

Planning and Optimization

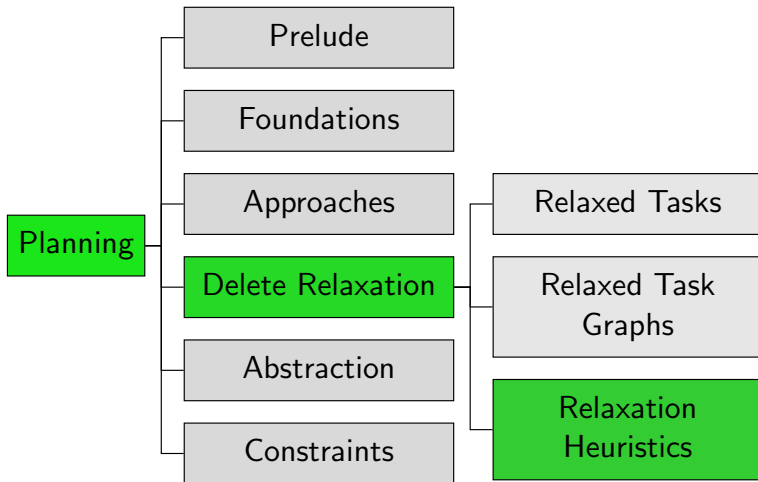
D7. Delete Relaxation: Analysis of h^{\max} and h^{add}

Malte Helmert and Gabriele Röger

Universität Basel

October 30, 2024

Content of the Course



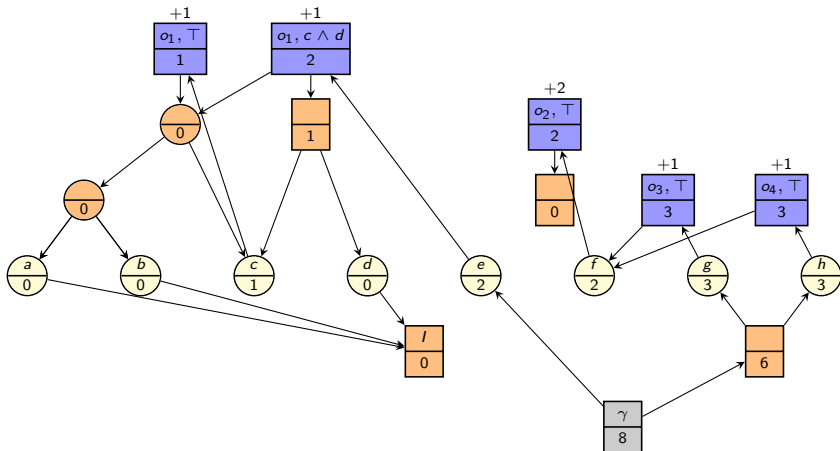
Choice Functions

Motivation

- In this chapter, we analyze the behaviour of h^{\max} and h^{add} more deeply.
- Our goal is to understand their shortcomings.
 - In the next chapter we then used 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

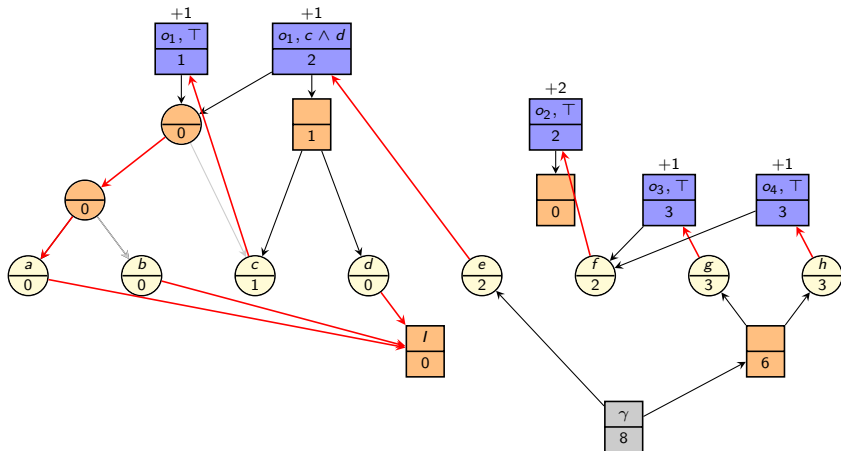
Preserve at most one outgoing arc of each OR node,
but node values may not change.



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

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_V .

