Algorithms and Satisfiability

12. Combining Multiple Heuristic Functions

“Looking at the same problem from Multiple Angles”

Álvaro Torralba

AALBORG UNIVERSITET

Spring 2023

Agenda

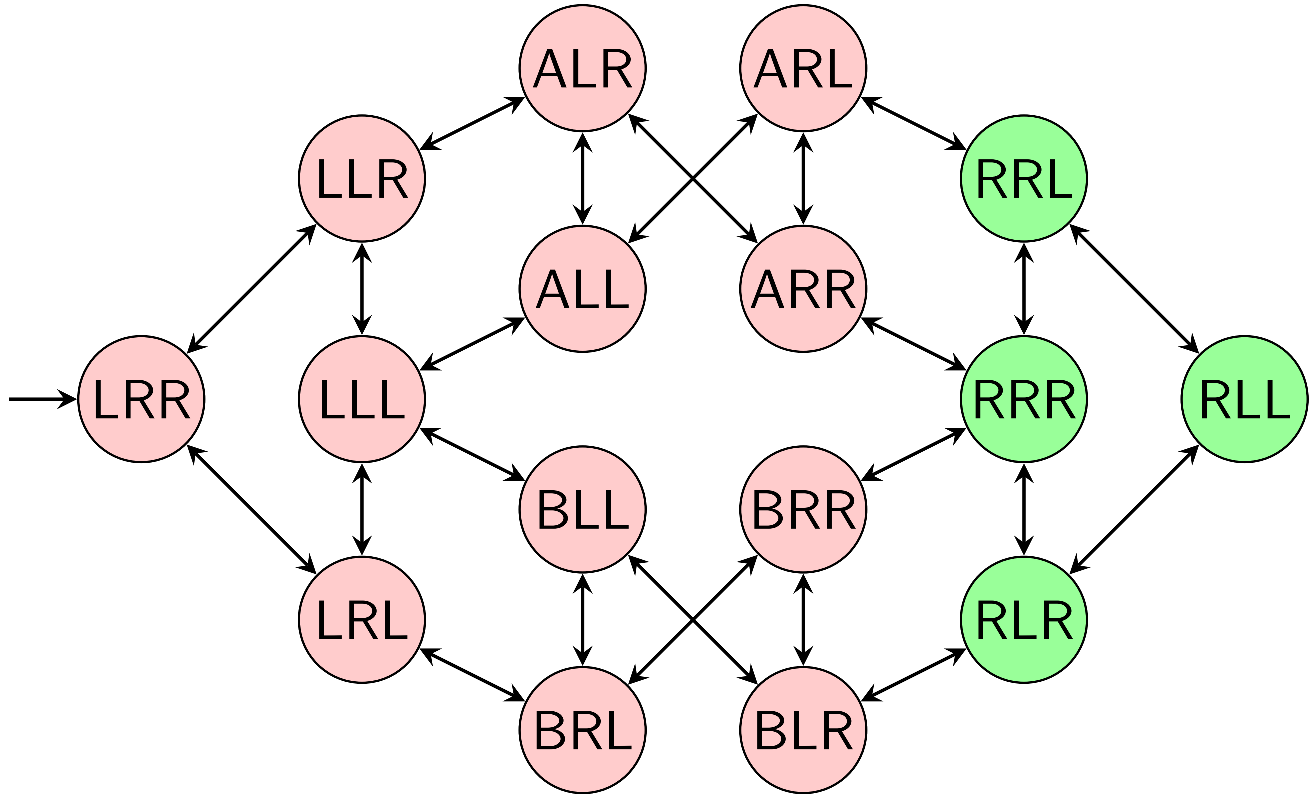
Solving Techniques

So far, we have seen a number of approaches to solve our problems:

1. Reduction: Can we reduce our problem to another problem and use existing algorithms?
   Example: encoding problems in SAT and/or planning (Mini-projects)
2. Incomplete Reduction:
   Example: Encode planning (bounding the number of layers in our plan L) into SAT. If no solution exists, increase L (Chapter 11).
3. Relaxation: Transform your problem into a relaxed problem, use the result to guide a search process. (Chapter 10).

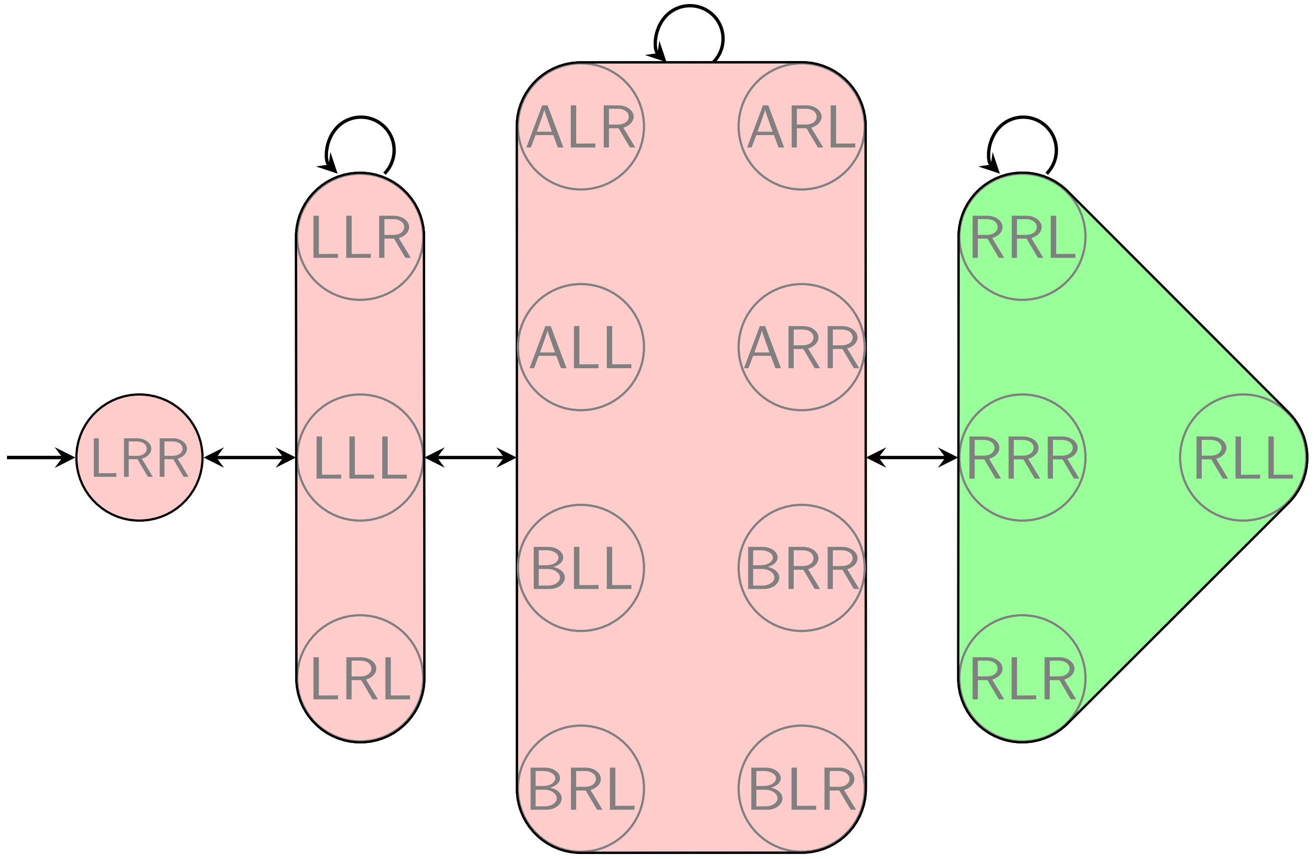
Abstractions in a Nutshell: Example

Concrete transition system: (of “Logistics mal anders”)

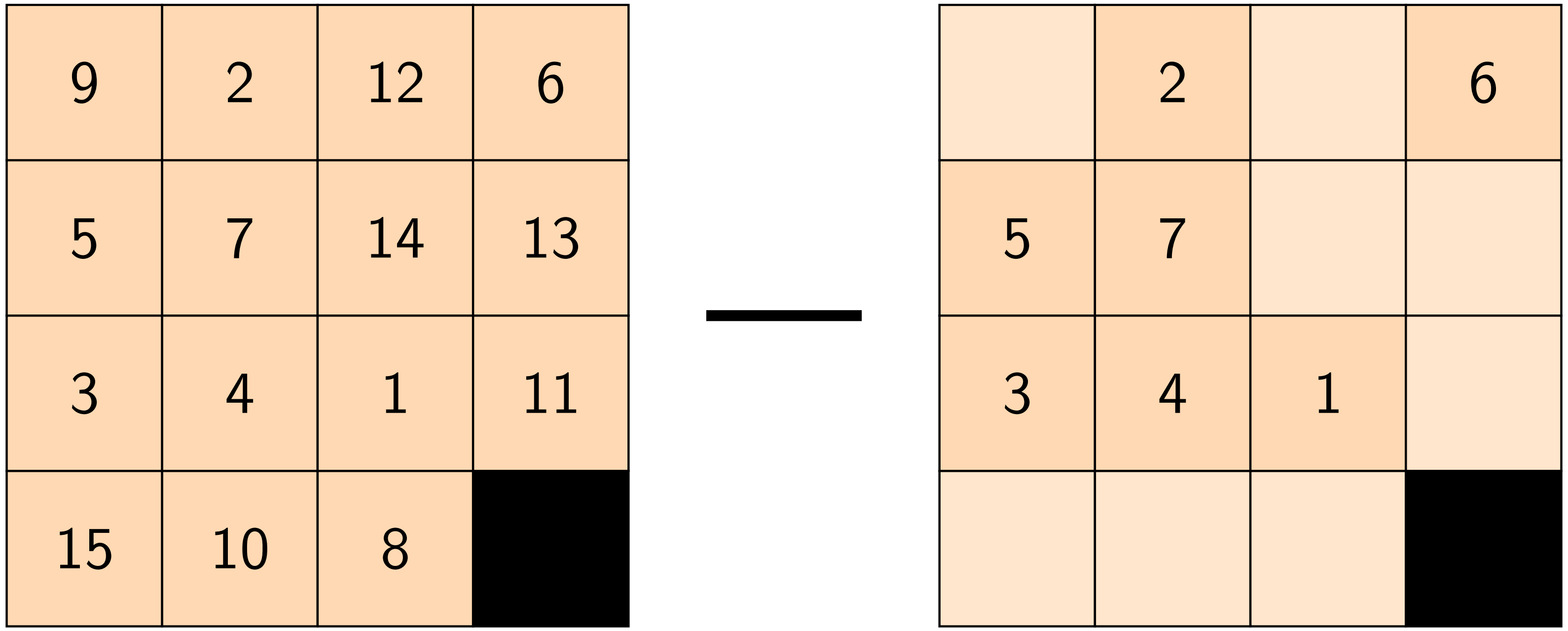


Abstractions in a Nutshell: Example

Abstract transition system: (of “Logistics mal anders”)



