LP-based Heuristics for Cost-optimal Classical Planning

Florian Pommerening Gabriele Röger Malte Helmert

Based on: ICAPS 2015 Tutorial

March 2017

Linear Programs

Linear Programs and Integer Programs

Linear Program

A linear program (LP) consists of:

- a finite set of real-valued variables V
- a finite set of linear inequalities (constraints) over V
- an objective function, which is a linear combination of V
- which should be minimized or maximized.

Integer program (IP): ditto, but with integer-valued variables

Linear Program: Example

Example:

$$\text{maximize} \quad 2x - 3y + z \quad \text{subject to}$$

$$\begin{aligned} x + 2y + z &\leq 10 \\ x \quad\quad - z &\leq 0 \end{aligned}$$

$$x \geq 0, \quad y \geq 0, \quad z \geq 0$$

⇝ unique optimal solution:
$x = 5$, $y = 0$, $z = 5$ (objective value 15)

Solving Linear Programs and Integer Programs

Complexity:

- LP solving is a polynomial-time problem.
- Finding solutions for IPs is NP-complete.

Common idea:

- Approximate IP solution with corresponding LP (LP relaxation).

Three Key Ideas

Cost Partitioning

Idea 1: Cost Partitioning

- create copies $\Pi_1, \ldots, \Pi_n$ of planning task Π
- each has its own operator cost function $cost_i$ (otherwise identical to Π)
- for all o: require $cost_1(o) + \cdots + cost_n(o) \leq cost(o)$

⇝ sum of solution costs in copies is admissible heuristic: $h^*_{\Pi_1} + \cdots + h^*_{\Pi_n} \leq h^*_{\Pi}$

Motivation:

- method for obtaining additive admissible heuristics
- very general and powerful

Operator Counting Constraints

Idea 2: Operator Counting Constraints

- linear constraints whose variables denote number of occurrences of a given operator
- must be satisfied by every plan that solves the task

Examples:

- $Y_{o_1} + Y_{o_2} \geq 1$ "must use o_1 or o_2 at least once"
- $Y_{o_1} - Y_{o_3} \leq 0$ "cannot use o_1 more often than o_3"

Motivation:

- declarative way to represent knowledge about solutions
- allows reasoning about solutions to derive heuristic estimates

Potential Heuristics

Idea 3: Potential Heuristics

Heuristic design as an optimization problem:

- Define simple numerical state features $f_1, \ldots, f_n$.
- Consider heuristics that are linear combinations of features:

$$h(s) = w_1 f_1(s) + \cdots + w_n f_n(s)$$

  with weights (potentials) $w_i \in \mathbb{R}$
- Find potentials for which h is admissible and well-informed.

Motivation:

- declarative approach to heuristic design
- heuristic very fast to compute if features are fast to compute

Connections

Three unrelated ideas?

- No! It turns out they are closely connected.

Tutorial Structure

1. ~~Introduction and Overview~~
2. Cost Partitioning
3. Operator Counting
4. Potential Heuristics

Cost Partitioning

Cost Partitioning

Idea 1: Cost Partitioning

- create copies $\Pi_1, \ldots, \Pi_n$ of planning task Π
- each has its own operator cost function $cost_i : \mathcal{O} \to \mathbb{R}_0^+$ (otherwise identical to Π)
- for all o: require $cost_1(o) + \cdots + cost_n(o) \leq cost(o)$

⇝ sum of solution costs in copies is admissible heuristic: $h^*_{\Pi_1} + \cdots + h^*_{\Pi_n} \leq h^*_\Pi$

Cost Partitioning

- for admissible heuristics $h_1, \ldots, h_n$,
  $h(s) = h_{1,\Pi_1}(s) + \cdots + h_{n,\Pi_n}(s)$
  is an admissible estimate
- $h(s)$ can be better or worse than any $h_{i,\Pi}(s)$
  $\rightarrow$ depending on cost partitioning
- strategies for defining cost-functions
  - uniform: $cost_i(o) = cost(o)/n$
  - zero-one: full operator cost in one copy, zero in all others
  - ...

Can we find an optimal cost partitioning?

Optimal Cost Partitioning

Optimal Cost Partitioning

Optimal Cost Partitioning with LPs

- Use variables for cost of each operator in each task copy
- Express heuristic values with linear constraints
- Maximize sum of heuristic values subject to these constraints

LPs known for

- abstraction heuristics
- landmark heuristic

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.

