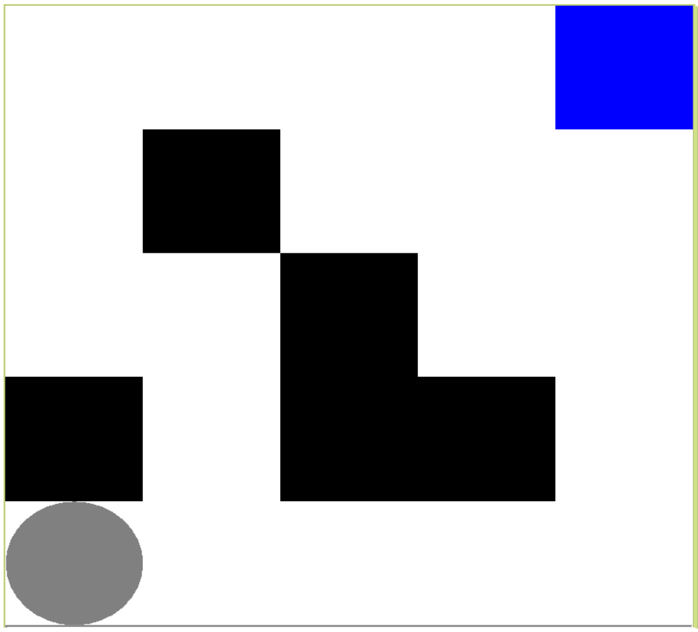
Jassimran Somal (jkaur9)

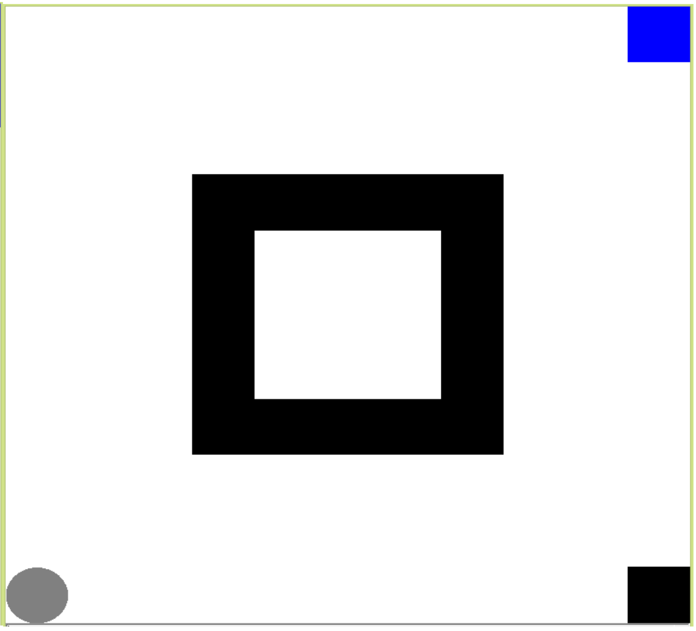
Machine Learning - Assignment 4

PROBLEMS

Last semester I took Artificial Intelligence for Robotics taught by Sebastian Thrun and we learned about few route planning algorithms for the robotic car. We mostly used dynamic programming and A-star. For this assignment, I chose similar grid domain problems which resemble with path planning in robotic car. I tried few other problems as well which are MDPs but having domain knowledge really helps to not only excel but one needs to know what you are really looking for. Hence these problems were very simple to implement and understand the working of Value Iteration and Policy Iteration. They have limited states and it is very easy to see if the generated solution is optimal. The processing of the problems can be visualized which helps in understanding algorithms better. Both the problems are shown below:



Problem 1



Problem 2

Problem 1 is a 5×5 grid where bottom-left is the starting position and top-right is the goal position. The reward at Goal postion is defined as 100, and -1 elsewhere. The actions that can be performed at any state are North, South, East or West. It has 5×5=25 states possible, however 5 among these states are unreachable. Hence, total number of states are 20.

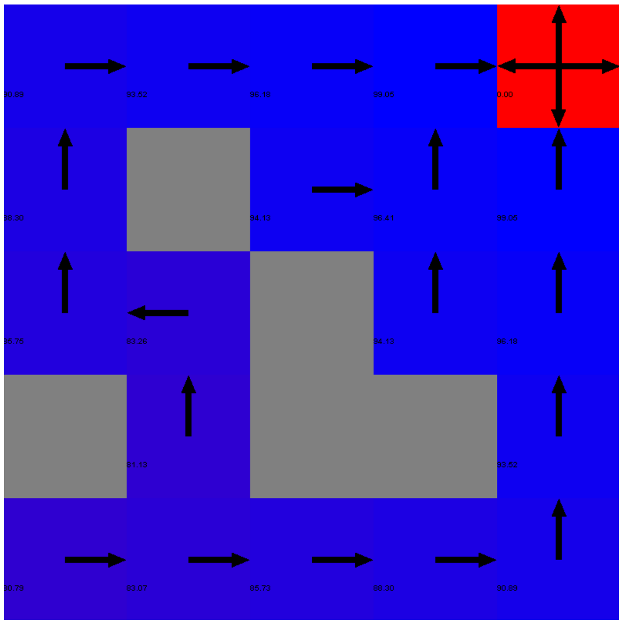
Problem 2 is a 11×11 grid where bottom-left is the starting position and top-right is the goal position. The reward at Goal postion is defined as 100, and -1 elsewhere. The actions that can be performed at any state are North, South, East or West. It has 11×11=121 states possible, however 17 among these states are unreachable. These further make another 9 states unreachable. Hence, total number of states are 95.

