AI Planning

9. Delete Relaxation Heuristics

It's a Long Way to the Goal, But How Long Exactly?
Part II: *Pretending Things Can Only Get Better*

Álvaro Torralba, Cosmina Croitoru

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Winter Term 2018/2019

Thanks to Prof. Jörg Hoffmann for slide sources

Agenda

We Need Heuristic Functions!

$\rightarrow$ Delete relaxation is a method to relax planning tasks, and thus automatically compute heuristic functions h.

We cover the 4 different methods currently known:

- Critical path heuristics: Done. $\rightarrow$ **Chapter 8**
- Delete relaxation: $\rightarrow$ **This Chapter, and Chapter 10**
- Abstractions: $\rightarrow$ **Chapter 11-13**
- Landmarks: $\rightarrow$ **Chapter 14**

$\rightarrow$ Each of these have advantages and disadvantages. (We will do a formal comparison in **Chapter 17**.)

$\rightarrow$ Delete relaxation is very wide-spread, and highly successful for satisficing planning! See Conclusion section and **Chapter 21**.

Pretending Things Can Only Get Better

"What was once true remains true forever."

Relaxed world: (after)

Our Agenda for This Chapter

→ Diff to AI'18: Our treatment here is more comprehensive, covering more heuristics and dealing with arbitrary action costs.

2. **The Delete Relaxation:** Gives the formal definition, and states some simple properties that immediately result in a simple "greedy" heuristic.
3. **What We Really Want is h^+:** The greedy heuristic is really bad. Ideally, what we want is h^+, only we can't actually compute it efficiently.
4. **The Additive and Max Heuristics:** Introduces the two most basic methods for computing practical delete relaxation heuristics. Explains their properties and weaknesses.
5. **The Relaxed Plan Heuristic:** Introduces a third, slightly less basic method for doing that, and explains why it addresses said weaknesses. Relaxed plans are the canonical delete relaxation heuristic, and extremely wide-spread.
6. **What about FDR Planning?** The above uses STRIPS. In this section we briefly point out that, by interpreting FDR variable/value pairs as STRIPS facts, everything remains exactly the same for FDR.

The Delete Relaxation

Definition (Delete Relaxation).

(i) *For a STRIPS action a, by a^+ we denote the corresponding delete relaxed action, or short relaxed action, defined by $pre_{a^+} := pre_a$, $add_{a^+} := add_a$, and $del_{a^+} := \emptyset$.*

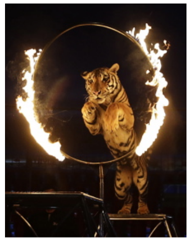
(ii) *For a set A of STRIPS actions, by A^+ we denote the corresponding set of relaxed actions, $A^+ := \{a^+ \mid a \in A\}$; similarly, for a sequence $\vec{a} = \langle a_1, \dots, a_n\rangle$ of STRIPS actions, by $\vec{a}^+$ we denote the corresponding sequence of relaxed actions, $\vec{a}^+ := \langle a_1^+, \dots, a_n^+\rangle$.*

(iii) *For a STRIPS planning task $\Pi = (P, A, c, I, G)$, by $\Pi^+ := (P, A^+, c, I, G)$ we denote the corresponding (delete) relaxed planning task.*

→ "+" super-script = delete relaxed. We'll also use this to denote states encountered within the relaxation.

Definition (Relaxed Plan). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task, and let s be a state. An (optimal) relaxed plan for s is an (optimal) plan for Π_s^+ where $\Pi_s = (P, A, c, s, G)$. A relaxed plan for I is also called a relaxed plan for Π.*

The Relaxed Dompteur



- $P = \{alive, haveTiger, tamedTiger, haveJump\}$. Short: $P = \{A, hT, tT, J\}$.
- Initial state I: $alive$.
- Goal G: $alive, haveJump$.
- Actions A:
  $getTiger$: pre $alive$; add $haveTiger$
  $tameTiger$: pre $alive$, $haveTiger$; add $tamedTiger$
  $jumpTamedTiger$: pre $alive$, $tamedTiger$; add $haveJump$
  $jumpTiger$: pre $alive$, $haveTiger$; add $haveJump$; del $alive$

→ Relaxed plan for this task? $\langle getTiger, jumpTiger\rangle$

$\langle getTiger, tameTiger, jumpTamedTiger\rangle$ works as well, but the previous one is "better" :-)

State Dominance

Definition (Dominance). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task, and let s, s' be states. We say that s' dominates s if $s' \supseteq s$.*

→ Dominance = "more facts true".

Proposition (Dominance). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task, and let s, s' be states where s' dominates s. We have:*

(i) *If s is a goal state, then s' is a goal state as well.*

(ii) *If $\vec{a}$ is applicable in s, then $\vec{a}$ is applicable in s' as well, and $s'[\![\vec{a}]\!]$ dominates $s[\![\vec{a}]\!]$.*

Proof. (i) is trivial. (ii) by induction over the length n of $\vec{a}$. Base case $n = 0$ is trivial. Inductive case $n \to n+1$ follows directly from induction hypothesis and the definition of $s[\![a]\!]$.

→ It is always better to have more facts true.

The Delete Relaxation and State Dominance

Proposition. *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. Let s be a state, and let $a \in A$ be applicable in s. Then:*

(i) $s[\![a^+]\!]$ *dominates* s.

(ii) *For any state s' that dominates s, $s'[\![a^+]\!]$ dominates $s[\![a]\!]$.*

Ergo 1: Any real plan also works in the relaxed world.

Proposition (Delete Relaxation is Over-Approximating). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task, let s be a state, and let $\vec{a}$ be a plan for Π_s. Then $\vec{a}^+$ is a relaxed plan for s.*

Proof. Prove by induction over the length of $\vec{a}$ that $s[\![\vec{a}^+]\!]$ dominates $s[\![\vec{a}]\!]$. Base case is trivial, inductive case follows from (ii) above.

Ergo 2: It is now clear how to find a relaxed plan.

- Applying a relaxed action can only ever make more facts true ((i) above).
- That cannot render the task unsolvable (proposition slide 10).

$\Longrightarrow$ So? Keep applying relaxed actions, stop if goal is true (see next slide).

Greedy Relaxed Planning

Greedy Relaxed Planning for Π_s^+

```
$s^+ := s$; $\vec{a}^+ := \langle\rangle$
while $G \not\subseteq s^+$ do:
    if $\exists a \in A$ s.t. $pre_a \subseteq s^+$ and $s^+[\![a^+]\!] \neq s^+$ /* i.e. $add_a \not\subseteq s^+$ */ then
        select one such $a$
        $s^+ := s^+[\![a^+]\!]$; $\vec{a}^+ := \vec{a}^+ \circ \langle a^+ \rangle$
    else return "$\Pi_s^+$ is unsolvable" endif
endwhile
return $\vec{a}^+$
```

Proposition. *Greedy relaxed planning is sound, complete, and terminates in time polynomial in the size of Π.*

Proof. Soundness: If $\vec{a}^+$ is returned then, by construction, $G \subseteq s[\![\vec{a}^+]\!]$. Completeness: If "$\Pi_s^+$ is unsolvable" is returned, then no relaxed plan exists for s^+ at that point. As s^+ dominates s, by the dominance proposition (slide 10), this implies that no relaxed plan can exist for s. Termination: Every $a \in A$ can be selected at most once because afterwards $s^+[\![a^+]\!] = s^+$.

