Planning for Artificial Intelligence

Lukáš Chrpa

Landmarks and LM-Cut Heuristic

Landmarks

Landmarks

- In general, a **landmark** is a formula that must be true at some point for every plan
- Landmarks can be (partially) **ordered**
- A **fact landmark** is a fact (or atom) that must be true at some point for every plan
- An **action landmark** is an action that must occur in every plan
- A **disjunctive** fact (action) landmark stands for that at least one of the fact must be true (at least one action must occur) in every plan
- A **conjunctive** fact landmark stands for that all the facts must be true at the same time in every plan

Fact and Action Landmarks

- A fact landmark implies an action landmark if the action is the only one achieving it
- An action landmark implies fact landmarks (action’s preconditions and effects)
- Deciding fact or action landmark is PSPACE-complete
  - The same as deciding whether a task without actions achieving the fact landmark, or an action standing for an action landmark, respectively, is solvable
- Subsets of fact or action landmarks can be identified easily

Landmark Orderings

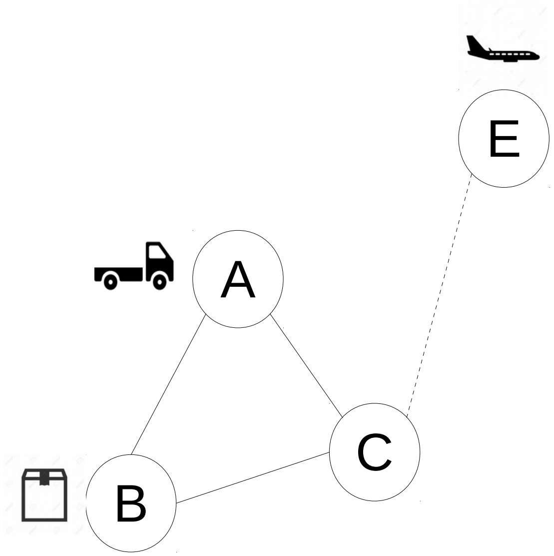
- For landmarks p and q we define the following types of ordering
  - **Natural ordering** $p \rightarrow q$ iff p is true some time before q
  - **Greedy necessary ordering** $p \rightarrow_{gn} q$ iff p is true one step before q becomes true for the first time
  - **Necessary ordering** $p \rightarrow_{n} q$ iff p is always true one step before q becomes true
- Deciding all types of orderings is PSPACE-complete
- Again, some landmark orderings can be identified easily

Landmark Graph

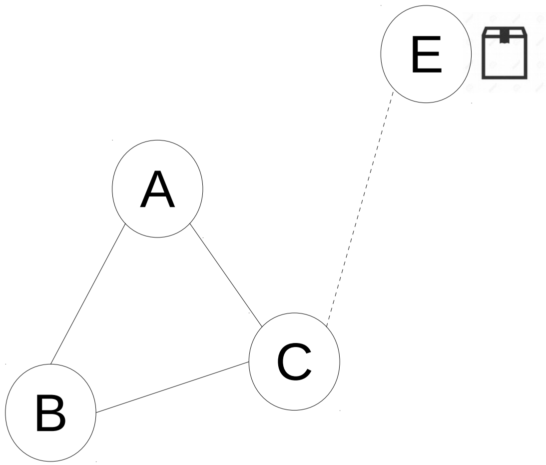
- Let LG=(V,E) be a directed graph, where V are landmarks and $(v_i,v_j) \in E$ if $v_i \rightarrow v_j$ (natural ordering between landmarks v_i and v_j). LG is a **landmark graph**

- Note that landmark graphs are often partial (as we don’t know all the landmarks as well as some of their orderings)

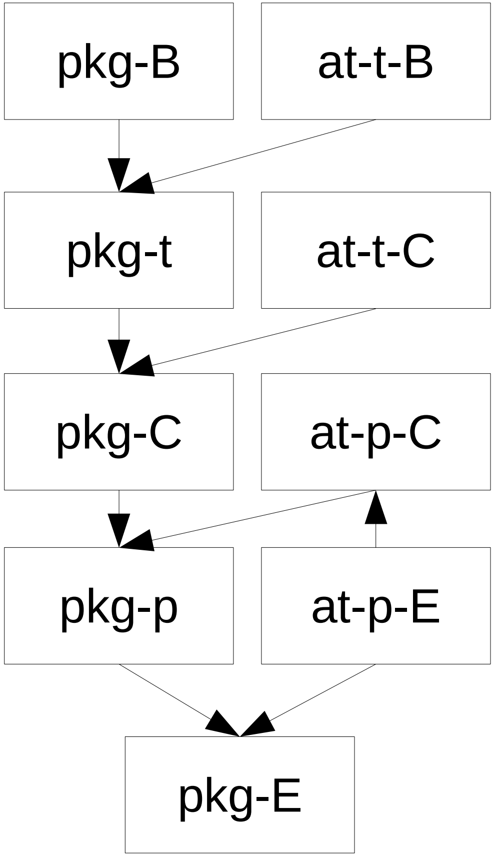
(Enhanced) Logistics Example of Landmark Graph



Initial state



Goal



Towards (Fact) Landmark Discovery

- Let $\Pi=(P,A,I,G)$ be a planning task and $p \in P$ be a fact such that $p \notin I$. We denote Π_{-p} a planning task, where $\Pi_{-p}=(P, A \setminus \{a \mid p \in add(a)\}, I, G)$.

