Planning and Optimization

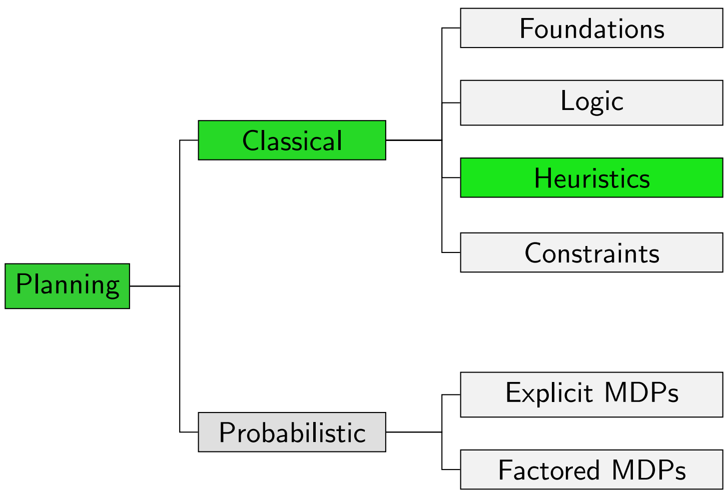
C6. Delete Relaxation: Best Achievers, h^{FF} and Comparison

Malte Helmert and Gabriele Röger

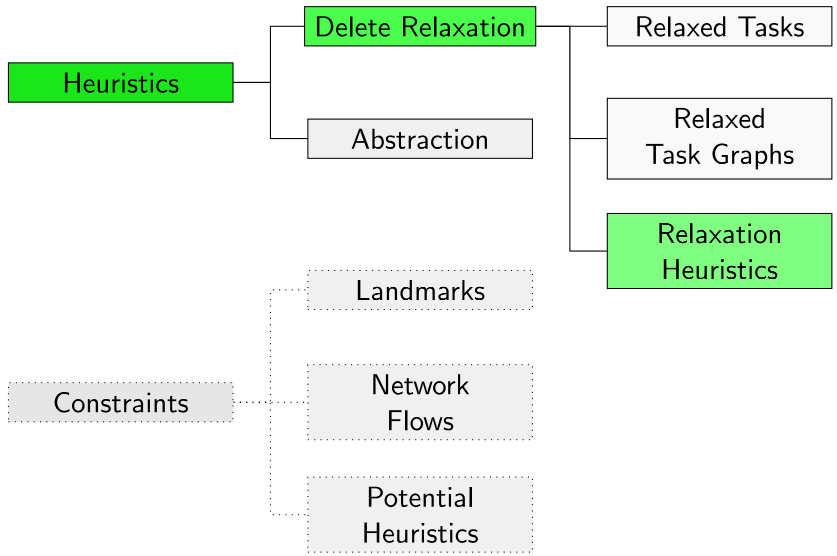
Universität Basel

Planning and Optimization
— C6. Delete Relaxation: Best Achievers, h^{FF} and Comparison

Content of this Course



Content of this Course: Heuristics



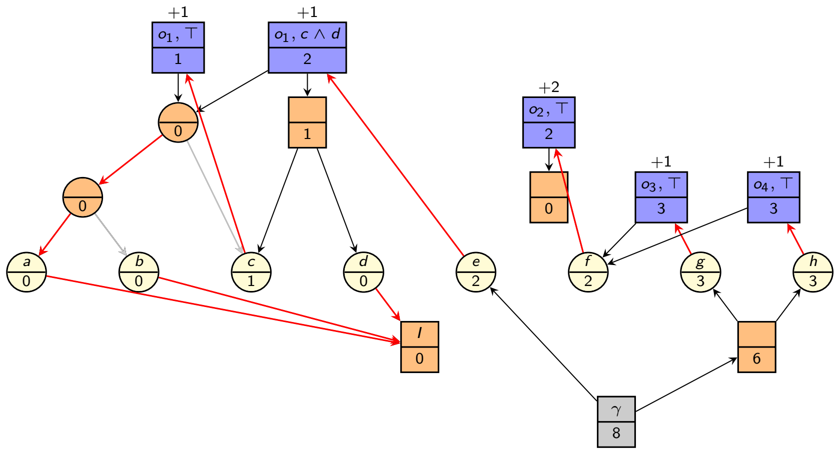
C6.1 Choice Functions

Motivation

- In this chapter, we analyze the behaviour of $h^{\max}$ and h^{add} more deeply.
- Our goal is to understand their shortcomings and use this understanding to devise an improved heuristic.
- As a preparation for our analysis, we need some further definitions that concern choices in AND/OR graphs.
- The key observation is that if we want to establish the value of a certain node n, we can to some extent choose how we want to achieve the OR nodes that are relevant to achieving n.

