CS 7641 Assignment #4: Markov Decision Processes

William Koppelman
wkoppelman3@gatech.edu

I. Introduction

In this assignment I will explore the capabilities of three algorithms in solving two Markov Decision Processes (MDPs). I will consider value iteration, policy iteration, and Q-learning and consider the strengths and weaknesses of each at solving MDPs with small and large number of states.

II. Markov Decision Processes

A. 5x5 Frozen Lake

For an MDP with a small number of states I chose a custom built, 5x5 Frozen Lake using the OpenAI gym library [2]. The custom grid world is pictured in Fig. 1. The idea behind a frozen lake is such that you try to move in a particular direction, but since you are walking on ice, there is an equal chance of

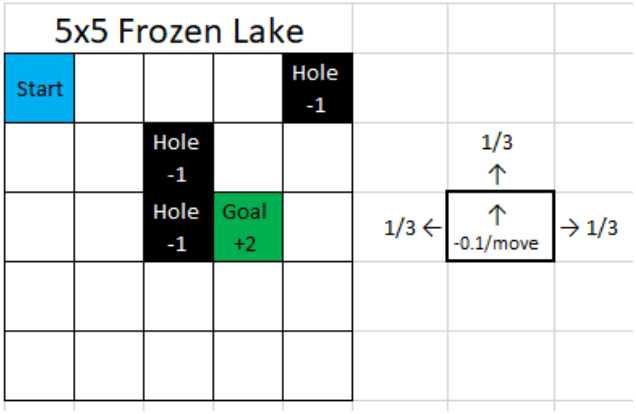


Fig. 1. 5x5 Frozen Lake Markov Decision Process

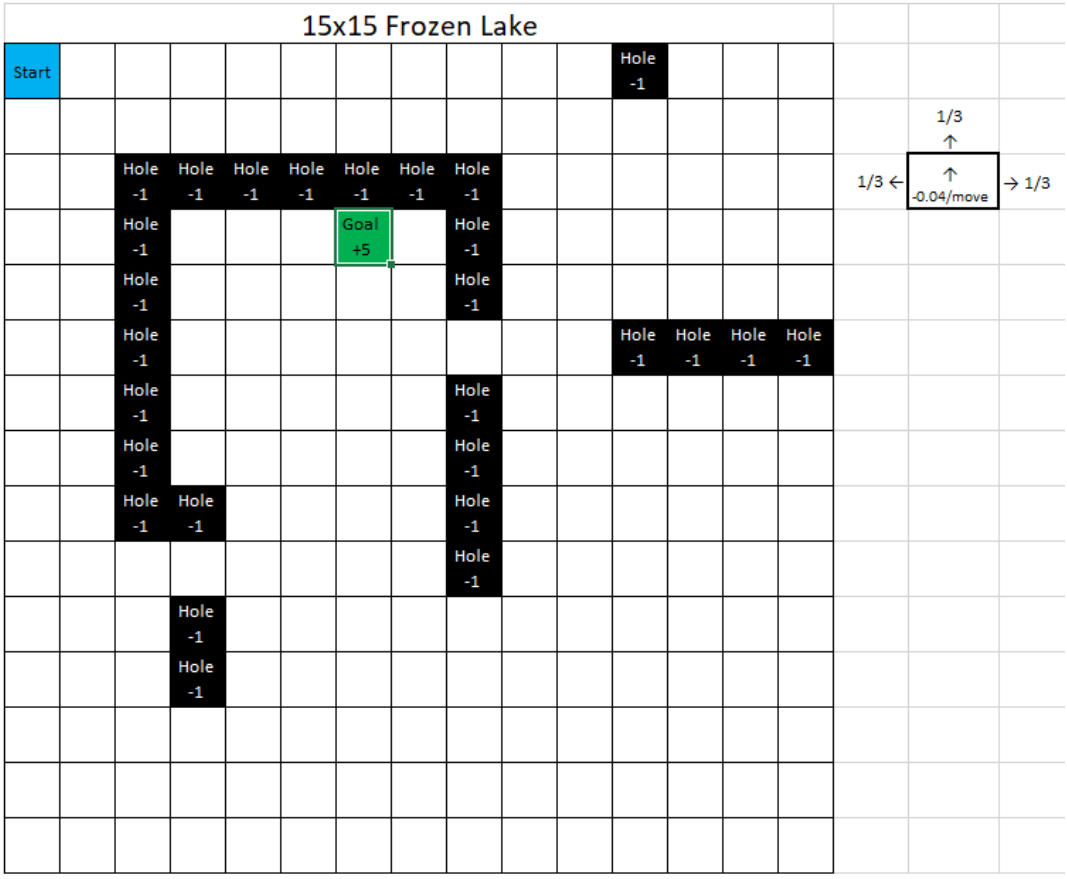


Fig. 2. 15x15 Frozen Lake Markov Decision Process

'slipping' and moving to an adjacent square instead of forward as seen in Fig. 1.

