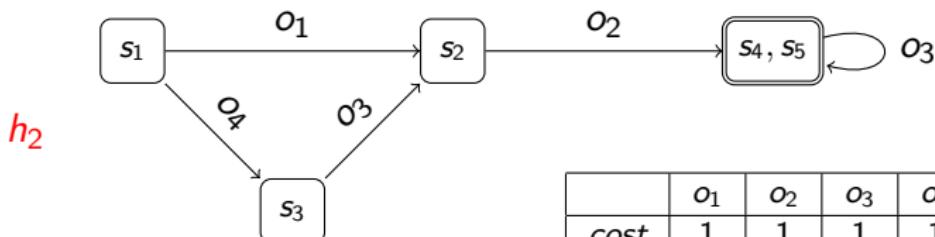


	o_1	o_2	o_3	o_4
<i>cost</i>	1	1	1	1

Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

- 1 Pick a heuristic h_i

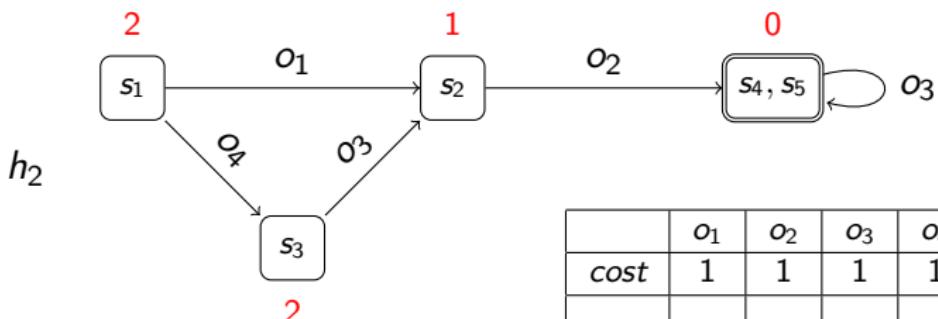


	o_1	o_2	o_3	o_4
$cost$	1	1	1	1

Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

② Compute h_i

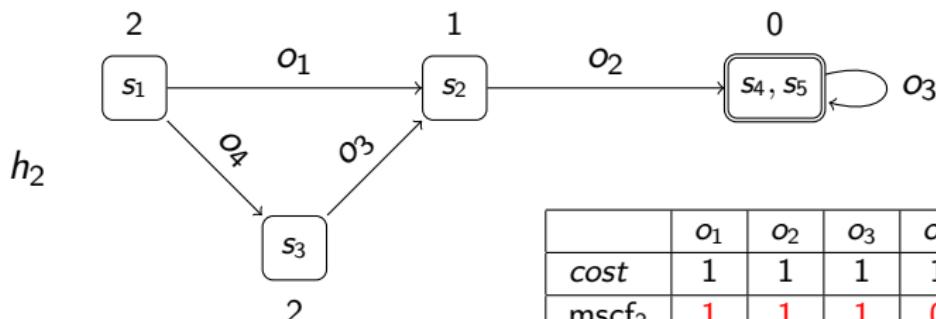
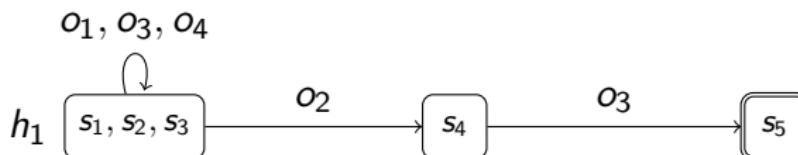