Preview: Choice Function & Best Achievers

Preserve at most one outgoing arc of each OR node
but node values may not change.



(precondition of o_1 modified to $c \vee (a \vee b)$)

Choice Functions

Definition (Choice Function)

Let G be an AND/OR graph with nodes N and OR nodes N_{OR}.

A choice function for G is a function $f : N' \to N$ defined on some set $N' \subseteq N_{\mathrm{OR}}$ such that $f(n) \in succ(n)$ for all $n \in N'$.

- In words, choice functions select (at most) one successor for each OR node of G.
- Intuitively, $f(n)$ selects by which disjunct n is achieved.
- If $f(n)$ is undefined for a given n, the intuition is that n is not achieved.

Reduced Graphs

Once we have decided how to achieve an OR node, we can remove the other alternatives:

Definition (Reduced Graph)

Let G be an AND/OR graph, and let f be a choice function for G defined on nodes N'.

The reduced graph for f is the subgraph of G where all outgoing arcs of OR nodes are removed except for the chosen arcs $\langle n, f(n)\rangle$ with $n \in N'$.

C6.2 Best Achievers

Choice Functions Induced by h^{max} and h^{add}

Which choices do h^{max} and h^{add} make?

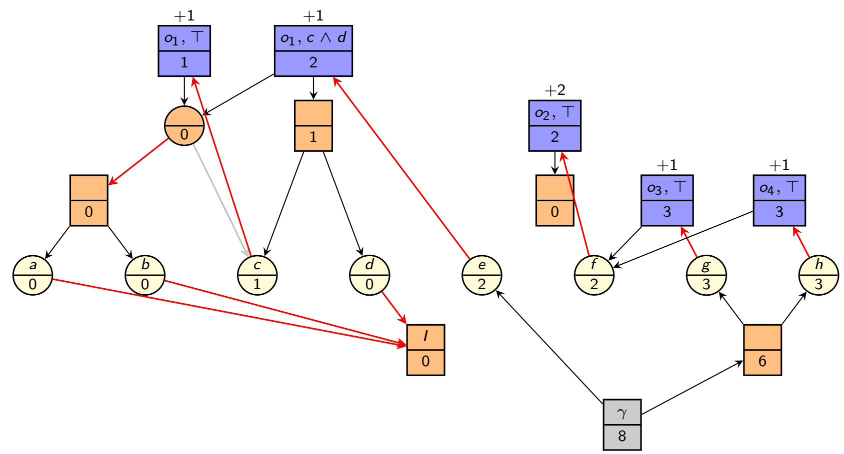
- At every OR node n, we set the cost of n to the minimum of the costs of the successors of n.
- The motivation for this is to achieve n via the successor that can be achieved most cheaply according to our cost estimates.

⇝ This corresponds to defining a choice function f with $f(n) \in \arg\min_{n' \in N'} n'.cost$ for all reached OR nodes n, where $N' \subseteq succ(n)$ are all successors of n processed before n.

- The successors chosen by this cost function are called best achievers (according to h^{max} or h^{add}).
- Note that the best achiever function f is in general not well-defined because there can be multiple minimizers. We assume that ties are broken arbitrarily.

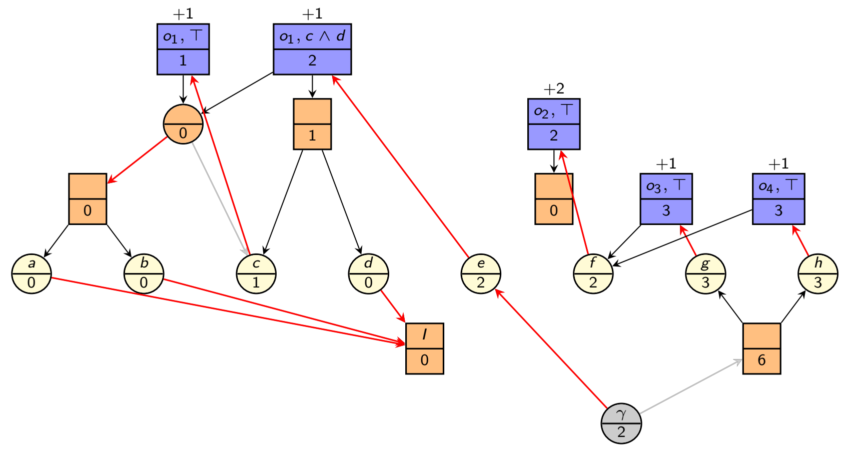
Example: Best Achievers (1)