This MDP has a total of 25 states. I created it such that there would be a risk-reward tradeoff. Using a goal value of 2, a hole value of -1 and a per move value of -0.1 I wanted to see if the solutions would take the riskier path along the top row or the less risky path along the left side and bottom row.

B. *15x15 Frozen Lake*

For my second, larger MDP I chose a 15x15 Frozen Lake. I wanted to use the same environment as my smaller MDP so that I could still provide some comparisons. This MDP has a total of 225 states. However, I changed the structure of the problem so that there were multiple ways it could be solved. The goal was in having the stochasticity of slipping along with the discounting factor gamma would provide for some interesting comparisons, especially with the exploration of Q-learning.

III. Value Iteration

Value iteration starts with a random or arbitrary value function. It then iteratively updates to the next time step for each state by maximizing the values using the transition matrix and reward function. This iteration continues until the change in values is less than a threshold. It can be proven to converge and satisfy the Bellman equations [1].

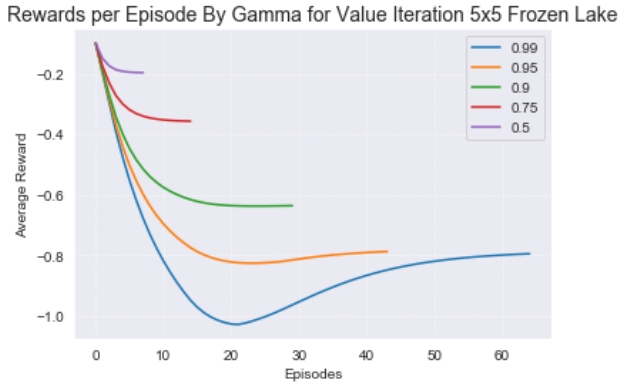


Fig. 3. 5x5 Frozen Lake VI comparison of rewards by gamma

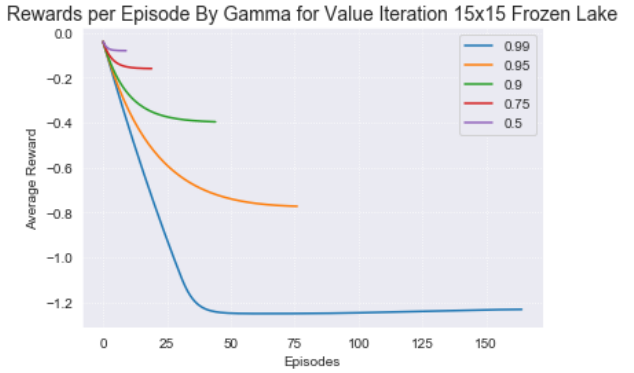


Fig. 4. 15x15 Frozen Lake VI comparison of rewards by gamma

A. *5x5 Frozen Lake*

I used value iteration to solve the 5x5 Frozen Lake using gamma=[0.5, 0.75, 0.9, 0.95, 0.99]. Fig. 3 shows the number of episodes vs. average reward until convergence. It is not surprising that the lower gamma resulted in faster convergence since the reward is being discounted so greatly that a fast solution needs to be found. In comparing the graphs for gamma values of 0.95 vs. 0.99 it is interesting that they end of