$\Longrightarrow$ It is easy to decide whether a relaxed plan exists!

Questionnaire

Question!

Say the task is to drive from Saarbrücken (SB) to Moscow (M). Which of the following relaxed plans could be returned by Greedy Relaxed Planning?

(A): Take the shortest route from SB to M

(B): Drive from SB to M via Madrid

(C): Drive from SB to both Hongkong and Capetown, then from SB to M

(D): Drive to Hongkong and the same route back to SB, then from SB to M

$\rightarrow$ (A): Yes. (B): Yes, the route may be sub-optimal. (C): Yes, the "route" may contain separate "branches" (it's a tree, not a path). (D): No, because every action on the greedy relaxed plan must achieve some new fact.

$\rightarrow$ Lower/upper bounds on the heuristic value: Length of optimal route/size of a maximal tree rooted at SB and spanning the entire map.

Greedy Relaxed Planning to Generate a Heuristic Function?

Using greedy relaxed planning to generate h

- In search state s during forward search, run greedy relaxed planning on Π_s^+.
- Set $h(s)$ to the cost of $\vec{a}^+$, or ∞ if "Π_s^+ is unsolvable" is returned.

$\rightarrow$ Is this h accurate? NO! Greedy relaxed planning may select arbitrary actions that aren't relevant at all, over-estimating dramatically (cf. previous slide).

$\rightarrow$ To be accurate, a heuristic needs to approximate the *minimum effort* needed to reach the goal.

- When we talk about "the distance to Moscow", we don't mean "via Madrid" ...
- There also is an issue of "brittleness": Greedy relaxed planning may give drastically different values for very similar states (and even for the same state if called twice). This is bound to be detrimental for search guidance.
- To the rescue: h^+.

h^+: The Optimal Delete Relaxation Heuristic

Definition (h^+). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task with state space $\Theta_\Pi = (S, A, c, T, I, G)$. The optimal delete relaxation heuristic h^+ for Π is the function $h^+ : S \mapsto \mathbb{R}_0^+ \cup \{\infty\}$ where $h^+(s)$ is defined as the cost of an optimal relaxed plan for s.*

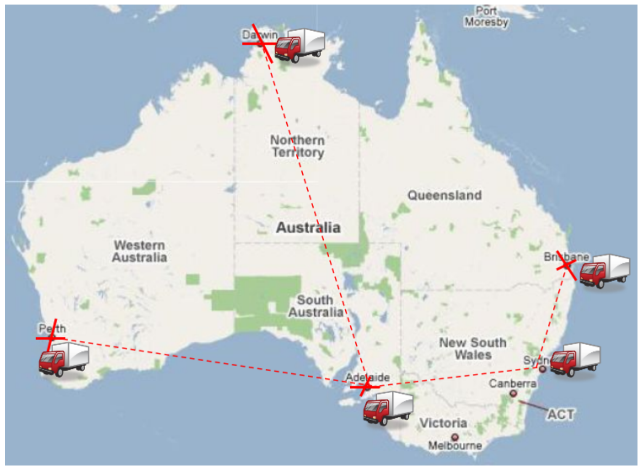
→ h^+= minimum effort to reach the goal under delete relaxation.

→ But won't h^+ usually under-estimate h^*? Yes, but that's just the effect of considering a relaxed problem. Arbitrarily adding actions useless within the relaxation (e.g., going to Moscow via Madrid) does not help to address it.

Proposition (h^+ is Consistent). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. Then h^+ is consistent, and thus admissible, safe, and goal-aware.*

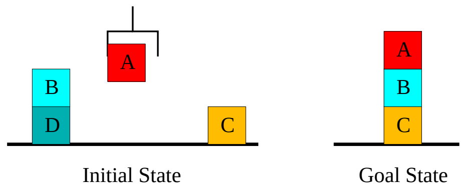
Proof. Let $s' = s[\![a]\!]$. We need to show that $h^+(s) \leq h^+(s') + c(a)$. Let π' be an optimal relaxed plan for s'. Construct $\pi := \langle a \rangle \circ \pi'$. It suffices to show that π is a relaxed plan for s. That is so because with Proposition slide 11 (ii), $s[\![a^+]\!]$ dominates $s[\![a]\!] = s'$, from which the claim follows with Proposition slide 10 (ii).

h^+ in TSP



h^+(TSP) = Minimum Spanning Tree

h^+ in the Blocksworld



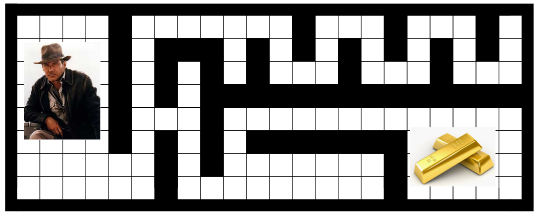
- Optimal plan: $\langle putdown(A), unstack(B, D), stack(B, C), pickup(A), stack(A, B)\rangle$.
- Optimal relaxed plan: $\langle stack(A, B), unstack(B, D), stack(B, C)\rangle$.

Observe: What can we say about the "search space surface" at the initial state here? The initial state lies on a local minimum under h^+, together with the successor state s where we stacked A onto B. All direct other neighbors of these two states have a strictly higher h^+ value.

Questionnaire

Question!

In the initial state of the Towers of Hanoi task with 5 discs, what is the value of h^+? (Assume STRIPS facts á la "on(disc1,disc2)", . . . , "on(disc5,peg1)")



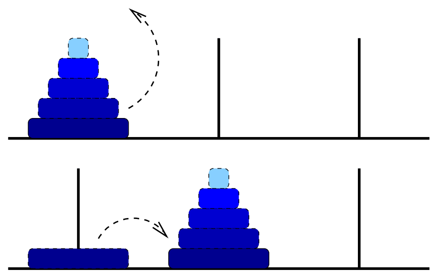
Question!

In this domain, h^+ is equal to?

(A): Manhattan Distance (B): h^*

Answer: Towers of Hanoi

→ The discs always "remain stacked", so we can just clear the bottom disc and move it over. For n discs, this takes $h^+(I) = n$ steps. So the correct answer here is "5".



Answer: Indiana, i.e., Finding a Path in a Graph

→ Manhattan Distance: No, relaxed plans can't walk through walls.

→ h^*: Yes!

