CS7641 Assignment 4: Markov Decision Process

Steel Ferguson
Sferguson42@gatech.edu

Introduction

In this assignment I explore two Markov Decision Process (MDP) problems at varying degrees of complexity. I analyze Value Iteration (VI) and Policy Iteration (PI) as model-based approaches to solving the problems along with Q-Learning (QL) which is a reinforcement learning approach that does not know the transition matrix.

Interesting Problems

I created both problems for this assignment, receiving my inspiration from the world-wide pandemic we are all a part of as well as the structure of existing problems.

Stir Crazy: My first problem is not grid based. It takes a structure similar to the Forest Management problem (having two choices and a progressing time component) but deals with the real issue we know we all experience during quarantine.
The agent begins on T=1 and T will increase by 1 regardless of the agent's choice. Each day the agent has the choice to 0: stay inside so as to not risk catching COVID-19 or to 1: go out to enjoy being outside of the house. If the agent stays in, he gets -2 rewards for going stir crazy (thus the name). If the agent goes out, he gets 5 rewards for the relief of not staying at home. However, if the agent goes out there is a chance that he/she will catch COVID-19. If the agent goes the entire game without catching COVID-19, the agent receives 100 points with a big sigh of relief. If the agent caught COVID-19 at all during the test, the agent receives no points for finishing.
The probability of catching COVID if the agent goes out increases through the game from 0 to 50%, so it the optimal policy at most sizes includes risking going out in the first part but then staying in as the game progresses.
The stochasticity in this problem is introduced with the chance of getting COVID-19. It differs from the next problem in a meaningful way in that the probability changes over "time".
Stir Crazy- Small, Medium, Large: For this analysis I have created games with 7, 52, and 365 time units, each following the same point structure and probability mentioned above (stay in: -2; go out: 5; entire game with no COVID-19: 100; probability of getting COVID-19 increasing from 0 to 50% to the end of the game linearly with time). There are n+1 different states where n is the number of time units. The agent is either in stage 1,2,3…n without ever having had COVID-19 or in n+1 if the agent has gotten COVID-19. If they are in this final state, their reward structure remains constant.
Rewards for the game are found by starting the agent in T=1 (first state) and giving the agent n turns (7, 52, or 365). I created my own rewards function to accomplish this.

C19 Grid: The second problem I used is similar to the grid world problem described in class with a starting point and a red and green end point. If the agent lands on the red square -1 point is given; if the agent lands on the green square 1 point is given. For each move in white space, the agent receives -0.2 points to motivate him/her to find an end state quickly.
The difference with this problem is also inspired by the pandemic. Instead of having all white cells and walls, I have introduced COVID-19+ squares. If the agent lands in the COVID-19

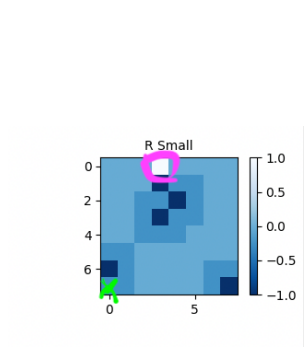
squares he/she will get -1 points. Also, since we know that COVID-19 is spread without contact, especially when a person is within 6 feet, squares surrounding the COVID-19 squares may also give the agent the virus. Squares adjacent to the COVID-19 squares give a -0.25 reward because of this.

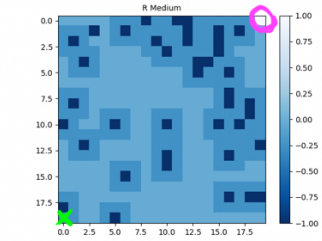
At each state the agent choosing a direction. To make things interesting, when the agent chooses a direction, there is a 10% chance that the agent will go either of the directions adjacent to the chosen one and an 80% chance of going the chosen direction.

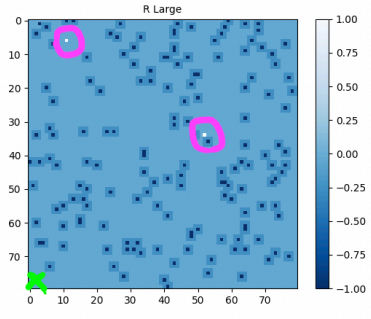
To assess rewards for this game, the agent begins in the bottom left corner and must hit a “green” or “red” exit point. Once that occurs, the game is over. I created a reward function to accomplish this.

