SAARLAND UNIVERSITY ARTIFICIAL INTELLIGENCE
Prof. Jana Koehler, Dr. Álvaro Torralba

Exercise Sheet 8
Solution

Exercise 30: Delete relaxation – Search with h^+. (4 Points)

Assume you have a robot that is able to clean the floor. It has a bin, that is filled when cleaning a place and that can be emptied at two specific locations. The map the robot lives in is given in Figure 1. The locations are connected with bidirectional edges, except that the robot cannot go back to the *Start* location (indicated by the arrows). Locations C and D are connected by a bridge which can only be passed if the bin is empty. The bin can only be emptied at B and G. Initially, the robot is located at *Start*. The problem is formalized in STRIPS as shown on the next page.

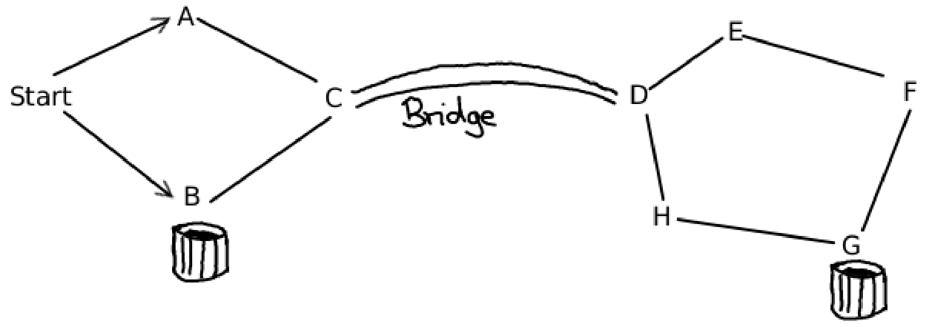

Figure 1: World map

$$\begin{aligned} P =&\{at(x) \mid x \in \{Start, A, \dots, H\}\} \\ &\cup \{dirty(x) \mid x \in \{A, \dots, H\}\} \\ &\cup \{clean(x) \mid x \in \{A, \dots, H\}\} \\ &\cup \{empty\} \end{aligned}$$

$$\begin{aligned} A = &\{moveDirty(x,y) \mid x,y \in \{Start, A, \dots, H\} \text{ and } x,y \text{ are connected in the graph}\} \\ &\cup \{moveClean(x,y) \mid x,y \in \{Start, A, \dots, H\} \text{ and } x,y \text{ are connected in the graph}\} \\ &\cup \{crossBridgeDirty(x,y) \mid x,y \in \{C,D\}\} \\ &\cup \{crossBridgeClean(x,y) \mid x,y \in \{C,D\}\} \\ &\cup \{emptyBin(x) \mid x \in \{B,G\}\}, \text{ where} \end{aligned}$$

- $moveDirty(x,y)$ for $x \in \{Start, A, ...H\}$, $y \in \{A, ...H\}$:
  - $pre : \{at(x), dirty(y)\}$
  - $add : \{at(y), clean(y)\}$
  - $del : \{at(x), dirty(y), empty\}$
- $moveClean(x,y)$ for $x \in \{Start, A, ...H\}$, $y \in \{A, ...H\}$:
  - $pre : \{at(x), clean(y)\}$
  - $add : \{at(y)\}$
  - $del : \{at(x)\}$
- $crossBridgeDirty(x,y)$ for x,y $\in \{C,D\}$:
  - $pre : \{at(x), dirty(y), empty\}$
  - $add : \{at(y), clean(y)\}$
  - $del : \{at(x), dirty(y), empty\}$
- $crossBridgeClean(x,y)$ for x,y $\in \{C,D\}$:
  - $pre : \{at(x), clean(y), empty\}$
  - $add : \{at(y)\}$
  - $del : \{at(x)\}$
- $emptyBin(x)$ for x $\in \{B,G\}$:
  - $pre : \{at(x)\}$
  - $add : \{empty\}$
  - $del : \{\}$