Finding a Path in a Graph, STRIPS: From x to y in graph (N, E)

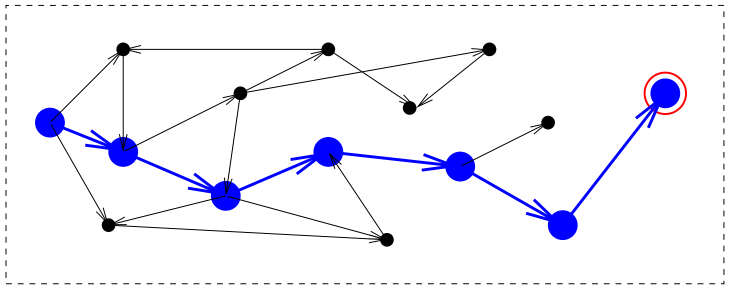
- $P = \{at(n) \mid n \in N\}$.
- $A = \{move(n, n') \mid (n, n') \in E\}$ where $move(n, n') = (\{at(n)\}, \{at(n')\}, \{at(n)\})$.
- $I = \{at(x)\}$; $G = \{at(y)\}$.

Proposition. *In the above STRIPS task* (P, A, c, I, G), $h^+(I) = h^*(I)$.

Proof. Say that $\vec{p} := \langle move(x, n_1), move(n_1, n_2), \dots, move(n_k, y)\rangle$ is an optimal relaxed plan for (P, A, c, I, G). Then $\vec{p}$ is a plan for (P, A, c, I, G) because $x, n_1, \dots, n_k, y$ is a shortest path from x to y. This traverses each node at most once, hence the deleted facts are not needed later on.

→ "Shortest paths never walk back", hence deleted facts are never needed again later on, hence delete relaxation is exact here.

h^+ in "Finding a Path in a Graph": Illustration



h^+(graph distance) = the real distance

(shortest paths never "walk back")

Questionnaire

Question!

Say the task is to drive from Saarbrücken (SB) to Moscow (M). Which of the following relaxed plans corresponds to the heuristic value returned by h^+?

(A): Take the shortest route from SB to M

(B): Drive from SB to M via Madrid

(C): Drive to Hongkong and Capetown in parallel, then from SB to M

(D): Drive to Hongkong and the same route back to SB, then from SB to M

→ (A): Yes. (B), (C), (D): Obviously not.

[Compare slide 13!]

How to Compute h^+?

Definition (PlanOpt$^+$). *By PlanOpt$^+$, we denote the problem of deciding, given a STRIPS planning task $\Pi = (P, A, c, I, G)$ and $B \in \mathbb{R}_0^+$, whether there exists a relaxed plan for Π whose cost is at most B.*

$\rightarrow$ By computing h^+, we would solve PlanOpt$^+$.

Theorem (Optimal Relaxed Planning is Hard). *PlanOpt$^+$ is* **NP***-complete.*

Proof. Membership: Easy (guess action sequences of length $|A|$). Hardness by reduction from SAT. Example: $\{C_1 = \{A\}, C_2 = \{\neg A\}\}$

- Actions setting variable to true, e.g.: pre empty, add $\{Atrue, Aset\}$.
- Actions setting variable to false, e.g.: pre empty, add $\{Afalse, Aset\}$.
- Actions satisfying clauses, e.g.: pre $Atrue$, add C_1sat; pre $Afalse$, add C_2sat.
- Goal: "X_iset" for all variables X_i, "C_jsat" for all clauses C_j.
- $B :=$ number of variables $+$ number of clauses ($= 3$ here).

And Now?

We approximate. (Business as usual)

Remember? (Chapter 7) *"Inadmissible heuristics typically arise as approximations of admissible heuristics that are too costly to compute. (Examples: Chapter 9)"*

$\rightarrow$ The delete relaxation heuristic we want is h^+. Unfortunately, this is hard to compute so the computational overhead is very likely to be prohibitive. All implemented systems using the delete relaxation approximate h^+ in one or the other way.

$\rightarrow$ We will look at the most wide-spread approaches to do so.

The Additive and Max Heuristics

Definition (h^{add}). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The additive heuristic h^{add} for Π is the function $h^{\text{add}}(s) := h^{\text{add}}(s, G)$ where $h^{\text{add}}(s, g)$ is the point-wise greatest function that satisfies $h^{\text{add}}(s, g) =$*

$$\begin{cases} 0 & g \subseteq s \\ min_{a \in A, g' \in add_a} c(a) + h^{\text{add}}(s, pre_a) & g = \{g'\} \\ \sum_{g' \in g} h^{\text{add}}(s, \{g'\}) & |g| > 1 \end{cases}$$

Definition (h^{max}). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The max heuristic h^{max} for Π is the function $h^{\text{max}}(s) := h^{\text{max}}(s, G)$ where $h^{\text{max}}(s, g)$ is the point-wise greatest function that satisfies $h^{\text{max}}(s, g) =$*

$$\begin{cases} 0 & g \subseteq s \\ min_{a \in A, g' \in add_a} c(a) + h^{\text{max}}(s, pre_a) & g = \{g'\} \\ \max_{g' \in g} h^{\text{max}}(s, \{g'\}) & |g| > 1 \end{cases}$$

The Additive and Max Heuristics: Properties

Proposition (h^{max} is Optimistic). *$h^{\text{max}} \leq h^+$, and thus $h^{\text{max}} \leq h^*$.*

Intuition. h^{max} simplifies relaxed planning by assuming that, to achieve a set g of subgoals, it suffices to achieve the single most costly $g' \in g$. Actual relaxed planning, i.e. h^+, can only be more expensive.

Proposition (h^{add} is Pessimistic). *For all STRIPS planning tasks Π, $h^{\text{add}} \geq h^+$. There exist Π and s so that $h^{\text{add}}(s) > h^*(s)$.*

Intuition. h^{add} simplifies relaxed planning by assuming that, to achieve a set g of subgoals, we must achieve every $g' \in g$ separately. Actual relaxed planning, i.e. h^+, can only be less expensive. Proof for inadmissiblity: see example on slide 34.

$\rightarrow$ Both h^{max} and h^{add} approximate h^+ by assuming that singleton subgoal facts are achieved independently. h^{max} estimates *optimistically* by the most costly singleton subgoal, h^{add} estimates *pessimistically* by summing over all singleton subgoals.

The Additive and Max Heuristics: Properties, ctd.

Proposition (h^{max} and h^{add} Agree with h^+ on ∞). *For all STRIPS planning tasks Π and states s in Π, $h^+(s) = \infty$ if and only if $h^{\text{max}}(s) = \infty$ if and only if $h^{\text{add}}(s) = \infty$.*

Proof. h^{max} and h^{add} agree on states with infinite heuristic value simply because their only difference lies in the use of the $\max$ vs. $\sum$ operations which does not affect this property.

$h^+(s) < \infty$ implies $h^{\text{max}}(s) < \infty$ because $h^{\text{max}} \leq h^+$. Vice versa, $h^{\text{max}}(s) < \infty$ implies $h^+(s) < \infty$ because h^{max} can then be used to generate a closed well-founded best-supporter function, from which a relaxed plan can be extracted, cf. the next section.

