

## Solutions to Exercises - Lecture 3

### Intelligent Systems Programing

#### Exercise 1

a)

We represent each of  $n$  disks with a number from  $S = \{1, \dots, n\}$  assuming that smaller disks are identified with smaller numbers, i.e., the  $i$ -th disk is smaller than the  $j$ -th disk if  $i < j$ . A *state* is defined with a partition of  $S$  into three sets  $(S_1, S_2, S_3)$  (i.e.,  $\cup_{i=1}^3 S_i = S$ , and  $S_i \cap S_j = \emptyset$  for all  $i \neq j$ ) where a set  $S_k$  represents disks that are on the  $k$ -th peg. Once a set  $S_k$  is given, we know exactly how the disks are put on the  $k$ -th peg, since there is only one way to order them (smaller on top). In particular, the topmost disk in set  $S_k$  is  $\min\{a \mid a \in S_k\}$ , which we denote simply as  $\min S_k$ .

We have six *actions*  $M(1, 2), M(1, 3), M(2, 1), M(2, 3), M(3, 1), M(3, 2)$ , where  $M(i, j)$  denotes an action of moving a smallest disk from the  $i$ -th peg, and putting it on the  $j$ -th peg. For each state only some of the actions might be legal, i.e., the topmost disk on the  $i$ -th peg must be smaller than topmost disk on the  $j$ -th peg ( $\min S_i < \min S_j$ ). The *Initial state* is given by  $S_1 = \{1, \dots, n\}$ ,  $S_2 = \emptyset$ ,  $S_3 = \emptyset$ .

The *actions function* for each state  $(S_1, S_2, S_3)$  returns all legal action-state pairs  $(M(i, j), (S'_1, S'_2, S'_3))$ , where the action  $M(i, j)$  is legal. For each such action  $M(i, j)$ , in the *results function* gives the resulting state in which the two affected pegs are  $S'_i = S_i \setminus \{\min S_i\}$ ,  $S'_j = S_j \cup \{\min S_i\}$  while the remaining one stays the same.

*Goal test* is given by checking whether  $S_3 = \{1, 2, \dots, n\}$ . Path cost is a positive constant, for example  $c(s, a, s') = 1$ .

b)

The state space is shown in Figure 1.

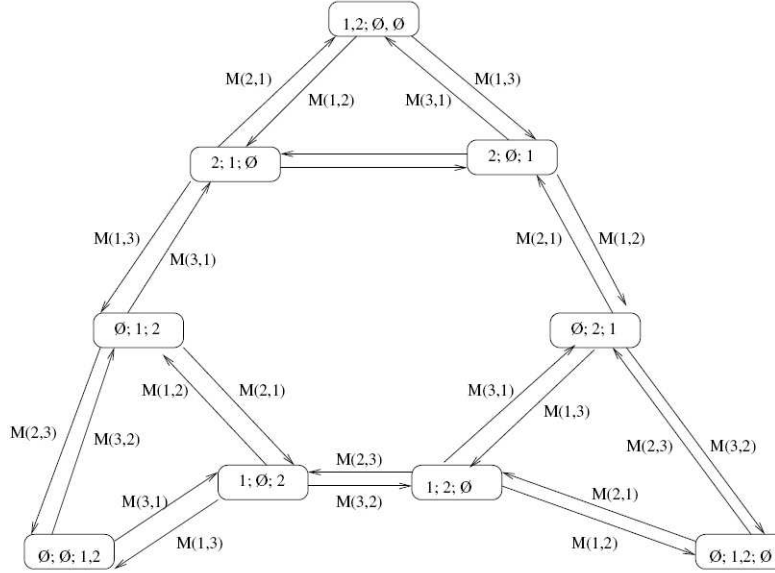


Figure 1: State space of Towers of Hanoi with  $n = 2$ .  $\emptyset$  denotes a peg without disks.

c)

Notice the difference between legal states (i.e., all states satisfying our definition of a state) and reachable states (i.e., states we can reach by executing some sequence of actions from the starting state). Reachable states are in general a subset of legal states. Since we are assuming all legal states are reachable, we need only to count the number of states ( $S_1, S_2, S_3$ ) that satisfy our definition from a). The number of all legal configurations is  $3^n$  since for every disk we can choose any of the three pegs to put it on.

## Exercise 2

a)

The nodes in the fringe of the A\*-algorithm when solving the specified problem are given in the table below. Notice that the fringe is a priority queue, and that the nodes in it are therefore ordered in accordance to their  $f$  value.

