Planning and Optimization

F3. Landmarks: Orderings & LM-Count Heuristic

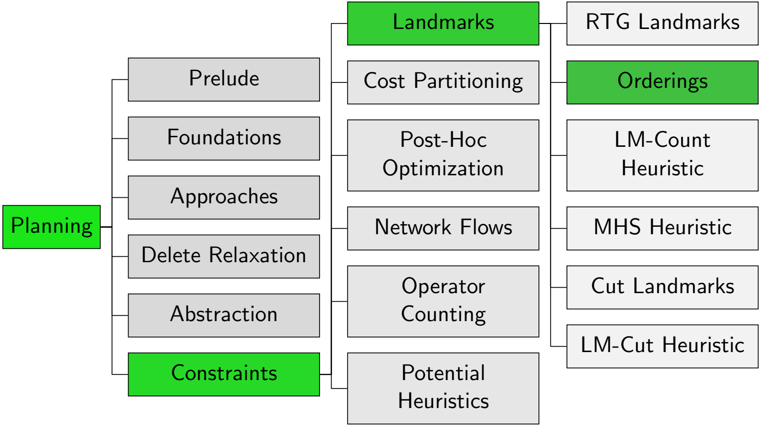
Malte Helmert and Gabriele Röger

Universität Basel

December 1, 2025

F3.1 Landmark Orderings

Content of the Course



Why Landmark Orderings?

- To compute a landmark heuristic estimate for state s we need landmarks for s.
- We could invest the time to compute them for every state from scratch.
- Alternatively, we can compute landmarks once and propagate them over operator applications.
- Landmark orderings are used to detect landmarks that should be further considered because they (again) need to be satisfied later.

- (We will later see yet another approach, where heuristic computation and landmark computation are integrated $\rightsquigarrow$ LM-Cut.)

Example

Consider task $\langle\{a, b, c, d\}, I, \{o_1, o_2, \ldots, o_n\}, d\rangle$ with

- $I(v) = \bot$ for $v \in \{a, b, c, d\}$,
- $o_1 = \langle\top, a \land b\rangle$, and
- $o_2 = \langle a, c \land \neg a \land \neg b\rangle$ (plus some more operators).

You know that a, b, c and d are all fact landmarks for I.

- What landmarks are still required to be made true in state $I[\![\langle o_1, o_2\rangle]\!]$?
- You get the additional information that variable a must be true immediately before d is first made true. Any changes?

Terminology

Let $\pi = \langle o_1, \dots, o_n \rangle$ be a sequence of operators applicable in state I and let φ be a formula over the state variables.

- φ is true at time i if $I[\![\langle o_1, \dots, o_i \rangle]\!] \models \varphi$.
- Also special case $i = 0$: φ is true at time 0 if $I \models \varphi$.
- No formula is true at time $i < 0$.
- φ is added at time i if it is true at time i but not at time $i - 1$.
- φ is first added at time i if it is true at time i but not at any time $j < i$.
  We denote this i by $\mathit{first}(\varphi, \pi)$.
- $\mathit{last}(\varphi, \pi)$ denotes the last time in which φ is added in π.

Landmark Orderings

Definition (Landmark Orderings)

Let φ and ψ be formula landmarks. There is

- a natural ordering between φ and ψ (written $\varphi \to \psi$) if in each plan π it holds that $\mathit{first}(\varphi, \pi) < \mathit{first}(\psi, \pi)$.
  "φ must be true some time strictly before ψ is first added."
- a greedy-necessary ordering between φ and ψ (written $\varphi \to_{\mathrm{gn}} \psi$) if for every plan $\pi = \langle o_1, \dots, o_n \rangle$ it holds that $s[\![\langle o_1, \dots, o_{\mathit{first}(\psi,\pi)-1} \rangle]\!] \models \varphi$.
  "φ must be true immediately before ψ is first added."
- a weak ordering between φ and ψ (written $\varphi \to_{\mathrm{w}} \psi$) if in each plan π it holds that $\mathit{first}(\varphi, \pi) < \mathit{last}(\psi, \pi)$.
  "φ must be true some time before ψ is last added."

Not covered: reasonable orderings, which generalize weak orderings

Natural Orderings

Definition

There is a natural ordering between φ and ψ (written $\varphi \to \psi$) if in each plan π it holds that $\mathit{first}(\varphi, \pi) < \mathit{first}(\psi, \pi)$.