$\rightarrow$ States for which no relaxed plan exists are easy to recognize, and that is done by both h^{max} and h^{add}. Approximation is needed only for the cost of an optimal relaxed plan, if it exists.

Uh-Oh, I Think I Got a Déjà Vu Here ...

Reminder: $\rightarrow$ slide 27

... $h^{\text{max}}(s) := h^{\text{max}}(s, G)$ *where* $h^{\text{max}}(s, g)$ ... *satisfies* $h^{\text{max}}(s, g) =$

$$\begin{cases} 0 & g \subseteq s \\ min_{a \in A, g' \in add_a} c(a) + h^{\text{max}}(s, pre_a) & g = \{g'\} \\ \max_{g' \in g} h^{\text{max}}(s, \{g'\}) & |g| > 1 \end{cases}$$

Reminder: $\rightarrow$ **Chapter 8**

... $h^1(s) := h^1(s, G)$ *where* $h^1(s, g)$ ... *satisfies* $h^1(s, g) =$

$$\begin{cases} 0 & g \subseteq s \\ min_{a \in A, regr(g,a) \neq \bot} c(a) + h^1(s, regr(g, a)) & |g| = 1 \\ \max_{g' \in g} h^1(s, \{g'\}) & |g| > 1 \end{cases}$$

Proposition. $h^{\text{max}} = h^1$.

Proof. Say $g = \{g'\}$. $regr(\{g'\}, a) \neq \bot$ if $add_a \cap \{g'\} \neq \emptyset$ and $del_a \cap \{g'\} = \emptyset$; then, $regr(g, a) = (\{g'\} \setminus add_a) \cup pre_a$. Because $add_a \cap del_a = \emptyset$, this is the same as saying "$g' \in add_a$, and $regr(g, a) = pre_a$".

Questionnaire

Question!

Say the task is to drive from Saarbrücken (SB) to Moscow (M). Which of the following relaxed plans corresponds to the heuristic value returned by h^{max} and h^{add}?

(A): Take the shortest route from SB to M

(B): Drive from SB to M via Madrid

(C): Drive to Hongkong and Capetown in parallel, then from SB to M

(D): Drive to Hongkong and the same route back to SB, then from SB to M

$\rightarrow$ (A): Yes, because h^{max} and h^{add} both are equal to h^* here: There is a single goal fact, and every action has a single precondition only, so all subgoals in the equations on slide 27 will contain a single fact only. Hence the $|g| > 1$ cases never occur and the equations simplify to r^* (cf. **Chapter 8**).

$\rightarrow$ (B), (C), (D): Obviously no, as (A) is correct.

Déjà Vus Can Be Useful!

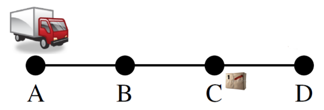
$\rightarrow$ You already know how to compute $h^{\text{max}} = h^1$. $\rightarrow$ **Chapter 8**
$\rightarrow$ Basically the same algorithm works for h^{add}!

Dynamic Programming algorithm computing h^{add} for state s

```
**new** table $T_0^{\text{add}}(g)$, for $g \in P$
For all $g \in P$: $T_0^{\text{add}}(g) := \begin{cases} 0 & g \in s \\ \infty & \text{otherwise} \end{cases}$
**fn** $Cost_i(g) := \begin{cases} T_i^{\text{add}}(g) & |g| = 1 \\ \sum_{g' \in g} T_i^{\text{add}}(g') & |g| > 1 \end{cases}$
**fn** $Next_i(g) := \min[Cost_i(g), \min_{a \in A, g' \in add_a} c(a) + Cost_i(pre_a)]$
**do forever**:
    **new** table $T_{i+1}^{\text{add}}(g)$, for $g \in P$
    For all $g \in P$: $T_{i+1}^{\text{add}}(g) := Next_i(g)$
    **if** $T_{i+1}^{\text{add}} = T_i^{\text{add}}$ **then** stop **endif**
    $i := i + 1$
**enddo**
```

Proposition. *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. Then the series $\{T_i^{\text{add}}(g)\}_{i=0,\dots}$ converges to $h^{\text{add}}(s, g)$, for all g.* (Proof omitted.)

Example: $h^{\max} = h^1$ in "Logistics"



- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X,Y), lo(X), ul(X)$.

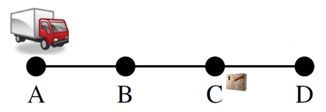
Content of Tables T_i^1:

i	$t(A)$	$t(B)$	$t(C)$	$t(D)$	$p(T)$	$p(A)$	$p(B)$	$p(C)$	$p(D)$
0	0	∞	∞	∞	∞	∞	∞	0	∞
1	0	1	∞	∞	∞	∞	∞	0	∞
2	0	1	2	∞	∞	∞	∞	0	∞
3	0	1	2	3	3	∞	∞	0	∞
4	0	1	2	3	3	4	4	0	4
5	0	1	2	3	3	4	4	0	4

$\rightarrow h^{\max}(I) = 4$.

$\rightarrow$ What if we had 101 packages at C with goal D? Then still $h^{\max}(I) = 4$ because each single package still has this same estimated cost.

Example: h^{add} in "Logistics"



- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X,Y), lo(X), ul(X)$.

Content of Tables T_i^{add}: (differences to content of T_i^1 shown in red)

i	$t(A)$	$t(B)$	$t(C)$	$t(D)$	$p(T)$	$p(A)$	$p(B)$	$p(C)$	$p(D)$
0	0	∞	∞	∞	∞	∞	∞	0	∞
1	0	1	∞	∞	∞	∞	∞	0	∞
2	0	1	2	∞	∞	∞	∞	0	∞
3	0	1	2	3	3	∞	∞	0	∞
4	0	1	2	3	3	4	5	0	7
5	0	1	2	3	3	4	5	0	7

$\rightarrow h^+(I) = 5 < 7 = h^{\text{add}}(I) < 8 = h^*(I)$.

BUT: $h^{\text{add}}(I) > h^+(I)$ because? $h^{\text{add}}(I)$ counts the cost of $dr(A,B), dr(B,C)$ 2 times, for the two preconditions $p(T)$ and $t(D)$ of $ul(D)$.

$\rightarrow$ What if the goal were $t(D), p(D)$? Then $h^{\text{add}}(I) = 10 > 5 = h^*(I) = h^+(I)$.

$\rightarrow$ What if we had 101 packages at C with goal D? Then $h^{\text{add}}(I) = 707 \gg 208 = h^*(I)$: For every package, cost 7 is added.

The Additive and Max Heuristics: So What?

Summary of typical issues in practice with h^{add} and $h^{\max}$:

- Both h^{add} and $h^{\max}$ can be computed reasonably quickly. (Well, compared to h^2 anyhow, never mind h^m for even larger m.)
- $h^{\max}$ is admissible, but is typically far too optimistic. (slide 33)
- h^{add} is not admissible, but is typically a lot more informed than $h^{\max}$. (slide 34)
- h^{add} is sometimes better informed than h^+, but "for the wrong reasons" (slide 34): Rather than accounting for deletes, it overcounts by ignoring positive interactions, i.e., sub-plans shared between subgoals.

