Markov Decision Processes

Jennifer S. Johnson

1. Why?

In some sense, we have spent the semester thinking about machine learning techniques for various forms of function approximation. It's now time to think about using what we've learned in order to allow an agent of some kind to act in the world more directly. This assignment asks you to consider the application of some of the techniques we've learned from reinforcement learning to make decisions.

2. The Problems Given to You

2.1. Come up with two interesting MDPs. Explain why they are interesting. They don't need to be overly complicated or directly grounded in a real situation, but it will be worthwhile if your MDPs are inspired by some process you are interested in or are familiar with. It's ok to keep it somewhat simple. For the purposes of this assignment, though, make sure one has a "small" number of states, and the other has a "large" number of states.

2.1.1. Atlanta Snowmageddon 2014 MDP

Two interesting Markov Decision Processes (MDPs) were adapted from the Frozen Lake/Save the Frisbee problem. As a native of Atlanta, I have experienced, first-hand, the effects of winter ice storms on the city, most recently “Snowmageddon 2014”.

Figure 1. Atlanta Snowmageddon 2014[1]. Photo credit: AJC photographer John Spink

Snowmageddon 2014 is the last big winter snow storm to hit Atlanta and bears its name due to traffic gridlock created when commuters, schools, and businesses all let out at the same time wreaking havoc on roads and interstates stranding commuters overnight in their vehicles. An already congested traffic situation was compounded by hazardous road conditions brought on by black ice which worsened as the temperature fell below freezing.

[1] https://www.wsbtv.com/news/local/atlanta/remember-the-last-big-winter-storm-to-hit-atlanta/481477110

The Snowmageddon 2014 debacle is rife with opportunities to employ MDP to identify optimal actions in navigating the various states of this icy environment. Below is a short list of MDPs for the Snowmageddon 2014 example:

- Coordinated school and work release decision process
- School superintendent's early school closure due to weather
- Department of Transportation's decisions about road treatment and timetable in advance of snow/icy conditions

Keeping the MDPs somewhat simple, the selected MDP for this assignment relates to the latter – **the safest roads to pass during icy conditions**. For this MDP, the commuter (agent) must optimally learn the safest route home (goal) based on its own experience. While the roads in some areas of the city may be clear and passable (unknown initially to the agent), there are patches of black ice randomly located between the agent's parked car (starting point) and the goal. Given the icy conditions of the road, if the agent encounters patches of ice, there is the potential to crash into other cars on the road, pedestrians, or any other object proximal to the agent's vehicle. As is the case with icy surfaces, the agent's car does not always move in the direction intended because of the slipperiness of the road conditions. For this reason (and the fact that I was a victim of Snowmageddon 2014), this is an interesting MDP because the randomness of the real-world hazards of navigating snow and ice can be closely replicated using stochastic methods. This particular MDP is designed such that the episode ends when the agent safely reaches the goal or careens into another object on or near the road. The agent receives a reward of 1 if goal is reached and a zero otherwise.

Two MDPs are identified for this Snowmageddon 2014 example. The first MDP is a 4x4 grid containing 16 states. This MDP will represent the one with a "small" number of states. The second, larger MDP is an 8x8 grid containing 64 states. Figure 2 below illustrates both of the Atlanta Snowmageddon 2014 MDPs which will be used for this assignment.