- We can directly determine natural orderings from the *LM* sets computed from the simplified relaxed task graph.
- For fact landmarks v, v' with $v \neq v'$, if $n_{v'} \in LM(n_v)$ then $v' \to v$.

Greedy-necessary Orderings

Definition

There is a greedy-necessary ordering between φ and ψ (written $\varphi \rightarrow_{\mathrm{gn}} \psi$) if in each plan where ψ is first added at time i, φ is true at time $i-1$.

- We can again determine such orderings from the sRTG.
- For an OR node n_v, we define the set of first achievers as $FA(n_v) = \{n_o \mid n_o \in succ(n_v) \text{ and } n_v \notin LM(n_o)\}$.
- Then $v' \rightarrow_{\mathrm{gn}} v$ if $n_{v'} \in succ(n_o)$ for all $n_o \in FA(n_v)$.

F3.2 Landmark Propagation

Example Revisited

Consider task $\langle\{a, b, c, d\}, I, \{o_1, o_2, \ldots, o_n\}, d\rangle$ with

- $I(v) = \bot$ for $v \in \{a, b, c, d\}$,
- $o_1 = \langle\top, a \land b\rangle$ and $o_2 = \langle a, c \land \neg a \land \neg b\rangle$ (plus some more).

You know that a, b, c and d are all fact landmarks for I.

- What landmarks are still required to be made true in state $I[\![\langle o_1, o_2\rangle]\!]$? All not achieved yet on the state path
- You get the additional information that variable a must be true immediately before d is first made true. Any changes?
  Exploit orderings to determine landmarks that are still required.
- There is another path to the same state where b was never true. What now?
  Exploit information from multiple paths.

Past and Future Landmarks

- In the following, $\mathcal{L}_I$ is always a set of formula landmarks for the initial state with set of orderings $\mathcal{O}_I$.
- The set $\mathcal{L}^*_{\text{past}}(s)$ of past landmarks of a state s contains all landmarks from $\mathcal{L}_I$ that are at some point true in every path from the initial state to s.
- The set $\mathcal{L}^*_{\text{fut}}(s)$ of future landmarks of a state s contains all landmarks from $\mathcal{L}_I$ that are also landmarks of s but not true in s.
- Past landmarks are important for inferring which orderings are still relevant, future landmarks are relevant for the heuristic estimates.
- Since the exact sets are defined over all paths between certain states, we use approximations.

Landmark State

Definition

Let $\mathcal{L}_I$ be a set of formula landmarks for the initial state.

A landmark state $\mathbb{L}$ is $\bot$ or a pair $\langle \mathcal{L}_{\text{past}}, \mathcal{L}_{\text{fut}} \rangle$ such that $\mathcal{L}_{\text{fut}} \cup \mathcal{L}_{\text{past}} = \mathcal{L}_I$.

$\mathbb{L}$ is valid in state s if

- $\mathbb{L} = \bot$ and Π has no s-plan, or
- $\mathbb{L} = \langle \mathcal{L}_{\text{past}}, \mathcal{L}_{\text{fut}} \rangle$ with $\mathcal{L}_{\text{past}} \supseteq \mathcal{L}^*_{\text{past}}$ and $\mathcal{L}_{\text{fut}} \subseteq \mathcal{L}^*_{\text{fut}}$.

Context in Search: LM-BFS Algorithm

$\mathbb{L}(\text{init}), \mathcal{L}_I, \mathcal{O}_I := \text{compute\_landmark\_info}(\text{init}())$
if $h(\text{init}(), \mathbb{L}(\text{init})) < \infty$ **then**
    $\textit{open}.\text{insert}(\langle \text{init}(), 0, h(\text{init}(), \mathbb{L}(\text{init}))\rangle)$
while $\textit{open} \neq \emptyset$ **do**
    $\langle s, g, v\rangle = \textit{open}.\text{pop}()$
    if $v < h(s, \mathbb{L}(s))$ **then**
        $\textit{open}.\text{insert}(\langle s, g, h(s, \mathbb{L}(s))\rangle)$
    else if $g < \textit{distances}(s)$ **then**
        $\textit{distances}(s) := g$
        if $\text{is\_goal}(s)$ **then return** $\text{extract\_plan}(s)$;
        foreach $\langle a, s'\rangle \in \textit{succ}(s)$ **do**
            $\mathbb{L}' := \text{progress\_landmark\_state}(\mathbb{L}(s), \langle s, a, s'\rangle)$
            $\mathbb{L}(s') := \text{merge\_landmark\_states}(\mathbb{L}(s'), \mathbb{L}')$
            if $\mathbb{L}(s') \neq \bot$ *and* $h(s', \mathbb{L}(s')) < \infty$ **then**
                $\textit{open}.\text{insert}(\langle s', g + \textit{cost}(a), h(s', \mathbb{L}(s'))\rangle)$

$\mathbb{L}(s) := \langle \mathcal{L}_I, \emptyset\rangle$ and $\textit{distances}(s) := \infty$ if read before set.