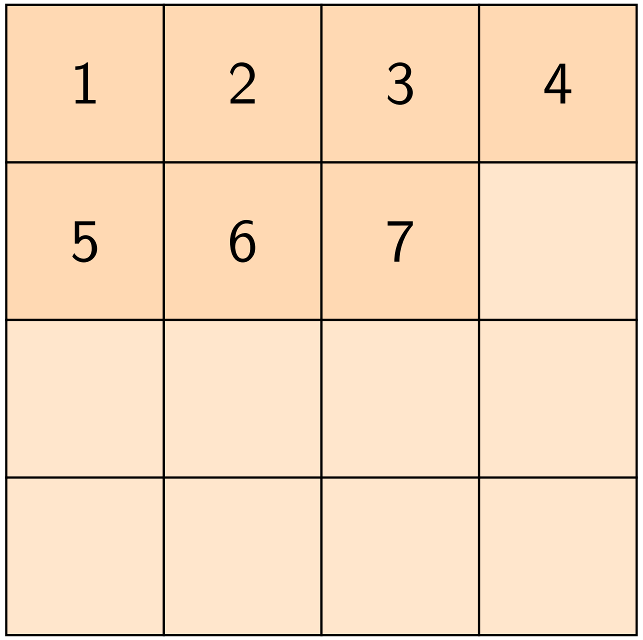
Pattern Databases in a Nutshell

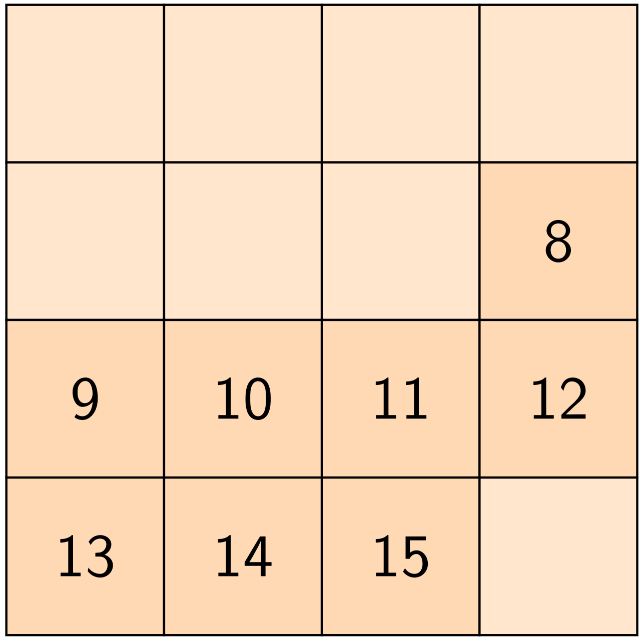


"Abstract the planning task by choosing a subset P of variables (the pattern), and ignoring the values of all other variables."

What is the problem with this Pattern Database heuristic? It completely ignores part of the problem!

From One to Many





Idea: Use multiple (Pattern Database Abstraction) heuristics!
Yet a new technique for problem solving:

- Decompose your problem into multiple sub-problems and combine the solutions
- Here, we cannot combine the plans directly (as they may be incompatible), but we can combine the heuristic values!

Multiple Abstractions

- One important practical question is how to come up with a suitable abstraction mapping α.
- Indeed, there is usually a huge number of possibilities, and it is important to pick good abstractions (i.e., ones that lead to informative heuristics).
- However, it is generally not necessary to commit to a single abstraction.

Combining Multiple Abstractions

Maximizing over several abstractions:

- Each abstraction mapping gives rise to an admissible heuristic.
- By computing the maximum of several admissible heuristics, we obtain another admissible heuristic which dominates the component heuristics.
- Thus, we can always compute several abstractions and maximize over the individual abstract goal distances.

Example (15-Puzzle)

- mapping to tiles 1–7 was arbitrary
  $\rightsquigarrow$ can use any subset of tiles
- with the same amount of memory required for the tables for the mapping to tiles 1–7, we could store the tables for nine different abstractions to six tiles and the blank
- use maximum of individual estimates

Additive Abstractions

Adding several abstractions:

- In some cases, we can even compute the sum of individual estimates and still stay admissible.
- Summation often leads to much higher estimates than maximization, so it is important to understand under which conditions summation of heuristics is admissible.

Adding Several Abstractions: Example

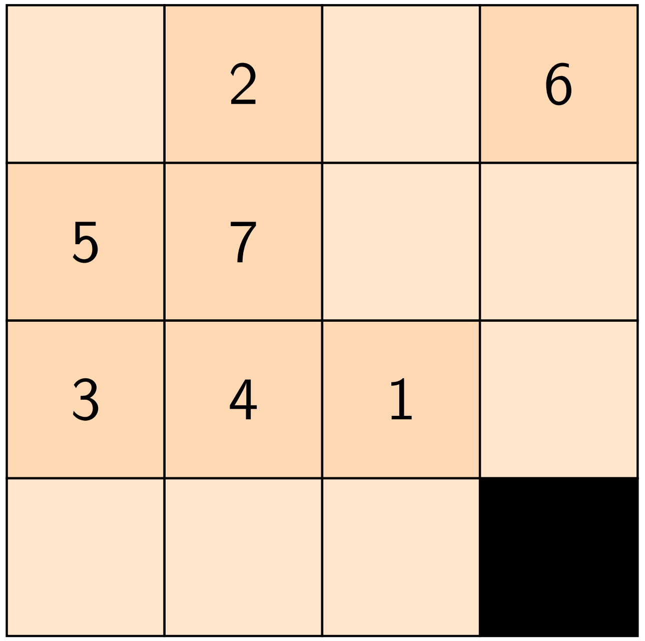
9	2	12	6
5	7	14	13
3	4	1	11
15	10	8	■

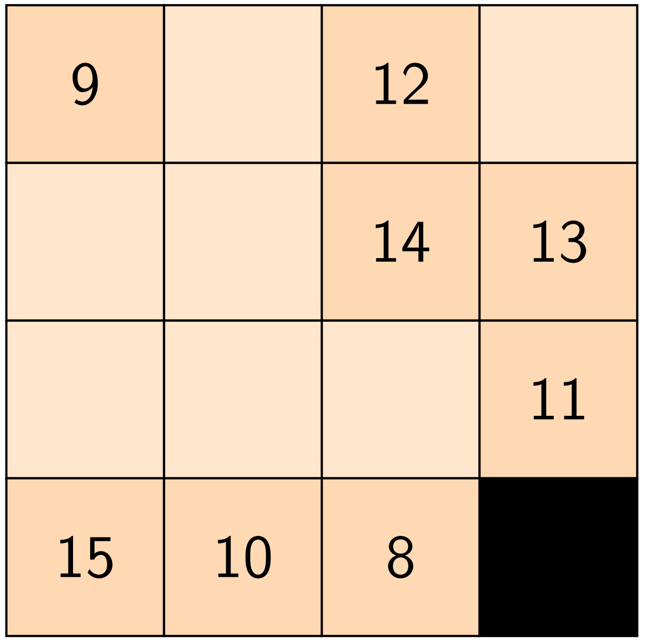
9	2	12	6
5	7	14	13
3	4	1	11
15	10	8	■

- 1st abstraction: ignore precise location of 8–15
- 2nd abstraction: ignore precise location of 1–7

$\rightsquigarrow$ Is the sum of the abstraction heuristics admissible?

Adding Several Abstractions: Example

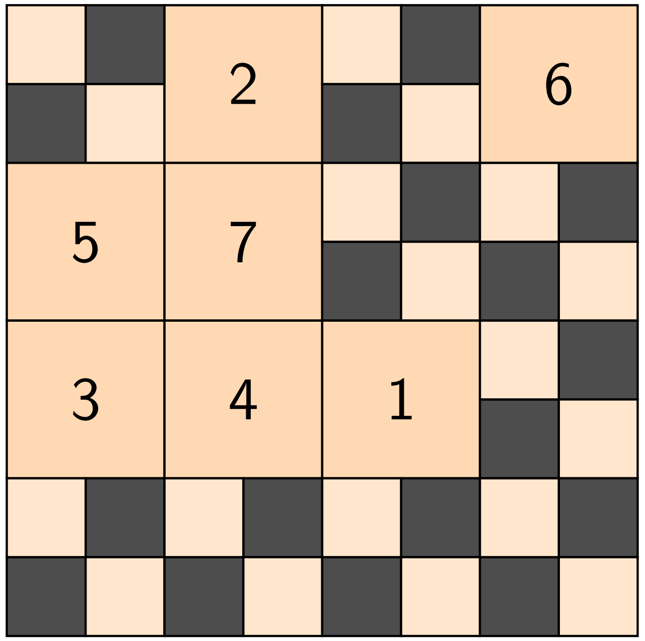


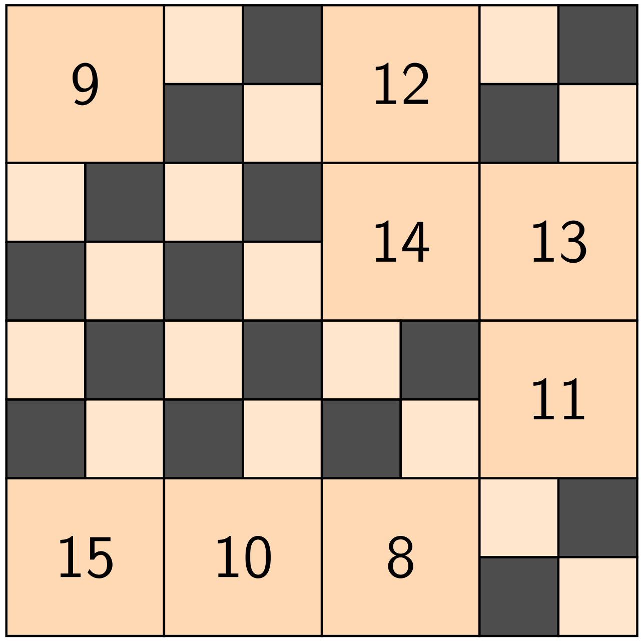


- 1st abstraction: ignore precise location of 8–15
- 2nd abstraction: ignore precise location of 1–7

$\rightsquigarrow$ The sum of the abstraction heuristics is not admissible.

Adding Several Abstractions: Example





- 1st abstraction: ignore precise location of 8–15 and blank
- 2nd abstraction: ignore precise location of 1–7 and blank

$\rightsquigarrow$ The sum of the abstraction heuristics is admissible.

