University of Basel

Enhancing Efficiency of LP-based Heuristic Search in Optimal Planning

Renato Farruggio <renato.farruggio@unibas.ch>

University of Basel

08.04.2024

Objectives

- Investigate different heuristics
- Find optimal parameters of LP-solver to compute a given heuristic

- Compare hypotheses with a benchmark

- Improve overall performance

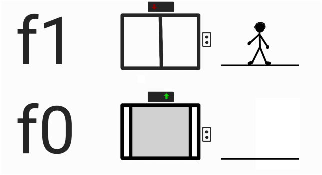
Presentation overview

- Background
  - Optimal Planning
  - Linear Programming
  - Heuristics
- Experiments and Results
- Conclusions
- Outlook

Optimal Planning

Planning

Planning is the assignment of finding a sequence of actions that leads from an initial state to a goal state in a predefined environment.



- E.g. Rubik's Cube, 15 Tile Puzzle, Elevator (Miconic)
- Different formalism: STRIPS, SAS+, PDDL, ...

Planning, formally

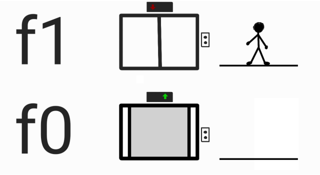
Definition (STRIPS Planning Task)

A STRIPS Planning Task is a 5-tuple $\Pi = \langle P, I, A, G, cost \rangle$, where

- P is a finite set of boolean state variables, called **atomic propositions**
- I is the initial state
- A is a set of actions
  Each action a is a triple of sets of atomic propositions (pre(a), add(a), del(a))
- G is the set of goal conditions
- *cost* is a function that maps all actions to real numbers

STanford **R**esearch **I**nstitute **P**roblem **S**olver (Fikes & Nilsson, 1971)

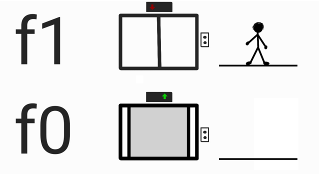
STRIPS planning task of the elevator example



STRIPS planning task of the elevator example

$\Pi = \langle P, I, A, G, cost \rangle$ with

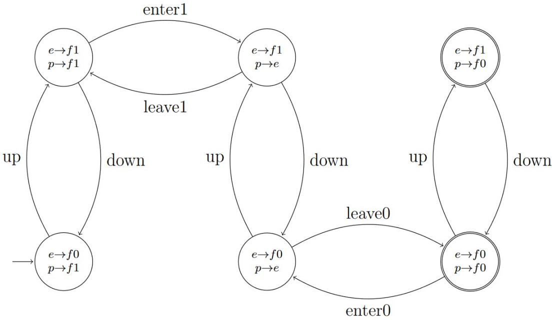
- $P = \{e \mapsto f0, e \mapsto f1, p \mapsto f0, p \mapsto f1, p \mapsto e\}$
- $I = \{e \mapsto f0, p \mapsto f1\}$
- $G = \{p \mapsto f0\}$
- $A = \{up, down, enter0, enter1, leave0, leave1\}$, where
  - $up = (\{e \mapsto f0\}, \{e \mapsto f1\}, \{e \mapsto f0\})$
  - $down = (\{e \mapsto f1\}, \{e \mapsto f0\}, \{e \mapsto f1\})$
  - $enter0 = (\{e \mapsto f0, p \mapsto f0\}, \{p \mapsto e\}, \{p \mapsto f0\})$
  - $enter1 = (\{e \mapsto f1, p \mapsto f1\}, \{p \mapsto e\}, \{p \mapsto f1\})$
  - $leave0 = (\{e \mapsto f0, p \mapsto e\}, \{p \mapsto f0\}, \{p \mapsto e\})$
  - $leave1 = (\{e \mapsto f1, p \mapsto e\}, \{p \mapsto f1\}, \{p \mapsto e\})$
- $cost(a) = 1$ for all actions a in A

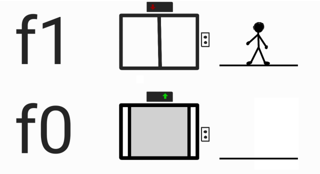


State space of elevator example

State space consists of

- States
- Actions (or operations)





