# LP-based Heuristics for Cost-optimal Classical Planning

## 1. Introduction and Overview

Florian Pommerening Gabriele Röger Malte Helmert

ICAPS 2015 Tutorial

June 7, 2015

# About This Tutorial

# About Us

Florian

Gabi

Malte

Questions? Don't be shy to talk to us and/or email!


- florian.pommerening@unibas.ch
- gabriele.roeger@unibas.ch
- malte.helmert@unibas.ch

# Target Audience

## Target Audience

Ideally:

- You know what classical planning is.
  keywords: STRIPS, $SAS^+$
- You know what planning as heuristic search is.
  keywords: $A^*$, admissible heuristic, consistent heuristic
- You are familiar with major concepts of planning heuristics.
  keywords: abstraction, landmarks, delete relaxation

Please ask questions at any time!

# Tutorial Topic

## Dissecting the Title

LP-based Heuristics for Cost-optimal Classical Planning

# Tutorial Topic

## Dissecting the Title

LP-based Heuristics for Cost-optimal Classical Planning

- Find path from initial to goal state
  in declaratively specified state space

# Tutorial Topic

## Dissecting the Title

LP-based Heuristics for Cost-optimal Classical Planning

- Find path from initial to goal state
  in declaratively specified state space
- with minimal total cost

# Tutorial Topic

## Dissecting the Title

LP-based Heuristics for Cost-optimal Classical Planning

- Find path from initial to goal state in declaratively specified state space
- with minimal total cost
- using heuristic search algorithms

# Tutorial Topic

## Dissecting the Title

LP-based Heuristics for Cost-optimal Classical Planning

- Find path from initial to goal state
  in declaratively specified state space
- with minimal total cost
- using heuristic search algorithms
- with cost estimates based on linear programming.

# Tutorial Topic

## Dissecting the Title

LP-based Heuristics for Cost-optimal Classical Planning

- Find path from initial to goal state
  in declaratively specified state space
- with minimal total cost
- using heuristic search algorithms
- with cost estimates based on linear programming.

# Background: 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:

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

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

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

$\rightsquigarrow$ 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).

# Some LP Theory: Duality

Some LP theory: Every LP has an alternative view (its dual).

- roughly: variables and constraints swap roles
- dual of maximization LP is minimization LP and vice versa
- same objective value if one exists
- dual of dual: original LP

# Three Key Ideas in This Tutorial

# Cost Partitioning

## Idea 1: Cost Partitioning

- create copies $\Pi_1, \ldots, \Pi_n$ of planning task $\Pi$
- each has its own operator cost function $cost_i$ (otherwise identical to $\Pi$)
- 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 $f_1, \dots, f_n$.
- 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

# 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