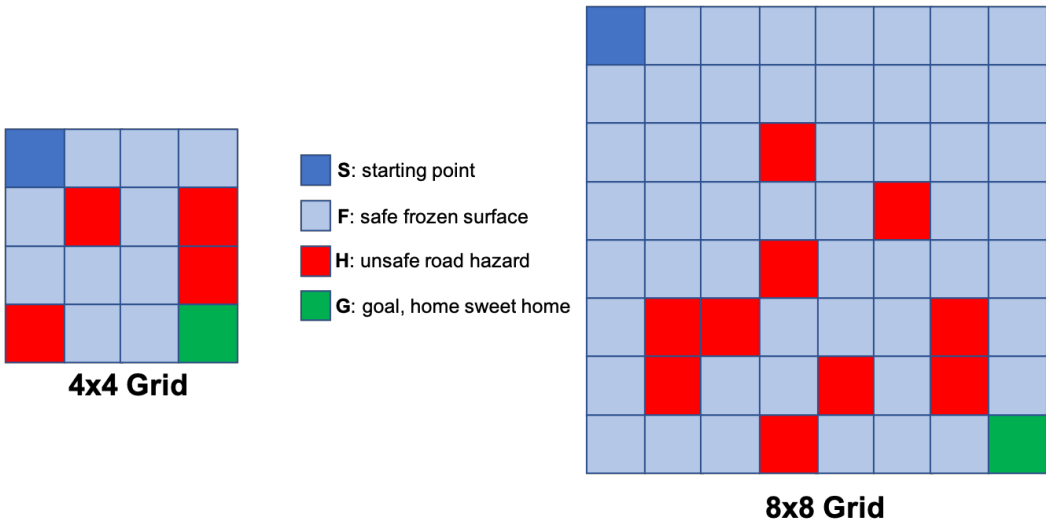


Figure 2. Two Atlanta Snowmageddon 2014 MDPs (4x4 and 8x8 grids)

2.2. Solve each MDP using value iteration as well as policy iteration. How many iterations does it take to converge? Which one converges faster? Why? Do they converge to the same answer? How did the number of states affect things, if at all?

2.2.1. Value Iteration for Atlanta Snowmageddon 2014 MDPs

For this assignment, the value iteration algorithm begins by initializing the value function and setting the discount factor equal to 1. Looping through the maximum iterations (max_iterations = 100,000), the difference in sum of values between iterations and the sum of values per state are calculated and stored, breaking the loop only when the value iteration has converged. Provided below (in Figures 3 through 6) are side-by-side performance plots for both MDPs – 4x4 (left) and 8x8 (right).

As seen below, there is very little difference in the first of the two metrics graphed for both small and large MDPs. In Figure 5, it is interesting (and potentially uneventful) to note the time spikes at iteration counts that are nearly half of the value at which the iteration converges. (The peak times each highlighted by a red box.)

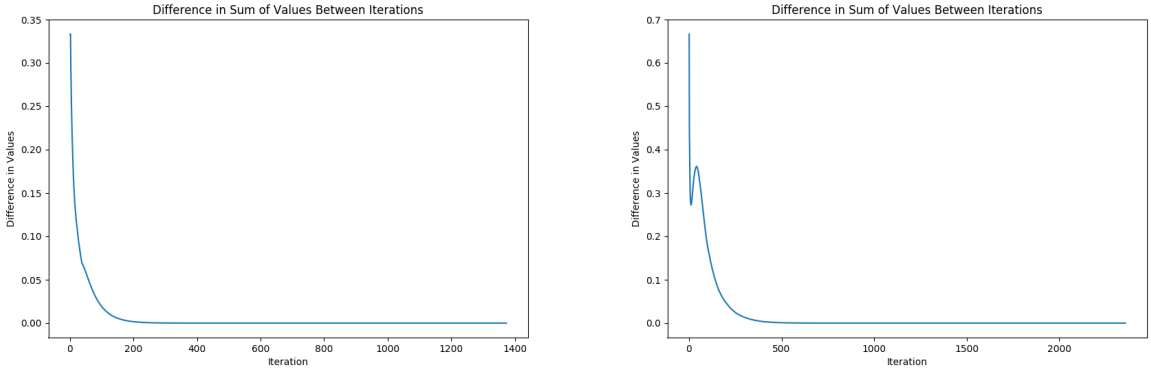


Figure 3. Difference in Sum of Values Between Iterations (4x4 and 8x8 grids respectively)