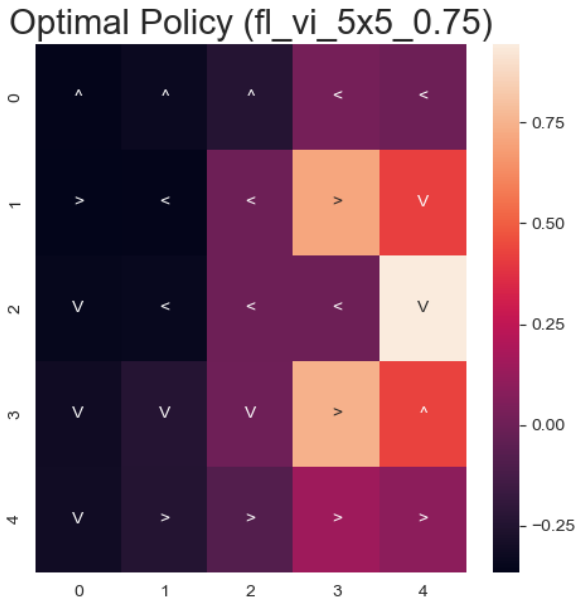


Fig. 5. 5x5 Frozen Lake value iteration optimal policy for gamma=0.75

Fig. 6. 5x5 Frozen Lake value iteration optimal policy for gamma=0.99

converging to a similar average reward when the early episodes of gamma=0.99 drop well below that of gamma=0.95.

Figs. 5 and 6 compare the optimal policies using gamma=0.75 and gamma=0.99. These show the risk vs. reward tradeoff that I was looking to explore. When the rewards are being discounted more (gamma=0.75) the optimal policy is to take the riskier path along the top row while when there is less discounting (gamma=0.99) the optimal policy is to take the safer path down the left side and across the bottom row.

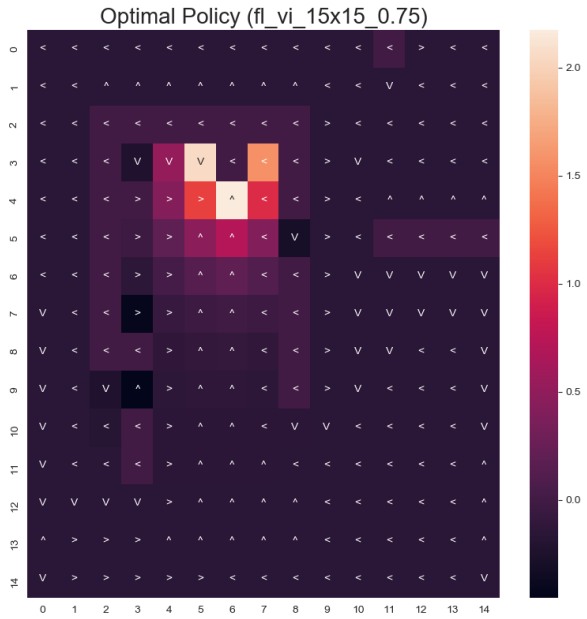


Fig. 7. 15x15 Frozen Lake value iteration opt. policy for gamma=0.75

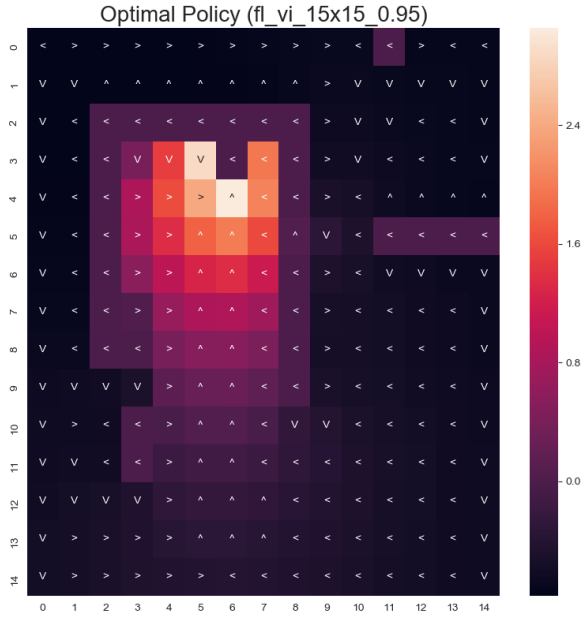


Fig. 8. 15x15 Frozen Lake value iteration opt. policy for gamma=0.95

B. *15x15 Frozen Lake*

The 15x15 Frozen Lake was evaluated with the same gamma values. The rewards per episode for each gamma are shown in Fig. 4. It is interesting to note that while the number of states is an order of magnitude larger than the small MDP, the larger MDP took only 1.25-2.5x more episodes to converge.

In Figs. 7 and 8 we can compare the optimal policies for a highly discounted (gamma=0.75) and moderately discounted (gamma=0.95) setup. Fig. 7 shows a path that is somewhat risky, going along the top row, however, when it comes to the shortcut in row 5, it does not opt to take it. The risk is not worth it in that case.

