

# Planning and Optimization

## F1. Constraints: Introduction

Malte Helmert and Gabriele Röger

Universität Basel

November 27, 2024

# Planning and Optimization

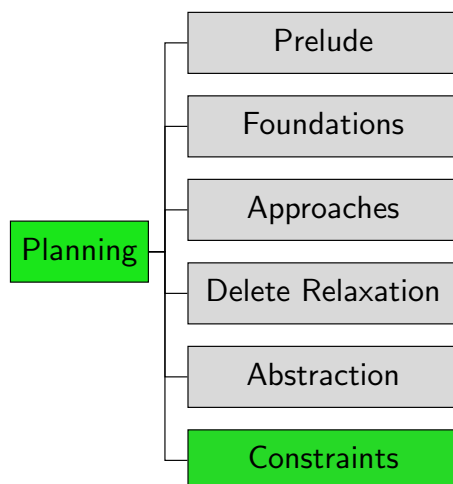
November 27, 2024 — F1. Constraints: Introduction

## F1.1 Constraint-based Heuristics

## F1.2 Multiple Heuristics

## F1.3 Summary

## Content of the Course



## F1.1 Constraint-based Heuristics

## Coming Up with Heuristics in a Principled Way

### General Procedure for Obtaining a Heuristic

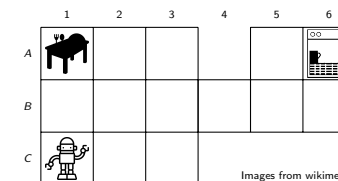
Solve a simplified version of the problem.

Major ideas for heuristics in the planning literature:

- ▶ delete relaxation
- ▶ abstraction
- ▶ critical paths
- ▶ landmarks
- ▶ network flows
- ▶ potential heuristic

Landmarks, network flows and potential heuristics are based on **constraints** that can be specified for a planning task.

## Constraints: Example



FDR planning task  $\langle V, I, O, \gamma \rangle$  with

- ▶  $V = \{robot-at, dishes-at\}$  with
  - ▶  $dom(robot-at) = \{A1, \dots, C3, B4, A5, \dots, B6\}$
  - ▶  $dom(dishes-at) = \{Table, Robot, Dishwasher\}$
- ▶  $I = \{robot-at \mapsto C1, dishes-at \mapsto Table\}$
- ▶ operators
  - ▶ move- $x-y$  to move from cell  $x$  to adjacent cell  $y$
  - ▶ pickup dishes, and
  - ▶ load dishes into the dishwasher.
- ▶  $\gamma = (robot-at = B6) \wedge (dishes-at = Dishwasher)$

Images from wikipedia

## Constraints

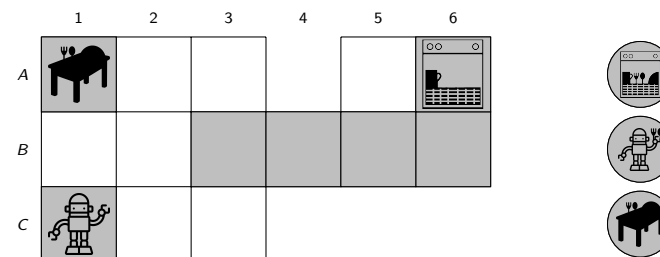
Some heuristics exploit **constraints** that describe something that holds in every solution of the task.

For instance, every solution is such that

- ▶ a variable takes a certain value in at least one visited state.  
(a **fact landmark** constraint)

## Fact Landmarks: Example

Which values do *robot-at* and *dishes-at* take in every solution?



- ▶  $robot-at = C1, dishes-at = Table$  (initial state)
- ▶  $robot-at = B6, dishes-at = Dishwasher$  (goal state)
- ▶  $robot-at = A1, robot-at = B3, robot-at = B4, robot-at = B5, robot-at = A6, dishes-at = Robot$

## Constraints

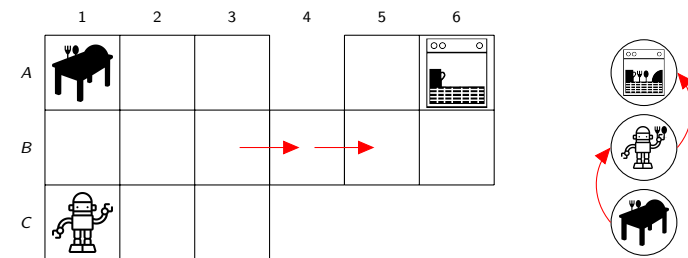
Some heuristics exploit **constraints** that describe something that holds in every solution of the task.

