AI Planning for Autonomy

5. Delete Relaxation Heuristics

It's a Long Way to the Goal, But How Long Exactly?
Part I: *Acting As If the World Can Only Get Better*

Chris Ewin & Tim Miller

THE UNIVERSITY OF MELBOURNE

With slides by Nir Lipovetsky

Agenda

Agenda

Motivation: For Planning Systems

Remember?

Imagine . . .

You are a planning system.

Somebody: *Hey you, here's that PDDL input. Solve it.*

You (thinking): *Hm, the only method I have is heuristic search.*

You (thinking): *Hm, I need a heuristic function.*
[Note: You programmer is presently in Honolulu.]

Motivation: For Planning Systems

Remember?

Imagine . . .

You are a planning system.

Somebody: *Hey you, here's that PDDL input. Solve it.*

You (thinking): *Hm, the only method I have is heuristic search.*

You (thinking): *Hm, I need a heuristic function.*
[Note: You programmer is presently in Honolulu.]

You: I already tried goal counting, and it performed badly. *What now?!?*

Motivation

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

$\rightarrow$ Every h yields good performance only in some domains! (Search reduction vs. computational overhead)
$\rightarrow$ We must come up with as many alternative methods as possible!

We cover the 4 different methods currently known:

- Critical path heuristics:
- Delete relaxation. Soon to be Done.
- Abstractions.
- Landmarks.

$\rightarrow$ Delete relaxation is very wide-spread, and highly successful for satisficing planning!

We introduce the method in STRIPS.

Agenda

Reminder: Relaxing the World by Ignoring Delete Lists

"What was once true remains true forever."

Reminder: Relaxing the World by Ignoring Delete Lists

"What was once true remains true forever."

Real world: (before)

Reminder: Relaxing the World by Ignoring Delete Lists

"What was once true remains true forever."

Real world: (after)

Reminder: Relaxing the World by Ignoring Delete Lists

"What was once true remains true forever."

Relaxed world: (before)

Reminder: Relaxing the World by Ignoring Delete Lists

"What was once true remains true forever."

Relaxed world: (after)

Reminder: Relaxing the World by Ignoring Delete Lists

"What was once true remains true forever."

Real world: (before)

Reminder: Relaxing the World by Ignoring Delete Lists

"What was once true remains true forever."

Real world: (after)

Reminder: Relaxing the World by Ignoring Delete Lists

"What was once true remains true forever."

Relaxed world: (before)

Reminder: Relaxing the World by Ignoring Delete Lists

"What was once true remains true forever."

Relaxed world: (after)

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 $del_{a^+} :=$*

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 $\vec{a} = \langle a_1, \ldots, a_n \rangle$ of STRIPS actions, by $\vec{a}^+$ we denote the corresponding sequence of relaxed actions, $\vec{a}^+ := \langle a_1^+, \ldots, a_n^+ \rangle$.*

(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.*

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 $\vec{a} = \langle a_1, \ldots, a_n \rangle$ of STRIPS actions, by $\vec{a}^+$ we denote the corresponding sequence of relaxed actions, $\vec{a}^+ := \langle a_1^+, \ldots, a_n^+ \rangle$.*

(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.)

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 $\vec{a} = \langle a_1, \ldots, a_n \rangle$ of STRIPS actions, by $\vec{a}^+$ we denote the corresponding sequence of relaxed actions, $\vec{a}^+ := \langle a_1^+, \ldots, a_n^+ \rangle$.*

(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?

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 $\vec{a} = \langle a_1, \ldots, a_n \rangle$ of STRIPS actions, by $\vec{a}^+$ we denote the corresponding sequence of relaxed actions, $\vec{a}^+ := \langle a_1^+, \ldots, a_n^+ \rangle$.*

(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:** $\{at(Sy), v(Sy)\}$.

A Relaxed Plan for "TSP" in Australia

1. **Initial state:** $\{at(Sy), v(Sy)\}$.
2. **Apply** $drive(Sy, Br)^+$**:**

A Relaxed Plan for "TSP" in Australia

1. **Initial state:** $\{at(Sy), v(Sy)\}$.
2. **Apply** $drive(Sy, Br)^+$**:** $\{at(Br), v(Br), at(Sy), v(Sy)\}$.

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** $drive(Sy, Ad)^+$**:**

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** $drive(Sy, Ad)^+$**:** $\{at(Ad), v(Ad), at(Br), v(Br), at(Sy), v(Sy)\}$.

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** $drive(Ad, Pe)^+$**:**

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** $drive(Ad, Pe)^+$**:** $\{at(Pe), v(Pe), at(Ad), v(Ad), at(Br), v(Br), at(Sy), v(Sy)\}$.

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** $drive(Ad, Da)^+$**:**

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** $drive(Ad, Da)^+$**:** $\{at(Da), v(Da), at(Pe), v(Pe), at(Ad), v(Ad), at(Br), v(Br), at(Sy), v(Sy)\}$.

State Dominance

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?

State Dominance

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.

State Dominance

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.

The Delete Relaxation and State Dominance

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^+.

The Delete Relaxation and State Dominance

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.

The Delete Relaxation and State Dominance

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?

The Delete Relaxation and State Dominance

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 $\vec{a}^+$

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!

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 heuristic safe?

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 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?

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 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?

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 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.

h^+: 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.*

h^+: 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.)

h^+: 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.]

h^+ in "TSP" in Australia

- 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:

h^+ in "TSP" in Australia

- 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:

h^+ in "TSP" in Australia

- 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: $\langle drive(Sy, Br), drive(Sy, Ad), drive(Ad, Pe), drive(Ad, Da)\rangle$.
- $h^*(I) =$

h^+ in "TSP" in Australia

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

h^+ in "TSP" in Australia

- 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.