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

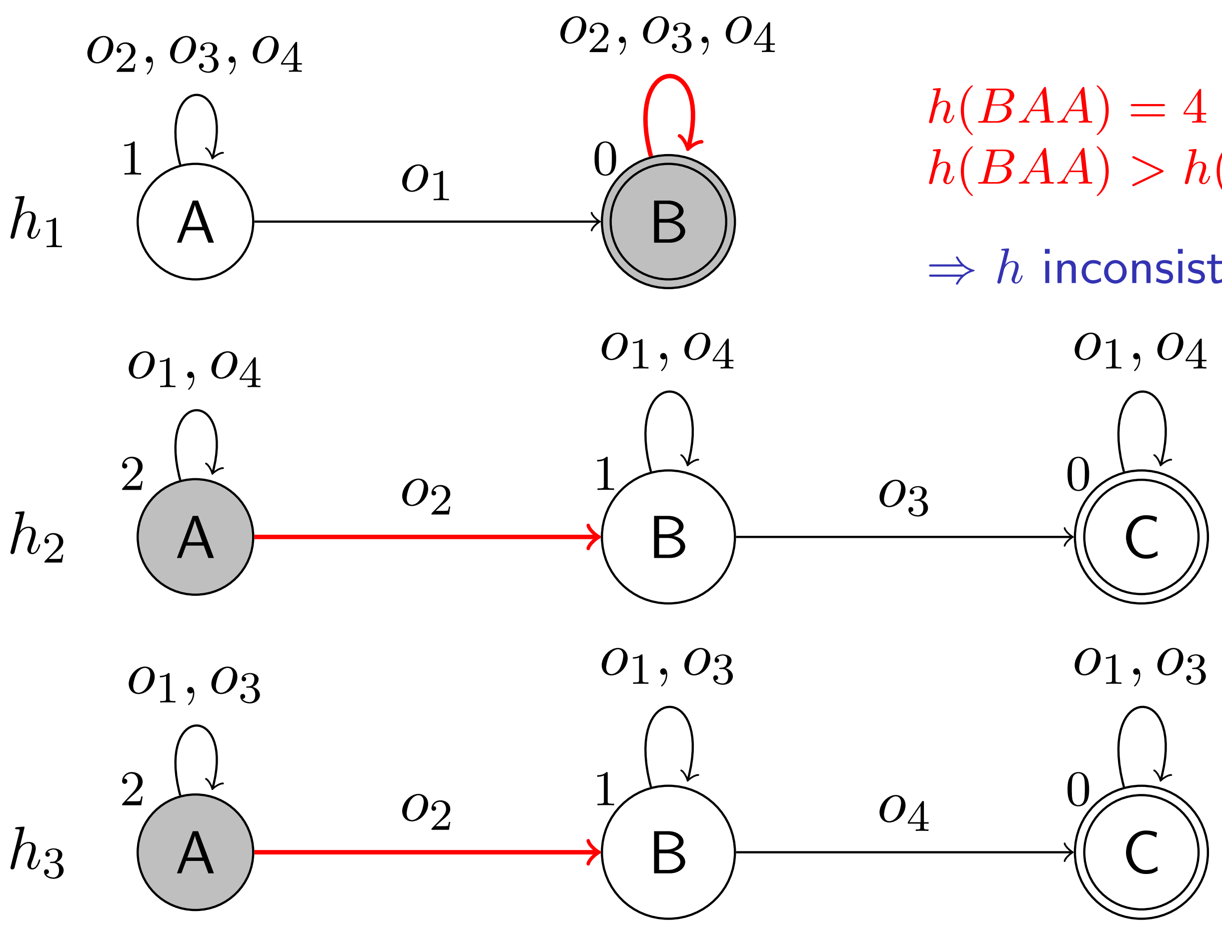
Consider an FDR planning task $\langle V, I, \{o_1, o_2, o_3, o_4\}, \gamma\rangle$ with $V = \{v_1, v_2, v_3\}$ with $dom(v_1) = \{A, B\}$ and $\text{dom}(v_2) = \text{dom}(v_3) = \{A, B, C\}$, $I = \{v_1 \mapsto A, v_2 \mapsto A, v_3 \mapsto A\}$,

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

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

Combining Heuristics Admissibly: Example

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



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

$\Rightarrow$ h inconsistent and inadmissible

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

Our Agenda for This Chapter

2. **Orthogonality**: A simple criteria to combine multiple abstractions.
3. **Cost Partitioning:** We introduce the concept, illustrate it, and prove admissibility of the partitioned sum.
4. **Saturated Cost-Partitioning**: A practical strategy for computing cost partitionings.
5. **Optimal Cost Partitioning** We show how, for abstraction heuristics, we can always find *the best possible* cost partitioning in polynomial time.

Orthogonal Abstractions

Terminology: If $s = t$ in (s, l, t), then the transition is called a self-loop.

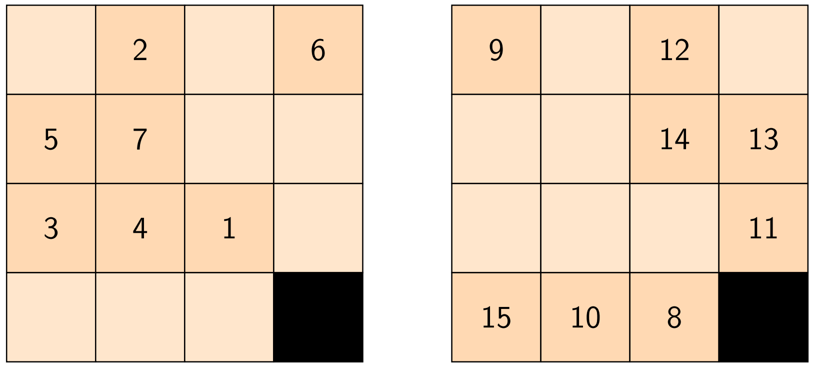
Definition (Affecting Transition Labels). *Let α be an abstraction of Θ, and let l be one of the labels in Θ. We say that l affects α if Θ^α has at least one non-self-loop transition labeled by l, i.e., if there exists a transition $(\alpha(s), l, \alpha(t))$ with $\alpha(s) \neq \alpha(t)$.*

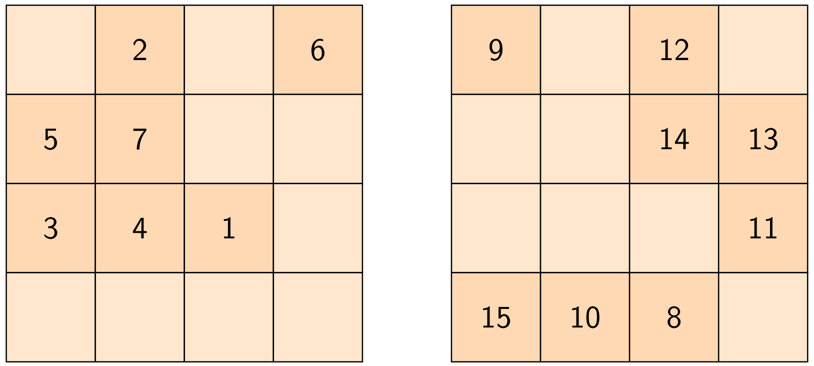
$\rightarrow$ Here is a simple sufficient criterion for additivity:

Definition (Orthogonal Abstractions). *Let α_1 and α_2 be abstractions of Θ. We say that α_1 and α_2 are orthogonal if no label of Θ affects both α_1 and α_2.*

Orthogonal Abstractions: Example 15-Puzzle

Reminder: A label affects α if it labels a non-self loop transition in Θ^{α}. We say that α_1 and α_2 are orthogonal if no label of Θ affects both α_1 and α_2.





$\rightarrow$ Are the left-hand side abstraction mappings α_{left} and α_{right} orthogonal? No. E.g., consider the action that moves the blank upwards here, mapping the current state s to state t. This transition is not a self-loop in either of the two abstractions: $\alpha_{\text{left}}(s) \neq \alpha_{\text{left}}(t)$ and $\alpha_{\text{right}}(s) \neq \alpha_{\text{right}}(t)$.

$\rightarrow$ Are the right-hand side abstraction mappings α_{left} and α_{right} orthogonal? Yes. Say a is any action that affects α_{left}. Then a moves a tile t_i for $i \in \{1, \dots, 7\}$. Neither that t_i nor the blank are accounted for in α_{right} so a labels only self-loops there. Same vice versa.

Orthogonality and Additivity

Theorem (Orthogonal Abstractions are Additive). *Let $\alpha_1, \dots, \alpha_n$ be pairwise orthogonal abstractions for the same transition system Θ. Then $\sum_{i=1}^{n} h^{\alpha_i}$ is consistent and goal-aware, and thus also admissible and safe.*

$\rightarrow$ Intuition for admissibility: "Self-loops don't count." Every transition in an optimal solution path affects at most one of the abstractions, and thus is counted in at most one of the abstraction heuristics.

Cost Partitioning

Definition (Cost Partitioning)

Let Π be a planning task with operators O.

A cost partitioning for Π is a tuple $\langle c_1, \dots, c_n \rangle$, where

- $c_i : O \to \mathbb{R}_0^+$ for $1 \leq i \leq n$ and
- $\sum_{i=1}^{n} c_i(o) \leq c(o)$ 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 $c_i(o)$.

Solution: Cost partitioning

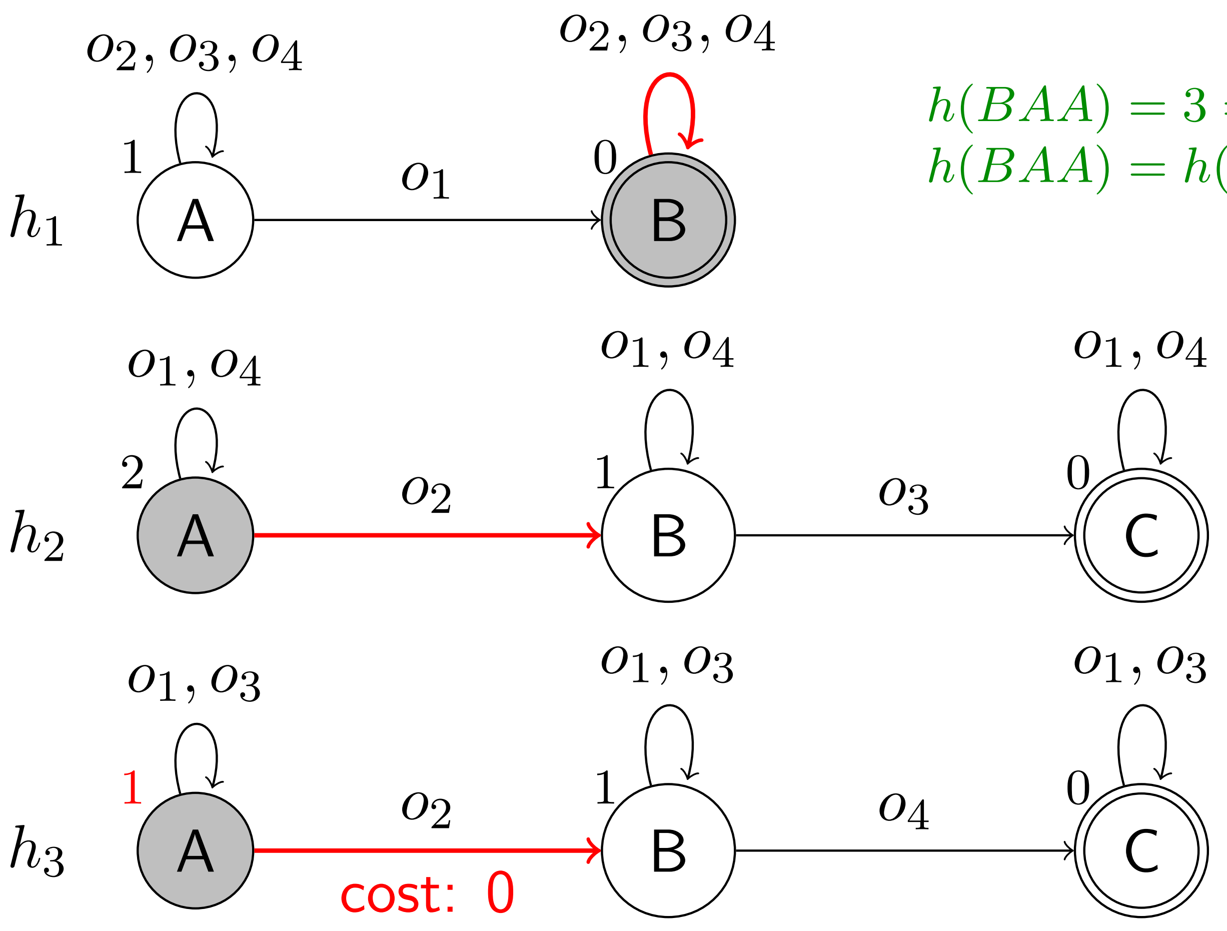
h is not admissible because $c(o_2)$ is considered in h_2 and h_3

