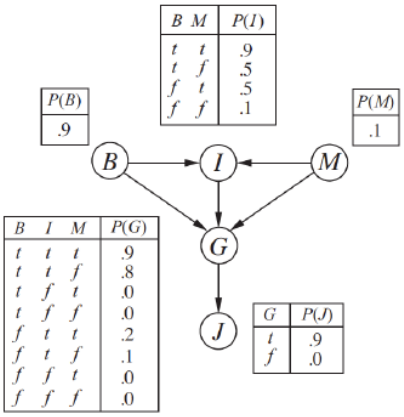
AE for AI – Homework #3

Question 1

Given the Bayesian Network here:



Part a)

Using the definition of a Bayesian Network we know that:

$$P(B, M, I, G, J) = P(B)P(M)P(I|B, M)P(G|B, M, I)P(J|G)$$

Also, remembering Bayes Theorem:

$$P(A|B) = \frac{P(A, B)}{P(B)}$$

The network structure assert that:

iii) P(M|G,B,I) = P(M|G,B,I,J)

Part b)

The probability that: B = true, I = true, M= false, G= true, J= true

$$P(B = T, I = T, M = F, G = T, J = T)$$

$$= P(B = T)P(M = F)P(I = T\,|B = T, M = F)\,P(G = T\,|B = T, M = F, I = T)P(J = T\,|G = T)$$

$$= (0.9)(1 - 0.1)(0.5)(0.8)(0.9)$$

$$= 0.2916$$

Part c)

$$P(J = T|B = T, U = T, M = T)$$

$$= \alpha\, P(J = T, B = T, I = T, M = T)$$

$$= \alpha \sum_{G} P(J = T, B = T, I = T, M = T)$$

$$= \alpha\, P(B = T)P(M = T)P(I = T\,|B = T, M = T) \sum_{G} P(G|B = T, M = T, I = T)P(J = T\,|G)$$

$$= \alpha\, [(0.9)(0.1)(0.9)]\, \{(0.9)(0.9), (0.1)(0.0)\}$$

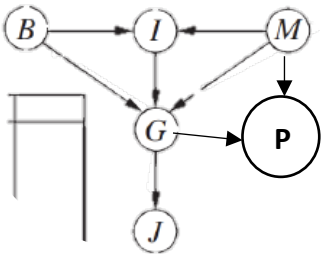
$$= \alpha\, \{0.06561, 0.0\}$$

If FoundGuilty (G=T): 0.06561

If not FoundGuilty (G=F): 0.0

Part d)

Define PresidentialParton = P



The assumption is made that a PresidentialPardon is only dependent on whether or not M (PoliticallyMotivatedProsecutor) and whether or not G (FoundGuilty).

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).

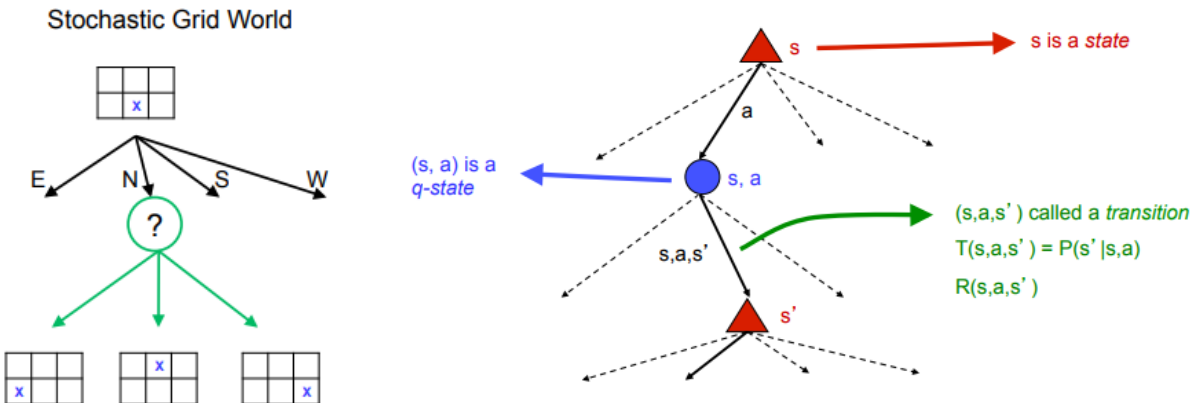
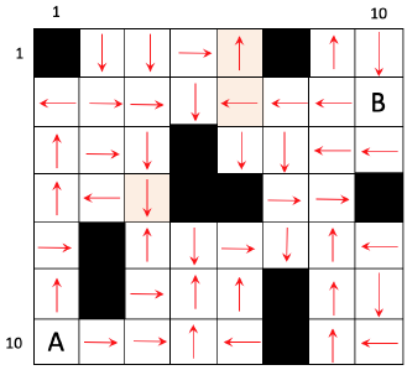