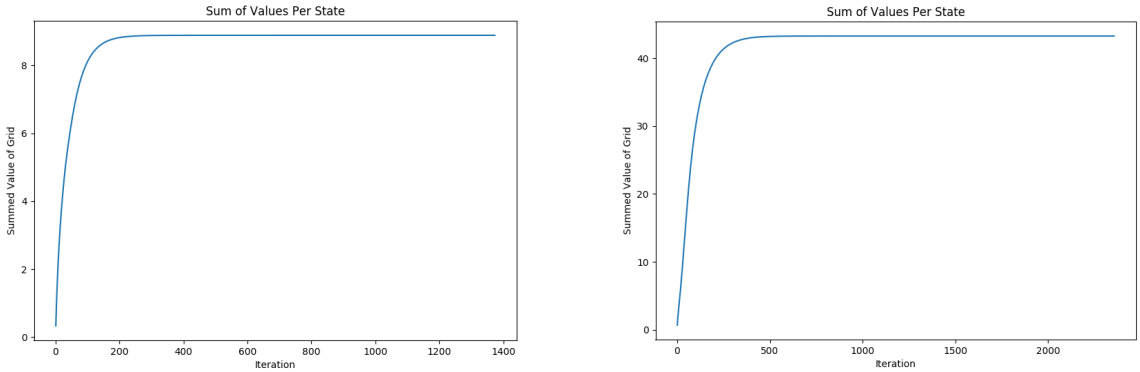


Figure 4. Sum of Values Per State (4x4 and 8x8 grids respectively)

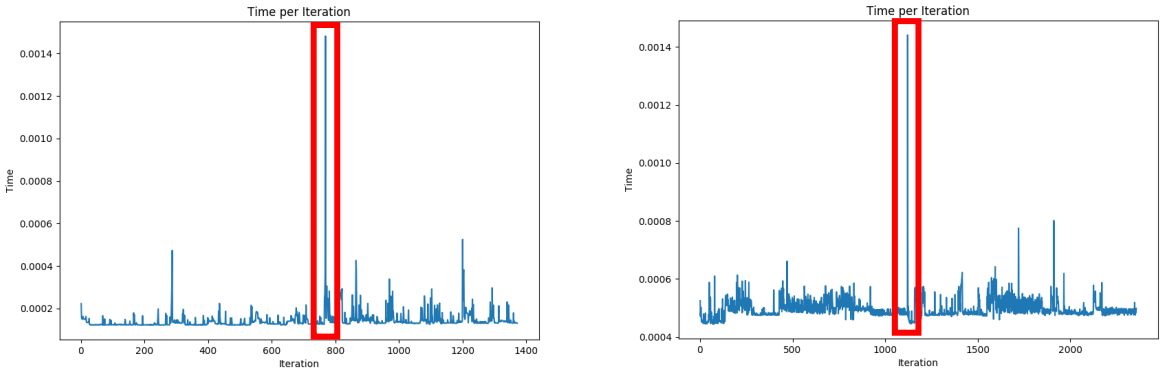


Figure 5. Time Per Iteration (4x4 and 8x8 grids respectively)

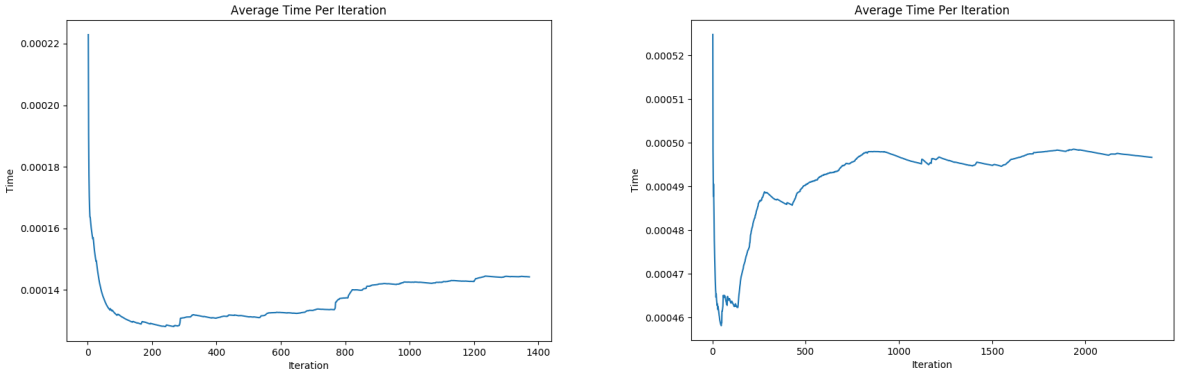


Figure 6. Average Time Per Iteration (4x4 and 8x8 grids respectively)

The average time per iteration plots (above in Figure 6) show similar behaviors for both the small and large MDPs. An initially high average time decays with increasing iterations; although, for the large MDP, the average values creep back up and stabilize after about 1500 iterations.

