Planning and Optimization

E5. Cost Partitioning

Malte Helmert and Gabriele Röger

Universität Basel

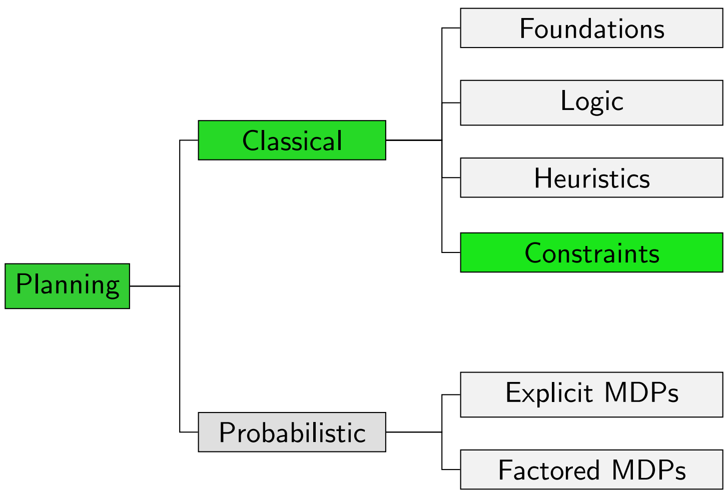
E5.1 Introduction

E5.2 Cost Partitioning

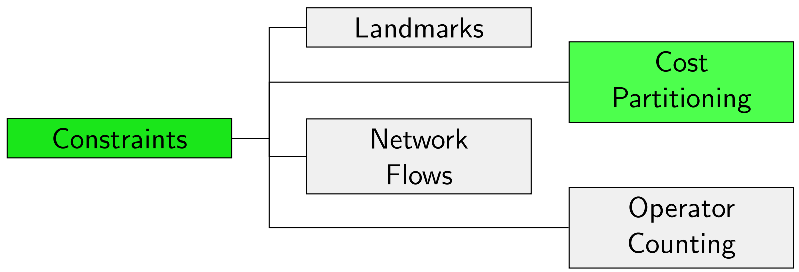
E5.3 Saturated Cost Partitioning

E5.4 Summary

Content of this Course



Content of this Course: Constraints



E5.1 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.

Additivity

When is it impossible to sum up abstraction heuristics admissibly?

- Abstraction heuristics are consistent and goal-aware.
- Sum of goal-aware heuristics is goal aware.
- $\Rightarrow$ Sum of consistent heuristics not necessarily consistent.

Combining Heuristics Admissibly: Example

Example

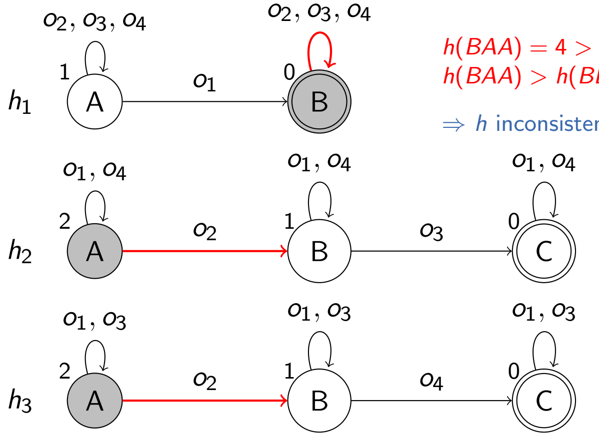
Consider an FDR planning task $\langle V, I, \{o_1, o_2, o_3, o_4\}, \gamma\rangle$ with $V = \{v_1, v_2, v_3\}$ with $dom(v_1) = \{A, B\}$ and $\mathrm{dom}(v_2) = \mathrm{dom}(v_3) = \{A, B, C\}$, $I = \{v_1 \mapsto A, v_2 \mapsto A, v_3 \mapsto A\}$,

$$o_1 = \langle v_1 = \mathrm{A}, v_1 := \mathrm{B}, 1\rangle$$
$$o_2 = \langle v_2 = \mathrm{A} \land v_3 = \mathrm{A}, v_2 := \mathrm{B} \land v_3 := \mathrm{B}, 1\rangle$$
$$o_3 = \langle v_2 = \mathrm{B}, v_2 := \mathrm{C}, 1\rangle$$
$$o_4 = \langle v_3 = \mathrm{B}, v_3 := \mathrm{C}, 1\rangle$$

and $\gamma = (v_1 = \mathrm{B}) \land (v_2 = \mathrm{C}) \land (v_3 = \mathrm{C})$.