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 $c_3(o_2) = 0$



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

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

Solution: Cost partitioning

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

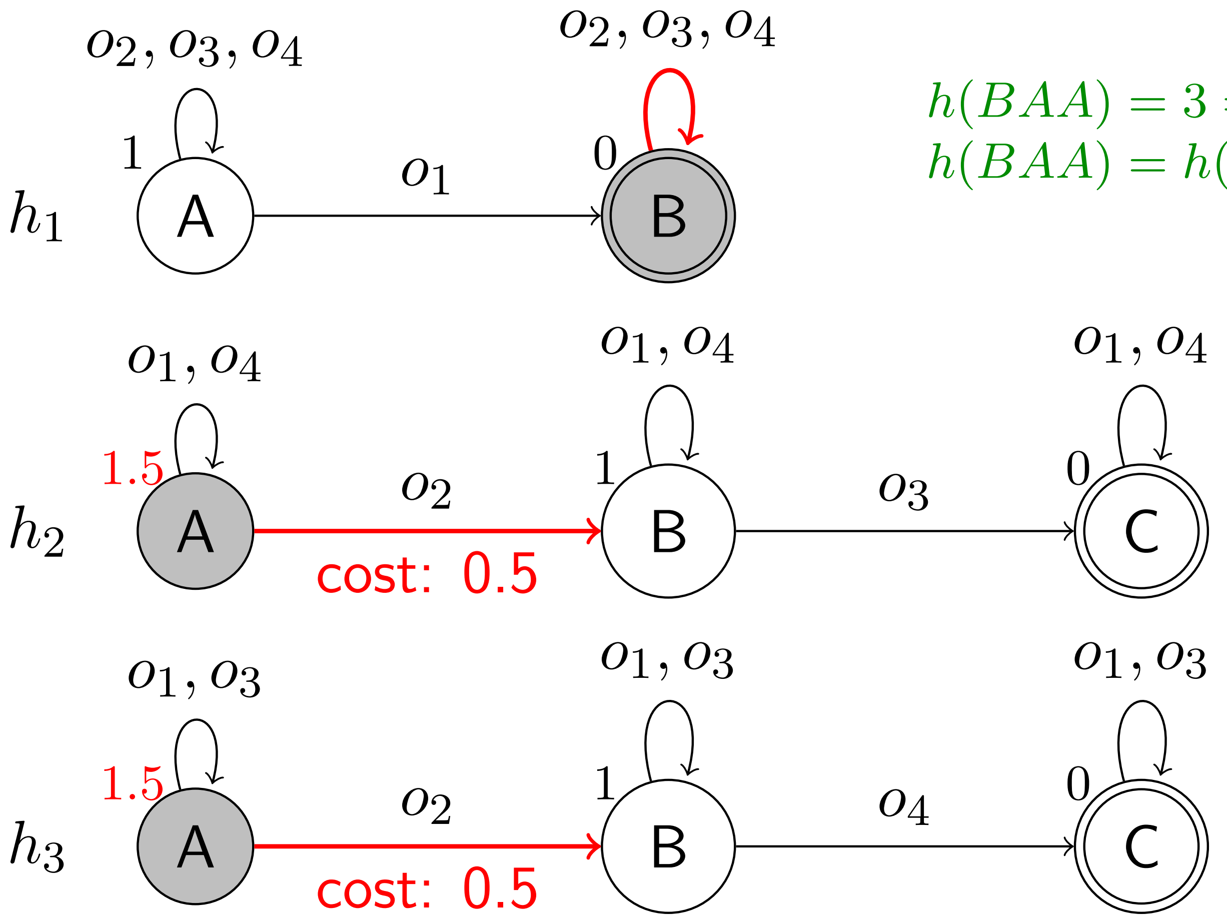
Is there anything we can do about this?

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 $c_2(o_2) = c_3(o_2) = \frac{1}{2}$



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

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

Solution: Cost partitioning

h is not admissible because $c(o_2)$ 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

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

What about o_1, o_3 and o_4?

Cost Partitioning: Admissibility

Theorem (Sum of Solution Costs is Admissible)

Let Π be a planning task, $\langle c_1, \dots, c_n \rangle$ 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 Preserves Admissibility

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 $\langle \Pi_1, \ldots, \Pi_n \rangle$ be induced by a cost partitioning.

For admissible heuristics $h_1, \ldots, h_n$, 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

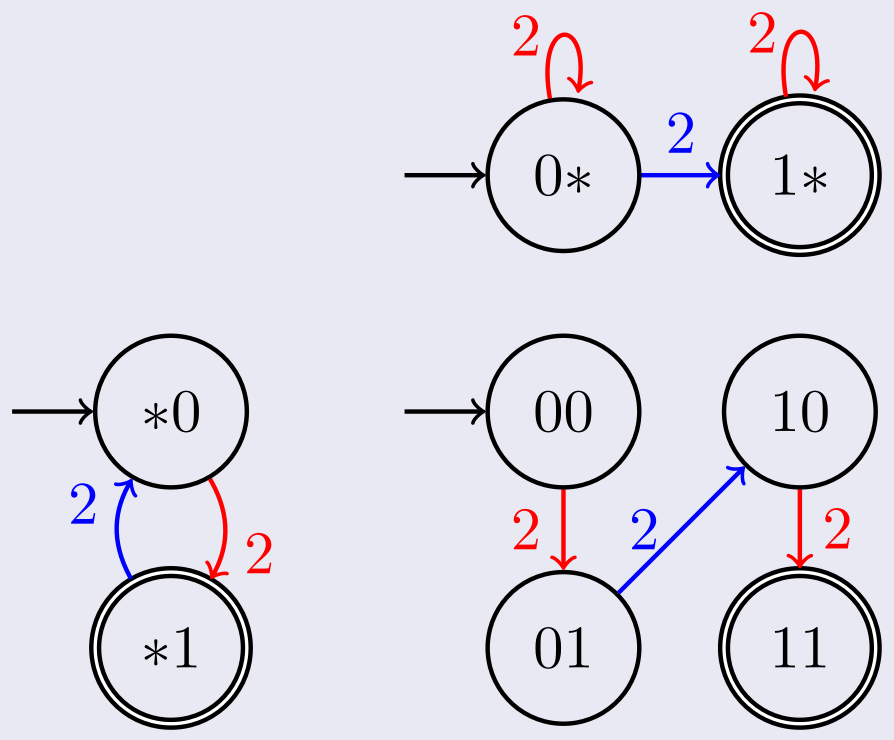
Theorem (Cost Partitioning Preserves Consistency)

Let Π be a planning task and let $\langle \Pi_1, \dots, \Pi_n \rangle$ be induced by a cost partitioning $\langle c_1, \dots, c_n \rangle$.

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.

Cost Partitioning: Example

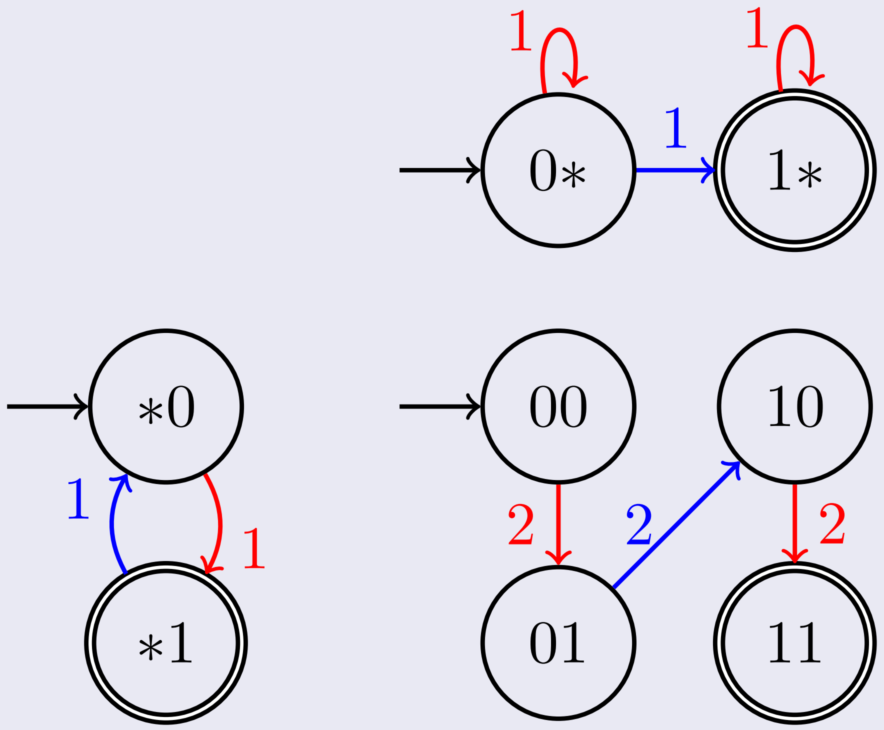
Example (No Cost Partitioning)



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

Cost Partitioning: Example

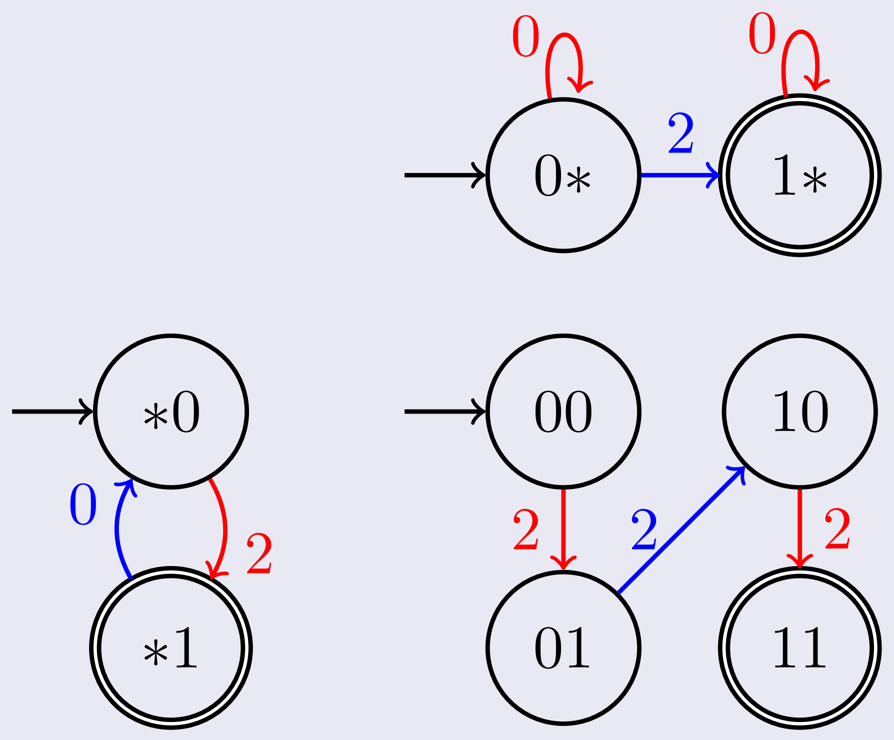
Example (Cost Partitioning 1)