Tutorial Structure

1. ~~Introduction and Overview~~
2. ~~Cost Partitioning~~
3. Operator Counting
4. Potential Heuristics

Operator-counting

Operator Counting

Reminder:

Idea 2: Operator Counting Constraints

- linear constraints whose variables denote number of occurrences of a given operator
- must be satisfied by every plan that solves the task

Examples:

- $Y_{o_1} + Y_{o_2} \geq 1$ "must use o_1 or o_2 at least once"
- $Y_{o_1} - Y_{o_3} \leq 0$ "cannot use o_1 more often than o_3"

Motivation:

- declarative way to represent knowledge about solutions
- allows reasoning about solutions to derive heuristic estimates

Operator Counting Heuristics

Operator occurrences in potential plans

(2,1,0)

(1,1,2) (0,0,0)

(0,0,1)

(1,2,1)

(3,0,2)

(1,3,1)

(3,2,2)

(2,2,0)

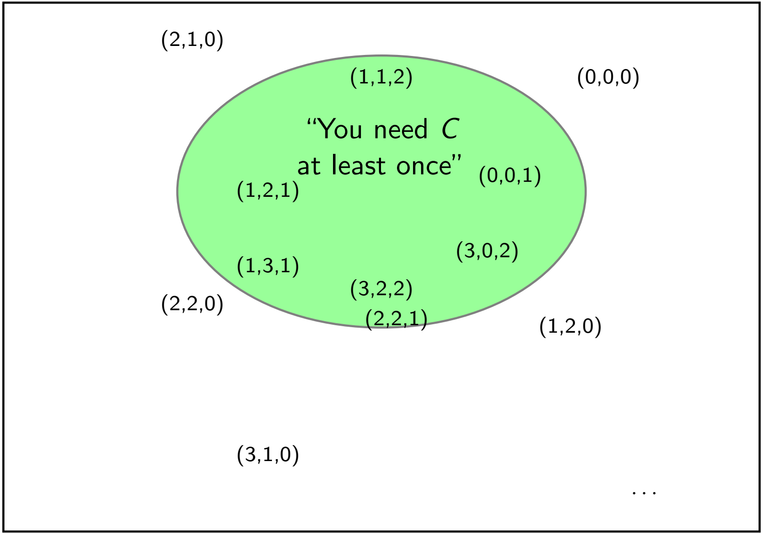
(2,2,1) (1,2,0)

(3,1,0)

. . .

