

Heuristic Search Planner Techniques

Dra. M^a Dolores Rodríguez Moreno

Objectives

Specific Objectives

- Heuristic search planning techniques

Source

- Stuart Russell & Peter Norvig (2009). Artificial Intelligence: A Modern Approach. Chapter 3, 10. (3rd Edition). Ed. Pearson
- Dana Nau's slides for Automated Planning. Licensed under License <https://creativecommons.org/licenses/by-nc-sa/2.0/>
- Jorg Hoffmann. FF: The Fast-Forward Planning System. AAAI'01

Outline

- Motivation
- Introduction
- Heuristics
 - Max-cost and additive cost
 - Delete-Relaxation
 - Landmarks
- FF
- Conclusions

Motivation

- Neither forward nor backward search is efficient without a good heuristic function
- How we can introduce domain-independent heuristics in planning?

Outline

- Motivation
- **Introduction**
- Heuristics
 - Max-cost and additive cost
 - Delete-Relaxation
 - Landmarks
- FF
- Conclusions

Introduction (I)

- Heuristic: method to decide among alternative courses of action to find the most effective way to achieve a goal (Pearl, 1984)
- Perfect heuristic $h^*(s) = \min\{c(s, a)\}$
- Admissible heuristic $h(s) \leq h^*(s) \forall s$ (often weak)
- Non-admissible heuristic (most planners)

Introduction (II)

- Heuristic Search Planners (HSP) transform planning problems into heuristic search problems extracting heuristics functions, rather than enter them by hand
- Problems
 - Number of explored nodes is very high
 - Heuristic calculation in each step
- Examples of heuristics:
 - HSP: additive (h^{add})
 - FF: delete-relaxation

Outline

- Motivation
- Introduction
- Heuristics
 - Max-cost and additive cost
 - Delete-Relaxation
 - Landmarks
- FF
- Conclusions

Heuristics (I)

Ignoring Deletes

- h^{\max}
- h^+

Abstractions

- PDB
- M&S

Landmarks

- h^{LM}

Critical Paths

- h^1
- h^2
- h^3
- ...

[\(Source\)](#)

Heuristics (II)

- Ignoring deletes: simplifies the world assuming that anything that was ever true in the past will remain true in the future
- Landmark: something that has to be true at some point along any plan
- Abstraction: ignores part of the description of the problem
 - i.e 15 puzzle, ignoring tiles 8-15 and uses tiles 1-7 to estimate the distances in the real puzzle
- Critical Paths: it looks at subproblems of fixed size. It is parametrized

Heuristics (III)

- Returns an estimate $h(s)$ of the minimum cost $h^*(s)$ of getting from the state s to a goal state
- If the algorithm always finds optimal solutions, then the heuristic function will be admissible
- They can be domain-specific or domain-independent
 - Max-Cost and Additive Cost (h^{\max} & h^{add})
 - Delete-Relaxation (h^+)
 - Landmarks (h^{LM})

Outline

- Motivation
- Introduction
- Heuristics
 - Max-cost and additive cost
 - Delete-Relaxation
 - Landmarks
- FF
- Conclusions

Max-cost heuristic

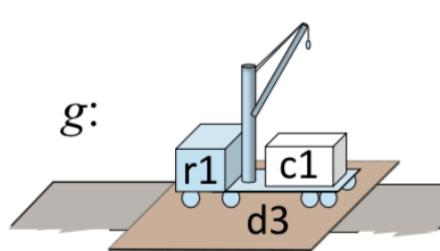
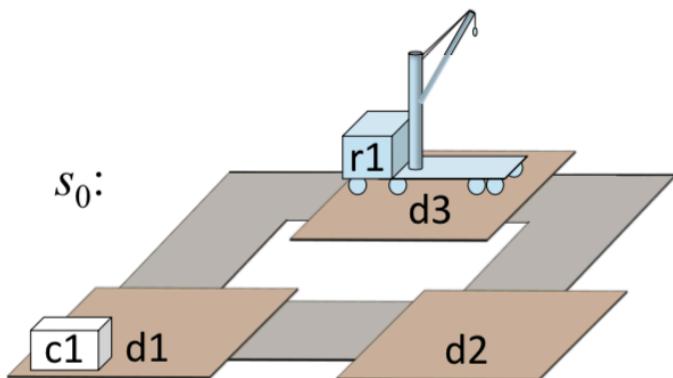
- The max-cost of a set of literals $g = \{g_1, \dots, g_k\}$ is defined recursively as the largest max-cost of each g_i individually, where each g_i 's max-cost is the minimum, over all actions that can produce g_i , of the action's cost plus the max-cost of its preconditions
- Computation of h^{\max} can be visualized as an And/Or search going backward from g
- Admissible
- Planning: a goal (i.e., a set of literals such as g or the preconditions of an action) can be reached by achieving just one of the goal's literals, namely, the one that is the most expensive to achieve

Additive heuristic

- h^{add} is similar to h^{max} but adds the costs of each set of literals rather than taking their maximum
- Not admissible (better in practice)

Heuristic calculation: example (I)

$B = \text{Robots} \cup \text{Docks} \cup \text{Containers} \cup \{\text{nil}\};$
 $\text{Robots} = \{r_1\};$
 $\text{Docks} = \{d_1, d_2, d_3\};$
 $\text{Containers} = \{c_1\}.$



load(r, c, l)
pre: $\text{cargo}(r) = \text{nil}$, $\text{loc}(c) = l$, $\text{loc}(r) = l$
eff: $\text{cargo}(r) \leftarrow c$, $\text{loc}(c) \leftarrow r$
cost: 1

unload(r, c, l)
pre: $\text{cargo}(r) = c$, $\text{loc}(r) = l$
eff: $\text{cargo}(r) \leftarrow \text{nil}$, $\text{loc}(c) \leftarrow l$
cost: 1