Also, grid domain problems also represent many more real world problems as well like most of the shortest-path planning problems, a lot of games and etc.

Value Iteration:

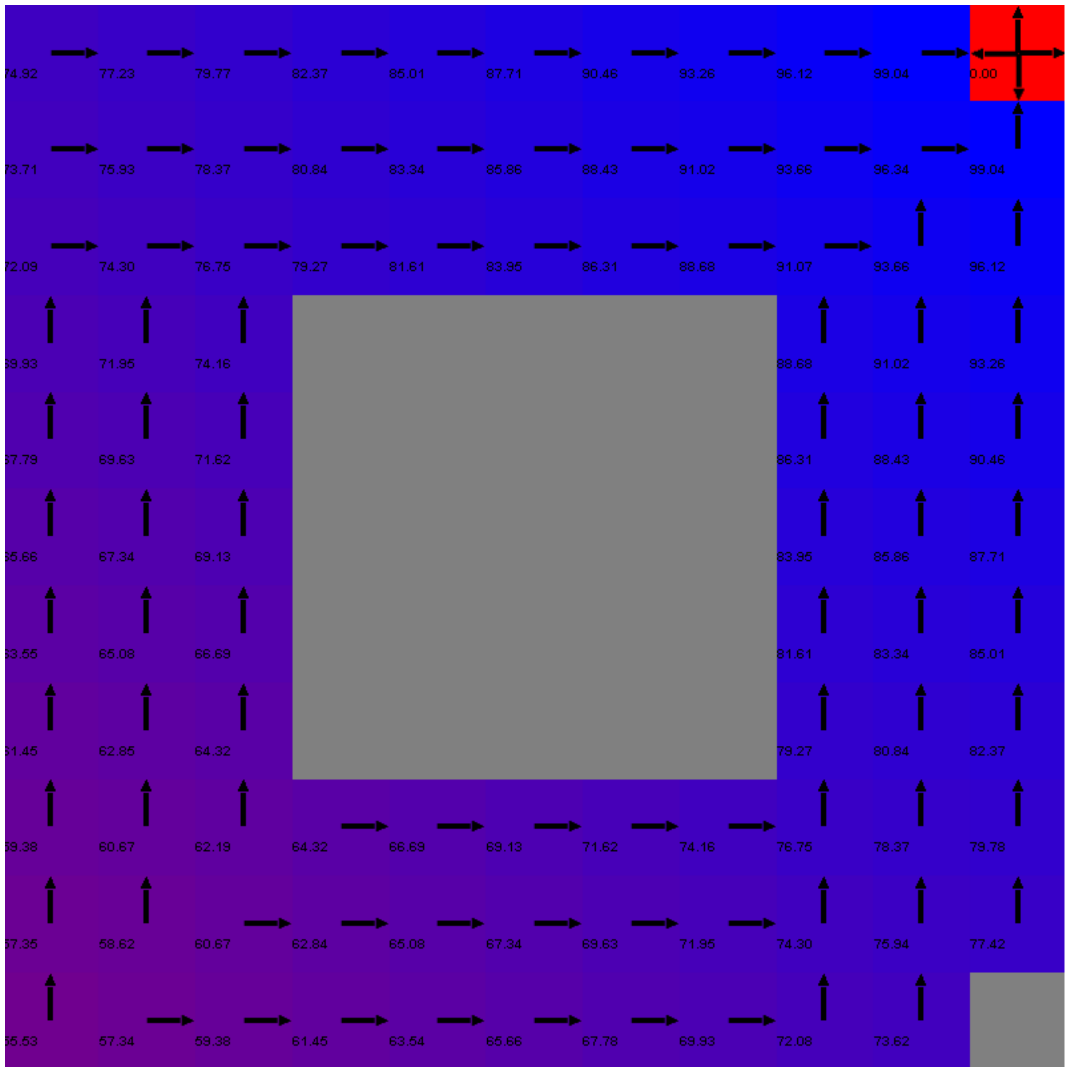
Problem 1:

```
Starting reachability analysis
Finished reachability analysis; # states: 20
Passes: 20
Num generated: 260; num unique: 20
```



Problem 2:

```
Starting reachability analysis
Finished reachability analysis; # states: 95
Passes: 29
Num generated: 1500; num unique: 95
```



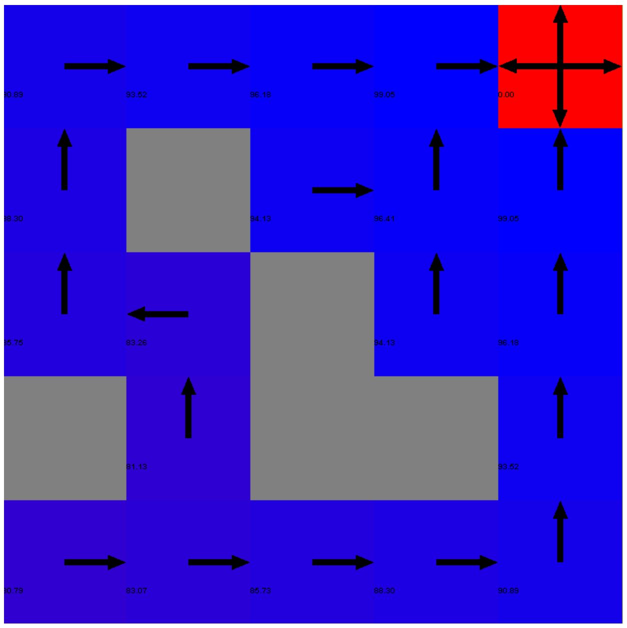
Policy Iteration:

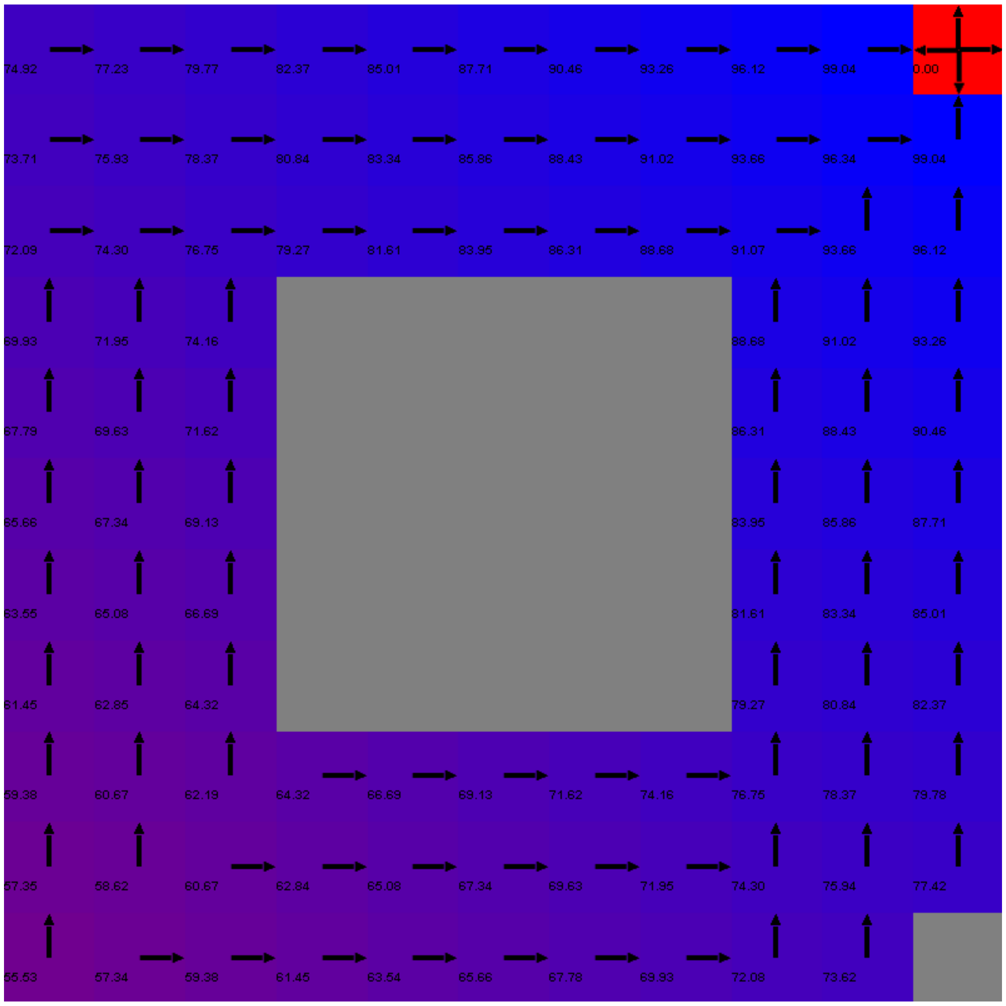
Problem 1:

```
Starting reachability analysis
Finished reachability analysis; # states: 20
Passes: 12
Num generated: 260; num unique: 20
```