Important definitions

- Task: One instance of a problem with atomic propositions, initial state, goal conditions, etc.
  - E.g. Elevator planning task
- Problem: The assignment of calculating a heuristic value for one state of a task
  - E.g. Calculating the heuristic value for one state
- Plan: A sequence of actions that end in a goal state
  - E.g. $\pi = \langle up, down, up, enter1, leave1, enter1, down, leave0 \rangle$
- Cost of a plan: Sum of all action costs of a plan π
  - E.g. π has a cost of 8
- Optimal Plan: A plan with minimal cost
  - E.g. $\langle up, enter1, down, leave1 \rangle$

Heuristic

- Estimation of actual cost from a given state to the goal state
- h : States $\rightarrow \mathbb{R}$

- Not exact, but relatively cheap to calculate
- Used for an informed guess on where to go next

- Example:
  - Estimate road length from Basel to Paris with the straight line distance (413 km)

Larger examples

- Elevator example is tiny
- Solution is obvious
- What if there are more floors, elevator, and passengers?

Fast Downward Planning System

- Developed by the AI research group at the University of Basel
- Open Source
- Offers several heuristics
  - We use 6 of them
- Offers several search algorithms
  - We used A*
- Input: Task
  - E.g. Elevator example
- Output: Optimal plan
  - E.g. $\langle up, enter1, down, leave0 \rangle$

Linear Programs (LPs)

Linear Program (LP)

- Standardized optimization problem
- Minimize (or maximize) linear **objective function** subject to **constraints**

Linear Program (LP)

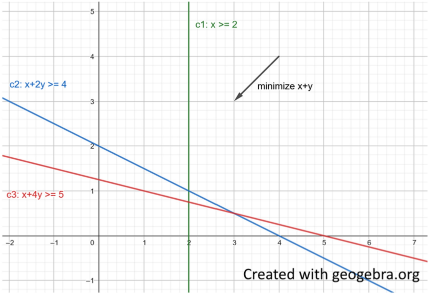
- Standardized optimization problem
- Minimize (or maximize) linear **objective function** subject to **constraints**

$$
\begin{aligned}
\text{Minimize} \quad & x + y & \\
\text{subject to} \quad & x \geq 2 & \text{(c1)} \\
& x + 2y \geq 4 & \text{(c2)} \\
& x + 4y \geq 5 & \text{(c3)}
\end{aligned}
$$

Linear Program (LP)

- Standardized optimization problem
- Minimize (or maximize) linear **objective function** subject to **constraints**

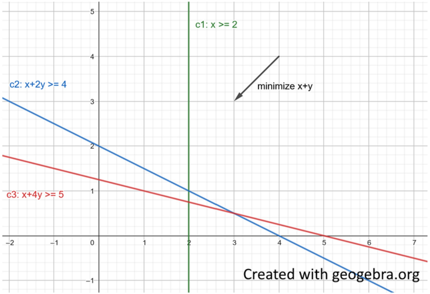
$$
\begin{aligned}
\text{Minimize} \quad & x + y & \\
\text{subject to} \quad & x \geq 2 & \text{(c1)} \\
& x + 2y \geq 4 & \text{(c2)} \\
& x + 4y \geq 5 & \text{(c3)}
\end{aligned}
$$



Linear Program (LP)

- Standardized optimization problem
- Minimize (or maximize) linear **objective function** subject to **constraints**

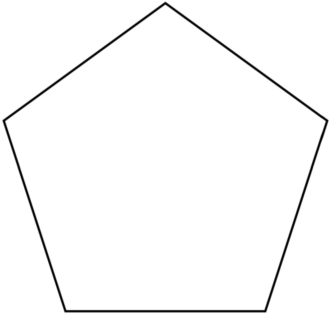
$$\begin{aligned} \text{Minimize} \quad & x + y & \\ \text{subject to} \quad & x \geq 2 & \text{(c1)} \\ & x + 2y \geq 4 & \text{(c2)} \\ & x + 4y \geq 5 & \text{(c3)} \end{aligned}$$