	o_1	o_2	o_3	o_4
$cost$	1	1	1	1

Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

③ Compute minimal saturated cost function $mscf_i$ for h_i

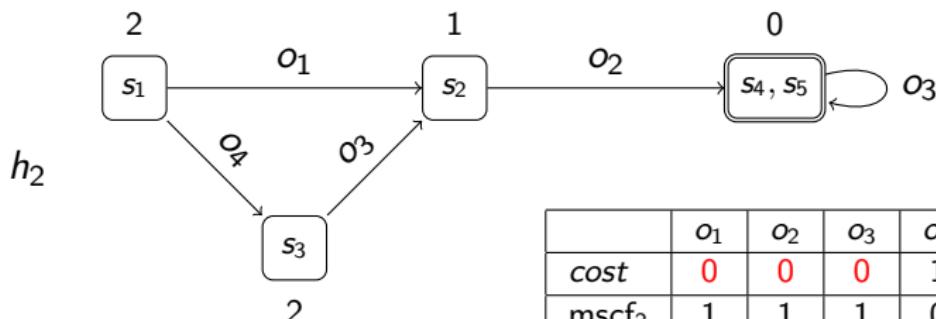


	o_1	o_2	o_3	o_4
$cost$	1	1	1	1
$mscf_2$	1	1	1	0

Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

4 Decrease $cost(o)$ by $mscf_i(o)$ for all operators o

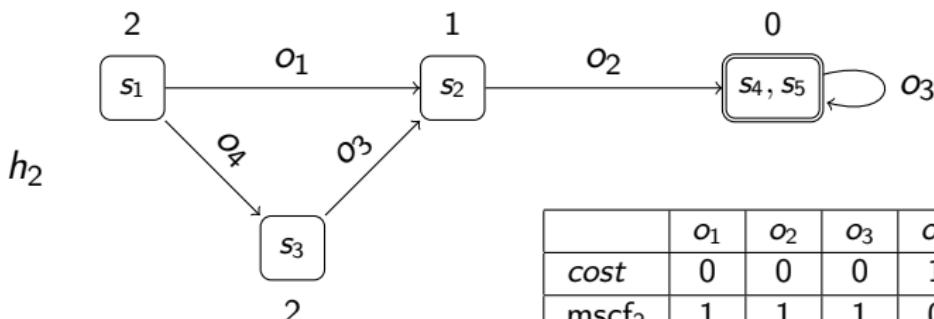


	o_1	o_2	o_3	o_4
$cost$	0	0	0	1
$mscf_2$	1	1	1	0

Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

- 1 Pick a heuristic h_i

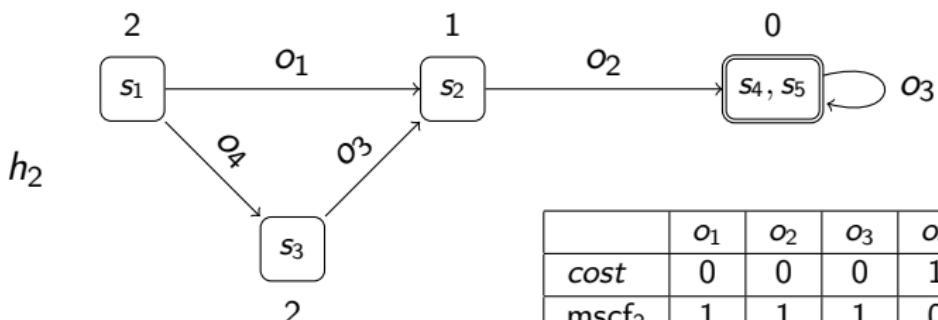
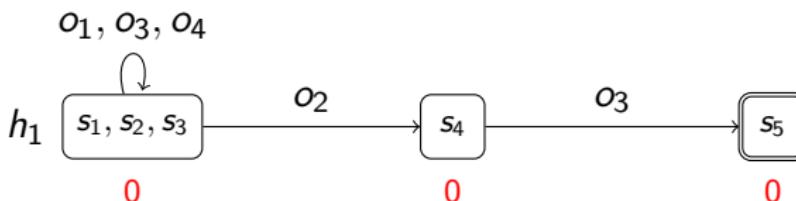