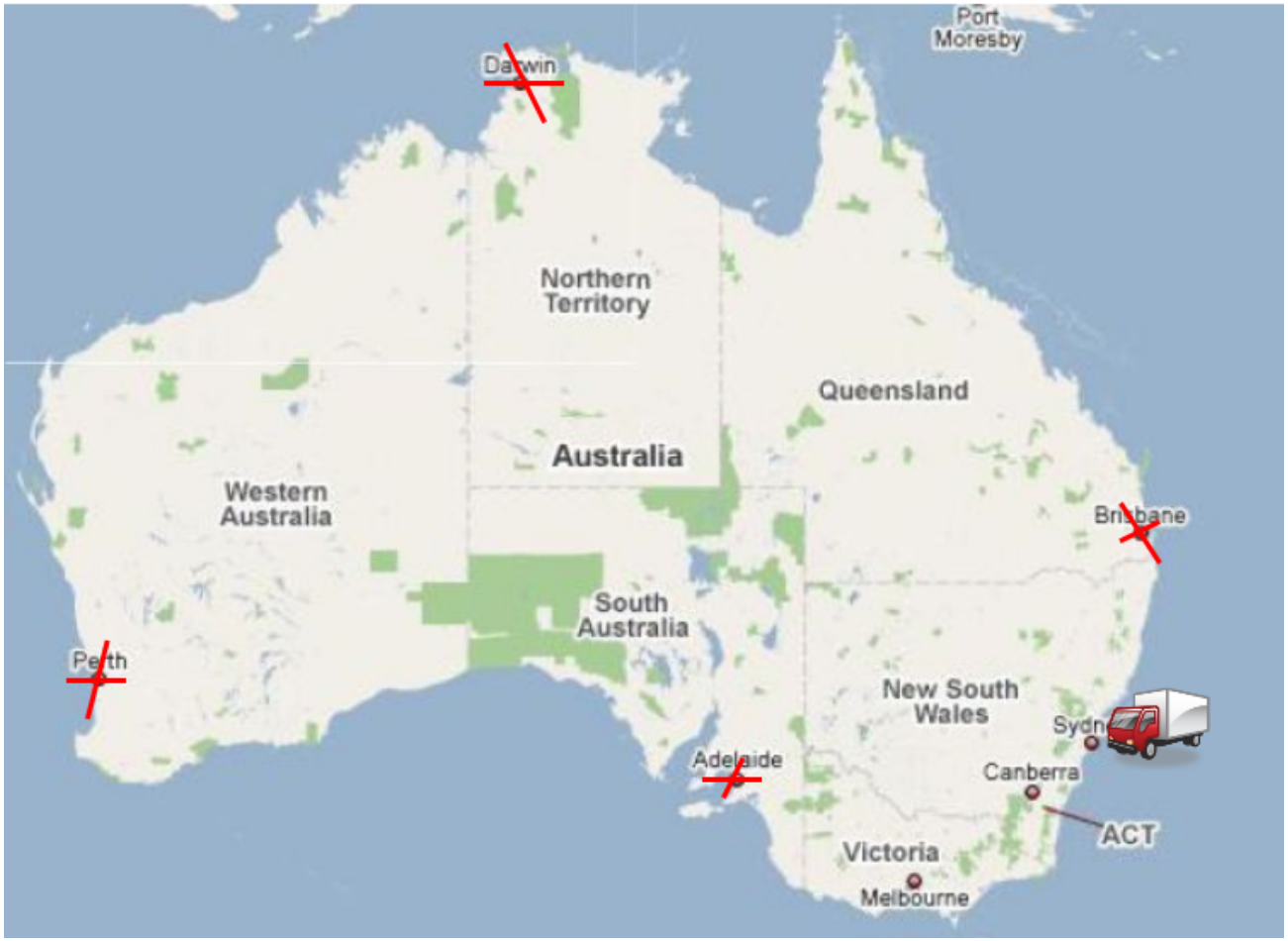
Planning vs. Relaxed Planning:

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

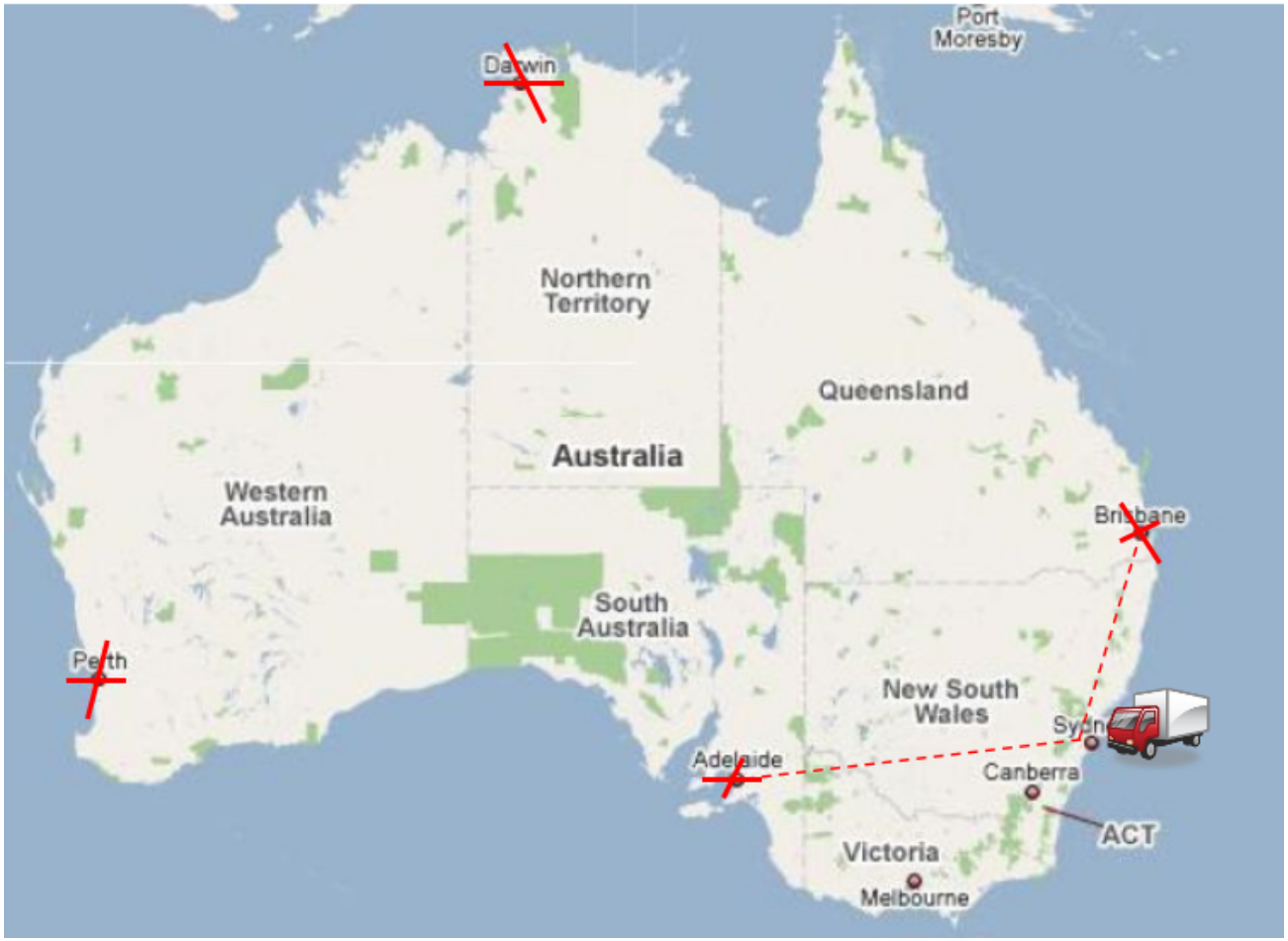
Reminder: h^+ in (the real) TSP

Reminder: h^+ in (the real) TSP

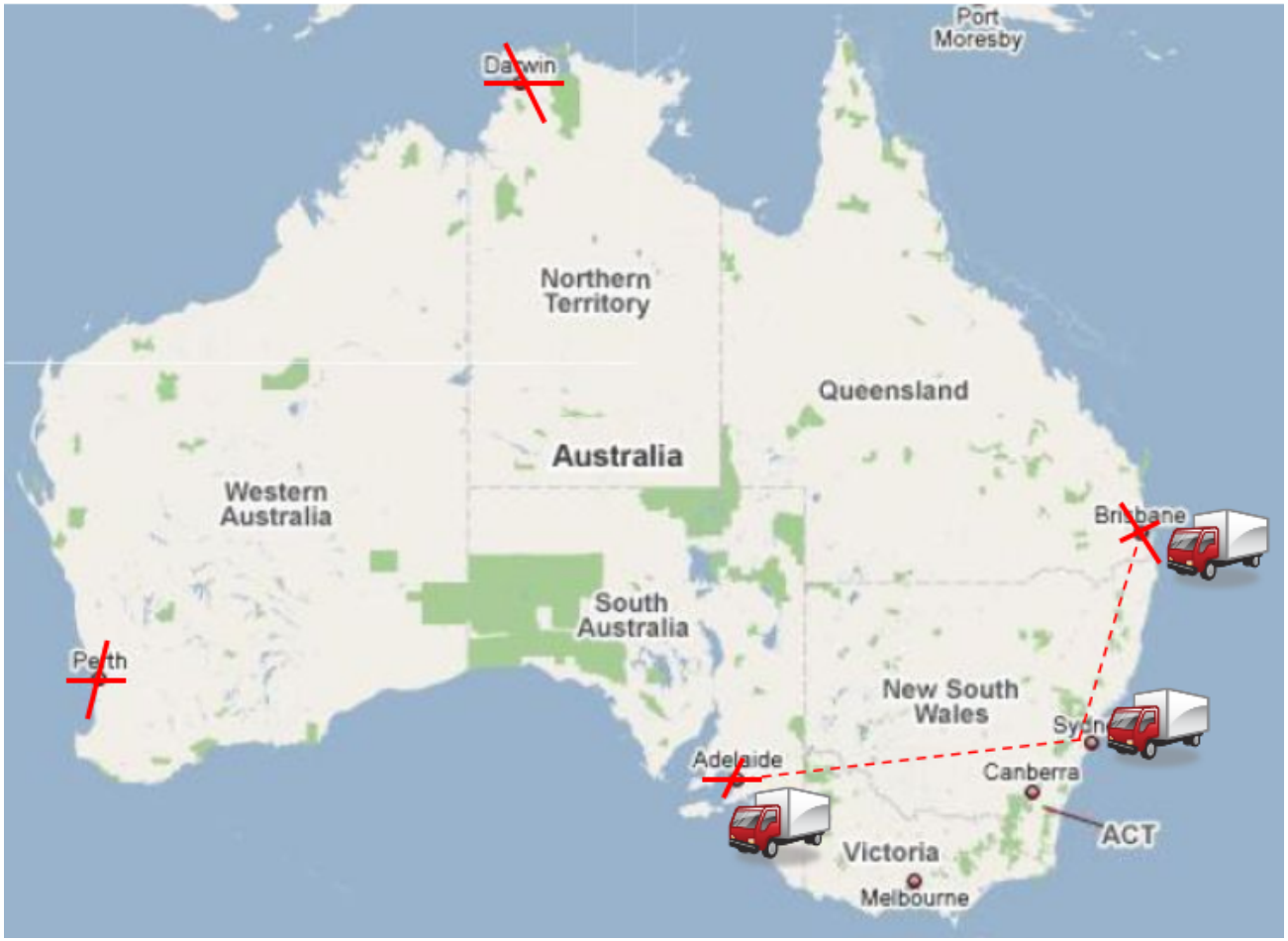
Reminder: h^+ in (the real) TSP



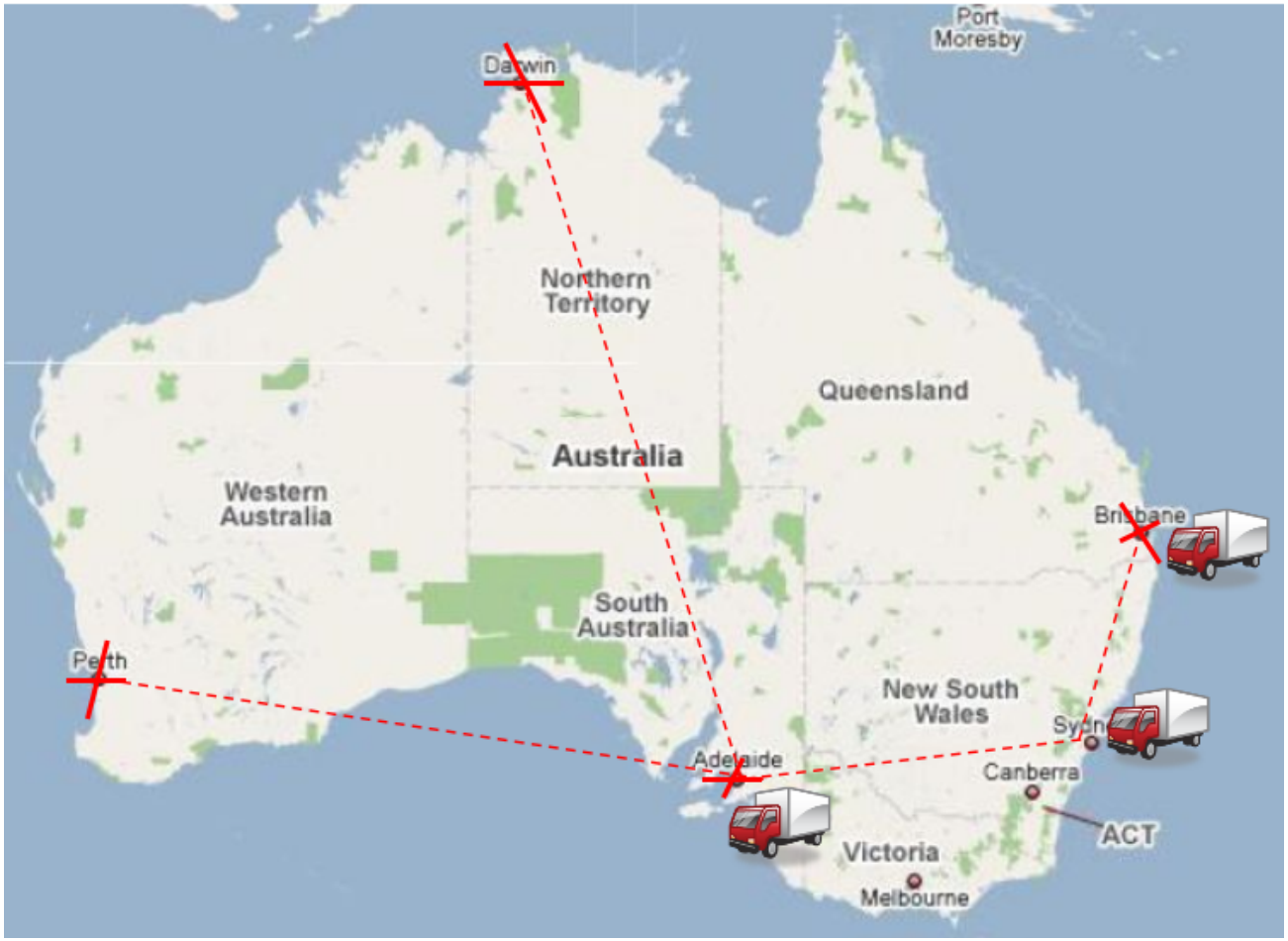
Reminder: h^+ in (the real) TSP



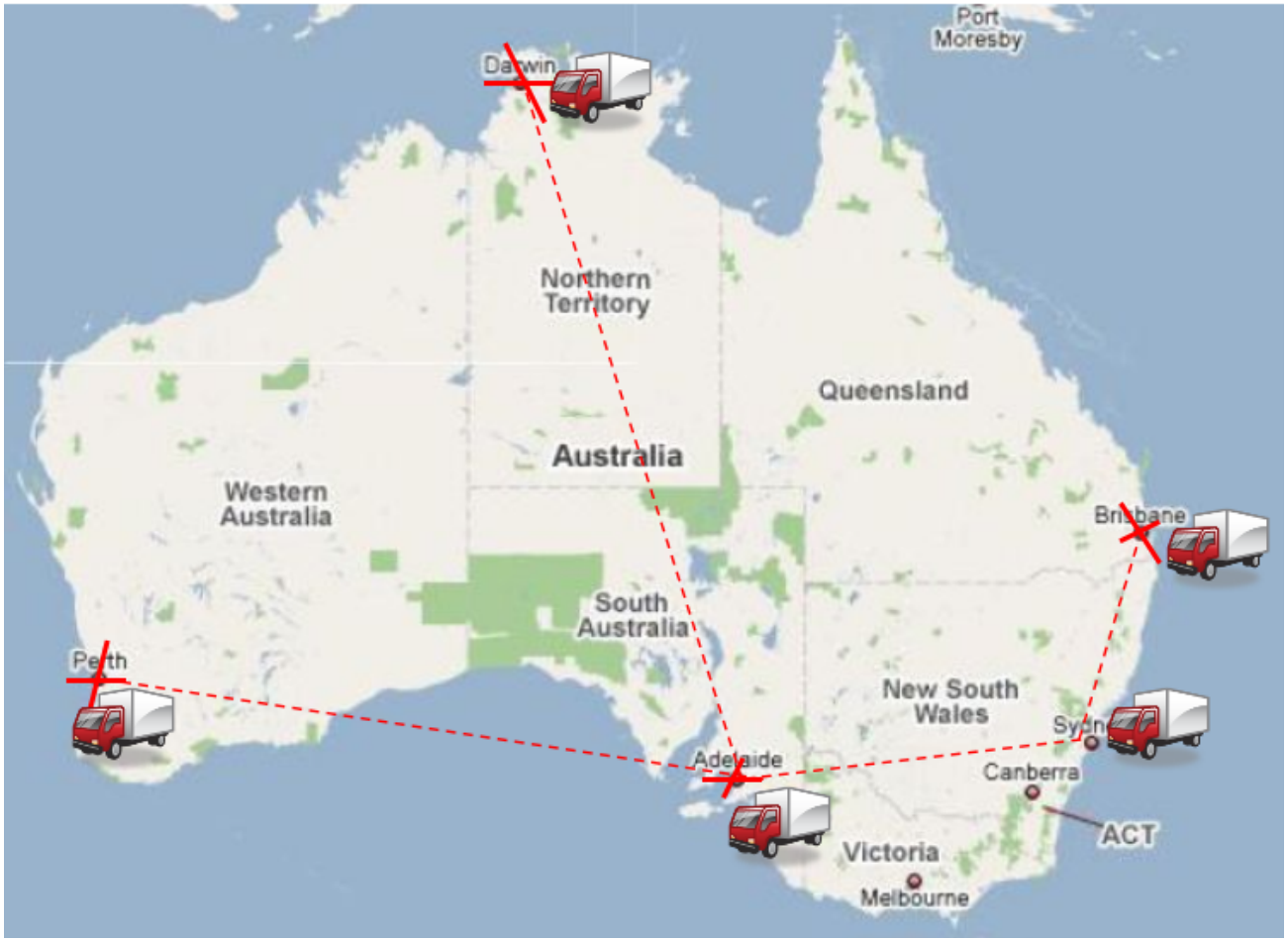
Reminder: h^+ in (the real) TSP



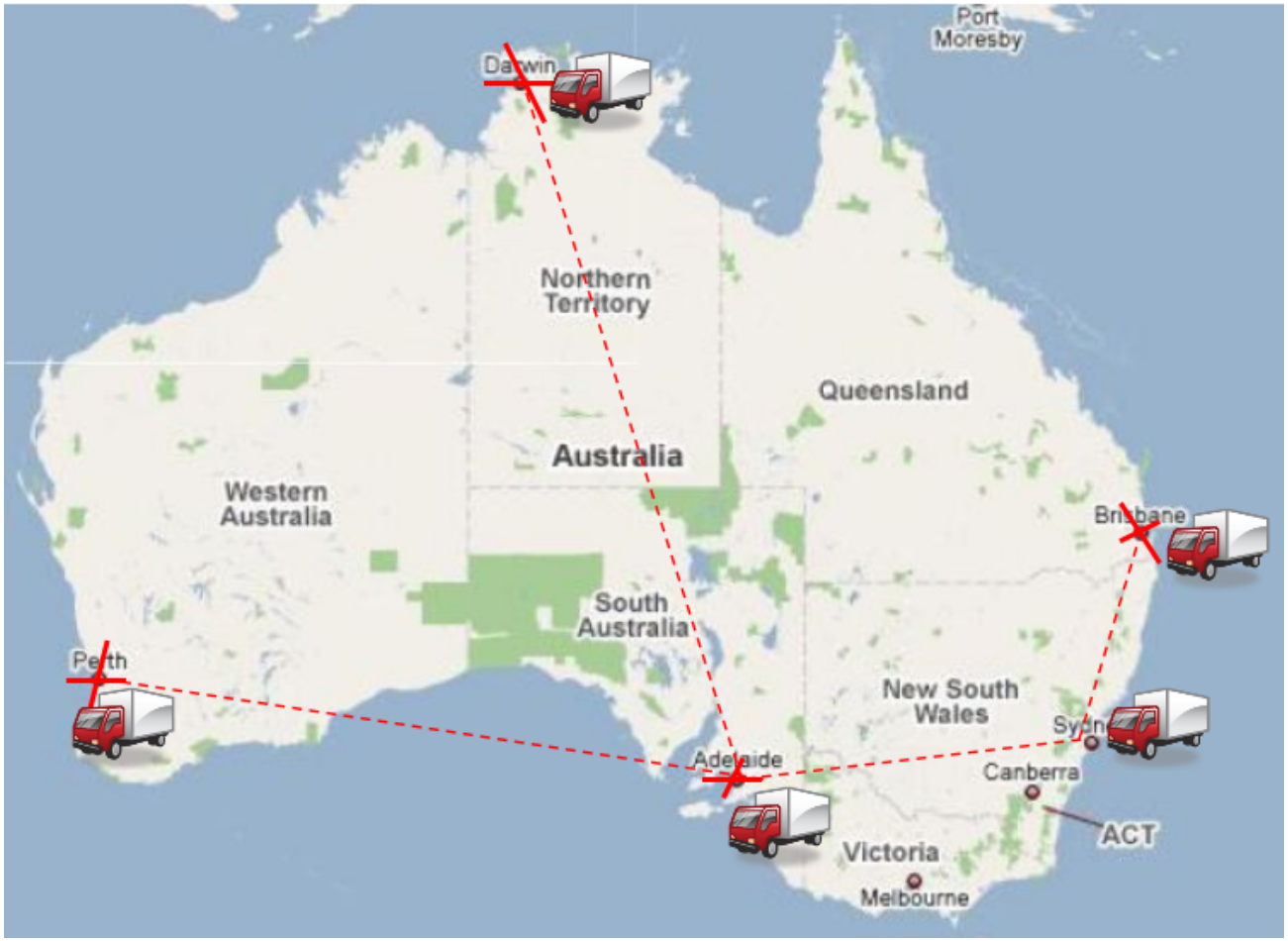
Reminder: h^+ in (the real) TSP



Reminder: h^+ in (the real) TSP

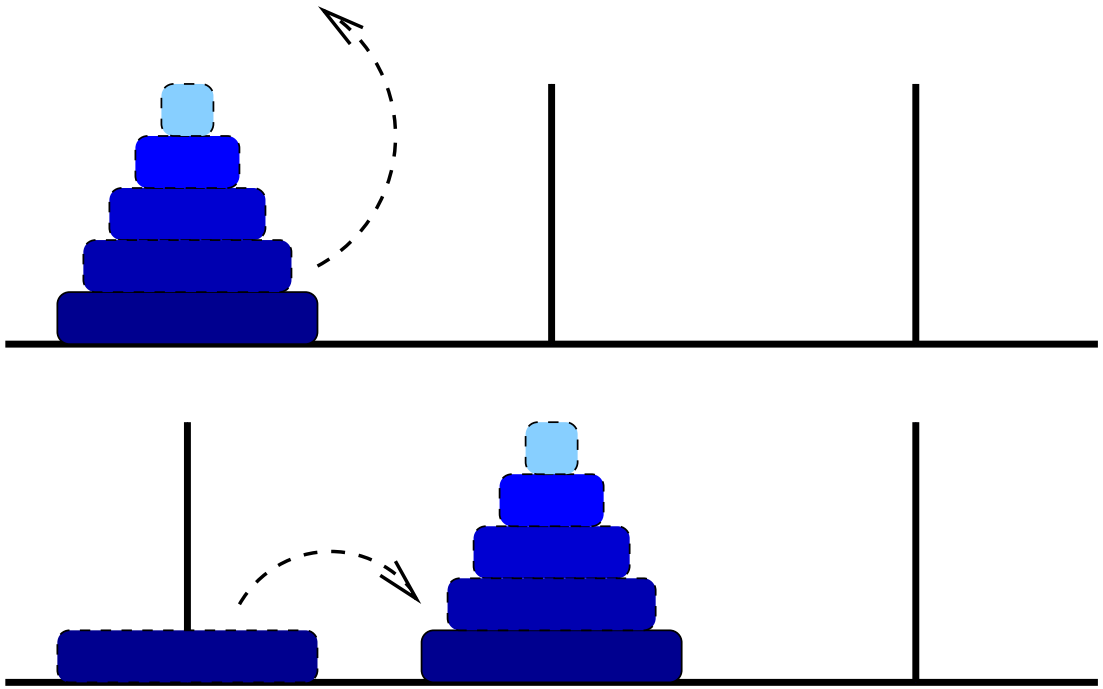


Reminder: h^+ in (the real) TSP



$h^+(\text{TSP}) = \text{Minimum Spanning Tree!}$

Reminder: h^+ in Hanoi



$h^+(\text{Hanoi}) = n$, not 2^n

