# Assignment #1-3 Consultations

## Exercises on heuristic computation

Michaela Urbanovská

PUI Tutorial
Week 8

# Assignment #1-3 Consultations

# Relaxation heuristics

$F = \{a, b, c, d, e, f\}$
$s_{init} = \{a, b\}$
$s_{goal} = \{d, f\}$

$O =$

| | pre | add | del | c |
|---|---|---|---|---|
| $o_1$ | a,b | b | a | 1 |
| $o_2$ | b,c | a | b | 1 |
| $o_3$ | c | e | $\emptyset$ | 2 |
| $o_4$ | c,e | d | e | 2 |
| $o_5$ | a,c,d | f | a | 1 |
| $o_6$ | a,b | c | $\emptyset$ | 1 |

Compute

- $h^{add}(s_{init})$
- $h^{max}(s_{init})$

# $h^{max}$

**Algorithm 1:** Algorithm for computing $\mathrm{h}^{\max}(s)$.

**Input:** $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$, state $s$
**Output:** $\mathrm{h}^{\max}(s)$

**for each** $f \in s$ **do** $\Delta_1(s, f) \leftarrow 0$;
**for each** $f \in \mathcal{F} \setminus s$ **do** $\Delta_1(s, f) \leftarrow \infty$;
**for each** $o \in \mathcal{O}, pre(o) = \emptyset$ **do**
  **for each** $f \in add(o)$ **do** $\Delta_1(s, f) \leftarrow \min\{\Delta_1(s, f), \mathrm{c}(o)\}$ ;
**end**
**for each** $o \in \mathcal{O}$ **do** $U(o) \leftarrow |\mathrm{pre}(o)|$;
$C \leftarrow \emptyset$;
**while** $s_{goal} \not\subseteq C$ **do**
  $k \leftarrow \arg\min_{f \in \mathcal{F} \setminus C} \Delta_1(s, f)$;
  $C \leftarrow C \cup \{k\}$;
  **for each** $o \in \mathcal{O}, k \in pre(o)$ **do**
    $U(o) \leftarrow U(o) - 1$;
    **if** $U(o) = 0$ **then**
      **for each** $f \in add(o)$ **do**
        $\Delta_1(s, f) \leftarrow \min\{\Delta_1(s, f), \mathrm{c}(o) + \Delta_1(s, k)\}$;
      **end**
    **end**
  **end**
**end**
$\mathrm{h}^{\max}(s) = \max_{f \in s_{goal}} \Delta_1(s, f)$;

# Relaxation heuristics

$F = \{a, b, c, d, e, f\}$
$s_{init} = \{a, f\}$
$s_{goal} = \{d\}$

$O =$

| | pre | add | del | c |
|---|---|---|---|---|
| $o_1$ | a,f | b | f | 1 |
| $o_2$ | c | a,d | c | 1 |
| $o_3$ | a,b | c,e | a | 1 |
| $o_4$ | c,b | d | c,b | 2 |
| $o_5$ | a | e,f | $\emptyset$ | 3 |

Compute

- $h^{LM-Cut}(s_{init})$
- $h^{FF}(s_{init})$

# $h^{LM-Cut}$

**Algorithm 2:** Algorithm for computing $\mathrm{h}^{\text{lm-cut}}(s)$.

**Input:** $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$, state $s$
**Output:** $\mathrm{h}^{\text{lm-cut}}(s)$

```
if h^max(Π, s_init) = ∞ then
    h^lm-cut(s) ← ∞ and terminate;
end
h^lm-cut(s) ← 0;
Π_1 = ⟨F' = F ∪ {I, G}, O' = O ∪ {o_init, o_goal}, s'_init = {I}, s'_goal = {G}, c_1⟩, where
    pre(o_init) = {I}, add(o_init) = s, del(o_init) = ∅, pre(o_goal) = s_goal, add(o_goal) = {G},
    del(o_goal) = ∅, c_1(o_init) = 0, c_1(o_goal) = 0, and c_1(o) = c(o) for all o ∈ O;
i ← 1;
while h^max(Π_i, s'_init) ≠ 0 do
    Construct a justification graph G_i from Π_i;
    Construct an s-t-cut C_i(G_i, n_I, n_G) = (N_i^0, N_i^* ∪ N_i^b);
    Create a landmark L_i as a set of labels of edges that cross the cut C_i, i.e., they
        lead from N_i^0 to N_i^*;
    m_i ← min_{o ∈ L_i} c_i(o);
    h^lm-cut(s) ← h^lm-cut(s) + m_i;
    Set Π_{i+1} = ⟨F', O', s'_init, s'_goal, c_{i+1}⟩, where c_{i+1}(o) = c_i(o) − m_i if o ∈ L_i, and
        c_{i+1}(o) = c_i(o) otherwise;
    i ← i + 1;
end
```

# $h^{FF}$

Overall algorithm:

- Create reachability graph
- Mark the final G node
- Apply rules layers by layer until every marked node is justified

Justified node definitions:

- Action node is justified if all precondition fact nodes are marked
- Fact node is justified if at least one predecessor node is marked

- Starting with marked goal node, apply the following rules **layer by layer** until **all marked nodes are justified**

