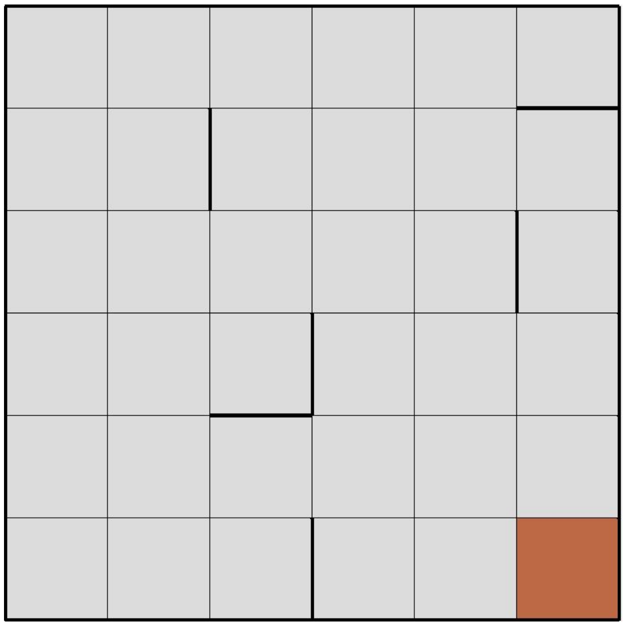
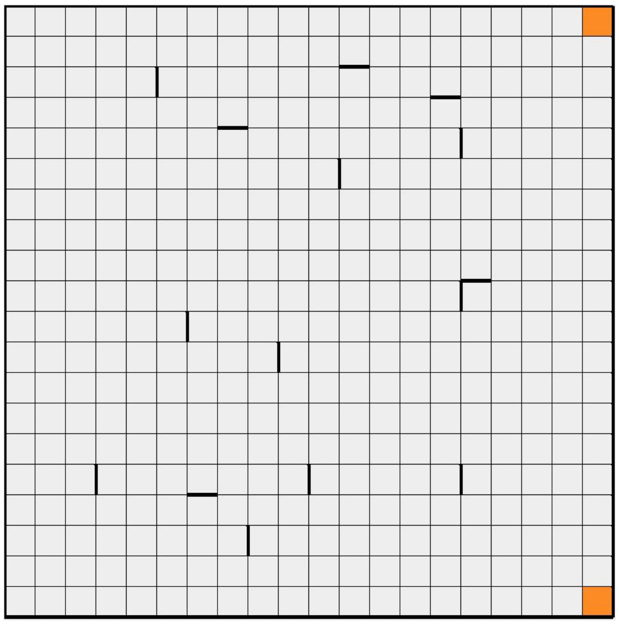
Jeffrey Minowa
CS 4641
Markov Decision Processes

Problems Used

There were two MDP problems that were created to display the differences in performance of the algorithms. Walls were created by having a -50 value when crossing them, and goals were created by giving the algorithm a reward when found. The simple 5x5 grid had fewer walls, 25 different states, and one goal state. This problem was fairly small and a contrast to the larger 20x20 grid with 400 states, many more walls, and multiple goal states. These two are good to juxtapose because they display different samples for the algorithms to iterate. With the smaller grid, both policy and value iteration were quick to converge while avoiding walls. With the larger grid, it was interesting to see how the algorithm made preferences toward the multiple goal states, given the placement of the walls.

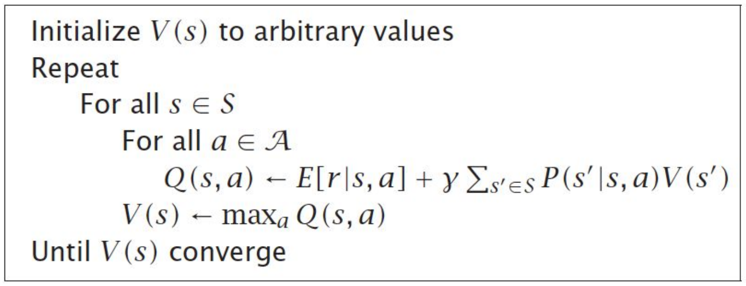




The hyperparameter tested was PJOG which represented the percent chance of going a suboptimal path. This introduces the idea of exploration vs exploitation. Exploration's goal is to find other paths to take in order to find potential optimal, unexplored paths. The goal of exploitation is to use paths already found to hone in on the optimal state by using previous information.