The value iterations converge at iterations 1373 and 2357 for the 4x4 and 8x8 grids, respectively. Intuitively, this makes sense given that the 8x8 grid is twice as large as the 4x4 grid. The ratio of the number of iterations to convergence between the two MDPs is nearly double (1:1.72). In the context of the Snowmageddon 2014 MDPs, this is the equivalent of the commuter (agent) attempting to learn its way home (goal) optimally while traversing unknown area double the size of the original commute. The **large MDP performs this task in efficient time** despite the fact that it has four times as many states as the small MDP.

The average policy scores are 0.819 and 1.0 for the 4x4 and 8x8 grids, respectively. Again, this makes sense since the larger grid provides more opportunities to gain a higher policy score. The large MDP is able to achieve a higher, more optimal policy score attributable to the grid design. In the small MDP, four of 16 or a quarter of the states are unsafe/pose the threat of death while, for the large MDP, 10 of 64 or approximately 16% of states pose the same threat. In other words, it is **more likely to find an optimal route in the large MDP given its structure**.

Comparing the total time for value iteration to converge for both grids, **the larger 8x8 grid takes three times as long as the 4x4 grid to converge**. This difference can be accounted for due to size and the resulting increase in computation required. It will be interesting to compare these times with the policy iteration time to convergence. I expect that policy iteration will require fewer steps/iterations, but will be slower than value iteration due to the computational complexity of policy iteration.

2.2.2. Policy Iteration for Atlanta Snowmageddon 2014 MDPs

The policy iteration algorithm begins by initializing a random policy and setting the discount factor equal to 1. Looping through the maximum iterations (max_iterations = 200,000), the number of value function iterations is tracked, breaking the loop when the policy iteration is optimal (i.e., converges). Provided below (in Figures 7 through 10) are side-by-side performance plots for both MDPs – 4x4 (left) and 8x8 (right).

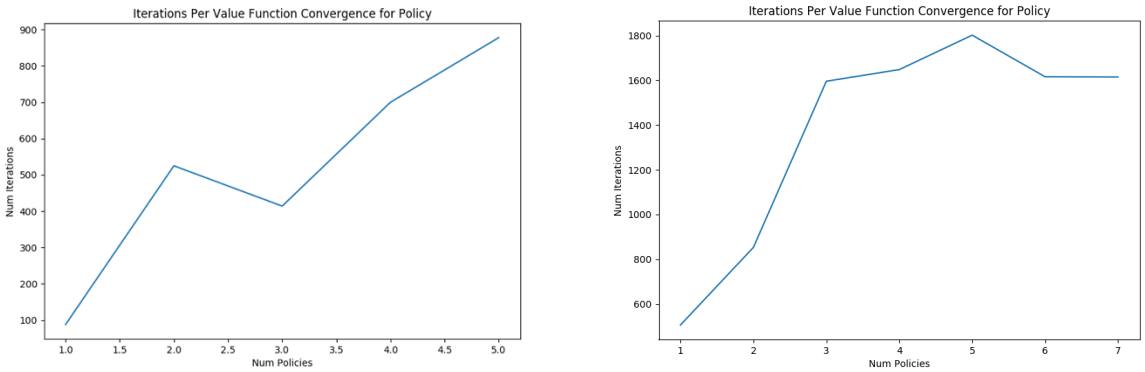


Figure 7. Iterations Per Value Function Convergence for Policy (4x4 and 8x8 grids respectively)

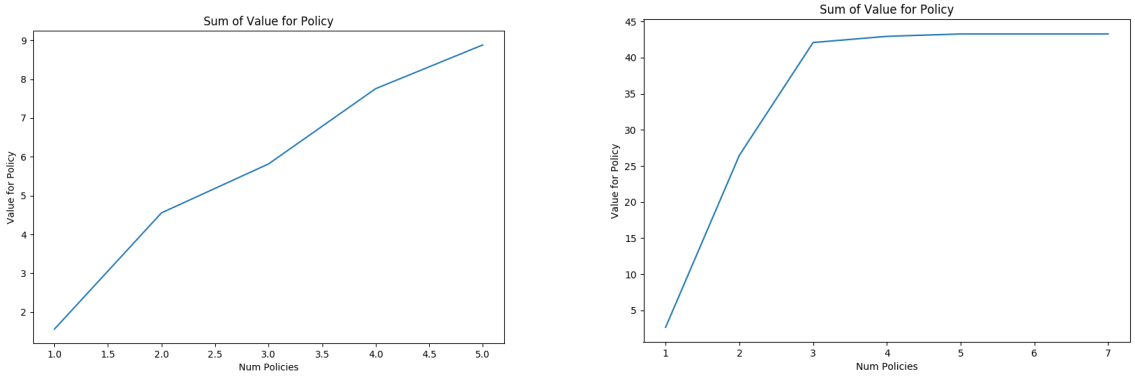


Figure 8. Sum of Value for Policy (4x4 and 8x8 grids respectively)

