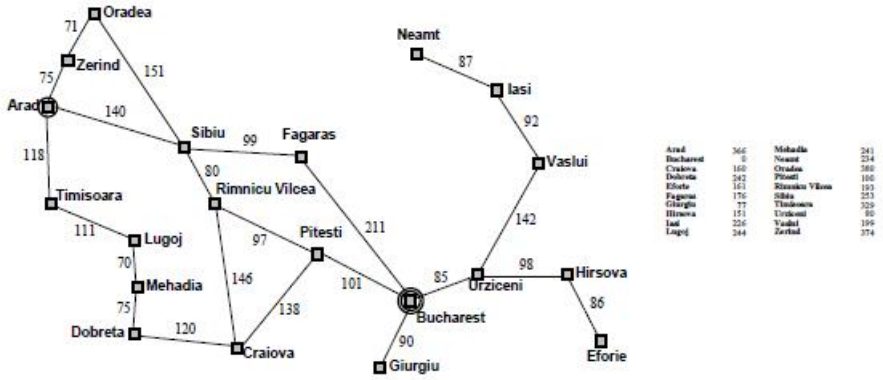
4 - Generating Heuristic Functions

知识点 & 题目

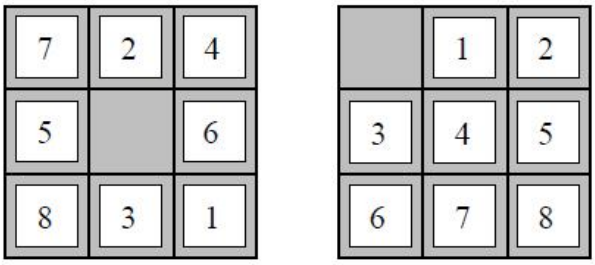
Relaxing is a methodology to construct heuristic functions.

- Relaxation means to simplify the problem, and take the solution to the simpler problem as the heuristic estimate for the solution to the actual problem.



How to derive straight-line distance by relaxation?

- Problem $\mathcal{P}$: Route finding.
- Simpler problem $\mathcal{P}'$: Route finding for birds.
- Perfect heuristic h'^* for $\mathcal{P}'$: Straight-line distance.
- Transformation r: Pretend you're a bird.



Start State Goal State

Perfect heuristic h^* for $\mathcal{P}$: Actions = "A tile can move from square A to square B if A is adjacent to B and B is blank."

- How to derive the Manhattan distance heuristic? $\mathcal{P}'$: Actions = "A tile can move from square A to square B if A is adjacent to B."
- How to derive the misplaced tiles heuristic? $\mathcal{P}'$: Actions = "A tile can move from square A to square B."
- h'^* (resp. r) in both: optimal cost in $\mathcal{P}'$ (resp. use different actions).
- Here: Manhattan distance = 18, misplaced tiles = 8.

- Propositions P: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Da\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Da\}$.
- Actions $a \in A$: $drive(x, y)$ where x, y have a road; $pre_a = \{at(x)\}$, $add_a = \{at(y), v(y)\}$, $del_a = \{at(x)\}$.
- Initial state I: $at(Sy), v(Sy)$.
- Goal G: $at(Sy), v(x)$ for all x.

Let's "act as if we could achieve each goal directly":
- Problem $\mathcal{P}$: All STRIPS planning tasks.
- Simpler problem $\mathcal{P}'$: All STRIPS planning tasks with empty preconditions and deletes.
- Perfect heuristic h'^* for $\mathcal{P}'$: Optimal plan cost ($= h^*$).
- Transformation r: Drop the preconditions and deletes.
- Heuristic value here? 4.