Background on Value Iteration and Policy Iteration

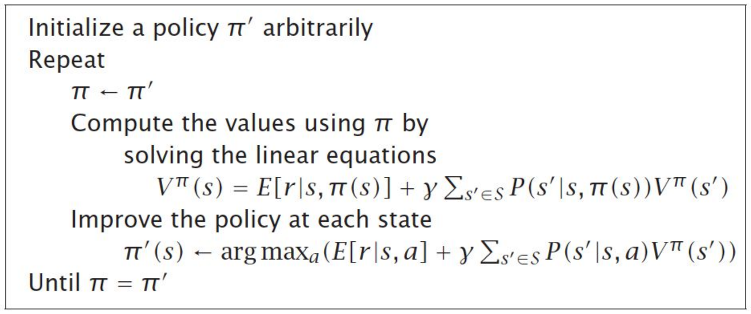
Two similar algorithms that are introduced in this paper are value iteration and policy iteration. Both of the algorithms need to know the transition states prior to actually being run. Each have their pros and cons and can be used to solve Markov Decision Processes (MDPs) which will be discussed later in the paper.
For now, we will define the overarching idea behind each algorithm:
Value iteration, as the name suggests, iteratively recalculates the values of the states until there is convergence of values for each state.



```
Initialize $V(s)$ to arbitrary values
Repeat
    For all $s \in S$
        For all $a \in \mathcal{A}$
            $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$
        $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge
```

Equation from [3]

Policy iteration, on the other hand, computes its policy by solving a set of linear equations from expected reward (first equation) and iteratively checks to see if other policies would improve performance until a policy is guaranteed to be optimal [1].



```
Initialize a policy $\pi'$ arbitrarily
Repeat
    $\pi \leftarrow \pi'$
    Compute the values using $\pi$ by
        solving the linear equations
            $V^{\pi}(s) = E[r|s,\pi(s)] + \gamma \sum_{s' \in S} P(s'|s,\pi(s))V^{\pi}(s')$
    Improve the policy at each state
        $\pi'(s) \leftarrow \arg\max_a (E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V^{\pi}(s'))$
Until $\pi = \pi'$
```

Equation from [3]

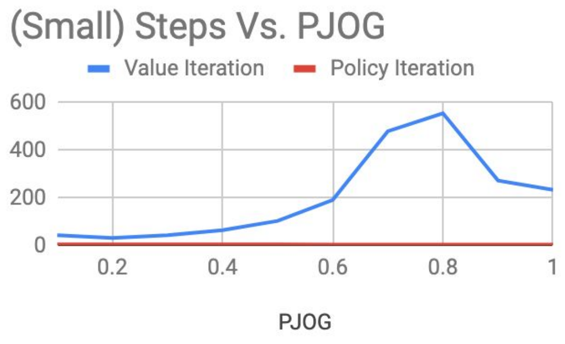
For both algorithms, the expected values from states further from the current state provide diminishing returns the further away it is from the current state. This is done so that other states provide input to the current state's policy while preventing infinite value from the iterations.

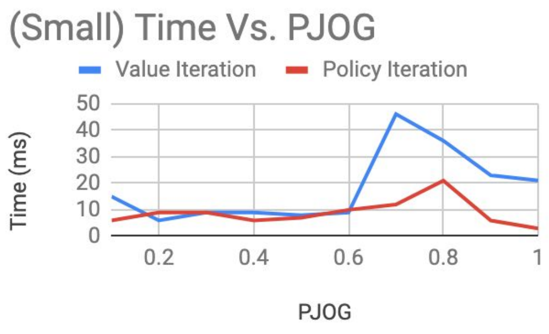
Small Map Comparison

As expected, the number of steps required for the policy iteration is much lower than the value iteration, especially closer to larger PJOG values. As can be seen by the line graphs below, policy iteration consistently has a smaller number of steps needed to converge onto its answer. This is because policy iteration is more efficient in terms of number of iterations needed. However, policy iteration also has the downside of being computationally more expensive per iteration. This difference is caused by the fact that value iteration continually recalculates the value of being in each state until convergence, whereas, policy iteration recalculates based on policy and not value which results in it having a two step process - calculating its linear equations and finding out if a policy needs changing. Clearly, finding the same policy will be faster than recalculation until a specific confidence (in all cases in the paper, the precision value used was .0001) which is why the number of iterations is consistently low.

