Relaxation heuristics

h^{max}, h^{add} and Assignment #1-3

Michaela Urbanovská

PUI Tutorial
Week 5

Lecture check

- Any questions regarding the lecture?

Obtaining heuristics

- Many different possible ways to obtain a heuristic in general
- One general idea: solve a **simplified** version of the problem

Obtaining heuristics

- Many different possible ways to obtain a heuristic in general
- One general idea: solve a **simplified** version of the problem
  - relaxation

Obtaining heuristics

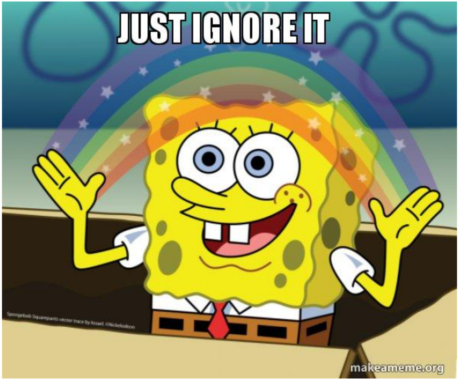
- Many different possible ways to obtain a heuristic in general
- One general idea: solve a **simplified** version of the problem
  - relaxation
  - abstraction

Obtaining heuristics

- Many different possible ways to obtain a heuristic in general
- One general idea: solve a **simplified** version of the problem
  - relaxation
  - abstraction
- This week: **relaxation**

Relaxation heuristic

- Relaxation is
  - general design technique
  - usually ignoring something
  - simplifying the problem



Relaxation heuristic

- What do we relax? Delete effects

Relaxation heuristic

- What do we relax? Delete effects
- But PDDL doesn't have any...we need STRIPS!
- **Delete relaxation**

Relaxed STRIPS planning task

Relaxation of a STRIPS planning task $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$ is the planning task $\Pi^+ = \langle F, O^+, s_{init}, s_{goal}, c \rangle$ which contains set of relaxed operators.

Relaxation heuristic

- What do we relax? Delete effects
- But PDDL doesn't have any...we need STRIPS!
- **Delete relaxation**

Relaxed STRIPS planning task

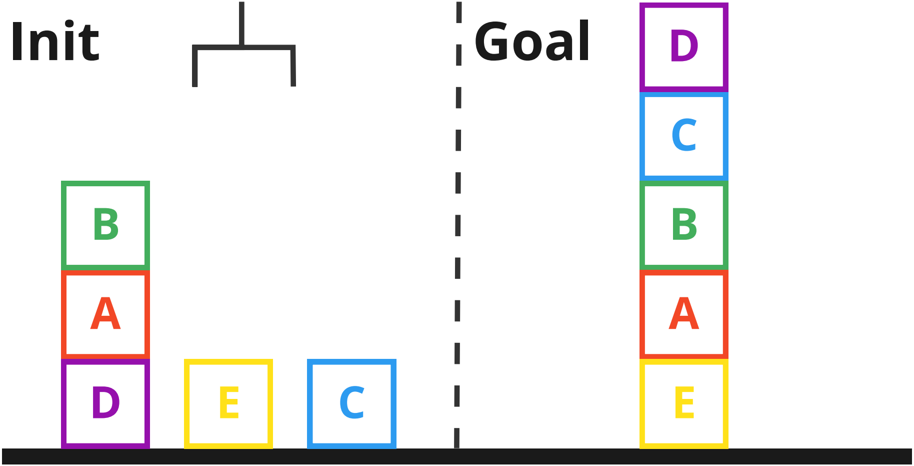
Relaxation of a STRIPS planning task $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$ is the planning task $\Pi^+ = \langle F, O^+, s_{init}, s_{goal}, c \rangle$ which contains set of relaxed operators.