Heuristic value: $1 + 1 = 2$

Cost Partitioning: Example

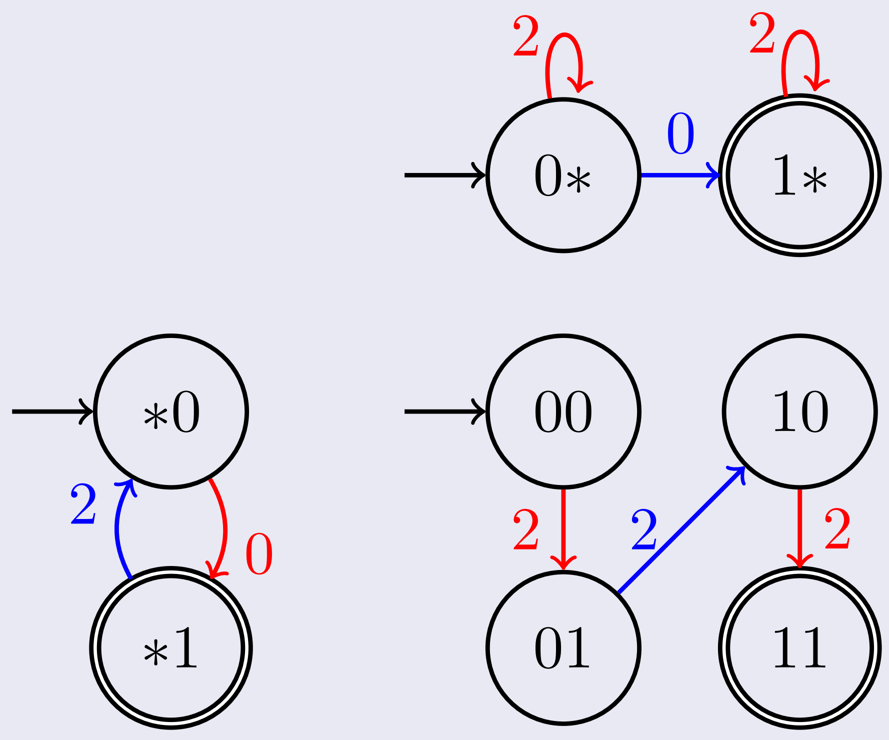
Example (Cost Partitioning 2)



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 (afterwards)

Not So Simple: The Attack of the Zombie Tomatoes

Planning task: Goal: A and B both true. Initial state: A and B both false. Actions: $carA$ effect A cost 1; $carB$ effect B cost 1; $fancyCar$ effect A and B cost 1.5.

Heuristics: h_1: PDB with $P = \{A\}$; h_2: PDB with $P = \{B\}$.

A
RETURN OF THE KILLER TOMATOES
A
AB
SIE KOMMEN NICHT IN FRIEDEN!
LEBENDE TOTE AUS DEM WELTALL!!
ZOMBIES FROM OUTER SPACE
B
B

$\rightarrow$ They are not orthogonal due to the fancyCar action!

Question

What is a good cost-partitioning here?

$c_1(carA) = 1$, $c_2(carA) = 0$; $c_1(carB) = 0$, $c_2(carB) = 1$;
$c_1(fancyCar) = 0.75$, $c_2(fancyCar) = 0.75$.
Then, $h_1[c_1] = h_2[c_2] = 0.75$, so the sum is 1.5!

Dominating Orthogonal PDBs

Reminder:

An action a affects a projection π_P if there exists a variable $v \in P$ on which eff_a is defined. Patterns $P_1, \dots, P_n$ are orthogonal if every action affects at most one P_i. Then, $\sum_{i=1}^{n} h^{P_i}$ is admissible.

Theorem (Cost Partitionings Can Dominate the Sum of Orthogonal PDBs). *Let Π be a planning task, and let $\{P_1, \dots, P_n\}$ be an orthogonal pattern collection. For each i and $a \in A$, define $c_i(a) := c(a)$ if a affects α_i, and $c_i(a) := 0$ otherwise. Then $c_1, \dots, c_n$ is a cost partitioning, and for all states s we have $\sum_{i=1}^{n} h^{P_i}(s) = \sum h[c](s)$.*

$\rightarrow$ Orthogonality for PDBs is subsumed by "0/1" cost partitionings, putting the entire cost of each action into the PDB it affects.

($\rightarrow$ This works for arbitrary abstractions, not just for PDBs.)

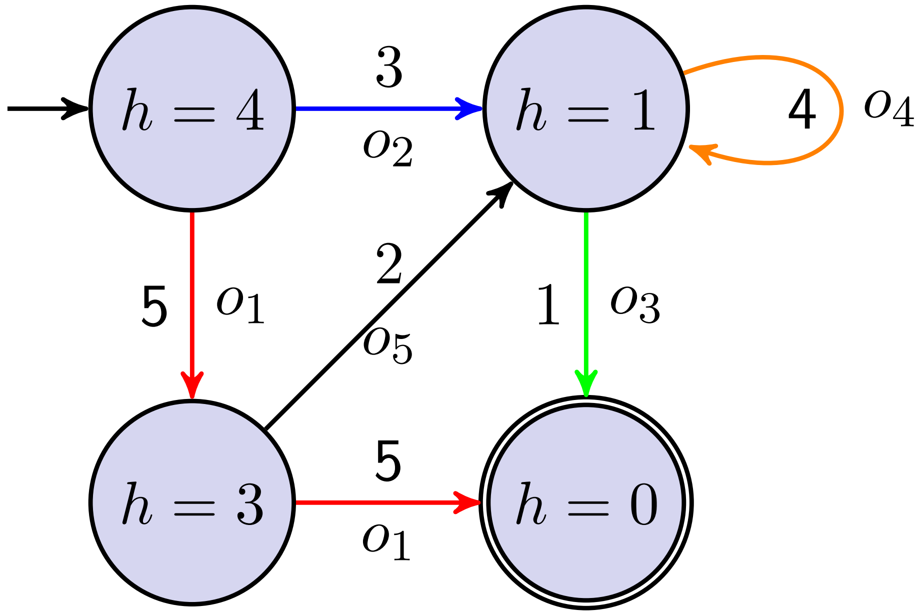
Idea

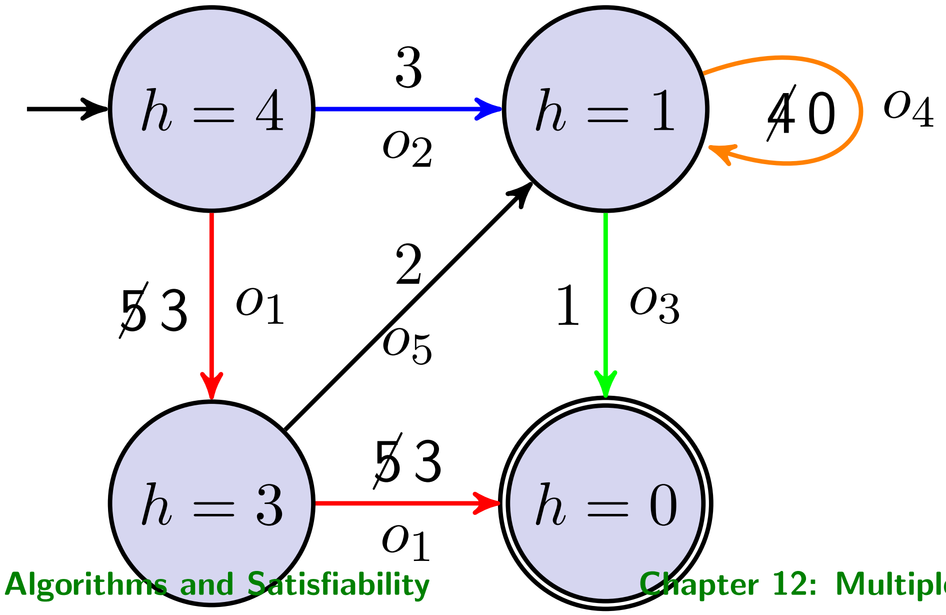
Heuristics do not always "need" all operator costs

- 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 Example





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 c if

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

How much cost we actually need to assign to the current abstraction, while keeping the same heurisitc values?

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:

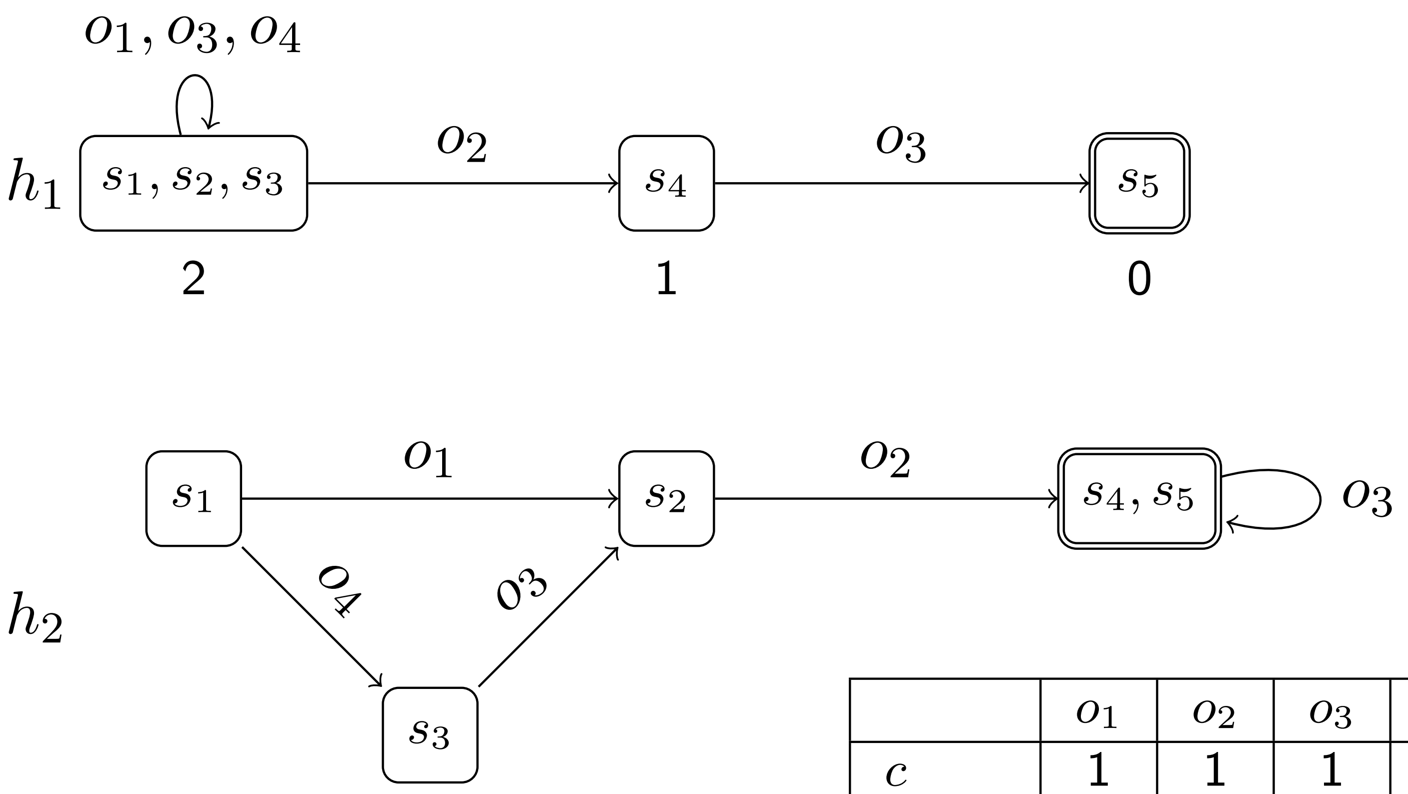
1. Pick a heuristic h_i that hasn't been picked before. Terminate if none is left.
2. Compute h_i given current c
3. Compute minimal saturated cost function mscf_i for h_i
4. Decrease $c(o)$ by $\text{mscf}_i(o)$ for all operators o

