Assignment 1 - Theoretical Assignment
AI and Robotics (236609)

Prunotto, Alice
alicep@campus.technion.ac.il
Computer Science Department
Technion - Israel Institute of Technology

Elashkin, Andrew
swag@campus.technion.ac.il
Computer Science Department
Technion - Israel Institute of Technology

November 28, 2022

1 Theoretical Assignment

1.1 State Space Search

1.1.1

(a) An heuristic function is said to be admissible if it never overestimates the true cost to the nearest goal: $h(x) \leq h^*(x)$.

Node A: $h(A) = 8$, $h^*(A) = 9$, so $h(A) \leq h^*(A)$.

Node B: $h(B) = 3$, $h^*(B) = 6$, so $h(B) \leq h^*(B)$.

Node C: $h(C) = 7$, $h^*(C) = 8$, so $h(C) \leq h^*(C)$.

So h is an admissible heuristic function.

An heuristic function is said to be consistent if, when going from neighboring nodes s to s', the heuristic difference step cost never overestimates the actual step cost: $\forall s \in S, \forall s' \in SUCC(s), h(s) - h(s') \leq COST(s, s')$. For example we consider B and its neighbour A, the shortest path is [A, C, B] which has a cost of 1+2=3, so $h(A) - h(B)$ should be less than 3, but $h(A) - h(B) = 8 - 3 = 5$. So the function is not consistent.

(b) Execution of A^*:

(c) The path to the goal found by A^* is [A, C, B, D].

(d) If cycle checking would be used, the path found would be [A, B, D] so the result would actually be worse.

And if path checking (detecting permutations of the same sequence of nodes) would be used the search would not provide a different solution.

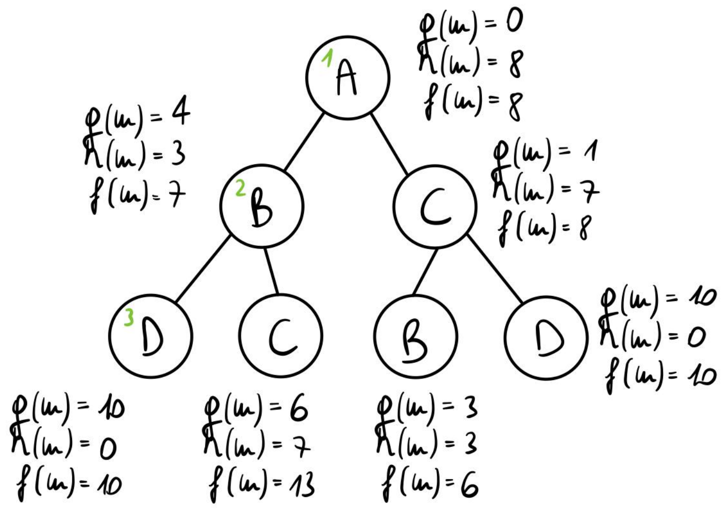

Figure 1: Execution of A*

1.1.2

While running A^* with heuristic h, a node that was in the closed-list was put in the open-list. This means that h is not consistent because for consistent heuristics every node that is put in the closed-list (expanded) guarantees an optimal path to it. So there would be no need to put a node back in the open-list because every new node would give a worse path.

So the answer is (iv).

1.1.3

While running A^* with some heuristic h, a goal node was expanded and put into the closed-list with value f. It is not possible that the value of the node will be updated at future steps because the algorithm stops .

1.2 Modeling

1.2.1

1. The models described below operate over discrete state space, so we assume that there exist a sampling function that transforms the environment from continuous domain to discrete state space.

   (a) For Classical Planning we will use Propositional STRIPS planning task defined by a tuple $P = \langle \mathcal{F}, I, \mathcal{A}, G, \mathcal{C} \rangle$, where

   - $\mathcal{F}$ is a set of fluents that represents the state of one specific block: if it is in the tower or not, if there is something on top of it, if it is in one of the sectors on the table. A state s is represented by the fluents that are true in s.
   - $I \subseteq F$ is the initial state.

- $G \subseteq F$ is the goal state where the part of fluents representing the inclusion of block in the tower are all $True$.
- $\mathcal{A}$ is the set of actions, each action encodes the movement of one specific block from one place to another. The precondition describe if this specific movement is legit, and add and del changes the state of the fluents in the current state.
- $\mathcal{C} : \mathcal{A} \to \mathbb{R}_0^+$ is a cost function that assigns each action a non-negative cost. Moving block on the table is more expensive than put it in the tower.

(b) Factored Markov-Decision Process (MDP) will be defined by the tuple $\langle \mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where

- $\mathcal{X}$ represents the location of each cube and the manipulator positions.
- $\mathcal{A}$ represents the actions manipulator can perform: extend joint 1, extend joint 2, rotate the base, rotate the grip, grip the cube.
- $\mathcal{P}$ describes a deterministic joint probability distribution.
- $\mathcal{R}$ is a reward function defining the reward of the agent controlling the manipulator that assigns positive reward for placing a cube in the tower and a smaller negative reward for each other action
- γ is a discount factor $\gamma \in [0, 1]$.

(c) We can assume that the agent uses camera to capture the environment state. That means, that there will exist some uncertainty about the actual environment state. A Partially Observable Markov Decision Process(POMDP) in this case will be a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \Omega, \mathcal{O}, b_0 \rangle$ where $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$ and γ are the same as in (b), and

- Ω is a set of observations derived from the camera.
- $\mathcal{O}$ is a sensor function.
- b_0 the initial belief.

2. (a) The real world is continuous, and since we need to sample it to use fluents in Classical planning, we can easily get a state explosion. Each block can not simply be encoded as a set of coordinates, but needs to be represented as a Boolean formula of some sort, if we actually want to solve the task. That makes the solution itself computationally exhaustive.

(b) Markov-Decision Process does not take into account the world uncertainty. By applying MDP we assume the complete knowledge of the environment, that can lead to errors. MDP also implies state independence, that can lead to infinite loops if the agent does not have memory of its own.

(c) With Partially Observable Markov Decision Process we can have an issue with the belief space, which is continuous and high-dimensional. The belief space will have one fewer dimension than the number of states in the model, making it an optimization problem over a continuous space with many thousands of dimensions, which is not feasible.

2 Practical Assignment

Practical part is in assignment1.py file.