Principles of AI Planning

12. The LM-cut heuristic

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Bernhard Nebel and Robert Mattmüller

January 8th, 2014

1 The LM-cut heuristic

- Motivation
- Definitions
- Finding and exploiting landmarks
- Admissibility
- Summary

Motivation

- RPG-based relaxation heuristics seen so far,
  - either admissible, but not very informative (h_{max}),
  - or quite informative, but not admissible (h_{add}, h_{sa}, h_{FF}).
- $\rightsquigarrow$ no useful relaxation heuristic for optimal planning yet.
- This chapter: informative admissible relaxation heuristic ($h_{\text{LM-cut}}$).

Motivation

Combination of several ideas:

- Delete relaxation
- Landmarks
- Cost partitioning

Motivation

- Technique for derivation of heuristics: landmarks.
- We assume delete-free planning tasks Π^+ and apply landmark heuristics to Π^+.
- Aim: approximate optimal delete relaxation heuristic h^+ as precisely as possible.

Assumptions and Definitions

- Assumption: STRIPS tasks with action costs 0 or 1.
- Zero-cost actions:
  - dummy action $o_{\mathbf{s}}$ constructing initial state from unique initial proposition **s**
  - dummy action $o_{\mathbf{t}}$ construction unique dummy goal proposition **t** from actual goal propositions
  - actions o already accounted for in the heuristic computation
- In the algorithm we will present, action cost values will be iteratively decremented.
  - In first iteration, costs $c_1(o_{\mathbf{s}}) = c_1(o_{\mathbf{t}}) = 0$, and $c_1(a) = 1$ for all other actions.
  - Cost functions in later iterations i are denoted by c_i.

Landmarks

Definition (Landmark)

A landmark of a planning task Π is a set of actions L such that each plan for Π contains at least one action from L.

$\rightsquigarrow$ if there are n disjoint landmarks for a planning task Π with initial state I, then $h(I) = n$ is an admissible estimate.

- landmarks in this sense are also called disjunctive action landmarks.

Landmarks

Example

$\langle A, I, \{o_1, o_2, o_3, o_4, o_5\}, \gamma\rangle$ with

$$
\begin{array}{ll}
A = \{a,b,c,d,e,f,g\} & I = \{a \mapsto 1\} \cup \{x \mapsto 0 \mid x \neq a\} \\
o_1 = \langle a, b \wedge c\rangle & o_2 = \langle a, c \wedge d\rangle \\
o_3 = \langle a, d \wedge e\rangle & o_4 = \langle a, e \wedge b\rangle \\
o_5 = \langle a, f\rangle & o_6 = \langle b \wedge c \wedge d \wedge e \wedge f, g\rangle \\
\gamma = g &
\end{array}
$$

(Minimal) landmarks:

$$
\begin{array}{ll}
\{o_1, o_2\} \text{ (because of } c\text{)}, & \{o_2, o_3\} \text{ (because of } d\text{)}, \\
\{o_3, o_4\} \text{ (because of } e\text{)}, & \{o_4, o_1\} \text{ (because of } b\text{)}, \\
\{o_5\} \text{ (because of } f\text{)}, & \{o_6\} \text{ (because of } g\text{)}
\end{array}
$$

Landmarks

Example (ctd.)

But at most four disjoint landmarks, e.g., $\{o_1, o_2\}, \{o_3, o_4\}, \{o_5\}, \{o_6\}$.

$\rightsquigarrow h_{\mathrm{LM}}(I) = 4$ is admissible.

Motivation

Idea for algorithm:
Iteratively compute disjoint disjunctive action landmarks.

How?

- Compute one landmark L_1.
- Compute another landmark L_2 that is disjoint from L_1.
- Compute another landmark L_3 that is disjoint from L_1 and L_2.
- ...
- Stop when no more such landmarks exist.

Assumptions and Definitions

Definition (Precondition-choice function)

A precondition-choice function (pcf) is a function D that maps each action into one of its preconditions.
(We assume that each action has at least one precondition.)

Definition (Justification graph)

The justification graph for a pcf D, denoted by $G(D)$, is a directed graph whose vertices are the propositions and which has an edge (p,q) labeled with a iff the action a adds q and $D(a) = p$.

Assumptions and Definitions

Definition (Cut)

For two nodes **s** and **t** in a justification graph, an **s**-**t** cut in that justification graph is a subset C of its edges such that all paths from **s** to **t** use an edge from C.