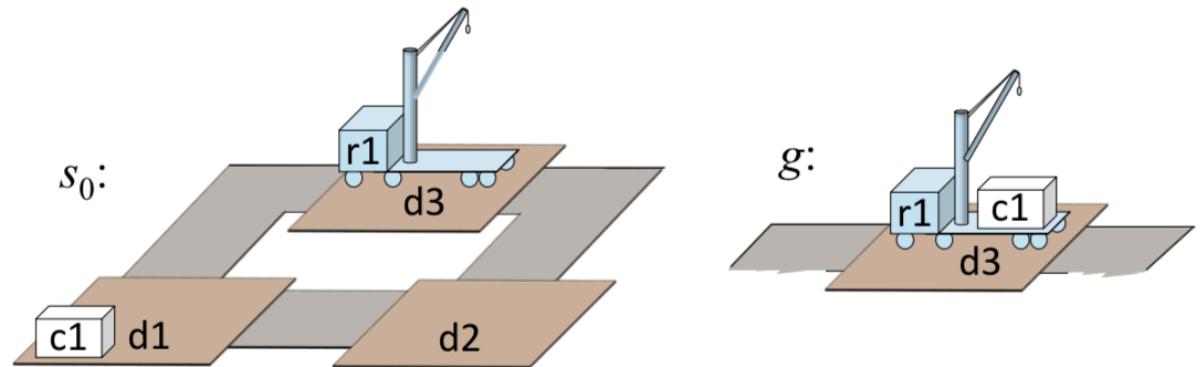
move(r, d, e)
pre: $\text{loc}(r) = d$
eff: $\text{loc}(r) \leftarrow e$
cost: 1

$s_0 = \{\text{loc}(r_1) = d_3, \text{cargo}(r_1) = \text{nil}, \text{loc}(c_1) = d_1\};$
 $g = \{\text{loc}(r_1) = d_3, \text{loc}(c_1) = r_1\}.$

Heuristic calculation: example (II)

- Actions?

- $a_1 = \text{move}(r_1, d_3, d_1)$
- $a_2 = \text{move}(r_1, d_3, d_2)$



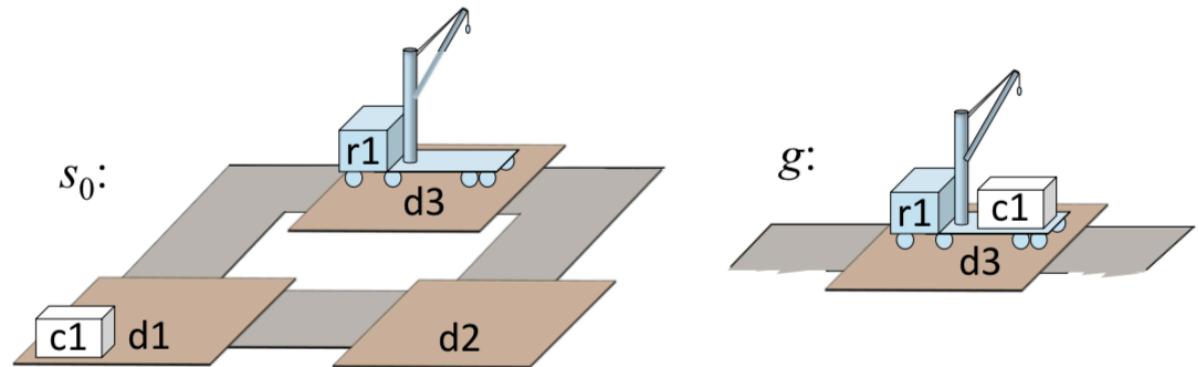
$$s_1 = \gamma(s_0, a_1) = \{\text{loc}(r_1) = d_1, \text{cargo}(r_1) = \text{nil}, \text{loc}(c_1) = d_1\}$$

$$s_2 = \gamma(s_0, a_2) = \{\text{loc}(r_1) = d_2, \text{cargo}(r_1) = \text{nil}, \text{loc}(c_1) = d_1\}$$

Heuristic calculation: example (II)

- Actions?

- $a_1 = \text{move}(r_1, d_3, d_1)$
- $a_2 = \text{move}(r_1, d_3, d_2)$

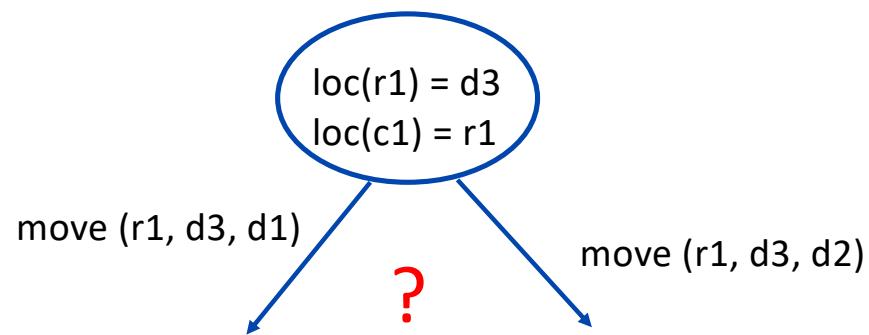


$$s_1 = \gamma(s_0, a_1) = \{\text{loc}(r_1) = d_1, \text{cargo}(r_1) = \text{nil}, \text{loc}(c_1) = d_1\}$$

$$s_2 = \gamma(s_0, a_2) = \{\text{loc}(r_1) = d_2, \text{cargo}(r_1) = \text{nil}, \text{loc}(c_1) = d_1\}$$

