Heuristic Search Planner Techniques

Dra. Mª 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. Pearsons
- 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 $g = \{g_r,...,g_k\}$ is defined recursively as the largest max-cost of each g_i individually, where each gi's max-cost is the minimum, over all actions that can produce gi, 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)

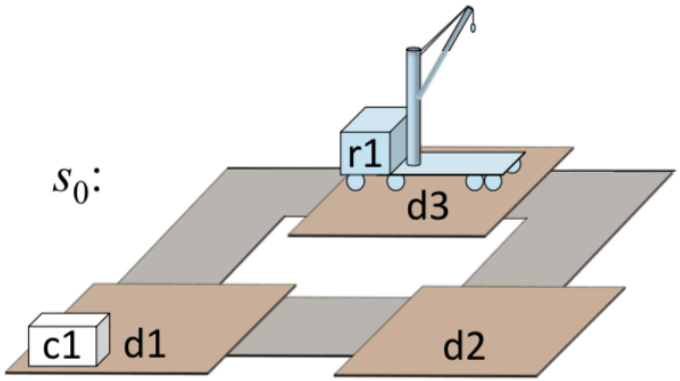
$$B = Robots \cup Docks \cup Containers \cup \{\mathsf{nil}\};$$
$$Robots = \{\mathsf{r}_1\};$$
$$Docks = \{\mathsf{d}_1, \mathsf{d}_2, \mathsf{d}_3\};$$
$$Containers = \{\mathsf{c}_1\}.$$

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

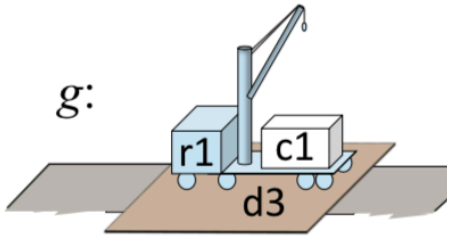
unload(r, c, l)
pre: cargo$(r) = c$, loc$(r) = l$
eff: cargo$(r) \leftarrow$ nil, loc$(c) \leftarrow l$
cost: 1

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

s_0:



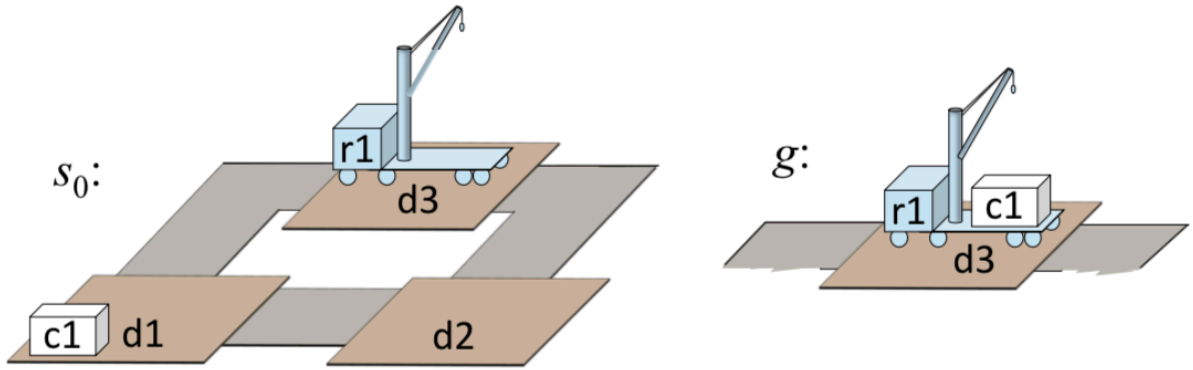
g:



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

Heuristic calculation: example (II)

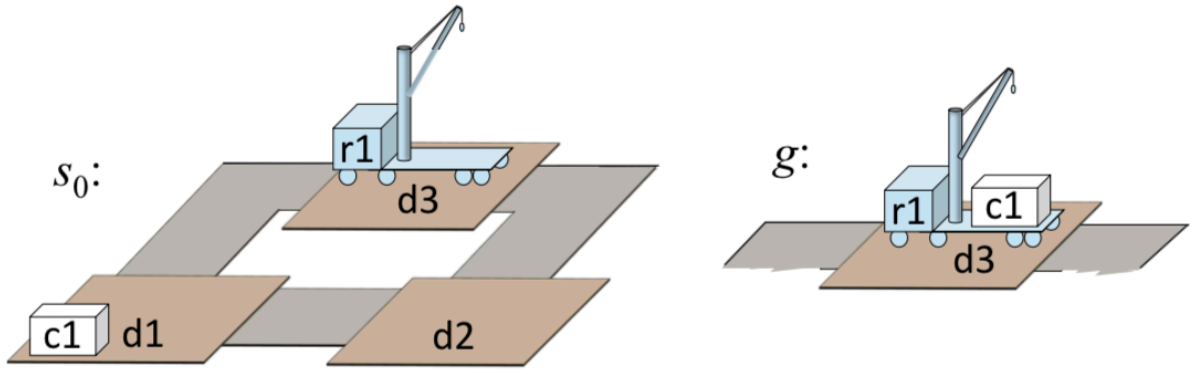
- Actions?
  - a1 = move (r1, d3, d1)
  - a2 = move(r1, d3, d2)



s1 = γ(s0, a1) = {loc(r1) = d1, cargo(r1) = nil, loc(c1) = d1}
s2 = γ(s0, a2) = {loc(r1) = d2, cargo(r1) = nil, loc(c1) = d1}

Heuristic calculation: example (II)