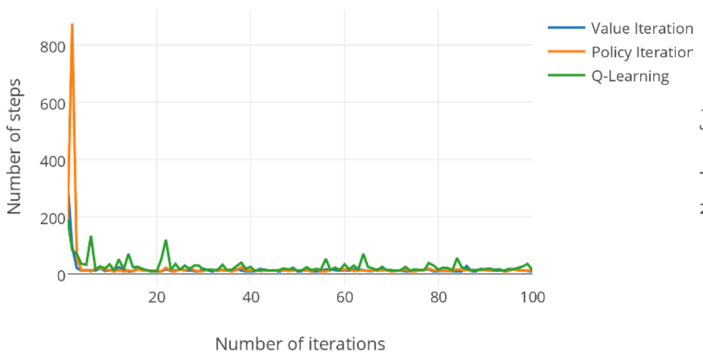
Problem 2:

```
Starting reachability analysis
Finished reachability analysis; # states: 95
Passes: 19
Num generated: 1500; num unique: 95
```

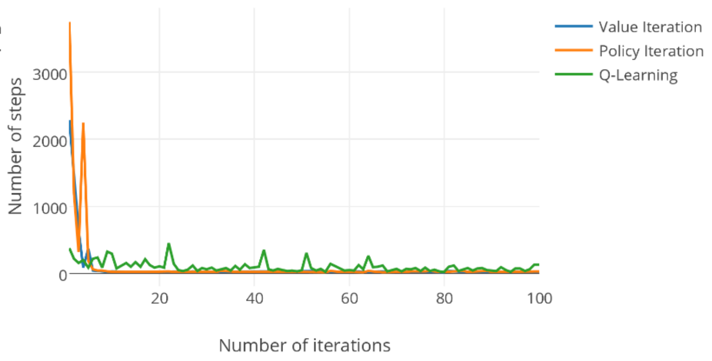




Problem1: Iterations vs Number of steps needed to reach goal

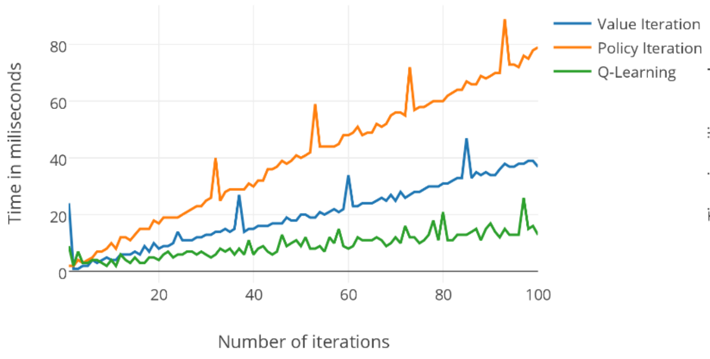


Problem2: Iterations vs Number of steps needed to reach goal state

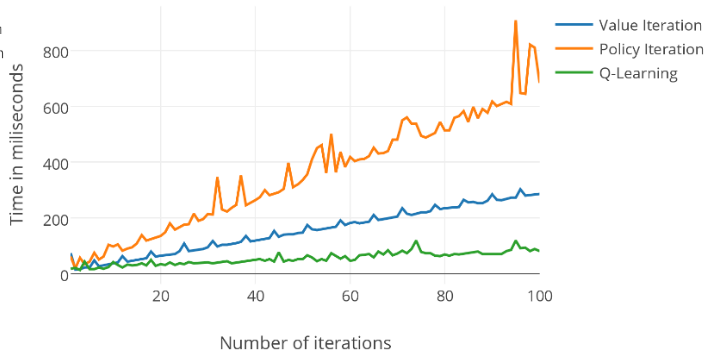


As shown above, both Value Iteration and Policy Iteration generate approximately same value function. The policy is same, with same number of steps in the plan.