  $\rightarrow$ Such overcounting can result in dramatic over-estimates of h^*!

$\rightarrow$ Recall: To be accurate, a heuristic needs to approximate the *minimum effort* needed to reach the goal.

$\rightarrow$ Relaxed plans (up next) keep h^{add}'s informativity but avoid over-counting.

Relaxed Plans, Basic Idea

$\rightarrow$ First compute a best-supporter function bs, which for every fact $p \in P$ returns an action that is deemed to be the cheapest achiever of p (within the relaxation). Then extract a relaxed plan from that function, by applying it to singleton subgoals and collecting all the actions.

$\rightarrow$ The best-supporter function can be based directly on $h^{\max}$ or h^{add}, simply selecting an action a achieving p that minimizes $[c(a)$ plus the cost estimate for $pre_a]$. That is, a best achiever of p in the equation characterizing $h^{\max}$ respectively h^{add} (cf. slide 27).

And now for the details:

- To be concrete: the best-supporter functions we will actually use.
- How to extract a relaxed plan given a best-supporter function.
- What is a best-supporter function, in general?

Preview: The Best-Supporter Functions we Will Use

Definition (Best-Supporters from $h^{\max}$ and h^{add}). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task, and let s be a state.*

The $h^{\max}$ supporter function $bs_s^{\max} : \{p \in P \mid 0 < h^{\max}(s, \{p\}) < \infty\} \mapsto A$ is defined by $bs_s^{\max}(p) := \arg\min_{a \in A, p \in add_a} c(a) + h^{\max}(s, pre_a)$.

The h^{add} supporter function $bs_s^{\text{add}} : \{p \in P \mid 0 < h^{\text{add}}(s, \{p\}) < \infty\} \mapsto A$ is defined by $bs_s^{\text{add}}(p) := \arg\min_{a \in A, p \in add_a} c(a) + h^{\text{add}}(s, pre_a)$.

Example h^{add} in "Logistics":

Heuristic values:

	$t(A)$	$t(B)$	$t(C)$	$t(D)$	$p(T)$	$p(A)$	$p(B)$	$p(C)$	$p(D)$
h^{add}	0	1	2	3	3	4	5	0	7

Yield best-supporter function:

	$t(A)$	$t(B)$	$t(C)$	$t(D)$	$p(T)$	$p(A)$	$p(B)$	$p(C)$	$p(D)$
bs^{add}	–	$dr(A,B)$	$dr(B,C)$	$dr(C,D)$	$lo(C)$	$ul(A)$	$ul(B)$	–	$ul(D)$

Relaxed Plan Extraction

Relaxed Plan Extraction for state s and best-supporter function bs

```
$Open := G \setminus s$; $Closed := \emptyset$; $RPlan := \emptyset$
while $Open \neq \emptyset$ do:
    select $g \in Open$
    $Open := Open \setminus \{g\}$; $Closed := Closed \cup \{g\}$;
    $RPlan := RPlan \cup \{bs(g)\}$; $Open := Open \cup (pre_{bs(g)} \setminus (s \cup Closed))$
endwhile
return $RPlan$
```

→ Starting with the top-level goals, iteratively close open singleton subgoals by selecting the best supporter.

This is fast! Number of iterations bounded by $|P|$, each near-constant time.

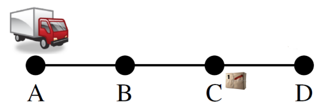
But is it correct?

→ What if $g \notin add_{bs(g)}$? Doesn't make sense. → Condition (A).

→ What if $bs(g)$ is undefined? Segmentation fault. → Condition (B).

→ What if the support for g eventually requires g itself (then already in *Closed*) as a precondition? Then this does not yield a relaxed plan. → Condition (C).

Relaxed Plan Extraction from h^{add} in "Logistics"



- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X,Y), lo(X), ul(X)$.

	$t(A)$	$t(B)$	$t(C)$	$t(D)$	$p(T)$	$p(A)$	$p(B)$	$p(C)$	$p(D)$
bs^{add}	–	$dr(A,B)$	$dr(B,C)$	$dr(C,D)$	$lo(C)$	$ul(A)$	$ul(B)$	–	$ul(D)$

Extracting a relaxed plan:

1. $bs_s^{\text{add}}(p(D)) = ul(D)$; opens $t(D), p(T)$.
2. $bs_s^{\text{add}}(t(D)) = dr(C,D)$; opens $t(C)$.
3. $bs_s^{\text{add}}(t(C)) = dr(B,C)$; opens $t(B)$.
4. $bs_s^{\text{add}}(t(B)) = dr(A,B)$; opens nothing.
5. $bs_s^{\text{add}}(p(T)) = lo(C)$; opens nothing.
6. Anything more? No, open goals empty at this point.

Best-Supporter Functions

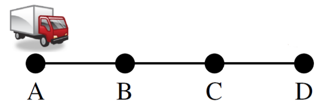
→ For relaxed plan extraction to make sense, it requires a *closed well-founded* best-supporter function:

Definition (Best-Supporter Function). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task, and let s be a state. A best-supporter function for s is a partial function $bs : (P \setminus s) \mapsto A$ such that $p \in add_a$ whenever $a = bs(p)$.*

The support graph of bs is the directed graph with vertices $(P \setminus s) \cup A$ and arcs $\{(a, p) \mid a = bs(p)\} \cup \{(p, a) \mid p \in pre_a\}$. We say that bs is closed if $bs(p)$ is defined for every $p \in (P \setminus s)$ that has a path to a goal $g \in G$ in the support graph. We say that bs is well-founded if the support graph is acyclic.

- "$p \in add_a$ whenever $a = bs(p)$": Condition (A).
- bs is closed: Condition (B). ("bs will be defined wherever it takes us to")
- bs is well-founded: Condition (C). (Relaxed plan extraction starts at the goals, and chains backwards in the support graph. If there are cycles, then this backchaining may not reach the currently true state s, and thus not yield a relaxed plan.)

Support Graphs and Condition (C) in "Logistics"



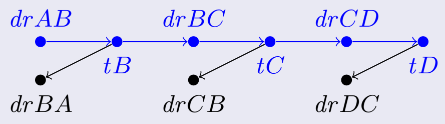
- Initial state: tA.
- Goal: tD.
- Actions: $drXY$.