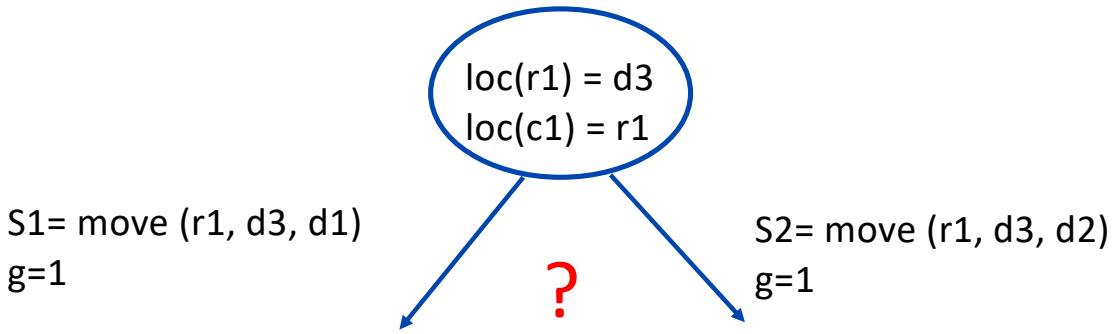
- Actions?
  - a1 = move (r1, d3, d1)
  - a2 = move(r1, d3, d2)



s1 = γ(s0, a1) = {loc(r1) = d1, cargo(r1) = nil, loc(c1) = d1}
s2 = γ(s0, a2) = {loc(r1) = d2, cargo(r1) = nil, loc(c1) = d1}

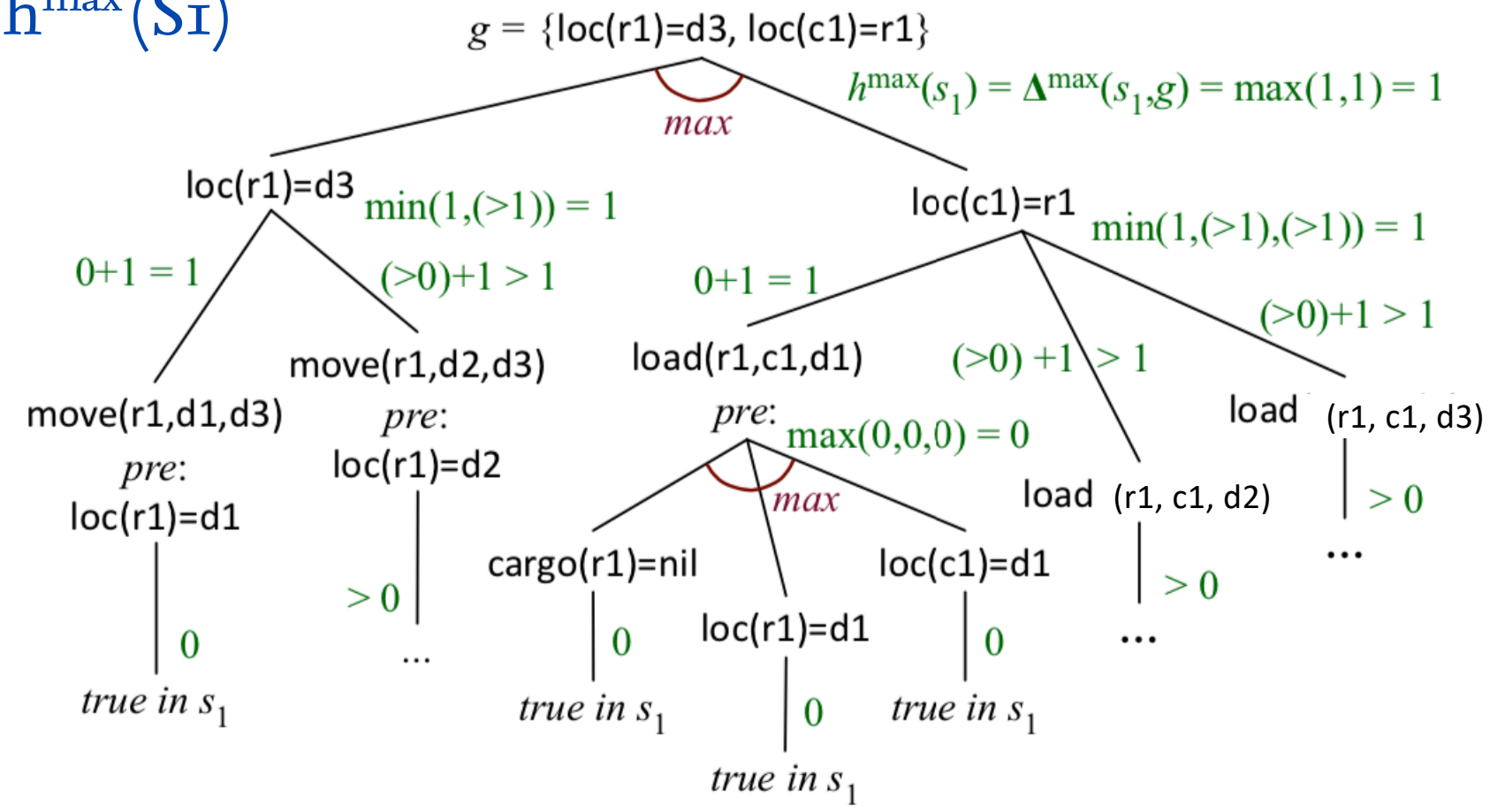
Can you calculate the value of h^{add} and h^{max} for S2? We just calculate for S1