When **s** and **t** are clear, we simply call C a cut.

Theorem (Cuts correspond to landmarks)

Let C be a cut in a justification graph for an arbitrary pcf. Then the edge labels for C are a landmark. □

Assumptions and Definitions

Definition ($h_{\max}$ values of atoms)

Given a fixed initial state s and an action cost function c, the $h_{\max}$ value of an atom a, denoted by $h^c_{\max}(a)$, is the value the RPG proposition node for atom a in the last RPG layer is labeled with after the RPG computation (with layer 0 initialized with state s and action costs given by c) has converged/stabilized.

LM-cut Heuristic: Motivation

- In general exponentially many pcfs, i.e., we cannot compute all relevant landmarks.
- The LM-cut heuristic is a method to compute pcfs and cuts in a goal-directed way.
- Efficient partitioning of actions into cuts.

$\rightsquigarrow$ currently best admissible planning heuristic

Pseudocode of LM-cut heuristic

Initialize $h = 0$ and $i = 1$.

Step 1. Compute $h^{c_i}_{\max}(a)$ values for every atom $a \in A$. Terminate if $h^{c_i}_{\max}(\mathbf{t}) = 0$.

Step 2. Compute pcf D_i: Modify actions by keeping only one proposition in the precondition of each action: a proposition maximizing $h^{c_i}_{\max}$, breaking ties arbitrarily.

Step 3. Construct justification graph G_i of D_i: Vertices are the propositions; for each action $a = \langle p, q_1 \wedge \ldots \wedge q_k \rangle$ and each $j = 1, \ldots, k$, there is an edge from p to q_j with cost $c_i(a)$ and label a.

Step 4. ...

Pseudocode of LM-cut heuristic (ctd.)

Step 4. Construct an **s**-**t**-cut $C_i = (V^0_i, V^*_i \cup V^b_i)$ of G_i as follows: V^*_i contains all propositions from which **t** can be reached through a zero-cost path, V^0_i contains all propositions reachable from **s** without passing through some propositions in V^*_i, and V^b_i contains all remaining propositions. Clearly, $\mathbf{s} \in V^0_i$ and $\mathbf{t} \in V^*_i$.

Step 5. Determine disjunctive action landmark: Let L_i be the set of labels of the edges that cross the cut C_i (i.e., lead from V^0_i to V^*_i).

Step 6. Decrease action costs: Define $c_{i+1}(a) := c_i(a)$ if $a \notin L_i$, and $c_{i+1}(a) := 0$ if $a \in L_i$.

Step 7. Increase heuristic value: $h := h + 1$.

Step 8. Set $i := i + 1$ and go to Step 1.

Example

Adaptation/simplification of running example from Chapter 8: planning task $\langle A, I, \{o_\mathbf{s}, o_1, o_2, o_3, o_4, o_\mathbf{t}\}, \gamma\rangle$ with

$$A = \{\mathbf{s}, a, b, c, d, e, f, g, h, \mathbf{t}\}$$
$$I = \{\mathbf{s} \mapsto 1, a \mapsto 0, b \mapsto 0, c \mapsto 0, d \mapsto 0, e \mapsto 0, f \mapsto 0, g \mapsto 0, h \mapsto 0, \mathbf{t} \mapsto 0\}$$
$$o_\mathbf{s} = \langle \mathbf{s}, a \wedge c \wedge d\rangle$$
$$o_1 = \langle c \wedge d, b\rangle$$
$$o_2 = \langle a \wedge b, e\rangle$$
$$o_3 = \langle a, f\rangle$$
$$o_4 = \langle f, g \wedge h\rangle$$
$$o_\mathbf{t} = \langle e \wedge g \wedge h, \mathbf{t}\rangle$$
$$\gamma = \mathbf{t}$$

Example

- Cheapest sequential (relaxed) plan: $\langle o_\mathbf{s}, o_1, o_2, o_3, o_4, o_\mathbf{t}\rangle$ with cost $h^+(I) = 4$ (recall that $o_\mathbf{s}$ and $o_\mathbf{t}$ cost nothing).
- Parallel (relaxed) plan witnessing $h_{\max}(I) = 2$: $\langle \{o_\mathbf{s}\}, \{o_1, o_3\}, \{o_2, o_4\}, \{o_\mathbf{t}\}\rangle$.