How to do it (well-founded)

Best-supporter function:

p	$bs(p)$
$t(B)$	$dr(A,B)$
$t(C)$	$dr(B,C)$
$t(D)$	$dr(C,D)$

Yields support graph backchaining:

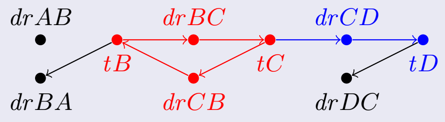


How NOT to do it (not well-founded)

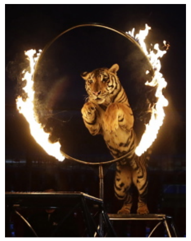
Best-supporter function:

p	$bs(p)$
$t(B)$	$dr(C,B)$
$t(C)$	$dr(B,C)$
$t(D)$	$dr(C,D)$

Yields support graph backchaining:



Questionnaire



- $P = \{alive, haveTiger, tamedTiger, haveJump\}$.
  Short: $P = \{A, hT, tT, J\}$.
- Initial state I: $alive$.
- Goal G: $alive, haveJump$.
- Actions A:
  $getTiger$: pre $alive$; add $haveTiger$
  $tameTiger$: pre $alive$, $haveTiger$; add $tamedTiger$
  $jumpTamedTiger$: pre $alive$, $tamedTiger$; add $haveJump$
  $jumpTiger$: pre $alive$, $haveTiger$; add $haveJump$; del $alive$

Question!

What is the h^{add} best supporter for $haveJump$? And the $h^{\max}$ best supporter?

$\rightarrow$ There are two candidates in each case, namely the actions adding $haveJump$: $jumpTiger$ and $jumpTamedTiger$.

The precondition of $jumpTiger$ has $h^{\text{add}} = h^{\max}$ value 1, and that of $jumpTamedTiger$ has $h^{\text{add}} = h^{\max}$ value 2. So both h^{add} and $h^{\max}$ force us to use $jumpTiger$.

$h^{\max}$ and h^{add} Supporter Functions: Correctness

Proposition. *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task such that, for all $a \in A$, $c(a) > 0$. Let s be a state where $h^+(s) < \infty$. Then both $bs_s^{\max}$ and bs_s^{add} are closed well-founded supporter functions for s.*

Proof. Since $h^+(s) < \infty$ implies $h^{\max}(s) < \infty$, it is easy to see that $bs_s^{\max}$ is closed ($h^{\max}(s, G) < \infty$, and recursively $h^{\max}(s, pre_a) < \infty$ for the best supporters).

If $a = bs_s^{\max}(p)$, then a is the action yielding $0 < h^{\max}(s, \{p\}) < \infty$ in the $h^{\max}$ equation.

Since $c(a) > 0$, we have $h^{\max}(s, pre_a) < h^{\max}(s, \{p\})$ and thus, for all $q \in pre_a$, $h^{\max}(s, \{q\}) < h^{\max}(s, \{p\})$.

Transitively, if the support graph contains a path from fact vertex r to fact vertex t, then $h^{\max}(s, \{r\}) < h^{\max}(s, \{t\})$. Thus there can't be cycles in the support graph and $bs_s^{\max}$ is well-founded. Similar for bs_s^{add}.

[$\rightarrow$ One can also use $h^{\max}$ and h^{add} for 0-cost actions, by appropriate tie-breaking in cases where $h^{\max}(s, \{p\}) = h^{\max}(s, pre_a)$. Details omitted.]

Relaxed Plan Extraction: Correctness

Proposition. *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task, let s be a state, and let bs be a closed well-founded best-supporter function for s. Then the action set $RPlan$ returned by relaxed plan extraction can be sequenced into a relaxed plan $\vec{a}^+$ for s.*

Proof. Order a before a' whenever the support graph contains a path from a to a'. Since the support graph is acyclic, such a sequencing $\vec{a} := \langle a_1, \dots, a_n \rangle$ exists.

We have $p \in s$ for all $p \in pre_{a_1}$, because otherwise $RPlan$ would contain the action $bs(p)$, necessarily ordered before a_1.

We have $p \in s \cup add_{a_1}$ for all $p \in pre_{a_2}$, because otherwise $RPlan$ would contain the action $bs(p) \neq a_1$, necessarily ordered before a_2.

Iterating the argument, over $p \in pre_{a_{i+1}}$ and $s \cup add_{a_1} \cup \dots \cup add_{a_i}$, shows that $\vec{a}^+$ is a relaxed plan for s.

The Relaxed Plan Heuristic

Definition (Relaxed Plan Heuristic). *A heuristic function is called a relaxed plan heuristic, denoted h^{FF}, if, given a state s, it returns ∞ if no relaxed plan exists, and otherwise returns $\sum_{a \in RPlan} c(a)$ where $RPlan$ is the action set returned by relaxed plan extraction on a closed well-founded best-supporter function for s.*

Recall: (that this makes sense because)

- If a relaxed plan exists, then there exists a closed well-founded best-supporter function bs (cf. slide 44).
- Relaxed plan extraction on bs yields a relaxed plan (previous slide).

Observe in "Logistics" (slide 40):
$h^{\text{FF}}(I) = 5 = h^+(I) < 7 = h^{\text{add}}(I) < 8 = h^*(I)$. **BUT:**

$\rightarrow$ If the goal is $t(D), p(D)$? $h^{\text{add}}(I) = 10 > 5 = h^*(I) = h^{\text{FF}}(I) = h^+(I)$.

$\rightarrow$ If we have 101 packages at C that need to go to D? $h^{\text{FF}}(I) = 205$ because relaxed plan extraction selects the drive actions only once. By contrast, $h^{\text{add}}(I) = 707$ overcounts these actions, cf. slide 34.

The Relaxed Plan Heuristic: Properties

Proposition (h^{FF} is Pessimistic and Agrees with h^+ on ∞). *For all STRIPS planning tasks Π, $h^{\text{FF}} \geq h^+$; for all states s, $h^+(s) = \infty$ if and only if $h^{\text{FF}}(s) = \infty$. There exist Π and s so that $h^{\text{FF}}(s) > h^*(s)$.*

Proof. $h^{\text{FF}} \geq h^+$ follows directly from the previous slide. Agrees with h^+ on ∞: Direct from definition. Inadmissiblity: Whenever bs makes sub-optimal choices. $\rightarrow$ **Exercise, perhaps**

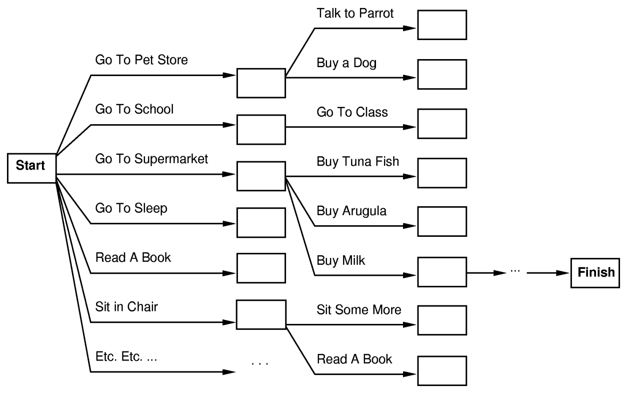
$\rightarrow$ Relaxed plan heuristics have the same theoretical properties as h^{add}.

So what's the point?