Figure 1: Stochastic MDP visualizations from Dan Klein's slides

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

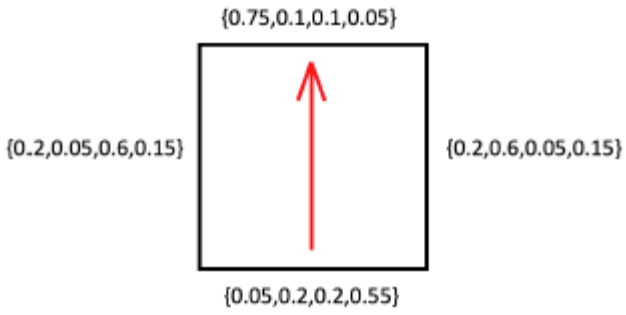
The initial condition, terminal condition, and state space was given as a 7 x 8 grid world with obstacles and disturbances.



The goal of the problem is to navigate the environment from the start (A) to the goal (B). Therefore the obstacle spaces, boundary spaces, and start/goal locations must be properly defined. The environment was chosen as followed:

- Moving towards a square outside of the grid results in "bouncing" back, therefore the chosen action will result in the agent being in the same space and earning the reward in that space.
- Moving into a "obstacle" is treated as a "hole", meaning that it is an absorbing state that results in a reward of (-10) and ends all future movement. Therefore, moving into this space will results in a failed mission.
- The start space is treated as a free movement with no wind, and is then blocked off as an obstacle. This is because if it is treated as a normal space it will influence actions that stay in the bottom corner, since the (-1) reward is much higher than the (-10) rewards.
- The goal space is an absorbing state that results in a (-10) reward. Moving into this space results in a successful mission.

The action space was defined by the Up, Right, Down, Left {U,R,D,L} movements in the grid world. These were encoded as integers {0,1,2,3} which were then used in accordance with the disturbance direction {0,1,2,3} to determine the "case" of the relative directions of the wind and action-space. This



is critical since the transition matrix is a function of the disturbance and the action. The disturbance information was given in the form below:

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

The full transition matrix $P(s'|s,a)$ is calculated using the function $\_create\_transition\_matrix()$. This function loops through each action, disturbance, and resulting next states to create a matrix:

$$T[r0, c0, a, w, r1, c1]$$

$$r0, c0 = initial\ row - column\ state$$

$$r1, c1 = next\ row - column\ state$$

$$a = action$$

$$w = disturbance$$

```
(0, [0.75, 0.10, 0.05, 0.10]),  # Action 0 deg
(1, [0.20, 0.60, 0.15, 0.05]),  # Action 90 deg right
(2, [0.05, 0.20, 0.55, 0.20]),  # Action 180 deg right
(3, [0.20, 0.05, 0.15, 0.60])   # Action 270 deg right
```

Temporary masks assist in creating "bouncing" states for the states outside the grid. The transition matrix is initially made a function of the disturbance (w) to assist in the efficiency of the code and to provide flexibility for a changing environment.

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.

However, it will be seen in the results section that by selecting a default reward value of -1 the agent was unable to make it to the goal. This is because of the way the obstacle costs propagate outwards via Value Iteration causing the resulting in lower (higher negative) rewards around some obstacles. Therefore, the agent is influenced to take actions towards these obstacles. By changing the default reward to 0.5 or lower, the agent is influenced towards the goal.