But How to Compute h^+?

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$^+$.

But How to Compute h^+?

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. Membership:

But How to Compute h^+?

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. Membership: Guess action sequences of length $|A|$ – in a relaxed plan, each action is applied at most once!

But How to Compute h^+?

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$^+$.

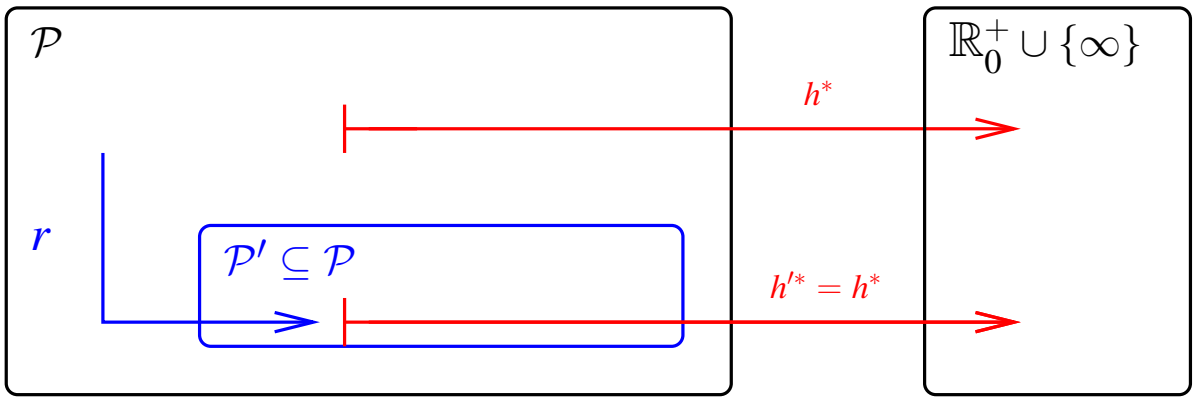
Theorem (Optimal Relaxed Planning is Hard). *PlanOpt$^+$ is* **NP***-complete.*

Proof. Membership: Guess action sequences of length $|A|$ – in a relaxed plan, each action is applied at most once!