1) Mark all immediate predecessors of a marked unjustified action node
2) Mark the immediate predecessor of a marked unjustified atom node with only one immediate predecessor
3) Mark an immediate predecessor of a marked unjustified atom node connected via an idle arc (to the same atom in the previous layer)
4) Mark any immediate predecessor of a marked unjustified atom node

# $h^{flow}$ and $h^{pot}$

$V = \{v_1, v_2, v_3\}$
$D_{v_1} = \{A, B\}, D_{v_2} = \{C, D\}, D_{v_3} = \{E, F\}$
$s_{init} = \{v_1 = A, v_2 = C, v_3 = E\}$
$s_{goal} = \{v_3 = F\}$

$O =$

| | pre | eff | c |
|---|---|---|---|
| $o_1$ | $v_1 = A$ | $v_1 = B$ | 3 |
| $o_2$ | $v_1 = B, v_2 = C$ | $v_1 = A, v_2 = D$ | 1 |
| $o_3$ | $v_1 = A, v_2 = D$ | $v_3 = F$ | 1 |
| $o_4$ | $v_2 = D, v_3 = E$ | $v_2 = C$ | 2 |

Compute

- $h^{flow}$
- $h^{pot}$
- $h^{M\&S}$

## LP formulation

$$\text{minimize} \sum_{o \in O} c(o) x_o$$

$$\text{subject to } LB_{V,v} \leq \sum_{o \in prod(\langle V,v \rangle)} x_o - \sum_{o \in cons(\langle V,v \rangle)} x_o, \forall V \in \mathbf{V}, \forall v \in D_V$$

$$\text{where } LB_{V,v} = \begin{cases} 0 & \text{if } V \in vars(s_{goal}) \text{ and } s_{goal}[V] = v \text{ and } s[V] = v, \\ 1 & \text{if } V \in vars(s_{goal}) \text{ and } s_{goal}[V] = v \text{ and } s[V] \neq v, \\ -1 & \text{if } (V \notin vars(s_{goal}) \text{ or } s_{goal}[V] \neq v) \text{ and } s[V] = v, \\ 0 & \text{if } (V \notin vars(s_{goal}) \text{ or } s_{goal}[V] \neq v) \text{ and } s[V] \neq v, \end{cases}$$

## LP formulation

$$\text{maximize} \sum_{V \in \mathbf{V}} P_{V, s_{init}[V]}$$

$$\text{subject to } P_{V,v} \leq M_V, \forall V \in \mathbf{V}, \forall v \in D_V$$

$$\sum_{V \in \mathbf{V}} maxpot(V, s_{goal}) \leq 0$$

$$\sum_{V \in vars(eff(o))} (maxpot(V, pre(o)) - P_{V, eff(o)[V]}) \leq c(o), \forall o \in O$$

$$\text{where } maxpot(V, p) = \begin{cases} P_{V,p[V]} & \text{if } V \in vars(p), \\ M_V & otherwise. \end{cases}$$

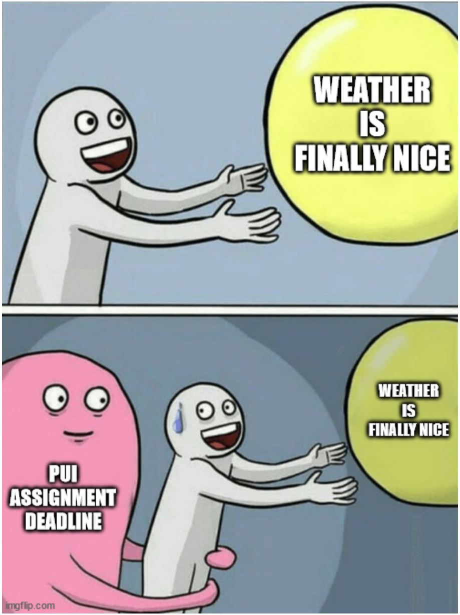
The **value of $h^{pot}$ heuristic** for the state $s$ is

$$h^{pot}(s) = \begin{cases} \sum_{V \in \mathbf{V}} P_{V,s[V]} & \text{if the solution is } \textbf{feasible} \\ \infty & \text{if the solution is } \textbf{not feasible} \end{cases}$$

# Merge & Shrink

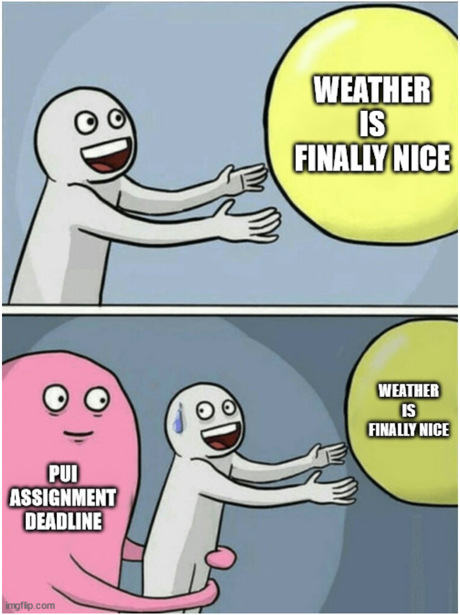
1. Create atomic projections (one per variable)
2. Merge two arbitrary transition systems (synchronized product)
3. Shrink the new transition graph (merge states together to create smaller abstraction)
4. Repeat 2 and 3 until you're left with one abstraction in which you can find the solution

# The End




not Feedback form link

# The End




Feedback form link