Results

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

Part a)

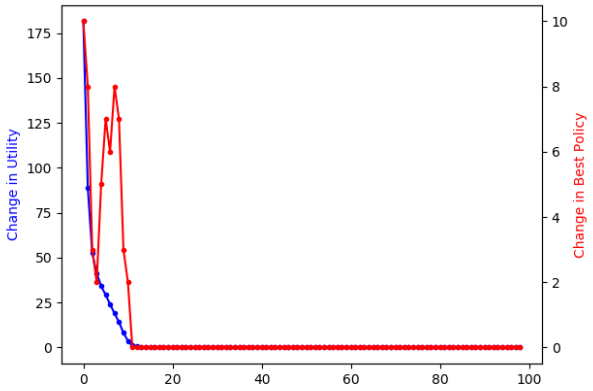
Value Iteration was used with a discount factor of 0.95 and the environment had a default reward of -1.0 and no special rewards.

The final utility matrix:

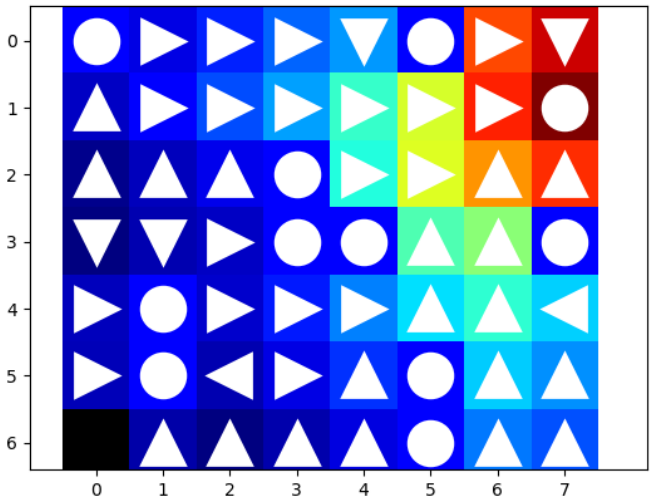
```
[[-10. -11.  -9.  -8.  -7. -10.   6.   8.]
 [-11. -10.  -8.  -6.  -3.   1.   6.  10.]
 [-13. -12. -10. -10.  -4.   1.   4.   6.]
 [-13. -12. -11. -10. -10.  -3.  -1. -10.]
 [-12. -10. -11.  -9.  -7.  -5.  -3.  -5.]
 [-12. -10. -12. -11.  -9. -10.  -5.  -7.]
```

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

The convergence plot:



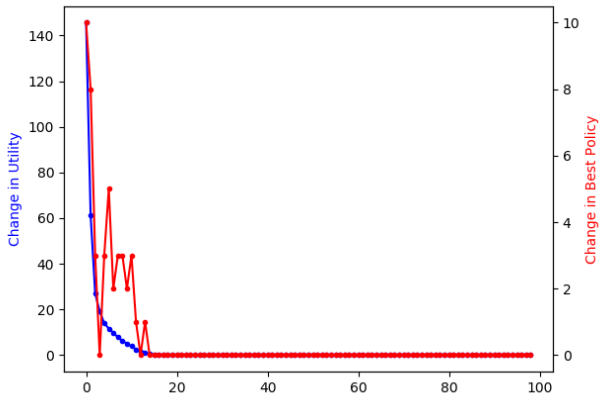
The resulting optimal policy, with color scale of expected utility:

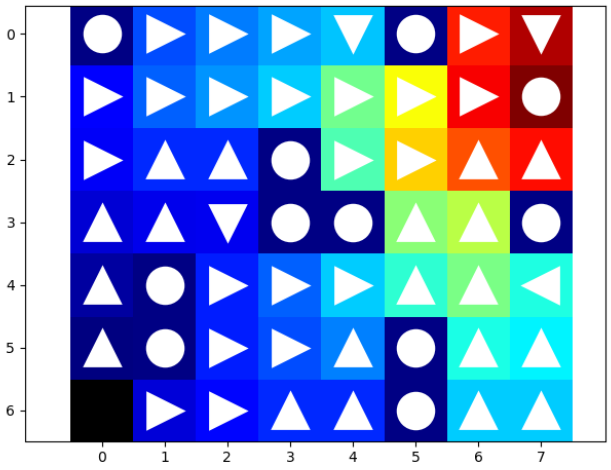


As mentioned earlier, and can be clearly seen from the above, the resulting policy results in actions into the absorbing obstacle states near the start state.