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

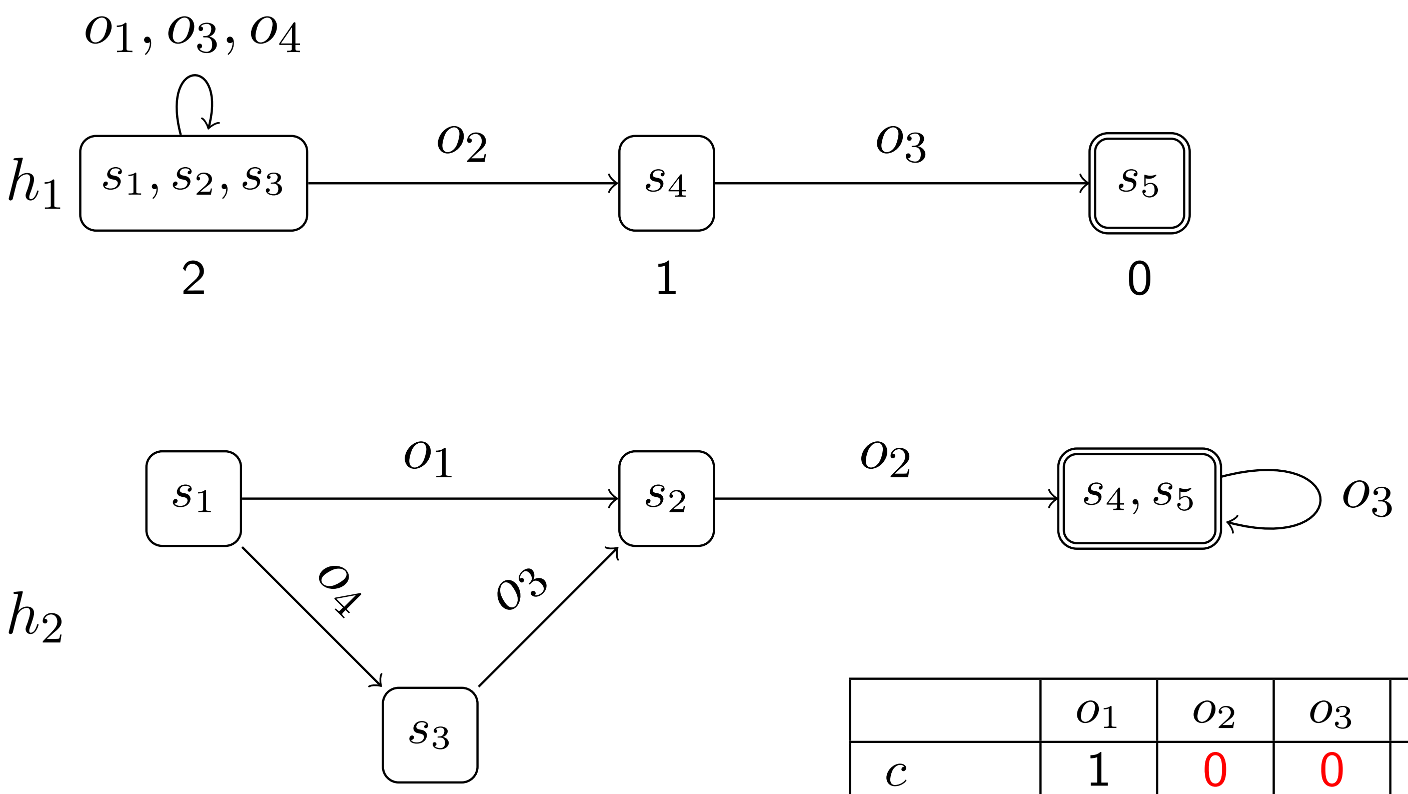


	o_1	o_2	o_3	o_4
c	1	1	1	1
mscf$_1$	0	1	1	0

Example

Consider the abstraction heuristics h_1 and h_2

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

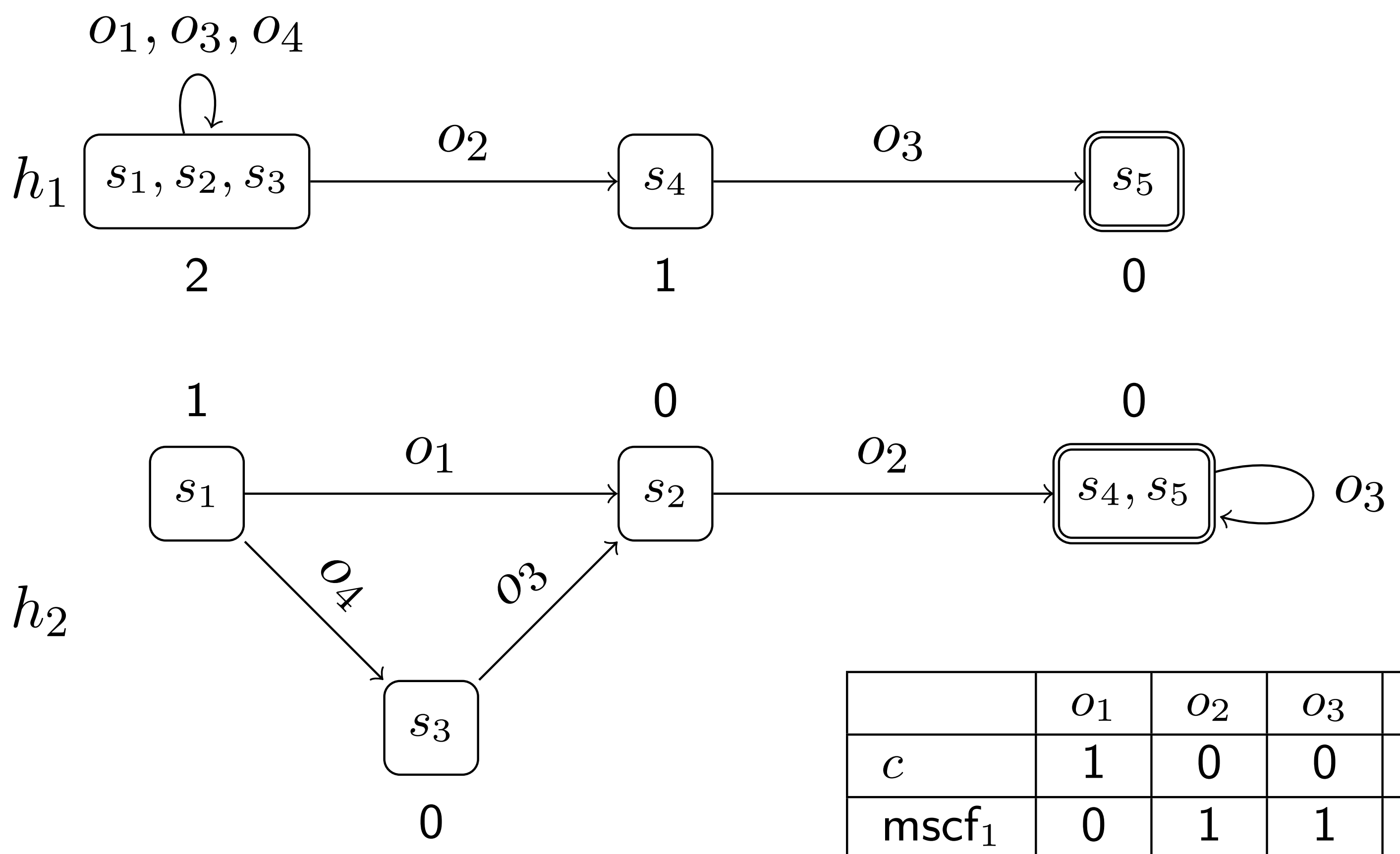


	o_1	o_2	o_3	o_4
c	1	0	0	1
mscf_1	0	1	1	0

Example

Consider the abstraction heuristics h_1 and h_2

3. Compute minimal saturated cost function mscf_i for h_i

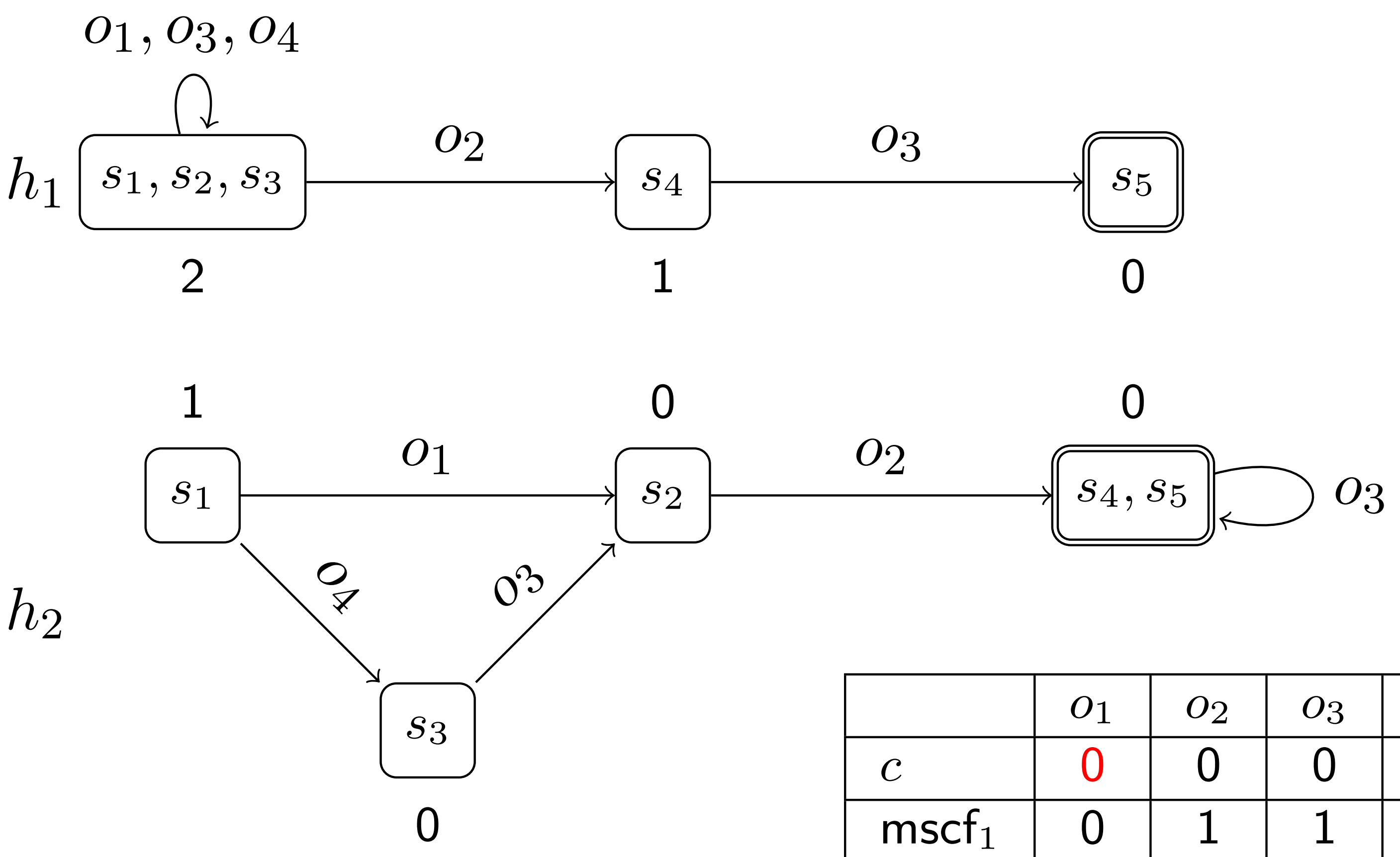