For the moderately discounted setup (Fig. 8), it is taking the much safer route all the way down the left hand side to the bottom row before moving to the center. It is being extremely safe by keeping a multiple row buffer from holes on the bottom row.

IV. Policy Iteraion

Policy iteration iteratively improves the policy. It does this by starting with a policy, evaluating it, and then improving it until the policy stops changing, it is also guaranteed to converge [1]. Typically, policy iteration will require fewer iterations but the iterations will take longer.

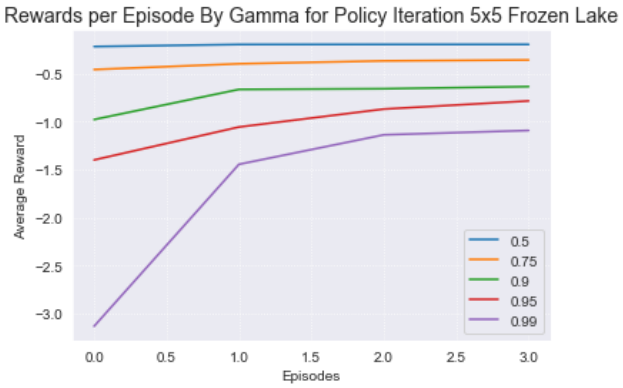


Fig. 9. 5x5 Frozen Lake PI comparison of rewards by gamma

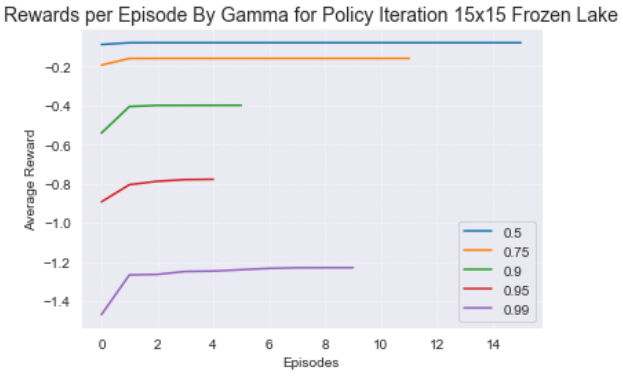


Fig. 10. 15x15 Frozen Lake PI comparison of rewards by gamma

A. *5x5 Frozen Lake*

For the 5x5 Frozen Lake we can see in Fig. 9 that all values of gamma converge in 3 episodes. This is far fewer than with value iteration. Comparing Figs. 5 and 11 we find that the optimal policy is the same, since they both converge. However, when comparing Figs. 3 and 9 we see that the rewards are different. This is because value iteration keeps iterating until the reward is below a threshold, even if the optimal policy is already found. Policy iteration is an obvious improvement on this.