Search tree

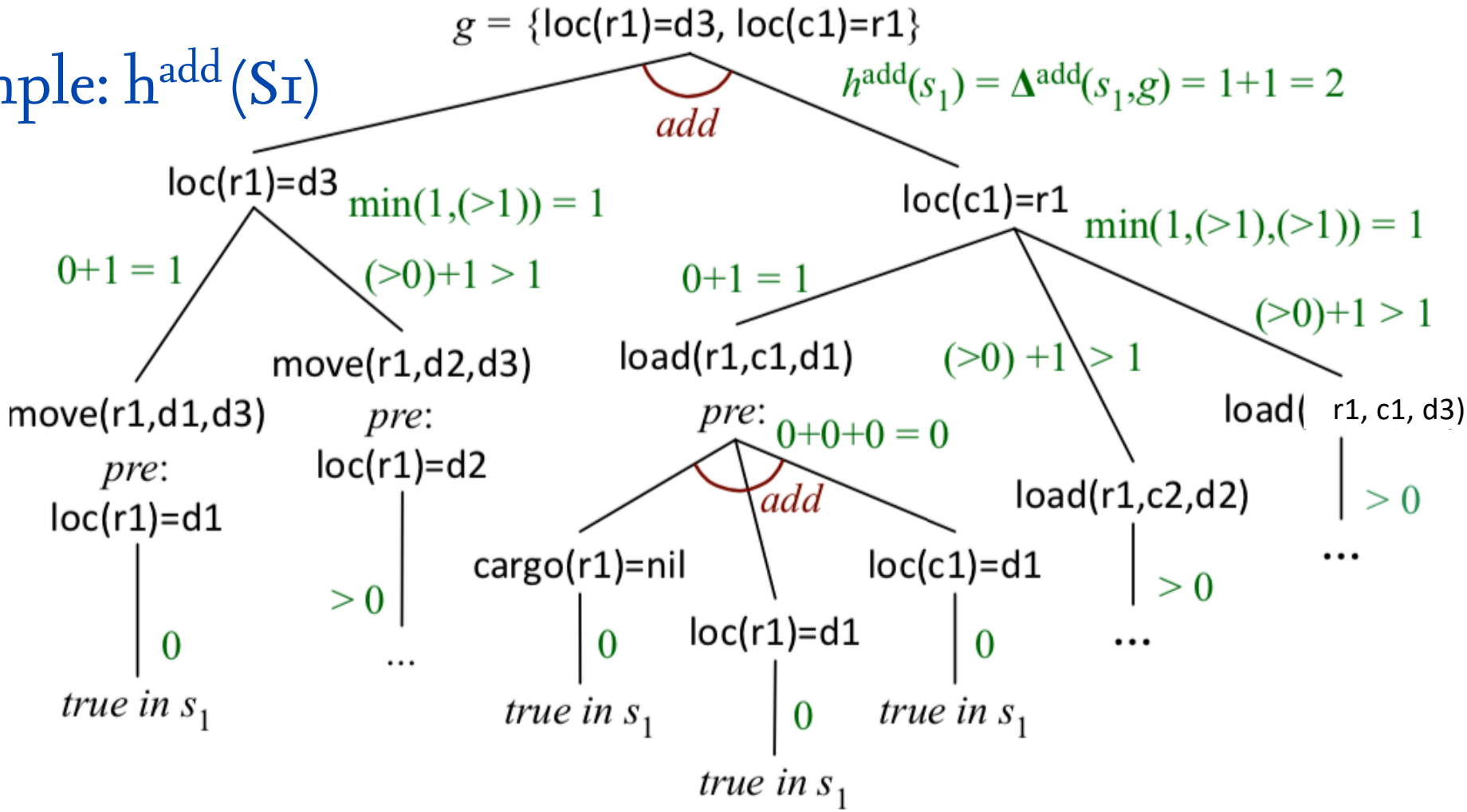


$$s_0 = \{\text{loc(r1)} = \text{d3}, \text{cargo(r1)} = \text{nil}, \text{loc(c1)} = \text{d1}\};$$
$$g = \{\text{loc(r1)} = \text{d3}, \text{loc(c1)} = \text{r1}\}.$$