For instance, every solution is such that

- ▶ a variable takes some value in at least one visited state. (a **fact landmark** constraint)
- ▶ an action must be applied. (an **action landmark** constraint)

## Action Landmarks: Example

Which actions must be applied in every solution?



- ▶ pickup
- ▶ load
- ▶ move-B3-B4
- ▶ move-B4-B5

## Constraints

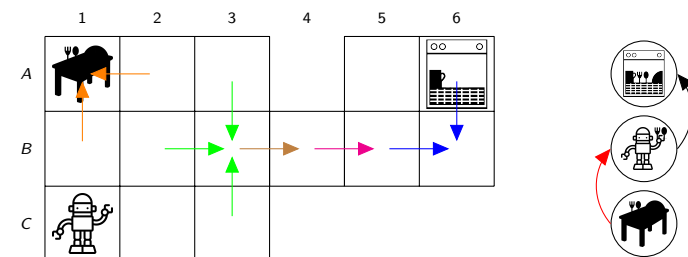
Some heuristics exploit **constraints** that describe something that holds in every solution of the task.

For instance, every solution is such that

- ▶ a variable takes some **value** in at least one visited state. (a **fact landmark** constraint)
- ▶ an action must be applied. (an **action landmark** constraint)
- ▶ at least one action from a set of actions must be applied. (a **disjunctive action landmark** constraint)

## Disjunctive Action Landmarks: Example

Which set of actions is such that at least one must be applied?



- ▶ {pickup}
- ▶ {load}
- ▶ {move-B3-B4}
- ▶ {move-B4-B5}
- ▶ {move-A6-B6, move-B5-B6}
- ▶ {move-A3-B3, move-B2-B3, move-C3-B3}
- ▶ {move-B1-A1, move-A2-A1}
- ▶ ...

## Constraints

Some heuristics exploit **constraints** that describe something that holds in every solution of the task.

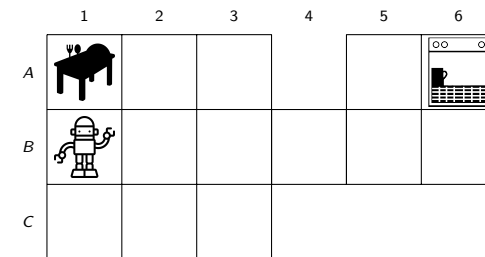
For instance, every solution is such that

- ▶ a variable takes some value in at least one visited state. (a **fact landmark** constraint)
- ▶ at least one action from a set of actions must be applied. (a **disjunctive action landmark** constraint)
- ▶ fact consumption and production is “balanced”. (a **network flow** constraint)

## Network Flow: Example

Consider the fact  $\text{robot-at} = B1$ .

How often are actions used that enter this cell?



**Answer:** as often as actions that leave this cell

If  $\text{Count}_o$  denotes how often operator  $o$  is applied, we have:

$$\begin{aligned} &\text{Count}_{\text{move-A1-B1}} + \text{Count}_{\text{move-B2-B1}} + \text{Count}_{\text{move-C1-B1}} = \\ &\text{Count}_{\text{move-B1-A1}} + \text{Count}_{\text{move-B1-B2}} + \text{Count}_{\text{move-B1-C1}} \end{aligned}$$

## F1.2 Multiple Heuristics

## Combining Admissible Heuristics Admissibly

Major ideas to combine heuristics admissibly:

- ▶ maximize
- ▶ canonical heuristic (for abstractions)
- ▶ **minimum hitting set** (for landmarks)
- ▶ **cost partitioning**
- ▶ **operator counting**

Often computed as solution to a **(integer) linear program**.