Hardness: By reduction from SAT.

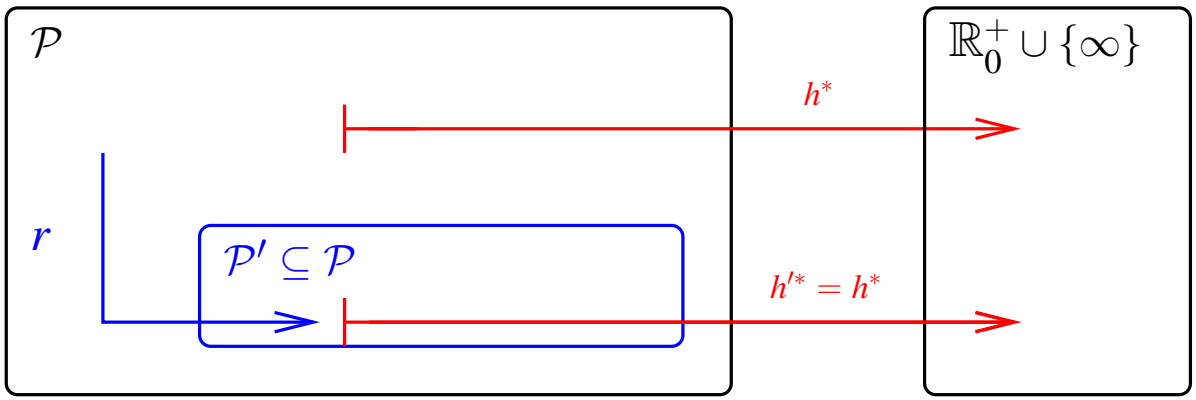
- For each variable $v_i \in \{v_1, \ldots, v_m\}$ in the CNF, three facts v_i, $notv_i$, and $setv_i$; for each clause $c_j \in \{c_1, \ldots, c_n\}$ in the CNF, one fact $satc_j$.
- Actions $setvtrue_i$: $(\emptyset, \{v_i, setv_i\}, \emptyset)$ and $setvfalse_i$: $(\emptyset, \{notv_i, setv_i\}, \emptyset)$.
- Actions $makesatc_j$: $(\{v_i\}, \{satc_j\}, \emptyset)$ where v_i appears positively in clause c_j; $(\{notv_i\}, \{satc_j\}, \emptyset)$ where v_i appears negatively in clause c_j.
- Initial state $\emptyset$, goal $\{setv_1, \ldots, setv_m, satc_1, \ldots, satc_n\}$; $B := m + n$.