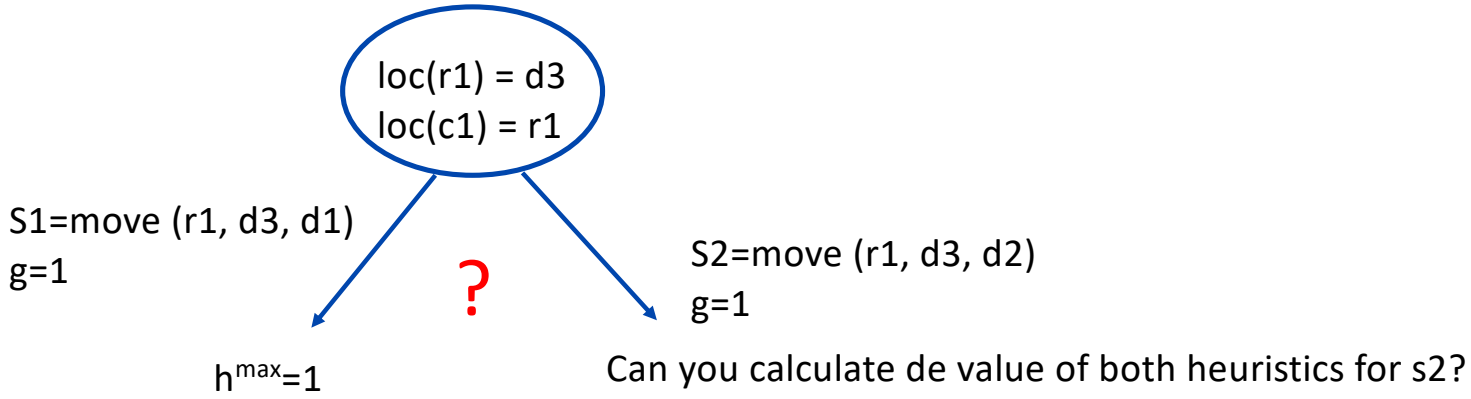
Example: $h^{max}(S_1)$



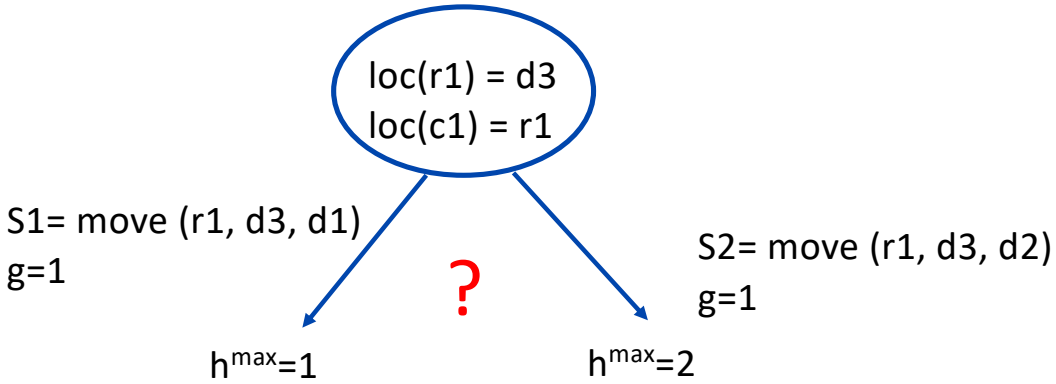
Example: $h^{add}(S_1)$



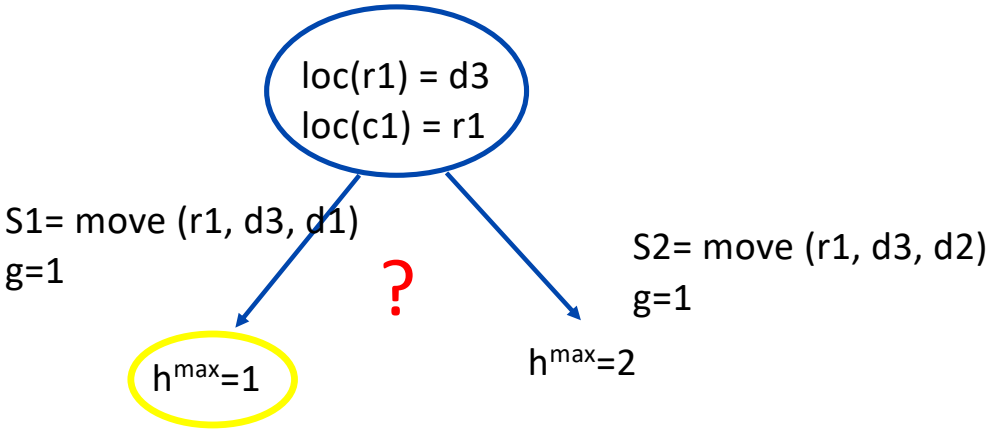
Search tree


loc(r1) = d3
loc(c1) = r1
S1=move (r1, d3, d1)
g=1
?
S2=move (r1, d3, d2)
g=1
h^{max}=1
Can you calculate de value of both heuristics for s2?

Search tree


loc(r1) = d3
loc(c1) = r1
S1= move (r1, d3, d1)
g=1
?
S2= move (r1, d3, d2)
g=1
h^max=1
h^max=2

Search tree


loc(r1) = d3
loc(c1) = r1
S1= move (r1, d3, d1)
g=1
?
S2= move (r1, d3, d2)
g=1
h^max=1
h^max=2

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

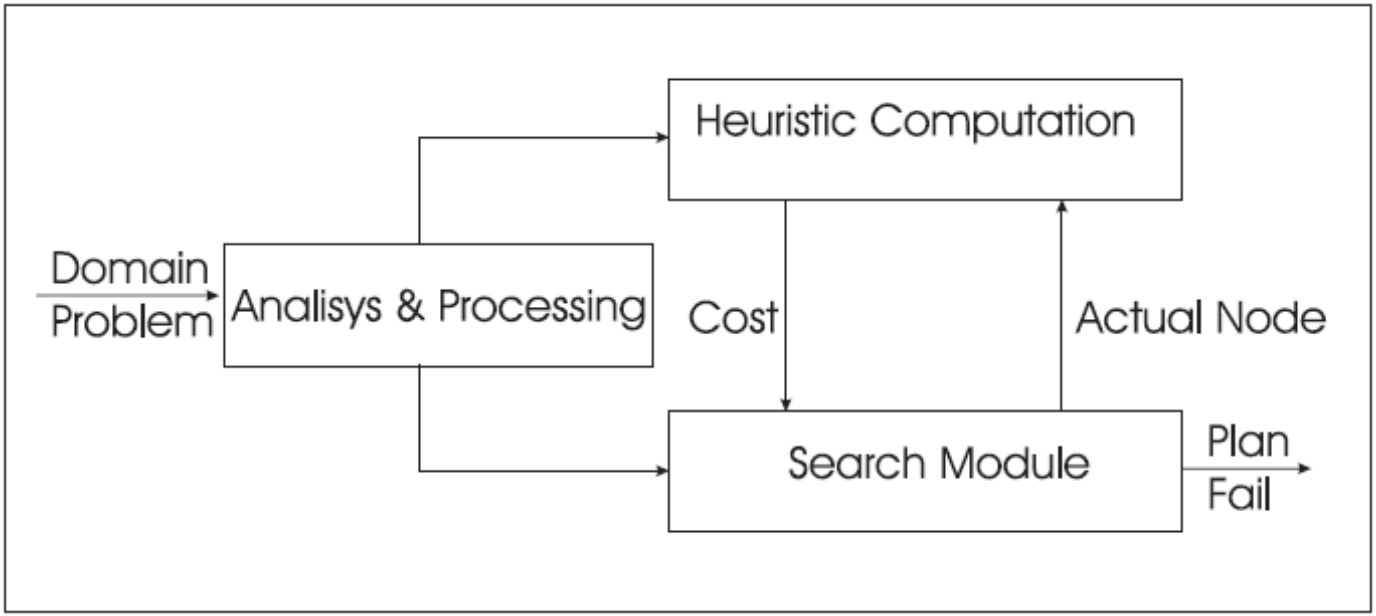
- To compute $h^{LM}(s1)$, we count the number of landmarks between s1 and g
- If we start in s1, then every solution plan must include a state in which cargo(r1) = c1
  - This is the only landmark for s1
  - $h^{LM}(s1) = 1$
- If we start in state s2, then the landmark computation will find two landmarks:
  - cargo(r1) = c1
  - loc(r1) = d1
  - $h^{LM}(s2) = 2$

