5 - Delete Relaxation Heuristics

知识点 & 题目

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

Every h yields good performance only in some domains! (Search reduction vs. computational overhead)

Relaxed world: "What was once true remains true forever."

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

$\rightarrow$ "+" super-script = delete relaxed. We'll also use this to denote states encountered within the relaxation. (For STRIPS, s^+ is a fact set just like s.)

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

$\rightarrow$ Anybody remember what Π_s is? $\Pi_s = (F, A, c, s, G)$

A Relaxed plan for "TSP" in Australia:

1. Initial state: $\{at(Sy), v(Sy)\}$.
2. Apply $drive(Sy, Br)^+$: $\{at(Br), v(Br), at(Sy), v(Sy)\}$.
3. Apply $drive(Sy, Ad)^+$: $\{at(Ad), v(Ad), at(Br), v(Br), at(Sy), v(Sy)\}$.
4. Apply $drive(Ad, Pe)^+$: $\{at(Pe), v(Pe), at(Ad), v(Ad), at(Br), v(Br), at(Sy), v(Sy)\}$.
5. Apply $drive(Ad, Da)^+$: $\{at(Da), v(Da), at(Pe), v(Pe), at(Ad), v(Ad), at(Br), v(Br), at(Sy), v(Sy)\}$.

Delete Relaxation & State Dominance

The delete relaxation is aka ignoring delete lists.

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

$\rightarrow$ For example, on the previous slide, who dominates who? Each state along the relaxed plan dominates the previous one, simply because the actions don't delete any facts.

Proposition (Dominance). *Let $\Pi^+ = (F, A^+, c, I, G)$ 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 $appl(s'^+, \vec{a}^+)$ dominates $appl(s^+, \vec{a}^+)$.*

Proof. (i) is trivial. (ii) by induction over the length n of $\vec{a}^+$. Base case $n = 0$ is trivial. Inductive case $n \rightarrow n + 1$ follows directly from induction hypothesis and the definition of $appl(.,.)$.

$\rightarrow$ It is always better to have more facts true.

Proposition. *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task, let s be a state, and let $a \in A$. Then $appl(s, a^+)$ dominates both (i) s and (ii) $appl(s, a)$.*

Proof. Trivial from the definitions of $appl(s, a)$ and a^+.

$\Longrightarrow$ Optimal relaxed plans admissibly estimate the cost of optimal plans:

Proposition (Delete Relaxation is Admissible). *Let $\Pi = (F, A, c, I, G)$ 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 $appl(s, \vec{a}^+)$ dominates $appl(s, \vec{a})$. Base case is trivial, inductive case follows from (ii) above.

$\Longrightarrow$ It is now clear how to find a relaxed plan:

- Applying a relaxed action can only ever make more facts true ((i) above).
- That can only be good, i.e., cannot render the task unsolvable (dominance proposition).

$\rightarrow$ 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 $appl(s^+, a^+) \neq s^+$ **then**
    select one such a
    $s^+ := appl(s^+, a^+)$; $\vec{a}^+ := \vec{a}^+ \circ \langle a^+ \rangle$
  else return "Π_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, $G \subseteq appl(s, \vec{a}^+)$. Completeness: If "$\Pi_s^+$ is unsolvable" is returned, then no relaxed plan exists for s^+ at that point; since s^+ dominates s, by the dominance proposition this implies that no relaxed plan can exist for s. Termination: Every $a \in A$ can be selected at most once because afterwards $appl(s^+, a^+) = s^+$.

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

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 heuristic safe? Yes: $h(s) = \infty$ only if no relaxed plan for s exists, which by admissibility of delete relaxation implies that no plan for s exists.

$\rightarrow$ Is this heuristic goal-aware? Yes, we'll have $G \subseteq s^+$ right at the start.

$\rightarrow$ Is this heuristic admissible? Would be if the relaxed plans were optimal; but they clearly aren't. So h isn't consistent either.

$\rightarrow$ To be informed (accurately estimate h^*), a heuristic needs to approximate the *minimum effort* needed to reach the goal. Greedy relaxed planning doesn't do this because it may select arbitrary actions that aren't relevant at all.

The optimal delete relaxation heuristic

Definition (h^+). *Let $\Pi = (F, A, c, I, G)$ 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.*

Corollary (h^+ is Admissible). *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task. Then h^+ is admissible, and thus safe and goal-aware.* (By admissibility of delete relaxation.)