Being a more intensive process in each step also provides insight to why policy iteration takes longer for lower PJOG values. For most of the PJOG values, policy iteration's time is significantly larger until the PJOG is large. This divergence is likely caused by value iteration's bias toward calculating to convergence of the smallest precision. When manually stepping through the algorithm, value iteration changes only one policy every 10s of iterations once it has come close to convergence. Those changing policies are usually in the states that can go either direction. Because the PJOG peaks at .8, that may be the area that value iteration has the closest values between two or more policies, and this would cause values to oscillate instead of converge.

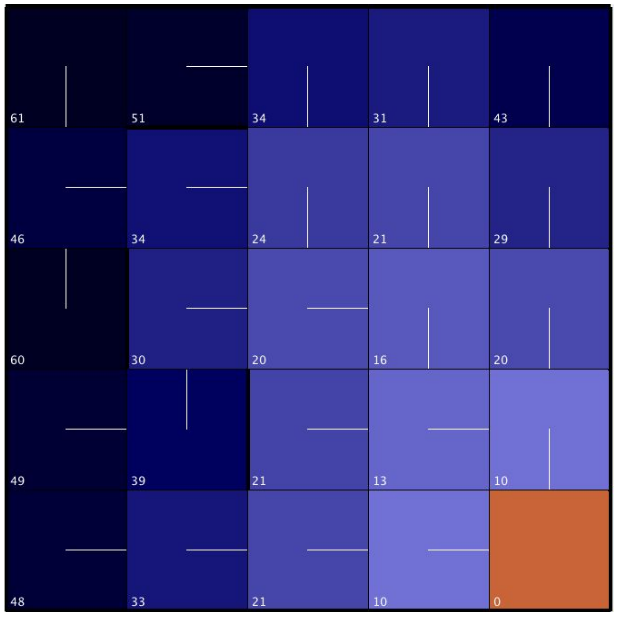
Another noteworthy aspect is that value iteration performs poorly in larger PJOG values because exploring the optimal map and converging is difficult when there is a high chance of randomness. This may be why policy iteration outperforms value iteration on both charts for large PJOG values.

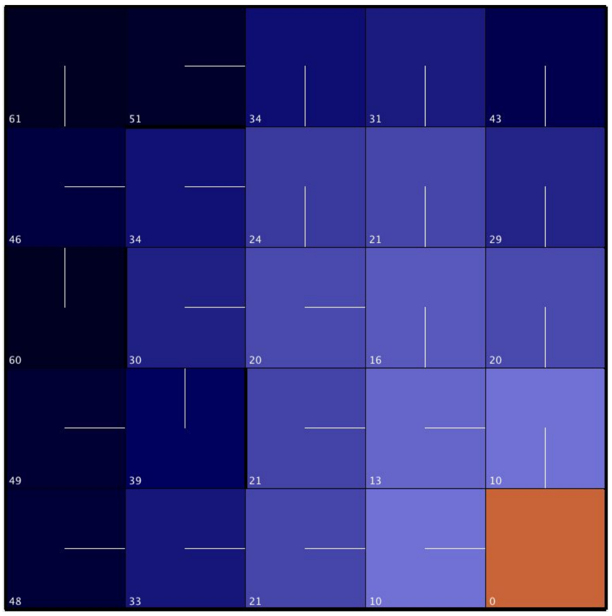




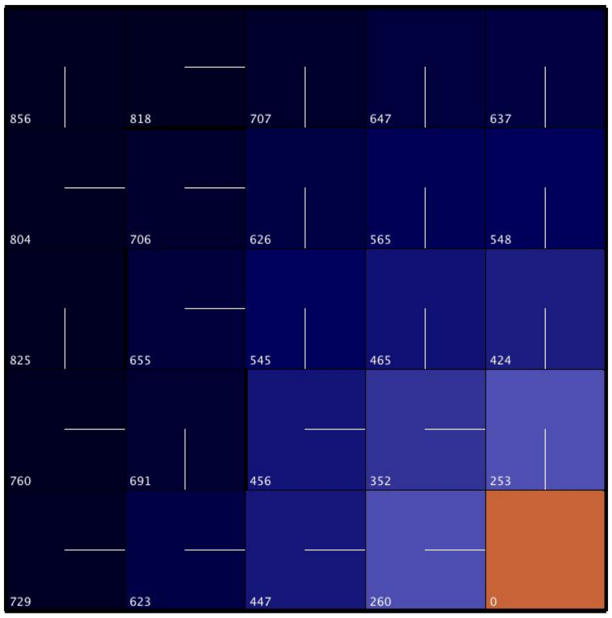
However, for low PJOG values (.3 in the figures below), the policies that both algorithms converge to are exactly the same in the case of the small grids regardless of algorithm.