$I = \{at(Start), clean(B), empty\} \cup \{dirty(x) \mid x \in \{A, C, \dots, H\}\}$

a) Compute h^{FF} for the initial state I given the following goal state:

$G = \{clean(x) \mid x \in \{A, ...H\}\} \cup \{empty\}$

Write down the action and fact sets for each iteration of the RPG algorithm to iteratively compute the value of h^{FF}. In the plan extraction, for each step t, write down which actions are selected and give the resulting G_{t-1} you obtain. What is the final value of $h^{FF}(I)$?

b) Run A* search on the problem using h^+ as heuristic. Use the following goal state: $G = \{clean(x) |\ x \in \{A, C, D, G, H\}\} \cup \{empty\}$
If several nodes have the same $g+h$ value, rely on the heuristic and choose the one with the smaller h value. If there are several nodes with same g and same h value, break ties by using the alphabetical order on location names. In each search node, mention the literals that are true, the g and h values as well as their sum and the expansion order. You can omit the $dirty(x)$ facts in the state representation. In the search graph, mark duplicate nodes and annotate the edges with the corresponding action names.

c) Comment on how well h^+ guides the search in terms of expanded nodes and how often the tie breaking rule was used instead of having a unique smallest value in the open list. How would search behave if we remove the constraint for the bin to be empty from the *crossBridge* actions? There is no need to draw the search space again, just comment as before on the number of expanded nodes and the usage of the tie breaking rule.

(Solution)

a)
- $F_0 = I = \{at(Start), clean(B), empty\} \cup \{dirty(x) \mid x \in \{A, C, \dots, H\}\}$
- $A_0 = \{moveDirty(Start, A), moveClean(Start, B)\}$
- $F_1 = F_0 \cup \{at(A), clean(A), at(B)\}$
- $A_1 = A_0 \cup \{emptyBin(B), moveDirty(A, C), moveDirty(B, C), moveClean(Start, A)\}$
- $F_2 = F_1 \cup \{at(C), clean(C)\}$
- $A_2 = A_2 \cup \{crossBridgeDirty(C, D), moveDirty(C, A), moveClean(C, A), moveClean(C, B), moveClean(A, C), moveClean(B, C)\}$
- $F_3 = F_2 \cup \{at(D), clean(D)\}$
- $A_3 = A_2 \cup \{moveDirty(D, E), moveDirty(D, H), crossBridgeDirty(D, C), crossBridgeClean(C, D), crossBridgeClean(D, C)\}$
- $F_4 = F_3 \cup \{at(E), clean(E), at(H), clean(H)\}$
- $A_4 = A_3 \cup \{moveDirty(E, F), moveDirty(H, G), moveDirty(E, D), moveDirty(H, D), moveClean(E, D), moveClean(H, D), moveClean(D, E), moveClean(D, H)\}$
- $F_5 = F_4 \cup \{at(F), clean(F), at(G), clean(G)\} \supseteq G$

During the computation of h^{FF}, we select the following actions:

We start with $G_5 = \{clean(F), clean(G)\}$.

$t = 5$: select $moveDirty(E, F)$ and $moveDirty(H, G)$,

$G_4 = \{at(E), clean(E), at(H), clean(H)\}$

$t = 4$: select $moveDirty(D, E)$ and $moveDirty(D, H)$,

$G_3 = \{at(D), clean(D)\}$

$t = 3$: select $crossBridgeDirty(C, D)$,

$G_2 = \{at(C), clean(C)\}$

$t = 2$: select $moveDirty(A, C)$,

$G_1 = \{at(A), clean(A)\}$

$t = 1$: select $moveDirty(Start, A)$,

$G_0 = \{at(Start), dirty(A), dirty(C), dirty(D), dirty(E), dirty(F), dirty(G), dirty(H), empty\}$

h^{FF} returns 7 for the initial state.

b) See Figure 2.

c) The expansion of state $s_1 = \{at(B), clean(B), empty\} \cup \{dirty(x) \mid x \in \{A, C, \ldots, H\}\}$ and its successor is not helpful for finding the goal. There are two superfluous expansions. Moreover, the tie breaking rule prevents search from exploring more successor states of s_1, as the nodes on the solution path also have f values 9.

