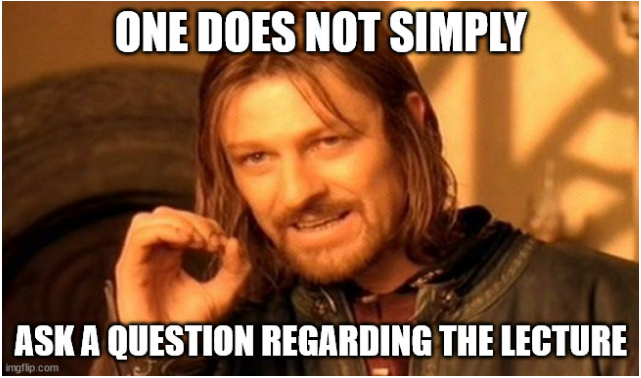
# LM-Cut and $h^{FF}$

Michaela Urbanovská

PUI Tutorial
Week 6

# Lecture check

- Any questions regarding the lecture?



# LM-Cut

## LM-Cut Heuristic

- Relaxation heuristic
- Uses disjunctive operator landmarks
- Admissible (and actually very successful) heuristic

Things we need to know to compute it

- Disjunctive operator landmark
- Operator supporter
- Justification graph
- s-t cut

# LM-Cut

## Disjunctive operator landmark

Disjunctive operator landmark $L \subseteq O$ is set of operators such that every plan $\pi$ contains at least one operator $o \in L$.

## Example

Suppose we have problem with 2 existing plans:

- $\pi_1 = (o_1, o_2, o_3, o_4)$
- $\pi_2 = (o_1, o_5, o_2, o_6)$

Which of the following sets are disjunctive operator landmarks?

1. $\{o_1\}$
2. $\{o_1, o_3\}$
3. $\{o_2, o_3\}$

$\{o_1, o_2, o_3, o_4\}$

$\{o_3, o_4\}$

$\{o_4, o_6\}$

# LM-Cut

## Disjunctive operator landmark

Disjunctive operator landmark $L \subseteq O$ is set of operators such that every plan $\pi$ contains at least one operator $o \in L$.

## Example

Suppose we have problem with 2 existing plans:

- $\pi_1 = (o_1, o_2, o_3, o_4)$
- $\pi_2 = (o_1, o_5, o_2, o_6)$

Which of the following sets are disjunctive operator landmarks?

- $\{o_1\}$
- $\{o_1, o_3\}$
- $\{o_2, o_3\}$
- $\{o_1, o_2, o_3, o_4\}$
- $\{o_3, o_4\}$
- $\{o_4, o_6\}$

# LM-Cut

## Delta function

- Function $\Delta_1$ from previous tutorial ($h^{max}$)
- $\Delta_1(s, f) =$

$$\begin{cases} 0 & \text{if } f \in s, \\ \inf & \text{if } \forall o \in O : f \notin add(o), \\ min\{c(o) + \Delta_1(s, o) | o \in O, f \in add(o)\} & otherwise. \end{cases}$$

## Supporter of an operator

Supporter is a **fact**.
Function $supp(o) = argmax_{f \in pre(o)} \Delta_1(s, f)$ maps each operator $o \in O$ to its **supporter**.
($s$ denotes the state where we compute the heuristic estimate)

# LM-Cut

## Justification graph

$G = (N, E)$ is a **directed labeled** multigraph.

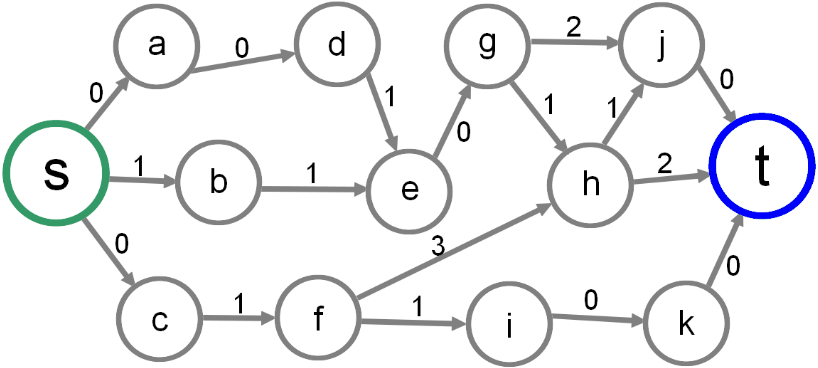
- $N = \{n_f | f \in F\}$ *(set of nodes)*
- $E = \{(n_s, n_t, o) | o \in O, s = supp(o), t \in add(o)\}$ *(set of edges)*
- Edge $e = (a, b, l)$ denotes edge from $a$ to $b$ with label $l$