Outline

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

FF (I)

- General architecture

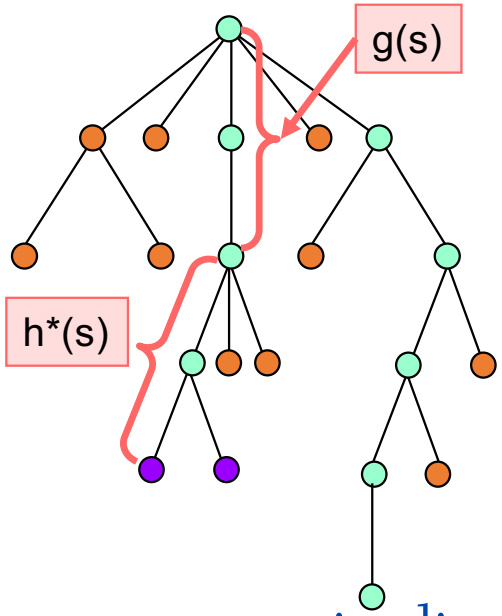


FF (II)

- **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

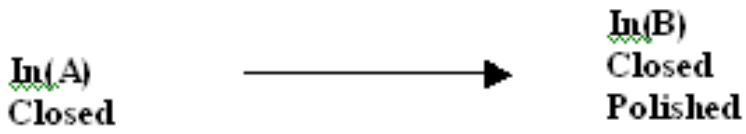


FF Heuristic

- Build GP until all goals are reached
  - First fact layer = S_0 and first action layer contains all actions that are applicable in S_0
  - The union of all **add** effects of these actions with the facts that are already there forms the second fact layer
  - To this layer, again all actions are applied, and so on, until a fact layer is reached that contains all goals
- When goals are reached, extract a relaxed plan:
  - Start at the top graph layer m, working on all goals
  - At each layer i, if a goal is present in layer $i-1$, then insert it into the goals to be achieved at $i-1$
  - Else, select action in layer $i-1$ that adds the goal, & insert the action's preconditions into the goals at $i-1$
  - Once all goals at i are worked on, continue with the goals at $i-1$
  - Stop when the first layer is reached
  - The process results in a relaxed plan $\langle O_0, ..., O_{m-1} \rangle$, where each O_i is the set of actions selected at time step i, with prevalence of NoOp over normal actions
  - Estimate solution length by counting the actions in that plan, NoOp counts as **0**

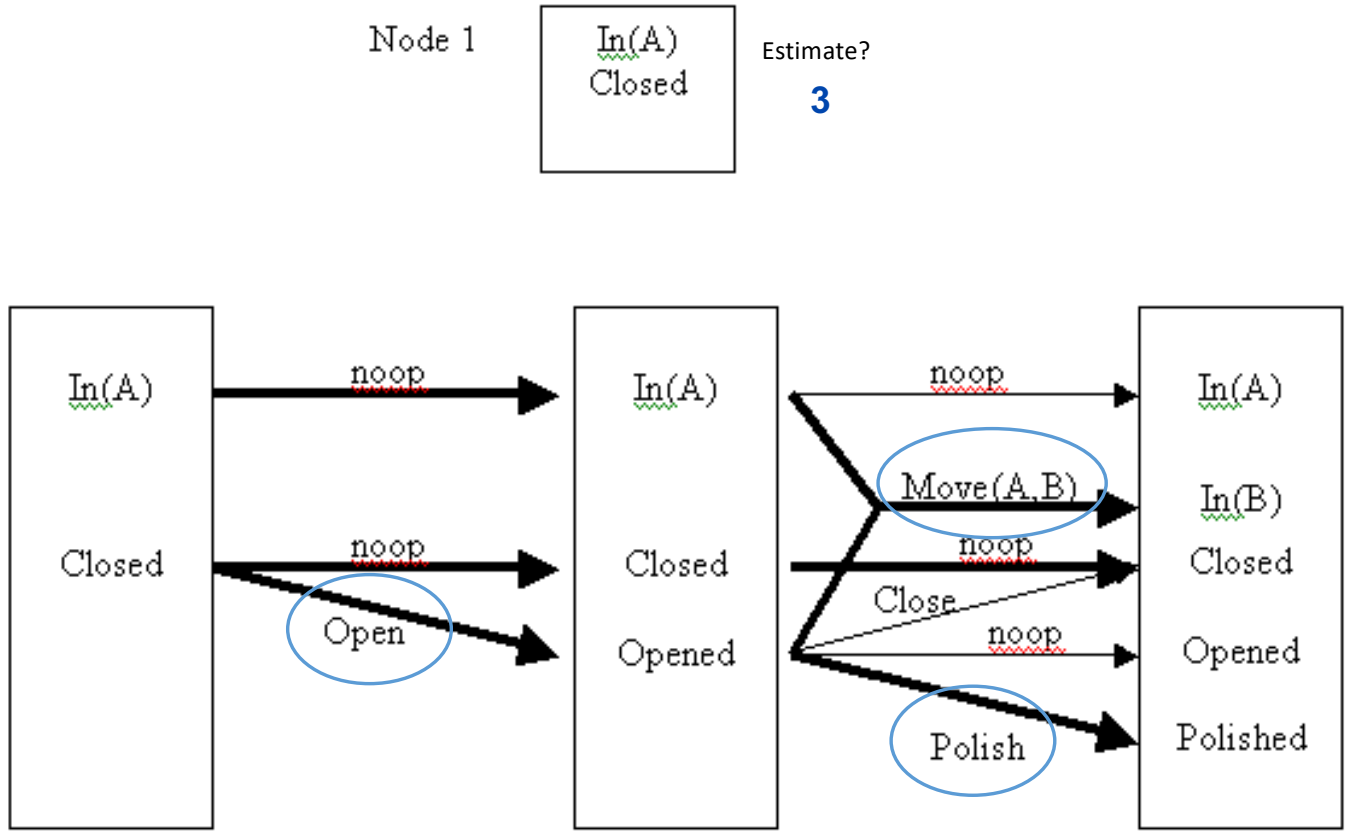
FF: h (I)