If we would remove the constraint from *crossBridge*, the h-value of node $\{at(A), clean(A), clean(B)\} \cup \{dirty(x) \mid x \in \{C, \ldots, H\}\}$ would be 5 and the one of $\{at(B), clean(B)\} \cup \{dirty(x) \mid x \in \{A, C, \ldots, H\}\}$ would be 6, so search would take the left branch and find the optimal solution without superfluous expansions. This is due to the nature of h^+, it ignores deletes and therefore underestimates the cost of emptying the bin before crossing the bridge.

(/Solution)

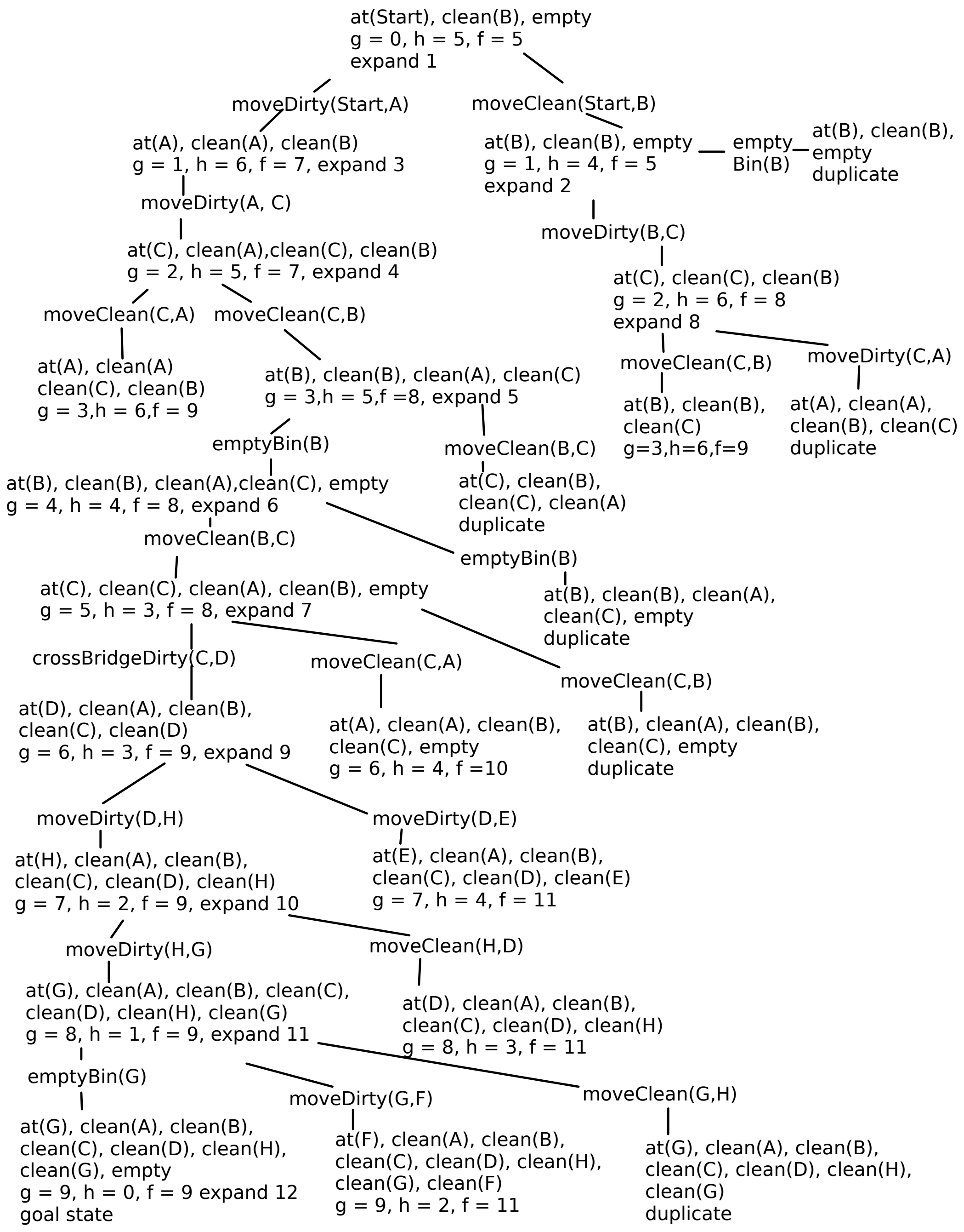
at(Start), clean(B), empty g = 0, h = 5, f = 5 expand 1
moveDirty(Start,A)
moveClean(Start,B)
at(A), clean(A), clean(B) g = 1, h = 6, f = 7, expand 3
at(B), clean(B), empty g = 1, h = 4, f = 5 expand 2
empty Bin(B)
at(B), clean(B), empty duplicate
moveDirty(A, C)
moveDirty(B,C)
at(C), clean(A),clean(C), clean(B) g = 2, h = 5, f = 7, expand 4
at(C), clean(C), clean(B) g = 2, h = 6, f = 8 expand 8
moveClean(C,A)
moveClean(C,B)
at(A), clean(A) clean(C), clean(B) g = 3,h = 6,f = 9
at(B), clean(B), clean(A), clean(C) g = 3,h = 5,f =8, expand 5
moveClean(C,B)
moveDirty(C,A)
at(B), clean(B), clean(C) g=3,h=6,f=9
at(A), clean(A), clean(B), clean(C) duplicate
emptyBin(B)
moveClean(B,C)
