03 Heuristic Search

Table of contents

How to Relax Informally

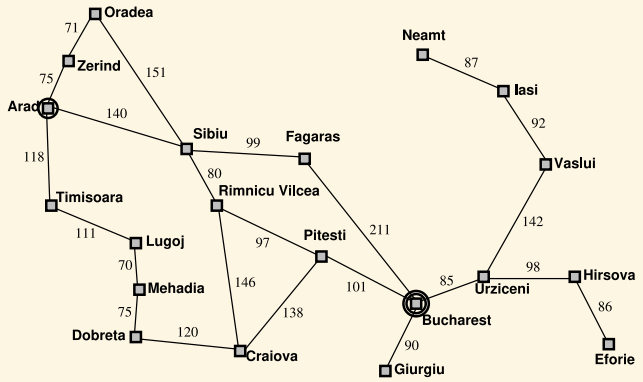
Informal definition of Relaxation

- 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 *estimate* h^*.
- You define a transformation, r, that *simplifies* instances from $\mathcal{P}$ into instances $\mathcal{P}'$.
- Given $\Pi \in \mathcal{P}$, you estimate heuristic $h^*(\Pi)$ by computing heuristic $h'^*(r(\Pi))$.

Relaxation

Relaxation means simplifying a problem by taking the solution to a simpler problem as the heuristic estimate for the solution to the actual problem.

Relaxation in Route-Finding



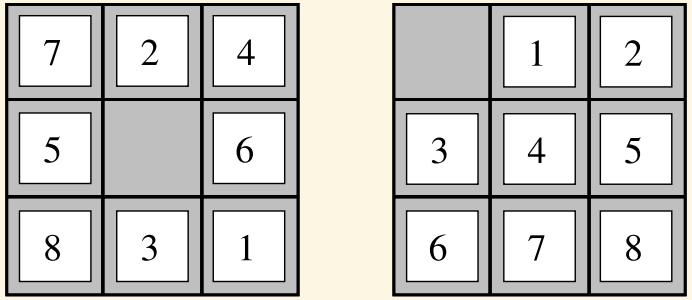
Distance from **Bucharest** to

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

How do we derive straight-line distances in route-finding by relaxation?:

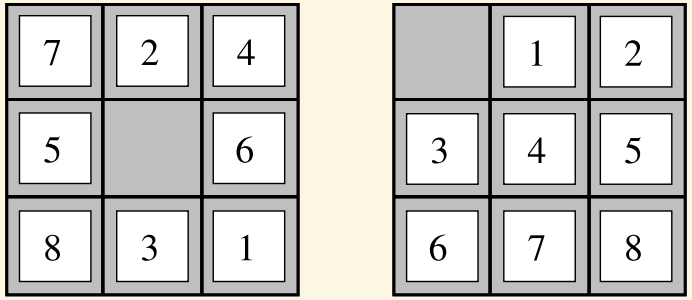
- Problem $\mathcal{P}$?: Route finding.
- Simpler problem $\mathcal{P}'$?:
- Perfect heuristic h'^* for $\mathcal{P}'$?:
- Relaxation Transformation r?:

Relaxation in the 8-Puzzle



- Action: move(X,Y)
- Pre: blank(X), adjacent(X,Y)
- Add: blank(X)
- Del: blank(Y)

Relaxation in the 8-Puzzle?



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

How do we derive a heuristic which specifies Manhattan (city-block) distance?:

How do we derive a heuristic which counts the number of misplaced tiles?:

“Goal-Counting” Relaxation for Logistics in Australia?

- 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 can 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 ?:

Heuristic value here?:

Notes:

- Optimal STRIPS planning with empty preconditions and deletes is still NP-hard! (Reduction from MINIMUM COVER, of goal set by add lists.)
- Relaxation approximates the perfect heuristic h'^* for $\mathcal{P}'$.

How to Relax Formally

Relaxation: Definition for Planning

Definition: Relaxation

Let $\mathcal{P}$ be a class of planning problems and $h^*(\Pi)$ denote the optimal plan cost of problem $\Pi \in \mathcal{P}$.

