AE for AI – Homework #3

Question 1

Question 2

An agent is located in a 2D grid world with the task of arriving at the goal. The problem is a Markov Decision Problem (MDP), in particular a Stochastic Shortest Path (SSP) problem (see Bertsekas's **Approximate Dynamic Programming** book) which can be solved via Value Iteration (VI) and Policy Iteration (PI). Traditional MDP graph search algorithms cannot be used since this is a SSP problem. A visualization of the type of problem can be seen in the figure below (from Dan Klein's MDP slides).

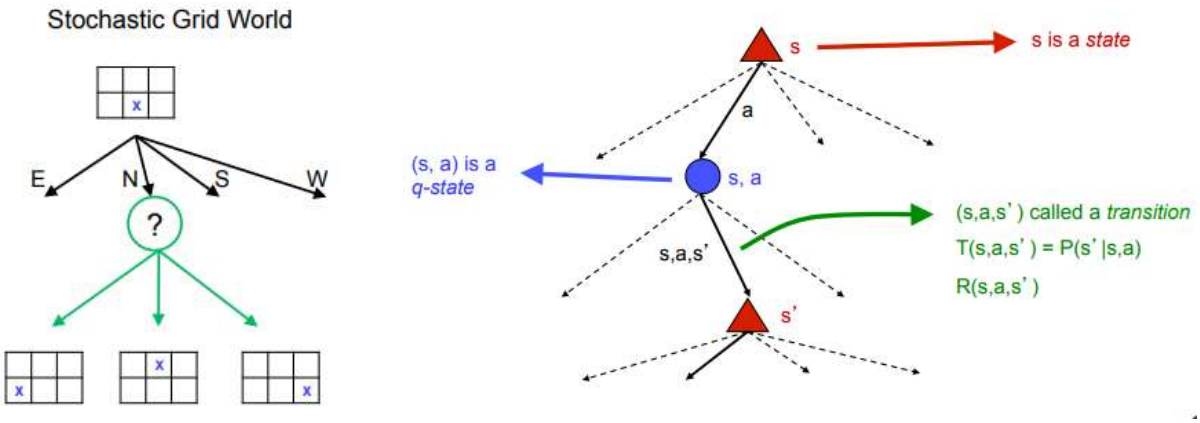
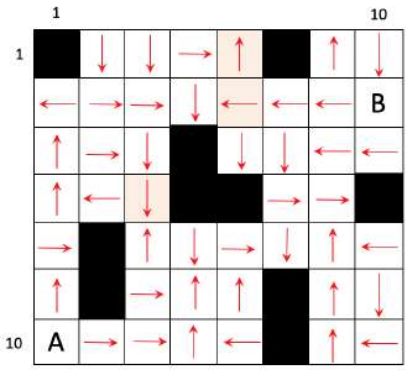


Figure 1: Stochastic MDP visualizations from Dan Klein's slides

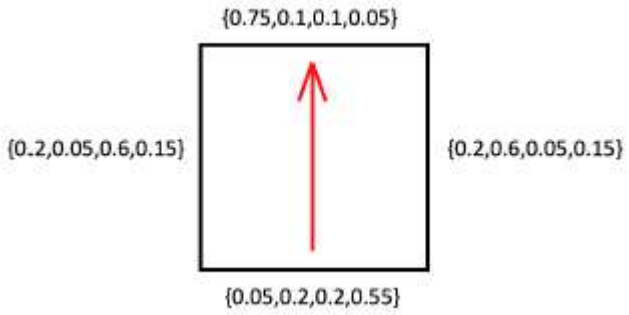
The implementation of VI or PI requires the problem setup as an MDP, which includes the state space, action space, transition matrix (probability matrix), and reward function.

Problem Setup

The initial condition, terminal condition, and state space was given as a 7 x 8 grid world with obstacles and disturbances.



The action space was defined by the Up, Down, Left, Right movements in the grid world. In addition, the disturbance information was given in the form below:



This can be translated into a pseudo - Transition Matrix as shown here:

```
[('WithWind', [0.75, 0.10, 0.10, 0.05]),
('AgainstWind', [0.05, 0.20, 0.20, 0.55]),
('SideWindR', [0.20, 0.60, 0.05, 0.15]),
('SideWindL', [0.20, 0.60, 0.05, 0.15])]
```

The reward function was dependent on the state. Each open grid cell was given a reward value of -1 (negative to influence less steps). The goal state was given a reward of 10 and the obstacles were given rewards of -10.

NEED TO FIX DESCRIPTION ABOVE

NEED MORE DETAILS ON IMPLEMENTAITON WITH ABSORBING STATES AND BOUNCING STATES AT TERMINALS

Results

Value Iteration was implemented in Python and the results are shown below.

Part a)

Initial results

[[-10. -11. -9. -8. -7. -10. 6. 8.]

[-11. -10. -8. -6. -3. 1. 6. 10.]

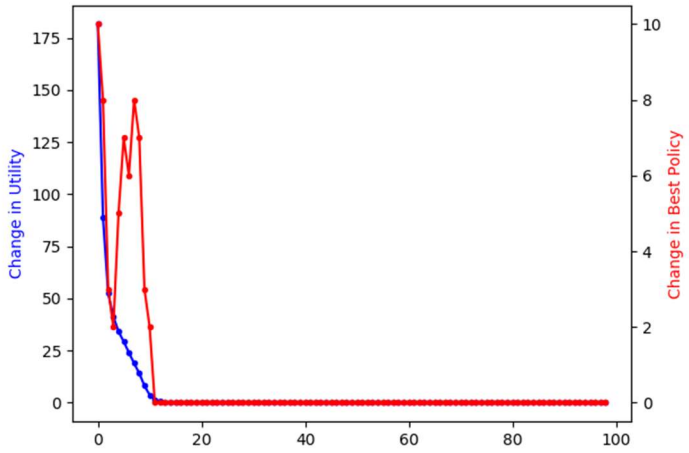
[-13. -12. -10. -10. -4. 1. 4. 6.]

