Foundations of Artificial Intelligence

36. Automated Planning: Delete Relaxation Heuristics

Malte Helmert and Gabriele Röger

University of Basel

May 8, 2017

Foundations of Artificial Intelligence

May 8, 2017 — 36. Automated Planning: Delete Relaxation Heuristics

Automated Planning: Overview

Chapter overview: planning

- 33. Introduction
- 34. Planning Formalisms
- 35.–36. Planning Heuristics: Delete Relaxation
  - 35. Delete Relaxation
  - 36. Delete Relaxation Heuristics
- 37. Planning Heuristics: Abstraction
- 38.–39. Planning Heuristics: Landmarks

36.1 Relaxed Planning Graphs

Relaxed Planning Graphs

- relaxed planning graphs: represent which variables in Π^+ can be reached and how
- graphs with variable layers V^i and action layers A^i
  - variable layer V^0 contains the variable vertex v^0 for all $v \in I$
  - action layer A^{i+1} contains the action vertex a^{i+1} for action a if V^i contains the vertex v^i for all $v \in pre(a)$
  - variable layer V^{i+1} contains the variable vertex v^{i+1} if previous variable layer contains v^i, or previous action layer contains a^{i+1} with $v \in add(a)$

German: relaxierter Planungsgraph, Variablenknoten, Aktionsknoten

Relaxed Planning Graphs (Continued)

- goal vertices G^i if $v^i \in V^i$ for all $v \in G$
- graph can be constructed for arbitrary many layers but stabilizes after a bounded number of layers $\rightsquigarrow V^{i+1} = V^i$ and $A^{i+1} = A^i$ (Why?)
- directed edges:
  - from v^i to a^{i+1} if $v \in pre(a)$ (precondition edges)
  - from a^i to v^i if $v \in add(a)$ (effect edges)
  - from v^i to G^i if $v \in G$ (goal edges)
  - from v^i to v^{i+1} (no-op edges)

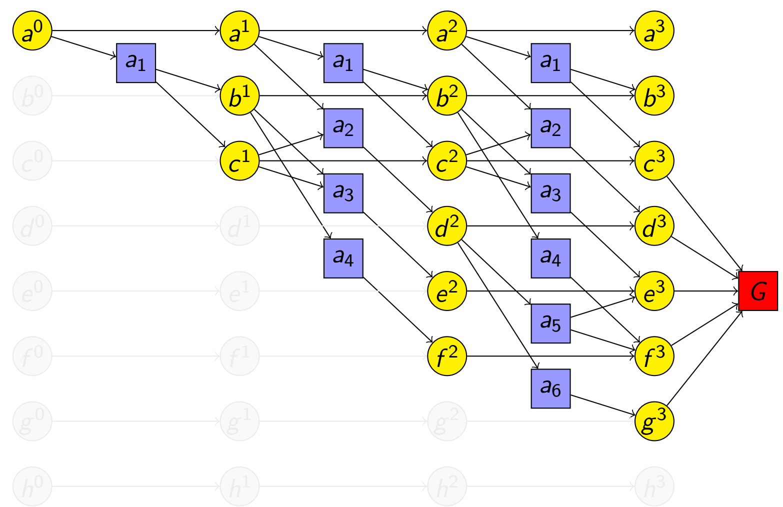
German: Zielknoten, Vorbedingungskanten, Effektkanten, Zielkanten, No-Op-Kanten

Illustrative Example

We will write actions a with $pre(a) = \{p_1, \dots, p_k\}$, $add(a) = \{a_1, \dots, a_l\}$, $del(a) = \emptyset$ and $cost(a) = c$ as $\langle p_1, \dots, p_k \to a_1, \dots, a_l \rangle_c$

$$V = \{a, b, c, d, e, f, g, h\}$$
$$I = \{a\}$$
$$G = \{c, d, e, f, g\}$$
$$A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$$
$$a_1 = \langle a \to b, c \rangle_3$$
$$a_2 = \langle a, c \to d \rangle_1$$
$$a_3 = \langle b, c \to e \rangle_1$$
$$a_4 = \langle b \to f \rangle_1$$
$$a_5 = \langle d \to e, f \rangle_1$$
$$a_6 = \langle d \to g \rangle_1$$

Illustrative Example: Relaxed Planning Graph



Generic Relaxed Planning Graph Heuristic

Heuristic Values from Relaxed Planning Graph

function *generic-rpg-heuristic*($\langle V, I, G, A\rangle, s$):
    $\Pi^+ := \langle V, s, G, A^+\rangle$
    for $k \in \{0, 1, 2, \dots\}$:
        $rpg := RPG_k(\Pi^+)$ [relaxed planning graph to layer k]
        if *rpg* contains a goal node:
            Annotate nodes of *rpg*.
            if termination criterion is true:
                return heuristic value from annotations
        else if graph has stabilized:
            return ∞

$\rightsquigarrow$ general template for RPG heuristics

$\rightsquigarrow$ to obtain concrete heuristic: instantiate highlighted elements

Concrete Examples for Generic RPG Heuristic

Many planning heuristics fit this general template.