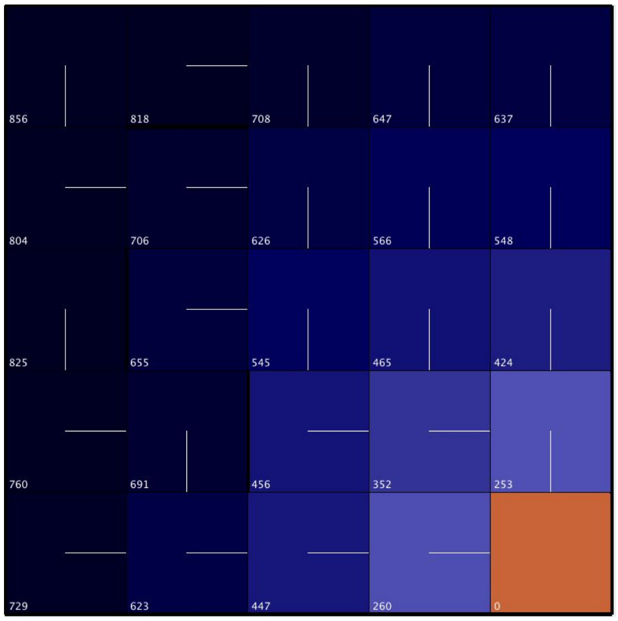
(Left: Value Iteration, Right: Policy Iteration with PJOG = .3)





(Left: Value Iteration, Right: Policy Iteration with PJOG = .7)

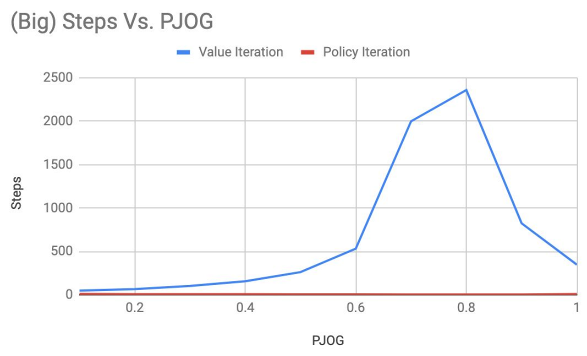


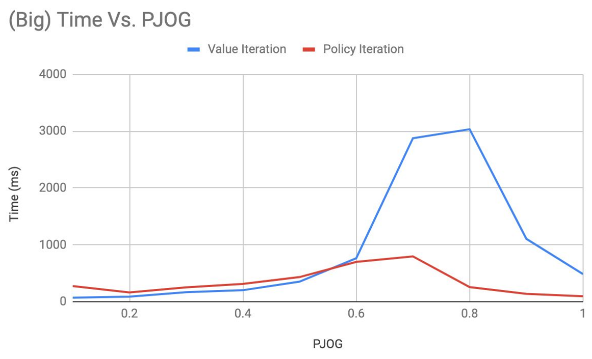


Big Map Comparison

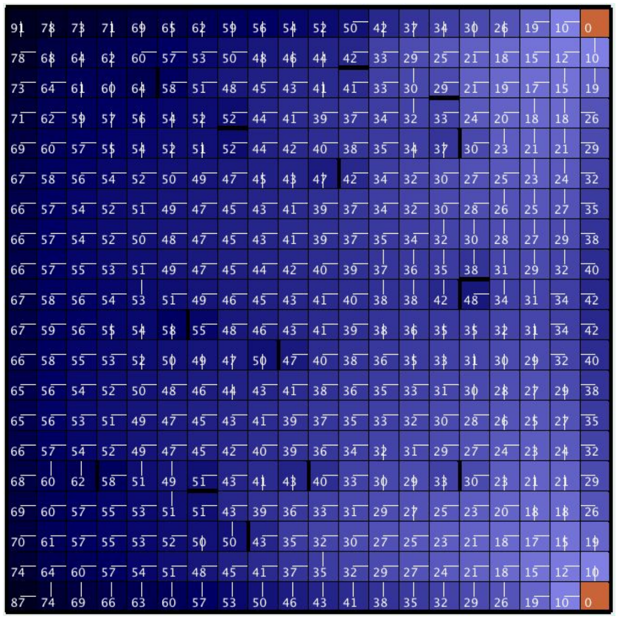
Compared to the small map graphs, the big map's charts for PJOG vs Iterations and PJOG vs Time are similar in shape to the small map's charts. Again, the number of steps for value iteration is magnitudes larger than policy iteration, regardless of the PJOG values. And again, the time the algorithms take to converge cross a little past the .5 mark, displaying that increase in randomness causes value iteration to suffer in performance.