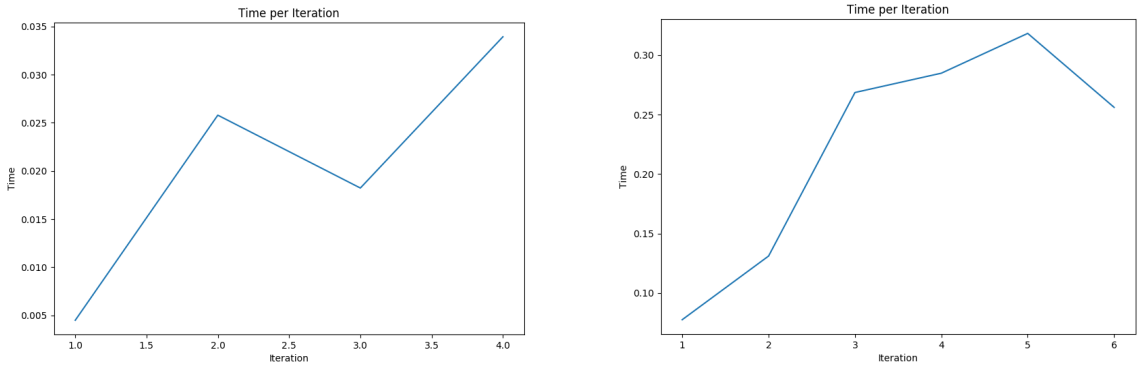


Figure 9. Time Per Iteration (4x4 and 8x8 grids respectively)

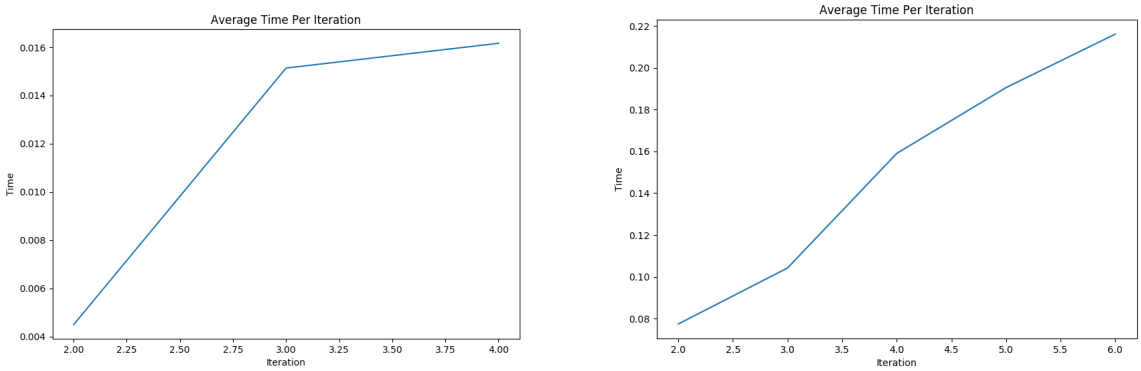


Figure 10. Average Time Per Iteration (4x4 and 8x8 grids respectively)

The policy iterations converge at steps 5 and 12 for the 4x4 and 8x8 grids, respectively. Intuitively, this makes sense given that the 8x8 grid is twice as large as the 4x4 grid, and therefore has a greater area to traverse.

The average policy scores are 0.77 and 1.0 for the 4x4 and 8x8 grids, respectively. Again, this makes sense since the larger grid provides more opportunities to gain a higher policy score due to its design. I expect that **the addition of more unsafe road hazards to the large MDP would result in a lower average policy score than that obtained above**.

Comparing the total time for value iteration to converge for both grids, the larger 8x8 grid (3.012 seconds) takes six times as long as the 4x4 grid (0.5191 seconds) to converge. This difference can be accounted for due to size and the resulting increase in computation required. **The policy iteration performs fewer steps/iterations but is slower than value iteration due to the computational complexity of policy iteration**.