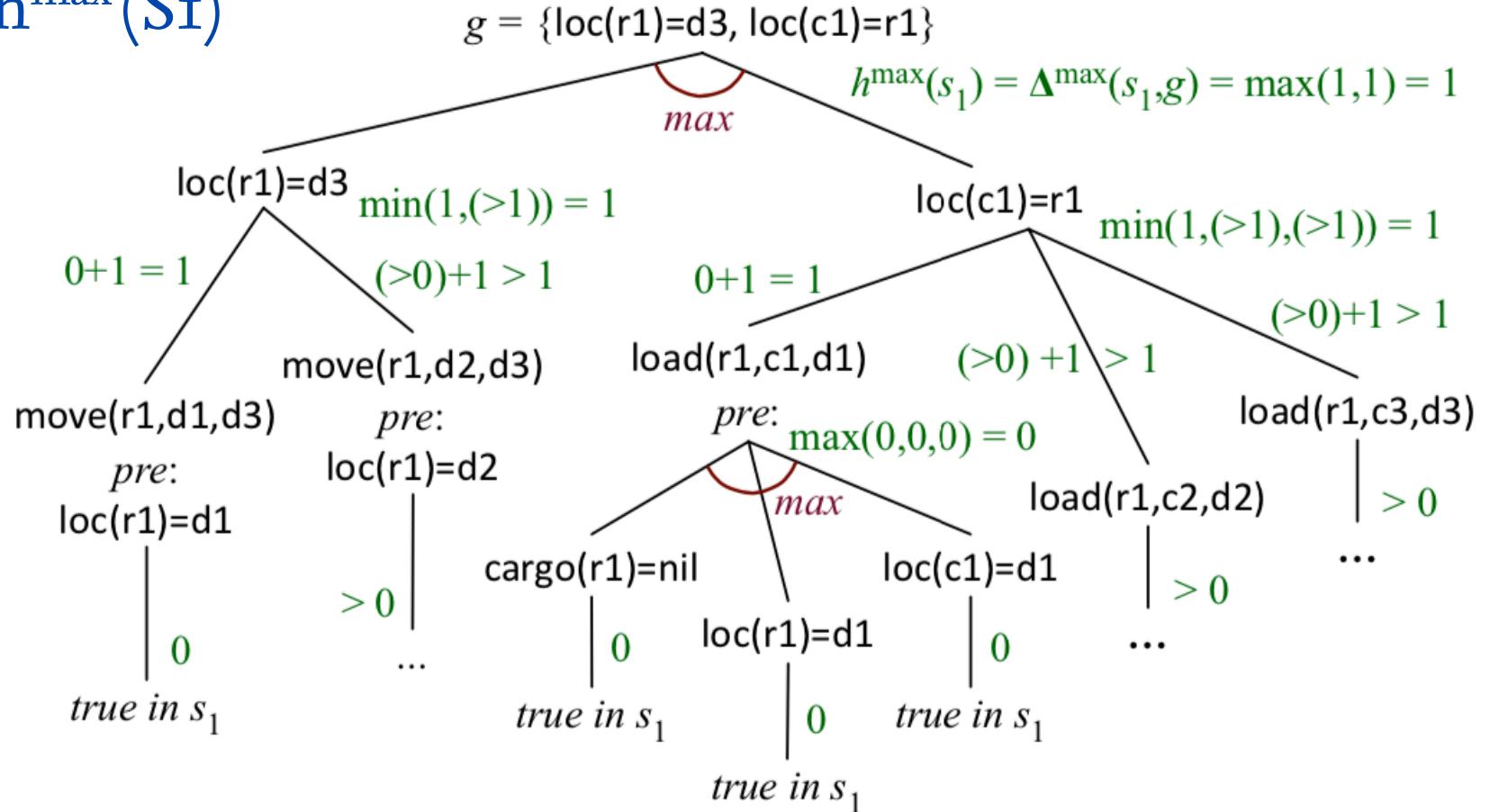
Can you calculate the value of h^{add} and h^{max} for S_2 ? We just calculate for S_1