Theorem: p is a fact landmark iff Π_{-p} is unsolvable

- It also holds that if the (delete-)relaxed task Π^{+}_{-p} is unsolvable, then Π_{-p} is unsolvable
  - Let’s find some (fact) landmarks by leveraging **delete-relaxation !**

Landmark Discovery by the Backchaining Method

- Let Π=(P,A,I,G) be a planning task, then

  1) for each **p∈G**, it is the case that **p** is a **fact landmark**

  2) if **p** is a **fact landmark** and p∉I, then for each

  $\mathbf{q \in \bigcap_{a \in \{a' | a' \in A,\ p \in add(a')\}} pre(a)}$ it is the case that **q** is a **fact landmark** and $\mathbf{q \rightarrow_n p}$

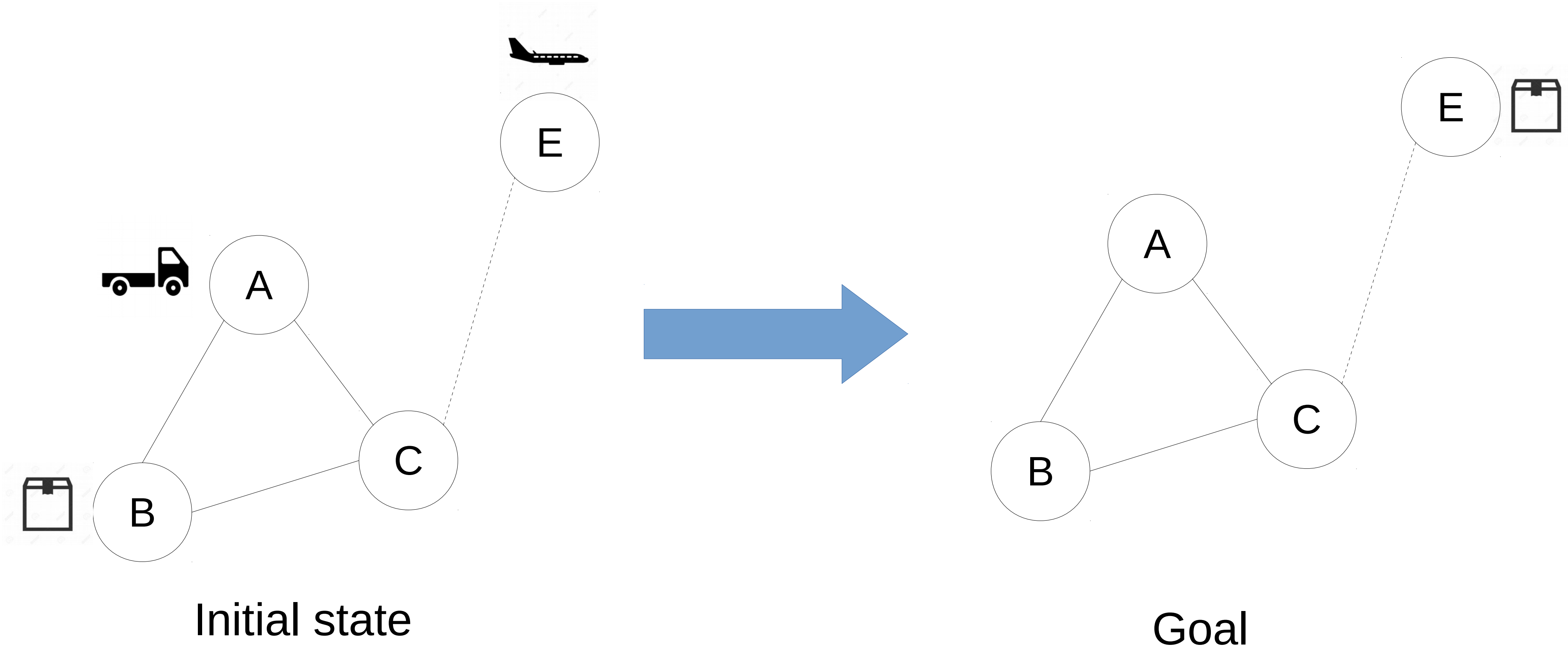
  - q is in preconditions of all actions achieving p

- Can we improve ?

Concerning First Achievers

- An action is a **first achiever** of a fact (or atom) if it achieves (adds) it for the first time
- For a planning task Π and a fact landmark p, we construct a **reachability graph** for Π_{-p} (p won't be reachable unless p∈I)
  - Any action applicable in this graph can possibly be applied before p becomes true → **possible first achievers**
  - The rule 2) of the backchaining method is enhanced by **considering only actions applicable in the last atom layer of the reachability graph**
    - we then get $q \rightarrow_{gn} p$
    - also, more fact landmarks can be identified, why ?

(Enhanced) Logistics Example



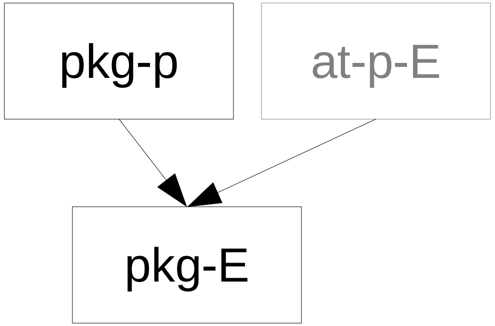
(Enhanced) Logistics Example – Landmark Identification

Goal fact: pkg-E

- achieved only by unload-p-E
- pkg-p, at-p-E are preconditions of unload-p-E and thus fact landmarks