Move (x,y) **Pre:** in(x) opened **Add:** in(y) **Del:** in(x) **Open** **Pre:** closed **Add:** opened **Del:** closed	**Close** **Pre:** opened **Add:** closed **Del:** opened **Polish** **Pre:** opened **Add:** polished **Del:**

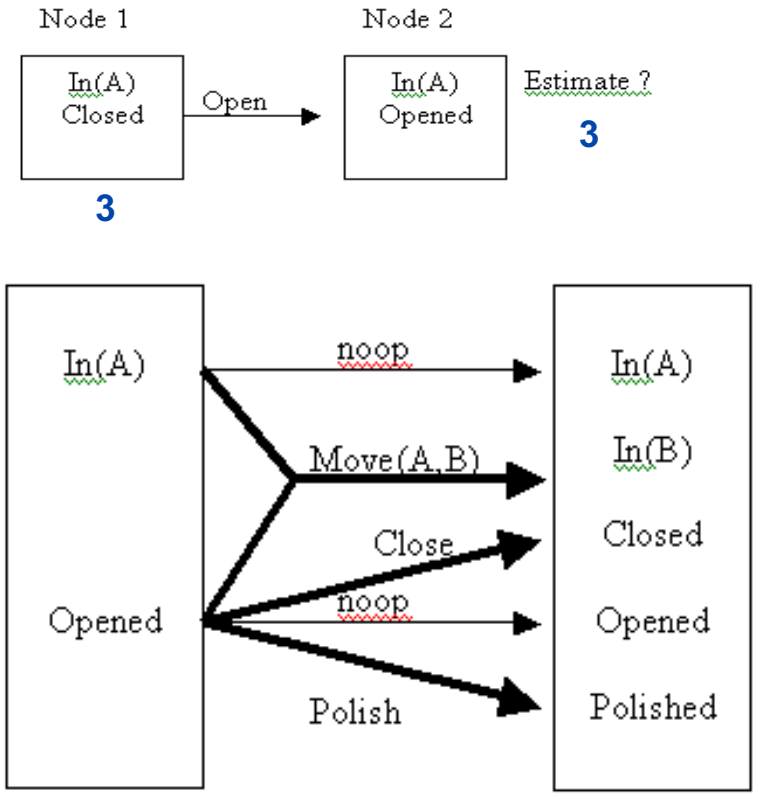


h_{ff}^{+}: number of operators in a relaxed GP

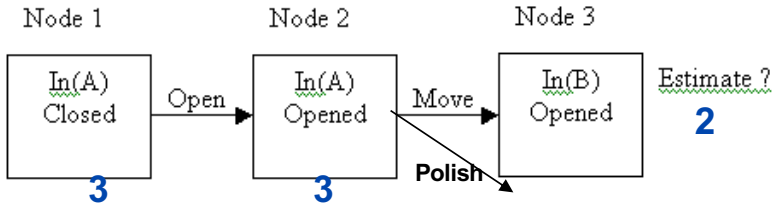
FF: h (II)

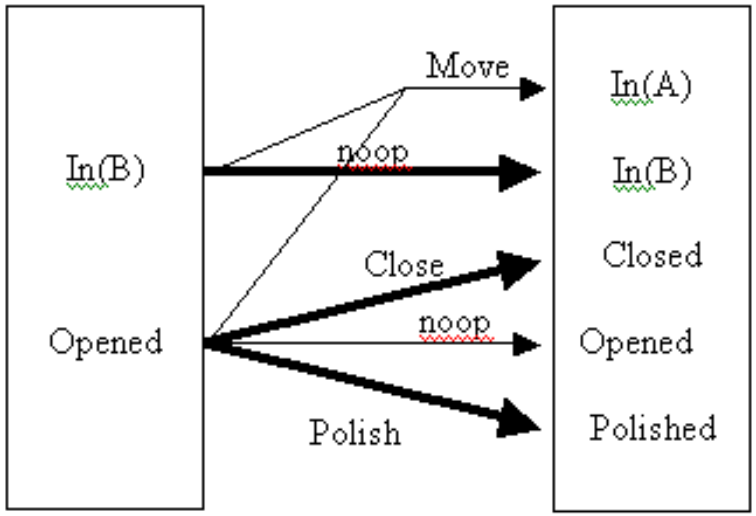

Node 1
In(A)
Closed
Estimate?
3
In(A)
Closed
noop
noop
Open
In(A)
Closed
Opened
noop
Move(A,B)
noop
Close
noop
Polish
In(A)
In(B)
Closed
Opened
Polished