h^+ as a Relaxation Heuristic



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

h^+ as a Relaxation Heuristic



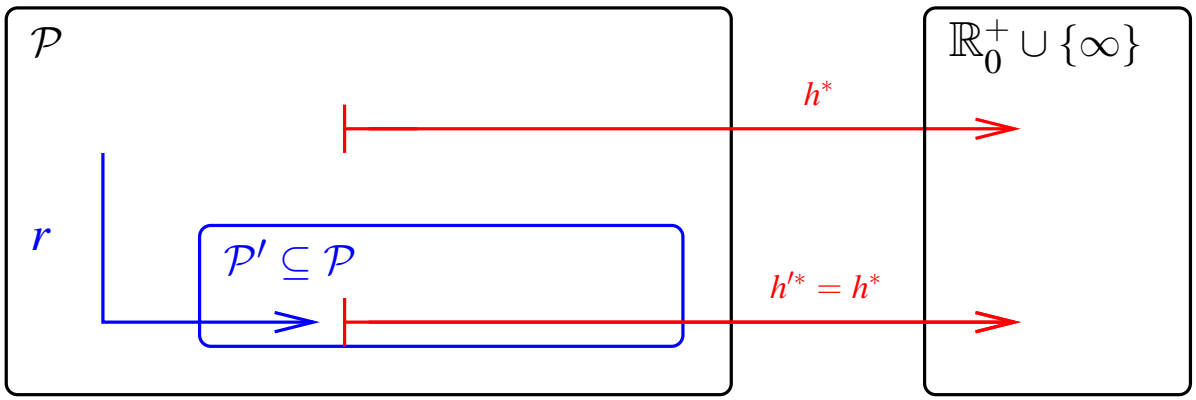
where, for all $\Pi \in \mathcal{P}$, $h^*(r(\Pi)) \leq h^*(\Pi)$.

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.

$\rightarrow$ Is this a native relaxation?

h^+ as a Relaxation Heuristic



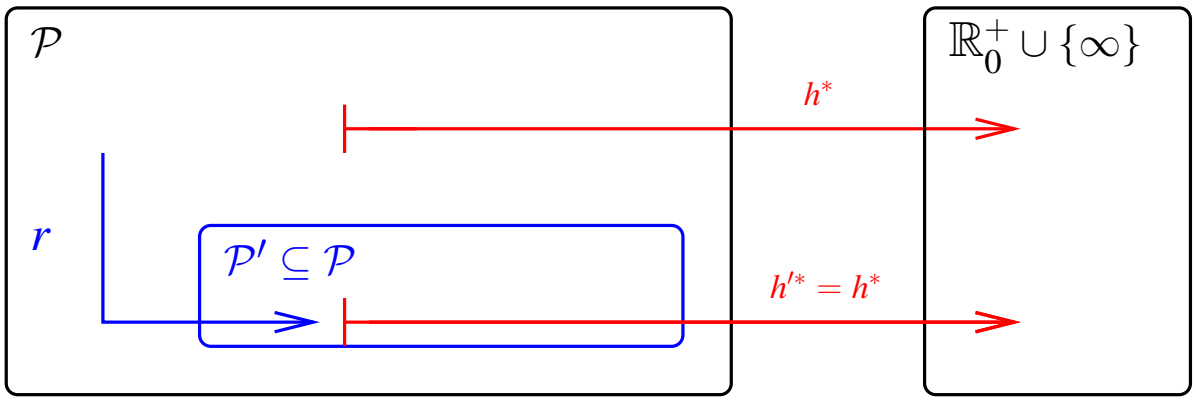
where, for all $\Pi \in \mathcal{P}$, $h^*(r(\Pi)) \leq h^*(\Pi)$.

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.

$\rightarrow$ Is this a native relaxation? Yes.

h^+ as a Relaxation Heuristic



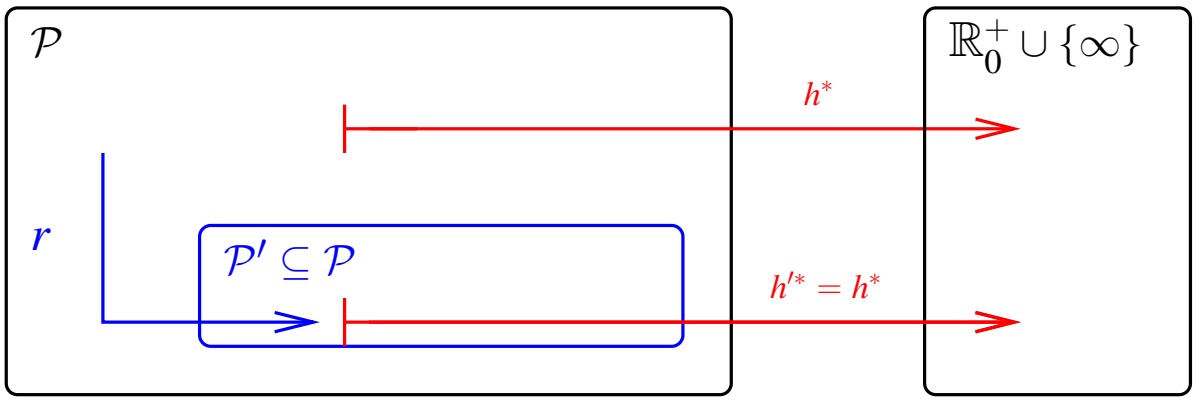
where, for all $\Pi \in \mathcal{P}$, $h^*(r(\Pi)) \leq h^*(\Pi)$.

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.