best achievers for h^{add}



Example: Best Achievers (2)

best achievers for h^{add}; modified goal $e \vee (g \wedge h)$



Best Achiever Graphs

- Observation: The h^{max}/h^{add} costs of nodes remain the same if we replace the RTG by the reduced graph for the respective best achiever function.
- The AND/OR graph that is obtained by removing all nodes with infinite cost from this reduced graph is called the best achiever graph for h^{max}/h^{add}.
  - We write G^{max} and G^{add} for the best achiever graphs.
- G^{max} (G^{add}) is always acyclic: for all arcs $\langle n, n' \rangle$ it contains, n is processed by h^{max} (by h^{add}) after n'.

Paths in Best Achiever Graphs

Let n be a node of the best achiever graph.
Let N_{eff} be the set of effect nodes of the best achiever graph.
The cost of an effect node is the cost of the associated operator.
The cost of a path in the best achiever graph is the sum of costs of all effect nodes on the path.

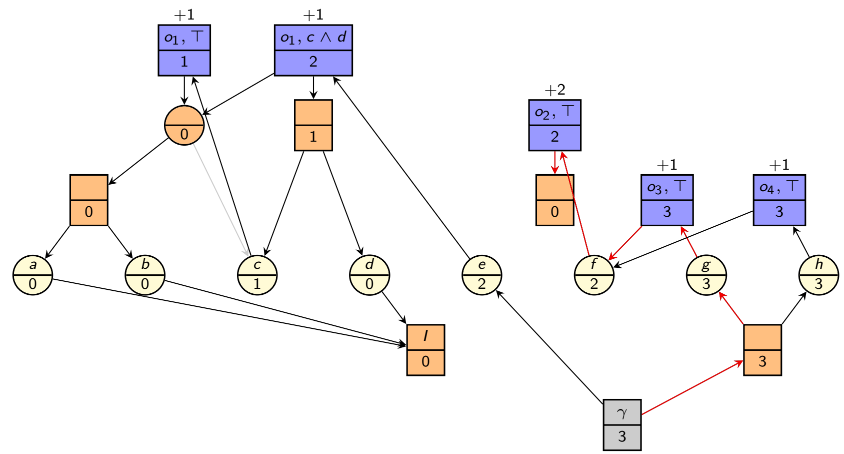
The following properties can be shown by induction:

- $h^{max}(n)$ is the maximum cost of all paths originating from n in G^{max}. A path achieving this maximum is called a critical path.
- $h^{add}(n)$ is the sum, over all effect nodes n', of the cost of n' multiplied by the number of paths from n to n' in G^{add}.

In particular, these properties hold for the goal node n_γ if it is reachable.

Example: Undercounting in h^{max}

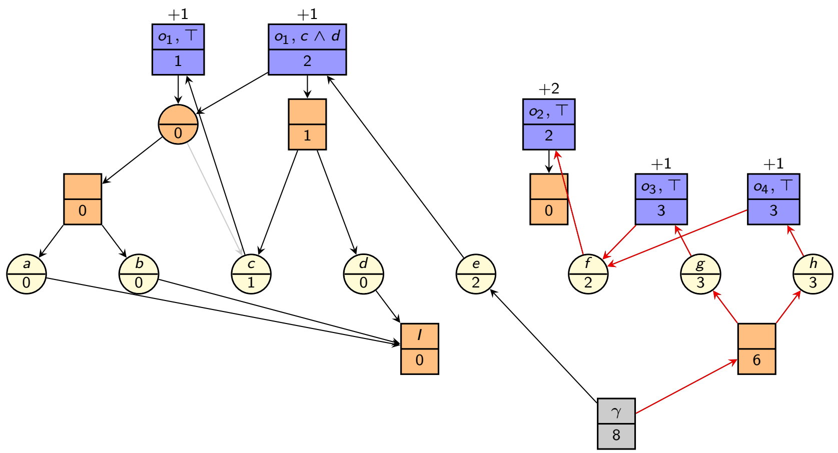
G^{max}: undercounting in h^{max}



$\rightsquigarrow$ o_1 and o_4 not counted because they are off the critical path

