Assignment No. 1 - Theoretical Assignment
AI and Robotics (236609)

Goldstein, Sharon
sharongold@campus.technion.ac.il
Computer Science & Physics
Technion - Israel Institute of Technology

Granov, Yotam
yotam.g@campus.technion.ac.il
Mechanical Engineering & Physics
Technion - Israel Institute of Technology

November 28, 2022

1 State Space Search

1.1

In the graph shown in Figure 1, A is the start node and D is the goal state. In addition, $h(A) = 8, h(B) = 3, h(C) = 7$ and $h(D) = 0$.

(a) First, we will calculate h^* for all nodes in the graph:

$$h^*(D) = 0, h^*(C) = 8, h^*(B) = 6, h^*(A) = 9$$

A heuristic h is admissible if $\forall s \in S : 0 \leq h(s) \leq h^*(s)$. The given heuristic maintains:

$$\begin{aligned} 0 \leq 0 \leq 0 &\Rightarrow 0 \leq h(D) \leq h^*(D) \\ 0 \leq 7 \leq 8 &\Rightarrow 0 \leq h(C) \leq h^*(C) \\ 0 \leq 3 \leq 6 &\Rightarrow 0 \leq h(B) \leq h^*(B) \\ 0 \leq 8 \leq 9 &\Rightarrow 0 \leq h(A) \leq h^*(A) \end{aligned}$$

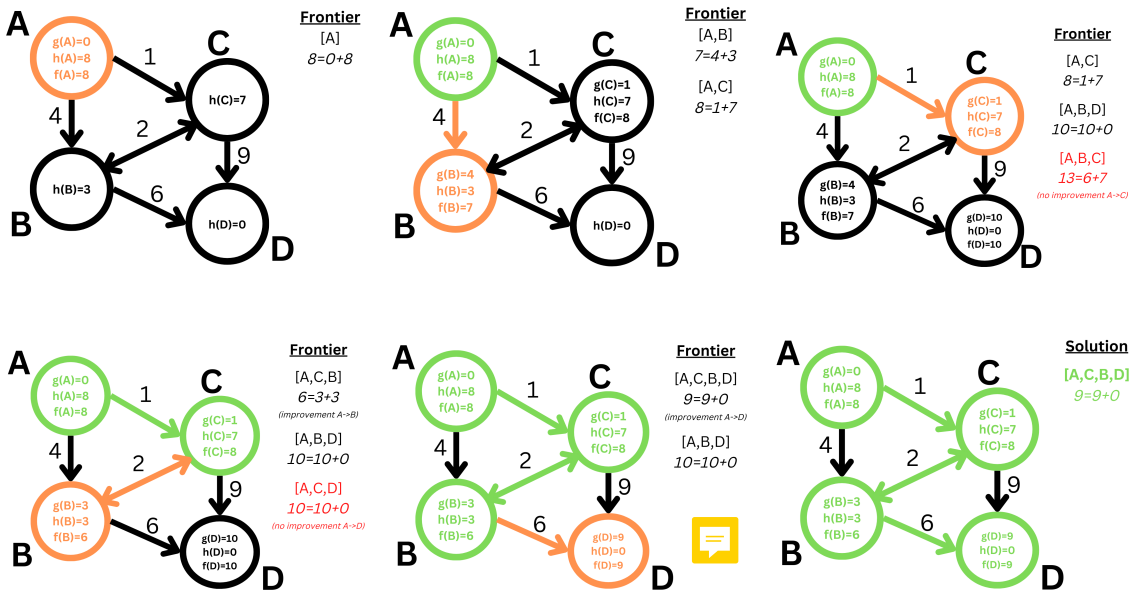
Therefore, the given heuristic is indeed **admissible**.

For a heuristic to be consistent, on the other hand, it needs to fulfill the following condition: $\forall s \in S, s' \in succ(s) : h(s) - h(s') \leq cost(s, s')$. In our case, we see that:

$$h(A) - h(B) = 8 - 3 = 5 \not\leq 4 = cost(A, B)$$

and so it's clear that h is **not consistent**.

(b)



Note that in these graphics, orange coloring indicates that a node is next on the queue to be explored (since it has the lowest $f(m)$ value of all nodes in the frontier), and green indicates that the shortest path to that node has been found and confirmed.

✓ (c) Our A* execution found that the path ***[A,C,B,D]*** is the shortest between nodes A and D, and has a total cost of 9.

(d) Path checking will change the search. Permutations of paths will not be checked and in our case, when the heuristic is not consistent, a path might change during the run (as explained in 1.2). For example, in the given graph and path checking A*, node B will be expanded before node C ($f(B) = 7 < 8 = f(C)$) so when we expand C, $[A, B, C]$ is in the OPEN list, and the optimal path $[A, C, B]$ will not be in the OPEN list and will not be found.
Cycle checking is relevant when a cycle can reduce the total cost, which means there is an edge with a negative cost. Since our edge costs are non - negative, cycle checking will not change the search in the given graph.

1.2

✓ While running A* with heuristic h, a node that was in the closed-list was put in the open-list. In this case, h is **not consistent** (option (iv)). A node returned from the closed list to the open list, so it's scanned path is not optimal and needed to be updated. If a heuristic is consistent, the path to every node is optimal. Therefore, if a path to a node is not optimal (like in this case), the heuristic is not consistent. We saw this happen in our example (where h was not consistent), when the node B was added to the closed-list with $f(B) = 7$ but later re-added to the open-list (and later again to the closed-list) with $f(B) = 6$.

1.3

✓ While running A* with some heuristic h, a goal node was expanded and put into the closed-list with value f. It's **not possible** that the value of the node will be updated at future steps (option (iv)) because the A* algorithm will terminate as soon as the goal node is put into the closed-list, regardless of the admissibility or consistency of h.

2 Modeling

✓ ### 2.1

For each of the models below, we'll suggest an appropriate definition of each of its elements. The elements of interest are the arm and the cubes.

(a) **Classical Planning:**
Each cube will be represented by its 3d coordinates and its color. The arm will be represented as the angles of the joints, the holder 3d coordinates, the holder position/pressure and a flag if the holder is occupied. A cube is held when the holder is occupied and its coordinates are equal to a cubes.

(b) **MDP** (S, A, P, R, γ)**:**
The states are the same as in (a). The actions will represent the arm's movement (joint change or holder state and pressure). The probability will depend on the actuation and the abilities of its movement, the low-level controllers making the change. The reward will be a negative one – the difference between the given state and the wanted one.

Assuming a specific tower is wanted in a specific place, this difference can be the sum distance of the final and present place of each cube. If only a general color condition is given, the difference can be the distance between neighbor cubes in the tower.

(c) **POMDP** $(S, A, P, R, \gamma, \Omega, O, b_0)$**:**
We will assume our previous states are the expectation values with a relevant STD (generally using Gaussian spread, unless more information is given such as physical problems, the low level controllers code and more). The observation function depends on the relevant sensor.

2.2

Classical planning and MDP do not take into consideration the observation abilities of the robot and assume we can observe the full state. It can not fully capture the data known while the agent plan. In our case, the robot is not fully aware of the location and type of the cubes around him. While the POMDP addresses it and adds a sensor model. The POMDP allows to better sense the agents surroundings but it fits only to a single agent setting, and does not include information about another agents or known changes in the surroundings.