Principles of AI Planning

13. Planning as search: the LM-cut heuristic

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel and Robert Mattmüller
January 11th, 2017

The LM-cut heuristic

Motivation

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

March 11th, 2017
B. Nebel, R. Mattmüller – AI Planning 3 / 33
Motivation

Example

Let $\Pi$ be an SAS* planning task and $s$ a state from $\Pi$.
Assume we know the following:
- In each plan starting in $s$, at least one of the operators $a_1$ and $a_2$ is applied.
- In each plan starting in $s$, at least one of the operators $a_3$ and $a_4$ is applied.
- In each plan starting in $s$, the operator $a_5$ is applied.
- In each plan starting in $s$, the operator $a_6$ is applied.
- Operators $a_1, a_2, a_3, a_4, a_5$, and $a_6$ are pairwise different.

**Question:** Does this give us a lower bound on $h^*(s)$?

**Answer:** Yes! The number of landmarks, i.e., $h^*(s) \geq 4$.

Landmarks

Definition (Landmark)

A **landmark** of an SAS* planning task $\Pi$ is a set of actions $L$ such that each plan for $\Pi$ contains at least one action from $L$. A landmark $L$ for $\Pi$ is **minimal** if no $L' \subseteq L$ is a landmark for $\Pi$.

**Note:** Landmarks in this sense are also called **disjunctive action landmarks**.

**Theorem**

Let $\Pi$ with initial state $I$ be an SAS* planning task. If there are $n$ **disjoint landmarks** for $\Pi$, then $h(I) = n$ is an admissible heuristic estimate for state $I$.

**Proof.**

Obvious.

Counterexample for non-disjoint landmarks: Knowing that

- in each plan starting in $s$, at least one of the operators $a_1$ and $a_2$ is applied, and
- in each plan starting in $s$, at least one of the operators $a_2$ and $a_3$ is applied,

does not imply that $h^*(s) \geq 2$, since the one-step action sequence $a_2$ might be a plan for $s$.

Landmarks

**Example**

$\langle A, l, \{a_1, a_2, a_3, a_4, a_5\}, \gamma \rangle$ with

- $A = \{a, b, c, d, e, f, g\}$
  - $l = \{a \mapsto 1\} \cup \{x \mapsto 0 \mid x \neq a\}$
- $a_1 = \langle a, b \wedge c \rangle$
- $a_2 = \langle a, c \wedge d \rangle$
- $a_3 = \langle a, d \wedge e \rangle$
- $a_4 = \langle a, e \wedge b \rangle$
- $a_5 = \langle a, f \rangle$
- $a_6 = \langle b \wedge c \wedge d \wedge e \wedge f \wedge g \rangle$

- $\gamma = g$

**(Minimal) landmarks:**

- $\{a_1, a_2\}$ (because of $c$), $\{a_2, a_3\}$ (because of $d$),
- $\{a_3, a_4\}$ (because of $e$), $\{a_4, a_5\}$ (because of $b$),
- $\{a_5\}$ (because of $f$), $\{a_6\}$ (because of $g$)
The LM-cut heuristic

Motivation

Definitions

Finding and exploiting landmarks

Admissibility

Summary

Landmarks

Example (ctd.)

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

\( \Rightarrow h_{LM}(l) = 4 \) is admissible.

Theorem

Let \( \Pi \) be an SAS\(^+\) planning task, and let \( \Pi^+ \) be its delete relaxation. Let \( L^* = \{ o^* \mid o \in L \} \) be a landmark for \( \Pi^+ \). Then \( L \) is also a landmark for \( \Pi \).

Proof.

Let \( L^* \) be a landmark for \( \Pi^+ \). Then every plan \( \pi^+ \) for \( \Pi^+ \) uses some action \( o^* \in L^* \).

Let \( \pi' \) be some plan for \( \Pi \). We need to show that \( \pi' \) uses some action \( o \in L \). Since \( \pi' \) is a plan for \( \Pi \), also \( \pi'^+ \) is a plan for \( \Pi^+ \).

By assumption, \( \pi'^+ \) must use some action \( o^* \in L^* \). But then, \( \pi' \) uses action \( o \in L \).

\( \square \)

For the rest of this chapter, we assume delete-free planning tasks \( \Pi = \Pi^+ \) and search for landmarks for \( \Pi^+ \), which gives us a good approximation of the optimal delete relaxation heuristic \( h^+ \).