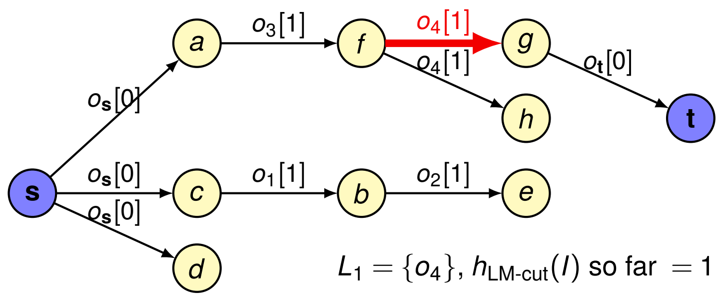
Our aim: Get closer to $h^+(I) = 4$ using $h_{\text{LM-cut}}$ than using $h_{\max}$.

Example: Iteration 1

prop p	$\mathbf{s}$	a	b	c	d	e	f	g	h	$\mathbf{t}$
$h^{c_1}_{\max}(p)$	0	0	1	0	0	2	1	2	2	2

action o	$o_\mathbf{s}$	o_1	o_2	o_3	o_4	$o_\mathbf{t}$
pcf $D_1(o)$	$\mathbf{s}$	c	b	a	f	g

$o_\mathbf{s}[0] = \langle \mathbf{s}, a \wedge c \wedge d\rangle$
$o_1[1] = \langle c \wedge d, b\rangle$
$o_2[1] = \langle a \wedge b, e\rangle$
$o_3[1] = \langle a, f\rangle$
$o_4[1] = \langle f, g \wedge h\rangle$
$o_\mathbf{t}[0] = \langle e \wedge g \wedge h, \mathbf{t}\rangle$



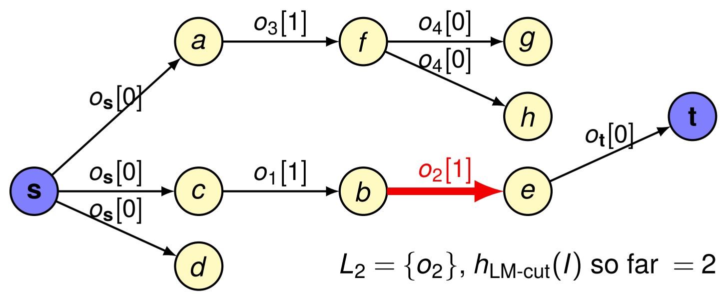
$L_1 = \{o_4\}$, $h_{\text{LM-cut}}(I)$ so far $= 1$

Example: Iteration 2

prop p	$\mathbf{s}$	a	b	c	d	e	f	g	h	$\mathbf{t}$
$h^{c_2}_{\max}(p)$	0	0	1	0	0	2	1	1	1	2

action o	$o_\mathbf{s}$	o_1	o_2	o_3	o_4	$o_\mathbf{t}$
pcf $D_2(o)$	$\mathbf{s}$	c	b	a	f	e

$o_\mathbf{s}[0] = \langle \mathbf{s}, a \wedge c \wedge d\rangle$
$o_1[1] = \langle c \wedge d, b\rangle$
$o_2[1] = \langle a \wedge b, e\rangle$
$o_3[1] = \langle a, f\rangle$
$o_4[0] = \langle f, g \wedge h\rangle$
$o_\mathbf{t}[0] = \langle e \wedge g \wedge h, \mathbf{t}\rangle$



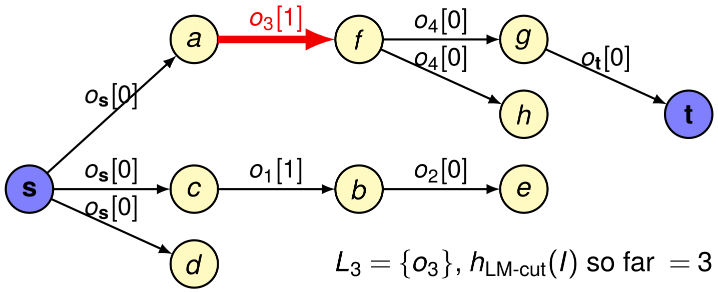
$L_2 = \{o_2\}$, $h_{\text{LM-cut}}(I)$ so far $= 2$

