Automated Planning

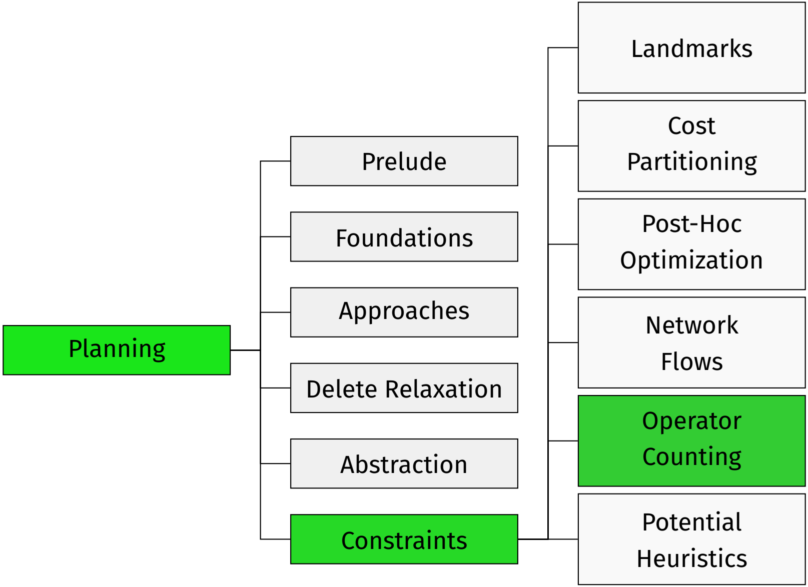
F10. Operator Counting

Jendrik Seipp

Linköping University

based on slides from the AI group at the University of Basel

Content of this Course



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 Π be a planning problem with operators $\{o_{\mathrm{red}}, o_{\mathrm{green}}, o_{\mathrm{blue}}\}$. The flow constraint for some atom a is the constraint

$$1 + \mathit{Count}_{o_{\mathrm{green}}} = \mathit{Count}_{o_{\mathrm{red}}} \text{ if}$$

- a is true in the initial state
- a is false in the goal
- o_{green} produces a
- o_{red} consumes a

In natural language, the flow constraint expresses that

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 Π 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

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

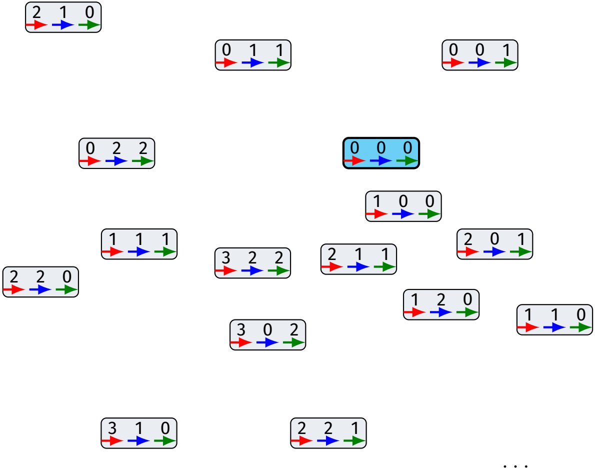
- a is true in the initial state
- a is false in the goal
- o_{green} produces a
- o_{red} consumes a

In natural language, the flow constraint expresses that

every plan uses o_{red} once more than o_{green}.