Problem 1: Iterations vs Time taken to generate optimal policy

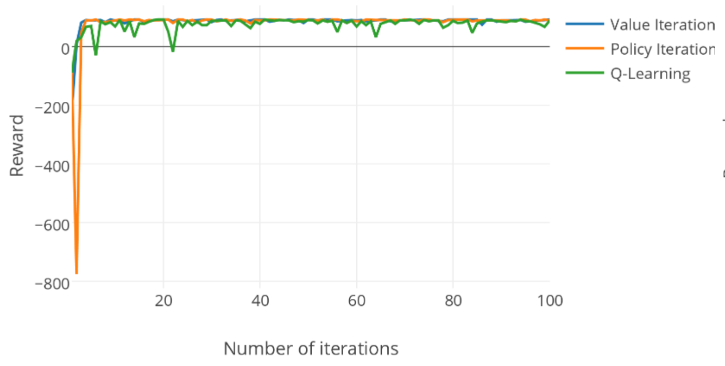


Problem2: Iterations vs Time taken to generate optimal policy

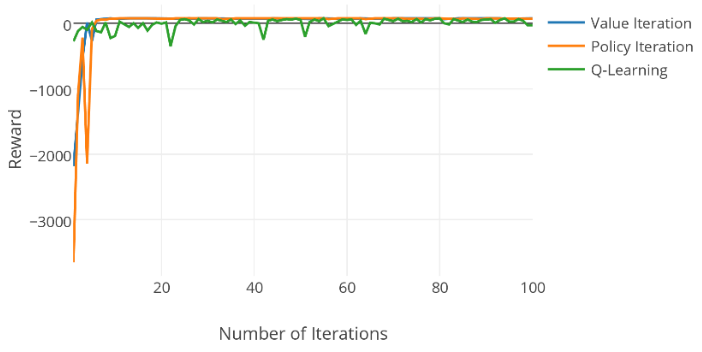


Policy Iteration takes significantly more time to converge as compared to Value Iteration. However, in both the problems, the slowest algorithm is Policy Iteration and the fastest seems to be Q-Learning.

Problem1: Iterations vs Reward Gained for optimal policy

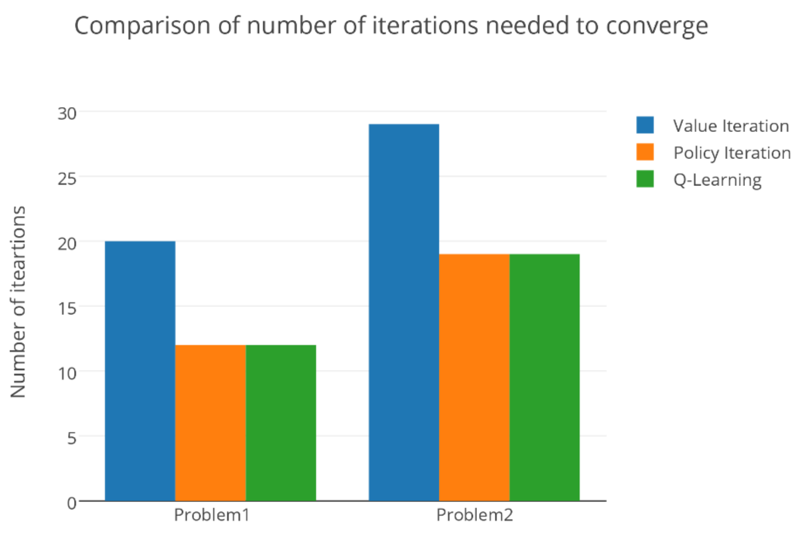


Problem 2: Iterations vs Reward Gained for optimal policy



Initial values of reward are lower for Policy Iteration because a random policy was picked up initially but with further iterations, all the algorithms have more positive rewards.

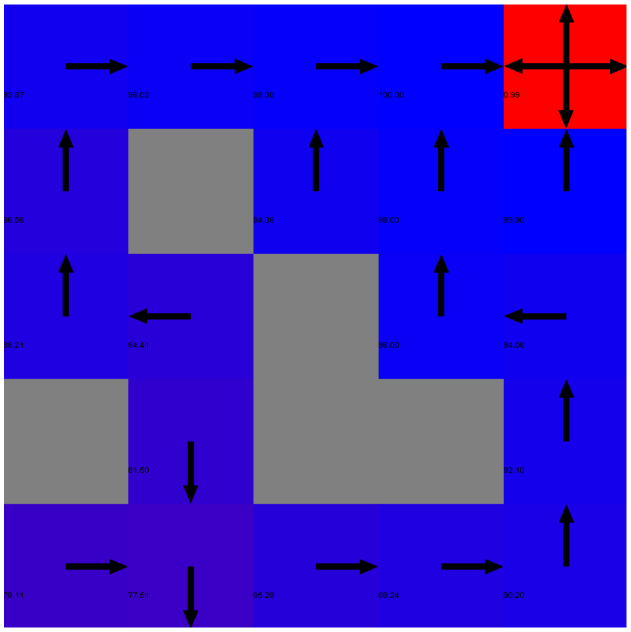
Value Iteration takes more number of iterations to converge as compared to Policy Iteration. It should be noticed that each individual iteration of policy iteration takes longer, because it requires solving the complete system of linear equations because of the two steps of Policy Iteration i.e. Policy Evaluation and Policy Improvement. Hence it takes more time for computation. However, both Value and policy iteration converged to same answer as shown above in grid maps. They both resulted in exactly same optimal policy and with almost similar optimal grid values.



With increase in number of states (from 20 states to 95 states), the iterations needed to converge increased. Also, time taken to generate the optimal policy increased with increase in number of states.

<u>Q-Learning Algorithm:</u>

I chose Q-learning to solve the above two MDPs. It is a model –free reinforcement learning technique, i.e. unlike Policy Iteration and Value Iteration algorithms which know the model before they begin their search, Q-Learning has no concept of the model when it begins learning and it learns how to behave from direct experience with the environment. It learns the Q-values and then take actions based on learned Q-values.