Combining Heuristics Admissibly: Example

Let $h = h_1 + h_2 + h_3$. Where is consistency constraint violated?



$h(BAA) = 4 > h^*(BAA)$ and
$h(BAA) > h(BBB) + cost(o_2)$

$\Rightarrow$ h inconsistent and inadmissible

Consider solution $\langle o_1, o_2, o_3, o_4 \rangle$

Solution: Cost partitioning

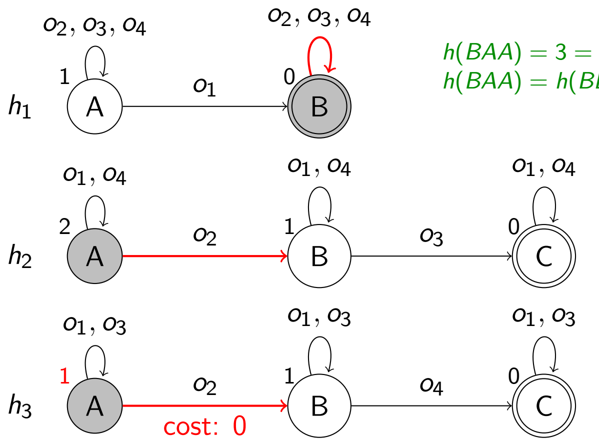
h is not admissible because $cost(o_2)$ is considered in h_2 and h_3

Is there anything we can do about this?

Solution 1:
We can ignore the cost of o_2 in h_2 or h_3 by setting its cost to 0.

Combining Heuristics Admissibly: Example

Assume $cost_3(o_2) = 0$



$h(BAA) = 3 = h^*(BAA)$ and
$h(BAA) = h(BBB) + cost(o_2)$

Consider solution $\langle o_1, o_2, o_3, o_4 \rangle$

Solution: Cost partitioning

h is not admissible because $cost(o_2)$ is considered in h_2 and h_3

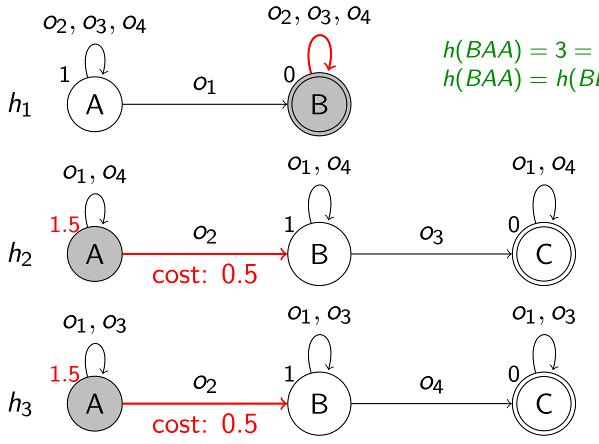
Is there anything we can do about this?

Solution 1:
We can ignore the cost of o_2 in h_2 or h_3 by setting its cost to 0.
This is called a zero-one cost partitioning.

Solution 2: Consider a cost of $\frac{1}{2}$ for o_2 both in h_2 and h_3.

Combining Heuristics Admissibly: Example

Assume $cost_2(o_2) = cost_3(o_2) = \frac{1}{2}$



$h(BAA) = 3 = h^*(BAA)$ and
$h(BAA) = h(BBB) + cost(o_2)$

Consider solution $\langle o_1, o_2, o_3, o_4 \rangle$

Solution: Cost partitioning

h is not admissible because $cost(o_2)$ is considered in h_2 and h_3

Is there anything we can do about this?

Solution 1:
We can ignore the cost of o_2 in h_2 or h_3 by setting its cost to 0.
This is called a zero-one cost partitioning.

Solution 2: Consider a cost of $\frac{1}{2}$ for o_2 both in h_2 and h_3.
This is called 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?

E5.2 Cost Partitioning

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

- $cost_i : O \to \mathbb{R}_0^+$ 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 state s of Π. Then

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

□

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.