Landmark: pkg-p

- achieved by load-p-C and load-p-E
- no shared preconditions …



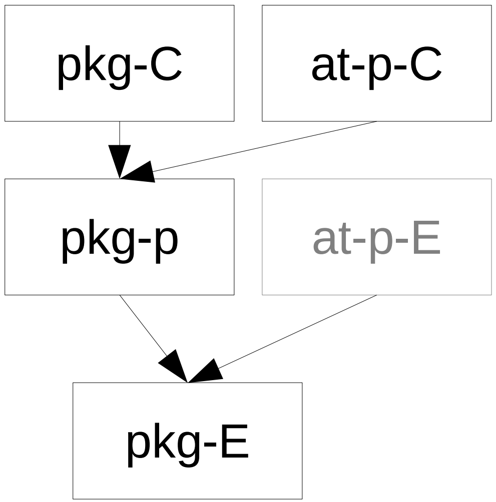
(Enhanced) Logistics Example – Landmark Identification

Goal fact: pkg-E

- achieved only by unload-p-E
- pkg-p, at-p-E are preconditions of unload-p-E and thus fact landmarks

Landmark: pkg-p

- achieved by load-p-C and ~~load-p-E~~
- pkg-C, at-p-C are preconditions of load-p-C and thus fact landmarks



(Enhanced) Logistics Example – Landmark Identification

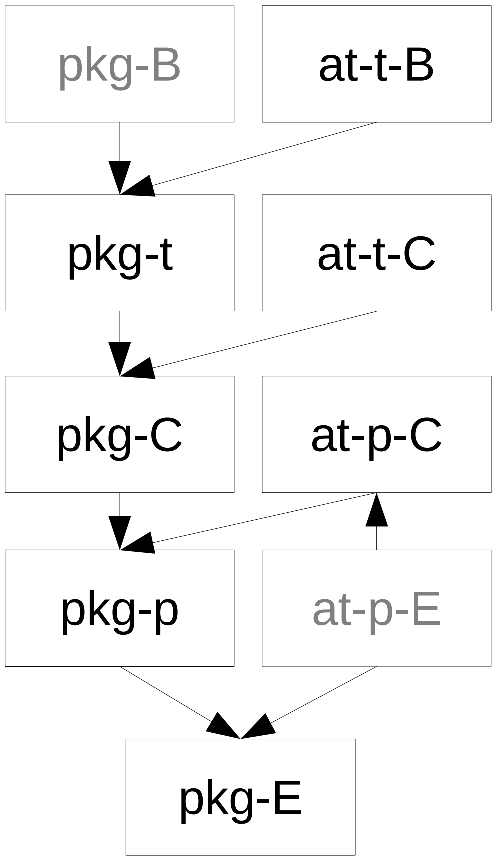
Goal fact: pkg-E

- achieved only by unload-p-E
- pkg-p, at-p-E are preconditions of unload-p-E and thus fact landmarks

Landmark: pkg-p

- achieved by load-p-C and ~~load-p-E~~
- pkg-C, at-p-C are preconditions of load-p-C and thus fact landmarks

… to think about at home



Domain Transition Graph

- A **Domain Transition Graph** of a variable v (DTG_v) represents how the value of v can change
- For a planning task (V,A,I,G) and a variable $v \in V$, DTG_v is defined as follows:
  - Nodes are D(v)
  - (d,d') is an edge iff
    - $d \neq d'$
    - $\exists a \in A: (v=d') \in eff(a)$ and $(v=d) \in pre(a)$, or a has no precondition on v

Landmark Discovery via DTG

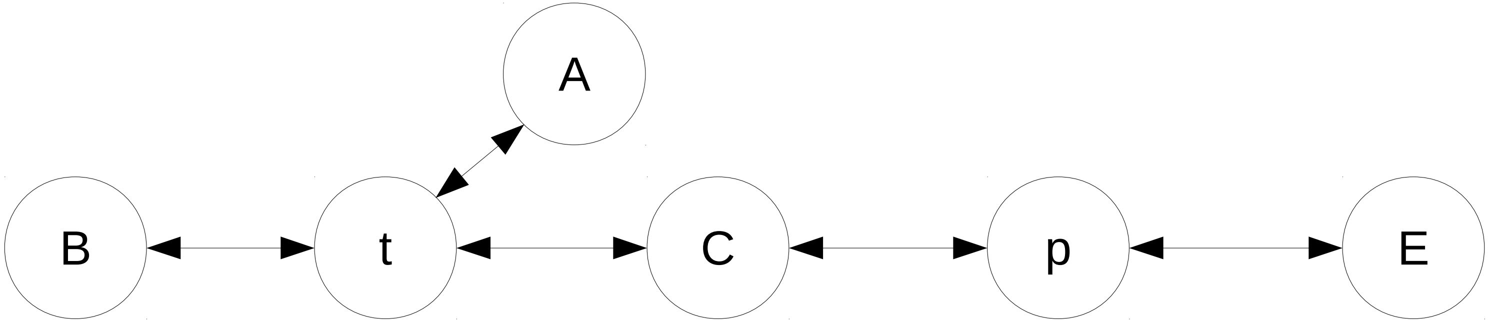
Having DTG_v, where:

- $I[v]=d_0$
- $v=d$ is a fact landmark
- d' is on every path from d_0 to d in DTG_v

then, **v=d' is a fact landmark** and $(v=d') \rightarrow (v=d)$

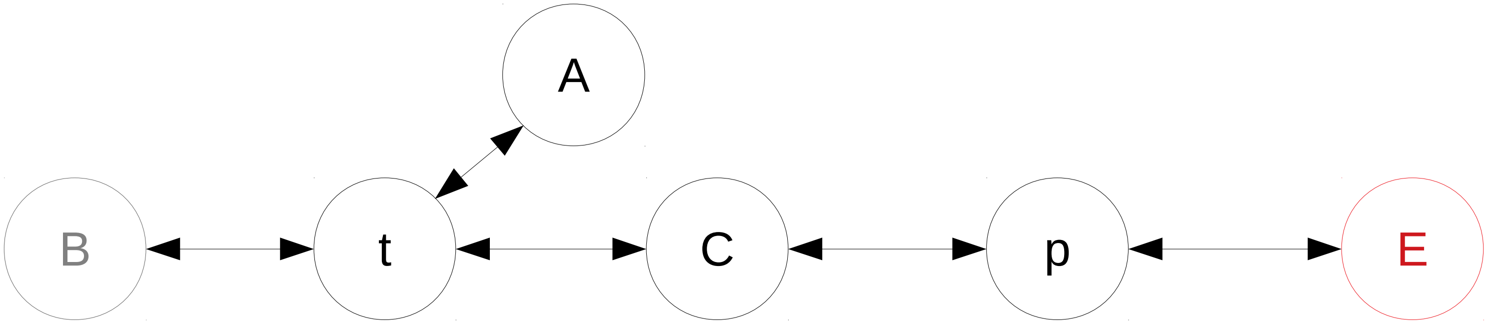
(Enhanced) Logistics Example – Landmark Identification from DTG

Let's consider DTG_v (where v represents a position of the package)



(Enhanced) Logistics Example – Landmark Identification from DTG

Let's consider DTG_v (where v represents a position of the package)

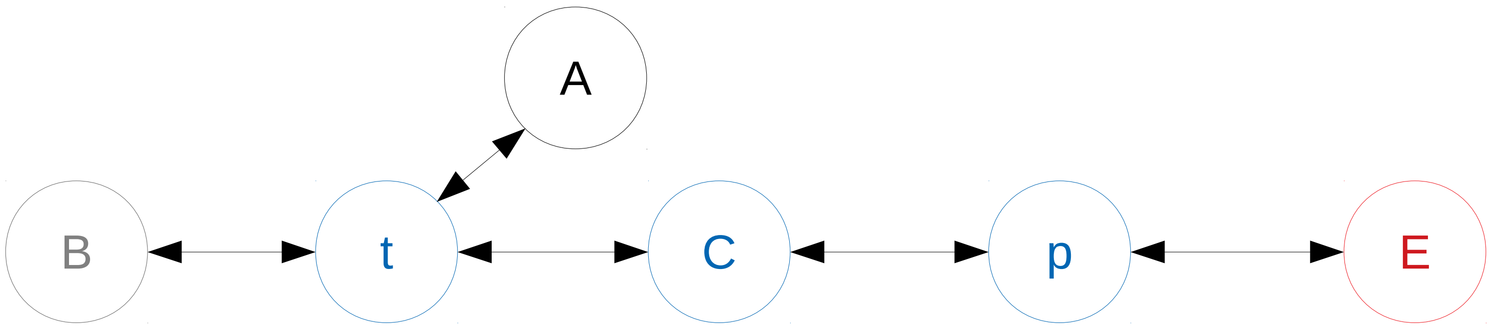


Initial state: v=B

Goal: v=E

(Enhanced) Logistics Example – Landmark Identification from DTG

Let's consider DTG_v (where v represents a position of the package)



Initial state: v=B

Goal: v=E

Identified landmarks: v=t, v=C, v=p

How to use Landmarks ?

- Assume that we constructed a landmark graph in a preprocessing phase
- Intuitively, landmarks can be used as subgoals (according to their ordering)
  - works well in the Logistic example
  - recall Sussman anomaly (not so good)
  - prone to dead-ends
- For **heuristics**

Landmark Heuristics

Landmark Heuristic

- The landmarks that have yet to be achieved after reaching a state s via a sequence of actions π

$$L(s,\pi)=|(L \setminus Accepted(s,\pi)) \cup ReqAgain(s,\pi)|$$

- L is the set of **all discovered (fact) landmarks**
- $Accepted(s,\pi) \subseteq L$ is the set of **accepted** landmarks
- $ReqAgain(s,\pi) \subseteq Accepted(s,\pi)$ is the set of accepted landmarks that have to be **achieved again**

Accepted Landmarks

- A landmark p is accepted wrt s and π if
  - p becomes true in s
  - all predecessors of p (in the landmark graph) have been accepted
- Once a landmark is accepted, it remains accepted

Required Again Landmarks

- A landmark p is required again wrt s and π if at least one of the following holds
  - p is false in s while being a goal (*false goal*)
  - p is false in s while being a greedy-necessary predecessors of some unaccepted landmark (*open-prerequisite*)

Multi-path Dependence

- Assume that a state s was achieved by two sequences of actions π_1 and π_2 such that
  - π_1 achieved a landmark p while π_2 did not
  - do we need to achieve p after s ?

Multi-path Dependence

- Assume that a state s was achieved by two sequences of actions π_1 and π_2 such that
  - π_1 achieved a landmark p while π_2 did not
  - do we need to achieve p after s ?
    - Yes, because p has to become true at some point in **all** plans (including those starting with π_2)

Landmark Heuristic

- Introduced in the well known LAMA planner (LAMA won IPC 2008 and 2011)
  - One component of LAMA
- **Inadmissible**
  - because a single action can achieve multiple landmarks
- Can be very informative in some domains
  - recall our Logistics example

LM-Cut Heuristic

i-g form of Relaxed Planning Tasks

- A relaxed planning task (P,A,i,g) is in **i-g form** if
  - $i,g \in P$
  - every action has at least one precondition
  - convention: an i-g form action will be represented in form $a=(pre(a) \rightarrow add(a))_{c(a)}$
- How "normal" relaxed planning tasks can be converted to i-g form ?

i-g form of Relaxed Planning Tasks

- A relaxed planning task (P,A,i,g) is in **i-g form** if
  - $i,g \in P$
  - every action has at least one precondition
  - convention: an i-g form action will be represented in form $a=(pre(a) \rightarrow add(a))_{c(a)}$
- How “normal” relaxed planning tasks can be converted to i-g form ?
- Introducing **initial and goal actions**, i.e., $a_I=(i \rightarrow I)_0$ and $a_G=(G \rightarrow g)_0$
- Actions with empty preconditions will get i into their preconditions

Justification Graph

- A **precondition choice function (pcf)** $X:A \to P$ for a relaxed planning task in i-g form (P,A,i,g) maps each action to one of its preconditions, i.e., $X(a) \in pre(a)$ for each $a \in A$

- Let X be pcf for (P,A,i,g). The **justification graph** for X is the directed edge-labeled graph J=(V,E), where
  - V=P (vertices are atoms from P)
  - For each $a \in A$ and $p \in add(a)$, $(X(a),a,p) \in E$