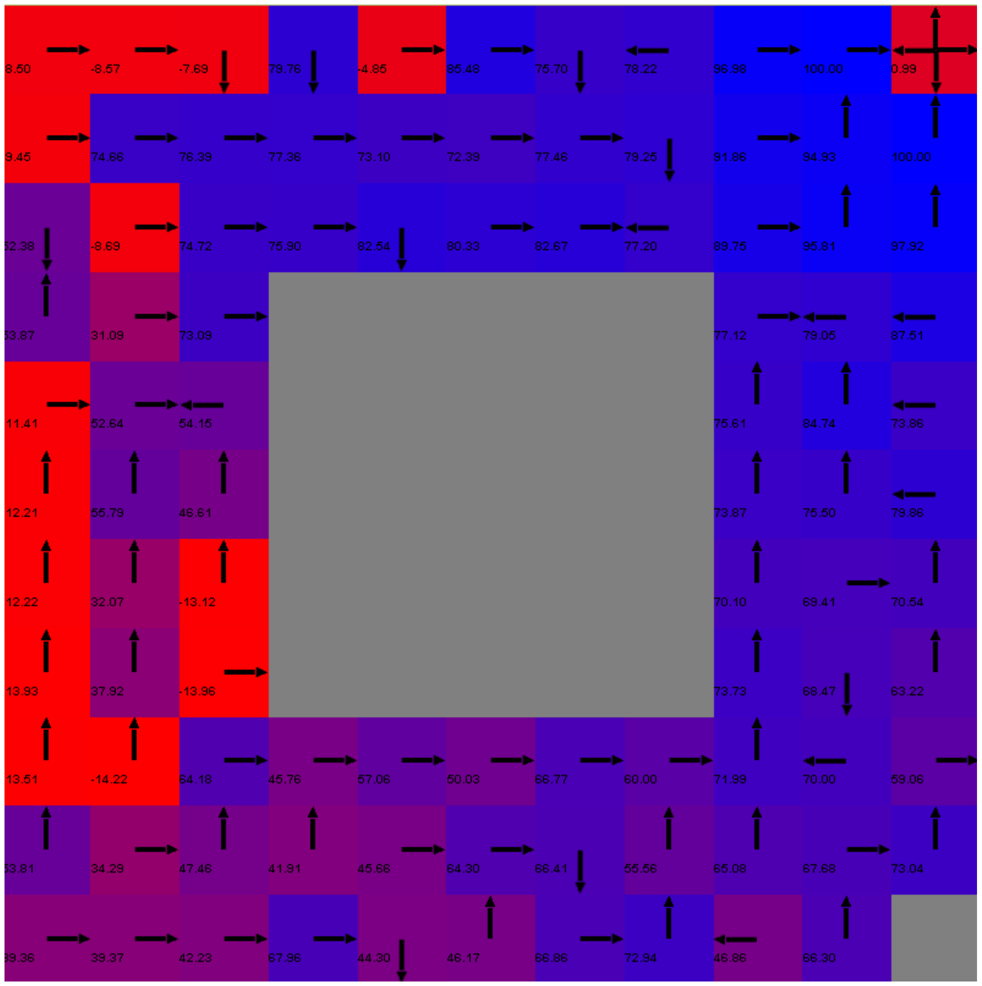


<u>Problem 1:</u>

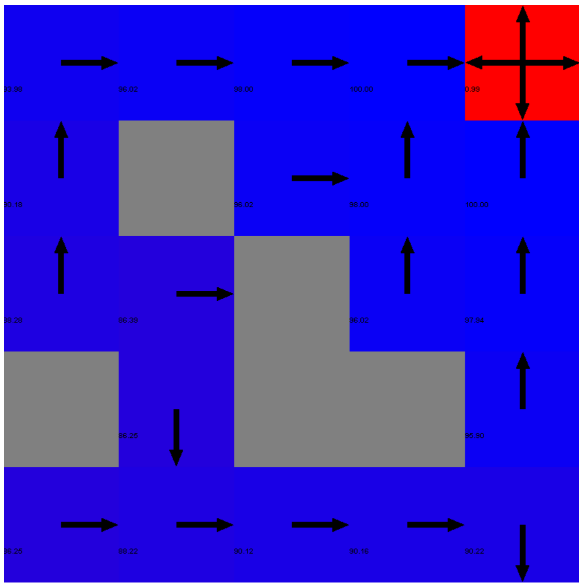
```
Starting reachability analysis
Finished reachability analysis; # states: 20
Passes: 12
Num generated: 260; num unique: 20
```

<u>Problem 2:</u>

```
Starting reachability analysis
Finished reachability analysis; # states: 95
Passes: 19
Num generated: 1500; num unique: 95
```