- In practice, h^{FF} typically does not over-estimate h^* (or not by a large amount, anyway).
  $\rightarrow$ h^{FF} *may* be inadmissible, just like h^{add}, but for more subtle reasons.
- Can h^{FF} over-count, i.e., count sub-plans shared between subgoals more than once? No, due to the set union in "$RPlan := RPlan \cup \{bs(g)\}$".

Helpful Actions Pruning: Idea & Impact

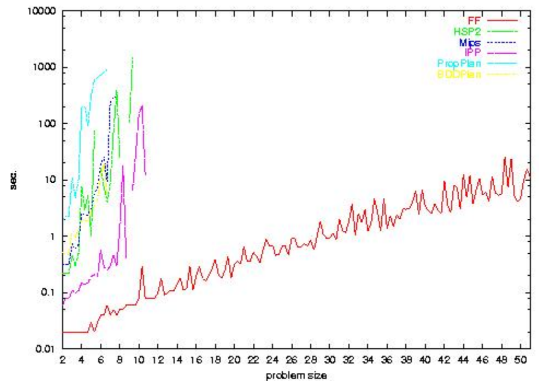
$\rightarrow$ In search, expand only those actions contained in the relaxed plan.



Relaxed plan = "Go To Supermarket, Buy Milk, ..."

Helpful Actions Pruning: Idea & Impact

$\rightarrow$ In search, expand only those actions contained in the relaxed plan.



(Schedule domain: many tools, many objects.)

Relaxed plan does *not* drill holes into objects that need to be painted.

Helpful Actions Pruning

Definition (Helpful Actions). *Let h^{FF} be a relaxed plan heuristic, let s be a state, and let $RPlan$ be the action set returned by relaxed plan extraction on the closed well-founded best-supporter function for s which underlies h^{FF}. Then an action a applicable to s is called helpful if it is contained in $RPlan$.*

Remarks:

- Initially introduced in FF [Hoffmann and Nebel (2001)], restricting Enforced Hill-Climbing to use *only* the helpful actions.
- There is no guarantee that the actually needed actions will be helpful, so this does not preserve completeness (cf. slide 43).
- Fast Downward uses the term *preferred operators*, for similar concepts for a broad variety of heuristic functions h.
- Fast Downward (the real one, not the stripped one in the **Exercises**) offers a variety of ways for using preferred operators.
- Preferred operators may have more impact on performance than different heuristic functions [Richter and Helmert (2009)].

Questionnaire

Question!

Say the task is to drive from Saarbrücken (SB) to Moscow (M). Which of the following relaxed plans may be returned by Relaxed Plan Extraction from h^{max} and h^{add}?

(A): Take the shortest route from SB to M

(B): Drive from SB to M via Madrid

(C): Drive to Hongkong and Capetown in parallel, then from SB to M

(D): Drive to Hongkong and the same route back to SB, then from SB to M

→ (A): Yes, because h^{max} and h^{add} both are equal to h^* in this domain, cf. slide 31.

→ (B), (C), (D): Obviously no, as (A) is correct.

Ignoring Deletes When the Language Doesn't Have Any?

Reminder: → Chapter 2

Definition (FDR Planning Task). *A finite-domain representation planning task, short FDR planning task, is a 5-tuple $\Pi = (V, A, c, I, G)$ where:*

- *V is a finite set of state variables, each $v \in V$ with a finite domain D_v.*
- *A is a finite set of actions; each $a \in A$ is a pair (pre_a, eff_a) of partial variable assignments referred to as the action's precondition and effects.*
- *...*

We refer to pairs $v = d$ of variable and value as facts. We identify (partial) variable assignments with sets of facts.

→ "Delete relaxation" = "act as if all facts that were once true will remain true forever" = "FDR state variables accumulate, rather than change, their values".

→ In practice (in particular, in the Fast Downward implementation), simply formulate the algorithms relative to the "FDR facts" $v = d$.

→ What follows is the machinery needed to make this formal.

Delete Relaxed FDR Planning

Definition (Delete Relaxed FDR). *Let $\Pi = (V, A, c, I, G)$ be an FDR planning task. Denote by $P_V := \{v = d \mid v \in V, d \in D_v\}$ the set of (FDR) facts. The relaxed state space of Π is the labeled transition system $\Theta^+_\Pi = (S^+, L, c, T, I, S^{+G})$ where:*

- *The states (also relaxed states) $S^+ = 2^{P_V}$ are the subsets s^+ of P_V.*
- *The labels $L = A$ are Π's actions; the cost function c is that of Π.*
- *The transitions are $T = \{s^+ \xrightarrow{a} s'^+ \mid pre_a \subseteq s^+, s'^+ = s^+ \cup eff_a\}$.*
- *The initial state I is identical to that of Π.*
- *The goal states are $S^{+G} = \{s^+ \in S^+ \mid G \subseteq s^+\}$.*

An (optimal) relaxed plan for $s^+ \in S^+$ is an (optimal) solution for s^+ in Θ^+_Π. A relaxed plan for I is also called a relaxed plan for Π.
Let $\Theta_\Pi = (S, A, c, T, I, G)$ be the state space of Π. The optimal delete relaxation heuristic h^+ for Π is the function $h^+ : S \mapsto \mathbb{R}^+_0 \cup \{\infty\}$ where $h^+(s)$ is defined as the cost of an optimal relaxed plan for s.

→ FDR states contain exactly one fact for each variable $v \in V$. There is no such restriction on FDR relaxed states.

Done With FDR-2-STRIPS

Reminder: $\rightarrow$ Chapter 2

Proposition. *Let $\Pi = (V, A, c, I, G)$ be an FDR planning task, and let Π^{STR} be its STRIPS translation. Then Θ_Π is isomorphic to the sub-system of $\Theta_{\Pi^{\text{STR}}}$ induced by those $s \subseteq P_V$ where, for each $v \in V$, s contains exactly one fact of the form $v = d$. All other states in $\Theta_{\Pi^{\text{STR}}}$ are unreachable.*

Observe: Θ^+_Π has transition $s^+ \xrightarrow{a} s'^+$ if and only if $s^+[\![a^{\text{STR}+}]\!] = s'^+$ in Π^{STR}. (Because $s^+[\![a^{\text{STR}+}]\!] = s^+ \cup \mathit{eff}_a$)

Proposition. *Denote by h^*_Π and h^+_Π the perfect heuristic and the optimal delete relaxation heuristic in Π, and denote by $h^*_{\Pi^{\text{STR}}}$ and $h^+_{\Pi^{\text{STR}}}$ these heuristics in Π^{STR}. Then, for all states s of Π, $h^*_\Pi(s) = h^*_{\Pi^{\text{STR}}}(s)$ and $h^+_\Pi(s) = h^+_{\Pi^{\text{STR}}}(s)$.*

$\rightarrow$ Given an FDR task Π, *everything* we have done here can be done for Π by doing it within Π^{STR}.