Example

$a_1 = (i \rightarrow x, y)_3$

$a_2 = (i \rightarrow x, z)_4$

$a_3 = (i \rightarrow y, z)_5$

$a_4 = (x, y, z \rightarrow g)_0$

Example – Justification Graph

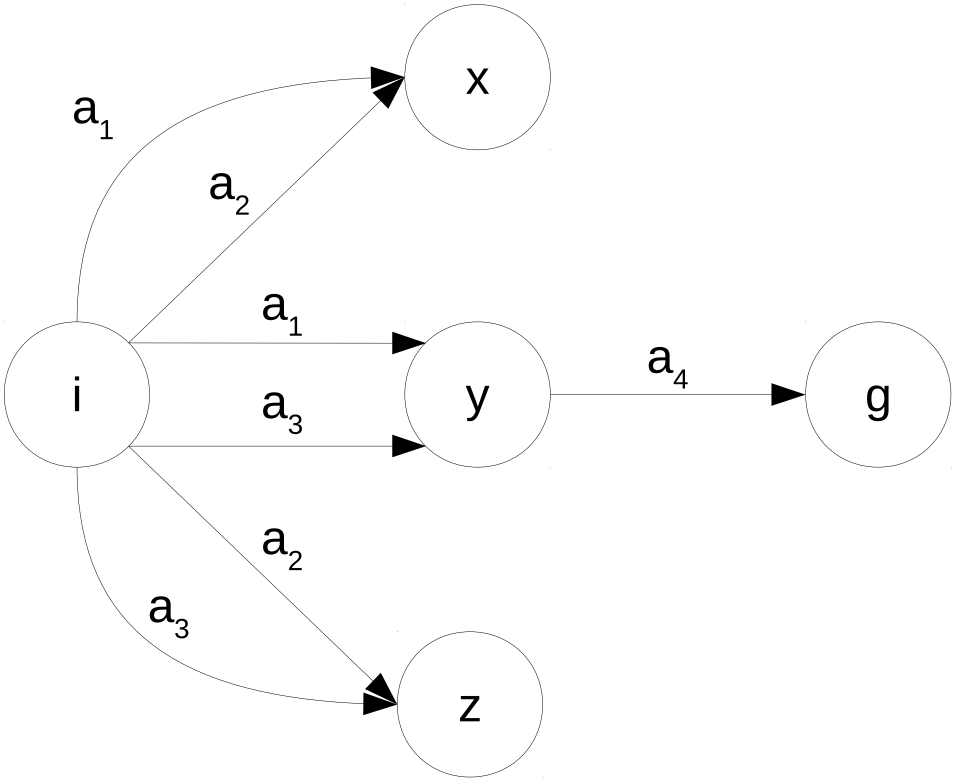
$a_1=(i \rightarrow x,y)_3$

$a_2=(i \rightarrow x,z)_4$

$a_3=(i \rightarrow y,z)_5$

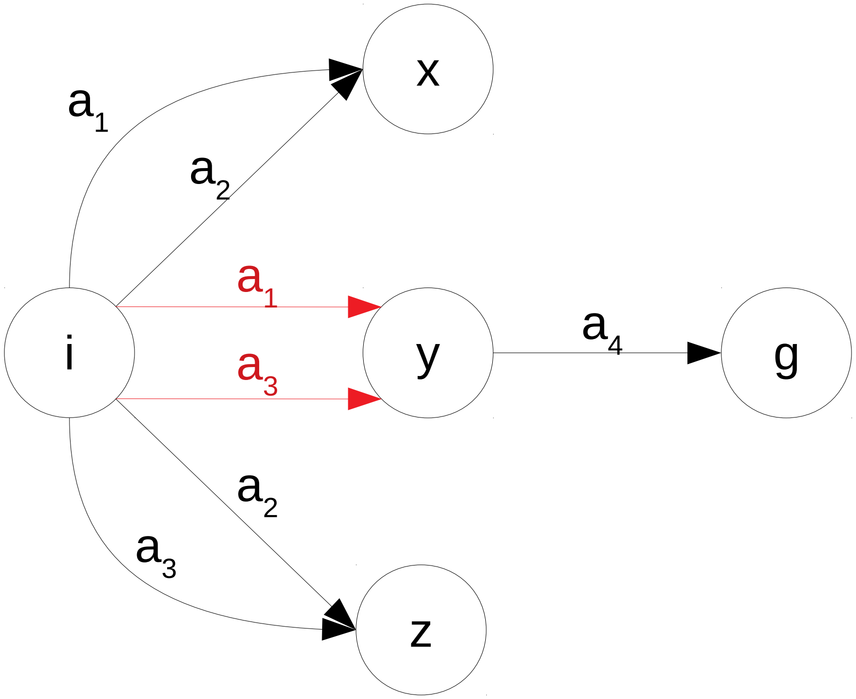
$a_4=(x,y,z \rightarrow g)_0$

pcf in red



Cuts

- A **cut C** in a justification graph is a subset of its edges such that all paths from i to g contain an edge from C



Disjunctive Action Landmarks

Theorem: Let C be a cut in the justification graph for pcf X. The set of edge-labels from C is a **disjunctive action landmark**

- Note that the justification graph represents a simpler problem (only one action precondition is considered)
- Cuts are disjunctive action landmarks for the simplified problem and thus also for the original problem
- With all “cut landmarks” we can compute the value of h^+
  - However, the number of pcfs is exponential

LM-Cut

- Set $h^{LM\text{-}Cut}(I)=0$, then iterate

1) Compute $\mathbf{h^{max}}$ for all atoms. If $h^{max}(g)=0$, terminate

2) Let X be a pcf choosing preconditions with **maximal $\mathbf{h^{max}}$ value**

3) Compute the **justification graph** for X

4) Compute a **cut** L such that **cost(L)>0** (details on the next slide)

5) $h^{LM\text{-}Cut}(I)$+=cost(L)

6) For each action $a \in L$, c(a)=c(a) - cost(L)

LM-Cut

- Compute a cut L such that cost(L)>0 as follows
  - The **goal zone** V_g of the justification graph consists of all vertices having a path to g with all edges (on that path) having zero-cost actions
  - The cut contains all edges (v,a,v') such that $v \notin V_g$ and $v' \in V_g$ and v can be reached from I without traversing a goal zone node
  - $cost(L)=min_{a \in L} c(a)$