Example: Iteration 3

prop p	$\mathbf{s}$	a	b	c	d	e	f	g	h	$\mathbf{t}$
$h^{c_3}_{\max}(p)$	0	0	1	0	0	1	1	1	1	1

action o	$o_\mathbf{s}$	o_1	o_2	o_3	o_4	$o_\mathbf{t}$
pcf $D_3(o)$	$\mathbf{s}$	c	b	a	f	g

$o_\mathbf{s}[0] = \langle \mathbf{s}, a \wedge c \wedge d \rangle$
$o_1[1] = \langle c \wedge d, b \rangle$
$o_2[0] = \langle a \wedge b, e \rangle$
$o_3[1] = \langle a, f \rangle$
$o_4[0] = \langle f, g \wedge h \rangle$
$o_\mathbf{t}[0] = \langle e \wedge g \wedge h, \mathbf{t} \rangle$



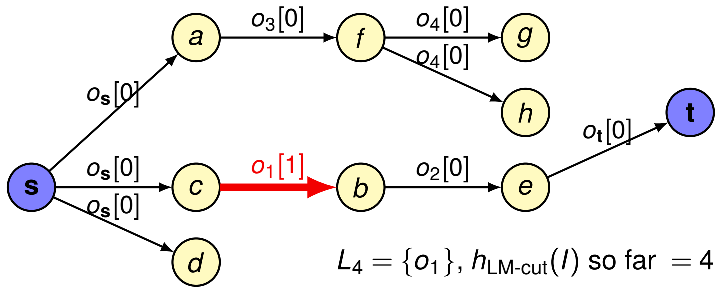
$L_3 = \{o_3\}$, $h_{\text{LM-cut}}(I)$ so far $= 3$

Example: Iteration 4

prop p	$\mathbf{s}$	a	b	c	d	e	f	g	h	$\mathbf{t}$
$h^{c_4}_{\max}(p)$	0	0	1	0	0	1	0	0	0	1

action o	$o_\mathbf{s}$	o_1	o_2	o_3	o_4	$o_\mathbf{t}$
pcf $D_4(o)$	$\mathbf{s}$	c	b	a	f	e

$o_\mathbf{s}[0] = \langle \mathbf{s}, a \wedge c \wedge d \rangle$
$o_1[1] = \langle c \wedge d, b \rangle$
$o_2[0] = \langle a \wedge b, e \rangle$
$o_3[0] = \langle a, f \rangle$
$o_4[0] = \langle f, g \wedge h \rangle$
$o_\mathbf{t}[0] = \langle e \wedge g \wedge h, \mathbf{t} \rangle$



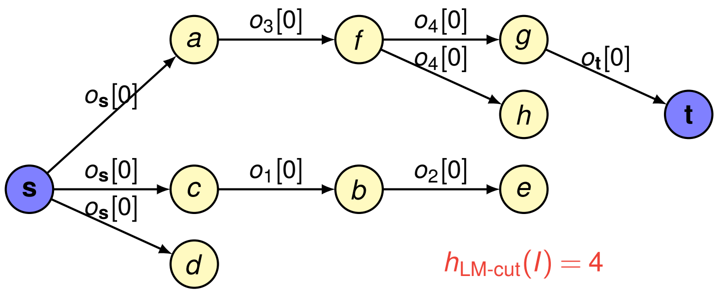
$L_4 = \{o_1\}$, $h_{\text{LM-cut}}(I)$ so far $= 4$

Example: Iteration 5

prop p	$\mathbf{s}$	a	b	c	d	e	f	g	h	$\mathbf{t}$
$h^{c_5}_{\max}(p)$	0	0	0	0	0	0	0	0	0	0

action o	$o_\mathbf{s}$	o_1	o_2	o_3	o_4	$o_\mathbf{t}$
pcf $D_5(o)$	$\mathbf{s}$	c	b	a	f	g

$o_\mathbf{s}[0] = \langle \mathbf{s}, a \wedge c \wedge d \rangle$
$o_1[0] = \langle c \wedge d, b \rangle$
$o_2[0] = \langle a \wedge b, e \rangle$
$o_3[0] = \langle a, f \rangle$
$o_4[0] = \langle f, g \wedge h \rangle$
$o_\mathbf{t}[0] = \langle e \wedge g \wedge h, \mathbf{t} \rangle$



$h_{\text{LM-cut}}(I) = 4$

Admissibility

Theorem

The LM-cut heuristic never overestimates h^+, i.e., it is admissible.