In this course:

- maximum heuristic $h^{\max}$ (Bonet & Geffner, 1999)
- additive heuristic h^{add} (Bonet, Loerincs & Geffner, 1997)
- Keyder & Geffner's (2008) variant of the FF heuristic h^{FF} (Hoffmann & Nebel, 2001)

German: Maximum-Heuristik, additive Heuristik, FF-Heuristik

remark:

- The most efficient implementations of these heuristics do not use explicit planning graphs, but rather alternative (equivalent) definitions.

36.2 Maximum and Additive Heuristics

Maximum and Additive Heuristics

- $h^{\max}$ and h^{add} are the simplest RPG heuristics.
- Vertex annotations are numerical values.
- The vertex values estimate the costs
  - to make a given variable true
  - to reach and apply a given action
  - to reach the goal

Maximum and Additive Heuristics: Filled-in Template

$h^{\max}$ and h^{add}

computation of annotations:

- costs of variable vertices:
  0 in layer 0;
  otherwise minimum of the costs of predecessor vertices
- costs of action and goal vertices:
  maximum ($h^{\max}$) or sum (h^{add}) of predecessor vertex costs;
  for action vertices a^i, also add $cost(a)$

termination criterion:

- stability: terminate if $V^i = V^{i-1}$ and costs of all vertices in V^i equal corresponding vertex costs in V^{i-1}

heuristic value:

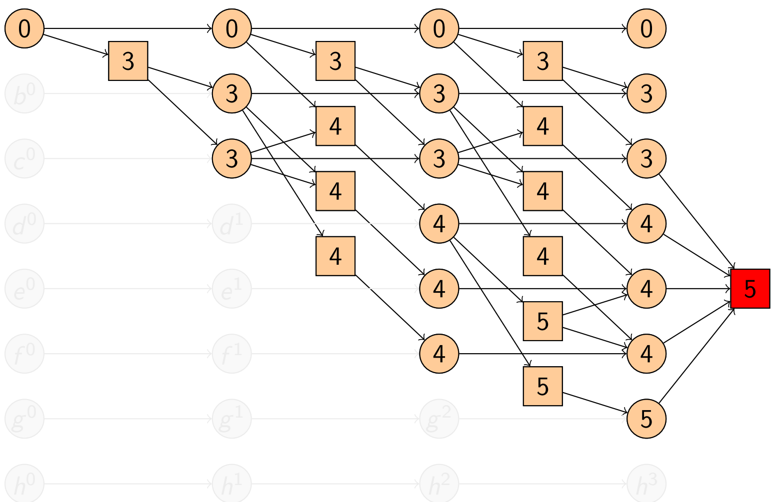
- value of goal vertex in the last layer

Maximum and Additive Heuristics: Intuition

intuition:

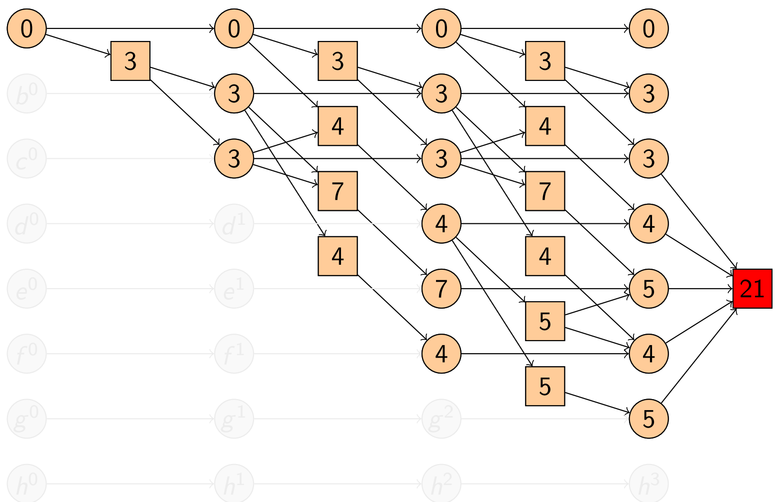
- variable vertices:
  - choose cheapest way of reaching the variable
- action/goal vertices:
  - $h^{\max}$ is optimistic: assumption:
    when reaching the most expensive precondition variable, we can reach the other precondition variables in parallel (hence maximization of costs)
  - h^{add} is pessimistic: assumption:
    all precondition variables must be reached completely independently of each other (hence summation of costs)

Illustrative Example: $h^{\max}$



$$h^{\max}(\{a\}) = 5$$

Illustrative Example: h^{add}



$$h^{\text{add}}(\{a\}) = 21$$

$h^{\max}$ and h^{add}: Remarks

comparison of $h^{\max}$ and h^{add}:

- both are safe and goal-aware
- $h^{\max}$ is admissible and consistent; h^{add} is neither.