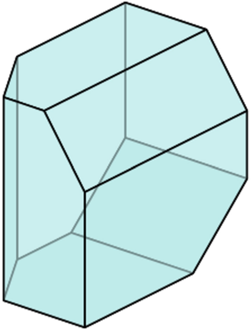


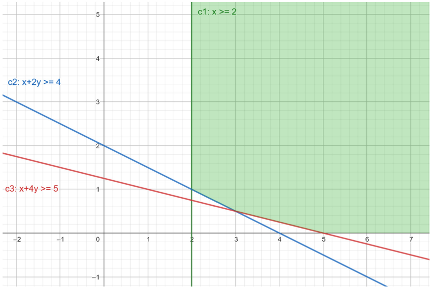
Solution: $x = 2, y = 1 \rightarrow x + y = 3$

Linear Program: Feasibility

- Feasible point: Point where no constraint is violated
- Feasible region: Set of all feasible points
- Feasible region of LP: Convex polytope
  - E.g. polygon (2D), polyhedron (3D), unbounded polytope







Linear Program (LP)

- Standardized optimization problem
- Minimize (or maximize) linear **objective function** subject to **constraints**

$$
\begin{aligned}
\text{Minimize} \quad & x + y & \\
\text{subject to} \quad & x \geq 2 & \text{(c1)} \\
& x + 2y \geq 4 & \text{(c2)} \\
& x + 4y \geq 5 & \text{(c3)}
\end{aligned}
$$

Linear Program: Matrix formulation

- Standardized optimization problem
- Minimize (or maximize) linear **objective function** subject to **constraints**

$$\begin{aligned} \text{Minimize} \quad & x + y & \\ \text{subject to} \quad & x \geq 2 & \text{(c1)} \\ & x + 2y \geq 4 & \text{(c2)} \\ & x + 4y \geq 5 & \text{(c3)} \end{aligned}$$

$$\text{Minimize} \quad \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\text{subject to} \quad \begin{pmatrix} 1 & 0 \\ 1 & 2 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} 2 \\ 4 \\ 5 \end{pmatrix}$$

Linear Program: Matrix formulation

- Standardized optimization problem
- Minimize (or maximize) linear **objective function** subject to **constraints**

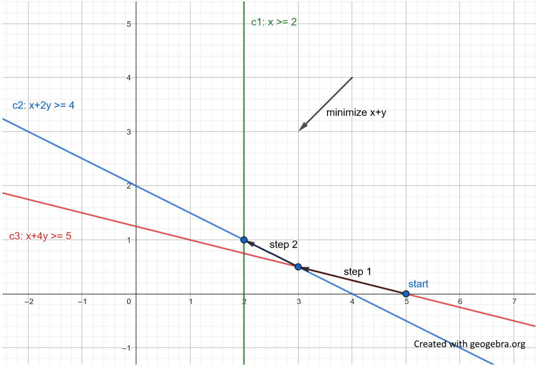
$$\begin{aligned} \text{Minimize} \quad & x + y & \\ \text{subject to} \quad & x \geq 2 & \text{(c1)} \\ & x + 2y \geq 4 & \text{(c2)} \\ & x + 4y \geq 5 & \text{(c3)} \end{aligned}$$

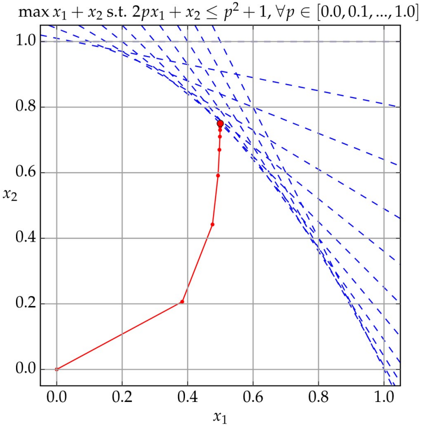
$$\begin{aligned} \text{Minimize} \quad & \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\ \text{subject to} \quad & \begin{pmatrix} 1 & 0 \\ 1 & 2 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} 2 \\ 4 \\ 5 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} \text{Minimize} \quad & c^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \geq b \end{aligned}$$

Solving LPs

Solving LPs: Simplex & Interior Points Method





IBM ILOG CPLEX Optimization Studio

- Short: **CPLEX**
- Commercial solver by IBM
- Not open-source
- Solves optimization problems, such as LPs
  - Input: Linear Program (LP)
  - Output: Solution
- In our case:
  - Input: Heuristic formulated as LP
  - Output: Solution (heuristic value)

Preprocessing

- Remove redundant constraints
  - E.g. $x \geq 2$ and $2x \geq 4$
  - E.g. $x \geq 0$ and $x \geq 1$
- Replace fixed variables
  - E.g. $x \geq 1$ and $x \leq 1$
- Presolve problem
  - E.g. $x + y \geq 5$ and $x + y + z \geq 7$
- Etc.