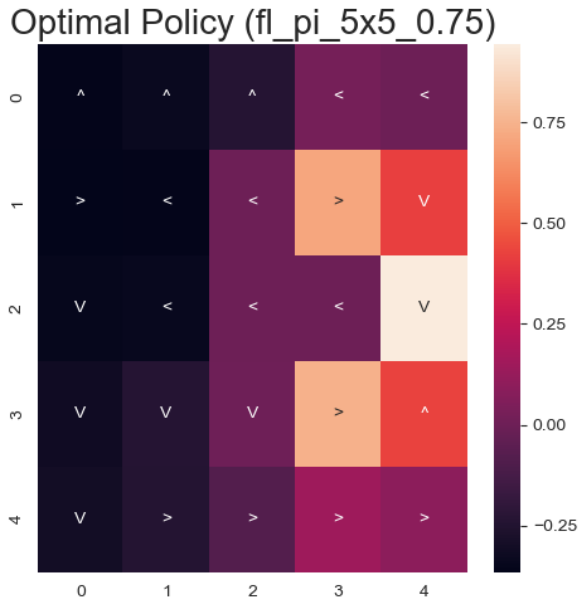


Fig. 11. 5x5 Frozen Lake policy iteration optimal policy for gamma=0.75

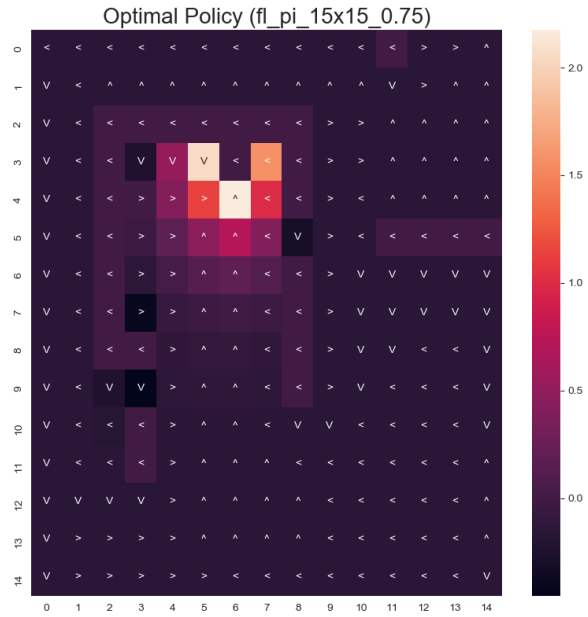


Fig. 12. 15x15 Frozen Lake policy iteration opt. policy for gamma=0.75

B. *15x15 Frozen Lake*

The 15x15 Frozen Lake converges in 4 to 14 episodes (Fig. 10), much faster than with value iteration. It also finds the same optimal policy as value iteration (Fig. 12 vs. Fig. 7).

From this initial analysis policy iteration has some clear advantages to value iteration.

V. Q-Learning

Q-learning is a reinforcement algorithm the uses exploration and exploitation. It still uses a discount factor as do value and policy iteration but Q-learning also has a factor alpha that determines the proportion of our previous value to use vs. how much of our predicted future value to use. Thus when alpha is high we use more of our predicted future value to explore a new space and, conversely, when alpha is low we exploit our current value [3].

For my implementation of Q-learning I varied both alpha ([0.01, 0.25, 0.5, 0.75, 0.99]) and gamma ([0.5, 0.75, 0.9, 0.95, 0.99]). I also used a decay function for the alpha so that it would begin by exploring more and then decay the alpha value to exploit more. I used a harmonic and an exponential decay.

A. *5x5 Frozen Lake*

Fig. 13 shows the Q-learning search of the 5x5 Frozen Lake MDP. The 2 columns of charts compare the different decays while the 5 rows compare the different alphas. You can see that none of them converge to an optimal policy. Q-learning, in theory, will converge in infinite time [4].

Fig. 15 compares 3 states of Q-learning after 10,000 episodes. They all have a gamma=0.75 and exponential decay of the alpha value. From top to bottom they range from exploitive to explorative (alpha=0.01, 0.5, 0.99). You can see that the bottom one (explorative) appears to take the risky route on the top row, similar to our optimal solution. In fact, it has 80% of the states the same as the optimal policy. The top two (exploitive and balanced) take the safe route.

B. *15x15 Frozen Lake*

Fig. 14 has the average rewards per 100 episodes with the same layout as Fig. 13. You first notice that there is a clear distinction between the 2 decay schedules. Again, none of the tests converged.

Fig. 16 shows a clear visualization in the tradeoffs between exploitation (top) and exploration (bottom). While none of the policies were optimal, it appears that exploration is necessary to discover the space.

VI. Discussion

Value and policy iteration are exact methods that each have their benefits. Table I shows a comparison between the methods. Since policy iteration is slower, it may not be appropriate for a large number of states. However, if it is a complex state space, policy iteration should be able to handle it better.

TABLE I. VALUE ITERATION VS. POLICY ITERATION COMPARISON (GAMMA=0.75)

Search Method	Episodes	Time (ms)	Converged
VI 5x5	15	0.0	True
PI 5x5	5	0.0	True
VI 15x15	20	53.3838	True
PI 15x15	12	138.0298	True

Q-learning took around 3 orders of magnitude longer to run. It also did not converge, though given longer time it should. However, it does appear powerful in exploring a space without prior knowledge.

VII. REFERENCES

[1] P. Abbeel, "Markov Decision Processes and Exact Solution Methods." [Online]. Available: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa12/slides/mdps-exact-methods.pdf. [Accessed: 14-Apr-2019].

[2] "Gym." [Online]. Available: https://github.com/openai/gym. [Accessed: 14-Apr-2019].

[3] "Q-learning." [Online]. Available: https://en.wikipedia.org/wiki/Q-learning [Accessed: 14-Apr-2019].