C19 Grid- Small, Medium, Large: In the figures below, I show the small grid (8x8), medium gird (20x20), and large grid (80x80). Points are color coded with dark blue being negative and white being positive. Dark dots with a medium-dark ring around them are COVID-19 squares. I have circled the positive exit squares in pink (two positive exits in the largest game to be nice to the agent) and put an X at the starting point.

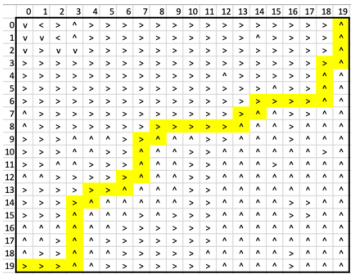
I have also shown the optimal strategy. I have highlighted the optimal path; however, all the points are part of the policy as the agent could choose to follow the yellow arrow road but end up off of the road due to the randomness (20% chance of going a different way). This stochasticity is constant but affects the game differently in different states.







	0	1	2	3	4	5	6	7
0	>	>	>	^	<	<	<	<
1	^	^	^	^	^	^	^	^
2	^	>	^	<	^	^	^	^
3	^	^	^	<	>	^	^	^
4	^	^	^	v	>	>	^	^
5	^	^	^	>	>	^	^	^
6	^	>	>	>	^	^	^	^
7	>	>	>	>	^	^	<	^



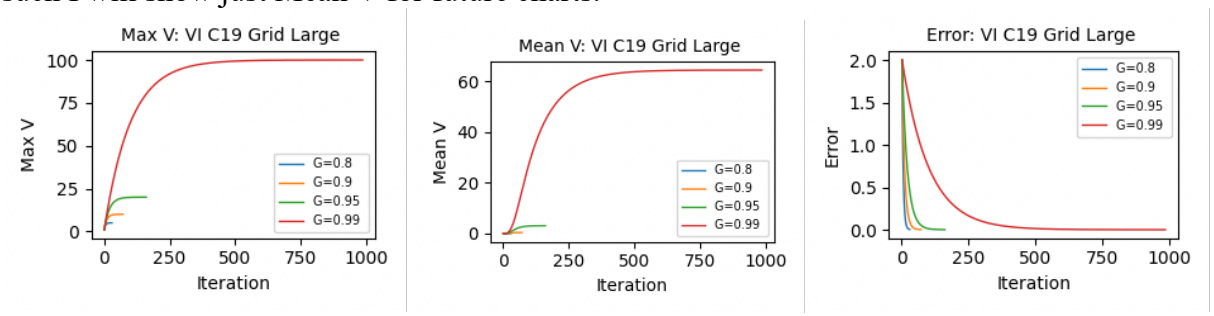
There are n x n states in each game, so the “medium” could probably qualify as large. I have included it for insights. The largest solution is omitted for complexity/ time constraints.

VI and PI

I first solved the MDP problems above using VI and PI. I was mostly interested in two things: convergence and rewards.

Convergence: In order to find convergence, I explored a number of factors (kindly provided in the MDPtoolbox-hiive). The first factor I considered was “error” which is, roughly speaking,

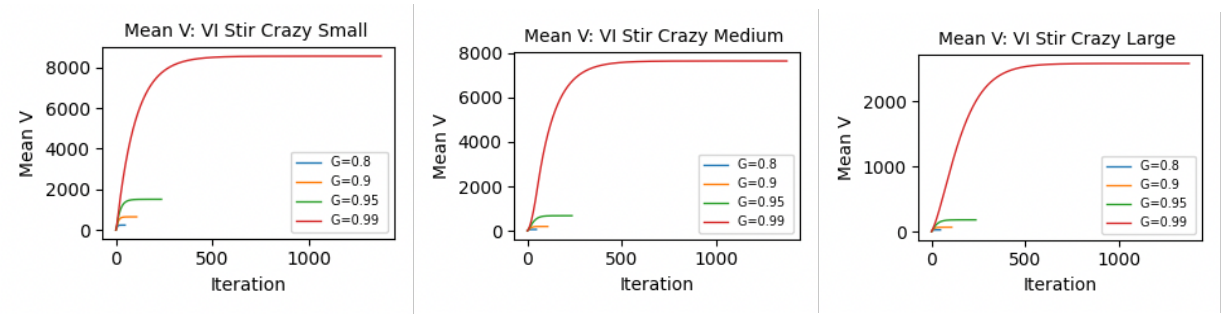
how far off is the current state from the optimal state. This should work appropriately for VI and PI as both should converge, eventually.
Next, I considered the "Mean V" which gives the average value in the states at the current round of exploration. Once the MDP is "solved" the values should stay roughly the same. Specifically, once the change in values is less than a certain threshold, Epsilon, the algorithm stops. The algorithm can also stop at a maximum number of iterations.
I also explored "Max V" which turned out to show convergence near the same time as the mean V. I expected it to come a bit sooner as the values would likely be propagated out to the rest of the states, mostly from the high states (e.g., finishing the game without COVID-19 in Stir Crazy or landing in the +1 reward spot in C19 Grid). That nuance was not enough to see in the charts I created. Here is an example of convergence showing to be the same time for these three metrics. Max and Mean arrive to their stead state (increasing from the baseline in this case) and error should shrink down to near zero. The flattening of the curves is similar across these metrics. As such I will show just Mean V for future charts.