- Usually: Less variables and constraints, but constraint matrix A gets more dense.

Warm starts

- Problem: Finding an initially feasible basis can be hard
- Idea: Keep the solution loaded in the solver
- Use this solution as initial variable assignments for simplex algorithm

Heuristics

Heuristic

- Heuristic assigns numbers to states
- Estimation of actual cost
- $h : \text{States} \rightarrow \mathbb{R}$

- Examples:
  - Count number of passengers at the desired floor
  - Perfect heuristic: Actual cost

State Equation Heuristic (SEH)

- Main idea: Consider only the net change of each fact

Minimize

$$\sum_{a \in A} cost(a) \mathrm{X}_a$$

subject to

$$\sum_{a \text{ produces } p} \mathrm{X}_a - \sum_{a \text{ consumes } p} \mathrm{X}_a \geq G(p) - I(p) \quad \forall p \in P$$

$$\mathrm{X}_a \geq 0 \quad \forall a \in A$$

State Equation Heuristic (SEH)

- Main idea: Consider only the net change of each fact
- Example: Elevator task, LP for SEH of initial state

Minimize

$$X_{\text{up}} + X_{\text{down}} + X_{\text{enter0}} + X_{\text{enter1}} + X_{\text{leave0}} + X_{\text{leave1}}$$

subject to

$$\begin{aligned}
p \mapsto e: \quad & X_{\text{enter0}} + X_{\text{enter1}} \\
& -X_{\text{leave0}} - X_{\text{leave1}} && \geq 0 \\
p \mapsto f0: \quad & X_{\text{leave0}} - X_{\text{enter0}} && \geq 1 \\
p \mapsto f1: \quad & X_{\text{leave1}} - X_{\text{enter1}} && \geq -1 \\
e \mapsto f0: \quad & X_{\text{down}} - X_{\text{up}} && \geq -1 \\
e \mapsto f1: \quad & X_{\text{up}} - X_{\text{down}} && \geq 0 \\
\text{for all } a \in A \quad & X_a && \geq 0
\end{aligned}$$

Tested heuristics

- State Equation Heuristic (SEH)
- Delete-Relaxation Heuristic (DEL)
- Post-Hoc Optimization Heuristic (PHO)
- Optimal Cost Partitioning of Disjunctive Action Landmarks Heuristic (OCP)
- Initial State Potential Heuristic (IPOT)
- Diverse Potential Heuristic (DPOT)

"Putting it all together"

"Enhancing Efficiency of LP-based Heuristic Search in Optimal Planning"

- Planning
  - Finding a plan in a predefined environment
- Optimal planning
  - Finding a plan with minimal costs
- Heuristic-based optimal planning
  - Utilize heuristics in the search algorithm
  - E.g. A* algorithm
- LP-based heuristics in Optimal Planning
  - This is what we wanted to enhance

General strategy in A* (heuristic search)

- In each step, calculate the heuristic values of all reachable states and choose the most promising state.
- "most promising" means least number of
  - Actual cost from initial state to new state, plus
  - Estimated cost from new state to goal state

Our work

- CPLEX is a black box
- CPLEX is used with default settings
- Our goal was to find better CPLEX settings

Experiments and results

Experimental approach

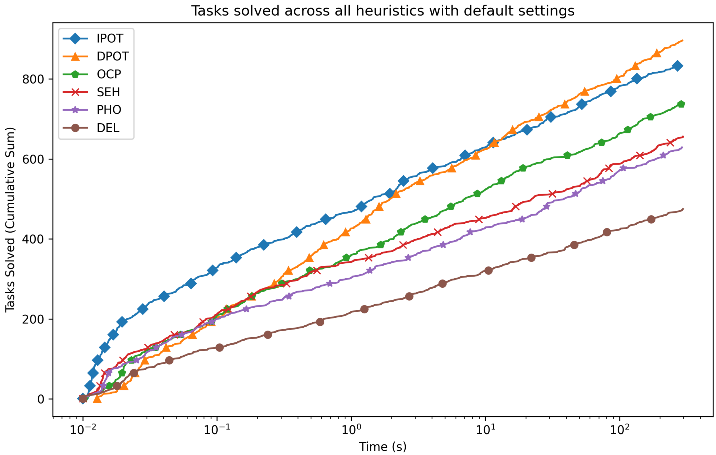
- Formulating hypotheses was impractical
- As alternative: Run tests and interpret results
- Find possible reasons for improved performance

Benchmarking

- Benchmarking set with 1827 tasks

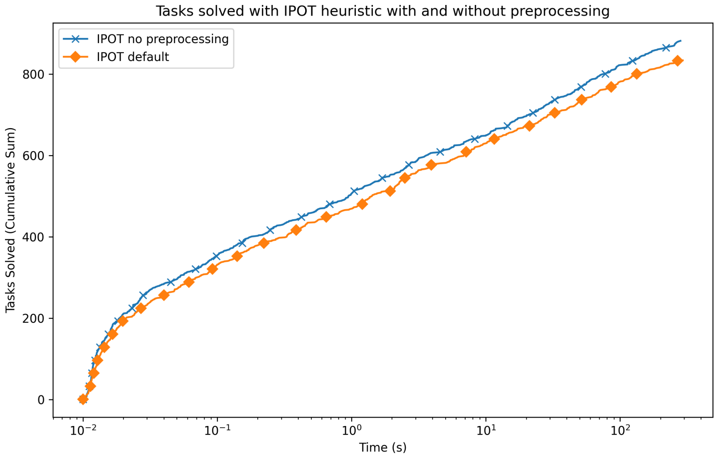
- Limit time to 5 minutes per task
- Limit memory usage to 3.5gb per task
- Single-threaded

Baseline

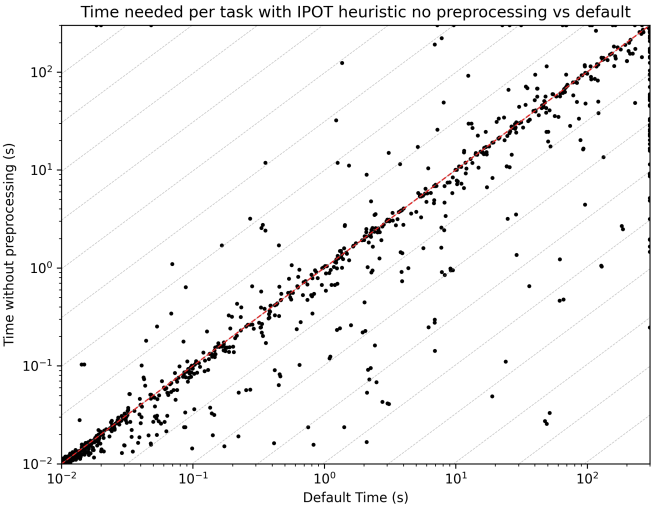


Tasks solved across all heuristics with default settings
IPOT
DPOT
OCP
SEH
PHO
DEL
Tasks Solved (Cumulative Sum)
0
200
400
600
800
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Time (s)

Disabling preprocessing for Initial State Potential Heuristic



Disabling preprocessing for Initial State Potential Heuristic

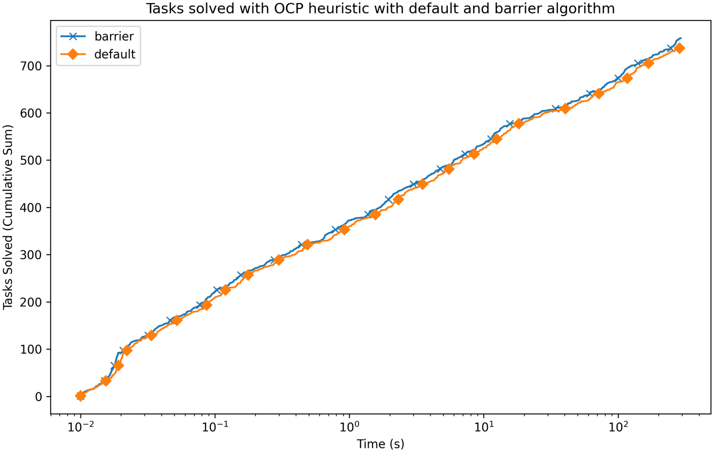


Reasons for improvement – Initial State Potential Heuristic without preprocessing