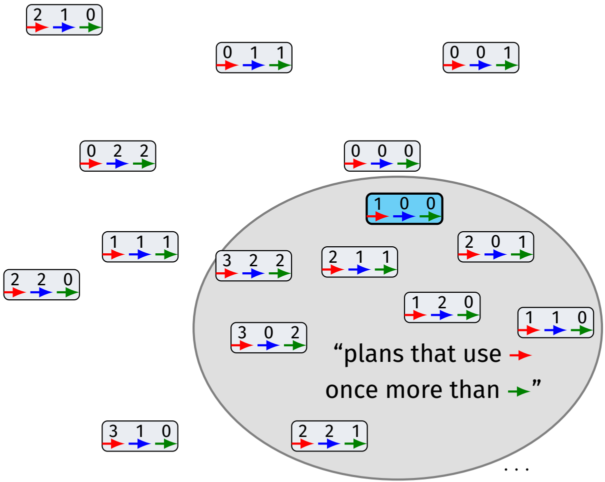
Reminder: Flow Heuristic

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



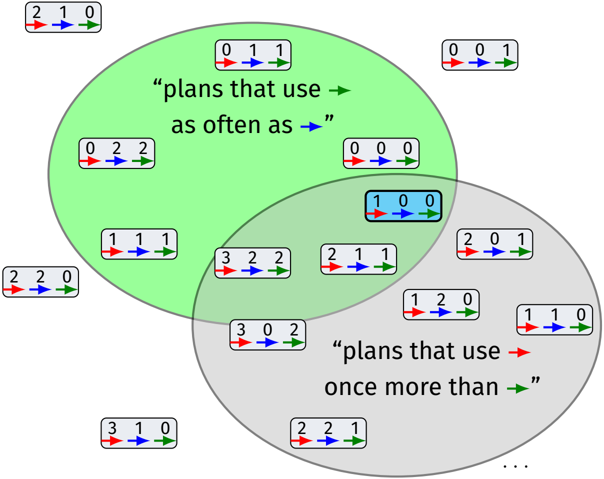
Reminder: Flow Heuristic

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



Reminder: Flow Heuristic

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



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 Π be a planning task with operators O and let s be a state. Let $\mathcal{V}$ be the set of integer variables 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 π for s setting each Count_o to the number of occurrences of o in π is a feasible variable assignment.

Operator-counting Heuristics

Definition (Operator-counting IP/LP Heuristic)

The operator-counting integer program IP_C for a set C of operator-counting constraints for state s is

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

$$C \text{ and } \mathrm{Count}_o \geq 0 \text{ for all } o \in O,$$

where O is the set of operators.

The IP heuristic h_C^{IP} is the objective value of IP_C,
the LP heuristic h_C^{LP} is the objective value of its LP-relaxation.

If the IP/LP is infeasible, the heuristic estimate is ∞.

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

$\Longrightarrow$ Have we already seen other operator-counting constraints?

Reminder: Minimum Hitting Set for Landmarks

Variables

Non-negative variable Applied_o for each operator o

Objective

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

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 Count_o for each operator o

Objective

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

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

$$\sum_{o \in O : o \text{ relev. for } \alpha} X_o \geq h^\alpha(s) \quad \text{for } \alpha \in \{\alpha_1, \dots, \alpha_n\}$$

$$X_o \geq 0 \quad \text{for all } o \in O$$

Operator Counting with Post-hoc Optimization Constraints

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

Variables

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

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

Objective

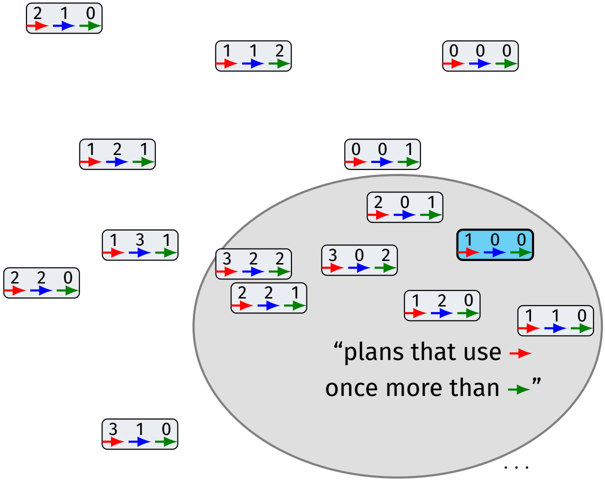
Minimize $\sum_{o \in O} cost(o) \cdot \text{Count}_o$

Subject to

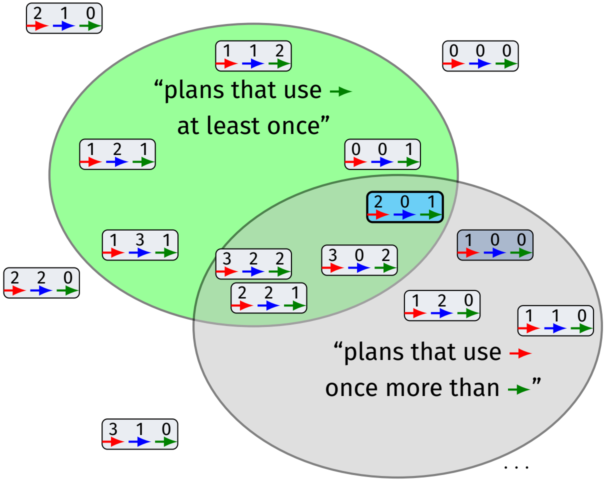
$$\sum_{o \in O: o \text{ relev. for } \alpha} cost(o) \cdot \text{Count}_o \geq h^{\alpha}(s) \quad \text{for } \alpha \in \{\alpha_1, \dots, \alpha_n\}$$