[4] F. S. Melo, "Convergence of Q-learning: a simple proof." [Online]. Available: http://users.isr.ist.utl.pt/~mtjspaan/readingGroup/ProofQlearning.pdf. [Accessed: 14-Apr-2019].

[FIGURES CONTINUE ON FOLLOWING PAGES]

Fig. 13. 5x5 Frozen Lake Q-learning comparison of rewards by gamma with increasing alpha (top to bottom) and harmonic decay (left), exp. decay (right)

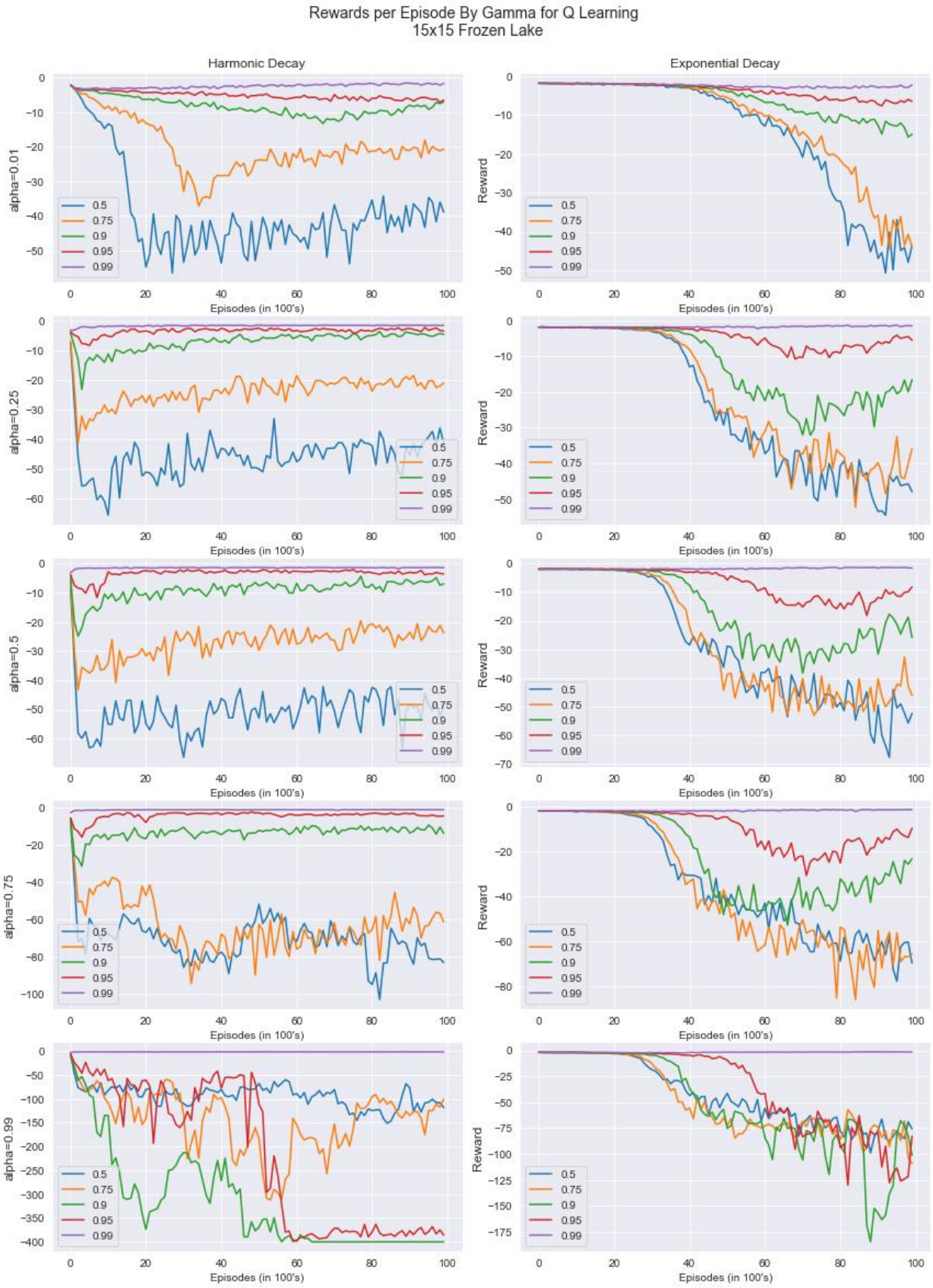
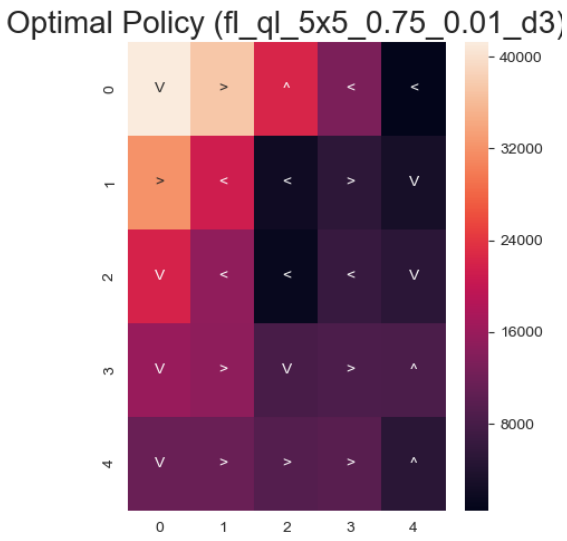
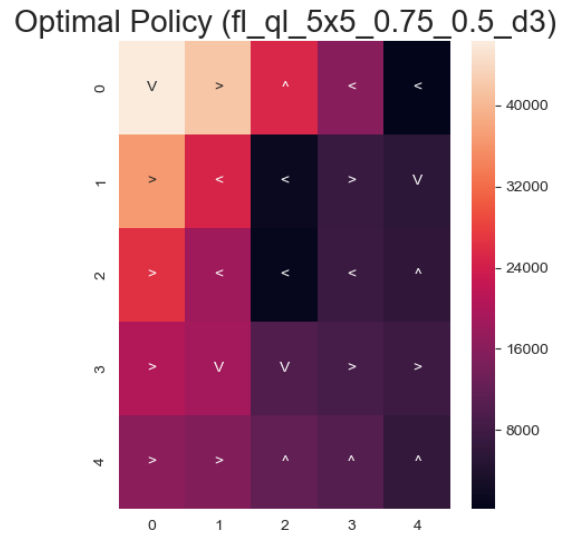