	o_1	o_2	o_3	o_4
c	1	0	0	1
mscf_1	0	1	1	0
mscf_2	1	0	0	1

Example

Consider the abstraction heuristics h_1 and h_2

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

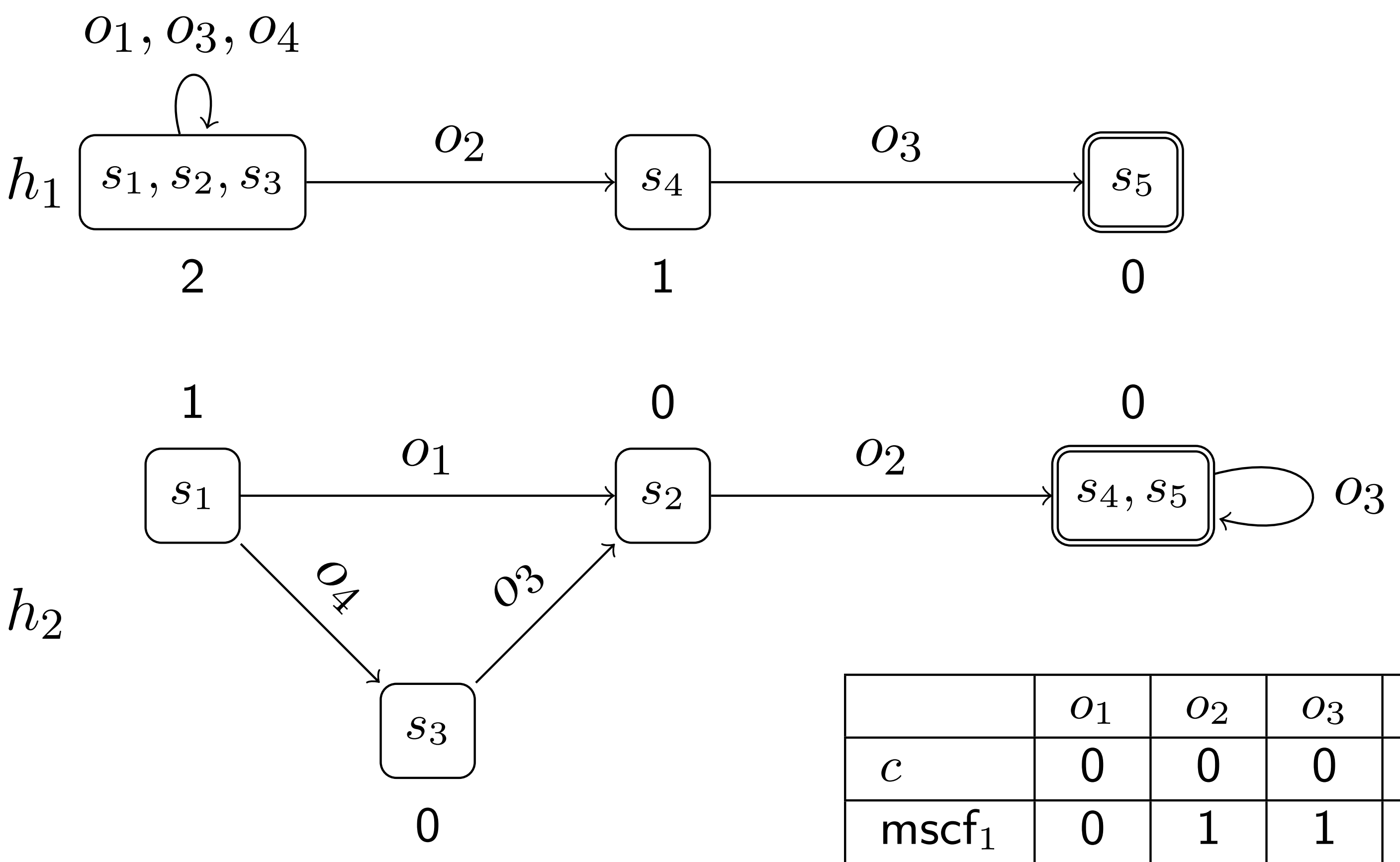


	o_1	o_2	o_3	o_4
c	0	0	0	0
mscf_1	0	1	1	0
mscf_2	1	0	0	1

Example

Consider the abstraction heuristics h_1 and h_2

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



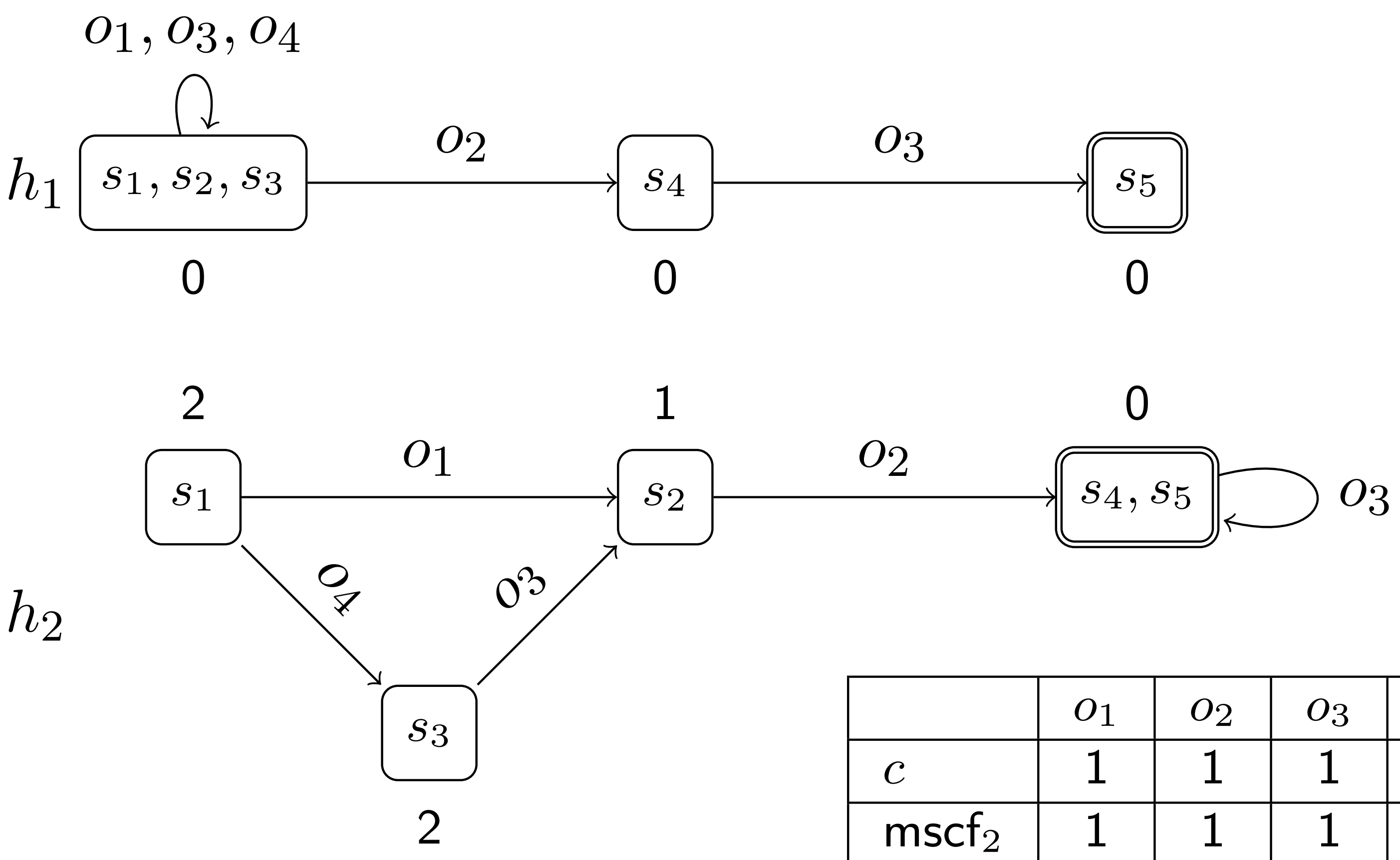
	o_1	o_2	o_3	o_4
c	0	0	0	0
mscf$_1$	0	1	1	0
mscf$_2$	1	0	0	1