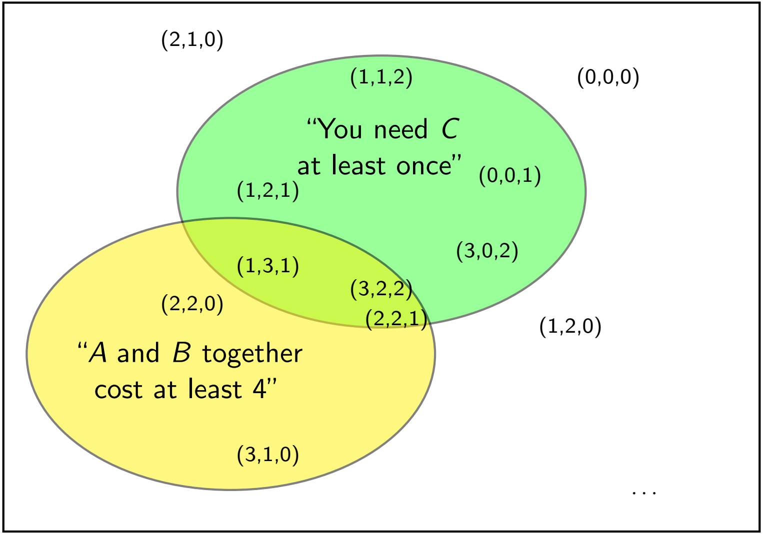
Operator Counting Heuristics

Operator occurrences in potential plans



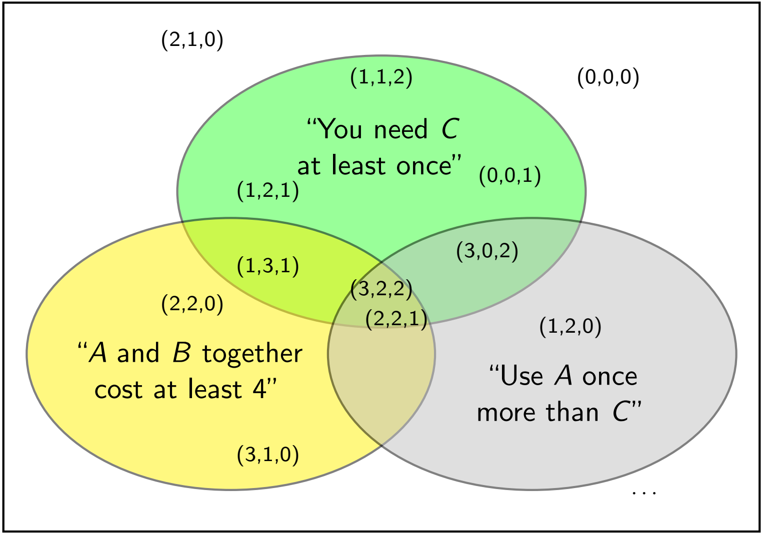
Operator Counting Heuristics

Operator occurrences in potential plans



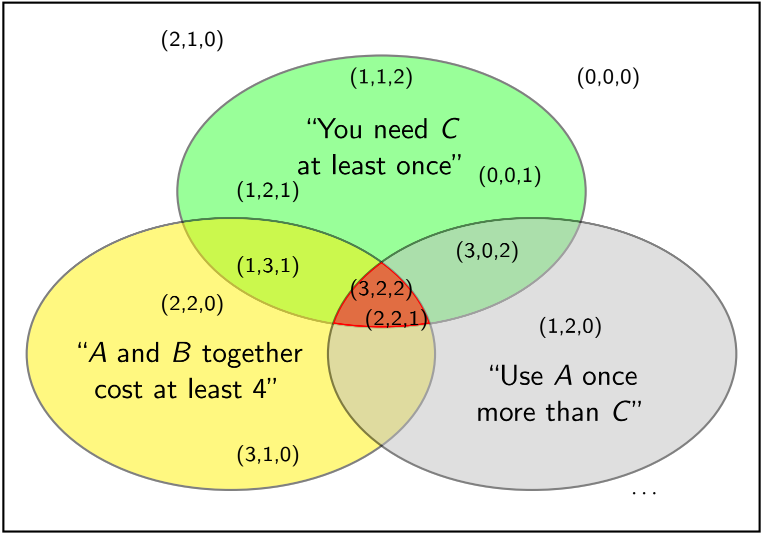
Operator Counting Heuristics

Operator occurrences in potential plans



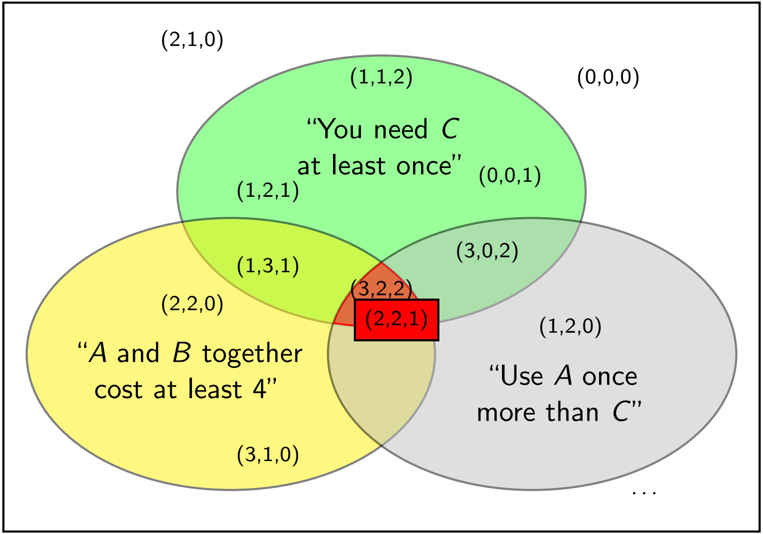
Operator Counting Heuristics

Operator occurrences in potential plans



Operator Counting Heuristics

Operator occurrences in potential plans



Operator-counting Heuristics

Operator-counting IP/LP Heuristic

$$\text{Minimize } \sum_o Y_o \cdot cost(o) \text{ subject to}$$

$Y_o \geq 0$ and some operator-counting constraints

Operator-counting constraint

- Set of linear inequalities
- For every plan π there is an LP-solution where Y_o is the number of occurrences of o in π .

Properties of Operator-counting Heuristics

Admissibility

Operator-counting (IP and LP) heuristics are admissible.

Computation time

Operator-counting LP heuristics are solvable in polynomial time.

Adding constraints

Adding constraints can only make the heuristic more informed.

State-equation Heuristic

State-equation Heuristic (SEQ)

Main idea:

- Facts can be produced (made true) or consumed (made false) by an operator
- Number of producing and consuming operators must balance out for each fact

State-equation Heuristic

Net-change constraint for fact f

$$G(f) - S(f) = \sum_{f \in eff(o)} Y_o \quad - \sum_{f \in pre(o)} Y_o$$

Net-change constraint for fact f

$$G(f) - S(f) = \sum_{o \text{ produces } f} Y_o \quad - \sum_{o \text{ consumes } f} Y_o$$

Remark:

- Assumes transition normal form (not a limitation)
  - Operator mentions same variables in precondition and effect
  - General form of constraints more complicated

Tutorial Structure

1. ~~Introduction and Overview~~
2. ~~Cost Partitioning~~
3. ~~Operator Counting~~
4. Potential Heuristics

Overview

Potential Heuristics

Reminder:

Idea 3: Potential Heuristics

Heuristic design as an optimization problem:

- Define simple numerical state features $f_1, \dots, f_n$.
- Consider heuristics that are linear combinations of features:

$$h(s) = w_1 f_1(s) + \dots + w_n f_n(s)$$

  with weights (potentials) $w_i \in \mathbb{R}$
- Find potentials for which h is admissible and well-informed.