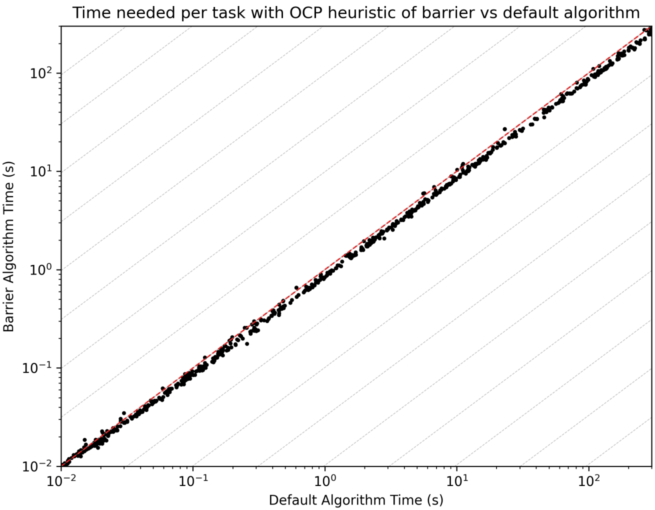
- Potential heuristics are only calculated once per Task
- Preprocessing is expensive, and not worth it in this case

- Why so much spread?
- Ignores all facts that do not occur in the initial state
  - Much faster if we are lucky
  - Much slower if we are unlucky

Using Barrier Method for Optimal Cost Partitioning Heuristic



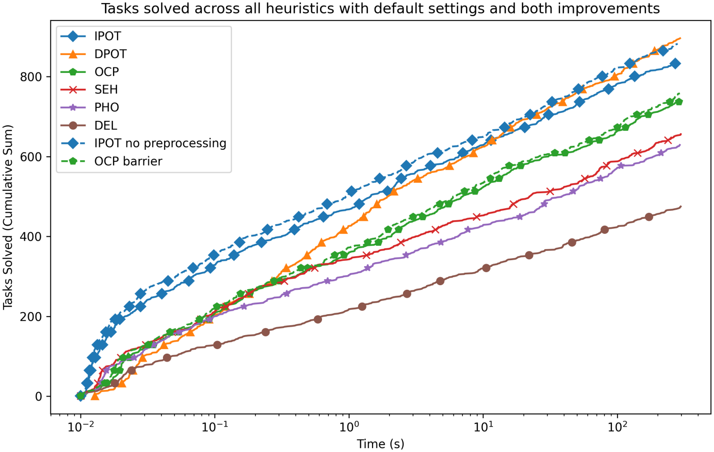
Using Barrier Method for Optimal Cost Partitioning Heuristic



Reasons for improvement – Optimal Cost Partitioning using Barrier method

- LPs can change a lot from one state to the next
- Old solution becomes infeasible
- Many cold starts
- Simplex cannot take advantage of warm starts

Results


Tasks solved across all heuristics with default settings and both improvements
IPOT
DPOT
OCP
SEH
PHO
DEL
IPOT no preprocessing
OCP barrier
Tasks Solved (Cumulative Sum)
0
200
400
600
800
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Time (s)

Conclusions

Conclusions

- In most cases CPLEX works well with default settings
- There is room for improvement

- Two improvements:
  - OCP works better with the barrier algorithm
  - IPOT works better with preprocessing disabled

Outlook

- Further investigate shared properties of improved/worsened tasks
- Further investigate preprocessing options and their impact
- Investigate "barrier ordering parameter" to speed up cholesky decomposition of barrier algorithm
- Investigate impact of these settings in other variants of the heuristics
- Investigate impact of these settings in other heuristics
- Try quadratic programming with barrier method
- Try Integer Programming (without the LP-relaxation)

Thank you!