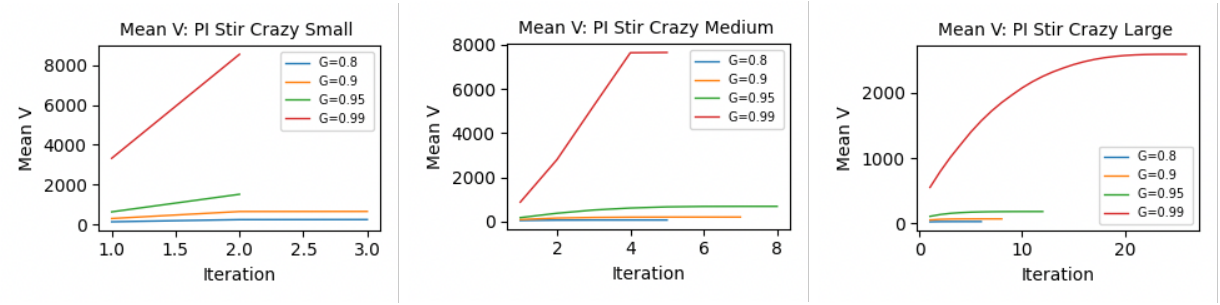


One note from this example is an overarching effect of Gamma. I looked at convergence with varying values of the decay factor Gamma (noted as "G" in the graphs). With a large G, the algorithm will take more iterations to arrive at the final answer. That turns out to be pretty constant across the board. That is because a higher G makes the problem "harder" by forcing the algorithm to look more states into the future (by discounting the future less).

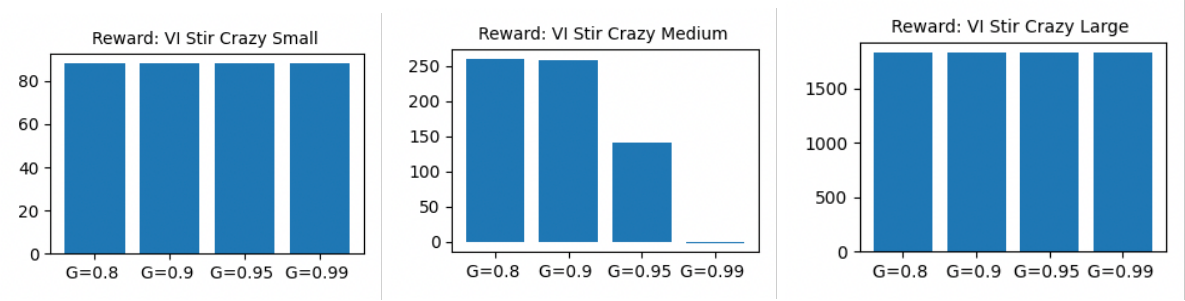
VI and PI For Stir Crazy

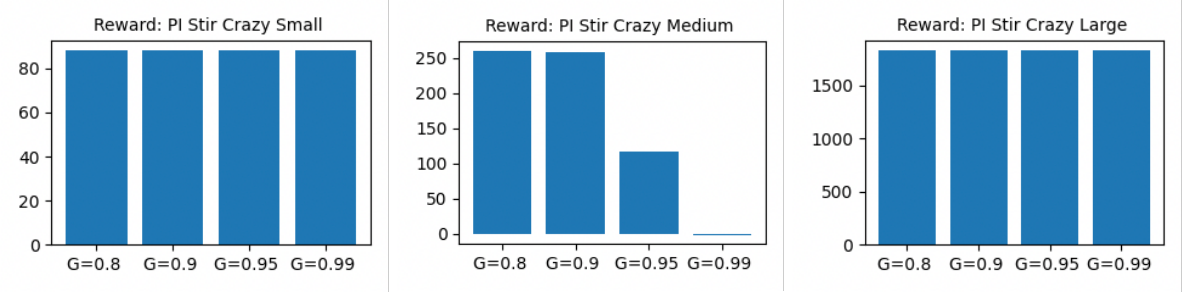
Below I consider convergency by iteration for VI and PI for the Stir Crazy problem at different sizes. One of the first observations is that PI has fewer iterations than VI. This is due to the nature of the algorithms. PI evaluates the policy and chooses a new policy, only evaluating the values of each state as necessary while VI first evaluates the states and then chooses a policy (easily) based on the values.
It is also important to note that the total value of "Mean V" is higher because it is counting values farther into the future, but that is not what we are looking at to determine convergence. Rather, we care about the flattening of the curve. Higher G makes the problem harder and creates a slower convergence as noted earlier.