$\rightsquigarrow$ h^{add} not suited for optimal planning

- However, h^{add} is usually much more informative than $h^{\max}$. Greedy best-first search with h^{add} is a decent algorithm.
- Apart from not being admissible, h^{add} often vastly overestimates the actual costs because positive synergies between subgoals are not recognized.

$\rightsquigarrow$ FF heuristic

36.3 FF Heuristic

FF Heuristic

The FF Heuristic

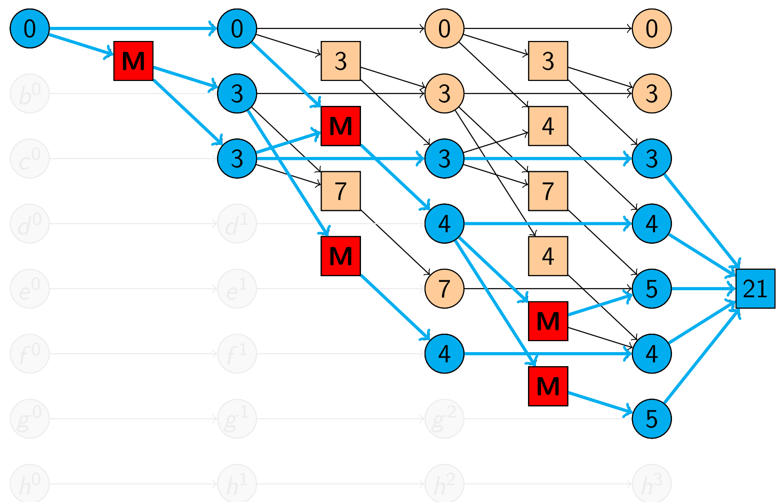
identical to h^{add}, but additional steps at the end:

- Mark goal vertex in the last graph layer.
- Apply the following marking rules until nothing more to do:
  - marked action or goal vertex?
    $\rightsquigarrow$ mark all predecessors
  - marked variable vertex v^i in layer $i \geq 1$?
    $\rightsquigarrow$ mark one predecessor with minimal h^{add} value
    (tie-breaking: prefer variable vertices; otherwise arbitrary)

heuristic value:

- The actions corresponding to the marked action vertices build a relaxed plan.
- The cost of this plan is the heuristic value.

Illustrative Example: h^{FF}



$$h^{\text{FF}}(\{a\}) = 3 + 1 + 1 + 1 + 1 = 7$$

FF Heuristic: Remarks

- Like h^{add}, h^{FF} is safe and goal-aware, but neither admissible nor consistent.
- approximation of h^+ which is always at least as good as h^{add}
- usually significantly better
- can be computed in linear time in the size of the description of the planning task
- computation of heuristic value depends on tie-breaking of marking rules (h^{FF} not well-defined)
- one of the most successful planning heuristics

Comparison of Relaxation Heuristics

Relationships of Relaxation Heuristics

Let s be a state in the STRIPS planning task $\langle V, I, G, A\rangle$.

Then

- $h^{\max}(s) \leq h^+(s) \leq h^*(s)$
- $h^{\max}(s) \leq h^+(s) \leq h^{\text{FF}}(s) \leq h^{\text{add}}(s)$
- h^* and h^{FF} are incomparable
- h^* and h^{add} are incomparable

further remarks:

- For non-admissible heuristics, it is generally neither good nor bad to compute higher values than another heuristic.
- For relaxation heuristics, the objective is to approximate h^+ as closely as possible.

36.4 Summary

Summary

- Many delete relaxation heuristics can be viewed as computations on relaxed planning graphs (RPGs).
- examples: $h^{\max}$, h^{add}, h^{FF}
- $h^{\max}$ and h^{add} propagate numeric values in the RPGs
  - difference: $h^{\max}$ computes the maximum of predecessor costs for action and goal vertices; h^{add} computes the sum
- h^{FF} marks vertices and sums the costs of marked action vertices.
- generally: $h^{\max}(s) \leq h^+(s) \leq h^{\text{FF}}(s) \leq h^{\text{add}}(s)$