

# Planning and Optimization

## F11. Operator Counting

Malte Helmert and Gabriele Röger

Universität Basel

December 15, 2025

# Planning and Optimization

December 15, 2025 — F11. Operator Counting

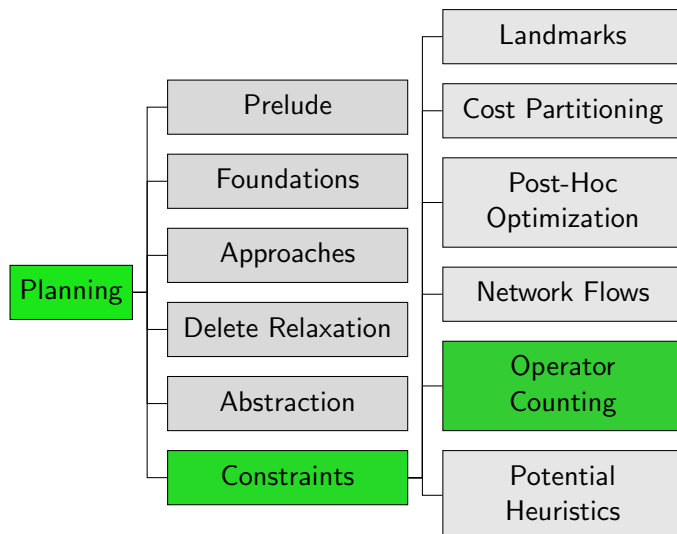
## F11.1 Introduction

## F11.2 Operator-counting Framework

## F11.3 Properties

## F11.4 Summary

## Content of the Course



## F11.1 Introduction

## Reminder: Flow Heuristic

In the previous chapter, we used **flow constraints** to describe how often operators must be used in each plan.

### Example (Flow Constraints)

Let  $\Pi$  be a planning problem with operators  $\{O_{\text{red}}, O_{\text{green}}, O_{\text{blue}}\}$ . The flow constraint for some atom  $a$  is the constraint

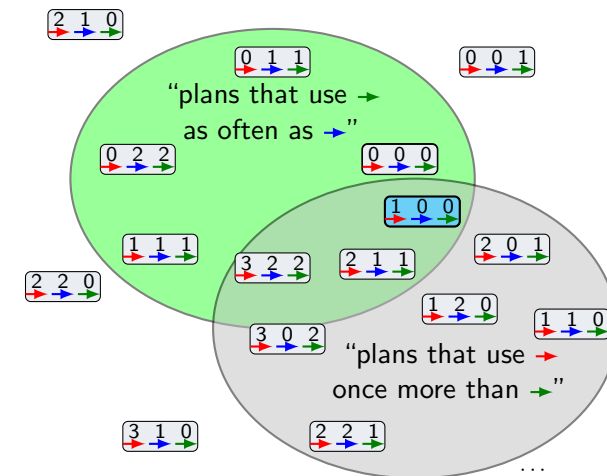
$$1 + \text{Count}_{O_{\text{green}}} = \text{Count}_{O_{\text{red}}} \text{ if}$$

- ▶  $a$  is true in the initial state      ▶  $O_{\text{green}}$  produces  $a$
- ▶  $a$  is false in the goal              ▶  $O_{\text{red}}$  consumes  $a$

In natural language, the flow constraint expresses that every plan uses  $O_{\text{red}}$  once more than  $O_{\text{green}}$ .

## Reminder: Flow Heuristic

Let us now observe how each flow constraint alters the **operator count solution space**.



## F11.2 Operator-counting Framework

## Operator Counting

### Operator counting

- ▶ generalizes this idea to a framework that allows to **admissibly combine different heuristics**.
- ▶ uses **linear constraints** ...
- ▶ ... that describe **number of occurrences** of an operator ...
- ▶ ... and must be satisfied by **every plan**.
- ▶ provides declarative way to describe **knowledge about solutions**.
- ▶ allows **reasoning about solutions** to derive heuristic estimates.