Example: Overcounting in h^{add}

G^{add}: overcounting in h^{add}



$\rightsquigarrow$ o_2 counted twice because there are two paths to $n_{o_2}^{\top}$

C6.3 The FF Heuristic

Inaccuracies in h^{max} and h^{add}

- h^{max} is often inaccurate because it undercounts: the heuristic estimate only reflects the cost of a critical path, which is often only a small fraction of the overall plan.
- h^{add} is often inaccurate because it overcounts: if the same subproblem is reached in many ways, it will be counted many times although it only needs to be solved once.

The FF Heuristic

Fortunately, with the perspective of best achiever graphs, there is a simple solution: count all effect nodes that h^{add} would count, but only count each of them once.

Definition (FF Heuristic)

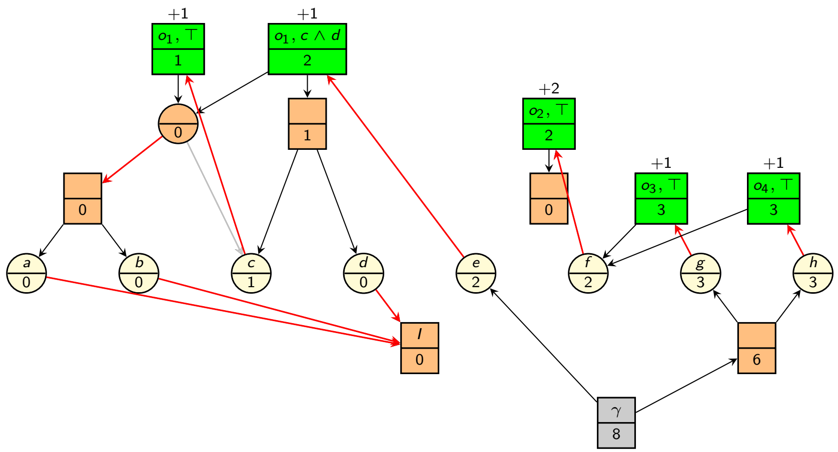
Let $\Pi = \langle V, I, O, \gamma \rangle$ be a propositional planning task in positive normal form. The FF heuristic for a state s of Π, written $h^{FF}(s)$, is computed as follows:

- Construct the RTG for the task $\langle V, s, O^+, \gamma \rangle$.
- Construct the best achiever graph G^{add}.
- Compute the set of effect nodes $\{n_{o_1}^{\chi_1}, \dots, n_{o_k}^{\chi_k}\}$ reachable from n_γ in G^{add}.
- Return $h^{FF}(s) = \sum_{i=1}^{k} cost(o_i)$.

Note: h^{FF} is not well-defined; different tie-breaking policies for best achievers can lead to different heuristic values

Example: FF Heuristic (1)

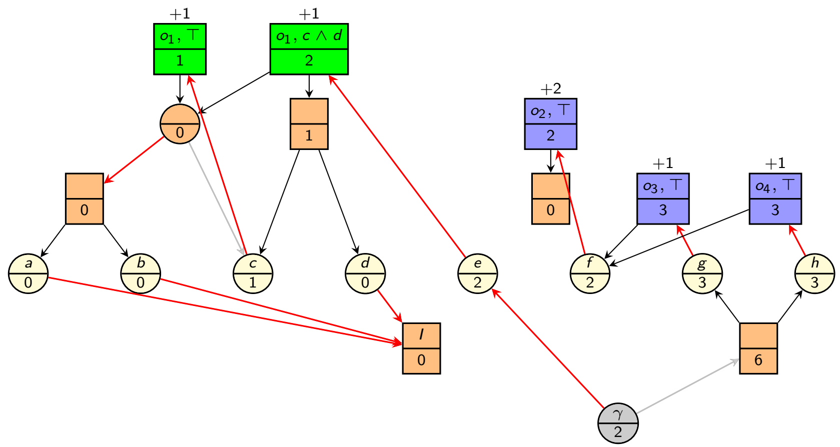
FF heuristic computation



$h^{FF}(s) = 1 + 1 + 2 + 1 + 1 = 6$

Example: FF Heuristic (2)

FF heuristic computation; modified goal $e \vee (g \wedge h)$



$h^{FF}(s) = 1 + 1 = 2$

C6.4 h^{max} vs. h^{add} vs. h^{FF} vs. h^+

Optimal Delete Relaxation Heuristic

Definition (h^+ Heuristic)

Let Π be a propositional planning task in positive normal form, and let s be a state of Π.