at(B), clean(B), clean(A),clean(C), empty g = 4, h = 4, f = 8, expand 6
at(C), clean(B), clean(C), clean(A) duplicate
moveClean(B,C)
emptyBin(B)
at(B), clean(B), clean(A), clean(C), empty duplicate
at(C), clean(C), clean(A), clean(B), empty g = 5, h = 3, f = 8, expand 7
crossBridgeDirty(C,D)
moveClean(C,A)
moveClean(C,B)
at(D), clean(A), clean(B), clean(C), clean(D) g = 6, h = 3, f = 9, expand 9
at(A), clean(A), clean(B), clean(C), empty g = 6, h = 4, f =10
at(B), clean(A), clean(B), clean(C), empty duplicate
moveDirty(D,H)
moveDirty(D,E)
at(H), clean(A), clean(B), clean(C), clean(D), clean(H) g = 7, h = 2, f = 9, expand 10
at(E), clean(A), clean(B), clean(C), clean(D), clean(E) g = 7, h = 4, f = 11
moveDirty(H,G)
moveClean(H,D)
at(G), clean(A), clean(B), clean(C), clean(D), clean(H), clean(G) g = 8, h = 1, f = 9, expand 11
at(D), clean(A), clean(B), clean(C), clean(D), clean(H) g = 8, h = 3, f = 11
emptyBin(G)
moveDirty(G,F)
moveClean(G,H)
at(G), clean(A), clean(B), clean(C), clean(D), clean(H), clean(G), empty g = 9, h = 0, f = 9 expand 12 goal state
at(F), clean(A), clean(B), clean(C), clean(D), clean(H), clean(G), clean(F) g = 9, h = 2, f = 11
at(G), clean(A), clean(B), clean(C), clean(D), clean(H), clean(G) duplicate

Figure 2: Solution of Exercise 30 b).