\( \Rightarrow \) It is sufficient to search for landmarks in the delete relaxation. This will only lead to too few discovered landmarks, not too many.

\( \Rightarrow \) Admissibility of the heuristic will be preserved.
Computing Disjoint Disjunctive Action Landmarks

Naive approach:
1. Compute set $L = \{L_1, \ldots, L_n\}$ of all minimal landmarks of planning task $\Pi$.
2. Compute a cardinality-maximal subset $L' \subseteq L$ such that all $L_i, L_j \in L'$, $L_i \neq L_j$, are pairwise disjoint, and return their number, $|L'|$.

Drawbacks of naive approach: Both steps too complicated.

Simpler incomplete approach:
Compute set $L = \{L_1, \ldots, L_n\}$ of some disjoint minimal landmarks for $\Pi$ incrementally.
- Compute some landmark $L_1$.
- When computing $L_{i+1}$, only consider candidates that are disjoint from all previous landmarks $L_1, \ldots, L_i$.
- Stop when no more such landmarks exist.

Assumptions and Definitions

To that end, in the algorithm we will present, action cost values will be iteratively decremented.
- In the first iteration, we have action costs $c_1(o_s) = c_1(o_t) = 0$, and $c_1(o) = 1$ for all other actions $o$.
- Cost functions in later iterations $i + 1$ are denoted by $c_{i+1}$ and will differ from $c_i$ in that costs of actions used in $L_i$ are set to zero.

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.)
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.

LM-cut Heuristic: Motivation

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

Currently best admissible planning heuristic

Pseudocode of LM-cut heuristic

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

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

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

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

Step 4. ...
Pseudocode of LM-cut heuristic (ctd.)

Step 4. Construct an s-t-cut \( C_i = (V^0_i, V^+_i \cup V^0_i) \) of \( G_i \) as follows: \( V^0_i \) 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 \( V^0_i \) contains all remaining propositions. Clearly, \( s \in V^0_i \) and \( t \in V^+_i \).

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 \( c_i+1(o) := c_i(o) \) if \( o \notin L_i \), and \( c_i+1(o) := 0 \) if \( o \in L_i \).

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 \( \langle A, I, \{o_8, o_1, o_2, o_3, o_4, o_1\}, \gamma \rangle \) with

\[
A = \{s, a, b, c, d, e, f, g, h, t\}
\]

\[
I = \{s \mapsto 1, a \mapsto 0, b \mapsto 0, c \mapsto 0, d \mapsto 0, e \mapsto 0, f \mapsto 0, g \mapsto 0, h \mapsto 0, t \mapsto 0\}
\]

\[
o_8 = \{s, a \land c \land d\}
\]

\[
o_1 = \{c \land d, b\}
\]

\[
o_2 = \{a \land b, e\}
\]

\[
o_3 = \{a, f\}
\]

\[
o_4 = \{f, g \land h\}
\]

\[
o_1 = \{e \land g \land h, t\}
\]

\[
\gamma = t
\]

Example: Iteration 1

- Cheapest sequential (relaxed) plan: \( \langle o_8, o_1, o_2, o_3, o_4, o_1 \rangle \) with cost \( h^*(I) = 4 \) (recall that \( o_8 \) and \( o_1 \) cost nothing).
- Parallel (relaxed) plan witnessing \( h_{\max}(I) = 2 \): \( \langle \{o_8\}, \{o_1, o_3\}, \{o_2, o_4\}, \{o_1\} \rangle \).

Our aim: Get closer to \( h^*(I) = 4 \) using \( h_{\text{LM-cut}} \) than using \( h_{\max} \).
**Example: Iteration 2**

The LM-cut heuristic:

- Motivation
- Definitions
- Finding and exploiting landmarks
- Admissibility

**Summary**

- Example: Iteration 2

```
prop \( p \)    | s | a | b | c | d | e | f | g | h | t
\h_{\text{LM-cut}}(p) | 0 | 0 | 1 | 0 | 0 | 2 | 1 | 1 | 1 | 2
```

Action: \( o \)

```
\text{pcf} D_3(0) | s | c | b | a | f | e
```

- \( o_1 \) = (s, a ∨ c ∧ d)
- \( o_2 \) = (c ∨ a ∧ b, d)
- \( o_3 \) = (a ∧ b, e)
- \( o_4 \) = (a, f)
- \( o_5 \) = (f, g ∨ h)
- \( o_6 \) = (e ∨ g ∧ h, t)