Summary

- The delete relaxation simplifies STRIPS by removing all delete effects of the actions.
- The cost of optimal relaxed plans yields the heuristic function h^+, which is admissible but hard to compute.
- We can approximate h^+ optimistically by $h^{\max}$, and pessimistically by h^{add}. $h^{\max}$ is admissible, h^{add} is not. h^{add} is typically much more informative, but can suffer from over-counting.
- Either of $h^{\max}$ or h^{add} can be used to generate a closed well-founded best-supporter function, from which we can extract a relaxed plan.
- The resulting relaxed plan heuristic h^{FF} does not do over-counting, but otherwise has the same theoretical properties as h^{add}; in practice, it typically does not over-estimate h^*.
- The delete relaxation can be applied to FDR simply by accumulating variable values, rather than over-writing them. This is formally equivalent to treating variable/value pairs like STRIPS facts.

Example Systems

HSP [Bonet and Geffner (2001)]

1. Search space: Progression (STRIPS-based).
2. Search algorithm: Greedy best-first search.
3. Search control: h^{add}.

FF [Hoffmann and Nebel (2001)]

1. Search space: Progression (STRIPS-based).
2. Search algorithm: Enforced hill-climbing ($\rightarrow$ Chapter 7).
3. Search control: h^{FF} extracted from $h^{\max}$ supporter function; helpful actions pruning.

LAMA [Richter and Westphal (2010)]

1. Search space: Progression (FDR-based).
2. Search algorithm: Multiple-queue greedy best-first search.
3. Search control: h^{FF} + a landmark heuristic ($\rightarrow$ Chapter 14); for each, one search queue all actions, one search queue only preferred operators.

Remarks

- HSP was competitive in the 1998 International Planning Competition (IPC'98); FF outclassed the competitors in IPC'00.
- The delete relaxation is still at large, in particular with the wins of LAMA and derivatives in the satisficing planning tracks of IPC'08, IPC'11,and IPC'14.
- I have personally done quite some work on understanding why this relaxation works so well, in the planning benchmarks [Hoffmann (2005, 2011)].
- It has always been a challenge to take *some* delete effects into account. Recent works of the FAI group allow, for the first time, to interpolate smoothly between h^+ and h^*: explicit conjunctions [Keyder *et al.* (2012, 2014); Hoffmann and Fickert (2015); Fickert *et al.* (2016)] and red-black planning [Katz *et al.* (2013); Katz and Hoffmann (2013); Domshlak *et al.* (2015)]. $\rightarrow$ Chapter 10

Remarks, ctd.

- While $h^{\max}$ is not informative in practice, other lower-bounding approximations of h^+ are very important for optimal planning: admissible landmark heuristics [Karpas and Domshlak (2009)] (**Chapters 14 and 16**); LM-cut heuristic [Helmert and Domshlak (2009)] (**Chapter 17**).
- The delete relaxation has also been applied in Model Checking [Kupferschmid *et al.* (2006)].

  → More generally, the relaxation principle is very generic and potentially applicable in many different contexts, as are all relaxation principles covered in this course.

Reading

- *Planning as Heuristic Search* [Bonet and Geffner (2001)].

  Available at:

  `http://www.dtic.upf.edu/~hgeffner/html/reports/hsp-aij.ps`

  Content: This is "where it all started": the first paper[1] explicitly introducing the notion of heuristic search and automatically generated heuristic functions to planning. Introduces the additive and max heuristics h^{add} and $h^{\max}$.

[1]Well, this is the first full journal paper treating the subject; the same authors published conference papers in AAAI'97 and ECP'99, which are subsumed by the present paper.

Reading, ctd.

- *The FF Planning System: Fast Plan Generation Through Heuristic Search* [Hoffmann and Nebel (2001)].

  Available at:

  `http://fai.cs.uni-saarland.de/hoffmann/papers/jair01.pdf`

  Content: The main reference for delete relaxation heuristics. Introduces the relaxed plan heuristic, extracted from the $h^{\max}$ supporter function.[2] Also introduces helpful actions pruning, and enforced hill-climbing.

[2]Done in a unit-cost setting presented in terms of relaxed planning graphs instead of $h^{\max}$, and not identifying the more general idea of using a well-founded best-supporter function (I used the same simpler presentation in the AI'18 core course). The notion of best-supporter functions (handling non-unit action costs) first appears in [Keyder and Geffner (2008)].

References I

Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33, 2001.

Carmel Domshlak, Jörg Hoffmann, and Michael Katz. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, 221:73–114, 2015.

Maximilian Fickert, Jörg Hoffmann, and Marcel Steinmetz. Combining the delete relaxation with critical-path heuristics: A direct characterization. *Journal of Artificial Intelligence Research*, 56(1):269–327, 2016.

Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 162–169. AAAI Press, 2009.

References II

Jörg Hoffmann and Maximilian Fickert. Explicit conjunctions w/o compilation: Computing $h^{\mathsf{FF}}(\Pi^C)$ in polynomial time. In Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*. AAAI Press, 2015.

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

Jörg Hoffmann. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research*, 24:685–758, 2005.

Jörg Hoffmann. Analyzing search topology without running any search: On the connection between causal graphs and h^+. *Journal of Artificial Intelligence Research*, 41:155–229, 2011.

Erez Karpas and Carmel Domshlak. Cost-optimal planning with landmarks. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1728–1733, Pasadena, California, USA, July 2009. Morgan Kaufmann.

References III

Michael Katz and Jörg Hoffmann. Red-black relaxed plan heuristics reloaded. In Malte Helmert and Gabriele Röger, editors, *Proceedings of the 6th Annual Symposium on Combinatorial Search (SOCS'13)*, pages 105–113. AAAI Press, 2013.

Michael Katz, Jörg Hoffmann, and Carmel Domshlak. Who said we need to relax *all* variables? In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, pages 126–134, Rome, Italy, 2013. AAAI Press.

Emil Keyder and Hector Geffner. Heuristics for planning with action costs revisited. In Malik Ghallab, editor, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08)*, pages 588–592, Patras, Greece, July 2008. Wiley.

Emil Keyder, Jörg Hoffmann, and Patrik Haslum. Semi-relaxed plan heuristics. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 128–136. AAAI Press, 2012.

Emil Keyder, Jörg Hoffmann, and Patrik Haslum. Improving delete relaxation heuristics through explicitly represented conjunctions. *Journal of Artificial Intelligence Research*, 50:487–533, 2014.

References IV

Sebastian Kupferschmid, Jörg Hoffmann, Henning Dierks, and Gerd Behrmann. Adapting an AI planning heuristic for directed model checking. In Antti Valmari, editor, *Proceedings of the 13th International SPIN Workshop (SPIN 2006)*, volume 3925 of *Lecture Notes in Computer Science*, pages 35–52. Springer-Verlag, 2006.

Silvia Richter and Malte Helmert. Preferred operators and deferred evaluation in satisficing planning. In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 273–280. AAAI Press, 2009.

Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39:127–177, 2010.