Context: Exploit Information from Multiple Paths

$\mathbb{L}(\text{init}), \mathcal{L}_I, \mathcal{O}_I := \text{compute\_landmark\_info}(\text{init}())$
if $h(\text{init}(), \mathbb{L}(\text{init})) < \infty$ **then**
    $\mathit{open}.\text{insert}(\langle \text{init}(), 0, h(\text{init}(), \mathbb{L}(\text{init}))\rangle)$
while $\mathit{open} \neq \emptyset$ **do**
    $\langle s, g, v\rangle = \mathit{open}.\text{pop}()$
    if $v < h(s, \mathbb{L}(s))$ **then**
        $\mathit{open}.\text{insert}(\langle s, g, h(s, \mathbb{L}(s))\rangle)$
    else if $g < \mathit{distances}(s)$ **then**
        $\mathit{distances}(s) := g$
        if $\text{is\_goal}(s)$ **then return** $\text{extract\_plan}(s)$;
        foreach $\langle a, s'\rangle \in \mathit{succ}(s)$ **do**
            $\mathbb{L}' := \text{progress\_landmark\_state}(\mathbb{L}(s), \langle s, a, s'\rangle)$
            $\mathbb{L}(s') := \text{merge\_landmark\_states}(\mathbb{L}(s'), \mathbb{L}')$
            if $\mathbb{L}(s') \neq \bot$ *and* $h(s', \mathbb{L}(s')) < \infty$ **then**
                $\mathit{open}.\text{insert}(\langle s', g + \mathit{cost}(a), h(s', \mathbb{L}(s'))\rangle)$

$\mathbb{L}(s) := \langle \mathcal{L}_I, \emptyset\rangle$ and $\mathit{distances}(s) := \infty$ if read before set.

Merging Landmark States

Merging combines the information from two landmark states.

merge_landmark_states($\mathbb{L}, \mathbb{L}'$)

if $\mathbb{L} = \bot$ *or* $\mathbb{L}' = \bot$ **then** return $\bot$;
$\langle \mathcal{L}_{\text{past}}, \mathcal{L}_{\text{fut}} \rangle := \mathbb{L}$
$\langle \mathcal{L}'_{\text{past}}, \mathcal{L}'_{\text{fut}} \rangle := \mathbb{L}'$
return $\langle \mathcal{L}_{\text{past}} \cap \mathcal{L}'_{\text{past}}, \mathcal{L}_{\text{fut}} \cup \mathcal{L}'_{\text{fut}} \rangle$

Theorem

If $\mathbb{L}$ and $\mathbb{L}'$ are valid in a state s then also merge_landmark_states($\mathbb{L}, \mathbb{L}'$) is valid in s.

Context: Progression for a Transition

$\mathbb{L}(\text{init}), \mathcal{L}_I, \mathcal{O}_I := \text{compute\_landmark\_info}(\text{init}())$
if $h(\text{init}(), \mathbb{L}(\text{init})) < \infty$ **then**
    $\mathit{open}.\text{insert}(\langle \text{init}(), 0, h(\text{init}(), \mathbb{L}(\text{init}))\rangle)$
while $\mathit{open} \neq \emptyset$ **do**
    $\langle s, g, v\rangle = \mathit{open}.\text{pop}()$
    if $v < h(s, \mathbb{L}(s))$ **then**
        $\mathit{open}.\text{insert}(\langle s, g, h(s, \mathbb{L}(s))\rangle)$
    else if $g < \mathit{distances}(s)$ **then**
        $\mathit{distances}(s) := g$
        if $\text{is\_goal}(s)$ **then return** $\text{extract\_plan}(s)$;
        foreach $\langle a, s'\rangle \in \mathit{succ}(s)$ **do**
            $\mathbb{L}' := \text{progress\_landmark\_state}(\mathbb{L}(s), \langle s, a, s'\rangle)$
            $\mathbb{L}(s') := \text{merge\_landmark\_states}(\mathbb{L}(s'), \mathbb{L}')$
            if $\mathbb{L}(s') \neq \bot$ *and* $h(s', \mathbb{L}(s')) < \infty$ **then**
                $\mathit{open}.\text{insert}(\langle s', g + \mathit{cost}(a), h(s', \mathbb{L}(s'))\rangle)$

$\mathbb{L}(s) := \langle \mathcal{L}_I, \emptyset\rangle$ and $\mathit{distances}(s) := \infty$ if read before set.