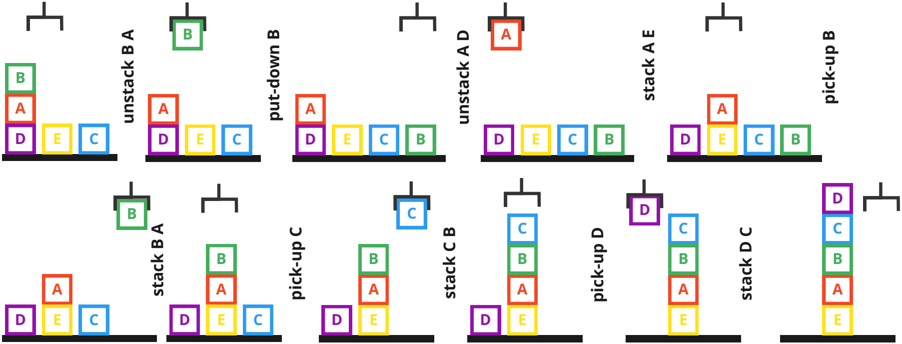
Relaxation of operators

Relaxation of operator $o = \langle pre(o), add(o), del(o) \rangle$ is operator $o^+ = \langle pre(o), add(o), \emptyset \rangle$.

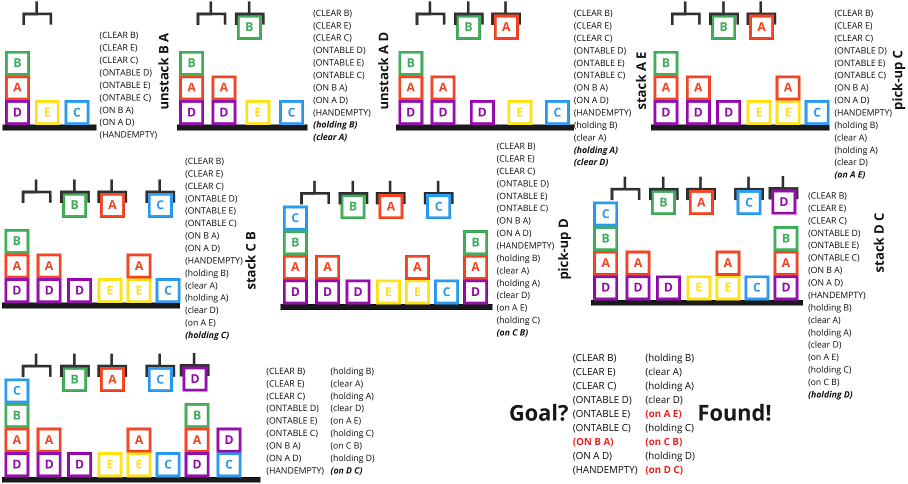
We can modify the PDDL accordingly.

Blocksworld planning problem




unstack B A
put-down B
unstack A D
stack A E
pick-up B
stack B A
pick-up C
stack C B
pick-up D
stack D C
A
B
C
D
E

Example - Blocks



Relaxation heuristic

h^+ heuristic

The h^+ heuristic computes length of the optimal relaxed plan π^+ which is an optimal solution to the relaxed problem Π^+.

- h^* - optimal for STRIPS definition Π
- h^+ - optimal for relaxed STRIPS definition Π^+

- Computation of h^+ is still complicated.
- We can compute an **estimate** of h^+.
  - h^{max}
  - h^{add}

h^{max} heuristic

- STRIPS planning task $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$
- $h^{max}(s)$ gives estimate of the distance from s to a state that satisfies s_{goal}
- $h^{max}(s) = max_{f \in s_{goal}} \Delta_1(s, f)$, where
  - $\Delta_1(s, o) = max_{f \in pre(o)} \Delta_1(s, f), \forall o \in O$
  - $\Delta_1(s, f) =$

$$\begin{cases} 0 & \text{if } f \in s, \\ \text{inf} & \text{if } \forall o \in O : f \notin add(o), \\ min\{c(o) + \Delta_1(s, o) | o \in O, f \in add(o)\} & otherwise. \end{cases}$$

h^{add} heuristic

h^{add} heuristic

- STRIPS planning task $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$
- $h^{add}(s)$ gives estimate of the distance from s to a state that satisfies s_{goal}
- $h^{add}(s) = \sum_{f \in s_{goal}} \Delta_0(s, f)$, where
  - $\Delta_0(s, o) = \sum_{f \in pre(o)} \Delta_0(s, f), \forall o \in O$
  - $\Delta_0(s, f) =$
    $$\begin{cases} 0 & \text{if } f \in s, \\ \text{inf} & \text{if } \forall o \in O : f \notin add(o), \\ min\{c(o) + \Delta_0(s, o) | o \in O, f \in add(o)\} & otherwise. \end{cases}$$

Exercise h^{add}, h^{max}

Compute $h^{max}(s_{init})$ for the following problem $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$:

$F = \{a, b, c, d, e, f, g\}$

$O =$

	pre	add	del	c
o_1	{a}	{c,d}	{a}	1
o_2	{a,b}	{e}	∅	1
o_3	{b,e}	{d,f}	{a,e}	1
o_4	{b}	{a}	∅	1
o_5	{d,e}	{g}	{e}	1

$s_{init} = \{a, b\}$ $s_{goal} = \{f, g\}$

h^{max} algorithm

Algorithm 1: Algorithm for computing $\mathrm{h}^{\max}(s)$.