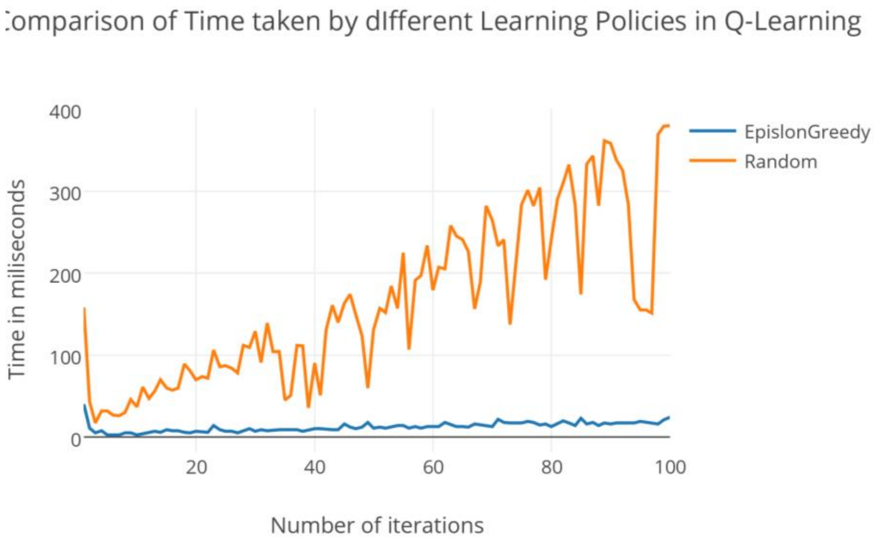


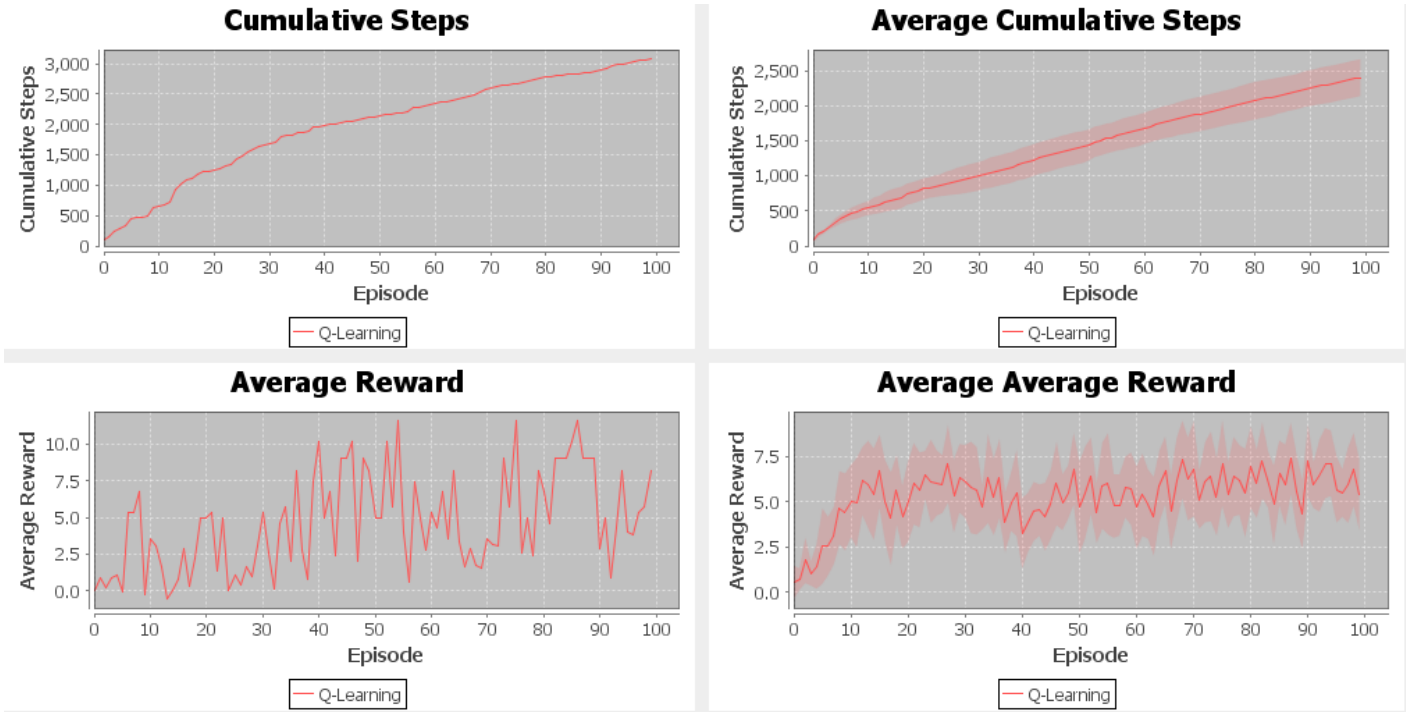
The value function generated by Q-Learning is quite similar to the ones generated by Value Iteration and Policy Iteration. The policy generated by Q-Learning performs comparatively poorly as compared to Value Iteration and Policy Iteration. It can be attributed to the fact that Q-Learning is model free learning methodology and is at a disadvantage as compared to Value Iteration and Policy Iteration because Q-Learning doesn't know about the model, rewards and etc. Q-learning rarely explored some states and was hence not able to assign the optimal policy for them. Q-Learning takes less time as compared to Policy Iteration and Value Iteration to converge.