A **choice function** for G is a function $f : N' \rightarrow N$ defined on some set $N' \subseteq N_V$ such that $f(n) \in \text{succ}(n)$ 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 $n \in N'$.

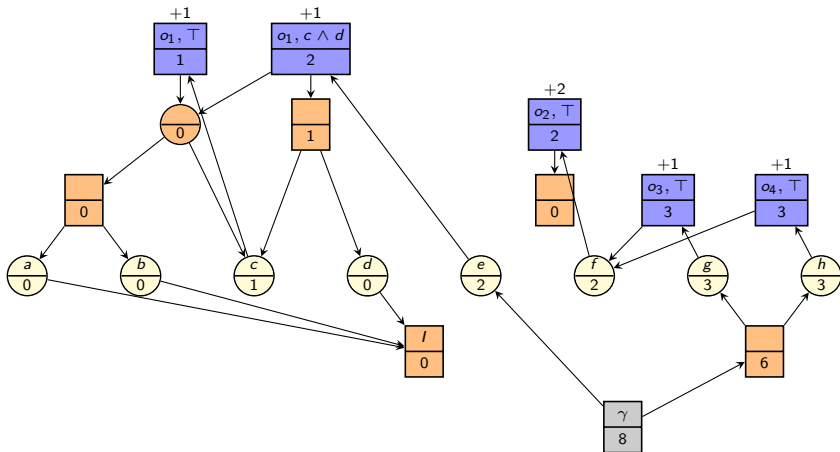
Best Achievers

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

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

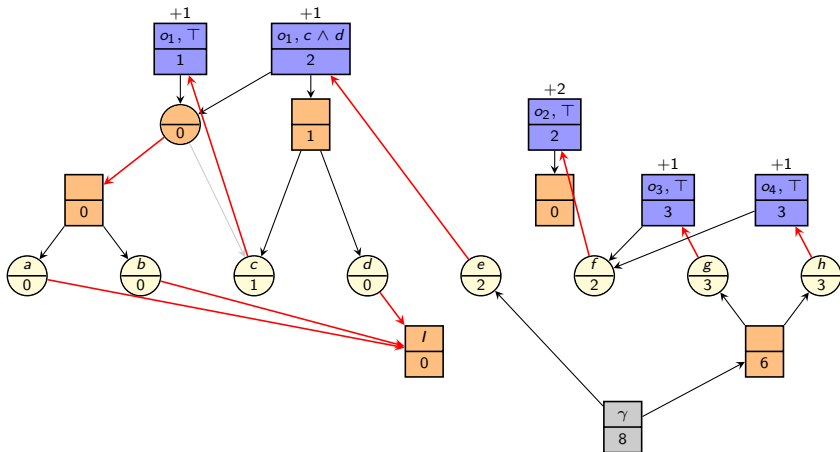
- At every OR node n , we set the cost of n to the **minimum** of the costs of the successors of n .
- The motivation for this is to achieve n via the successor that can be achieved **most cheaply** according to our cost estimates.
- ↪ This corresponds to defining a choice function f with $f(n) \in \arg \min_{n' \in N'} n'.\text{cost}$ for all reached OR nodes n , where $N' \subseteq \text{succ}(n)$ 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} 

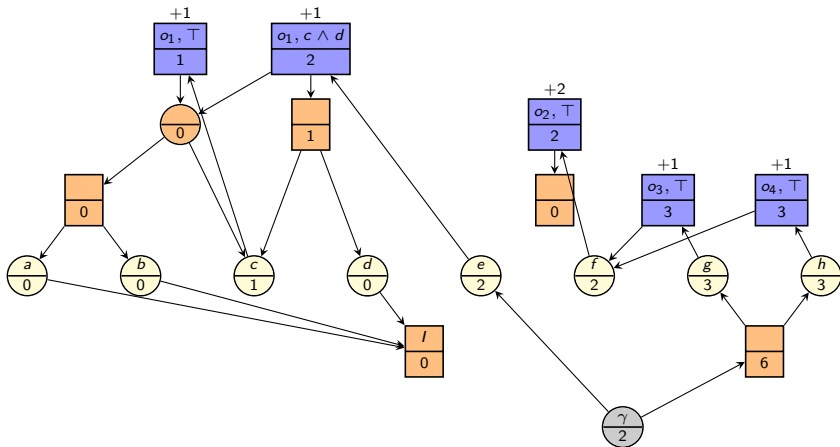
Example: Best Achievers (1)