FF: h (III)

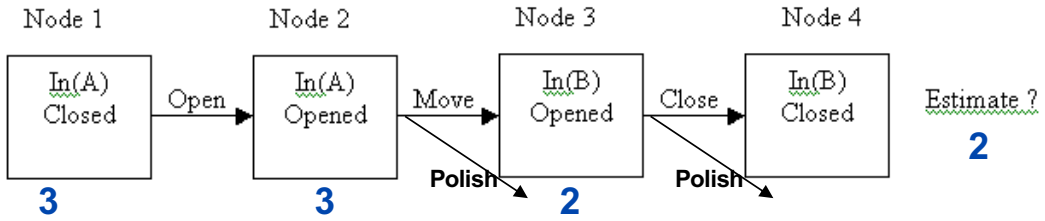

Node 1
Node 2
In(A)
Closed
Open
In(A)
Opened
Estimate ?
3
3
In(A)
Opened
noop
Move(A,B)
Close
noop
Polish
In(A)
In(B)
Closed
Opened
Polished

FF: h (IV)

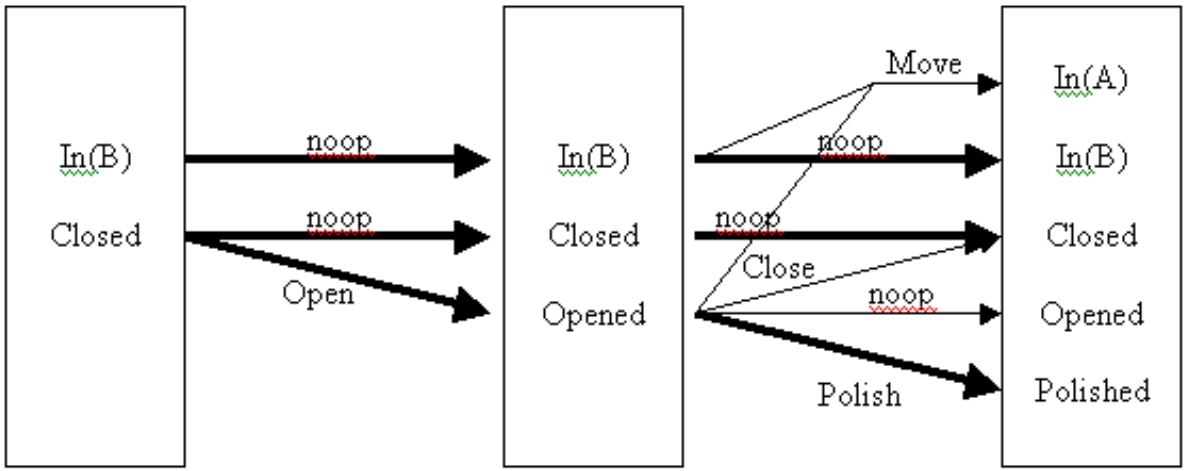

Node 1
Node 2
Node 3
In(A)
Closed
Open
In(A)
Opened
Move
In(B)
Opened
Estimate ?
2
Polish
3
3


In(B)
Opened
Move
noop
Close
noop
Polish
In(A)
In(B)
Closed
Opened
Polished

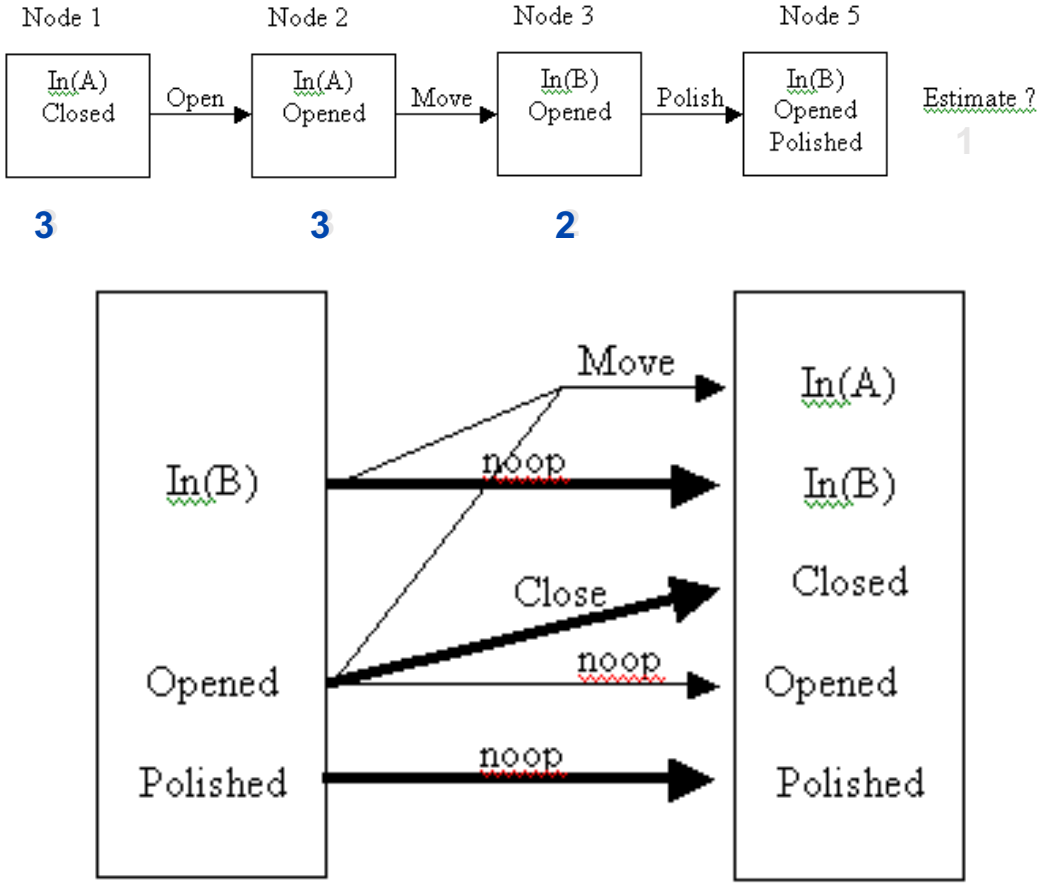
FF: h (V)