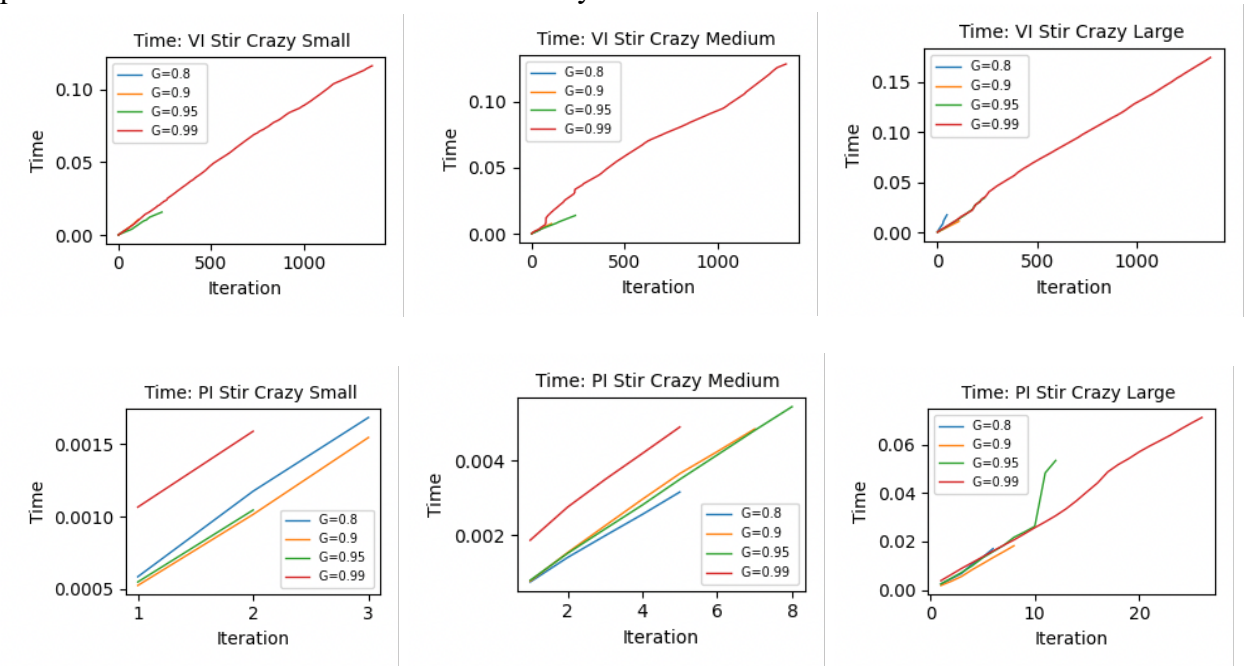


We can see convergence by a flattening of the Mean V, but we cannot see if we have landed on the optimal solution by that alone. I built a simulator function that runs the policy output by these solvers 100 times and averages the result. Below are values for each permutation. All values of G tested converged to the right solution for both the small (88) and the large version (1825). G of 0.95 and 0.99 arrive at suboptimal solutions for both VI and PI! Reasoning through it, I imagine that for specific number of time units, it would matter to different levels for far into the future the values look. When running VI and PI in MDPtoolbox I was not able to specify the exact start and stop criteria I cared about (starting at one and ending after n time periods). I found it interesting that the policy produced in these combinations leads to a suboptimal results total when calculated after the fact in the way specified. The fact that both VI and PI cam to nearly the same conclusion, leads me to believe it is more a factor of the problem itself at that G level.