Example – Computing LM-cut

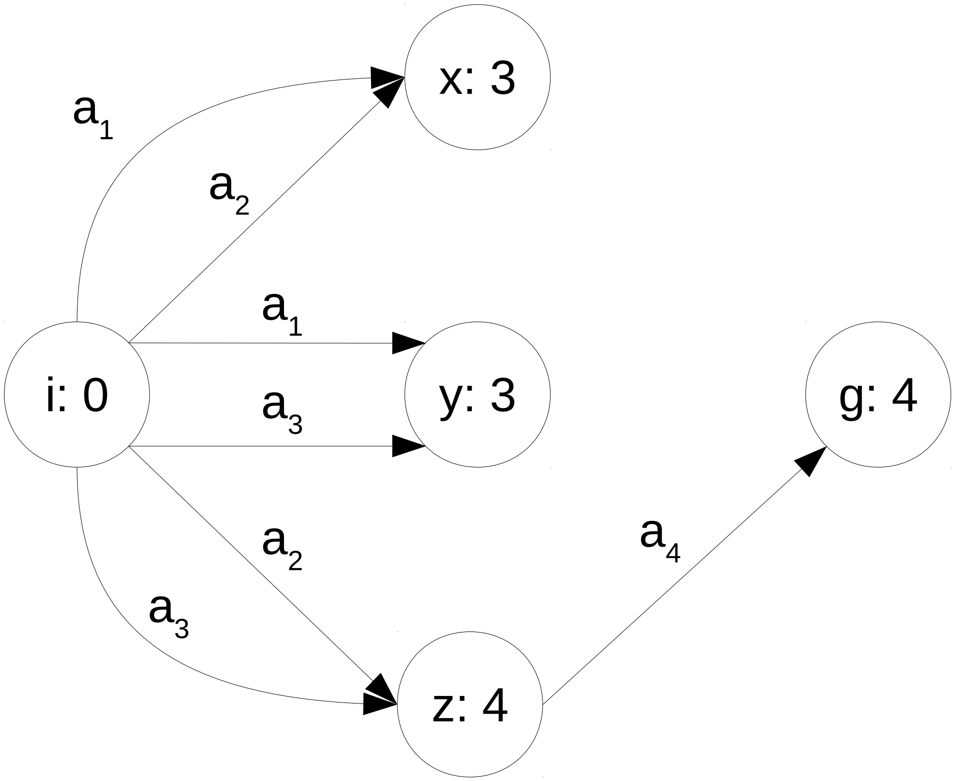
a_1=(i → x,y)$_3$

a_2=(i → x,z)$_4$

a_3=(i → y,z)$_5$

a_4=(x,y,z → g)$_0$

pcf in red



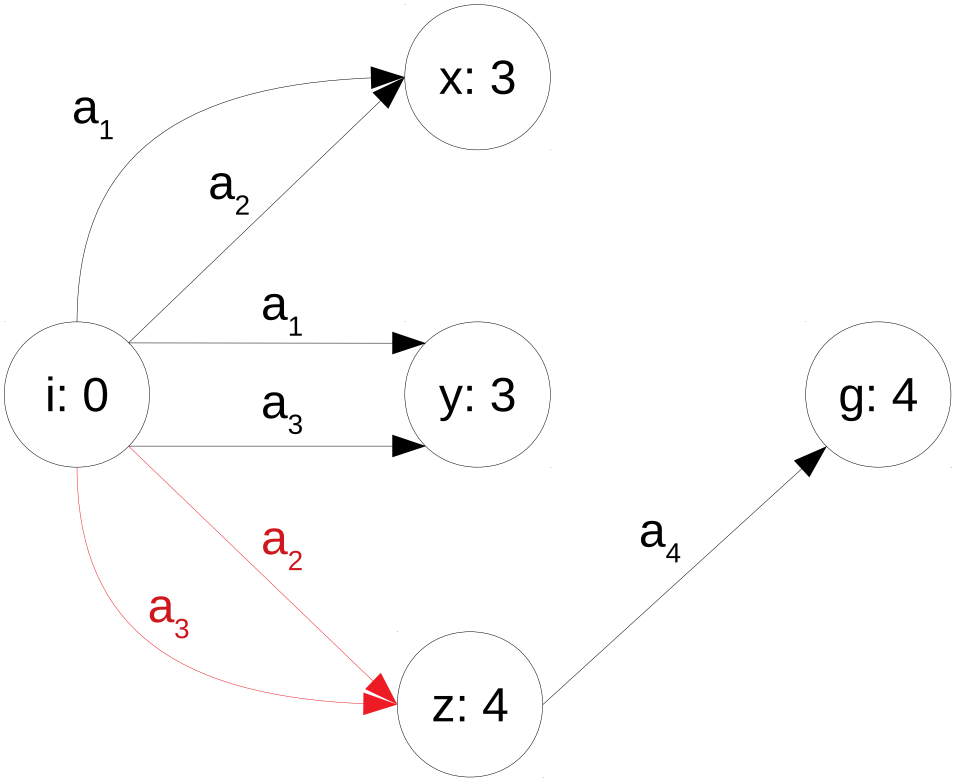
Example – Computing LM-cut

$a_1 = (i \rightarrow x,y)_3$

$a_2 = (i \rightarrow x,z)_4$

$a_3 = (i \rightarrow y,z)_5$

$a_4 = (x,y,z \rightarrow g)_0$



pcf in red

$L = \{a_2, a_3\}$

$cost(L) = 4$

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

Example – Computing LM-cut

$a_1 = (i \rightarrow x,y)_3$

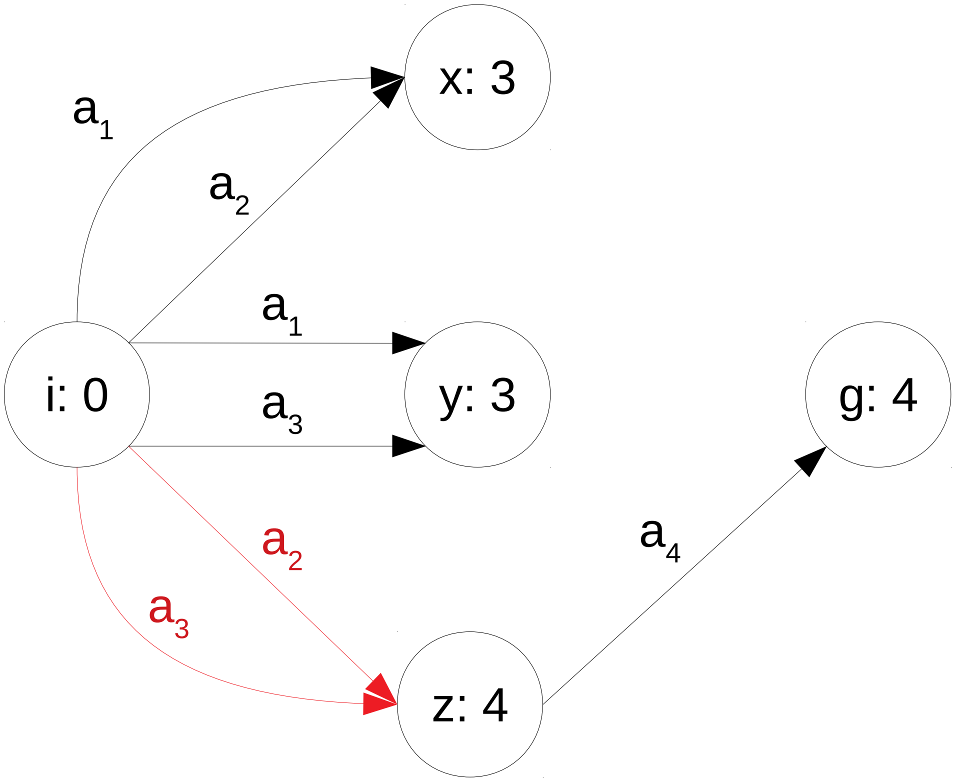
$a_2 = (i \rightarrow x,z)_0$

$a_3 = (i \rightarrow y,z)_1$

$a_4 = (x,y,z \rightarrow g)_0$

pcf in red

$L = \{a_2, a_3\}$
$cost(L) = 4$
$h^{LM\text{-}cut}(I) = 4$



Example – Computing LM-cut

$a_1 = (i \rightarrow x, y)_3$