# LM-Cut

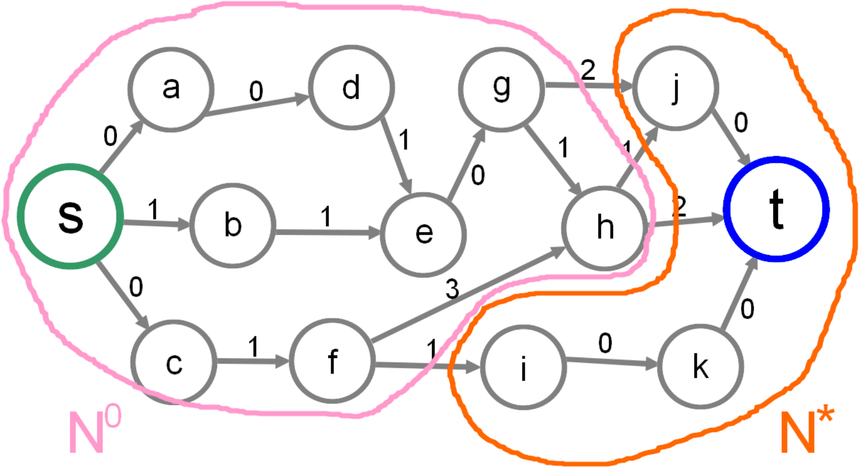
## s-t cut

- **s-t cut** $C(G, s, t) = (N^0, N^* \cup N^b)$
- partitioning of nodes from the **justification graph** $G = (N, E)$
- $N^*$ contains nodes from which $t$ can be reached with zero-cost path
- $N^0$ contains nodes which can be reached from $s$ without passing any node from $N^*$
- $N^b = N \setminus (N^0 \cup N^*)$ *(all the other nodes)*
- landmark $L$ corresponds to edges that cross the cut $C$ $\rightarrow$ lead from $N^0$ to $N^*$

# LM-Cut



# LM-Cut



# Example

Compute $h^{LM-Cut}(s_{init})$ for problem $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$:

$F = \{s, q_1, q_2, q_3, t\}$

$s_{init} = \{s\}$

$s_{goal} = \{t\}$

$O =$

| | pre | add | del | c |
|---|---|---|---|---|
| $o_1$ | s | $q_1, q_2$ | $\emptyset$ | 1 |
| $o_2$ | s | $q_1, q_3$ | $\emptyset$ | 1 |
| $o_3$ | s | $q_2, q_3$ | $\emptyset$ | 1 |
| $fin$ | $q_1, q_2, q_3$ | t | $\emptyset$ | 0 |

# LM-Cut

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

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

**if** $h^{\max}(\Pi, s_{init}) = \infty$ **then**
  $h^{\text{lm-cut}}(s) \leftarrow \infty$ and terminate;
**end**
$h^{\text{lm-cut}}(s) \leftarrow 0$;
$\Pi_1 = \langle \mathcal{F}' = \mathcal{F} \cup \{I, G\}, \mathcal{O}' = \mathcal{O} \cup \{o_{init}, o_{goal}\}, s'_{init} = \{I\}, s'_{goal} = \{G\}, c_1 \rangle$, where $\text{pre}(o_{init}) = \{I\}$, $\text{add}(o_{init}) = s$, $\text{del}(o_{init}) = \emptyset$, $\text{pre}(o_{goal}) = s_{goal}$, $\text{add}(o_{goal}) = \{G\}$, $\text{del}(o_{goal}) = \emptyset$, $c_1(o_{init}) = 0$, $c_1(o_{goal}) = 0$, and $c_1(o) = c(o)$ for all $o \in \mathcal{O}$;
$i \leftarrow 1$;
**while** $h^{\max}(\Pi_i, s'_{init}) \neq 0$ **do**
  Construct a justification graph $G_i$ from $\Pi_i$;
  Construct an s-t-cut $\mathcal{C}_i(G_i, n_I, n_G) = (N_i^0, N_i^\star \cup N_i^b)$;
  Create a landmark $L_i$ as a set of labels of edges that cross the cut $\mathcal{C}_i$, i.e., they lead from $N_i^0$ to $N_i^\star$;
  $m_i \leftarrow \min_{o \in L_i} c_i(o)$;
  $h^{\text{lm-cut}}(s) \leftarrow h^{\text{lm-cut}}(s) + m_i$;
  Set $\Pi_{i+1} = \langle \mathcal{F}', \mathcal{O}', s'_{init}, s'_{goal}, c_{i+1} \rangle$, where $c_{i+1}(o) = c_i(o) - m_i$ if $o \in L_i$, and $c_{i+1}(o) = c_i(o)$ otherwise;
  $i \leftarrow i + 1$;
**end**

# Example 2

Compute $h^{LM-Cut}(s_{init})$ for problem $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$:

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

$s_{init} = \{c, d\}$

$s_{goal} = \{a, e\}$

$O =$

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

# LM-Cut

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

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

```
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
```

# Recap

- You should be able to compute and implement LM-Cut
- You should know how to compute $h^{FF}$ heuristic with reachability graph

# The End


Feedback form link