Search tree

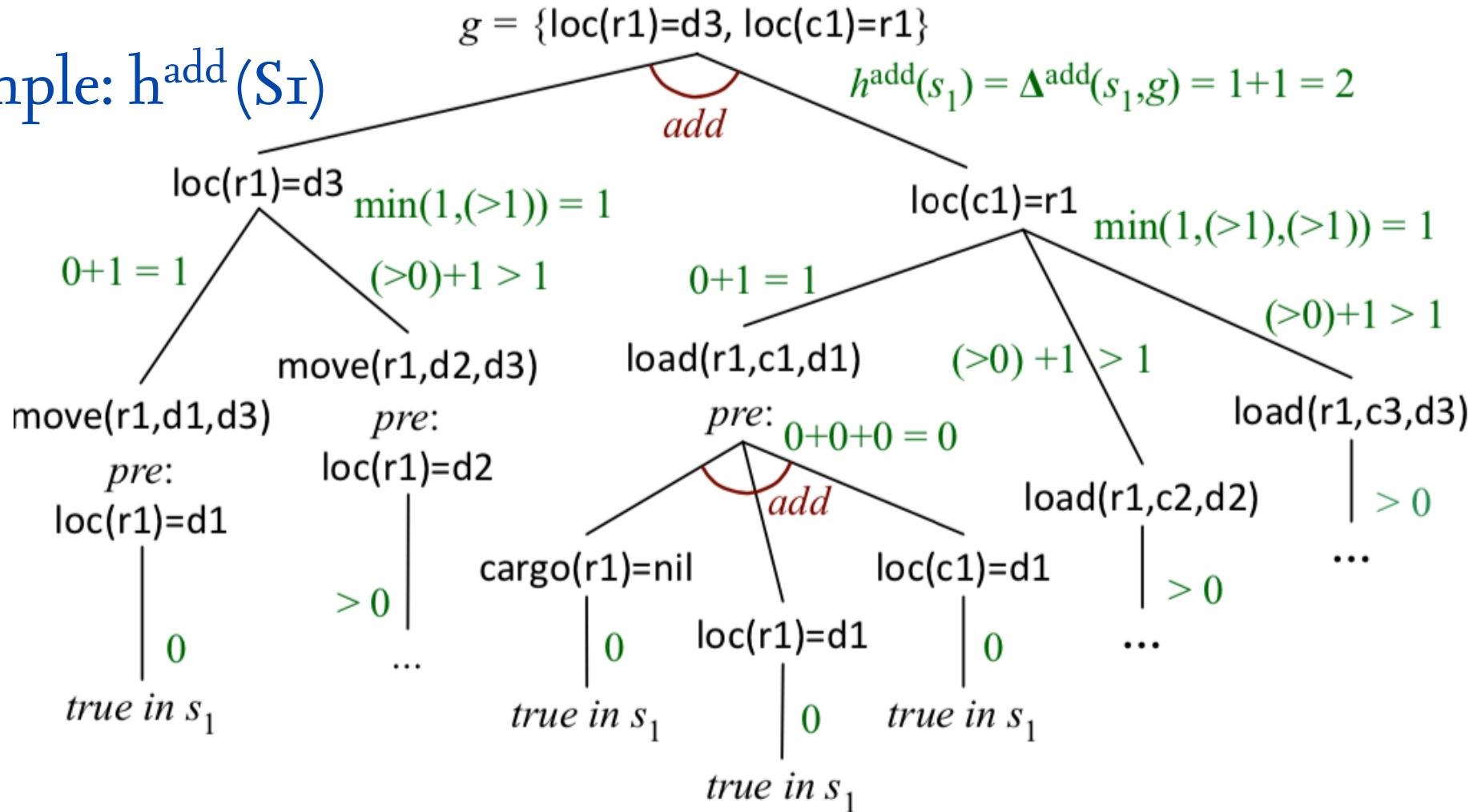


$s_0 = \{\text{loc}(r1) = d3, \text{cargo}(r1) = \text{nil}, \text{loc}(c1) = d1\};$
 $g = \{\text{loc}(r1) = d3, \text{loc}(c1) = r1\}.$

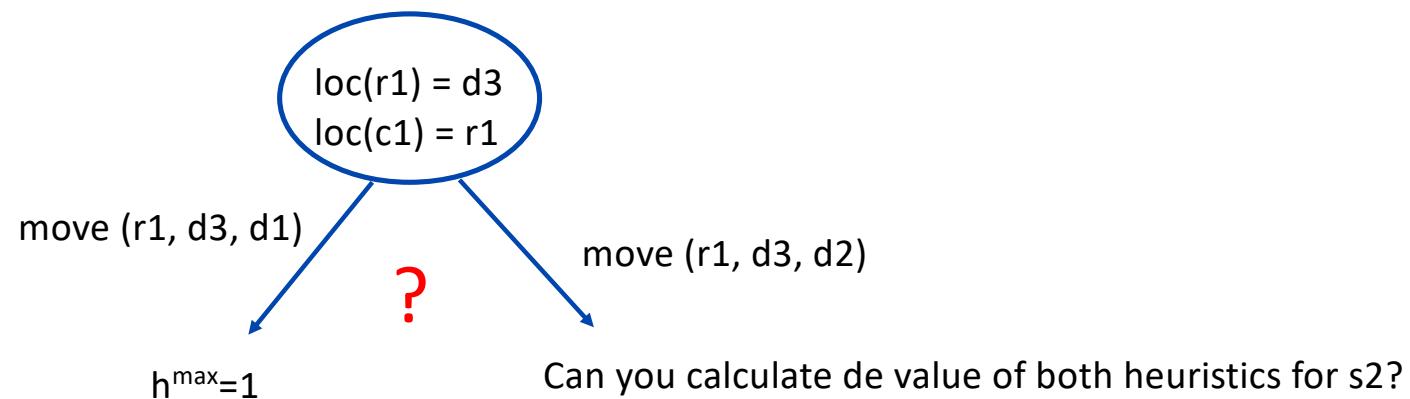
Example: $h^{\max}(S_1)$



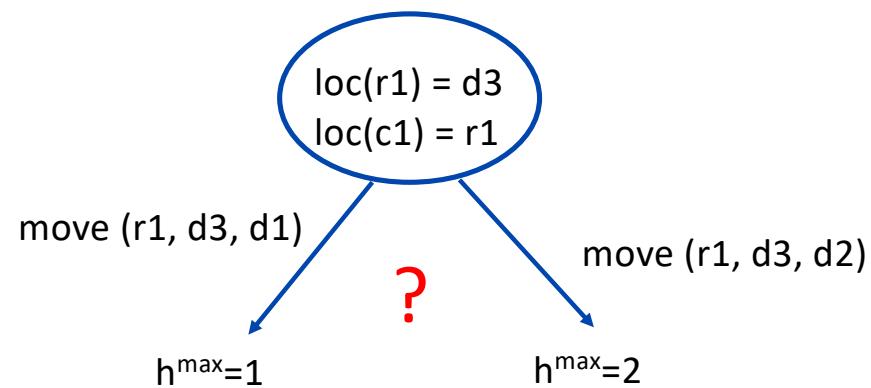
Example: $h^{\text{add}}(S_1)$



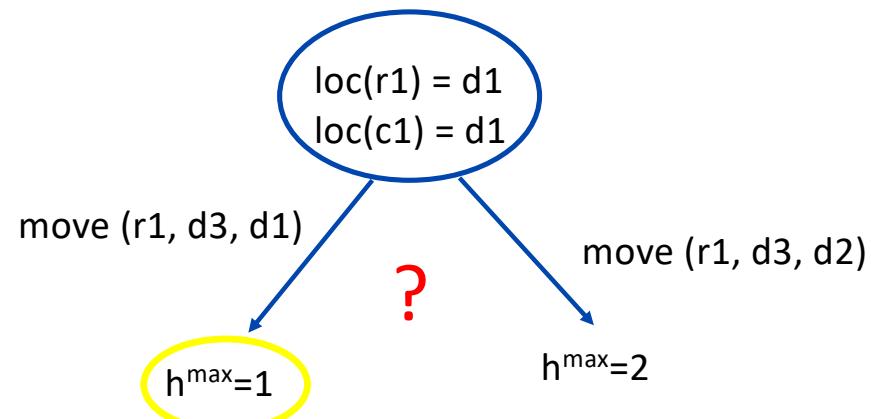
Search tree



Search tree



Search tree



Outline

- Motivation
- Introduction
- Heuristics
 - Max-cost and additive cost
 - Delete-Relaxation
 - Landmarks
- FF
- Conclusions

Delete-relaxation

- h^+ : applies new actions that never removes old atoms from a state (simply adds new ones)
- Admissible
- h^+ is expensive to compute: the problem of finding an optimal relaxed solution for a planning problem P is NP-hard
- Use in combination with a relaxed GP