best achievers for h^{add}

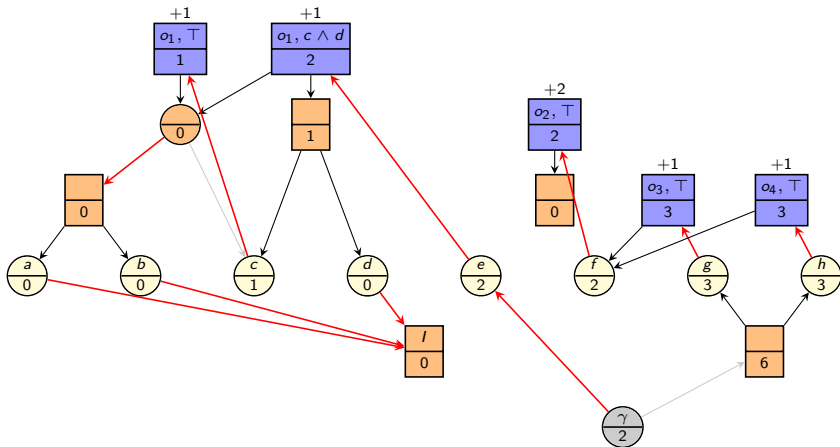


Example: Best Achievers (2)

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



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.

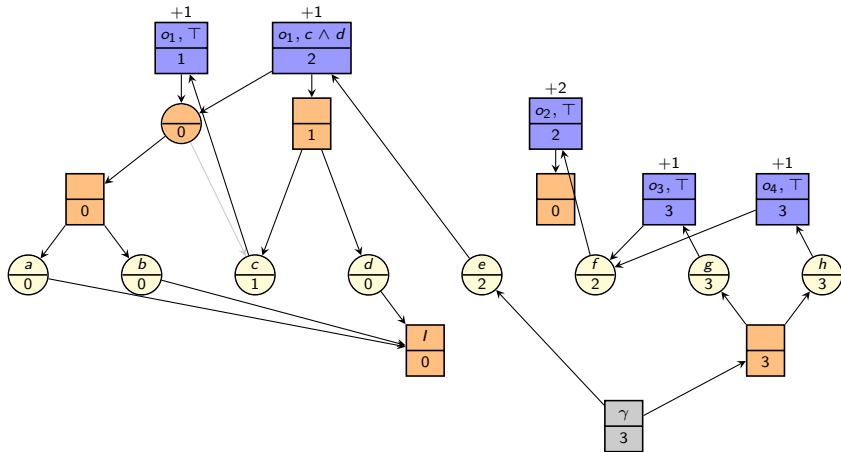
The **cost** of an **effect node** is the cost of the associated operator.

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:

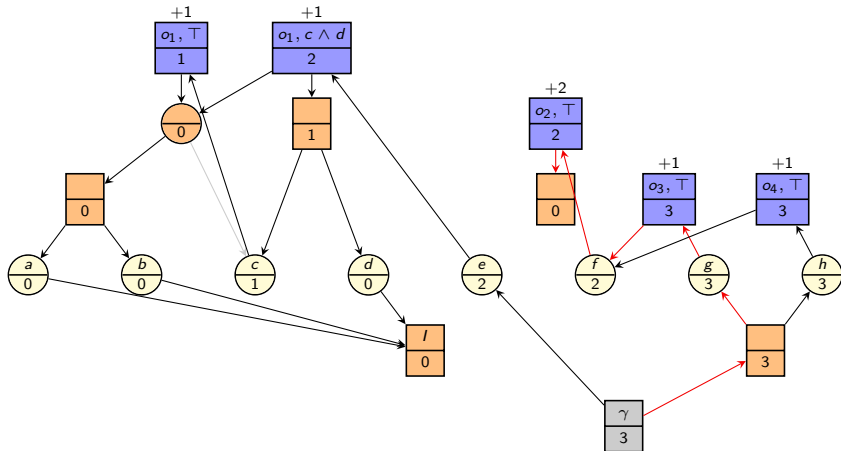
- $h^{\max}(n)$ is the **maximum cost** of all paths originating from n in G^{\max} . A path achieving this maximum is called a **critical path**.
- $h^{\text{add}}(n)$ is the **sum**, over all effect nodes n' , of the cost of n' multiplied by the **number of paths** from n to n' in G^{add} .