	o_1	o_2	o_3	o_4
<i>cost</i>	0	0	0	1
<i>mscf₂</i>	1	1	1	0

Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

② Compute h_i

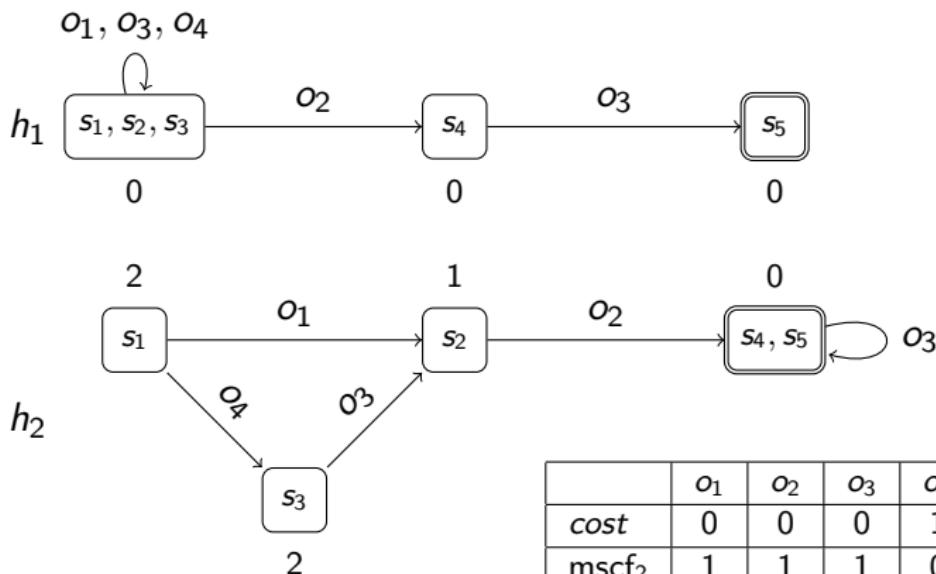


	o_1	o_2	o_3	o_4
<i>cost</i>	0	0	0	1
<i>mscf₂</i>	1	1	1	0

Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

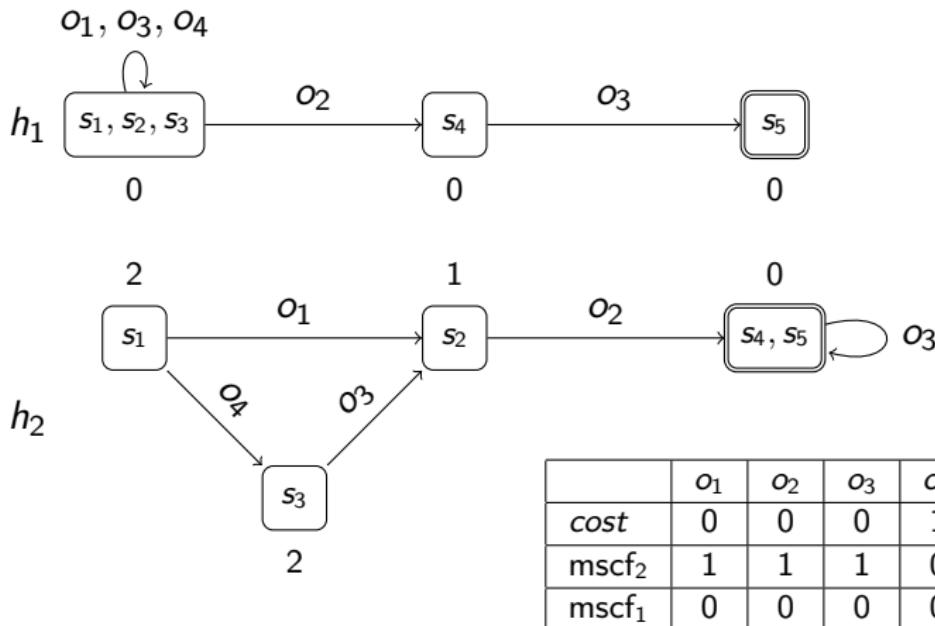
③ Compute minimal saturated cost function $mscf_i$ for h_i



Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

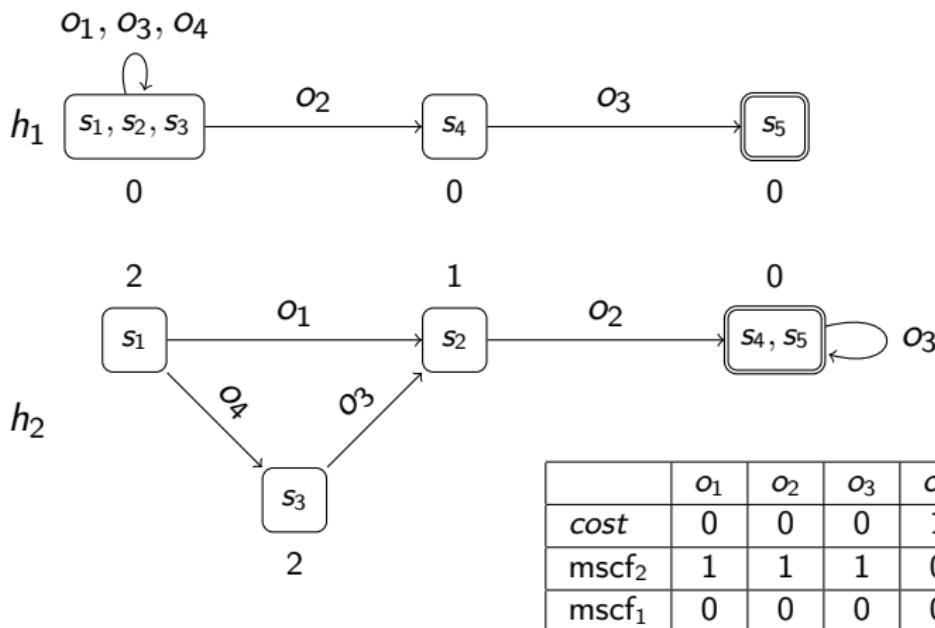
- ④ Decrease $cost(o)$ by $mscf_i(o)$ for all operators o



Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2

- 1 Pick a heuristic h_i . **Terminate if none is left.**



Influence of Selected Order

- quality **highly susceptible to selected order**
- there are almost always orders where SCP performs much better than uniform or zero-one cost partitioning
- but there are also often orders where SCP performs worse

Maximizing over multiple orders good solution in practice

SCP for Disjunctive Action Landmarks

For **disjunctive action landmarks** we also know how to compute a **minimal saturated cost function**:

Definition (MSCF for Disjunctive Action Landmark)

Let Π be a planning task and \mathcal{L} be a disjunctive action landmark. The **minimal saturated cost function** for \mathcal{L} is

$$\text{mscf}(o) = \begin{cases} \min_{o \in \mathcal{L}} \text{cost}(o) & \text{if } o \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

SCP for Disjunctive Action Landmarks

For **disjunctive action landmarks** we also know how to compute a **minimal saturated cost function**:

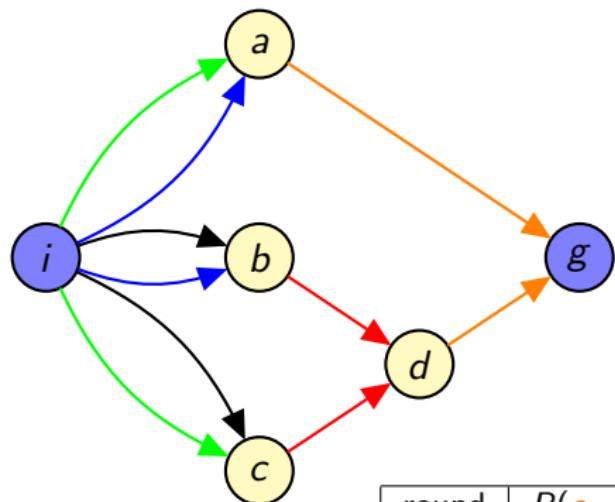
Definition (MSCF for Disjunctive Action Landmark)

Let Π be a planning task and \mathcal{L} be a disjunctive action landmark. The **minimal saturated cost function** for \mathcal{L} is

$$\text{mscf}(o) = \begin{cases} \min_{o \in \mathcal{L}} \text{cost}(o) & \text{if } o \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

Does this look familiar?

Reminder: LM-Cut



$$o_{blue} = \langle \{i\}, \{a, b\}, \{\}, 4 \rangle$$

$$o_{green} = \langle \{i\}, \{a, c\}, \{\}, 5 \rangle$$

$$o_{black} = \langle \{i\}, \{b, c\}, \{\}, 3 \rangle$$

$$o_{red} = \langle \{b, c\}, \{d\}, \{\}, 2 \rangle$$

$$o_{orange} = \langle \{a, d\}, \{g\}, \{\}, 0 \rangle$$

round	$P(o_{orange})$	$P(o_{red})$	landmark	cost
1	d	b	{ o_{red} }	2
2	a	b	{ o_{green} , o_{blue} }	4
3	d	c	{ o_{green} , o_{black} }	1
$h^{\text{LM-cut}}(I)$				7

SCP for Disjunctive Action Landmarks

Same algorithm can be used for **disjunctive action landmarks**, where we also have a **minimal saturated cost function**.

Definition (MSCF for Disjunctive Action Landmark)

Let Π be a planning task and \mathcal{L} be a disjunctive action landmark. The **minimal saturated cost function** for \mathcal{L} is

$$\text{mscf}(o) = \begin{cases} \min_{o \in \mathcal{L}} \text{cost}(o) & \text{if } o \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

Does this look familiar?

LM-Cut computes SCP over disjunctive action landmarks

Introduction
ooooooooo

Cost Partitioning
ooooooooo

Uniform Cost Partitioning
oooooo

Saturated Cost Partitioning
oooooooooooo

Summary
●○

Summary

Summary

- **Cost partitioning** allows to admissibly add up estimates of several heuristics.
- This can be better or worse than the best individual heuristic on the original problem, depending on the cost partitioning.
- **Uniform cost partitioning** distributes the cost of each operator uniformly among all heuristics that account for it.
- **Saturated cost partitioning** offers a good tradeoff between computation time and heuristic guidance.
- LM-Cut computes a SCP over disjunctive action landmarks.