Exercise 31: Delete relaxation – h^{max}, h^{FF}. (2.5 Points)

XinYue needs more cakes to send them to friends. Since she cannot bake them all, she wants to buy some delicious cakes. She also needs to buy wrapping paper. As she wants to support small shops, she does not want to go to the supermarket. Please help her to estimate the time she needs to purchase her cakes and wrapping paper. To do so, consider the map in Figure 3. Currently, XinYue is at home at location A, the supermarket is at location C, and the cake shop is at location E. The wrapping paper can be bought at location D. In the end, XinYue wants to have cakes and wrapping paper, and to be back at home. Note that she first needs to withdraw money at the cashpoint at location B. The problem is formalized as the following STRIPS planning problem $\Pi = (P, A, I, G)$:

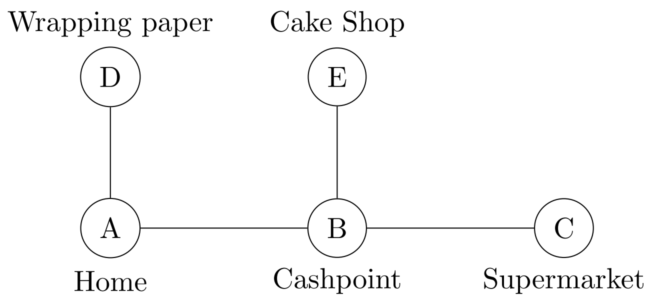

Figure 3: Illustration of the Map of Exercise 31.

$$
\begin{aligned}
P =& \{at(x) |\ x \in \{A, B, C, D, E\}\} \cup \{havePaper, haveCake, have2Money, have1Money\} \\
I =& \{at(A)\} \\
G =& \{at(A), havePaper, haveCake\} \\
A =& \{walk(x, y) \mid x, y \in \{A, B, C, D, E\} \wedge x, y \ are\ connected\} \\
& \cup \{buyPaper_2(x) \mid x \in \{C, D\}\} \cup \{buyPaper_1(x) \mid x \in \{C, D\}\} \\
& \cup \{buyCake_2(x) \mid x \in \{C, E\}\} \cup \{buyCake_1(x) \mid x \in \{C, E\}\} \\
& \cup \{withdrawMoney()\}, \text{ where}
\end{aligned}
$$

- $walk(x, y)$ for $x, y \in \{A, B, C, D, E\}$, x, y are connected in the map of Figure 3
  - $pre : \{at(x)\}$
  - $add : \{at(y)\}$

  - $del : \{at(x)\}$

- $buyPaper_2(x)$ for $x \in \{C, D\}$
  - $pre : \{at(x), have2Money\}$
  - $add : \{havePaper, have1Money\}$
  - $del : \{have2Money\}$
- $buyPaper_1(x)$ for $x \in \{C, D\}$
  - $pre : \{at(x), have1Money\}$
  - $add : \{havePaper\}$
  - $del : \{have1Money\}$
- $buyCake_2(x)$ for $x \in \{C, E\}$
  - $pre : \{at(x), have2Money\}$
  - $add : \{haveCake, have1Money\}$
  - $del : \{have2Money\}$
- $buyCake_1(x)$ for $x \in \{C, E\}$
  - $pre : \{at(x), have1Money\}$
  - $add : \{haveCake\}$
  - $del : \{have1Money\}$
- $withdrawMoney()$
  - $pre : \{at(B)\}$
  - $add : \{have2Money\}$
  - $del : \{have1Money\}$

a) Compute h^{FF} for the initial state as in Exercise 30 (a). Try to avoid the supermarket if possible.

b) Compute $h^{\max}$ for the initial state, using the relaxed planning graph that you computed in part (a). Fill in the table below. What is the value of $h^{\max}(I)$?

c) Compare $h^*(I)$ and $h^+(I)$ with $h^{\max}(I)$ and $h^{\text{FF}}(I)$. To do so, give the value of $h^*(I)$ and $h^+(I)$, and use the values of $h^{\max}(I)$ and $h^{\text{FF}}(I)$ from parts (a) and (b). Which of these heuristics are admissible, in general and in this particular planning task?