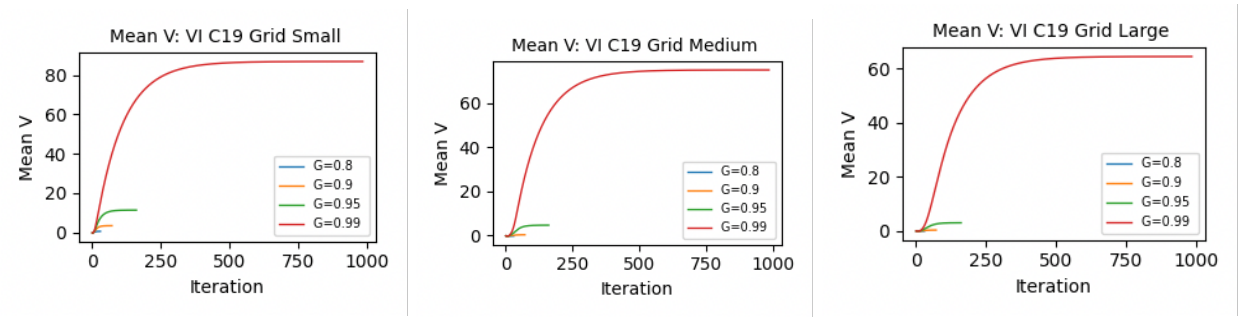


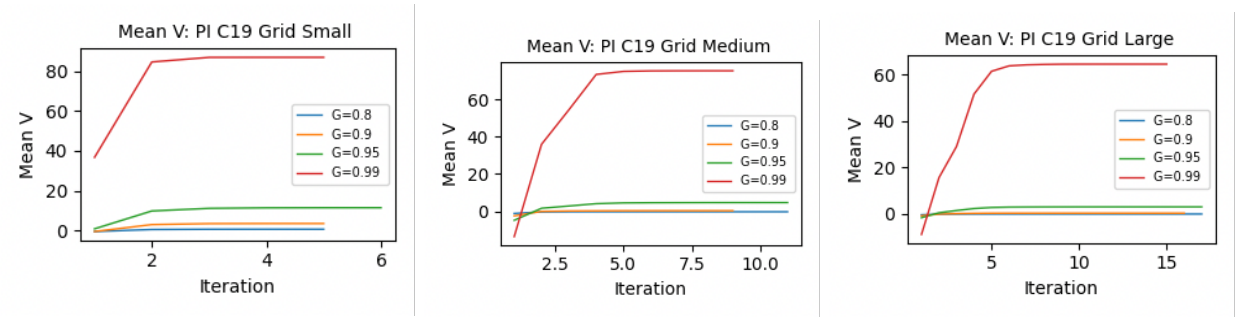


It is worth noting that although the time per iteration is higher for PI (basically always) but that the drastically reduced number of iterations more than compensates for that. This held true in my analysis of each permutation (problem and size), although there is a trend visible. If we look at the order of magnitude changes to the y-axis for the PI charts (bottom row below), we what seems to be exponential growth. The parallel charts for VI (top row below) show some increase but not drastic. Still, with the 365 states in the large problem we still see PI much faster than VI. Observed in these charts below (with iteration vs time) is that higher G also leads to hire time which we can also attribute that to the "harder" problem looking farther into the future.
In is also interesting to note that the G=0.8 takes more time per iteration for the Large Stir Crazy problem for VI which seems to be an anomaly.