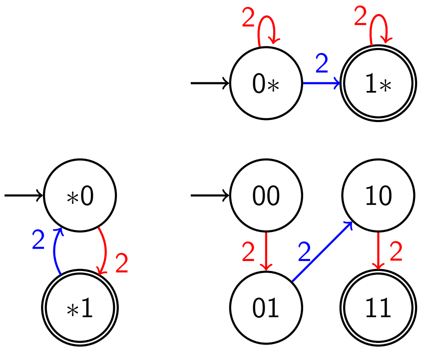
Let o be an operator that is applicable in state s.

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

□

Cost Partitioning: Example

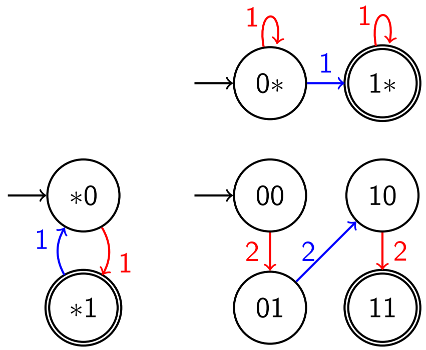
Example (No Cost Partitioning)



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

Cost Partitioning: Example

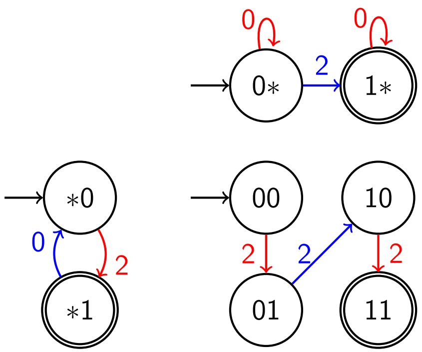
Example (Cost Partitioning 1)



Heuristic value: $1 + 1 = 2$

Cost Partitioning: Example

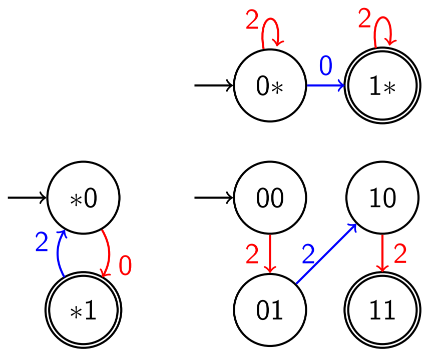
Example (Cost Partitioning 2)



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 $h_{i,\Pi}(s)$
  $\rightarrow$ depending on cost partitioning
- strategies for defining cost-functions
  - uniform
  - zero-one
  - saturated (now)
  - optimal (next chapter)

E5.3 Saturated Cost Partitioning

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. $\mathrm{scf}(o) \leq cost(o)$ for all operators o and
2. $h_{\Pi_{\mathrm{scf}}}(s) = h_{\Pi}(s)$ 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 for Π.
The minimal saturated cost function for α is

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

Algorithm

Saturated Cost Partitioning: Seipp & Helmert (2014)

Iterate:

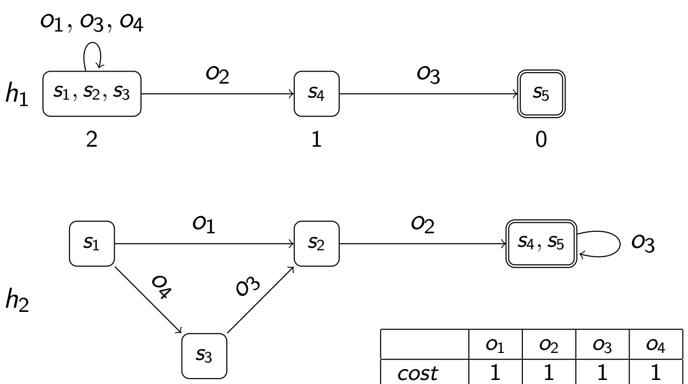
1. Pick a heuristic h_i that hasn't been picked before. Terminate if none is left.
2. Compute h_i given current *cost*
3. Compute minimal saturated cost function mscf_i for h_i
4. Decrease $cost(o)$ by $\text{mscf}_i(o)$ for all operators o

$\langle \text{mscf}_1, \dots, \text{mscf}_n \rangle$ 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