$\rightarrow$ To be informed (accurately estimate h^*), a heuristic needs to approximate the *minimum effort* needed to reach the goal. h^+ naturally does so by asking for the cheapest possible relaxed plans.

[$\rightarrow$ You might rightfully ask "But won't optimal relaxed plans usually under-estimate h^*?" Yes, but that's just the effect of considering a relaxed problem, and arbitrarily adding actions useless within the relaxation does not help to address it.]

- P: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- A: $drive(x, y)$ where x, y have a road.

$$c(drive(x,y)) = \begin{cases} 1 & \{x,y\} = \{Sy, Br\} \\ 1.5 & \{x,y\} = \{Sy, Ad\} \\ 3.5 & \{x,y\} = \{Ad, Pe\} \\ 4 & \{x,y\} = \{Ad, Da\} \end{cases}$$

- I: $at(Sy), v(Sy)$; G: $at(Sy), v(x)$ for all x.

Planning vs. Relaxed Planning:

- Optimal plan: $\langle drive(Sy, Br), drive(Br, Sy), drive(Sy, Ad), drive(Ad, Pe), drive(Pe, Ad), drive(Ad, Da), drive(Da, Ad), drive(Ad, Sy)\rangle$.
- Optimal relaxed plan: $\langle drive(Sy, Br), drive(Sy, Ad), drive(Ad, Pe), drive(Ad, Da)\rangle$.
- $h^*(I) = 20$; $h^+(I) = 10$.

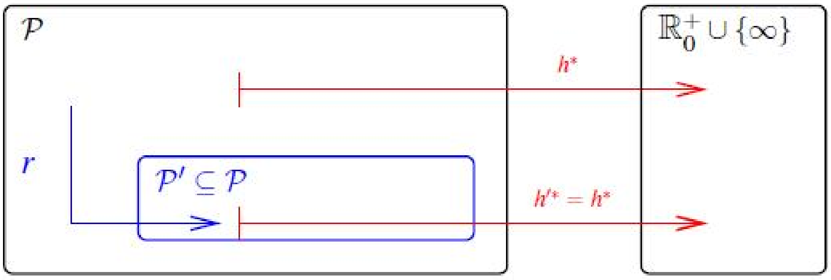
h^+(TSP) = Minimum Spanning Tree

h^+(Hanoi) = n, not 2^n L5 P16

Definition (Optimal Relaxed Planning). *By PlanOpt$^+$, we denote the problem of deciding, given a STRIPS planning task $\Pi = (F, A, c, I, G)$ 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: L5 P17



where, for all $\Pi \in \mathcal{P}$, $h^*(r(\Pi)) \leq h^*(\Pi)$.

For $h^+ = h^* \circ r$:

- Problem $\mathcal{P}$: All STRIPS planning tasks.
- Simpler problem $\mathcal{P}'$: All STRIPS planning tasks with empty deletes.
- Perfect heuristic h'^* for $\mathcal{P}'$: Optimal plan cost $= h^*$ on $\mathcal{P}'$.
- Transformation r: Drop the deletes.

→ Is this a native relaxation? Yes.
→ Is this relaxation efficiently constructible? Yes.
→ Is this relaxation efficiently computable? No.

Not efficiently computable:

- (a) approximate h'^*
- (b) design h'^* in a way so that it will typically be feasible
- (c) just live with it and hope for the best
  - Many known relaxations (in planning) are efficiently computable, some aren't. The latter use (a).
  - (b) and (c) are not used anywhere right now.
- 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.

The Additive and Max Heuristics

Definition (h^{add}). *Let $\Pi = (F, A, c, I, G)$ 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 $h^{\text{add}}(s, g) =$*

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

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

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

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

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

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

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 $h^{\max}(s) = \infty$ if and only if $h^{\text{add}}(s) = \infty$.*

$\rightarrow$ 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 pan, if it exists.