Therefore the default reward is changed to -0.5, resulting in the following:

```
[[-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.]]
```



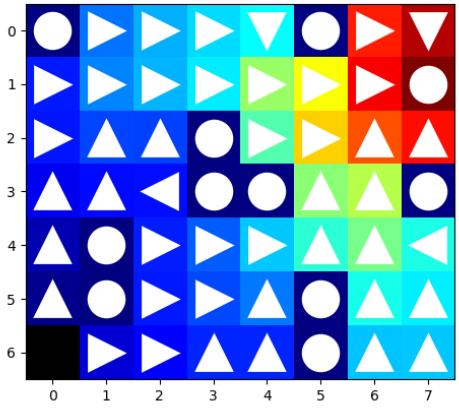


The resulting optimal policy clearly shows an influence to move toward the goal state.

Part b)

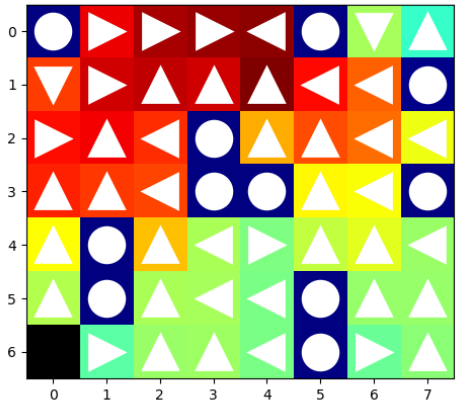
The problem was repeated using special rewards for cells (4,3), (1,5), and (2,5).

Special reward = 0.0



The resulting policy for the special reward of 0.0 is very similar to the previous policy, largely due to the small difference between the rewards. This is partially because of the reduced default reward to -0.5. However, it can be seen that the expected utility is different between the two cases as shown by the shade of blue.

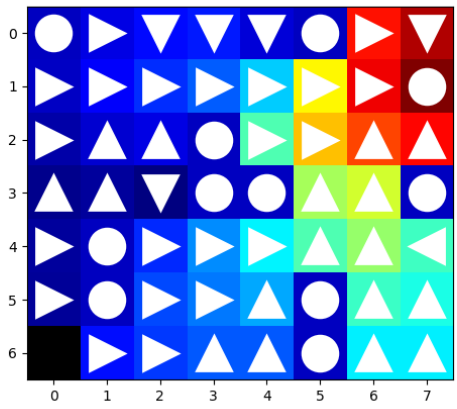
Special reward = 100.0



The resulting policy for a special reward of 100.0 is very different than the default case. This is due to the special rewards being higher than the goal reward. In this case, the total reward earned by the agent would be much higher, however the overall task of arriving at the goal location would most likely not be achieved.

Special reward = -3.0

The resulting policy for a special reward of -3.0 is slightly different than the default case. The expected utility of the special grid cells is slightly higher, resulting in the optimal policy attempting to steer clear of these locations.



Part c)

Using the default reward of -0.5, no special rewards, and the optimal policy found via Value Iteration the agent was able to successfully navigate the environment anywhere from 12 to 16 out of 50 times. The most recent runs are shown below:

Successes: 12/50

Average Reward: 1.808

Rewards: `[0. 4.5 -0.5 0.5 0.5 3.5 3.5 2.5 0.5 3.5 1.5 1.5 2.]`

The failed cases often land in (5,1), the first absorbing obstacle.