$\rightarrow$ Is this a native relaxation? Yes.
$\rightarrow$ Is this relaxation efficiently constructible?

h^+ as a Relaxation Heuristic



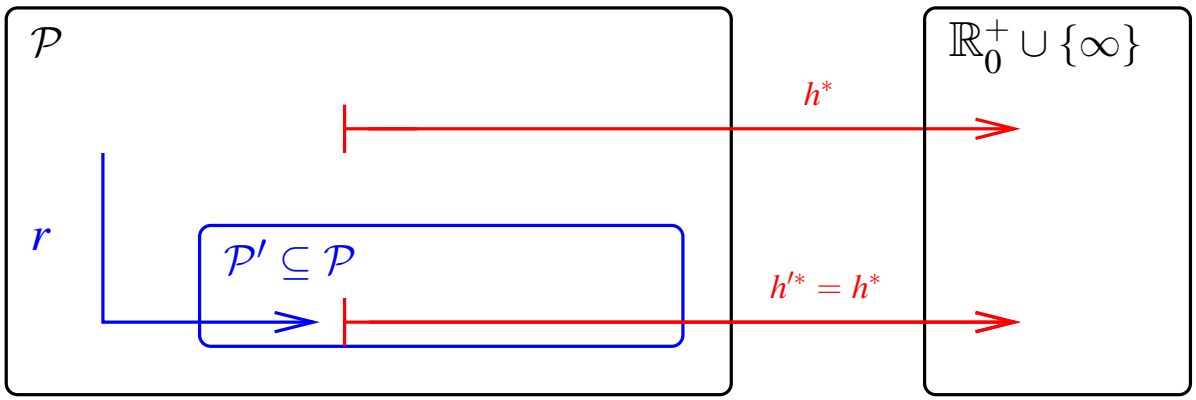
where, for all $\Pi \in \mathcal{P}$, $h^*(r(\Pi)) \leq h^*(\Pi)$.

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.

$\rightarrow$ Is this a native relaxation? Yes.
$\rightarrow$ Is this relaxation efficiently constructible? Yes.

h^+ as a Relaxation Heuristic



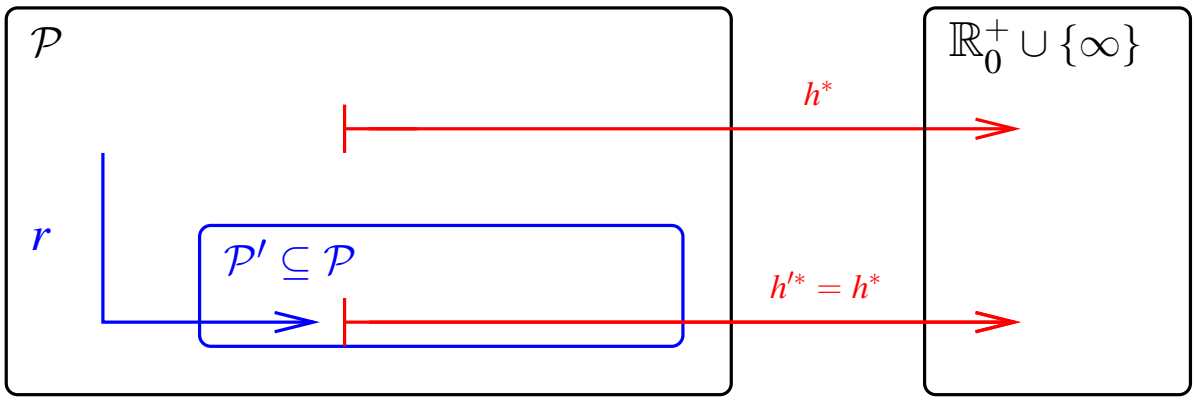
where, for all $\Pi \in \mathcal{P}$, $h^*(r(\Pi)) \leq h^*(\Pi)$.

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.

$\rightarrow$ Is this a native relaxation? Yes.
$\rightarrow$ Is this relaxation efficiently constructible? Yes.
$\rightarrow$ Is this relaxation efficiently computable?

h^+ as a Relaxation Heuristic



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.

$\rightarrow$ Is this a native relaxation? Yes.
$\rightarrow$ Is this relaxation efficiently constructible? Yes.
$\rightarrow$ Is this relaxation efficiently computable? No.

What shall we do with this relaxation?

Reminder: → **Lecture 4**

What if $\mathcal{R}$ is not efficiently computable?

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

What shall we do with this relaxation?

Reminder: → **Lecture 4**

What if $\mathcal{R}$ is not efficiently computable?

- Either (a) approximate h'^*, or (b) design h'^* in a way so that it will typically be feasible, or (c) just live with it and hope for the best.
- Many known relaxations (in planning) are efficiently computable, some aren't (like h^+). 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. We now look at the the most wide-spread approaches to do so.

Agenda

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 $h^{\text{max}}(s) := h^{\text{max}}(s, G)$ where $h^{\text{max}}(s, g)$ is the point-wise greatest function that satisfies $h^{\text{max}}(s, g) =$*

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

The Additive and Max Heuristics: Properties

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

The Additive and Max Heuristics: Properties

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

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

The Additive and Max Heuristics: Properties

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

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

$\rightarrow$ 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.

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 $h^{\max}(s) = \infty$ 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 $h^{\max}$ 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.)

Bellman-Ford for $h^{\max}$ 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	$at(Sy)$	$at(Ad)$	$at(Br)$	$at(Pe)$	$at(Da)$	$v(Sy)$	$v(Ad)$	$v(Br)$	$v(Pe)$	$v(Da)$

Bellman-Ford for $h^{\max}$ 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	∞	∞	∞	∞

Bellman-Ford for $h^{\max}$ 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	∞	∞

Bellman-Ford for $h^{\max}$ 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

Bellman-Ford for $h^{\max}$ 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

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

Bellman-Ford for $h^{\max}$ 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

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

Bellman-Ford for 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	$at(Sy)$	$at(Ad)$	$at(Br)$	$at(Pe)$	$at(Da)$	$v(Sy)$	$v(Ad)$	$v(Br)$	$v(Pe)$	$v(Da)$

Bellman-Ford for 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	∞	∞	∞	∞

Bellman-Ford for 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	∞	∞

Bellman-Ford for 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

Bellman-Ford for 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

$\rightarrow h^{\text{add}}(I) =$

Bellman-Ford for 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

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

Bellman-Ford for 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

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

$\rightarrow h^{\text{add}}(I) > h^+(I)$ because it counts the cost of $drive(Sy, Ad)$ 3 times:

Bellman-Ford for 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.

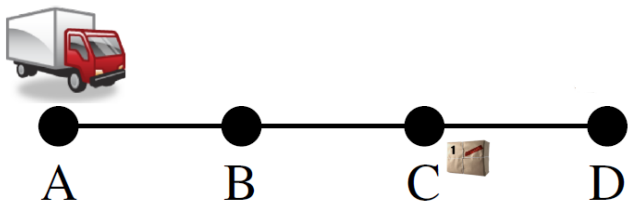
Content of Tables T_i^{add}:

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

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

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

Bellman-Ford for h^{add} in "Logistics"

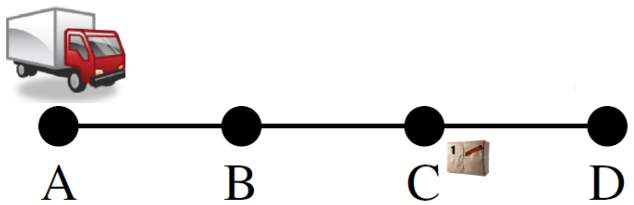


- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X, Y), lo(X), ul(X)$.