Outline

- Motivation
- Introduction
- Heuristics
 - Max-cost and additive cost
 - Delete-Relaxation
 - Landmarks
- FF
- Conclusions

Landmarks

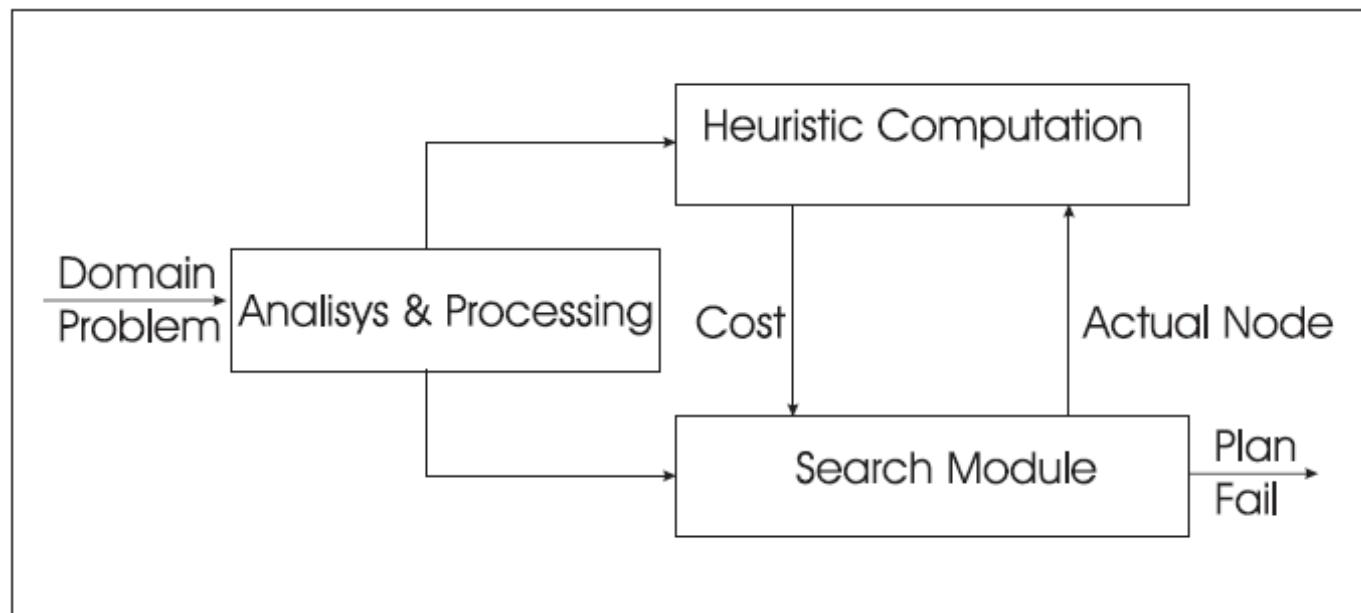
- To compute $h^{LM}(s_1)$, we count the number of landmarks between s_1 and g
- If we start in s_1 , then every solution plan must include a state in which $\text{cargo}(r_1) = c_1$
 - This is the only landmark for s_1
 - $h^{LM}(s_1) = 1$
- If we start in state s_2 , then the landmark computation will find two landmarks:
 - $\text{cargo}(r_1) = c_1$
 - $\text{loc}(r_1) = d_1$
 - $h^{LM}(s_2) = 2$

Outline

- Motivation
- Introduction
- Heuristics
 - Max-cost and additive cost
 - Delete-Relaxation
 - Landmarks
- FF: Fast-Forward
- Conclusions

FF (I)

- General architecture

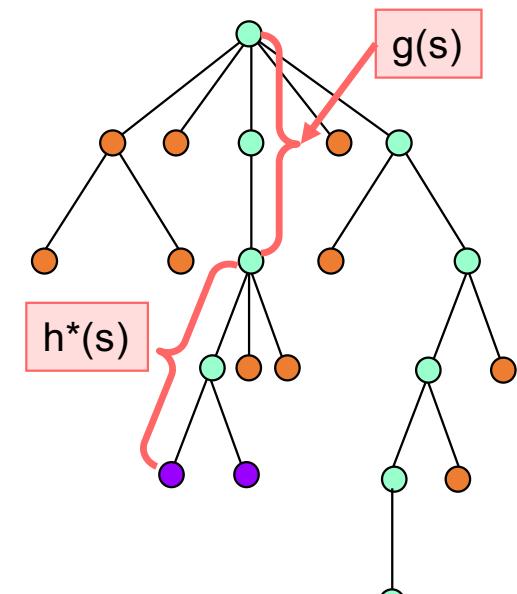


FF (II)

- **The Analysis & Processing module.** Analyze and process all the information from the domain and the initial state (table or vector with all the possible operators instantiated)
- **The Heuristic Computation module.** Computes the cost of applying a determined node in the search process
- **The Search module.** Depends on the heuristic computation, uses a search algorithm or a combination of them

FF search: A*

- For every state s , let
 - $g(s)$ = cost of the path from s_0 to s
 - $h^*(s)$ = least cost of all paths from s to goal nodes
 - $f^*(s) = g(s) + h^*(s)$ = least cost of all paths from s_0 to goal nodes that go through s
- Suppose $h(s)$ is an estimate of $h^*(s)$
 - Let $f(s) = g(s) + h(s)$
 - h is *admissible* if for every state s , $0 \leq h(s) \leq h^*(s)$
 - If h is admissible then f is a lower bound on f^* and A^* guarantees optimality
- In combination with Enforced Hill Climbing



FF: h (I)

Move (x,y)	Close
Pre: <i>in(x)</i>	Pre: <i>opened</i>
<i>opened</i>	Add: <i>closed</i>
Add: <i>in(y)</i>	Del: <i>opened</i>
Del: <i>in(x)</i>	
Open	Polish
Pre: <i>closed</i>	Pre: <i>opened</i>
Add: <i>opened</i>	Add: <i>polished</i>
Del: <i>closed</i>	Del:

In(A)

Closed



In(B)

Closed

Polished

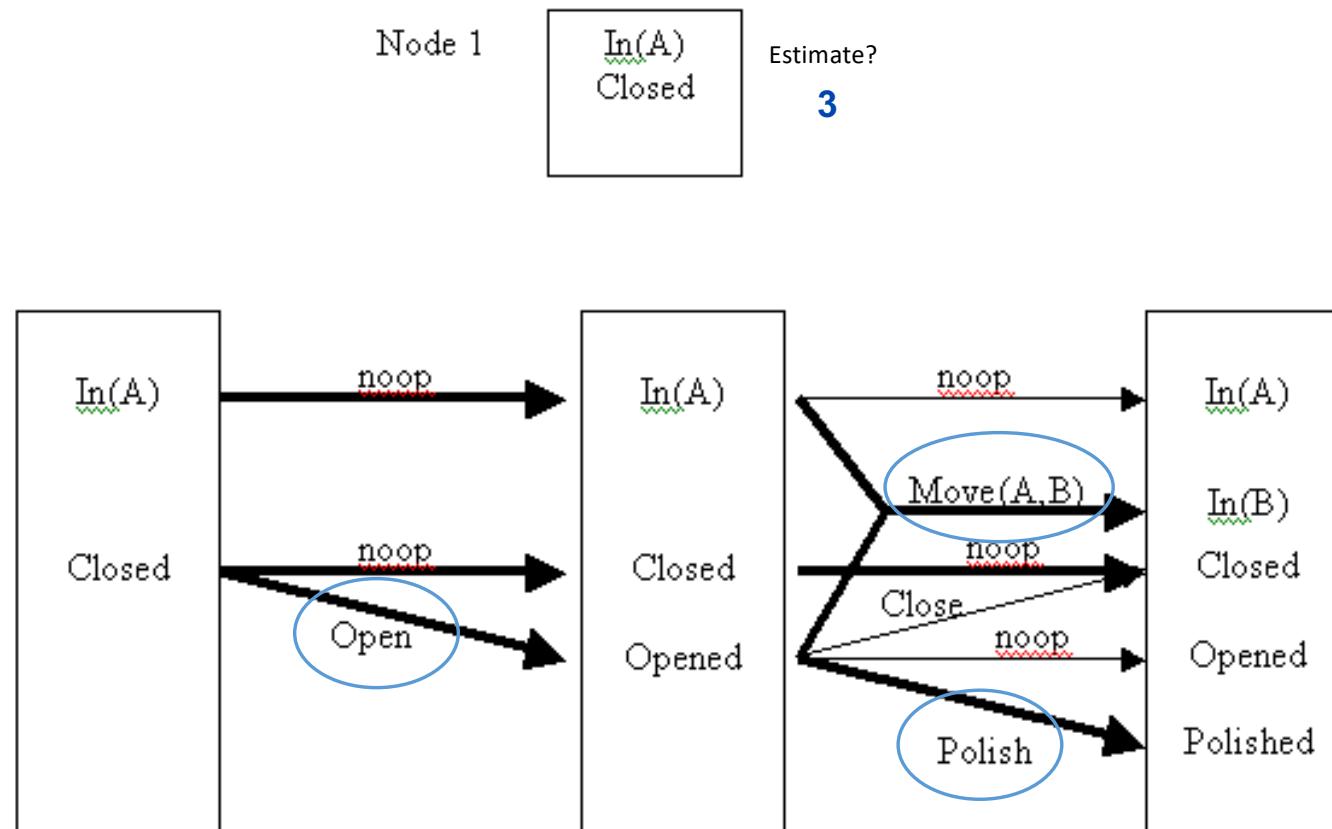
h⁺: number of operators in a relaxed GP



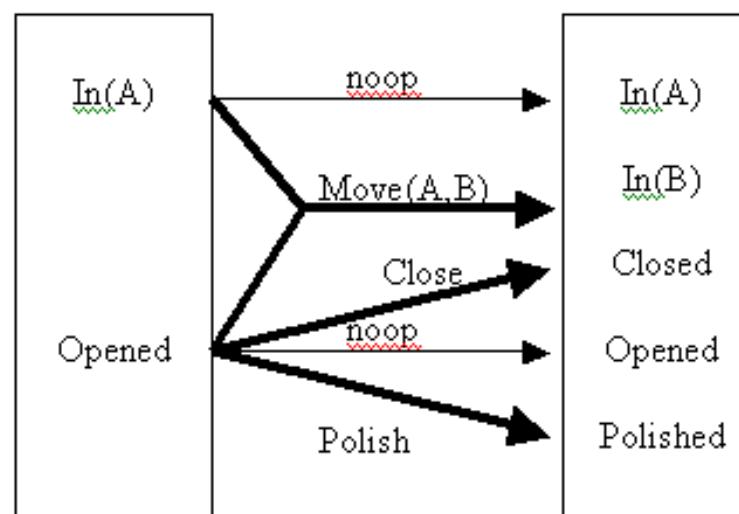
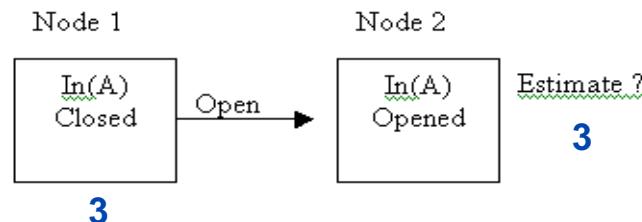
Universidad
de Alcalá