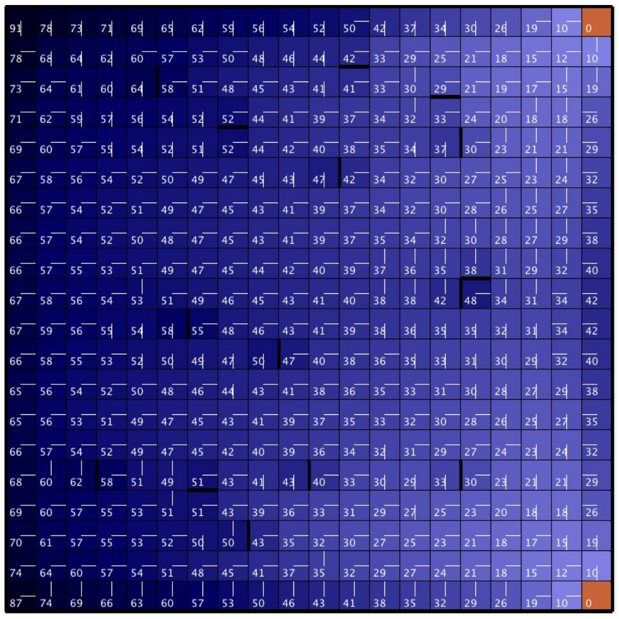
Interestingly, both algorithms have nearly identical policies after convergence even with the two goal states, proving that both find optimal policies to get to the goal states and have a bias toward one goal state over the other. Though it is worth noting that for extremely large grids, policy iteration would take a marginally larger time because as the number of steps increases, so will the time that policy iteration needs to calculate each step[1].





(Left: Value Iteration, Right: Policy Iteration)

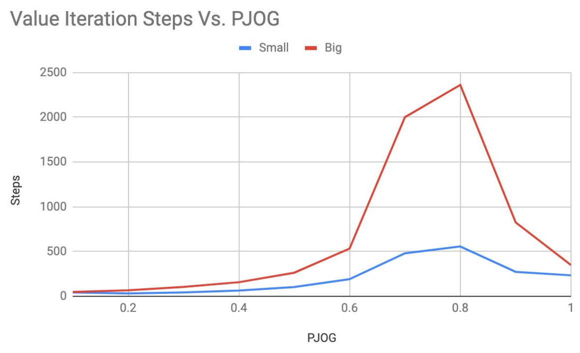


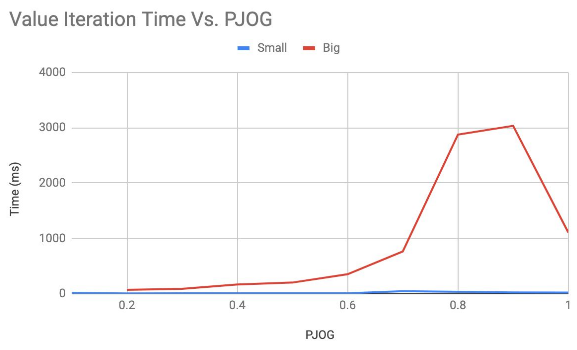


Small vs Big

Value Iteration

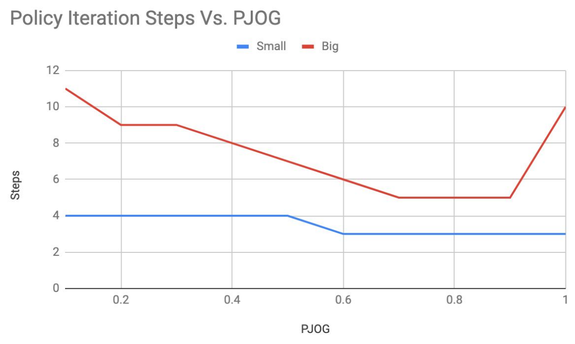
In regards to having a bigger grid for value iteration, the graphs provide a visual explanation of what changes happened. All the peaks in the small graph are exaggerated by the larger grid. Overall, the algorithm acts as expected when given more states; the algorithm does more work to compensate.