Value iteration and policy iteration produce average policy scores which converge to the same answer in the large 8x8 MDP and nears convergence for the smaller 4x4 MDP.

2.3. Now pick your favorite reinforcement learning algorithm and use it to solve the two MDPs. How does it perform, especially in comparison to the cases above where you knew the model, rewards, and so on? What exploration strategies did you choose? Did some work better than others?

The Q reinforcement algorithm is selected; the exploration parameters used are summarized below.

Table 1. Exploration Parameters for Q Reinforcement Algorithm

Parameters	Value
Total episodes	100,000
Learning rate	0.8
Maximum steps	99
Discount factor (gamma)	0.95
Exploration rate	1.0
Exploration probability at start	1.0
Minimum exploration probability	0.01
Exponential decay rate for exploration probability	0.005

Provided below are the performance plots for the Q reinforcement algorithm. The small and large MDPs have similar performance across episodes with exception to the average reward per episode. In the previous value and policy iterations, the average policy scores for the large MDP were at or near 1; in contrast to Figure 11, the average reward per episode is zero. (This problem only seems to affect the large MDP, but not the small MDP.) This inconsistent result could be a sign of the agent not having explored enough and the possibility of being stuck in a local minimum. It could also mean that Q was not properly initialized (i.e., for every state, the initial value "looks awesome and all other actions look terrible"[2]). To get "unstuck", further analysis could attempt some form of random restart/action or revisit the initialization of Q.

One additional note is the time per episode for the large MDP is twice that of the small MDP as shown in Figure 14.

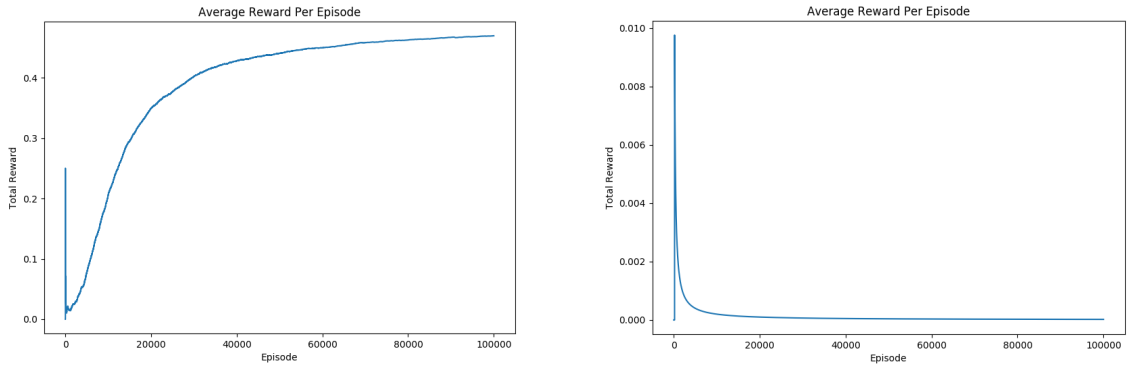


Figure 11. Average Reward Per Episode (4x4 and 8x8 grids respectively)

[2] CS 7641 Q-Learning Choosing Actions lecture – M. Littman, Ph.D. and C. Isbell, Ph.D.

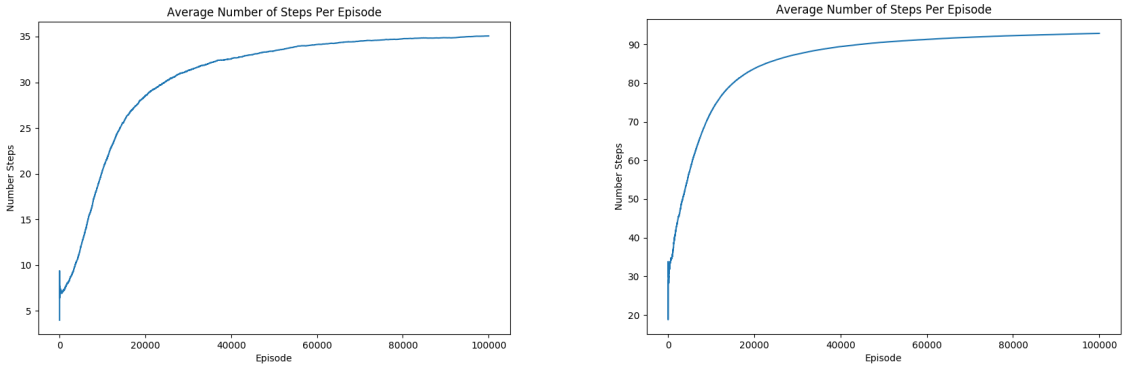


Figure 12. Average Number of Steps Per Episode (4x4 and 8x8 grids respectively)