A *relaxation* is a triple $\mathcal{R} = (\mathcal{P}', r, h'^*)$ where

- $\mathcal{P}'$ is another class of planning problems,
- $r : \mathcal{P} \to \mathcal{P}'$ is a transformation between the problems, and
- $h'^*(\Pi')$ is the optimal plan cost of $\Pi' \in \mathcal{P}'$, such that

for all $\Pi \in \mathcal{P}$ $h'^*(r(\Pi)) \leq h^*(\Pi)$, and the heuristic induced by $\mathcal{R}$ is $h^{\mathcal{R}}(\Pi) = h'^*(r(\Pi))$.

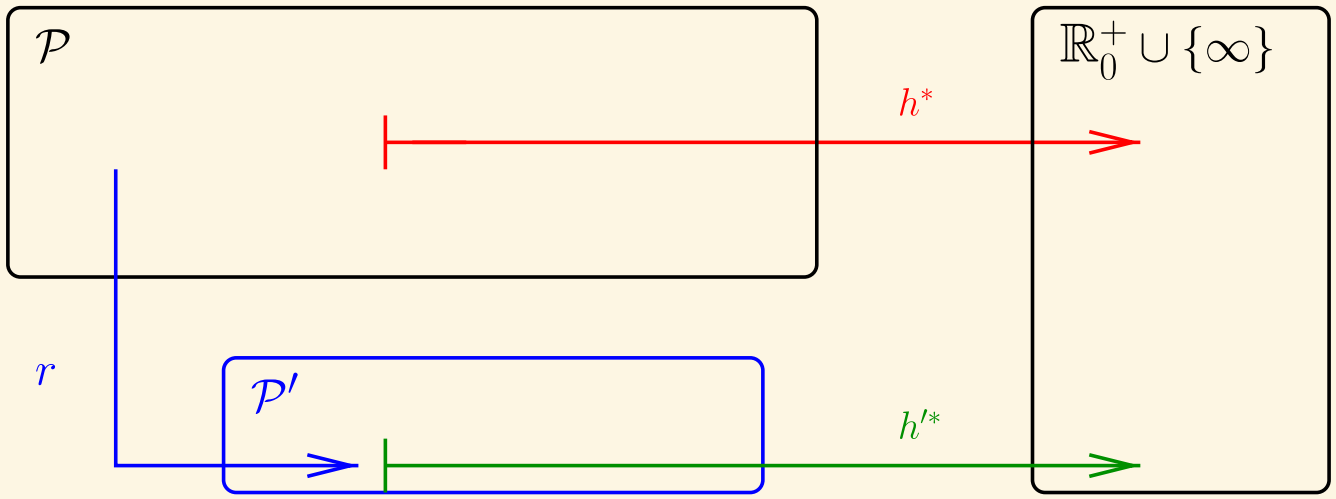
The relaxation is:

- Native if $\mathcal{P}' \subseteq \mathcal{P}$ and $h'^*(\Pi') = h^*(\Pi')$ for all $\Pi' \in \mathcal{P}'$;
- 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 $h'^*(\Pi')$.

The steps involved:

- 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^*prime 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 $h'^*(r(\Pi))$.
- Hence *goal counting* just approximates h'^* by number-of-false-goals.

Relaxations: Illustration (Example: Route-finding)



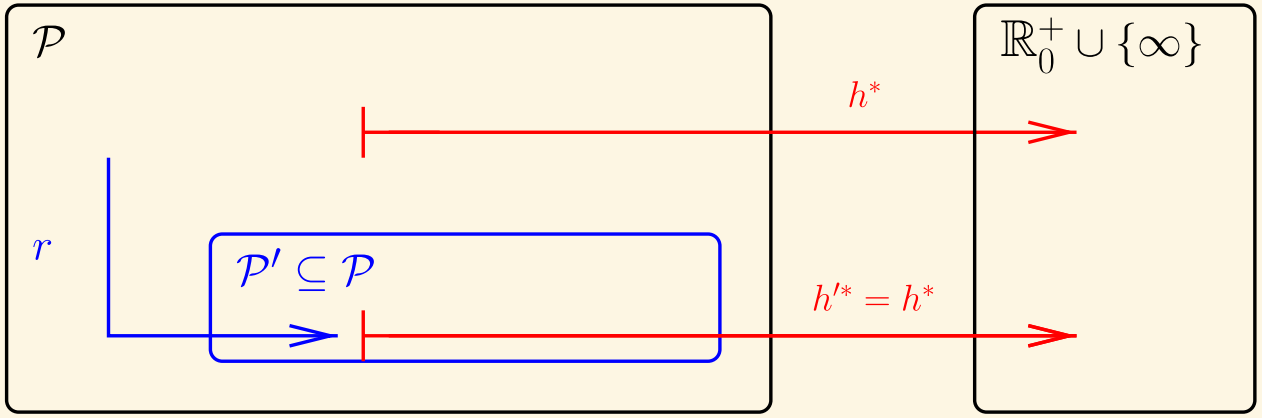
Problem $\mathcal{P}$: Route finding

Simpler problem $\mathcal{P}'$:

Perfect heuristic h'^* for $\mathcal{P}'$:

Transformation r?:

Native Relaxations: Illustration (Example: "Goal-counting")



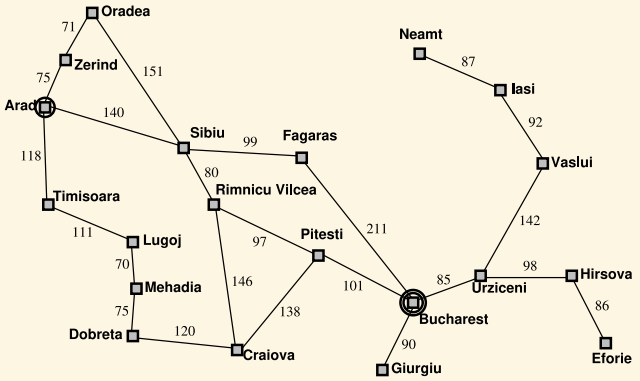
Problem $\mathcal{P}$: All STRIPS planning tasks.

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?:

Relaxation in Route-Finding: Properties?



Distance from **Bucharest** to

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

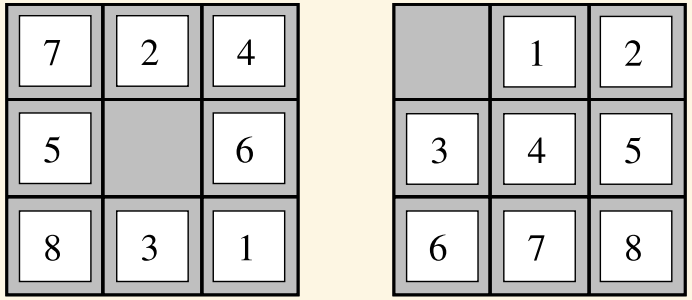
Assume relaxation $\mathcal{R} = (\mathcal{P}', r, h'^*)$: You are pretending to be a bird!

Native?:

Efficiently constructible?:

Efficiently computable?:

Relaxation in the 8-Puzzle: Properties?



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

Native?:

Efficiently constructible?:

Efficiently computable?:

What can we do with relaxations?

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

- Either (a) approximate r, or (b) design r in a way so that it will typically be feasible, or (c) just live with it and hope for the best.
- The vast majority of known relaxations (in planning) are efficiently constructible.

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. The latter use (a); (b) and (c) are not used anywhere right now.

"Goal-Counting" Relaxation: Properties?

- 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?:

Efficiently constructible?:

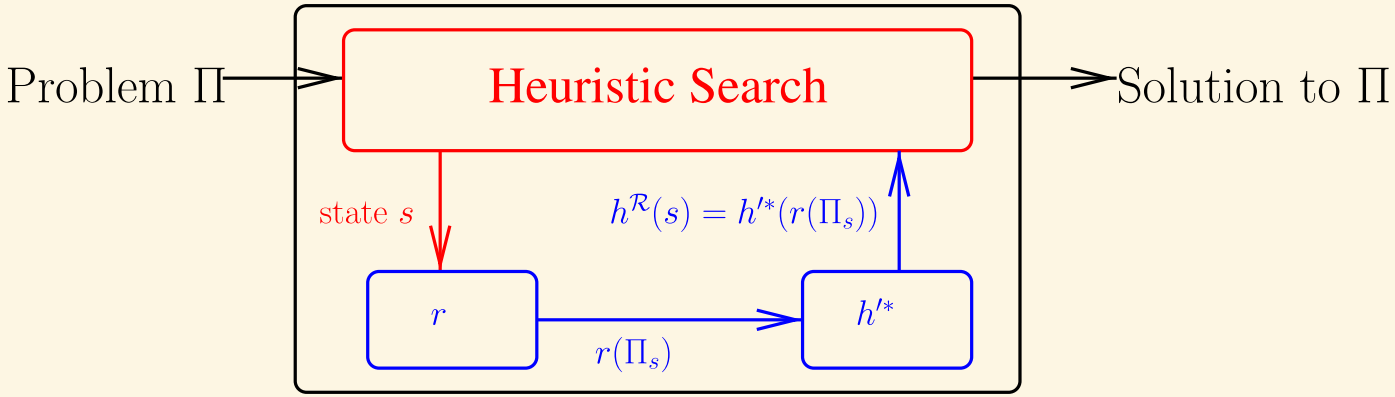
Efficiently computable?:

What approach, (a), (b) or (c), do we take if not construtible and/or computable?:

How to Relax During Search

How to Relax During Search: Diagram

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



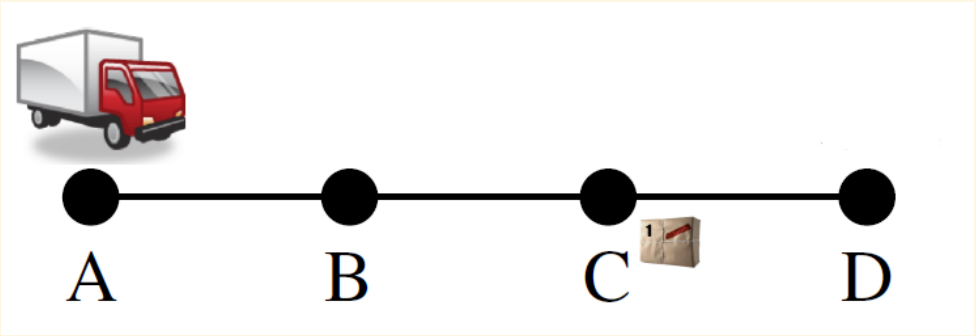
Π_s: is problem Π with initial state replaced by s

- i.e., $\Pi = (F, A, c, I, G)$ changed to (F, A, c, s, G) (initial sate s)

This is the task of finding a plan for search state s

- We will be using this notation in the course

How to Relax During Search: Logistics Problem



- Initial state I: AC
  - AC is the state encoding used here, that denotes that truck is at A and package is at C
- Goal G: AD
  - The goal is for the truck to take package to D from C, then return to A
- Actions A: $drXY$ (drive from X to Y), loX (load package X), ulX (unload X)
  - Each action also has corresponding pre, add & del

How to Relax During Search: Goal-Counting?

How to Relax During Search: Ignoring Deletes?

Conclusion

Robot Question

Question

Say we have a robot with one gripper, two rooms A and B, and n balls we must transport.

- The actions available are $moveXY$, $pickB$ and $dropB$; say $h =$ *"the number of balls not yet in room B"*.
- Can h be derived as $h^{\mathcal{R}}$ for a relaxation $\mathcal{R}$?

(A): No

(B): Yes, just drop the deletes

(C): Sure, *every* admissible h can be derived via a relaxation.

(D): I'd rather relax at the beach

Summary

- *Relaxation* is a method to compute heuristic functions.
- Given a problem $\mathcal{P}$ we want to solve, we define a relaxed problem $\mathcal{P}'$.
  - The heuristic is obtained by mapping into $\mathcal{P}'$ and taking the solution to this simpler problem as the estimate.
- Relaxations can be native, efficiently constructible, and/or efficiently computable.
  - None of these is a strict requirement for usefulness.
- During search, the relaxation is used *only inside* heuristic computation for each state.
  - It does not affect the rest of the search (This can be confusing, especially for native relaxations such as ignoring deletes.)

Goal Counting: Limitations

The goal-counting approximation, which is essentially $h =$ *"count the number of goals currently not true"* is a very uninformative heuristic function:

- The range of heuristic values is small $(0 \ldots |G|)$.
- We can transform any planning task into an equivalent one where $h(s) = 1$ for all non-goal states s.

How?:

Ignores almost all structure: heuristic value does not depend on the actions at all!

Is h safe/goal-aware/admissible/consistent?:

We will see how to compute better heuristic functions in the next Module.