3. Compute minimal saturated cost function mscf_i for h_i

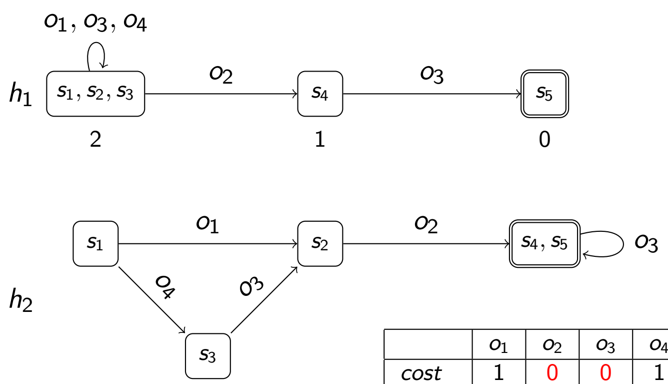


	o_1	o_2	o_3	o_4
cost	1	1	1	1
mscf_1	0	1	1	0

Example

Consider the abstraction heuristics h_1 and h_2

4. Decrease $cost(o)$ by $\text{mscf}_i(o)$ for all operators o

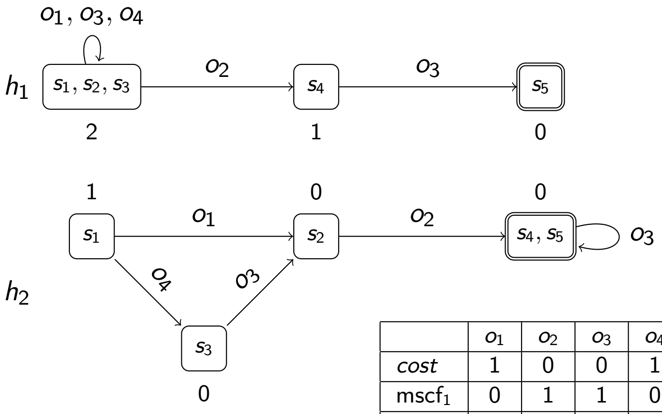


	o_1	o_2	o_3	o_4
cost	1	0	0	1
mscf_1	0	1	1	0

Example

Consider the abstraction heuristics h_1 and h_2

3. Compute minimal saturated cost function mscf_i for h_i

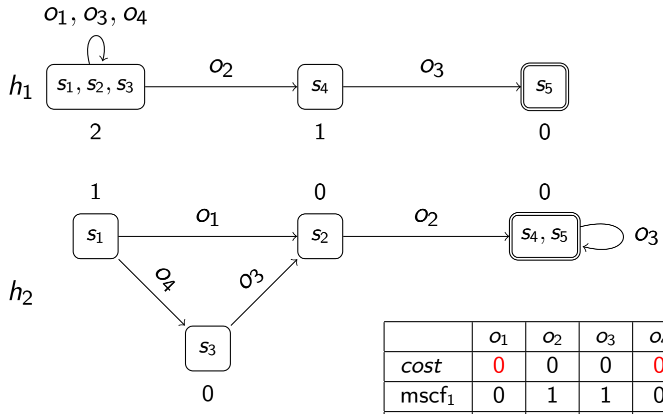


	o_1	o_2	o_3	o_4
cost	1	0	0	1
mscf_1	0	1	1	0
mscf_2	1	0	0	1

Example

Consider the abstraction heuristics h_1 and h_2

4. Decrease $cost(o)$ by $\text{mscf}_i(o)$ for all operators o

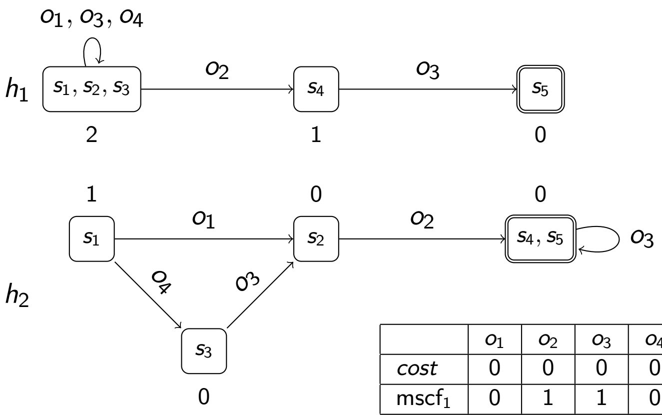


	o_1	o_2	o_3	o_4
cost	0	0	0	0
mscf_1	0	1	1	0
mscf_2	1	0	0	1

Example

Consider the abstraction heuristics h_1 and h_2

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



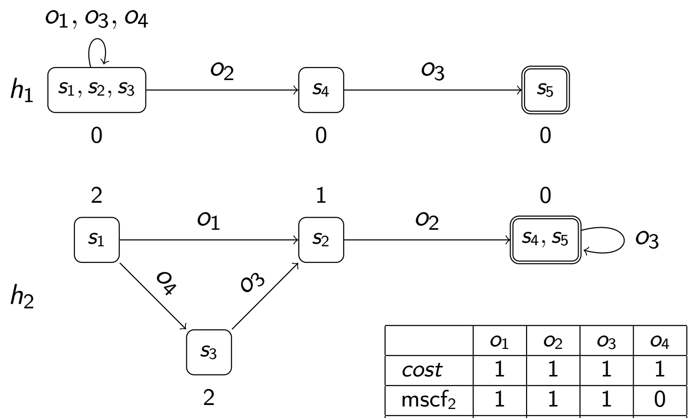
	o_1	o_2	o_3	o_4
cost	0	0	0	0
mscf_1	0	1	1	0
mscf_2	1	0	0	1

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

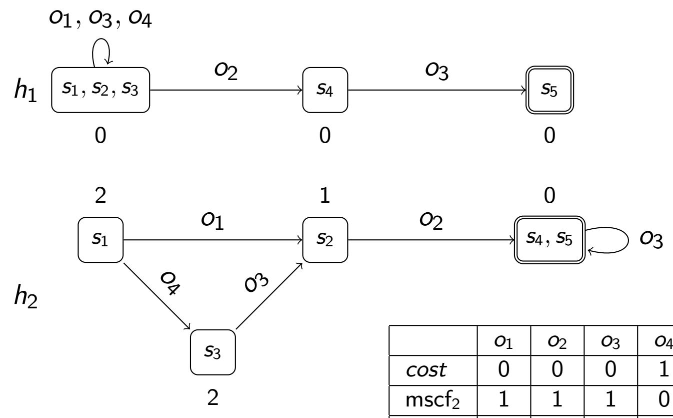
Consider the abstraction heuristics h_1 and h_2



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

Saturated Cost Partitioning: Order

Consider the abstraction heuristics h_1 and h_2