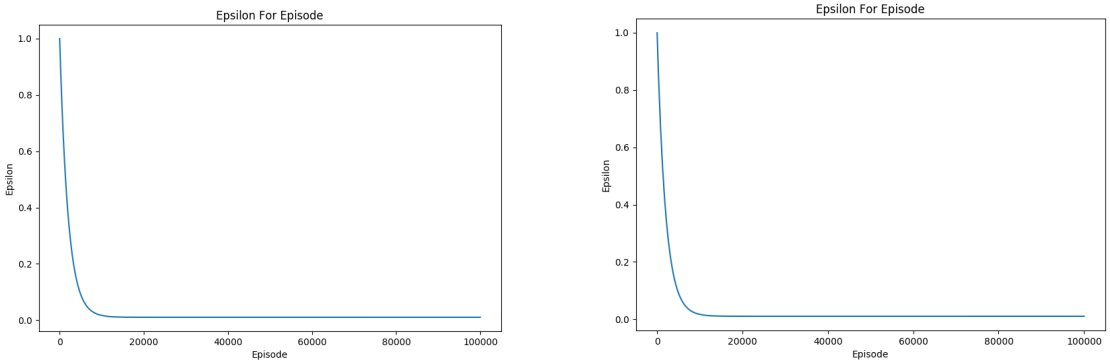


Figure 13. Epsilon Per Episode (4x4 and 8x8 grids respectively)

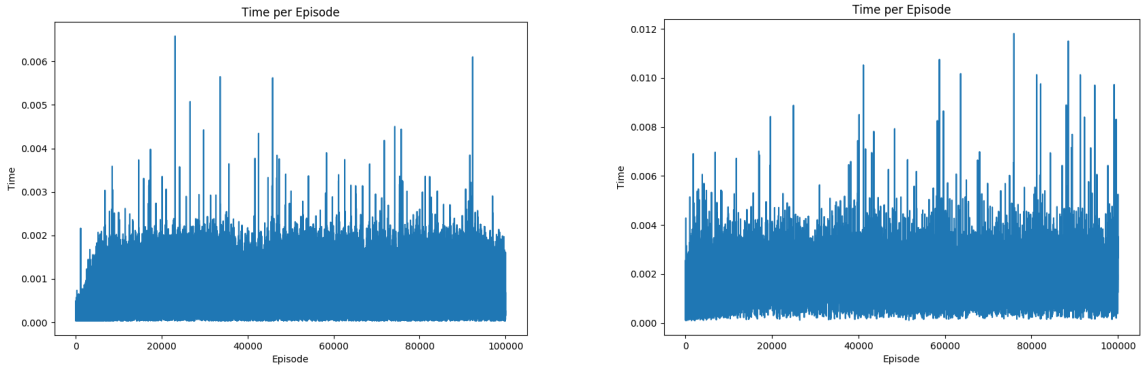


Figure 14. Time Per Episode (4x4 and 8x8 grids respectively)

