LP-based Heuristics for Cost-optimal Classical Planning

Florian Pommerening Gabriele Röger Malte Helmert

Based on: ICAPS 2015 Tutorial

March 2017

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

Tutorial Structure

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

Optimal 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: $h^*_{\Pi_1} + \cdots + h^*_{\Pi_n} \leq h^*_{\Pi}$

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

Optimal Cost Partitioning for Abstractions

Abstractions

- Simplified versions of the planning task, e.g. projections
- Cost of optimal abstract plan is admissible estimate

How to express the heuristic value as linear constraints?

Optimal Cost Partitioning for Abstractions

Abstractions

- Simplified versions of the planning task, e.g. projections
- Cost of optimal abstract plan is admissible estimate

How to express the heuristic value as linear constraints?
$\rightsquigarrow$ Shortest path problem in abstract transition system

LP for Shortest Path in State Space

Variables

Distance_s for each state s,
GoalDist

Objective

Maximize GoalDist

Subject to

$$\text{Distance}_{s_I} = 0 \quad \text{for the initial state } s_I$$

$$\text{Distance}_{s'} \leq \text{Distance}_s + cost(o) \quad \text{for all transition } s \xrightarrow{o} s'$$

$$\text{GoalDist} \leq \text{Distance}_{s_\star} \quad \text{for all goal states } s_\star$$

Optimal Cost Partitioning for Abstractions I

Variables

For each abstraction α:
Distance_s^α for each abstract state s,
$\text{cost}^\alpha(o)$ for each operator o,
GoalDist^α

Objective

Maximize $\sum_\alpha \text{GoalDist}^\alpha$

...

Optimal Cost Partitioning for Abstractions II

Subject to

for all operators o

$$\sum_{\alpha} \text{Cost}_o^{\alpha} \leq cost(o)$$

$$\text{Cost}_o^{\alpha} \geq 0 \qquad \text{for all abstractions } \alpha$$

and for all abstractions α

$$\text{Distance}_{s_I}^{\alpha} = 0 \qquad \text{for the abstract initial state } s_I$$

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

$$\text{GoalDist}^{\alpha} \leq \text{Distance}_{s_\star}^{\alpha} \qquad \text{for all abstract goal states } s_\star$$

Optimal Cost Partitioning for Landmarks

Disjunctive action landmark

- Set of operators
- Every plan uses at least one of them
- Landmark cost = cost of cheapest operator

Optimal Cost Partitioning for Landmarks

Variables

Cost_L for each landmark L

Objective

Maximize $\sum_L \mathsf{Cost}_L$

Subject to

$$\sum_{L:o\in L} \mathsf{Cost}_L \leq cost(o) \quad \text{for all operators } o$$

Optimal Cost Partitioning for Landmarks (Dual)

Variables

Occurrences_o for each operator o

Objective

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

Subject to

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

$$\text{Occurrences}_o \geq 0 \text{ for all operators } o$$

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-counting IP/LP Heuristic

Minimize $\sum_{o} Y_o \cdot cost(o)$ 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 .

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

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

Remark:

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

State-equation Heuristic (Constraints)

Net-change constraint for fact f

$$0 = \sum_{o \text{ produces } f} Y_o \quad - \sum_{o \text{ consumes } f} Y_o$$

State-equation Heuristic (Constraints)

Net-change constraint for fact f

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

- Special cases for goal and initial state
  - Add/Subtract one from net change

State-equation Heuristic (Constraints)

Net-change constraint for fact f

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

- Special cases for goal and initial state
  - Add/Subtract one from net change

State-equation Heuristic (Constraints)

Net-change constraint for fact f

$$G(f) - S(f) = \sum_{\substack{o \text{ always} \\ \text{produces } f}} Y_o + \sum_{\substack{o \text{ sometimes} \\ \text{produces } f}} Y_o - \sum_{\substack{o \text{ always} \\ \text{consumes } f}} Y_o - \sum_{\substack{o \text{ sometimes} \\ \text{consumes } f}} Y_o$$

- Special cases for goal and initial state
  - Add/Subtract one from net change
- Special case for operators that might produce/consume[1]

[1] Task normalization can get rid of this special case.

State-equation Heuristic (Constraints)

Net-change constraint for fact f

$$G(f) - S(f) = \sum_{\substack{o \text{ always} \\ \text{produces } f}} Y_o + \sum_{\substack{o \text{ sometimes} \\ \text{produces } f}} Y_o - \sum_{\substack{o \text{ always} \\ \text{consumes } f}} Y_o - \sum_{\substack{o \text{ sometimes} \\ \text{consumes } f}} Y_o$$

- Special cases for goal and initial state
  - Add/Subtract one from net change
- Special case for operators that might produce/consume[1]
  - Use upper bound and inequality instead of equality

[1] Task normalization can get rid of this special case.

State-equation Heuristic (Constraints)

Net-change constraint for fact f

$$G(f) - S(f) \leq \sum_{\substack{o \text{ always} \\ \text{produces } f}} Y_o + \sum_{\substack{o \text{ sometimes} \\ \text{produces } f}} Y_o - \sum_{\substack{o \text{ always} \\ \text{consumes } f}} Y_o$$

- Special cases for goal and initial state
  - Add/Subtract one from net change
- Special case for operators that might produce/consume[1]
  - Use upper bound and inequality instead of equality

[1]Task normalization can get rid of this special case.

State-equation Heuristic (Constraints)

Net-change constraint for fact f

$$G(f) - S(f) \leq \sum_{\substack{o \text{ always} \\ \text{produces } f}} Y_o + \sum_{\substack{o \text{ sometimes} \\ \text{produces } f}} Y_o - \sum_{\substack{o \text{ always} \\ \text{consumes } f}} Y_o$$

- Special cases for goal and initial state
  - Add/Subtract one from net change
- Special case for operators that might produce/consume[1]
  - Use upper bound and inequality instead of equality

[1] Task normalization can get rid of this special case.

Tutorial Structure

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

Potential Heuristics

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:

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

  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

Features

Definition (feature)

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

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:

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

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

The End

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

Thank you for your attention!