ISG

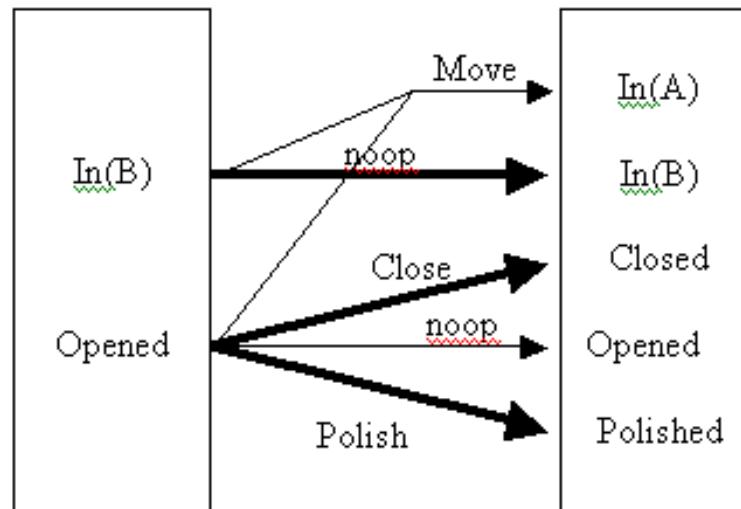
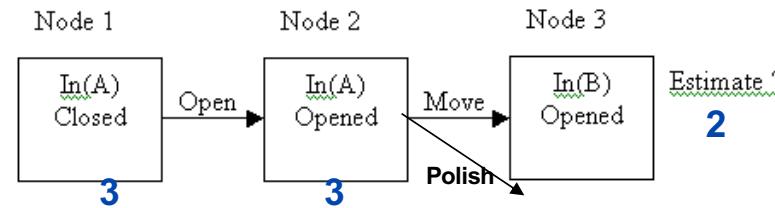
FF: h (II)



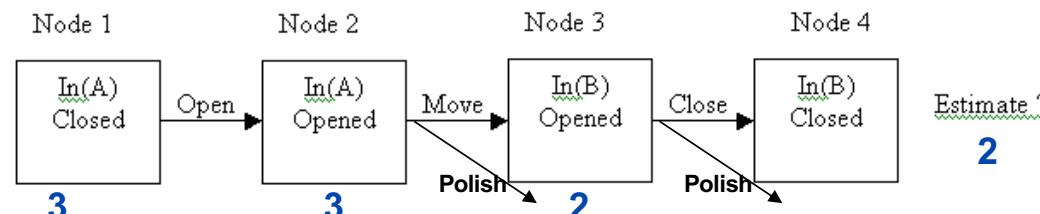
FF: h (III)



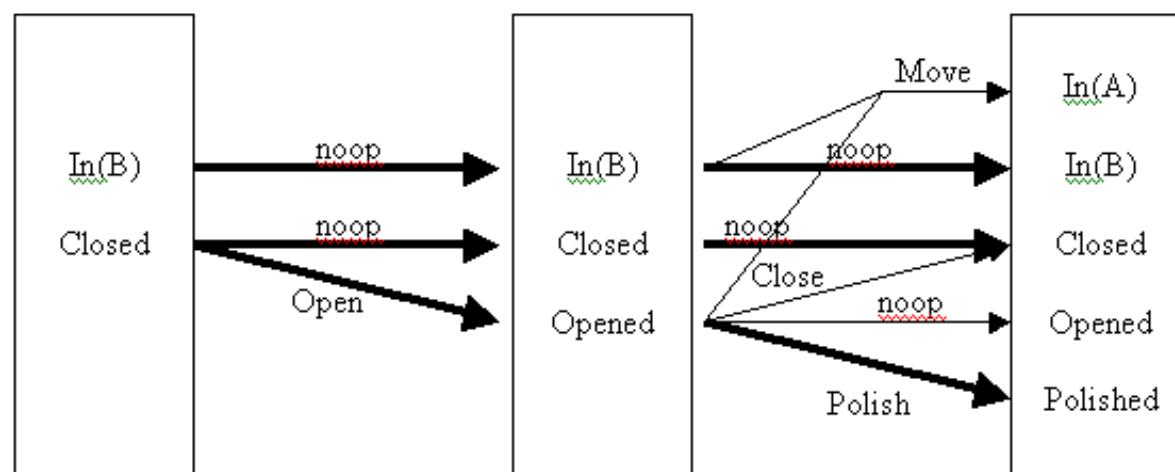
FF: h (IV)



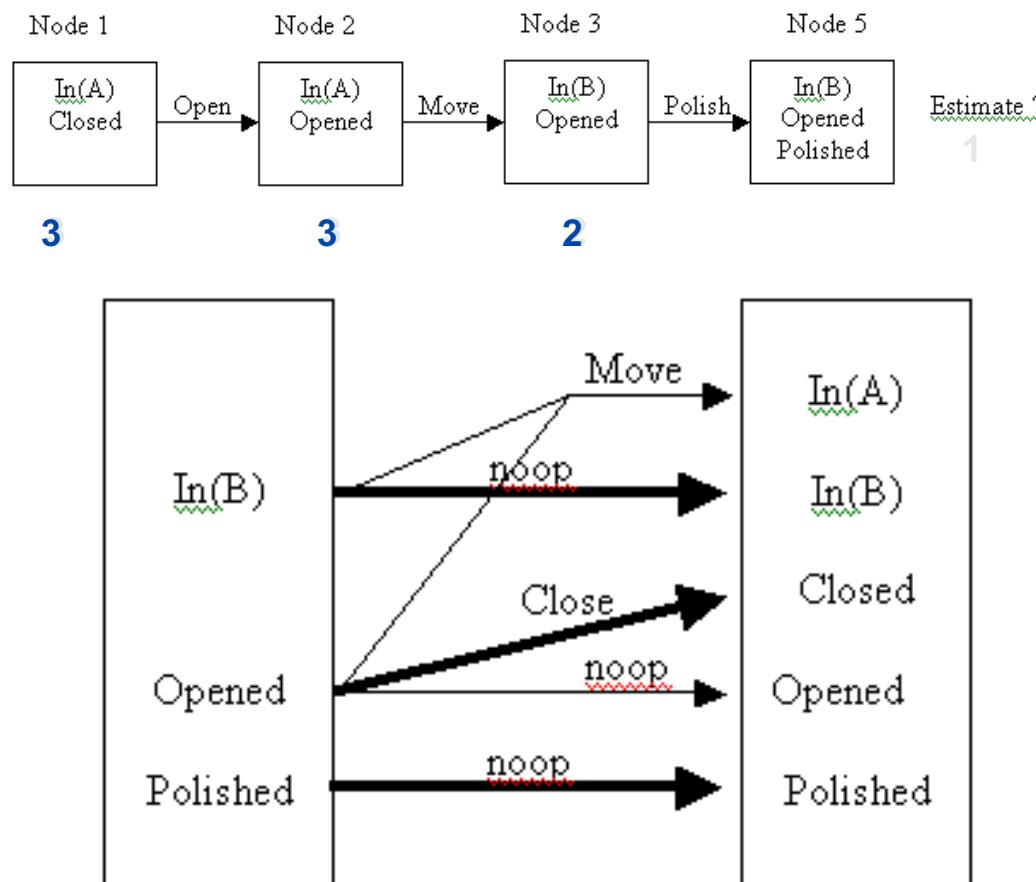
FF: h (V)



