Planning Algorithms in AI
PS3: Decision Making

Gonzalo Ferrer
Skoltech

2 December 2022

This problem set has two tasks and is individual work. You are encouraged to talk at the conceptual level with other students, discuss on the equations and even on results, but you may not show/share/copy any non-trivial code.

Submission Instructions

Your assignment must be submitted by 11:59pm on **Dec 14, Wed**. You are to upload your assignment directly to Canvas as **two** attachments:

1. A `.tar.gz` or `.zip` file *containing a directory* named after your name with the structure shown below.

```
yourname_ps3.zip:
yourname_ps3/run.py
yourname_ps3/utils.py
yourname_ps3/vi.py
yourname_ps3/mcts.py
yourname_ps3/escape_vi.mp4
yourname_ps3/escape_mcts.mp4
```

2. A PDF with the answers to the questions below. Printed notebooks or scanned versions of hand-written documents, converted to PDFs, are perfectly acceptable (reduced size). No other formats (e.g., .doc) are acceptable. Your PDF file should adhere to the following naming convention: `yourname_ps3.pdf`.

Homework received after 11:59pm is considered late and will be penalized as per the course policy. The ultimate timestamp authority is the one assigned to your upload by Canvas. No exceptions to this policy will be made.

Problem Description: Escape the Maze

In this problem set you are asked to play the game of *Escape the Maze*. The environment is a simple 2d grid world with some obstacles. In this game, there are two players, one is the **evader** (yellow dot) who wants to escape the maze by reaching the goal state. The second player, the **pursuer** (green dot) who wants to catch the evader.

You can find the environment in the file `data_ps3.npz` and visualize it by simply executing the `run.py` script.

To support you and your algorithm, we provide a series of function found in the `utils.py` file:

- `action_space`: This list contains all available actions.
- `plot_joint_enviroment`: This function is for visualization.
- `transition_function`: This is an implementation of $x' = f(x, u)$ for both the pursuer and evader.
- `state_consistency_check`: This function checks whether or not the state is valid, i.e., is an obstacle or out of bounds.
- `pursuer_transition`: A compact function that calculates the next action of the pursuer and includes transition function. The nature of the pursuer policy is probabilistic, meaning that their action is sampled from the distribution $\pi(u|x)$. To make things more interesting, the pursuer is fast and moves sometimes two actions consecutively. This is already incorporated in the code, there is no need to change anything.

Look for more details in help and in the code of these functions.

Task 1: Value Iteration

In this task, you will calculate the optimal policy for the evader to move. For now, the pursuer remains static.

A. (5 pts) Enumerate the action space. The coordinates of actions are $u = (\text{row}, \text{column})$.

B. (10 pts) Formulate the optimal cost-to-go G^* in recursive form. Also formulate how to obtain the optimal policy u^* from G^*.

C. (30 pts) Implement the VI algorithm for infinite length sequences. To show this, you are asked to include a picture of G^* after convergence. Attach a video of the animation.

   The cost of traversing each node $l(x, u) = 1$ only if propagation is possible (there is not obstacle or out of bounds).

   Hint: You may consider a terminal action u_T, with zero cost once the goal state is reached. Otherwise, VI will not converge (this is the K length plan problem).

D. (5 pts) Un-comment the pursuer transition function in line 54. Did you escape?

Task 2: Monte Carlo Tree Search

For this task, you are asked to implement MCTS with some considerations:

The first consideration/variation is on the **selection** and **expansion** parts. The original implementation of MCTS, as explained in class, expands the most promising node. Now, we will implement it in the following way:

1. We will start from the root node, follow the *tree policy* by looking at the best value of the children.
2. If the node accepts expansion, i.e., there are actions from the current node that expands the tree into new valid states, then we do. If not, then we go deeper into the policy tree and select the best next node among children according to their current values.
3. We will only expand to **non-visited** states. This is a variation from the MCTS explained in class, where building a tree may include cycles and recursive forms. This will be a question below. This strategy is similar to the breath-first algorithm from L02.
4. With this strategy, we might end up in a state that we can't propagate and that has no children. If that happens, that is fine, the expansion phase is skipped and we simply run the **simulation** step.

The second consideration is on the cost function. Designing a cost function is always a complex process, where tuning may be required. In this case, we want to reward not just escaping the maze, but also doing it fast.

For this reason, we will use the following reward at the terminal time T (when the simulations stops):

$$R_T(x_e, x_p) = \begin{cases} 0 & \text{if } x_e = x_p \\ 100 + \sum_t^T \frac{0.1}{d(x_e(t) - x_{goal})} - T & \text{if } x_e = x_{goal} \\ \sum_t^T \frac{0.1}{d(x_e(t) - x_{goal})} - T & \text{otherwise} \end{cases}$$

Use as the pursuer simulation function the `pursuer_transition` from `utils.py`. For the evader, you may use as the default policy the optimal policy to get to the goal, although it is optimal only in the number of steps.

A. (10 pts) Why do you think we have proposed the restriction of expanding only non-visited states?

   Hint: Calculate the number of nodes in a tree of depth 5.

B. (40 pts) Implement MCTS. Show a video of the evader trying to escape. You may start using 100 episodes in the MCTS algorithm and run simulation of a maximum of 50 samples/iterations per each episode. It might fail some times. Attach a video with the results.

   Hint: You may want to use a data structure to keep track of the number of iterations each node played, average value, parent, children and if a node has been visited or fully expanded.

C. (extra) Tune further the MCTS algorithm to improve its performance. Describe your proposed improvements and explain why you choose them.