Bellman-Ford for and h^{add}

Bellman-Ford variant computing h^{add} for state s

new table $T_0^{\text{add}}(g)$, for $g \in F$

For all $g \in F$: $T_0^{\text{add}}(g) := \begin{cases} 0 & g \in s \\ \infty & \text{otherwise} \end{cases}$

fn $c_i(g) := \begin{cases} T_i^{\text{add}}(g) & |g| = 1 \\ \sum_{g' \in g} T_i^{\text{add}}(g') & |g| > 1 \end{cases}$

fn $f_i(g) := \min[c_i(g), \min_{a \in A, g \in add_a} c(a) + c_i(pre_a)]$

do forever:

  new table $T_{i+1}^{\text{add}}(g)$, for $g \in F$

  For all $g \in F$: $T_{i+1}^{\text{add}}(g) := f_i(g)$

  if $T_{i+1}^{\text{add}} = T_i^{\text{add}}$ **then** stop **endif**

  $i := i + 1$

enddo

$\rightarrow$ Basically the same algorithm works for $h^{\max}$, just change $\sum$ for $\max$

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

$h^{\max}$ in "TSP" in Australia

- F: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- A: $drive(x, y)$ where x, y have a road.

$$c(drive(x,y)) = \begin{cases} 1 & \{x,y\} = \{Sy, Br\} \\ 1.5 & \{x,y\} = \{Sy, Ad\} \\ 3.5 & \{x,y\} = \{Ad, Pe\} \\ 4 & \{x,y\} = \{Ad, Da\} \end{cases}$$

- I: $at(Sy), v(Sy)$; G: $at(Sy), v(x)$ for all x.

Content of Tables T_i^1:

i	$at(Sy)$	$at(Ad)$	$at(Br)$	$at(Pe)$	$at(Da)$	$v(Sy)$	$v(Ad)$	$v(Br)$	$v(Pe)$	$v(Da)$
0	0	∞	∞	∞	∞	0	∞	∞	∞	∞
1	0	1.5	1	∞	∞	0	1.5	1	∞	∞
2	0	1.5	1	5	5.5	0	1.5	1	5	5.5
3	0	1.5	1	5	5.5	0	1.5	1	5	5.5

$\rightarrow h^{\max}(I) = 5.5 << 20 = h^*(I)$.

h^{add} in "TSP" in Australia

- F: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- A: $drive(x, y)$ where x, y have a road.

$$c(drive(x,y)) = \begin{cases} 1 & \{x,y\} = \{Sy, Br\} \\ 1.5 & \{x,y\} = \{Sy, Ad\} \\ 3.5 & \{x,y\} = \{Ad, Pe\} \\ 4 & \{x,y\} = \{Ad, Da\} \end{cases}$$

- I: $at(Sy), v(Sy)$; G: $at(Sy), v(x)$ for all x.

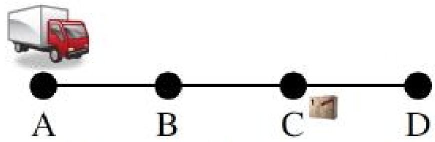
Content of Tables T_i^{add}:

i	$at(Sy)$	$at(Ad)$	$at(Br)$	$at(Pe)$	$at(Da)$	$v(Sy)$	$v(Ad)$	$v(Br)$	$v(Pe)$	$v(Da)$
0	0	∞	∞	∞	∞	0	∞	∞	∞	∞
1	0	1.5	1	∞	∞	0	1.5	1	∞	∞
2	0	1.5	1	5	5.5	0	1.5	1	5	5.5
3	0	1.5	1	5	5.5	0	1.5	1	5	5.5

$\rightarrow h^{add}(I) = 1.5 + 1 + 5 + 5.5 = 13 > 10 = h^+(I)$. But $< 20 = h^*(I)$.

$\rightarrow h^{add}(I) > h^+(I)$ because it counts the cost of $drive(Sy, Ad)$ 3 times:
As part of $h^{add}(I, \{v(Ad)\})$, $h^{add}(I, \{v(Pe)\})$, and $h^{add}(I, \{v(Da)\})$!

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}: (Table content T_i^1, where different, given 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 (4)	0	7 (4)
5	0	1	2	3	3	4	5 (4)	0	7 (4)

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

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

$\rightarrow$ So, what if $G = \{t(D), p(D)\}$? $h^{add}(I) = 10 > 5 = h^*(I) = h^+(I)$ because now $dr(A, B), dr(B, C), dr(C, D)$ is counted also as part of the goal $t(D)$.

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

- Both h^{add} and h^{max} can be computed reasonably quickly.
- h^{max} is admissible, but is typically far too optimistic.
- h^{add} is not admissible, but is typically a lot more informed than h^{max}.
- h^{add} is sometimes better informed than h^+, but for the "wrong reasons": rather than accounting for deletes, it overcounts by ignoring positive interactions, i.e., sub-plans shared between sub-goals.
- Such overcounting can result in dramatic over-estimates of h^*!!