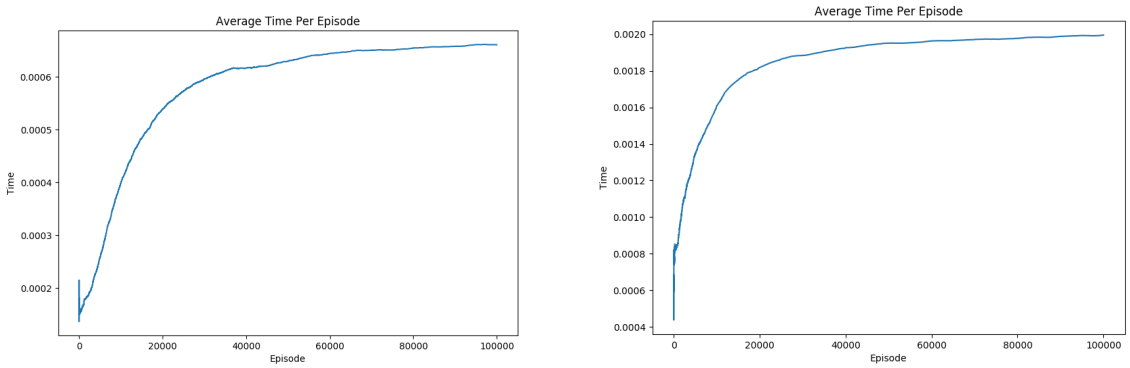


Figure 15. Average Time Per Episode (4x4 and 8x8 grids respectively)

Summarized below in Table 2 is a comparison of score and total time for the Q-learning algorithm.

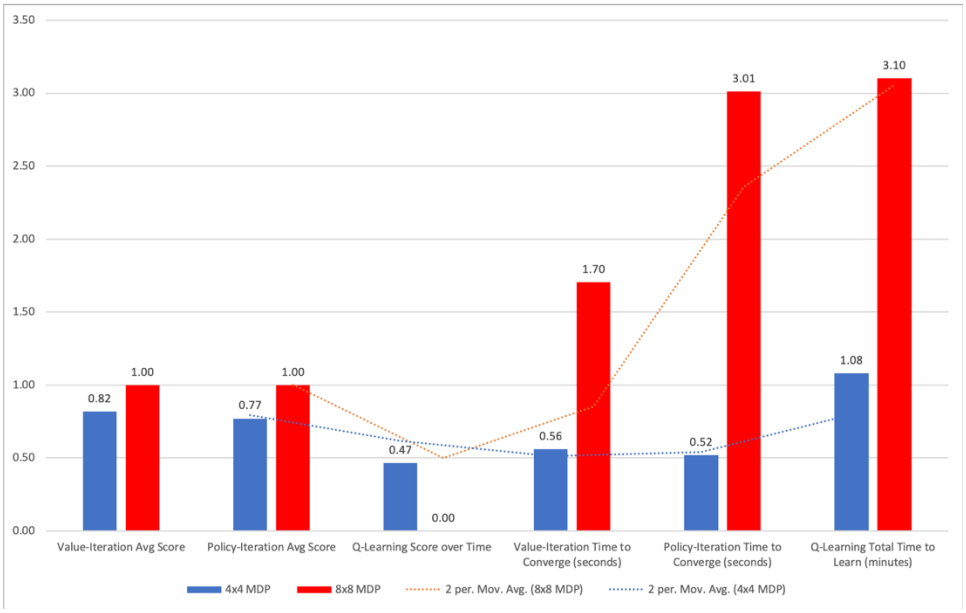
Table 2. Q-Learning Algorithm Performance Comparison

	Basic 4x4 MDP	**Modified 4x4 MDP**	**Basic 8x8 MDP**	**Modified 8x8 MDP**
Total iterations	15,000	100,000	15,000	100,000
Score over time	0.4809	**0.4654**	0.0	0.0
Total time (seconds)	10.83	**64.75**	26.29	**186.11**

The basic Q-learning algorithms were run using 15,000 iterations and the modified cases were run using 100,000 iterations. That modification resulted in a slight decrease in the score over time for the 100k-iterations 4x4 MDP and produced a total time for Q-learning that is six times slower than the MDP run with fewer iterations. Similarly, the total time for Q-learning is over ten times slower for the 100k-iterations 8x8 MDP. There was no observed change in the score over time for the large MDP for the reasons discussed above.

3. Conclusion

Reinforcement learning enables agents to determine the optimal behavior within a given context in order to maximize performance. Provided below in Figure 16 is a comparison of how value iteration, policy iteration and Q-learning algorithm perform against score and run time.



* **NOTE**: The Q-learning total time is given in minutes (not seconds)

Figure 16. Performance of Reinforcement Learning Algorithms

Value and policy iteration converge to (nearly) the same average score for both MDPs although their times vary by 3-4 factors depending on the number of states in the MDP. Q-learning performs better on the MDP with fewer states and appears to get stuck on a local minimum given its low learning score and much longer run time (over three minutes).