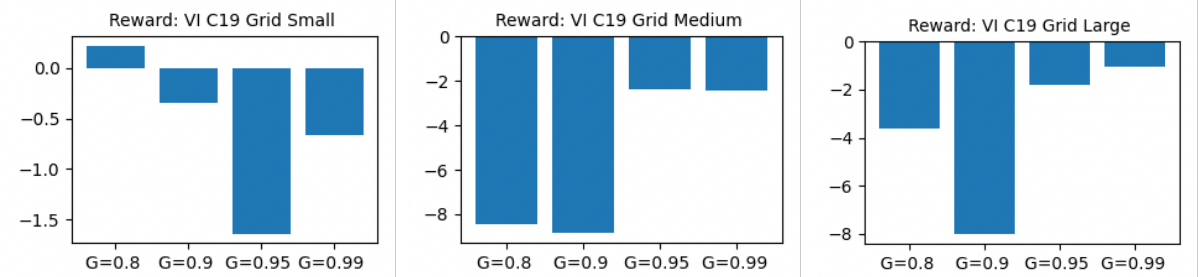


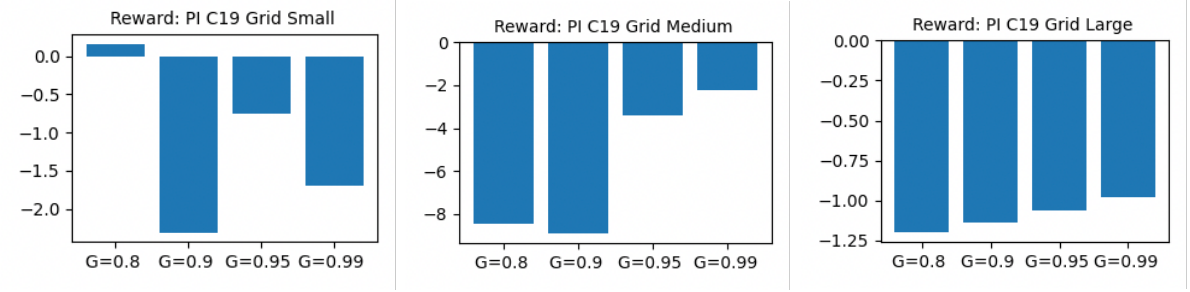
VI and PI For C19 Grid:
Overall, we see similar relative results for VI and PI on the C19 grid problem. Here it is notable how there is a much larger disparity in the effect of a larger G for VI than for PI, as far as relative increase (<50 to ~400 for VI vs 2 to 5 or for PI). Most of that can be attributed to the low number of iteration for the PI.





Looking at the states it arrived to was more interesting for the gird problem. I think that is mostly due to the nature of the problem (e.g., 4 decision per node and 4 possible end results per move).





Looking at the above, we can determine the different G needed to receive the best results on average. G=0.8 did best for the small problem. This makes sense as a smaller problem is less likely to have local optima. G=0.99 did best for the me medium and large problems.
I found it interesting though that, unlike for the small and medium problems, the VI and PI had significantly different policies based on the different G. Part of that may be attributable to the overall better performance of PI on the large problem.

There is a very similar story for time for the grid story, and the charts may seem a bit redundant.