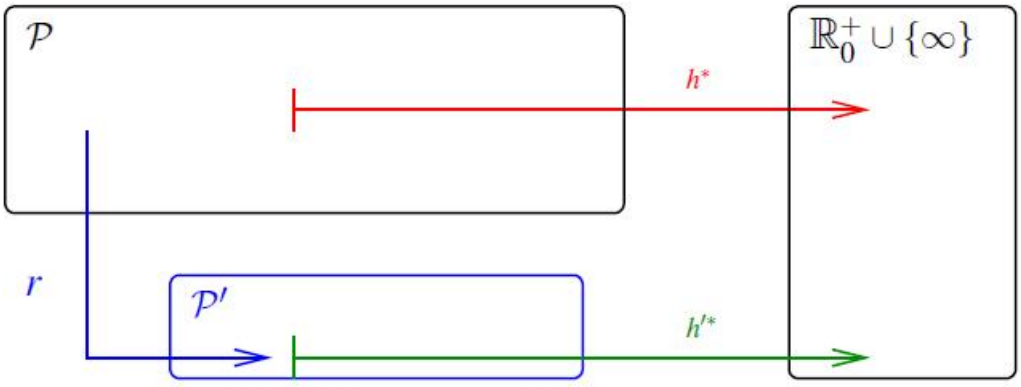
$\rightarrow$ Optimal STRIPS planning with empty preconditions and deletes is still **NP**-hard! (Reduction from MINIMUM COVER, of goal set by add lists.)

$\rightarrow$ Need to approximate the perfect heuristic h'^* for $\mathcal{P}'$. Hence goal counting: just approximate h'^* by number-of-false-goals.

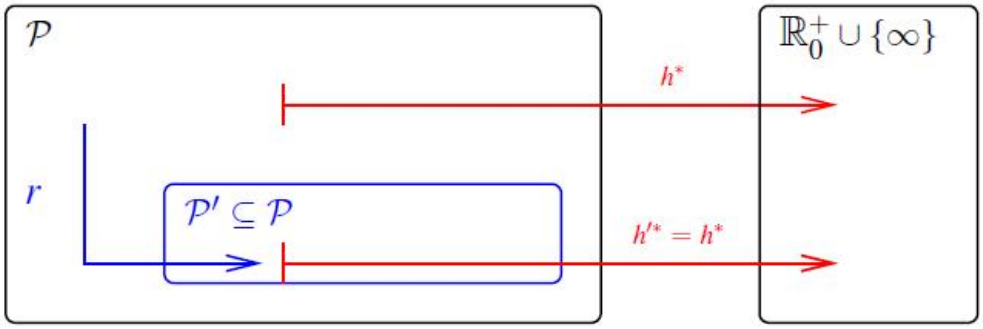
Relaxations

Definition (Relaxation). *Let $h^* : \mathcal{P} \mapsto \mathbb{R}_0^+ \cup \{\infty\}$ be a function. A relaxation of h^* is a triple $\mathcal{R} = (\mathcal{P}', r, h'^*)$ where $\mathcal{P}'$ is an arbitrary set, and $r : \mathcal{P} \mapsto \mathcal{P}'$ and $h'^* : P' \mapsto \mathbb{R}_0^+ \cup \{\infty\}$ are functions so that, for all $\Pi \in \mathcal{P}$, the relaxation heuristic $h^{\mathcal{R}}(\Pi) := h'^*(r(\Pi))$ satisfies $h^{\mathcal{R}}(\Pi) \leq h^*(\Pi)$. The relaxation is:*
- *native if $\mathcal{P}' \subseteq P$ and $h'^* = h^*$;*
- *efficiently constructible if there exists a polynomial-time algorithm that, given $\Pi \in \mathcal{P}$, computes $r(\Pi)$;*
- *efficiently computable if there exists a polynomial-time algorithm that, given $\Pi' \in \mathcal{P}'$, computes $h'^*(\Pi')$.*

Reminder:
- You have a problem, $\mathcal{P}$, whose perfect heuristic h^* you wish to estimate.
- You define a simpler problem, $\mathcal{P}'$, whose perfect heuristic h'^* can be used to (admissibly!) estimate h^*
- You define a transformation, r, from $\mathcal{P}$ into $\mathcal{P}'$.
- Given $\Pi \in \mathcal{P}$, you estimate $h^*(\Pi)$ by $h'^*(r(\Pi))$.