Input: $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$, state s
Output: $\mathrm{h}^{\max}(s)$

for each $f \in s$ **do** $\Delta_1(s, f) \leftarrow 0$;
for each $f \in \mathcal{F} \setminus s$ **do** $\Delta_1(s, f) \leftarrow \infty$;
for each $o \in \mathcal{O}, pre(o) = \emptyset$ **do**
    for each $f \in add(o)$ **do** $\Delta_1(s, f) \leftarrow \min\{\Delta_1(s, f), \mathrm{c}(o)\}$;
end
for each $o \in \mathcal{O}$ **do** $U(o) \leftarrow |\mathrm{pre}(o)|$;
$C \leftarrow \emptyset$;
while $s_{goal} \not\subseteq C$ **do**
    $k \leftarrow \arg\min_{f \in \mathcal{F} \setminus C} \Delta_1(s, f)$;
    $C \leftarrow C \cup \{k\}$;
    for each $o \in \mathcal{O}, k \in pre(o)$ **do**
        $U(o) \leftarrow U(o) - 1$;
        if $U(o) = 0$ **then**
            for each $f \in add(o)$ **do**
                $\Delta_1(s, f) \leftarrow \min\{\Delta_1(s, f), \mathrm{c}(o) + \Delta_1(s, k)\}$;
            end
        end
    end
end
$\mathrm{h}^{\max}(s) = \max_{f \in s_{goal}} \Delta_1(s, f)$;

Exercise h^{add}, h^{max}

Compute $h^{add}(s_{init})$ for the following problem $\Pi = \langle F, O, s_{init}, s_{goal}, c \rangle$:

$F = \{a, b, c, d, e, f, g\}$

$O =$

	pre	add	del	c
o_1	{a}	{c,d}	{a}	1
o_2	{a,b}	{e}	$\emptyset$	1
o_3	{b,e}	{d,f}	{a,e}	1
o_4	{b}	{a}	$\emptyset$	1
o_5	{d,e}	{g}	{e}	1

$s_{init} = \{a, b\}$ $s_{goal} = \{f, g\}$

What's the difference in the algorithm?

Heuristic properties

Heuristic dominance

Admissible heuristic h_1 dominates an admissible heuristic h_2 if for every state s $h_1(s) \geq h_2(s)$

- h^+ is **admissible**, **consistent**
- h^{max} is **admissible**, **consistent**
- h^{add} is **not admissible**, **nor consistent** but can be very informative
- $h^{max} \leq h^+ \leq h^*$

Assignment #1-3 - Planner

- Third and final part of Assignment #1
- **Task:** implement a planner that uses A* search and a relaxation heuristic to solve problem formulated in PDDL
- **Points:** maximum 15
- **Deadlines**
  - 10.4.2023 23:59 (Monday)
  - 12.4.2023 23:59 (Wednesday)

Assignment #1-3 - Planner

- **Submission:** archive with *planner.py* and *grounder.py* (from Assignment #1-2)
- **Output:** *plan.txt* file with cost of the plan, heuristic for the inital state, plan (operator sequence)
- **AE**
  - 10 private problem domains (one problem per domain)
  - 400 seconds for all problems (roughly 30 seconds per problem)
  - Validator, cost check, heuristic check, operator sequence check
- **Debugging:** 3 public problems are available including the PDDLs and plans
  - Public domains are also in the private data but with different problems

Assignment #1-3 - Planner

- The *parser.py* has been slightly altered so download it again
- Your own *grounder.py* should be used for this assignment
  - In case you don't have a working one, you will be provided with a reference grounder

Recap

- Know what is delete relaxation and which heuristics are based on it
- Know h^+, h^{max} and h^{add} heuristics
- Get into implementation of Assignment #1-3
- Be capable of implementing h^{max} for your assignment

The End

Feedback form link