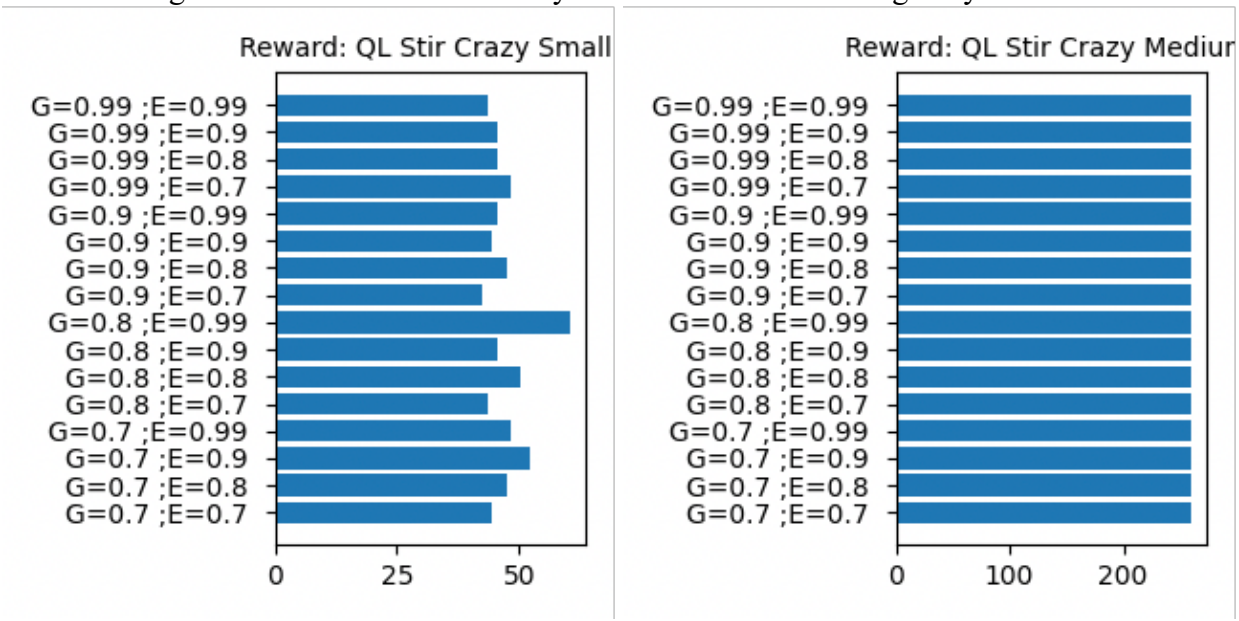
QL

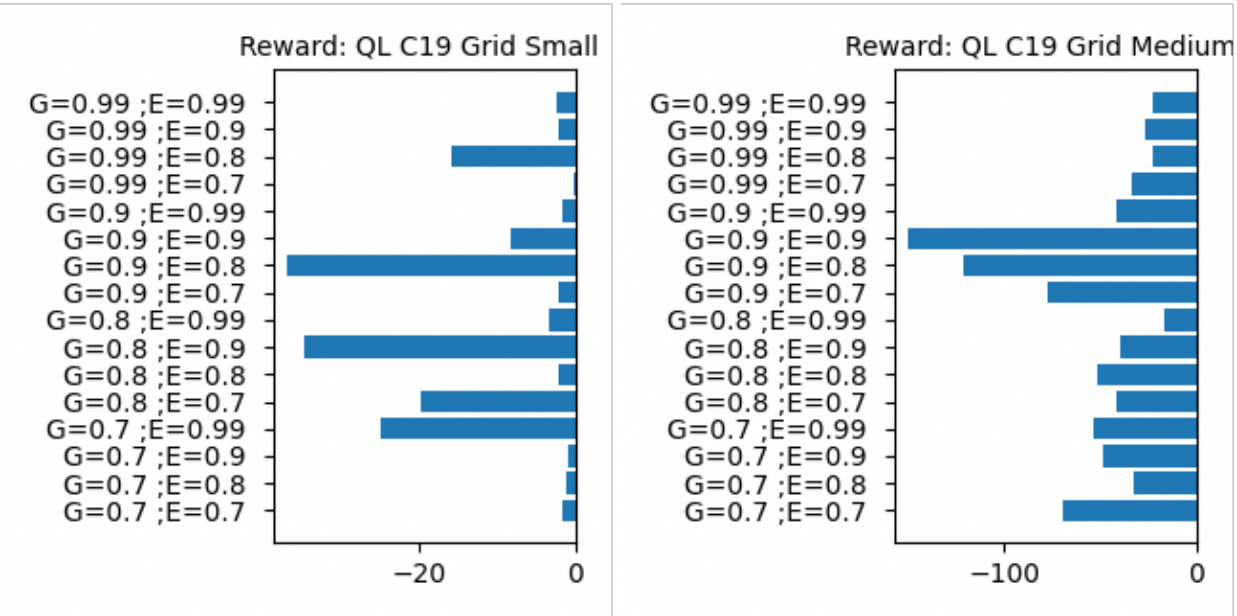
QL is rightfully in a different category than VI and PI even though they seek to accomplish the same tasks, generally. QL is not model based and therefore does not use the actual transition matrix in finding its solution. As such it is much more difficult to find the optimal solution and to converge in reasonable time. In fact, after 1 million iterations with a range of values for G and Epsilon, E, I was not able to get convergence on a number of the problems. The QL algorithm took the most time to run, by far.

I looked at QL convergence in the same way but it was a much different story. The mean V and error also seem to be recorded differently by iteration as each chart was very spiky. If you follow the tops of the frequent spikes (or ignore the frequent drops) you can get an idea of what the line is trending towards.

In learning how to understand how QL convergence manifests itself graphically by iteration, I also tried working backwards from the rewards. For that I will show the final policy's rewards first and then some of the corresponding mean v and error plots by iteration.

I looked at each problem and size by different G, similarly to VI and PI, but I also viewed them based on different Epsilon. The Epsilon is the main factor in understanding "how" the algorithm searches. For a high epsilon, it will be willing to search (choose randomly) instead of go towards the known highest values. The G then decays the E to become more "greedy".