(Solution)

g	$h^{\max}(I,g)$
$atC(A)$	0
$atC(B)$	1
$atC(C)$	2
$atC(D)$	1
$atC(E)$	2
$have2Money$	2
$have1Money$	3
$havePaper$	3
$haveCake$	3

Table 1: Table with the $h^{\max}$ values for every fact.

a) RPG:

- $F_0 = \{at(A)\}$
- $A_0 = \{walk(A,B), walk(A,D)\}$
  $F_1 = F_0 \cup \{at(B), at(D)\}$
- $A_1 = A_0 \cup \{walk(B,A), walk(D,A), walk(B,C), walk(B,E), withdrawMoney()\}$
  $F_2 = F_1 \cup \{at(C), at(E), have2Money\}$
- $A_2 = A_1 \cup \{walk(C,B), walk(E,B), buyPaper_2(C), buyCake_2(C),$
  $buyPaper_2(D), buyCake_2(E)\}$
  $F_3 = F_2 \cup \{haveMoney_1, havePaper, haveCake\}$
- $G \subseteq F_3 \to$ stop

Plan extraction:

- $G_3 = \{havePaper, haveCake\}$
  $havePaper$: $buyPaper_2(D)$
  $haveCake$: $buyCake_2(E)$
- $G_2 = \{at(E), have2Money\}$
  $at(E)$: $walk(B,E)$
  $have2Money$: $withDrawMoney$
- $G_1 = \{at(B), at(D)\}$
  $at(B)$: $walk(A,B)$
  $at(D)$: $walk(A,D)$
- $G_0 = I$

Total number of selected actions is 6 and thus $h^{\mathrm{FF}}(I) = 6$. Note that h^{FF} could also be 5 if we buy both, the paper and the cake, at the supermarket.

b) See Table 1 for the $h^{\max}$ values for every fact. Since the $h^{\max}$ value of *havePaper* and *haveCake* is 3 and XinYue is initially at A, $h^{\max}(I) = \max(3, 3, 0) = 3$.

c) An optimal plan is given by the action sequence $\langle walk(A, B), withdrawMoney(), walk(B, C), buyPaper_2, buyCake_1, walk(C, B), walk(B, A)\rangle$ and therefore, $h^*(I) = 7$. Furthermore, $h^+(I) = 5$, as Claus does not need to walk back home, $h^{\max}(I) = 3$.

$h^{\max}$ and h^+ are always admissible as given on the lecture slides. h^{FF} is not always admissible because it might choose "bad" actions during the plan extraction (for example, we selected *buyCake*(E) leading to a larger h^{FF} value than *buyCake*(C) together with *buyPaper*(C).

(/Solution)

Exercise 32. (2 Points)

Prove that, given an arbitrary planning task $\Pi = (P, A, I, G)$ and any state s, if $\langle a_1, \dots, a_n\rangle$ is a plan for (P, A, s, G), then $\langle a_1^+, \dots, a_n^+\rangle$ is a plan for $(P, A, s, G)^+$.

Hint: Denote the state-action sequence traversed by $\langle a_1, \dots, a_n\rangle$ as $seq = s_0, a_1, s_1, \dots, a_n, s_n$. Denote the state-action sequence traversed by $\langle a_1^+, \dots a_n^+\rangle$ as $seq^+ = s_0^+, a_1^+, s_1^+, \dots, a_n^+, s_n^+$. Then show that *(*) for* $1 \leq i \leq n$, a_i^+ *is applicable to* s_{i-1}^+*; and for* $0 \leq i \leq n$ *we have* $s_i \subseteq s_i^+$. Once (*) is proved, use it to prove that $\langle a_1^+, \dots, a_n^+\rangle$ is a plan for $(P, A, s, G)^+$.

Exercise 33. (1.5 Points)

As specified in the lecture, PlanLen$^+$ is the problem of deciding, given a STRIPS planning task Π and an integer B, whether or not there exists a delete-relaxed plan for Π of length at most B. Prove that PlanLen$^+$ is a member of **NP**.

Hint: There is a simple upper bound on the length of a relaxed plan, if one exists.