Progressing Landmark States

- If we expand a state s with transition $\langle s, o, s' \rangle$, we use progression to determine a landmark state for s' from the one we know for s.
- We will only introduce progression methods that preserve the validity of landmark states.
- Since every progression method gives a valid landmark state, we can merge results from different methods into a valid landmark state.

Basic Progression

Definition (Basic Progression)

Basic progression maps landmark state $\langle \mathcal{L}_{\text{past}}, \mathcal{L}_{\text{fut}} \rangle$ and transition $\langle s, o, s' \rangle$ to landmark state $\langle \mathcal{L}_{\text{past}} \cup \mathcal{L}_{\text{add}}, \mathcal{L}_{\text{fut}} \setminus \mathcal{L}_{\text{add}} \rangle$, where $\mathcal{L}_{\text{add}} = \{\varphi \in \mathcal{L}_I \mid s \not\models \varphi \text{ and } s' \models \varphi\}$.

"Extend the past with all landmarks added in s' and remove them from the future."

Goal Progression

Definition (Goal Progression)

Let γ be the goal of the task.
Goal progression maps landmark state $\langle \mathcal{L}_{\text{past}}, \mathcal{L}_{\text{fut}} \rangle$ and transition $\langle s, o, s' \rangle$ to landmark state $\langle \mathcal{L}_I, \mathcal{L}_{\text{goal}} \rangle$, where $\mathcal{L}_{\text{goal}} = \{\varphi \in \mathcal{L}_I \mid \gamma \models \varphi \text{ and } s' \not\models \varphi\}$.

"All landmarks that must be true in the goal but are false in s' must be achieved in the future."

Weak Ordering Progression

$\varphi \rightarrow_{\mathrm{w}} \psi$: "$\varphi$ must be true some time before ψ is last added."

Definition (Weak Ordering Progression)

The weak ordering progression maps landmark state $\langle \mathcal{L}_{\mathrm{past}}, \mathcal{L}_{\mathrm{fut}} \rangle$ and transition $\langle s, o, s' \rangle$ to landmark state $\langle \mathcal{L}_I, \{\psi \mid \exists \varphi \rightarrow_{\mathrm{w}} \psi : \varphi \notin \mathcal{L}_{\mathrm{past}}\} \rangle$.

"Landmark ψ must be added in the future because we haven't done something that must be done before ψ is last added."

Greedy-necessary Ordering Progression

$\varphi \rightarrow_{\text{gn}} \psi$: "$\varphi$ must be true immediately before ψ is first added."

Definition (Greedy-necessary Ordering Progression)

The greedy necessary ordering progression maps landmark state $\langle \mathcal{L}_{\text{past}}, \mathcal{L}_{\text{fut}} \rangle$ and transition $\langle s, o, s' \rangle$ to landmark state

- $\perp$ if there is a $\varphi \rightarrow_{\text{gn}} \psi \in \mathcal{O}_I$ with $\psi \notin \mathcal{L}_{\text{past}}, s \not\models \varphi$ and $s' \models \psi$, and
- $\langle \mathcal{L}_I, \{\varphi \mid s' \not\models \varphi \text{ and } \exists \varphi \rightarrow_{\text{gn}} \psi \in \mathcal{O}_I : \psi \notin \mathcal{L}_{\text{past}}, s' \not\models \psi\} \rangle$ otherwise.

"Landmark ψ has not been true, yet, and φ must be true immediately before it becomes true. Since φ is currently false, we must make it true in the future (before making ψ true)."

Natural Ordering Progression

$\varphi \to \psi$: φ must be true some time strictly before ψ is first added.