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

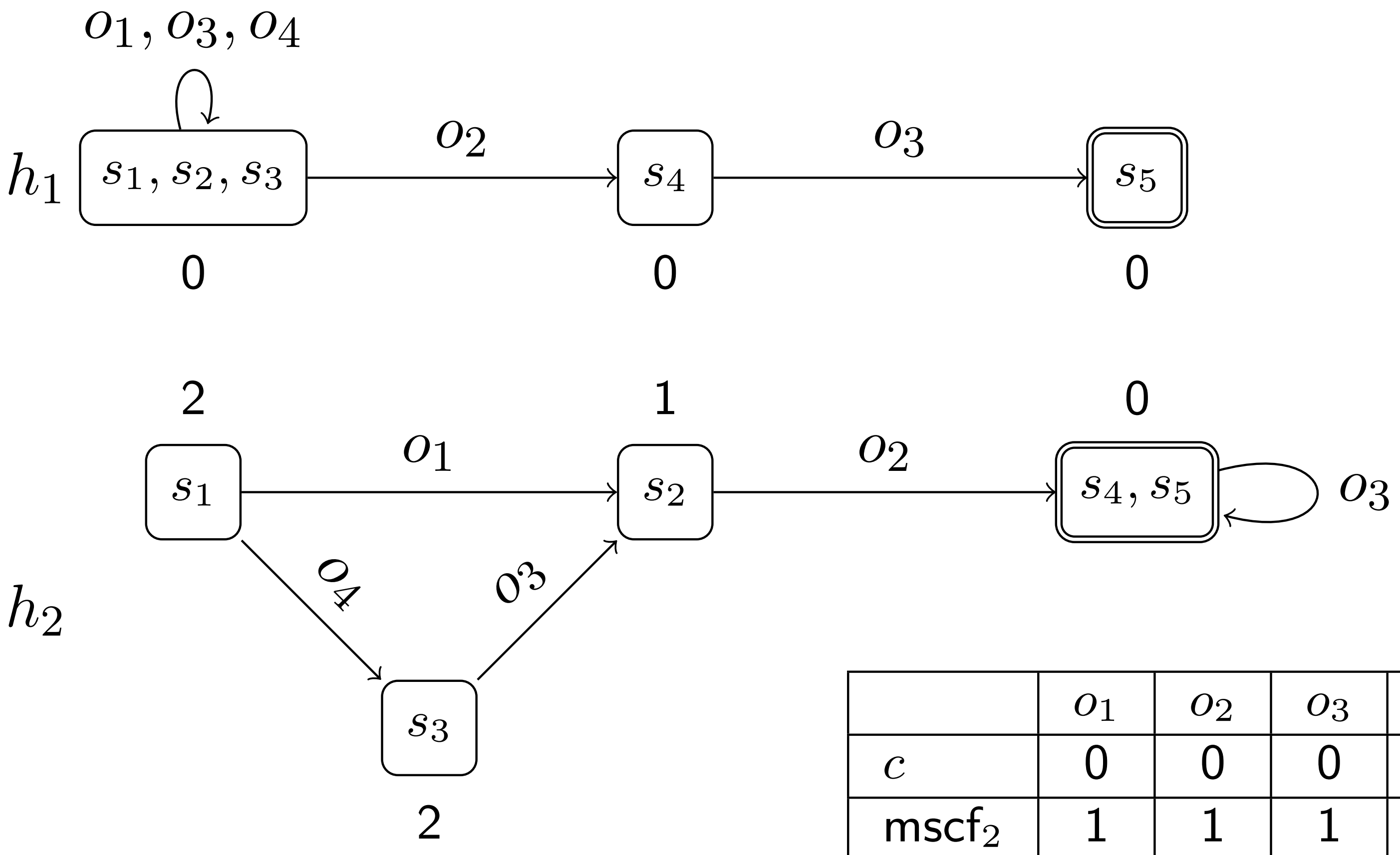
Consider the abstraction heuristics h_1 and h_2



	o_1	o_2	o_3	o_4
c	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
c	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

Optimal Cost Partitioning

Definition (Optimal Cost Partitioning). *Let Π be a planning task, let $h_1, \dots, h_n$ be admissible heuristic functions for Π, and let s be a state. An optimal cost partitioning for s and $h_1, \dots, h_n$ is any cost partitioning $c_1, \dots, c_n$ for which $\sum h[c](s)$ is maximal.*

$\rightarrow$ Optimal cost partitionings distribute costs in a way that yields the best possible lower bound, for a given state.

Question!

Does this definition sound completely impractical?

(A): Yes (B): No

$\rightarrow$ Yes it does. However, it isn't! In many cases, we can compute an optimal cost partitioning efficiently.

Optimal Cost Partitionings in Polynomial Time

How do we design an algorithm that solves this problem in polynomial time?

We don't! Reduce our problem into a problem with known polynomial-time complexity: Linear Programming

$$\begin{aligned} \text{maximize } & x + y - 2z \\ \text{subject to } & x - z < 10 \\ & z - y \geq 20 \end{aligned}$$

The objective function and constraints are linear (so no x^2 or $x \cdot y$ allowed, for example)

LP for Shortest Path in State Space

Variables

Non-negative variable Distance_s for each state s

Objective

Maximize Distance_{s_I}

Subject to

$$\begin{aligned} \text{Distance}_{s_\star} &= 0 && \text{for all goal states } s_\star \\ \text{Distance}_s &\leq \text{Distance}_{s'} + c(o) && \text{for all transitions } s \xrightarrow{o} s' \end{aligned}$$

Optimal Cost Partitioning for Abstractions

Variables

For each abstraction α:

Non-negative variable $\text{Distance}^{\alpha}_{s}$ for each abstract state s,

Non-negative variable Cost^{α}_{o} for each operator o

Objective

Maximize $\sum_{\alpha} \text{Distance}^{\alpha}_{\alpha(s_I)}$

Subject to

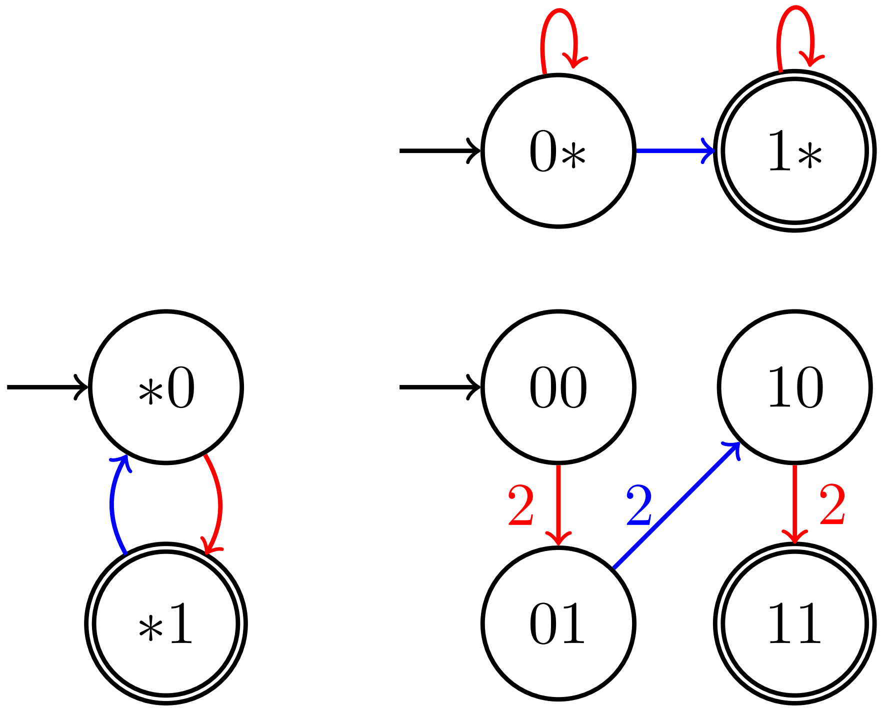
$\sum_{\alpha} \text{Cost}^{\alpha}_{o} \leq c(o)$ for all operators o

and for all abstractions α

$\text{Distance}^{\alpha}_{s_\star} = 0$ for all abstract goal states $s_\star$

$\text{Distance}^{\alpha}_{s} \leq \text{Distance}^{\alpha}_{s'} + \text{Cost}^{\alpha}_{o}$ for all transition $s \xrightarrow{o} s'$

Example (1)



Example (2)

Maximize $\text{Distance}_0^1 + \text{Distance}_0^2$ subject to

$$\begin{aligned}
\text{Cost}_{\text{red}}^1 + \text{Cost}_{\text{red}}^2 &\leq 2 \\
\text{Cost}_{\text{blue}}^1 + \text{Cost}_{\text{blue}}^2 &\leq 2 \\
\text{Distance}_1^1 &= 0 \\
\text{Distance}_0^1 &\leq \text{Distance}_0^1 + \text{Cost}_{\text{red}}^1 \\
\text{Distance}_0^1 &\leq \text{Distance}_1^1 + \text{Cost}_{\text{blue}}^1 \\
\text{Distance}_1^1 &\leq \text{Distance}_1^1 + \text{Cost}_{\text{red}}^1 \\
\text{Distance}_1^2 &= 0 \\
\text{Distance}_0^2 &\leq \text{Distance}_1^2 + \text{Cost}_{\text{red}}^2 \\
\text{Distance}_1^2 &\leq \text{Distance}_0^2 + \text{Cost}_{\text{blue}}^2 \\
\text{Distance}_s^\alpha &\geq 0 \quad \text{for } \alpha \in \{1,2\}, s \in \{0,1\} \\
\text{Cost}_o^\alpha &\geq 0 \quad \text{for } \alpha \in \{1,2\}, o \in \{\text{red}, \text{blue}\}
\end{aligned}$$

Caution

A word of warning

- optimization for every state gives best-possible cost partitioning
- but takes time

Better heuristic guidance often does not outweigh the overhead.

Exam Structure

Written Exam: June 6

→ Open book: Exercises won't ask literally about what is in the book, but they will aim to see if you have understood the concepts

- Tip: make sure that you understand the algorithms and concepts well. There won't be enough time to look up everything during the exam!
- You can expect exercises similar to the ones in the exercise sheets (some of the exercises were taken from previous exams)
- Also some questions related to the mini-projects (e.g. knowing about how problems can be encoded in SAT/Z3 and/or STRIPS/PDDL)

Thank you for attending the course!

Summary

- A cost partitioning distributes the cost of each action across n otherwise identical planning tasks. This can be used to admissibly sum up *any* ensemble of admissible heuristic functions.
- For every state and ensemble of PDB heuristics, there exists a cost partitioning that dominates orthogonal PDB heuristics; the domination can be strict.
- Optimal cost partitionings distribute action costs such that the lower bound for a given state is maximal.
- For PDBs and LMs, and for their combination, optimal cost partitionings can be computed in polynomial time by Linear Programming.
- In practice, computing optimal cost partitionings for every search state typically is too costly, and we need to approximate.
- Saturated cost-partitioning divides the cost, by greedily assigning all needed cost for each abstraction in order.

Historical Remarks

- The admissible combination of lower bounds has a long history. Famous instances pertain to additive PDBs in Game playing [**?**].
- In planning, this story also started with additive PDBs [**??**], then was extended to h^m among others [**?**]. The intuition always was to design the heuristics in a way making them *independent*.
- When I was in some project meeting somewhere in about 2005, someone from outside the area said "But what if we count each move only half in each of the heuristics?". The remark was received with confusion, then forgotten about.
- Then Michael & Carmel [**?**] suddenly came along and told us we'd been looking at 0/1 cost partitionings all the time, and how to find optimal general ones efficiently using LP.
- Since then, various works towards making this practical.
- Cost partitioning is not specific to planning, can be applied anywhere!

Reading

- *Optimal Additive Composition of Abstraction-Based Admissible Heuristics* [**?**].

  Available at:

  `http://fai.cs.uni-saarland.de/katz/papers/icaps08b.pdf`

  Content: Original paper proposing cost partitioning, and showing that, for certain classes of heuristics, optimal cost partitionings can be computed in polynomial time using Linear Programming. Specifically, the paper established this for abstractions as handled in this course, as well as for implicit abstractions represented through planning task fragments identified based on the causal graph.

Reading, ctd.

- *Diverse and Additive Cartesian Abstraction Heuristics* [**?**].

  Available at:

  `http://ai.cs.unibas.ch/papers/seipp-helmert-icaps2014.pdf`

  Content: Introduces the current state of the art technique for cost partitioning with abstraction heuristics, saturated cost partitioning, which partitions costs according to what is actually needed to preserve the abstraction heuristic.

References I