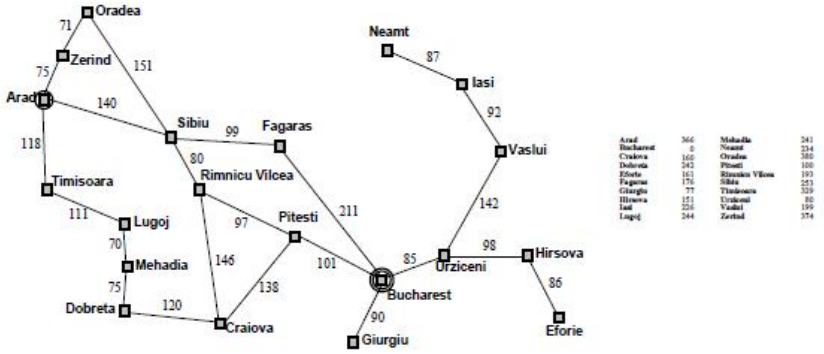


Example route-finding:
- Problem $\mathcal{P}$: Route finding.
- Simpler problem $\mathcal{P}'$: Route finding for birds.
- Perfect heuristic h'^* for $\mathcal{P}'$: Straight-line distance.
- Transformation r: Pretend you're a bird.



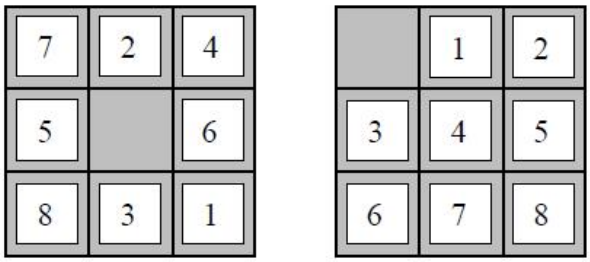
Example "goal-counting":

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



Relaxation $\mathcal{R} = (\mathcal{P}', r, h'^*)$**:** Pretend you're a bird.

- Native? No: Birds don't do route-finding. (Well, it's equivalent to trivial maps with direct routes between everywhere.)
- Efficiently constructible? Yes (pretend you're a bird).
- Efficiently computable? Yes (measure straight-line distance).



Relaxation $\mathcal{R} = (\mathcal{P}', r, h'^*)$**:** Use more generous actions rule to obtain Manhattan distance.

- Native? No: With the modified rules, it's not the "same puzzle" anymore. (Well, one could be generous in defining what the "same puzzle" is.)
- Efficiently constructible? Yes (exchange action set).
- Efficiently computable? Yes (count misplaced tiles/sum up Manhattan distances).

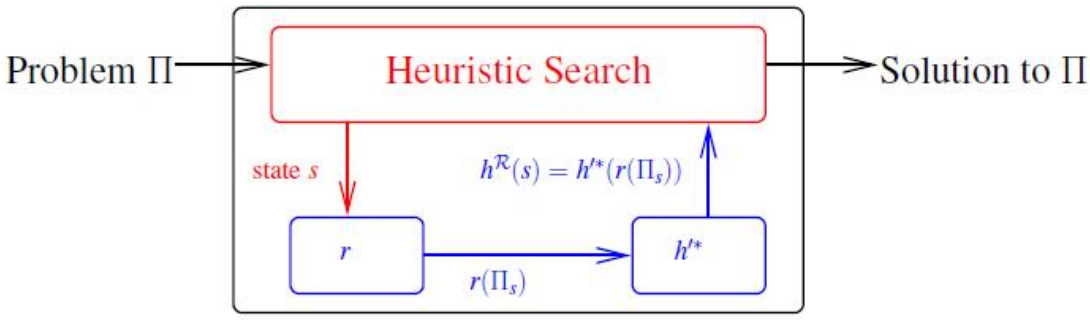
- Propositions P: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Da\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Da\}$.
- Actions $a \in A$: $drive(x, y)$ where x, y have a road; $pre_a = \{at(x)\}$, $add_a = \{at(y), v(y)\}$, $del_a = \{at(x)\}$.
- Initial state I: $at(Sy), v(Sy)$.
- Goal G: $at(Sy), v(x)$ for all x.

Relaxation $\mathcal{R} = (\mathcal{P}', r, h'^*)$: Remove preconditions and deletes, then use h^*.

- Native? Yes: Planning with empty preconditions and deletes is a special case of planning (i.e., a sub-class of $\mathcal{P}$).
- Efficiently constructible? Yes (drop preconditions and deletes).
- Efficiently computable? **No!** Optimal planning is still **NP**-hard in this case (MINIMUM COVER of goal set by add lists).