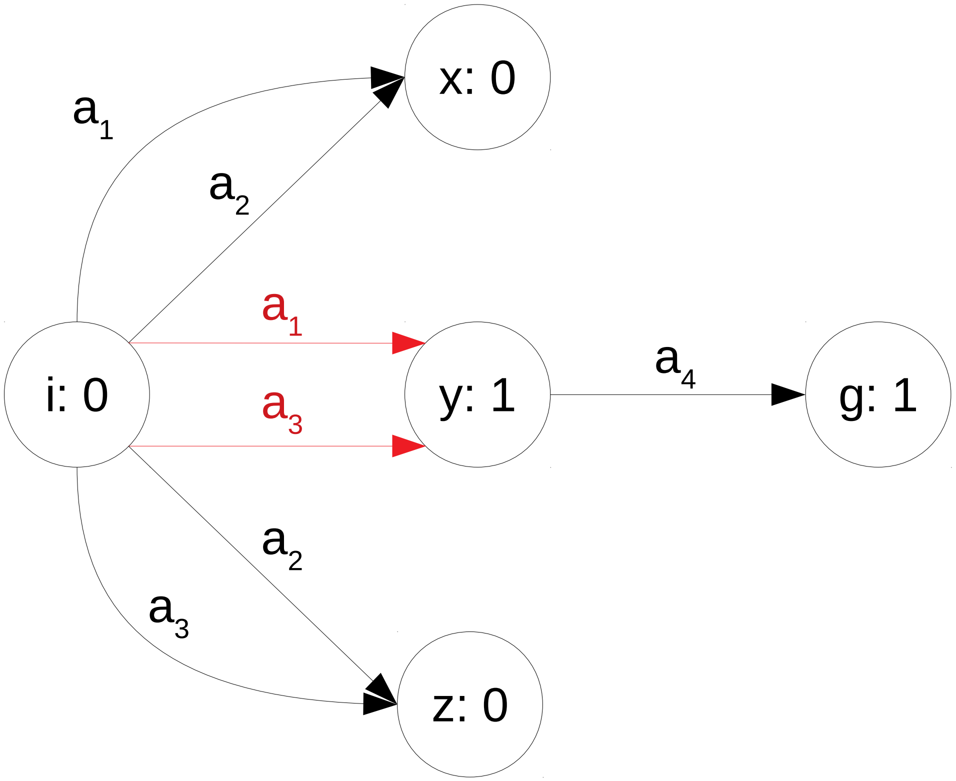
$a_2 = (i \rightarrow x, z)_0$

$a_3 = (i \rightarrow y, z)_1$

$a_4 = (x, y, z \rightarrow g)_0$

pcf in red

$L = \{a_1, a_3\}$
$cost(L) = 1$
$h^{LM\text{-}cut}(I) = 5$



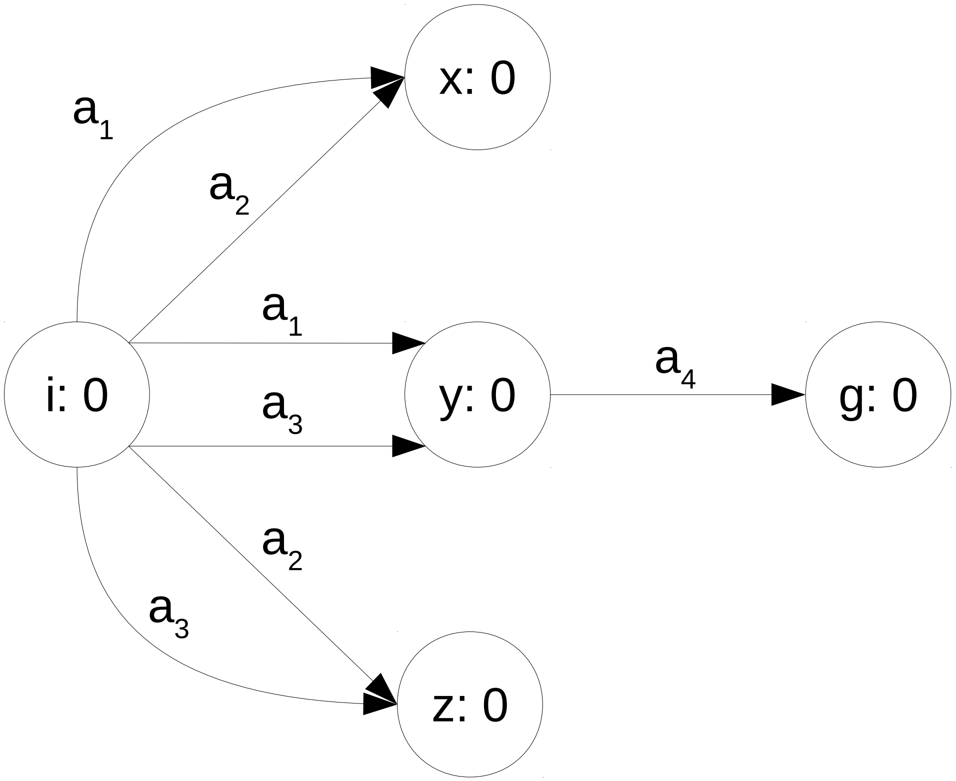
Example – Computing LM-cut

$a_1=(i \rightarrow x,y)_2$

$a_2=(i \rightarrow x,z)_0$

$a_3=(i \rightarrow y,z)_0$

$a_4=(x,y,z \rightarrow g)_0$



pcf in red

$h^{max}(g)=0 \rightarrow$ done !

$\mathbf{h^{LM\text{-}cut}(I)=5}$

LM-cut – Final Remarks

- LM-cut finds (some) **disjunctive action landmarks**
- It can be proven that $h^{LM\text{-}cut} \leq h^{+}$
- LM-cut heuristic is thus **admissible**

- LM-cut heuristic extracts landmarks for each (visited) state
- Other methods extracts landmarks once and then propagate them over the course of the search