- \( o_6[0] = \{ s, a ∨ c ∧ d \} \)
- \( o_1[1] = \{ c ∨ a ∧ b, d \} \)
- \( o_2[0] = \{ a ∧ b, e \} \)
- \( o_3[1] = \{ a, f \} \)
- \( o_4[0] = \{ f, g ∨ h \} \)
- \( o_5[0] = \{ e ∨ g ∧ h, t \} \)

**Example: Iteration 3**

```
prop \( p \)    | s | a | b | c | d | e | f | g | h | t
\h_{\text{LM-cut}}(p) | 0 | 0 | 1 | 0 | 0 | 2 | 1 | 1 | 1 | 1
```

Action: \( o \)

```
\text{pcf} D_3(0) | s | c | b | a | f | e
```

- \( o_1 \) = (s, a ∨ c ∧ d)
- \( o_2 \) = (c ∨ a ∧ b, d)
- \( o_3 \) = (a ∧ b, e)
- \( o_4 \) = (a, f)
- \( o_5 \) = (f, g ∨ h)
- \( o_6 \) = (e ∨ g ∧ h, t)

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

**Example: Iteration 4**

```
prop \( p \)    | s | a | b | c | d | e | f | g | h | t
\h_{\text{LM-cut}}(p) | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0
```

Action: \( o \)

```
\text{pcf} D_4(0) | s | c | b | a | f | e
```

- \( o_1 \) = (s, a ∨ c ∧ d)
- \( o_2 \) = (c ∨ a ∧ b, d)
- \( o_3 \) = (a ∧ b, e)
- \( o_4 \) = (a, f)
- \( o_5 \) = (f, g ∨ h)
- \( o_6 \) = (e ∨ g ∧ h, t)

\( o_6[0] = \{ s, a ∨ c ∧ d \} \)
\( o_1[1] = \{ c ∨ a ∧ b, d \} \)
\( o_2[0] = \{ a ∧ b, e \} \)
\( o_3[1] = \{ a, f \} \)
\( o_4[0] = \{ f, g ∨ h \} \)
\( o_5[0] = \{ e ∨ g ∧ h, t \} \)

**Example: Iteration 5**

```
prop \( p \)    | s | a | b | c | d | e | f | g | h | t
\h_{\text{LM-cut}}(p) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
```

Action: \( o \)

```
\text{pcf} D_5(0) | s | c | b | a | f | g
```

- \( o_1 \) = (s, a ∨ c ∧ d)
- \( o_2 \) = (c ∨ a ∧ b, d)
- \( o_3 \) = (a ∧ b, e)
- \( o_4 \) = (a, f)
- \( o_5 \) = (f, g ∨ h)
- \( o_6 \) = (e ∨ g ∧ h, t)

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

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

\( L_5 = \{ o_5 \}, h_{\text{LM-cut}}(l) \) so far = 5

\( h_{\text{LM-cut}}(l) = 4 \)
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.

Remark: $h_{LM-cut}$ 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, $h_{LM-cut}$ is still admissible. Proof via cost-partitioning argument.

Outlook: Non-unit-cost tasks

Example
Iter. 1: $D(t) = c \rightsquigarrow L_1 = \{o_2, o_3\}$
Outlook: Non-unit-cost tasks

Example
Iter. 2: \( D(t) = b \leadsto L_2 = \{ o_1, o_3 \} \) \[1\]

\[ o_1[3] = (s, a \land b) \]
\[ o_2[0] = (s, a \land c) \]
\[ o_3[1] = (s, b \land c) \]
\[ o_4[0] = (a \land b \land c, t) \]

Remark: The costs of \( o_3 \) (i.e., 5) were partitioned as follows:
- 4 cost units were used in the cost of \( L_1 \), and
- 1 cost unit was used in the cost of \( L_2 \).

Without this cost partitioning, we would have only found \( L_1 \) and counted it at a cost of 4. Landmark \( L_2 \) would not have been considered, since it is not disjoint from \( L_1 \).

Thus, we would have arrived at an unnecessarily low value \( h_{LM-cut}(I) = 4 \) instead of \( h_{LM-cut}(I) = 5 \).
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.