Iteration	Fringe
0	$\langle 244, 0, 244, L \rangle$
1	$\langle 311, 70, 241, M \rangle, \langle 440, 111, 329, T \rangle$
2	$\langle 387, 145, 242, D \rangle, \langle 440, 111, 329, T \rangle$
3	$\langle 425, 265, 160, C \rangle, \langle 440, 111, 329, T \rangle$
4	$\langle 440, 111, 329, T \rangle, \langle 503, 403, 100, P \rangle, \langle 604, 411, 193, RV \rangle$
5	$\langle 503, 403, 100, P \rangle, \langle 595, 229, 366, A \rangle, \langle 604, 411, 193, RV \rangle$
6	$\langle 595, 229, 366, A \rangle, \langle 604, 411, 193, RV \rangle, \langle 504, 504, 0, B \rangle$

**b)**

A\* returns  $\text{SOLUTION}(goal)$  where  $goal$  is a goal node. The mentioned algorithm  $\text{SOLUTION}$  traces the path from  $goal$  back to the initial node and returns the found path.

In this case, A\* therefore returns the path:  $L \rightarrow M \rightarrow D \rightarrow C \rightarrow P \rightarrow B$ .

### Exercise 3

**a)**

For  $w = 1$  we have greedy best first search, for  $w = 0.5$  we have  $A^*$  search and for  $w = 0$  we have uniform cost search.

**b)**

Notice that we can multiply  $f(n)$  with a positive number without changing the behavior of the algorithm. Now multiply  $f(n)$  with  $\frac{1}{(1-w)}$  which gives  $\frac{1}{(1-w)}f(n) = g(n) + \frac{w}{1-w}h(n)$ . The heuristic function  $\frac{w}{1-w}h(n)$  will be admissible for  $\frac{w}{1-w} \leq 1$  since  $h(n)$  is admissible. This is the case for  $w \leq 0.5$ .

### Exercise 4 (Challenge)

**a)**

Let  $k$  denote the number of edges in  $p^*(n)$ ,  $k = |p^*(n)|$ .

- If  $k = 0$ , a state  $n$  is a goal state  $s_G$ . Therefore  $h(n) = 0$  according to the definition of heuristic function  $h(n)$ . But also,  $h^*(n) = 0$  since obviously the cheapest path to goal has length 0. Therefore it holds  $h(n) \leq h^*(n)$ .

- Assume that the claim holds for paths with  $k$  edges. Let us show that the statement also holds for paths with  $k+1$  edges. Let  $p^*(n) = \{n, n_1, \dots, n_{k+1}\}$  be the optimal path with  $k+1$  edges where  $n_{k+1}$  is the goal state. A cost of each edge  $c(n_i, n_{i+1})$  is the standard cost of executing an action from a state  $n_i$  that leads to a state  $n_{i+1}$ .

Since the statement holds for paths with  $k$  edges, it also holds for state  $n_1$ . Namely, the optimal path from  $n_1$  to goal  $n_{k+1}$ ,  $p_1^*(n_1) = \{n_1, \dots, n_{k+1}\}$ , has  $k$  edges. Therefore:

$$h(n_1) \leq h^*(n_1) \quad (1)$$

Also, since  $p^*(n_1)$  is the optimal path from  $n_1$ , its length  $\sum_{i=1}^k c(n_i, n_{i+1})$  is equal to  $h^*(n_1)$ . Since the similar holds for  $h^*(n)$  it follows:

$$h^*(n) = c(n, n_1) + h^*(n_1) \quad (2)$$

Now we have everything we need to prove  $h(n) \leq h^*(n)$ . Since,  $h$  is consistent heuristic it holds  $h(n) \leq c(n, n_1) + h(n_1) \stackrel{(1)}{\leq} c(n, n_1) + h^*(n_1) \stackrel{(2)}{=} h^*(n)$ . This proves the induction step.

- Since the statement holds for  $k = 0$ , and from the fact that if the statement holds for  $k$  we can show that it also holds for  $k+1$ , according to the principle of mathematical induction, the statement holds for all  $k \in \mathbb{N}$ .

## b)

Assume that  $h$  is not admissible. Then for some state  $n_0$  it holds  $h(n_0) > h^*(n_0)$ . However, relation  $>$  makes sense only if both  $h(n_0)$  and  $h^*(n_0)$  are finite. But this means that a goal is reachable from  $n_0$  in finite number of steps, i.e., there is an optimal path  $p^*(n_0)$ . Then according to a) it must hold  $h(n_0) \leq h^*(n_0)$  which contradicts the initial assumption. Therefore  $h$  is admissible.