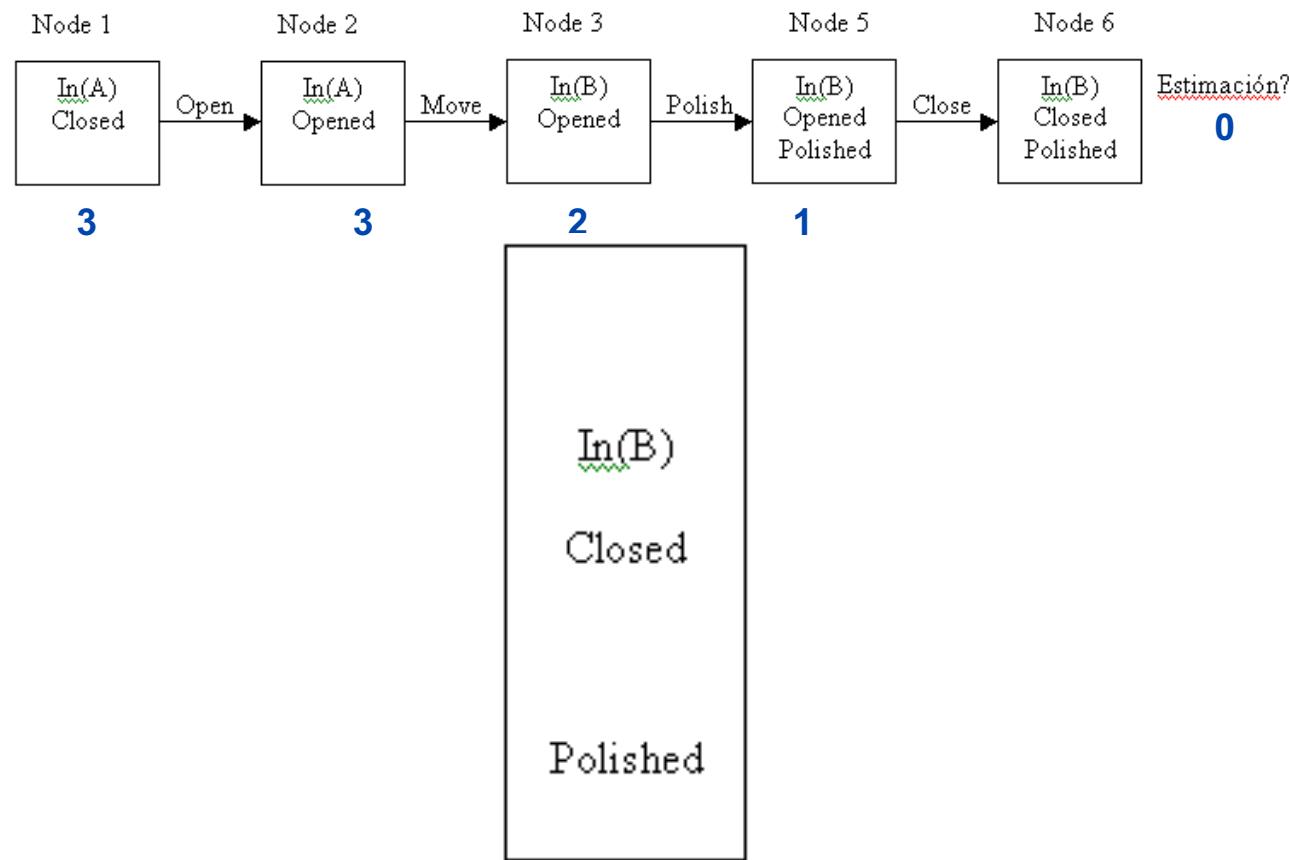
It does not improve the previous result, try Polish



FF: h (VI)



FF: h (VII)



Outline

- Motivation
- Introduction
- Heuristics
 - Max-cost and additive cost
 - Delete-Relaxation
 - Landmarks
- FF: Fast-Forward
- Conclusions

Conclusions

- HSP transforms planning problems into heuristic search problems
- FF: representative
 - h : relaxed GP (ignore delete list) and use the length as the heuristic
 - Is the heuristic admissible?
 - Performs forward state-space search (A*/EHC)

ToDo Example

- Calculate h^+ for the problem for the state S_1 move (r_1, d_3, d_1) and S_2 move (r_1, d_3, d_2)

load(r, c, l)

pre: $\text{cargo}(r) = \text{nil}$, $\text{loc}(c) = l$, $\text{loc}(r) = l$

eff: $\text{cargo}(r) \leftarrow c$, $\text{loc}(c) \leftarrow r$

cost: 1

$s_0 = \{\text{loc}(r1) = d3, \text{cargo}(r1) = \text{nil}, \text{loc}(c1) = d1\};$

$g = \{\text{loc}(r1) = d3, \text{loc}(c1) = r1\}.$

unload(r, c, l)

pre: $\text{cargo}(r) = c$, $\text{loc}(r) = l$

eff: $\text{cargo}(r) \leftarrow \text{nil}$, $\text{loc}(c) \leftarrow l$

cost: 1

move(r, d, e)

pre: $\text{loc}(r) = d$

eff: $\text{loc}(r) \leftarrow e$

cost: 1