## Operator-counting Constraint

### Definition (Operator-counting Constraints)

Let  $\Pi$  be a planning task with operators  $O$  and let  $s$  be a state.  
Let  $\mathcal{V}$  be the set of integer variables  $\text{Count}_o$  for each  $o \in O$ .

A linear inequality over  $\mathcal{V}$  is called an **operator-counting constraint** for  $s$  if for every plan  $\pi$  for  $s$  setting each  $\text{Count}_o$  to the number of occurrences of  $o$  in  $\pi$  is a feasible variable assignment.

## Operator-counting Heuristics

### Definition (Operator-counting IP/LP Heuristic)

The operator-counting integer program  $\text{IP}_C$  for a set  $C$  of operator-counting constraints for state  $s$  is

$$\begin{aligned} &\text{Minimize} && \sum_{o \in O} \text{cost}(o) \cdot \text{Count}_o && \text{subject to} \\ &&& C \text{ and } \text{Count}_o \geq 0 \text{ for all } o \in O, \end{aligned}$$

where  $O$  is the set of operators.

The **IP heuristic**  $h_C^{\text{IP}}$  is the objective value of  $\text{IP}_C$ ,  
the **LP heuristic**  $h_C^{\text{LP}}$  is the objective value of its LP-relaxation.

If the IP/LP is infeasible, the heuristic estimate is  $\infty$ .

## Operator-counting Constraints

- ▶ Adding more constraints can only remove feasible solutions.
- ▶ Fewer feasible solutions can only increase the objective value.
- ▶ Higher objective value means better informed heuristic

⇒ Have we already seen other operator-counting constraints?

## Reminder: Minimum Hitting Set for Landmarks

### Variables

Non-negative variable  $\text{Applied}_o$  for each operator  $o$

### Objective

Minimize  $\sum_o \text{cost}(o) \cdot \text{Applied}_o$

### Subject to

$$\sum_{o \in L} \text{Applied}_o \geq 1 \text{ for all landmarks } L$$

## Operator Counting with Disjunctive Action Landmarks

### Variables

Non-negative variable  $\text{Count}_o$  for each operator  $o$

### Objective

Minimize  $\sum_o \text{cost}(o) \cdot \text{Count}_o$

### Subject to

$$\sum_{o \in L} \text{Count}_o \geq 1 \text{ for all landmarks } L$$

## Reminder: Post-hoc Optimization Heuristic

For set of abstractions  $\{\alpha_1, \dots, \alpha_n\}$ :

### Variables

Non-negative variables  $X_o$  for all operators  $o \in O$

$X_o$  is cost incurred by operator  $o$

### Objective

Minimize  $\sum_{o \in O} X_o$

### Subject to

$$\begin{aligned} \sum_{o \in O: o \text{ relev. for } \alpha} X_o &\geq h^\alpha(s) && \text{for } \alpha \in \{\alpha_1, \dots, \alpha_n\} \\ X_o &\geq 0 && \text{for all } o \in O \end{aligned}$$

## Operator Counting with Post-hoc Optimization Constraints

For set of abstractions  $\{\alpha_1, \dots, \alpha_n\}$ :

### Variables

Non-negative variables  $\text{Count}_o$  for all operators  $o \in O$

$\text{Count}_o \cdot \text{cost}(o)$  is cost incurred by operator  $o$

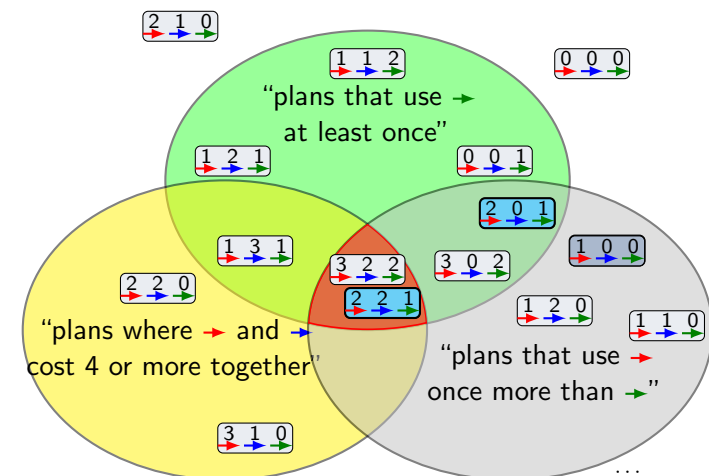
### Objective

Minimize  $\sum_{o \in O} \text{cost}(o) \cdot \text{Count}_o$

### Subject to

$$\begin{aligned} \sum_{o \in O: o \text{ relev. for } \alpha} \text{cost}(o) \cdot \text{Count}_o &\geq h^\alpha(s) && \text{for } \alpha \in \{\alpha_1, \dots, \alpha_n\} \\ \text{cost}(o) \cdot \text{Count}_o &\geq 0 && \text{for all } o \in O \end{aligned}$$

## Example



## Further Examples?

- ▶ The definition of operator-counting constraints can be extended to groups of constraints and auxiliary variables.
- ▶ With this extended definition we could also cover more heuristics, e.g., the perfect relaxation heuristic  $h^+$

## F11.3 Properties

## Admissibility

### Theorem (Operator-counting Heuristics are Admissible)

The IP and the LP heuristic are *admissible*.

#### Proof.

Let  $C$  be a set of operator-counting constraints for state  $s$  and  $\pi$  be an optimal plan for  $s$ . The number of operator occurrences of  $\pi$  are a feasible solution for  $C$ . As the IP/LP minimizes the total plan cost, the objective value cannot exceed the cost of  $\pi$  and is therefore an admissible estimate.  $\square$

## Dominance

### Theorem

Let  $C$  and  $C'$  be sets of operator-counting constraints for  $s$  and let  $C \subseteq C'$ . Then  $IP_C \leq IP_{C'}$  and  $LP_C \leq LP_{C'}$ .

#### Proof.

Every feasible solution of  $C'$  is also feasible for  $C$ . As the LP/IP is a minimization problem, the objective value subject to  $C$  can therefore not be larger than the one subject to  $C'$ .  $\square$

Adding more constraints can only improve the heuristic estimate.

## Heuristic Combination

Operator counting as **heuristic combination**

- ▶ Multiple operator-counting heuristics can be combined by computing  $h_C^{LP}/h_C^{IP}$  for the **union of their constraints**.
  - ▶ This is an **admissible** combination.
    - ▶ Never worse than maximum of individual heuristics
    - ▶ Sometimes even better than their sum
  - ▶ We already know a way of admissibly combining heuristics: cost partitioning.
- ⇒ **How are they related?**

## Connection to Cost Partitioning

### Theorem

Let  $C_1, \dots, C_n$  be sets of operator-counting constraints for  $s$  and  $C = \bigcup_{i=1}^n C_i$ . Then  $h_C^{LP}$  is the **optimal general cost partitioning** over the heuristics  $h_{C_i}^{LP}$ .

### Proof Sketch.

In  $LP_C$ , add variables  $Count_o^i$  and constraints  $Count_o = Count_o^i$  for all operators  $o$  and  $1 \leq i \leq n$ . Then replace  $Count_o$  by  $Count_o^i$  in  $C_i$ .

Dualizing the resulting LP shows that  $h_C^{LP}$  computes a cost partitioning. Dualizing the component heuristics of that cost partitioning shows that they are  $h_{C_i}^{LP}$ .

## Comparison to Optimal Cost Partitioning

- ▶ some heuristics are **more compact** if expressed as operator counting
- ▶ some heuristics **cannot be expressed** as operator counting
- ▶ **operator counting IP** even better than optimal cost partitioning
- ▶ Cost partitioning maximizes, so heuristics must be encoded perfectly to guarantee admissibility. Operator counting minimizes, so missing information just makes the heuristic weaker.

## F11.4 Summary

## Summary

- ▶ Many heuristics can be formulated in terms of **operator-counting constraints**.
- ▶ The operator counting heuristic framework allows to **combine the constraints** and to reason on the entire encoded declarative knowledge.
- ▶ The heuristic estimate for the combined constraints **can be better than the one of the best ingredient heuristic** but never worse.
- ▶ Operator counting is **equivalent to optimal general cost partitioning** over individual constraints.