Image sources

- All images are taken from wikipedia if not specified otherwise, except for the plots
- We have created all plots
- The Fast Downward logo was taken from www.fast-downward.org
- The elevator task sketch is an own creation
- The person in the elevator task sketch was kindly drawn by Katharina Pêtre

Additional material

Barrier Function

- In each step, find the minimum of a barrier function
- Want to find minimum of the barrier function at each iteration
- Can be approximated using the gauss-newton algorithm
- Compute Cholesky decomposition
- This is the bottleneck of this method

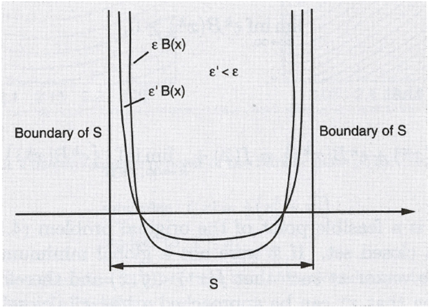
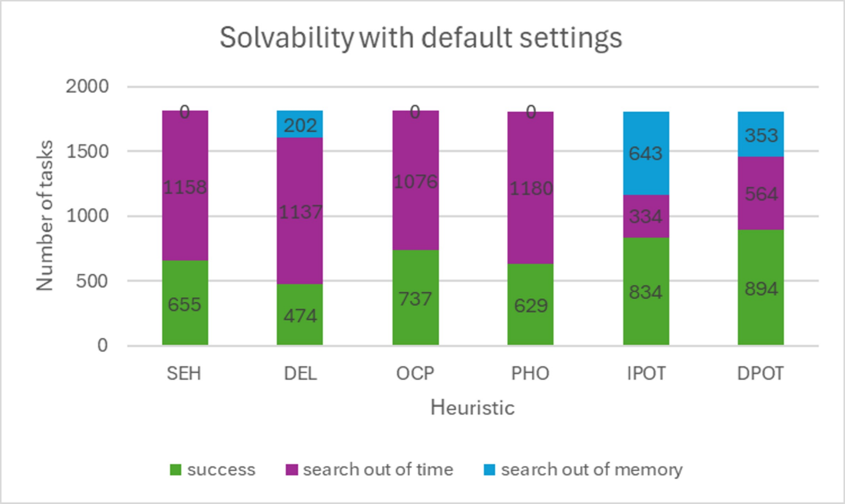


Image source:
Nonlinear Programming (Dimitri P. Bertsekas)

Default CPLEX parameters

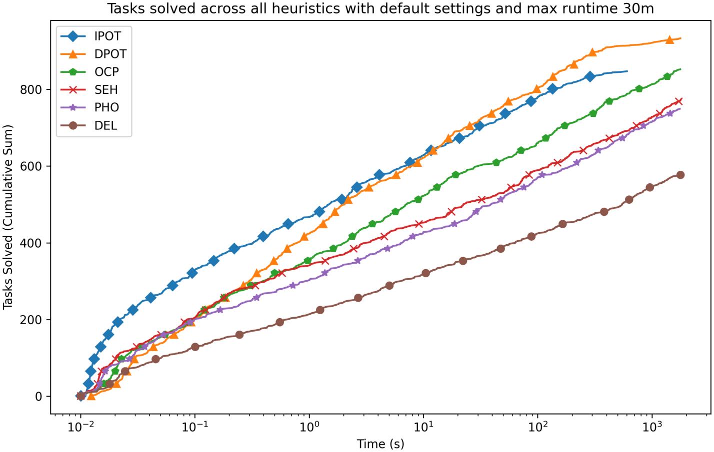
- Currently: Don't set any CPLEX parameters, let CPLEX choose
- CPLEX uses:
  - Simplex algorithm for every LP
  - Preprocessing is set to automatic (let CPLEX choose)
  - Warm starts problems