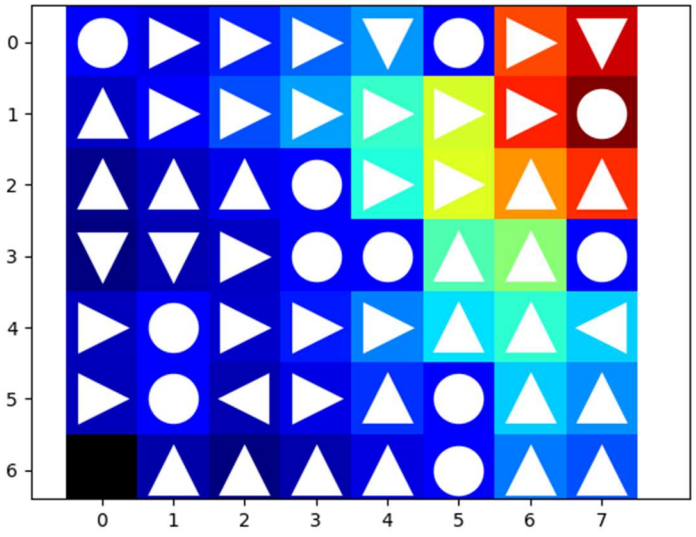
[-13. -12. -11. -10. -10. -3. -1. -10.]

[-12. -10. -11. -9. -7. -5. -3. -5.]

[-12. -10. -12. -11. -9. -10. -5. -7.]

[-1. -12. -13. -12. -11. -10. -7. -8.]]





BUT GET STUCK IN FIRST FEW STEPS BECAUSE OF THE COST

If I change default reward to -0.5, results are better:

[[-10. -6. -5. -4. -4. -10. 7. 9.]

[-8. -6. -5. -4. -0. 2. 8. 10.]

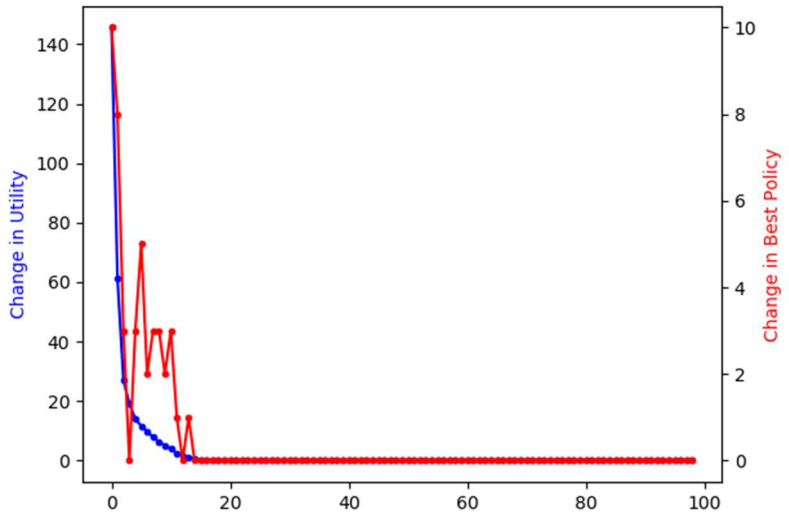
[-8. -7. -7. -10. -1. 3. 6. 7.]

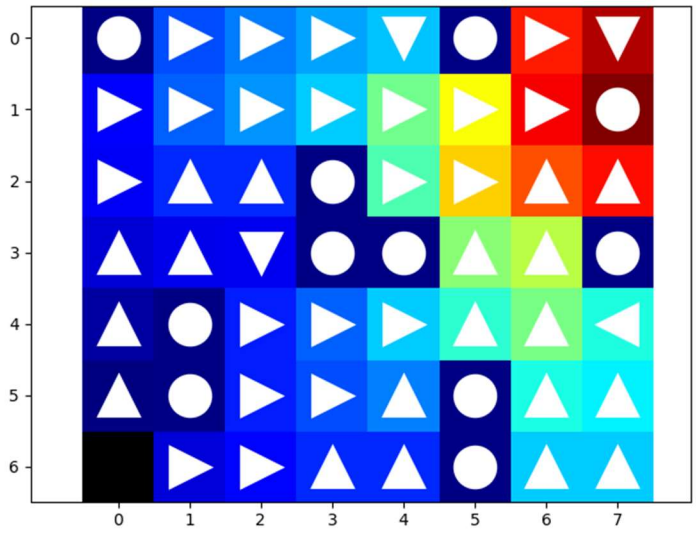
[-8. -8. -8. -10. -10. 0. 1. -10.]

[-9. -10. -7. -6. -4. -2. -0. -2.]

[-10. -10. -7. -6. -5. -10. -2. -3.]

[-0. -8. -7. -7. -7. -10. -4. -4.]]





Using the default reward, 16/50 cases result in arriving at the goal.

b)

Special reward = -0.3

Appendix

The code for this work can be found at: https://github.com/calebh94/path-planning.

Any questions can be sent to Caleb Harris at caleb.harris94@gatech.edu.