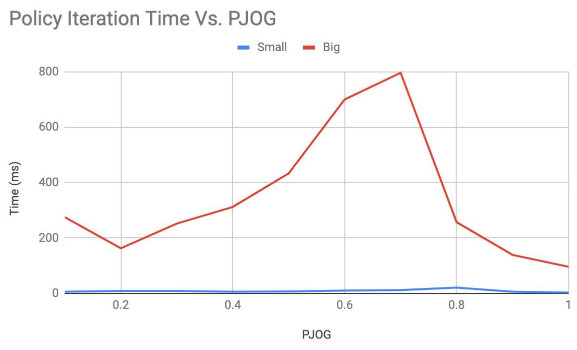




Policy Iteration

Policy iteration acts differently than value iteration in that it doesn't just scale the graph. A possible explanation on the PJOG vs Time graph divergence is that there is a more even split around .5 and .6, forcing the algorithm to take longer to compute its policies because it is unsure whether to go to the top goal or bottom goal. A possible explanation for the PJOG vs Steps graph's divergence around .9 and 1 is that there may be specific policy that oscillates between directions.





Q-Learning

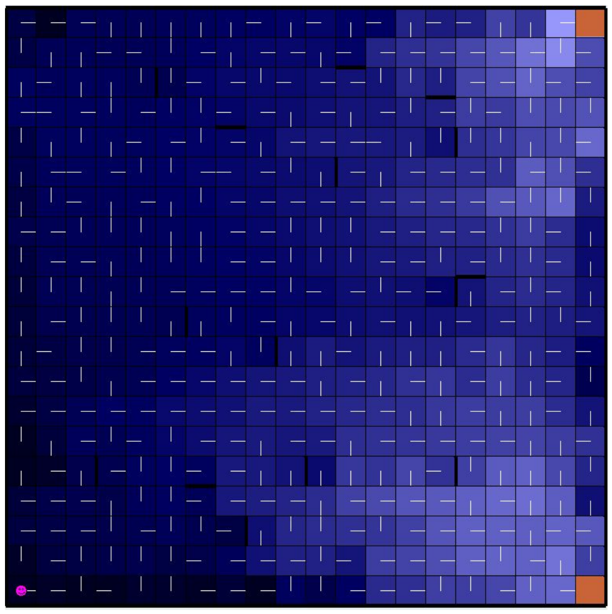
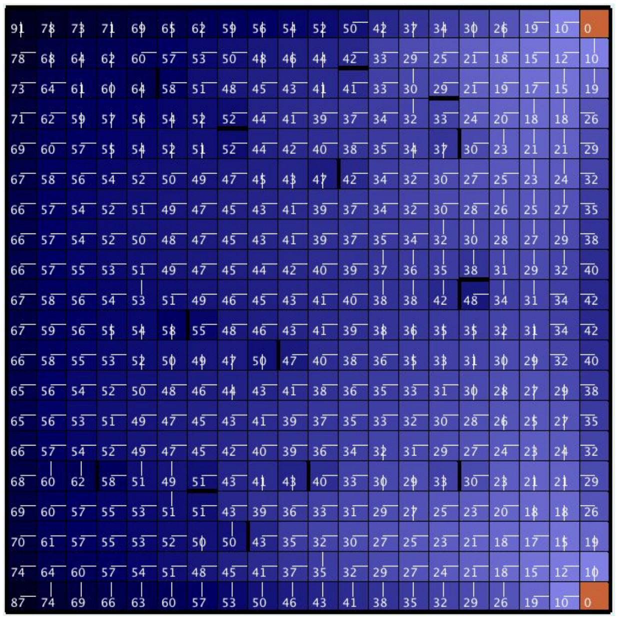
Q-learning is another angle to attack MDPs, but in this case, the transition function is unknown. Q-learning is quite different from the previous algorithms because it only knows information that the algorithm has visited and uses that information to make decisions. This particular Q-Learning algorithm also makes use of a decaying learning rate which is similar to simulated annealing's decreasing temperature. The decaying learning rate enforces the bias that early decisions are important in exploration, then goes toward the goal states closest to it. This is important because, unlike the other two algorithms, Q-Learning does not know its transition states, and computes them by exploring different paths. Thus, there is a significant difference in performance between