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

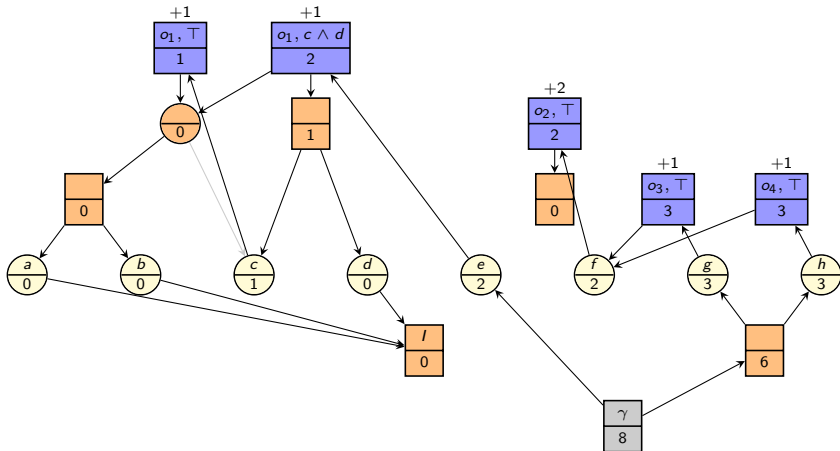
Example: Undercounting in h^{\max} G^{\max} : undercounting in h^{\max} 

Example: Undercounting in h^{\max}

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

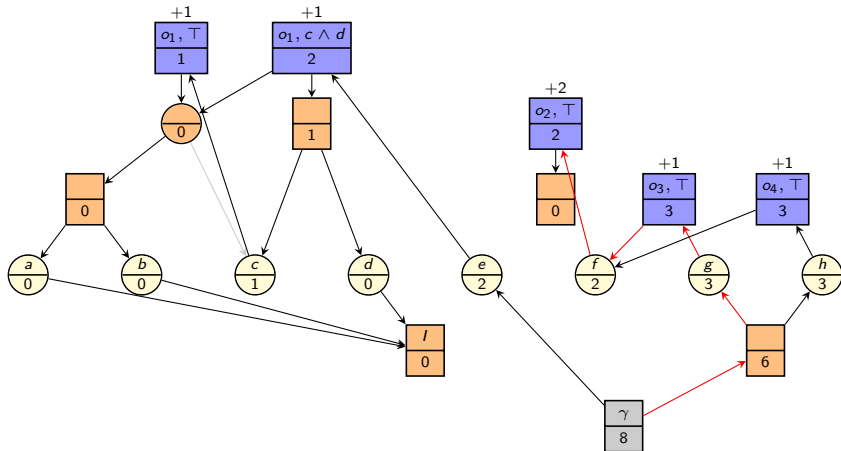


\rightsquigarrow o_1 and o_4 not counted because they are off the critical path

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

Example: Overcounting in h^{add}

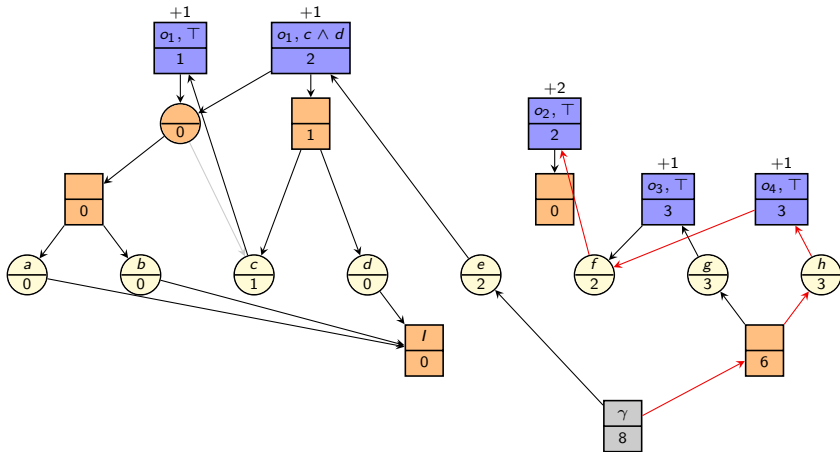
G^{add} : overcounting in h^{add}



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

Example: Overcounting in h^{add}

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



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

Summary

Summary

- h^{\max} and h^{add} can be used to decide **how** to achieve OR nodes in a relaxed task graph
~> **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.