$$cost(o) \cdot \text{Count}_o \geq 0 \quad \text{for all } o \in O$$

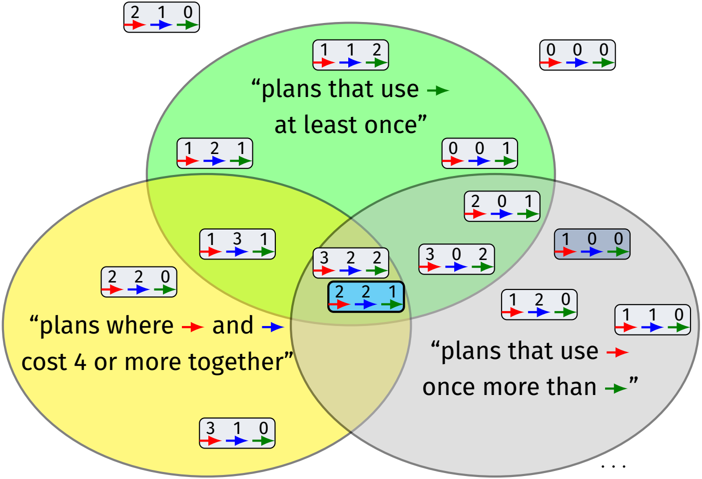
Example


2 1 0
1 1 2
0 0 0
1 2 1
0 0 1
2 0 1
1 3 1
1 0 0
3 2 2
3 0 2
2 2 0
2 2 1
1 2 0
1 1 0
"plans that use
once more than "
3 1 0
. . .

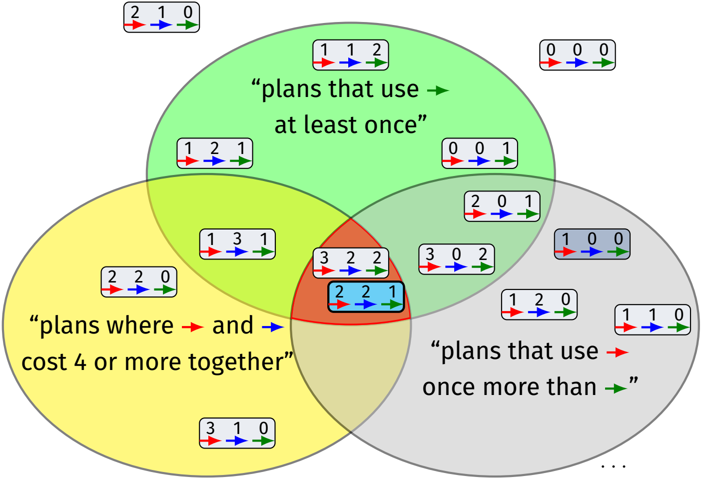
Example


2 1 0
1 1 2
0 0 0
"plans that use → at least once"
1 2 1
0 0 1
2 0 1
1 3 1
1 0 0
3 2 2
3 0 2
2 2 0
2 2 1
1 2 0
1 1 0
"plans that use → once more than →"
3 1 0
...

Example


2 1 0
1 1 2
0 0 0
“plans that use → at least once”
1 2 1
0 0 1
2 0 1
1 3 1
1 0 0
3 2 2
3 0 2
2 2 0
2 2 1
1 2 0
1 1 0
“plans where → and → cost 4 or more together”
“plans that use → once more than →”
3 1 0
. . .

Example


2 1 0
1 1 2
0 0 0
“plans that use at least once”
1 2 1
0 0 1
2 0 1
1 0 0
1 3 1
3 2 2
3 0 2
2 2 0
2 2 1
1 2 0
1 1 0
“plans where and cost 4 or more together”
“plans that use once more than ”
3 1 0
. . .

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^+.

Properties

Properties

Theorem (Operator-counting Heuristics are Admissible)

The IP and the LP heuristic are admissible.

Theorem (Dominance)

Let C and C' be sets of operator-counting constraints for s and let $C \subseteq C'$. Then $\mathrm{IP}_C \leq \mathrm{IP}_{C'}$ and $\mathrm{LP}_C \leq \mathrm{LP}_{C'}$.

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 $h^{\mathrm{LP}}_{\mathcal{C}}/h^{\mathrm{IP}}_{\mathcal{C}}$ 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.

  $\Longrightarrow$ How are they related?

Connection to Cost Partitioning

Theorem

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

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.

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.