Definition (Natural Ordering Progression)

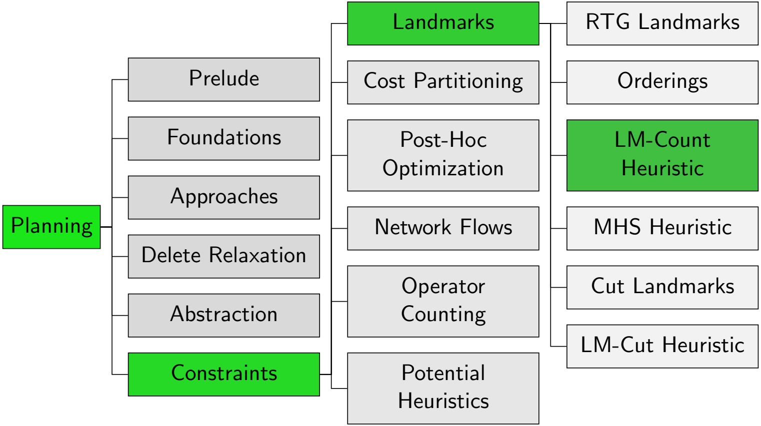
The natural ordering progression maps landmark state $\langle \mathcal{L}_{\text{past}}, \mathcal{L}_{\text{fut}} \rangle$ and transition $\langle s, o, s' \rangle$ to landmark state

- $\bot$ if there is a $\varphi \to \psi \in \mathcal{O}_I$ with $\varphi \notin \mathcal{L}_{\text{past}}$ and $s' \models \psi$, and
- $\langle \mathcal{L}_I, \emptyset \rangle$ otherwise.

Not (yet) useful: All known methods only find natural orderings that are true for every applicable operator sequence, so the interesting first case never happens in LM-BFS.

F3.3 Landmark-count Heuristic

Content of the Course



Landmark-count Heuristic

The landmark-count heuristic counts the landmarks that still have to be achieved.

Definition (LM-count Heuristic)

Let Π be a planning task, s be a state and $\mathbb{L} = \langle \mathcal{L}_{\mathrm{past}}, \mathcal{L}_{\mathrm{fut}} \rangle$ be a valid landmark state for s.

The LM-count heuristic for s and $\mathbb{L}$ is

$$h^{\text{LM-count}}(s, \mathbb{L}) = \begin{cases} \infty & \text{if } \mathbb{L} = \bot, \\ |\mathcal{L}_{\mathrm{fut}}| & \text{otherwise} \end{cases}$$

In the original work, $\mathcal{L}_{\mathrm{fut}}$ was determined without considering information from multiple paths and could not detect dead-ends.

LM-count Heuristic is Path-dependent

- LM-count heuristic gives estimates for landmark states, which depend on the considered paths.
- Search algorithms need estimates for states.
- $\rightsquigarrow$ we use estimate from the current landmark state.
- $\rightsquigarrow$ heuristic estimate for a state is not well-defined.

LM-count Heuristic is Inadmissible

Example

Consider STRIPS planning task $\Pi = \langle \{a, b\}, I, \{o\}, \{a, b\} \rangle$ with $I = \emptyset$, $o = \langle \emptyset, \{a, b\}, \emptyset, 1 \rangle$. Let $\mathcal{L} = \{a, b\}$ and $\mathcal{O} = \emptyset$.

Landmark state $\langle \emptyset, \mathcal{L} \rangle$ for the initial state is valid and the estimate is $h^{\text{LM-count}}(I, \langle \emptyset, \{a, b\} \rangle) = 2$
while $h^*(I) = 1$.

$\rightsquigarrow$ $h^{\text{LM-count}}$ is inadmissible.

LM-count Heuristic: Comments

- LM-Count alone is not a particularly informative heuristic.
- On the positive side, it complements h^{FF} very well.
- For example, the LAMA planning system alternates between expanding a state with minimal h^{FF} and minimal $h^{\text{LM-count}}$ estimate.
- The LM-sum heuristic is a cost-aware variant of the heuristic that sums up the costs of the cheapest achiever (= operator that adds the fact landmark) of each landmark.
- There is an admissible variant of the heuristic based on operator cost partitioning.

F3.4 Summary

Summary

- We can propagate landmark sets over action applications.
- Landmark orderings can be useful for detecting when a landmark that has already been achieved should be further considered.
- We can combine the landmark information from several paths to the same state.
- The LM-count heuristic counts how many landmarks still need to be satisfied.
- The LM-count heuristic is inadmissible (but there is an admissible variant).