## 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  $dom(v_2) = dom(v_3) = \{A, B, C\}$ ,  $I = \{v_1 \mapsto A, v_2 \mapsto A, v_3 \mapsto A\}$ ,

$$o_1 = \langle v_1 = A, v_1 := B, 1 \rangle$$

$$o_2 = \langle v_2 = A \wedge v_3 = A, v_2 := B \wedge v_3 := B, 1 \rangle$$

$$o_3 = \langle v_2 = B, v_2 := C, 1 \rangle$$

$$o_4 = \langle v_3 = B, v_3 := C, 1 \rangle$$

and  $\gamma = (v_1 = B) \wedge (v_2 = C) \wedge (v_3 = C)$ .

Let  $\mathcal{C}$  be the pattern collection that contains all atomic projections. What is the canonical heuristic function  $h^{\mathcal{C}}$ ?

**Answer:** Let  $h_i := h^{v_i}$ . Then  $h^{\mathcal{C}} = \max\{h_1 + h_2, h_1 + h_3\}$ .

## Reminder: Orthogonality and Additivity

Why can we add  $h_1$  and  $h_2$  ( $h_1$  and  $h_3$ ) admissibly?

### Theorem (Additivity for Orthogonal Abstractions)

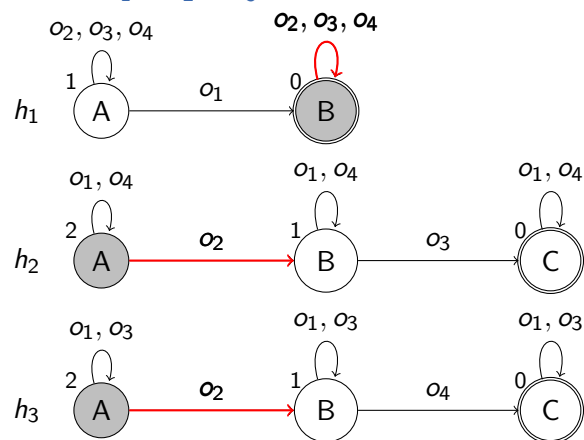
Let  $h^{\alpha_1}, \dots, h^{\alpha_n}$  be abstraction heuristics of the same transition system such that  $\alpha_i$  and  $\alpha_j$  are orthogonal for all  $i \neq j$ .

Then  $\sum_{i=1}^n h^{\alpha_i}$  is a safe, goal-aware, admissible and consistent heuristic for  $\Pi$ .

The proof exploits that every concrete transition induces state-changing transition in at most one abstraction.

## 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  $o_2$ .

## Prevent Inadmissibility

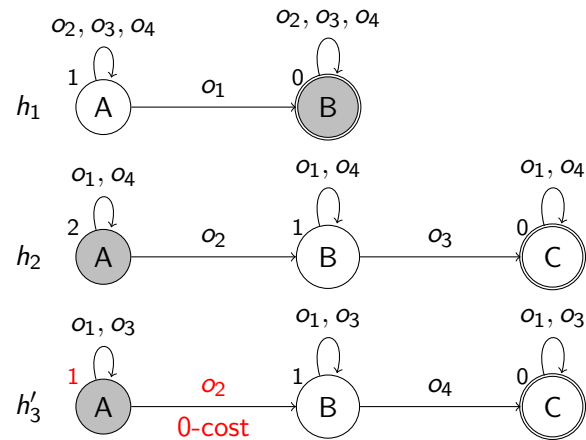
The reason that  $h_2$  and  $h_3$  are not additive is because the cost of  $o_2$  is considered in both.

Is there anything we can do about this?

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

## Combining Heuristics Admissibly: Example

Let  $h' = h_1 + h_2 + h_3$ , where  $h_3 = h^{v_3}$  assuming  $cost_3(o_2) = 0$ .



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

## Cost partitioning

Using the cost of every operator only in one heuristic is called a **zero-one cost partitioning**.

More generally, heuristics are additive if all operator costs are distributed in a way that the sum of the individual costs is no larger than the cost of the operator.

This can also be expressed as a constraint, the **cost partitioning constraint**:

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

(more details later)

## F1.3 Summary

## Summary

- ▶ Landmarks and network flows are **constraints** that describe something that holds in every solution of the task.
- ▶ Heuristics can be combined admissibly if the **cost partitioning constraint** is satisfied.