	o_1	o_2	o_3	o_4
cost	0	0	0	1
mscf_2	1	1	1	0
mscf_1	0	0	0	0

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

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

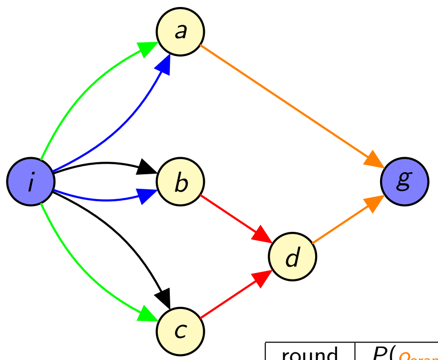
Definition (MSCF for Disjunctive Action Landmark)

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

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

Does this look familiar?

Reminder: LM-Cut



$o_{\text{blue}} = \langle\{i\}, \{a, b\}, \{\}, 4\rangle$
$o_{\text{green}} = \langle\{i\}, \{a, c\}, \{\}, 5\rangle$
$o_{\text{black}} = \langle\{i\}, \{b, c\}, \{\}, 3\rangle$
$o_{\text{red}} = \langle\{b, c\}, \{d\}, \{\}, 2\rangle$
$o_{\text{orange}} = \langle\{a, d\}, \{g\}, \{\}, 0\rangle$

round	$P(o_{\text{orange}})$	$P(o_{\text{red}})$	landmark	cost
1	d	b	$\{o_{\text{red}}\}$	2
2	a	b	$\{o_{\text{green}}, o_{\text{blue}}\}$	4
3	d	c	$\{o_{\text{green}}, o_{\text{black}}\}$	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

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

Does this look familiar?

LM-Cut computes SCP over disjunctive action landmarks

E5.4 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.
- Saturated cost partitioning offers good tradeoff between computation time and heuristic guidance
- LM-Cut computes SCP over disjunctive action landmarks