$\rightarrow$ On slide 28 with goal $t(D)$, if we have 100 packages at C that need to go to D, what is $h^{add}(I)$? $703 >> 203 = h^*(I) = h^+(I)$: For every package, a count of 7 which includes $dr(A, B), dr(B, C)$ for getting the package into the truck, and $dr(A, B), dr(B, C), dr(C, D)$ for getting the truck to D.

Relaxed Plans: Reduce over-counting

$\rightarrow$ First compute a best-supporter function bs, which for every fact $p \in F$ 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 sub-goals 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 the sum of $c(a)$ and the cost estimate for pre_a.

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

The $h^{\max}$ supporter function $bs_s^{\max} : \{p \in F \mid 0 < h^{\max}(s, \{p\}) < \infty\} \mapsto A$ 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 $bs_s^{\text{add}} : \{p \in F \mid 0 < h^{\text{add}}(s, \{p\}) < \infty\} \mapsto A$ 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

Yields 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 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 $RPlan$

$\rightarrow$ Starting with the top-level goals, iteratively close open singleton sub-goals by selecting the best supporter.

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

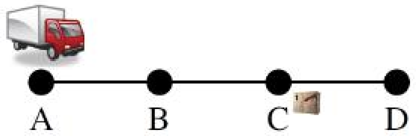
But is it correct?
$\rightarrow$ What if $g \not\in add_{bs(g)}$? Doesn't make sense. $\rightarrow$ Prerequisite (A).
$\rightarrow$ What if $bs(g)$ is undefined? Runtime error. $\rightarrow$ Prerequisite (B).
$\rightarrow$ What if the support for g eventually requires g itself as a precondition? Then this does not actually yield a relaxed plan. $\rightarrow$ Prerequisite (C).

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

Definition (Best-Supporter Function). *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task, and let s be a state. A best-supporter function for s is a partial function $bs : (F \setminus s) \mapsto A$ such that $p \in add_a$ whenever $a = bs(p)$.*
The support graph of bs is the directed graph with vertices $F \cup A$ and arcs $\{(p, a) \mid p \in pre_a\} \cup \{(a, p) \mid a = bs(p)\}$. We say that bs is closed if $bs(p)$ is defined for every $p \in (F \setminus s)$ 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 $a = bs(p)$": Prerequisite (A).
- bs is closed: Prerequisite (B).
- bs is well-founded: Prerequisite (C).

→ Intuition for (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.



- 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.

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

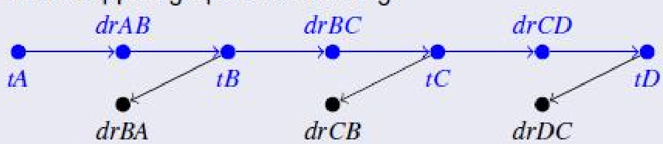
→ What if $G = \{t(D), p(D)\}$? $h^{\text{FF}}(I) = 5 = h^+(I) = h^*(I)$ because relaxed plan extraction selects the drive actions only once. By contrast, $h^{\text{add}}(I) = 10$ overcounts these actions, cf. slide 28.

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:

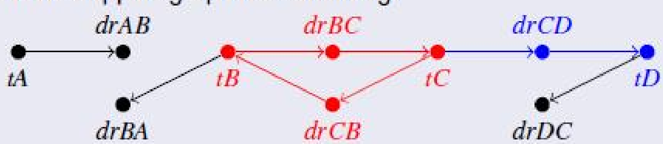


How to NOT do it (not well-founded)

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:



Proposition. *Let $\Pi = (F, A, c, I, G)$ 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 $h^{\text{max}}(s) < \infty$, it is easy to see that bs_s^{max} is closed (details omitted). If $a = bs_s^{\text{max}}(p)$, then a is the action yielding $0 < h^{\text{max}}(s, \{p\}) < \infty$ in the h^{max} equation. Since $c(a) > 0$, we have $h^{\text{max}}(s, pre_a) < h^{\text{max}}(s, \{p\})$ and thus, for all $q \in pre_a$, $h^{\text{max}}(s, \{q\}) < h^{\text{max}}(s, \{p\})$. Transitively, if the support graph contains a path from fact vertex r to fact vertex t, then $h^{\text{max}}(s, \{r\}) < h^{\text{max}}(s, \{t\})$. Thus there can't be cycles in the support graph and bs_s^{max} is well-founded. Similar for bs_s^{add}.

Proposition. *Let $\Pi = (F, A, c, I, G)$ 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 $bs(p)$, necessarily ordered before a_2. Iterating the argument shows that $\vec{a}^+$ is a relaxed plan for s.

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: If a relaxed plan exists, then there also exists a closed well-founded best-supporter function, see previous slide.

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

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

→ Relaxed plan heuristics have the same theoretical properties as h^{add}.

So what's the point?

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

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 $RPlan$.*

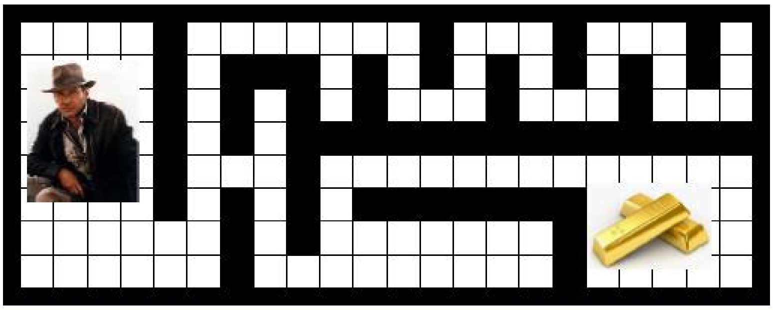
- Expanding only helpful actions does not guarantee completeness.
- Other planners use helpful actions as preferred operators, expanding first nodes resulting from helpful actions.

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}; it typically does not over-estimate h^*.

题目

Quiz



Question!

In this domain, h^+ is equal to?

(A): Manhattan Distance.

(B): h^*.

(C): Horizontal distance.

(D): Vertical distance.

→ (A): No, relaxed plans can't walk through walls. (B): Yes, optimal plan = shortest path = relaxed plan (deletes do not matter because "shortest paths never walk back"). (C), (D): No, relaxed plans must move both horizontally and vertically.

Question!

How does ignoring delete lists simplify FreeCell?

(A): You can move all cards immediately to their goal.

(B): Free cells remain free.

→ (A): No, we don't get any new moves in the relaxation. (B): Yes, when putting a card into a free cell, it's still free for another card.

Question!

How does ignoring delete lists simplify Sokoban?

(A): Free positions remain free.

(B): You can walk through walls.

(C): You can push 2 stones to same position.

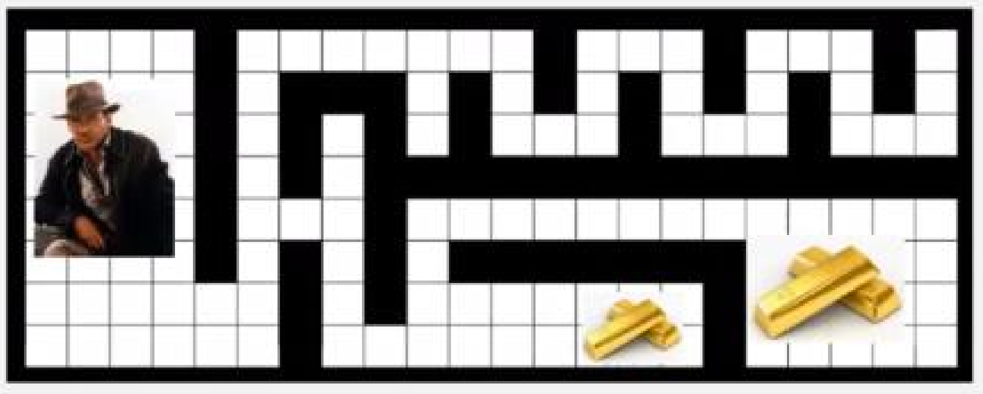
(D): Nothing ever becomes blocked.

→ (A), (C), (D): Yes (similar to above). (B): No, we don't get any new moves.

Question 1

1 / 1 pts

In this domain with 2 goals instead of 1, h+ is equal to?



- Manhattan Distance
- hmax
- h*
- Correct! Minimum Spanning Tree Distance

Question 2

1 / 1 pts

A delete relaxed plan solving the TSP can have more than one drive(Sydney, Adelaide) action

- True
- Correct! False

Question 5

1 / 1 pts

Which statement is correct?

Correct!

- ◉ Always hmax <= h+ and sometimes h* <= hadd
- ○ Sometimes hmax <= h+ and always h* <= hadd
- ○ Sometimes hmax <= h+ and sometimes h* <= hadd
- ○ Always hmax <= h+ and always h* <= hadd