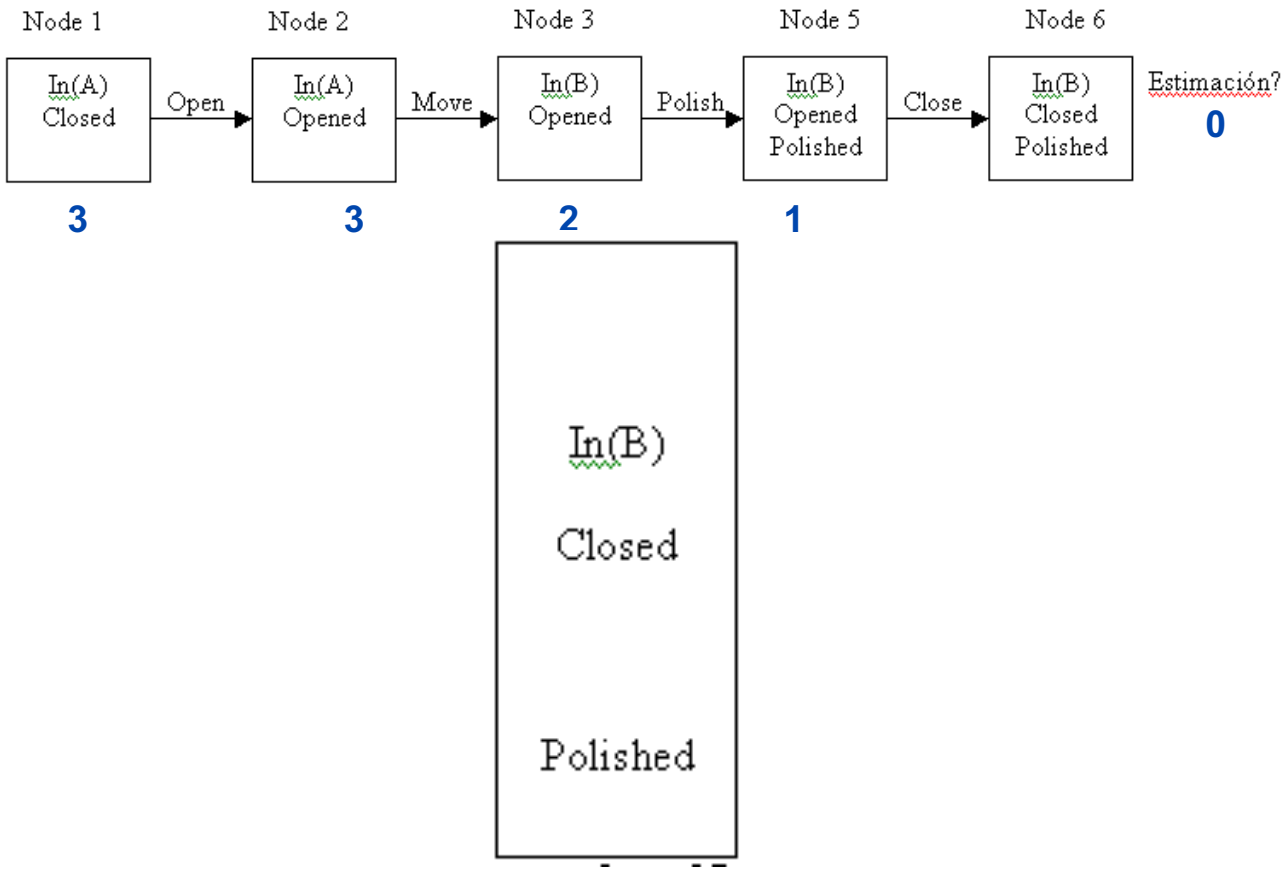
It does not improve the previous result, try Polish



FF: h (VI)


Node 1
In(A)
Closed
Open
Node 2
In(A)
Opened
Move
Node 3
In(B)
Opened
Polish
Node 5
In(B)
Opened
Polished
Estimate ?
1
3
3
2
In(B)
Opened
Polished
Move
noop
Close
noop
noop
In(A)
In(B)
Closed
Opened
Polished

FF: h (VII)


Node 1
In(A)
Closed
Open
Node 2
In(A)
Opened
Move
Node 3
In(B)
Opened
Polish
Node 5
In(B)
Opened
Polished
Close
Node 6
In(B)
Closed
Polished
Estimación?
0
3
3
2
1
In(B)
Closed
Polished

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 S1 move (r1, d3, d1) and S2 move (r1, d3, d2)

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

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

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

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