Proof sketch

- From every landmark found, at least one operator has to be applied in any relaxed plan.
- Each found landmark is counted only once and there is no overlap in operators used in landmarks, i.e., the landmarks that are found are disjoint (operator costs for all operators in a "used" landmark are reset to zero).
- Therefore, we count at most as many landmarks as there are operators in a shortest relaxed plan.

Admissibility

- Remark: $h_{\text{LM-cut}}$ can be generalized to planning tasks with non-unit costs.
  - Instead of setting operator costs to zero, decrease costs of all operators in landmark by the minimal cost of any operator in the landmark.
    This effectively leads to a cost partitioning of operator costs between landmarks: An operator can be (partly) counted in more than one landmark, but the sum of the weights it is counted with will not exceed its true cost.
  - Instead of incrementing heuristic value by one in each step, increase it by minimal cost of any operator in the landmark.

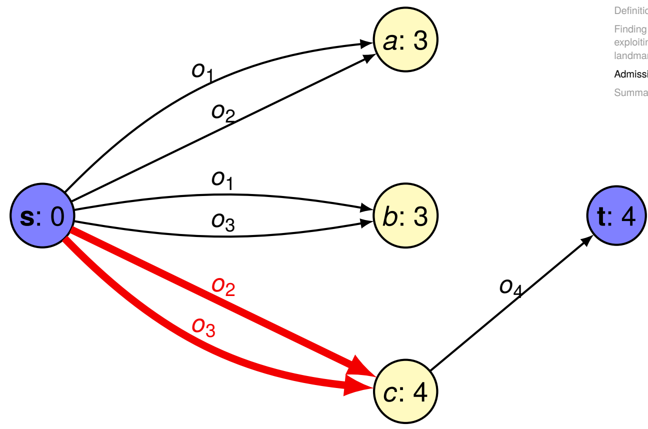
  Then, $h_{\text{LM-cut}}$ is still admissible. Proof via cost-partitioning argument.

Outlook: Non-unit-cost tasks

Example

Iter. 1: $D(\mathbf{t}) = a \rightsquigarrow L = \{o_2, o_3\}$ [4]

$o_1[3] = \langle \mathbf{s}, a \wedge b \rangle$
$o_2[4] = \langle \mathbf{s}, a \wedge c \rangle$
$o_3[5] = \langle \mathbf{s}, b \wedge c \rangle$
$o_4[0] = \langle a \wedge b \wedge c, \mathbf{t} \rangle$

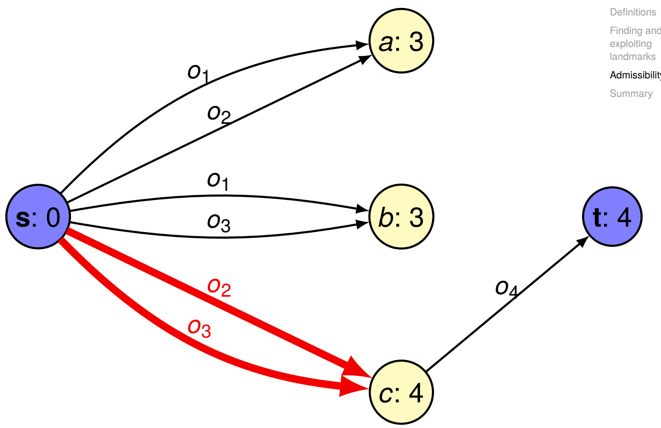


Outlook: Non-unit-cost tasks

Example

Iter. 1: $D(\mathbf{t}) = a \rightsquigarrow L = \{o_2, o_3\}$ [4] $\rightsquigarrow h_{\text{LM-cut}}(I) := 4$

$o_1[3] = \langle \mathbf{s}, a \wedge b \rangle$
$o_2[0] = \langle \mathbf{s}, a \wedge c \rangle$
$o_3[1] = \langle \mathbf{s}, b \wedge c \rangle$
$o_4[0] = \langle a \wedge b \wedge c, \mathbf{t} \rangle$