The optimal delete relaxation heuristic for s, written $h^+(s)$, is defined as the perfect heuristic $h^*(s)$ of state s in the delete-relaxed task Π^+.

- Reminder: We proved that $h^+(s)$ is hard to compute. (BCPlanEx is NP-complete for delete-relaxed tasks.)
- The optimal delete relaxation heuristic is often used as a reference point for comparison.

Relationships between Delete Relaxation Heuristics (1)

Theorem

Let Π be a propositional planning task in positive normal form, and let s be a state of Π.

Then:

1. $h^{max}(s) \leq h^+(s) \leq h^{FF}(s) \leq h^{add}(s)$
2. $h^{max}(s) = \infty$ *iff* $h^+(s) = \infty$ *iff* $h^{FF}(s) = \infty$ *iff* $h^{add}(s) = \infty$
3. h^{max} *and* h^+ *are admissible and consistent.*
4. h^{FF} *and* h^{add} *are neither admissible nor consistent.*
5. *All four heuristics are safe and goal-aware.*

Relationships between Delete Relaxation Heuristics (2)

Proof Sketch.

for 1:

- To show $h^{max}(s) \leq h^+(s)$, show that critical path costs can be defined for arbitrary relaxed plans and that the critical path cost of a plan is never larger than the cost of the plan. Then show that $h^{max}(s)$ computes the minimal critical path cost over all delete-relaxed plans.
- To show $h^+(s) \leq h^{FF}(s)$, prove that the operators belonging to the effect nodes counted by h^{FF} form a relaxed plan. No relaxed plan is cheaper than h^+ by definition of h^+.
- $h^{FF}(s) \leq h^{add}(s)$ is obvious from the description of h^{FF}: both heuristics count the same operators, but h^{add} may count some of them multiple times.

...

Relationships between Delete Relaxation Heuristics (3)

Proof Sketch (continued).

for 2: All of these are ∞ iff the task has no relaxed solution.

for 3: Admissibility follows from $h^{max}(s) \leq h^+(s)$ because we already know that h^+ is admissible. (We omit the argument for consistency.)

for 4: We can construct a counterexample to admissibility for h^{FF}. This also shows non-consistency because h^{FF} is goal-aware.

for 5: Goal-awareness is easy. Safety follows from 2.+3. □

C6.5 Summary

Summary

- h^{max} and h^{add} can be used to decide how to achieve OR nodes in a relaxed task graph $\rightsquigarrow$ best achievers
- Best achiever graphs help identify shortcomings of h^{max} and h^{add} compared to the perfect delete relaxation heuristic h^+.
  - h^{max} underestimates h^+ because it only considers the cost of a critical path for the relaxed planning task.
  - h^{add} overestimates h^+ because it double-counts operators occurring on multiple paths in the best achiever graph.
- The FF heuristic repairs this flaw of h^{add} and therefore approximates h^+ more closely.
- In general, $h^{max}(s) \leq h^+(s) \leq h^{FF}(s) \leq h^{add}(s)$.
- h^{max} and h^+ are admissible; h^{FF} and h^{add} are not.

Literature Pointers

(Some) delete-relaxation heuristics in the planning literature:

- additive heuristic h^{add} (Bonet, Loerincs & Geffner, 1997)
- maximum heuristic h^{max} (Bonet & Geffner, 1999)
- (original) FF heuristic (Hoffmann & Nebel, 2001)
- cost-sharing heuristic h^{cs} (Mirkis & Domshlak, 2007)
- set-additive heuristics h^{sa} (Keyder & Geffner, 2008)
- FF/additive heuristic h^{FF} (Keyder & Geffner, 2008)
- local Steiner tree heuristic h^{lst} (Keyder & Geffner, 2009)

$\rightsquigarrow$ also hybrids such as semi-relaxed heuristics and delete-relaxation landmark heuristics