Results with Random Policy.

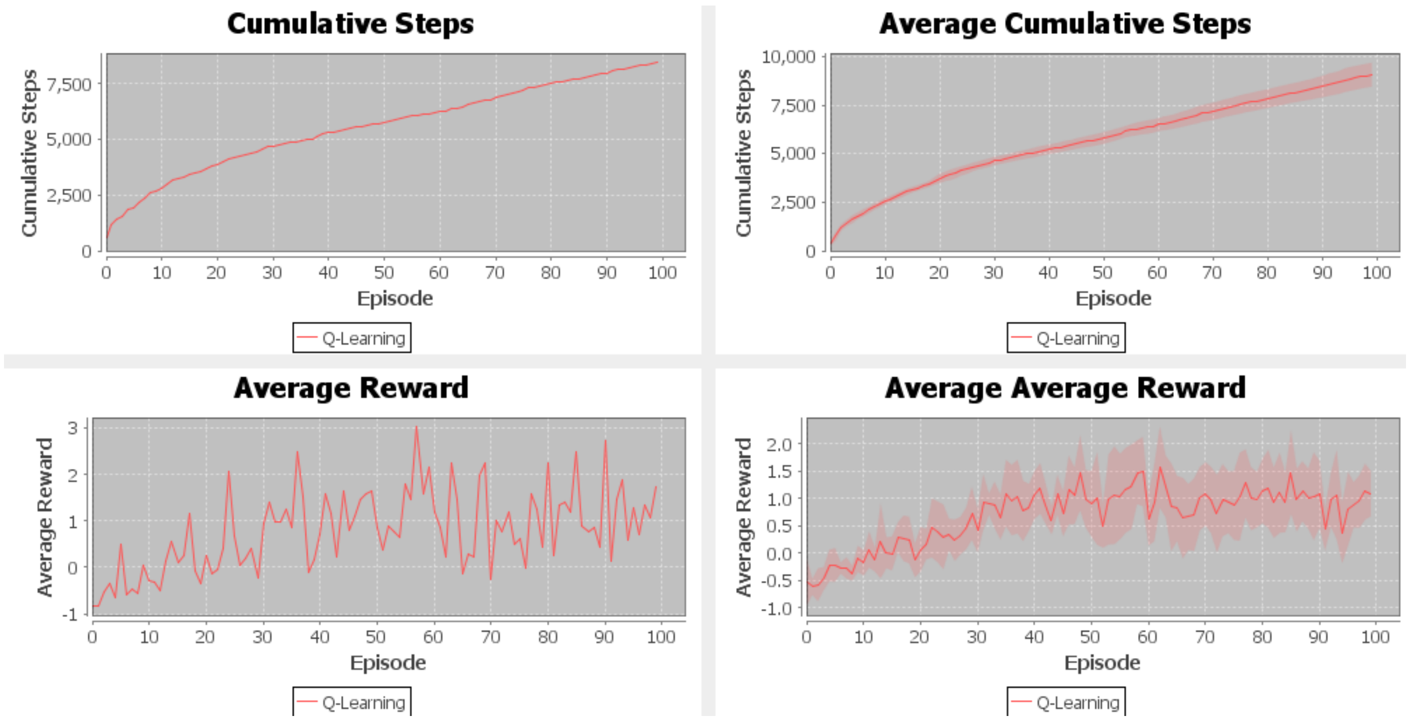
I used the default epsilon-greedy policy which selects random actions with uniform distribution from a set of available actions. Using this policy, either we can select random action with epsilon probability or we can select an action with 1-epsilon probability that gives maximum reward in given state. I also used Random Policy which generated slightly different optimal path, however it took considerably more time as compare to Epsilon Greedy policy





Problem 1: 5X5 Grid

From the graphs above and below, we can see that average reward keeps on increasing as we increase the number of episodes and so does the cumulative steps. Average reward was more with problem with less states. It will take lot more iterations for problem 2 to get better reward.



Problem 2: 11X11 Grid

Conclusion:

From these experiments, it is fair to conclude that RL algorithms do a pretty good job learning and guiding us to select an optimal policy. This can be useful in many real world. Few highlights of the results obtained today are:

- Both Value Iteration and Policy Iteration converged to the same optimal policy for both problems.
- Value Iteration takes less time as compared to policy iteration but Value iteration takes more steps to converge than Policy Iteration.
- Q-Learning performed little poorly as compared to Value Iteration and Policy Iteration because it couldn’t take advantage of know model structure of the domain.