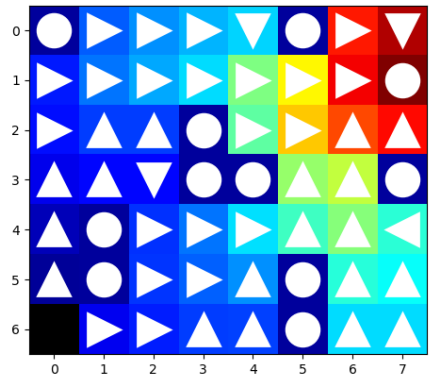
This result is as expected for the way the problem was setup. In previous iterations of the problem there were some differences such as the default reward being lower, the obstacles treated as normal squares or “rebounding states” and others. In these cases, I saw policies that looked unlikely to achieve the goal. This is how the final iteration of the problem setup was achieved and I believe it works well for the complex environment.

Question 3

The previous problem was repeated using Policy Iteration.

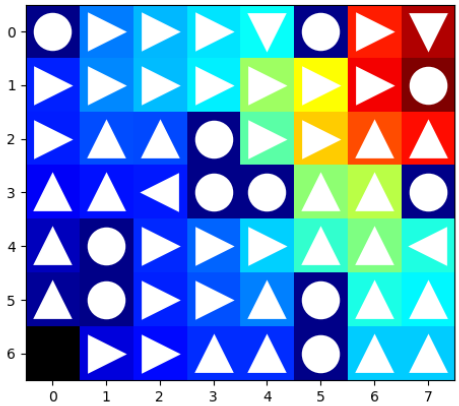
Part a)

Using Policy Iteration, a successful policy can be found with the default reward of -1.0, however to stay consistent with the previous problem a reward of -0.5 is used.

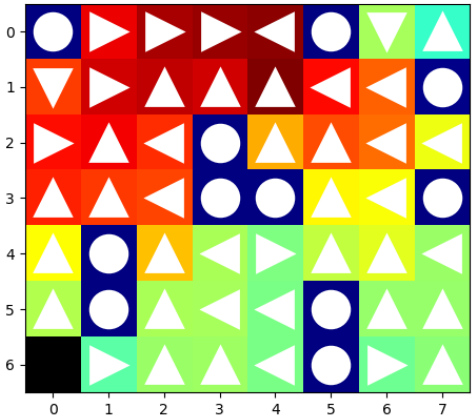


Part b)

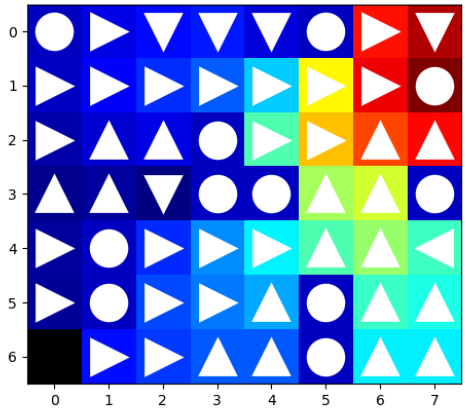
Special reward = 0.0



Special reward = 100.0



Special reward = -3.0



The special rewards all resulted in policy differences similarly to the Value Iteration problem. Please review those results for further discussion.

Part c)

Using the default reward of -0.5, no special rewards, and the optimal policy found via Policy Iteration the agent was able to successfully navigate the environment around 17 of 50 times. The most recent runs are shown below:

Successes: 17/50

Average Reward: 1.25

Rewards:

```
[0 0.5 2.5 1.5 4.5 4. 1. 0 0.5 3.5 -0.5 0.5 4.5 -4.5 1. 4.5 1.5 -2.5]
```

Discussion

In comparing Value Iteration and Policy Iteration the success of the resulting policy and the time requirements can be compared. In this small grid problem the memory-usage is similar.

	Successes (%)	**Run Time (s)**
Value Iteration	24	0.116
Policy Iteration	34	0.027

It is documented that policy iteration converges faster and the methodology leads to other techniques in reinforcement learning and other areas which start to discuss the tradeoffs of exploration and exploitation. Therefore, policy iteration does seem to be the proper choice for problems such as this.

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.