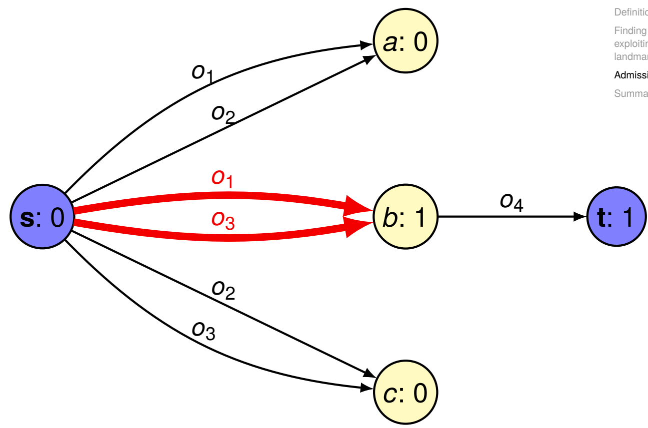


Outlook: Non-unit-cost tasks

Example

Iter. 2: $D(\mathbf{t}) = b \rightsquigarrow L = \{o_1, o_3\}$ [1]

$o_1[3] = \langle \mathbf{s}, a \wedge b \rangle$
$o_2[0] = \langle \mathbf{s}, a \wedge c \rangle$
$o_3[1] = \langle \mathbf{s}, b \wedge c \rangle$
$o_4[0] = \langle a \wedge b \wedge c, \mathbf{t} \rangle$

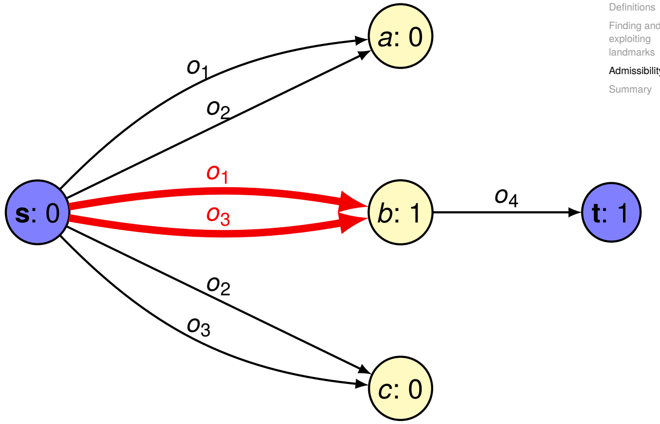


Outlook: Non-unit-cost tasks

Example

Iter. 2: $D(\mathbf{t}) = b \rightsquigarrow L = \{o_1, o_3\}$ [1] $\rightsquigarrow h_{\text{LM-cut}}(I) := 4 + 1 = 5$

$o_1[2] = \langle \mathbf{s}, a \wedge b \rangle$
$o_2[0] = \langle \mathbf{s}, a \wedge c \rangle$
$o_3[0] = \langle \mathbf{s}, b \wedge c \rangle$
$o_4[0] = \langle a \wedge b \wedge c, \mathbf{t} \rangle$

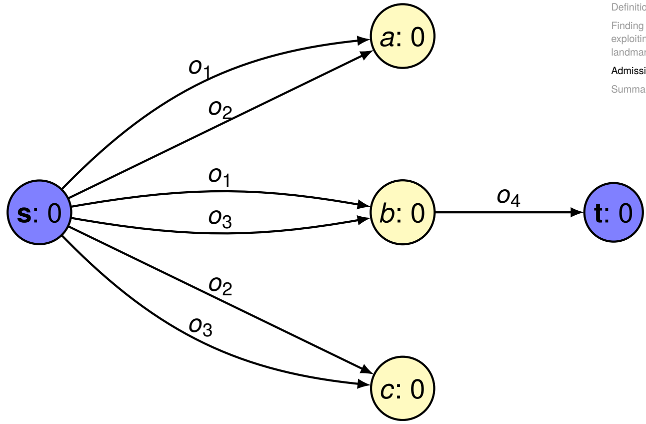


Outlook: Non-unit-cost tasks

Example

Iter. 3: $h_{\max}(\mathbf{t}) = 0 \rightsquigarrow$ done! $\rightsquigarrow h_{\text{LM-cut}}(I) = 5$

$o_1[2] = \langle \mathbf{s}, a \wedge b \rangle$
$o_2[0] = \langle \mathbf{s}, a \wedge c \rangle$
$o_3[0] = \langle \mathbf{s}, b \wedge c \rangle$
$o_4[0] = \langle a \wedge b \wedge c, \mathbf{t} \rangle$



Summary

- Landmarks are sets of actions such that each plan contains at least one of these actions.
- Cuts in justification graphs are a very general method to find landmarks.
- The LM-cut heuristic is an efficient admissible heuristic based on landmarks and cuts.
- It combines delete relaxation, landmarks, and cost partitioning.