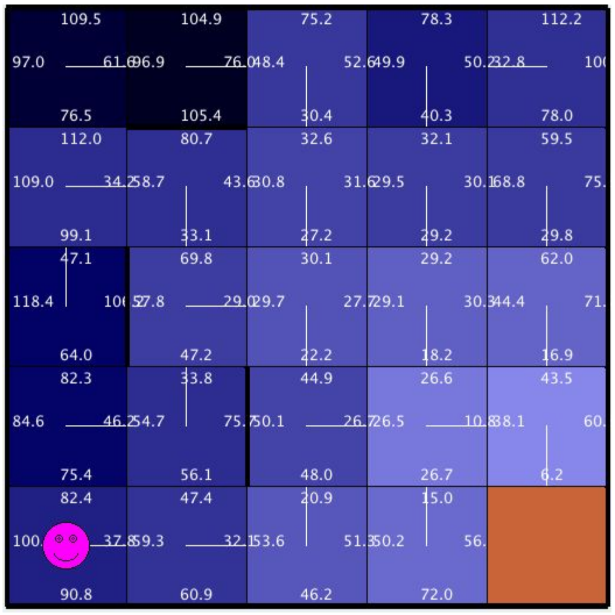
Q-Learning and both Policy Iteration and Value Iteration. Assuming the value iteration has the optimal path, Q-Learning's performance is subpar, and it takes much longer and many more iterations to get to the goal state to learn policies that resemble the optimal. Though this is expected because it is learning as it explores.

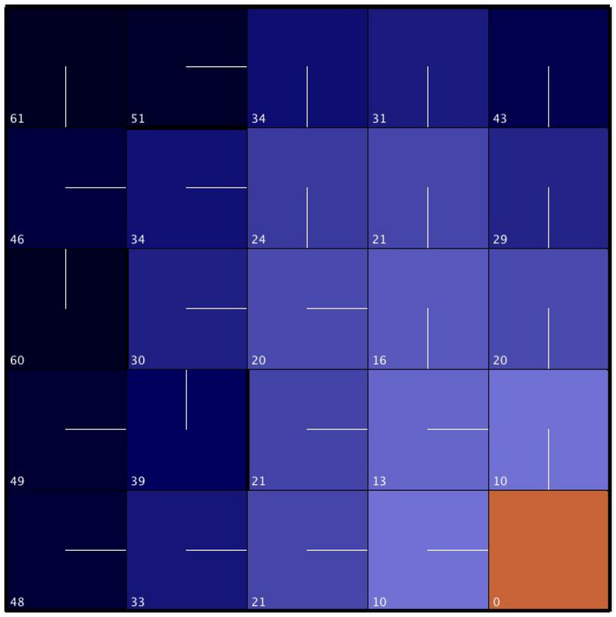
The algorithm readily exploits its paths as can be seen from the top left image; Q-learning takes preference toward the top right goal state over the bottom right because of the early stages where the walls are heavily to the right of the starting point, effectively pushing the learner up instead of right.

Big Map			**Small Map**	
PJOG	0.3		PJOG	0.3
Epsilon	0.1		Epsilon	0.1
Precision	0.001		Precision	0.001
Learning Rate	0.7		Learning Rate	0.7

(Left side: Q Learning, Right side: Value Iteration)





Conclusion

Because the graphs created from the big map and small map are so similar, these examples consistently reflect the benefits and drawbacks of both algorithms. Value iteration is faster, given exploitation is relatively small. Though policy iteration will take less steps, it is computationally more expensive. Policy iteration is also more robust to randomness as it does not need to converge via strict values. When comparing two different test sets, the value iteration performs as expected, i.e. similar graphs with exaggerated curves because of the increased number of states. The policy iteration performs differently, and my reasoning is that the cause is from indecisiveness amongst a few states for the larger grid. Q-learning is a completely different way to solve the MDP when there is no transition state available.

References

[1] “Markov Decision Process.” *Wikipedia*, Wikimedia Foundation, 6 Apr. 2019, en.wikipedia.org/wiki/Markov_decision_process.

[2] Kaelbling , Leslie. “Policy Iteration.” *Policy Iteration*, www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/node20.html.

[3] Alzantot, Moustafa. “Deep Reinforcement Learning Demysitifed (Episode 2) - Policy Iteration, Value Iteration and...” *Medium*, Medium, 9 July 2017, medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa.