cs5478 assignment3

Shicheng Chen A0215003A

March 2020

1 Question One

For the DSDA model, what search algorithm do you use and why? If the A^* algorithm is used, what is the search heuristic?

I use a Breadth-first search algorithm.

If the agent transfer from state A to state B validly, the path length is only 1. There are only limited states for mazes and com1. We can get an exact result very fast, so I did not use the A^* algorithm.

The evaluation function only cares about the length of the actual path that the robot traverse and whether the robot succeeds in reaching the goal without collision. Therefore, we do not need to care about the cost of LEFT and RIGHT actions.

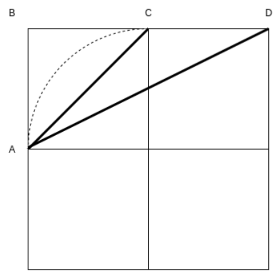
2 Question Two

For the CSDA model, how do you discretize the state space and the action space?

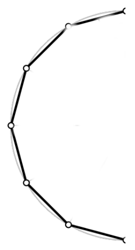
I discretize mazes and com1 as fine as possible. For example, the com1 map is 2500 by 983, and I get 2500*983=2457500 states. As we mentioned before, there is no cost on LEFT and RIGHT actions, so we do not need to include directions into states. We only care about the distance.

If we want to go to D from A, we will have several choices, as shown in figure 1a. Apparently, $A->D$ is shorter than $A->C->D$ or $A->B->C->D$. We can find that the dashed curve is longer than AC. Additionally, we can find that straight lines can always replace curve lines, as shown in figure 1b, so our agent only has two types of actions. The first one is FORWARD action; the other is turns actions. If a north-facing agent is at A, the agent can turn $0°, 45°, 63.43°$, or $71.56°(\arctan 3)$... clockwise. Noticeably, we do not execute moving action and turning simultaneously. For example, we only have $(0, v)$ or $(w, 0)$.

As we use such a way to discretize the state space and actions, here we choose to use Dijkstra's algorithm. It is a greedy algorithm, and it can handle a lot of states fast. The weight of the edge is the length of the edge. For example, the weight of AD is $\sqrt{5}$. Our agent only stops in the side of straight lines such



(a) Path AD is shortest path from location A to location D


(b) Straight lines can always replace curve lines

as A, B, C, D, which means that it cannot stay in the middle of the path line AB.

3 Question Three

For the DSPA MDP model, what reward function do you use? How do you solve the MDP?

We first show our states:

1. Each position represents a state. S_g, S_w, S_i denote goal state, wall states, and normal states, respectively.
2. Goal state S_g means the agent is at the goal.
3. Wall state S_w means that the distance between the agent and the wall is less than 1 unit.
4. Normal states $S_0, S_1 ... S_i$ show that the agent is in other places instead of the places we mentioned before.

Now, we show our rewards:

1. $R(S_w) = -1000$ represents that the agent goes to a position which is next to the wall, there will be a big negative reward since we do not want the agent to collide with the wall. The award of $R(S_w)$ needs to be tuned for different shortest paths to get a better SPL score.
2. $R(S_i) = -1$ shows that we want the agent to go to a goal as soon as possible.

Here, we do not include the action, since there is no cost on LEFT and RIGHT actions for the average success weighted by path length(SPL).

Initialize $V_0(s) = 0$

$$V_t(s) = R(s) + \Sigma_{s' \in S} T(s, s^{'}) V_{t-1}(s^{'})$$

for $t = 1, 2, 3...N$. Where $T(s, s^{'})$ is the probability of transition from state s to state $s^{'}$. For example, the detail of the function can be $V_t(s) = R(s) + 0.9V_{t-1}(s_0) + 0.05V_{t-1}(s_1) + 0.05V_{t-1}(s_2)$

3.1 How to use the code

```
python base_planner.py --goal '9,9' --com 0 --plan 2. #for the second plan
python base_planner.py --goal '9,9' --com 0 --plan 1. #for the first plan
```

I discretize mazes and com1 as fine as possible in the second question, so the control files include very high precision numbers. Please read in high precision method.