What shall we do with the relaxation? $\rightarrow$ Use method (a): Approximate h^* in $\mathcal{P}'$ by counting the number of goals not currently true.

Using a relaxation $\mathcal{R} = (\mathcal{P}', r, h'^*)$ during search:



$\rightarrow \Pi_s$: Π with initial state replaced by s, i.e., $\Pi = (F, A, c, I, G)$ changed to (F, A, c, s, G).

$\rightarrow$ The task of finding a plan for search state s.

How to Relax During Search: Illustration

- Goal-Counting L4 P21
  - The goal-counting approximation h ="count the number of goals currently not true" is a very **uninformative** heuristic function:
    - Range of heuristic values is small (0 - |G|).
    - We can transform any planning task into an equivalent one where h(s) = 1 for all non-goal states s.
      - How? Replace goal by new fact g and add a new action achieving g with precondition G.
    - Ignores almost all structure: Heuristic value does not depend on the actions at all.
    - Is h safe/goal-aware/admissible/consistent? Only **safe and goal-aware**.
- Ignoring Deletes L4 P22 More useful

Summary

- Relaxation is a method to compute heuristic functions.
- Given a problem P we want to solve, we define a relaxed problem P'.
    - We derive the heuristic by mapping into P' and taking the solution to this simpler problem as the heuristic estimate.
- Relaxations can be native, efficiently constructible, and/or efficiently computable.
    - None of this is a strict requirement to be useful.
- During search, the relaxation is used only inside the computation of the heuristic function on each state; the relaxation does not affect anything else.
    - This can be a bit confusing especially for native relaxations like ignoring deletes.

题目

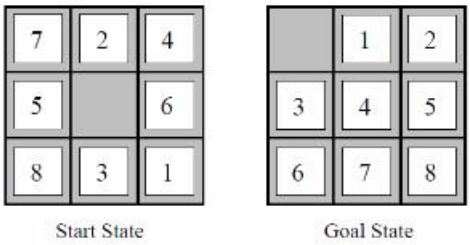
Quiz

Question 1

1 / 1 pts

For the 8-Puzzle problem, which of the following relaxations gives you the misplaced tiles heuristic?

Relaxation in the 8-Puzzle



Perfect heuristic h^* for $\mathcal{P}$: Actions = "A tile can move from square A to square B if A is adjacent to B and B is blank."

- How to derive the Manhattan distance heuristic? $\mathcal{P}'$: Actions = "A tile can move from square A to square B if A is adjacent to B."
- How to derive the misplaced tiles heuristic?

- Tiles can move from A to B if B is blank
- Tiles can move from A to B if A is adjacent to B
- Tils can move from A to B if A is adjacent to B and B is blank

Correct!

- Tiles can move from A to B with no restrictions

Question 2

1 / 1 pts

8-Puzzle relaxations are efficiently computable

Correct!

- True
- False

Question 3

1 / 1 pts

Removing preconditions and delete effects from PDDL planning problems yields a relaxation that is efficiently constructable

Correct!

- True

False

Question 4

0 / 1 pts

Removing preconditions and delete effects from PDDL planning problems yields a relaxation that is efficiently computable

You Answered

- True

Correct Answer

- False

In general no, even in the absence of preconditions and deletes this is still an NP-Hard problem. It's efficiently computable for the special case where the number of add effects is less than 3. Goal counting is an approximation of this relaxation which is efficiently computable.

Question 5

1 / 1 pts

Given the PDDL below, which precondition should be removed to relax 8-puzzle into Manhattan Distance

```
(define (domain PUZZLE)
  (:requirements :strips)
  (:predicates    (tile ?t)
        (position ?p)
        (adjacent ?from ?emp)
        (empty ?emp)
  )

  (:action move
    :parameters (?t ?from ?emp)
    :precondition (and (adjacent ?from ?emp) (at ?t ?from) (empty ?emp))
    :effects (and (not (at ?t ?from)) (not (empty ?emp)) (at ?t ?emp) (empty ?from))
```

- remove **adjacent** from preconditions

Correct!

- remove **empty** from preconditions
- remove **at** from preconditions
- None of the above