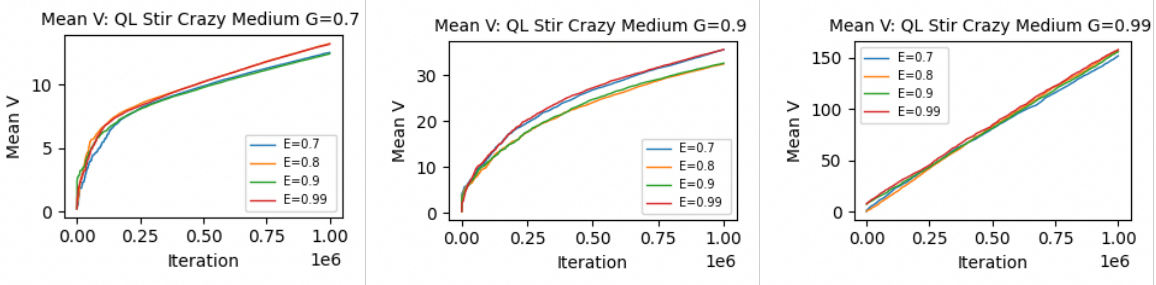




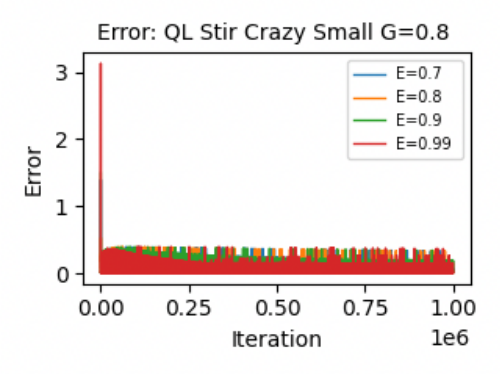
Looking at the graph above on the top right, we see that all permutations arrived at the correct solution. That apparent conversion looked like the below. (It is also interesting that this is the same one that VI and PI got wrong when G was high!) A convergence tell may be the linear increasing in values. After the "elbow" the graph increases linearly, which may be a function of how the mean v is evaluated and recorded in QL or MDPtoolbox-hiive. The error charts did not show anything particularly useful based on my understanding. They did not show a lowering and flattening.

In the above chart top left, we see G=0.8; E=0.99 as the winner. It still does not reach the 88 reward potential, even though this was the easiest problem (7 time periods with 2 choices). The charts look pretty similar to those shown below for mean V. The error does mostly seem to drop more than spike for E=.99 within the G=0.8, relative to the other E values, but it is still very unclear. That chart included below in a single row for followability.

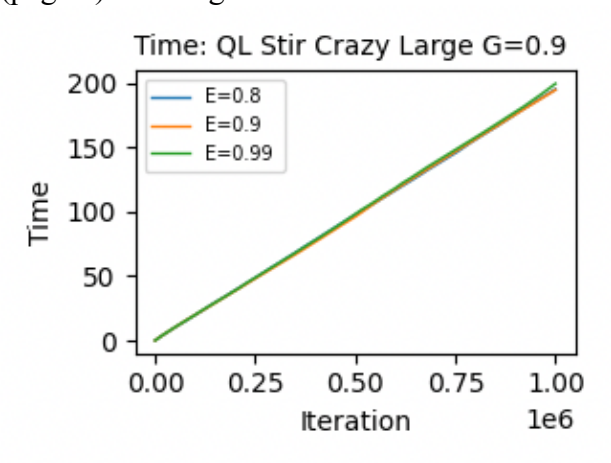
Stir Crazy Medium (Which did find the optimal policy)(Coordinates with top right above)

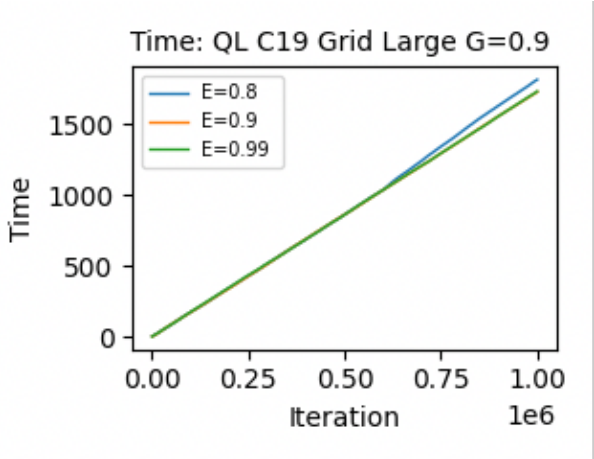


Stir Crazy Small (with E=0.99 showing as best) (Coordinates with top left)



One of the important aspect of QL vs VI and PI is that is requires more time and searching. Below are charts of time for the large problems. A comparison vs VI and PI of the Stir Crazy (page 5) is telling.





Conclusion

In this exercise I was able to see things that were, perhaps more obvious, but also was able to find things that would yield themselves to further discovery. Among those is the fact that QL for the Stir Crazy Medium size problem got to the optimal solution for all G and E while both VI and PI landed on sub optimal policies for higher G.

Work Cited

1. Python Markov Decision Process Toolbox; MDPtoolbox-hiive https://pymdptoolbox.readthedocs.io/en/latest/api/mdptoolbox.html
2. Sample grid world code https://stats.stackexchange.com/questions/339592/how-to-get-p-and-r-values-for-a-markov-decision-process-grid-world-problem