Content of Tables T_i^{add}: (Table content T_i^1, where different, given in red)

i	$t(A)$	$t(B)$	$t(C)$	$t(D)$	$p(T)$	$p(A)$	$p(B)$	$p(C)$	$p(D)$

Bellman-Ford for h^{add} in "Logistics"

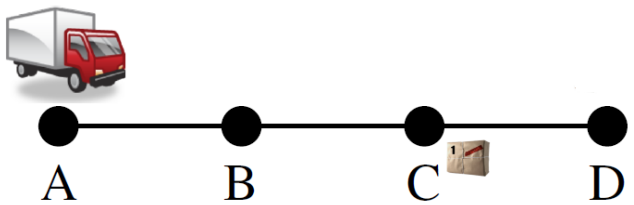


- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X, Y), lo(X), ul(X)$.

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	∞

Bellman-Ford for h^{add} in "Logistics"

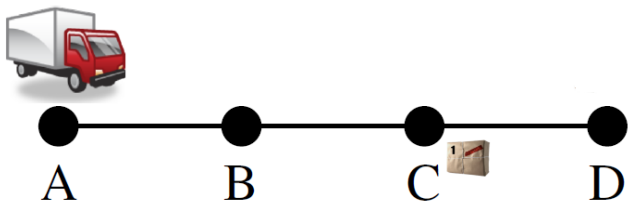


- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X, Y), lo(X), ul(X)$.

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	∞

Bellman-Ford for h^{add} in "Logistics"

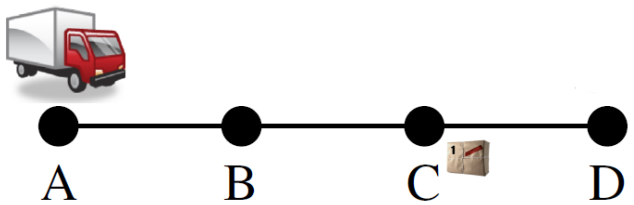


- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X, Y), lo(X), ul(X)$.

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	∞

Bellman-Ford for h^{add} in "Logistics"

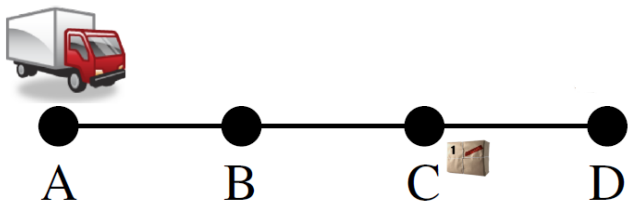


- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X, Y), lo(X), ul(X)$.

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	∞

Bellman-Ford for h^{add} in "Logistics"

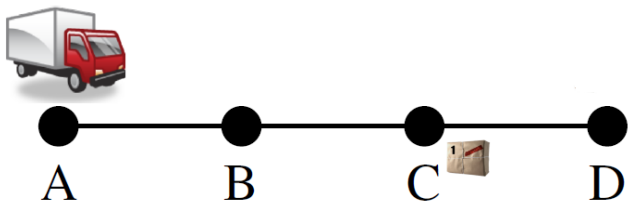


- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X, Y), lo(X), ul(X)$.

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)

Bellman-Ford for h^{add} in "Logistics"



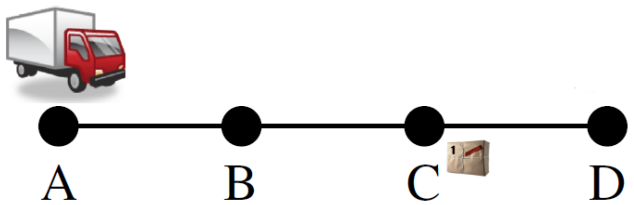
- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X, Y), lo(X), ul(X)$.

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)

$\rightarrow h^{\text{add}}(I) =$

Bellman-Ford for h^{add} in "Logistics"



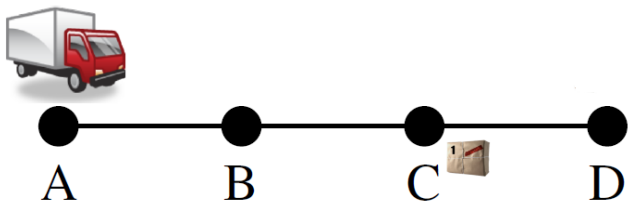
- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $dr(X, Y), lo(X), ul(X)$.

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)

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

Bellman-Ford for h^{add} in "Logistics"



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

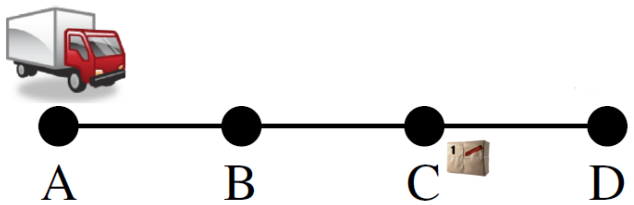
Content of Tables T_i^{add}: (Table content T_i^1, where different, given in red)

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)

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

$\rightarrow h^{\text{add}}(I) > h^+(I)$ because?

Bellman-Ford for h^{add} in "Logistics"



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

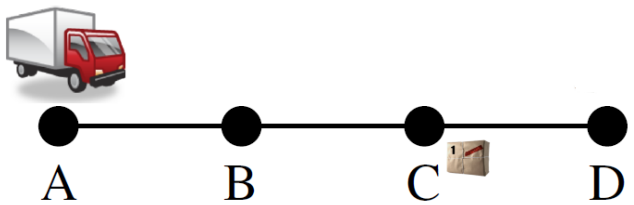
Content of Tables T_i^{add}: (Table content T_i^1, where different, given in red)

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)

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

$\rightarrow h^{\text{add}}(I) > h^+(I)$ 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 $p(D)$.

Bellman-Ford for 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)

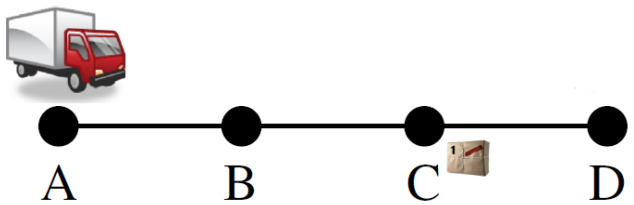
i	$t(A)$	$t(B)$	$t(C)$	$t(D)$	$p(T)$	$p(A)$	$p(B)$	$p(C)$	$p(D)$
0	0	∞	∞	∞	∞	∞	∞	0	∞
1	0	1	∞	∞	∞	∞	∞	0	∞
2	0	1	2	∞	∞	∞	∞	0	∞
3	0	1	2	3	3	∞	∞	0	∞
4	0	1	2	3	3	4	5 (4)	0	7 (4)
5	0	1	2	3	3	4	5 (4)	0	7 (4)

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

$\rightarrow h^{\text{add}}(I) > h^+(I)$ 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 $G = \{t(D), p(D)\}$? $h^{\text{add}}(I) =$

Bellman-Ford for 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)

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

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

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.
- 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 $h^{\text{add}}(I)$?

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.
- 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 $h^{\text{add}}(I)$? $703 >> 203 = h^*(I) = h^+(I)$: 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.

$\rightarrow$ Relaxed plans (up next) are a means to reduce this kind of over-counting.