Baseline

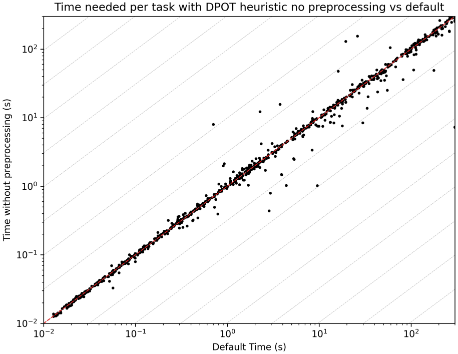


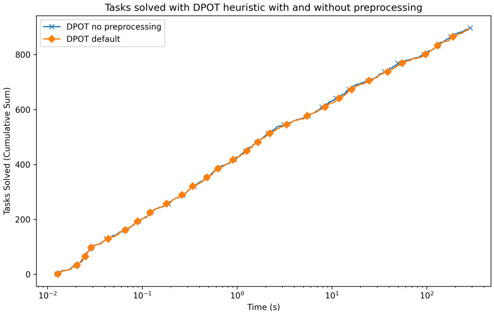
Solvability with default settings
2000
1500
1000
500
0
Number of tasks
0
1158
655
202
1137
474
0
1076
737
0
1180
629
643
334
834
353
564
894
SEH
DEL
OCP
PHO
IPOT
DPOT
Heuristic
success
search out of time
search out of memory

Baseline, given 30m instead of 5m

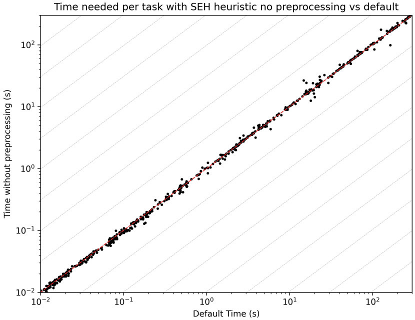


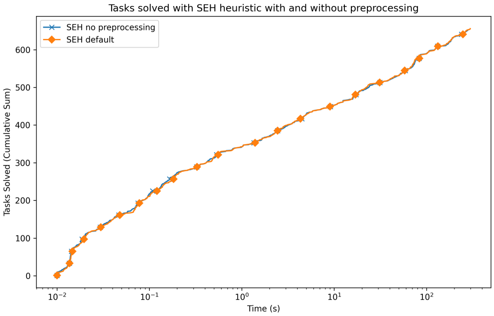
Disabling preprocessing for Diverse Potentials Heuristics



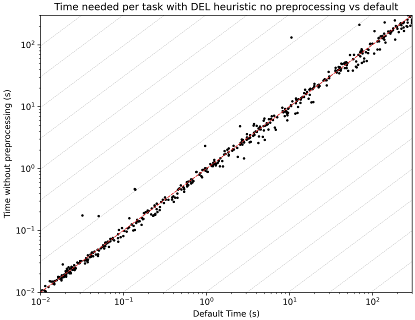


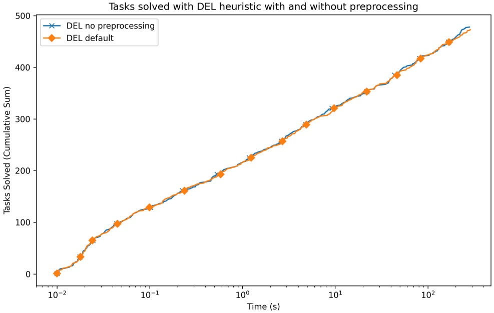
Disabling preprocessing for State Equation Heuristics



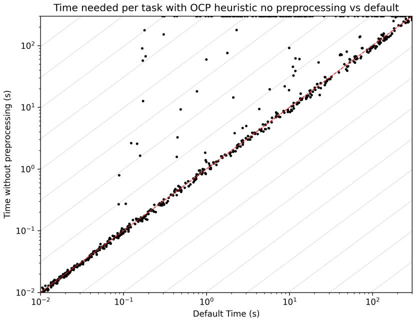


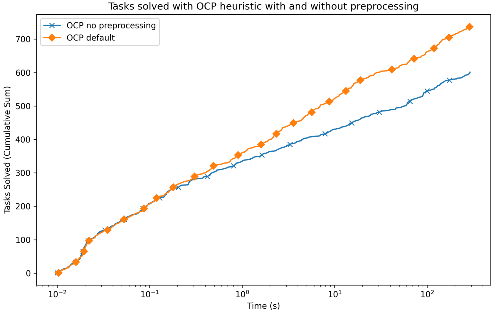
Disabling preprocessing for Delete Relaxation Heuristics





Disabling preprocessing for Optimal Cost Partitioning of Disjunctive Action Landmarks Heuristics





Disabling preprocessing for Post-Hoc Optimization Heuristics