Motivation:

- declarative approach to heuristic design
- heuristic very fast to compute if features are

Comparison to Previous Parts (1)

What is the same as in operator-counting constraints:

- We again use LPs to compute (admissible) heuristic values (spoiler alert!)

Comparison to Previous Parts (2)

What is different from operator-counting constraints (computationally):

- With potential heuristics, solving one LP defines the entire heuristic function, not just the estimate for a single state.
- Hence we only need one LP solver call, making LP solving much less time-critical.

Comparison to Previous Parts (3)

What is different from operator-counting constraints (conceptually):

- axiomatic approach for defining heuristics:
  - What should a heuristic look like mathematically?
  - Which properties should it have?
- define a space of interesting heuristics
- use optimization to pick a good representative

Potential Heuristics

Features

Definition (feature)

A (state) feature for a planning task is a numerical function defined on the states of the task: $f : S \to \mathbb{R}$.

Potential Heuristics

Definition (potential heuristic)

A potential heuristic for a set of features $\mathcal{F} = \{f_1, \dots, f_n\}$ is a heuristic function h defined as a linear combination of the features:

$$h(s) = w_1 f_1(s) + \cdots + w_n f_n(s)$$

with weights (potentials) $w_i \in \mathbb{R}$.

$\rightsquigarrow$ cf. evaluation functions for board games like chess

Atomic Potential Heuristics

Atomic features test if some proposition is true in a state:

Definition (atomic feature)

Let $X = x$ be an atomic proposition of a planning task.

The atomic feature $f_{X=x}$ is defined as:

$$f_{X=x}(s) = \begin{cases} 1 & \text{if variable } X \text{ has value } x \text{ in state } s \\ 0 & \text{otherwise} \end{cases}$$

- We only consider atomic potential heuristics, which are based on the set of all atomic features.
- Example for a task with state variables X and Y:

$$h(s) = 3f_{X=a} + \frac{1}{2}f_{X=b} - 2f_{X=c} + \frac{5}{2}f_{Y=d}$$

Finding Good Potential Heuristics

How to Set the Weights?

We want to find good atomic potential heuristics:

- admissible
- consistent
- well-informed

How to achieve this? Linear programming to the rescue!

Admissible and Consistent Potential Heuristics

Constraints on potentials characterize (= are necessary and sufficient for) admissible and consistent atomic potential heuristics:

Goal-awareness (i.e., $h(s) = 0$ for goal states)

$$\sum_{\text{goal facts } f} w_f = 0$$

Consistency

$$\sum_{\substack{f \text{ consumed} \\ \text{by } o}} w_f - \sum_{\substack{f \text{ produced} \\ \text{by } o}} w_f \leq cost(o) \quad \text{for all operators } o$$

Remarks:

- assumes transition normal form (not a limitation)
- goal-aware and consistent = admissible and consistent

Well-Informed Potential Heuristics

How to find a well-informed potential heuristic?

$\rightsquigarrow$ encode quality metric in the objective function and use LP solver to find a heuristic maximizing it

Examples:

- maximize heuristic value of a given state (e.g., initial state)
- maximize average heuristic value of all states (including unreachable ones)
- maximize average heuristic value of some sample states
- minimize estimated search effort

Connections

Connections

So what does this have to do with what we talked about before?

Connections

So what does this have to do with what we talked about before?

Theorem (Pommerening et al., AAAI 2015)

For state s, let $h^{\text{maxpot}}(s)$ denote the maximal heuristic value of all admissible and consistent atomic potential heuristics in s.

Then $h^{\text{maxpot}}(s) = h^{\text{SEQ}}(s) = h^{\text{gOCP}}(s)$.

- h^{SEQ}: state equation heuristic a.k.a. flow heuristic
- h^{gOCP}: optimal general cost partitioning of atomic projections

proof idea: compare dual of $h^{\text{SEQ}}(s)$ LP
to potential heuristic constraints optimized for state s

What Do We Take From This?

- general cost partitioning, operator-counting constraints and potential heuristics: facets of the same phenomenon
- study of each reinforces understanding of the others
- potential heuristics: fast admissible approximations of h^{SEQ}
- clear path towards generalization beyond h^{SEQ}:
  use non-atomic features

The End

1. ~~Introduction and Overview~~
2. ~~Cost Partitioning~~
3. ~~Operator Counting~~
4. ~~Potential Heuristics~~

Thank you for your attention!