National University of Singapore

CS4246/CS5446 AI Planning and Decision Making

Homework 2

Submission Deadline: **March 18, 2020 @ 23:59**

Please write the following at the top of your files (for both written assignment and programming assignment)

- Name
- Collaborators (write none if no collaborators)
- Source, if you obtained the solution through research, e.g. through the web.

On Collaboration

- While you may collaborate, you *must write up the solution yourself*.
- It is okay for the solution ideas to come from discussions. However, it is considered as plagiarism if the solution write-up is highly similar to your collaborator's write-up or to other sources.

Submission

- Your solution for both the written part and programming assignment should be submitted on LumiNUS.
  - For the written part, please type out your answers and submit a PDF file.
  - Name the file `[YourMatricNumber]_Homework1-Written.pdf`
  - For the programming part, you need to submit a zip file. Please follow the instructions at the end of the document clearly.
- For the programming assignment, code should be uploaded to aiVLE evaluation server **in addition to LumiNUS.**
- ***Late submission:*** You will incur a late penalty of 20% of your score for the late submissions.
  - If you submit the written and programming assignments at two different times, we will consider the later one as your submission time.
  - No submission will be accepted after March 22, 2020 @ 23:59.

---

1. **Markov Decision Process**

   (a) Assume that you are given a directed graph with n vertices and positive weights and would like to compute the shortest path from every vertex to a goal vertex g. Describe how to model the problem as a MDP so that it can be solved with an appropriately initialized value iteration algorithm, i.e. describe the state space, action space, transition function, and reward function.

   (b) Assume that we now have M agents on the same graph and the reward is the sum of the rewards of the agents. Can value iterations be used to solve the problem efficiently (as a function of M) in the following two cases? Why?

i. There is no restriction on the number of agents that can be at each vertex.
ii. Only one agent can be at each vertex at any time, i.e. having more than one agent at a vertex incurs a very large penalty.

(c) For the problem with M agents, the number of actions grows exponentialy with M. Suggest one way to allow each trial of Monte Carlo Tree Search with UCT to run efficiently.

2. **Partially Observable Markov Decision Process**

(a) One way to solve a POMDP is to treat it as a continuous MDP where each POMDP belief is a state in the MDP. We can then discretize the continuous MDP and solve the discretized problem using a discrete MDP solver. Discrete MDPs are known to be solvable in time polynomial in the number of states in the MDP. Hence the approach of discretizing the belief of the POMDP and solving the discretized problem as a MDP will also take polynomial time in the number of states of the POMDP. True of false? Justify your answer.

(b) Each trial in POMCP takes time exponential in the number of state variables as the size of the belief in a POMDP grows exponentially with the number of state variables. True of False, why?

3. **Monte Carlo Tree Search**

(a) In an MCTS simulation, a state has been visited 16 times and has two child nodes, A and B. Node A has a won 2 out 4 times whereas node B has won 8 out of 12 times. Which child node will the MCTS algorithm chose given the exploration parameter c is set to 1? Give the values of π_{UCT} for both the child nodes (consider log base 2 in UCT bound).

4. **Programming Assignment**

In the first homework, we learnt to solve deterministic planning problems with PDDL solvers. In this task, we will learn to solve planning problems in non-deterministic environments with two separate algorithms, Value Iteration and Monte Carlo Tree Search.

We will be using the same gym_grid_environment (https://github.com/cs4246/gym-grid-driving) to simulate the solutions. All dependencies have been installed in the docker image "cs4246/base" which you have already downloaded for Homework 1.

By now, you should be familiar with the gym_grid_environment. Go through the IPython Notebook file on Google Colab here, if you have already not done so in the previous programming assignment. You are now ready to begin solving the two tasks which are listed below.

(a) **Value Iteration:** In the crossing the road task in homework 1, we assumed all the cars move with a constant speed of -1. Most of the times, this assumption does not hold. There is inherent noise in the car speeds due to reasons like bumpy roads, uneven driving etc. This non-deterministic behaviour is not captured in the PDDL format.

For problems that are not very large, planning algorithms like Value Iteration can be used to handle non-deterministic behaviours. In this task, we model the problem as a Markov Decision Process and use Value Iteration algorithm to find the optimal policy.

We have provided the script which models the problem as an MDP. Every state in this MDP is represented as a tuple of features
$\langle \text{Agent}_x, \text{Agent}_y, \text{Car}^1_x, \text{Car}^1_y, \cdots, \text{Car}^N_x, \text{Car}^N_y, \text{isTerminal}\rangle$,
where:

- $\text{Agent}_x, \text{Agent}_y$ denote the x and y coordinates agent.
- $\text{Car}^N_x, \text{Car}^N_y$ denote the x and y coordinates of Car^N.
- isTerminal is a boolean variable denoting whether the state is terminal.

Your task is to fill up the function that implements the Value Iteration algorithm which is marked with "FILL ME".

You can test your code with the test configurations given in the script by running `python __init__.py` (present in the .zip file) on the docker (using the `docker run ...` command).

HINT : You can use Matrix multiplication to optimise the code, but refrain from using the function `np.matmul()`/`np.multiply()`/`np.dot()` as these functions interfere with the evaluation script. Instead you can use function `matmul()` defined in the same script.

(b) **Monte Carlo Tree Search :**

Unfortunately, we moved to a bigger parking lot and Value Iteration is not feasible as it does not scale well with large state spaces. However, we can use Monte Carlo Tree Search (MCTS) to handle such cases.

We have provided a separate script which solves the problem with MCTS, Your task is to complete the script by filling up 2 code snippets inside the functions marked with "FILL ME".

The functions required to be filled up are,

- backpropagate() : This function should implement the backpropation step of MCTS.
- chooseBestActionNode() : Populate the list bestNodes with all children having maximum value. The value of the i^{th} child node can be calculated with the formula $\frac{v_i}{n_i} + c\sqrt{\frac{\log N}{n_i}}$. where,
  - v_i : sum of returns from the i^{th} child node
  - n_i : Number of visits of the i^{th} child node
  - N : Number of visits of the current node

- c : The exploration constant. We set it to be 1.

  For more details on MCTS, it might be helpful to look here. You are also encouraged to look through the rest of the code to see how the entire MCTS algorithm is implemented.

You can test your code with the test configurations given in the script by running `python __init__.py` (present in the .zip file) on the docker (using the `docker run ...` command).

Submission Instructions

Please follow the instructions listed here. You have to make 2 separate submissions as mentioned below :

i. For Task (a) : You have been provided with a vi_agent.zip file in the correct submission format. Complete the functions marked with "FILL ME" in `__init__.py`. Your submission zip file should have the exact same structure as the zip file you received.

ii. For Task (b) : Modify the `__init__.py` in the mcts_agent.zip by completing the functions with "FILL ME" and make a separate submission.

Note :

i. Please remember to set the variable SUBMISSION in `__init__.py` to False when testing locally, and to True before making a submission.

ii. Please do not print anything to the console, as it might interfere with the grading script.