Fig. 14. 15x15 Frozen Lake Q-learning comparison of rewards by gamma with increasing alpha (top to bottom) and harmonic decay (left), exp. decay (right)





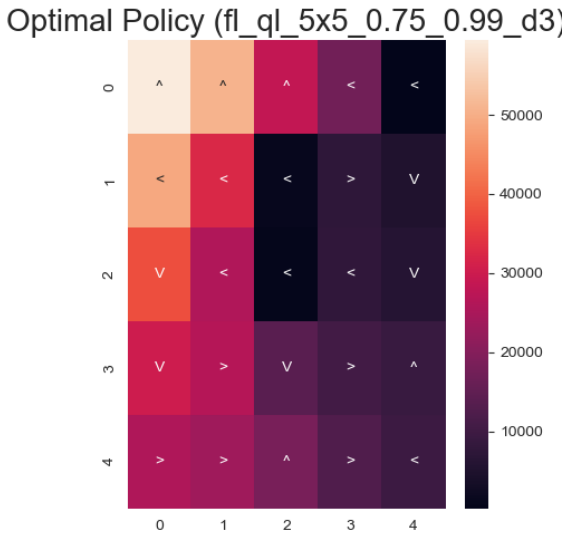
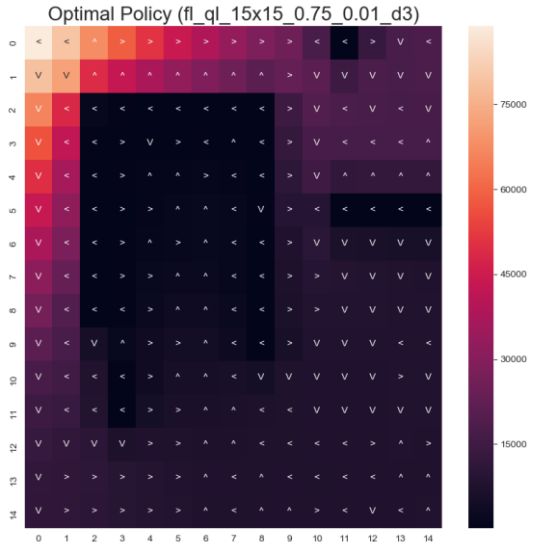
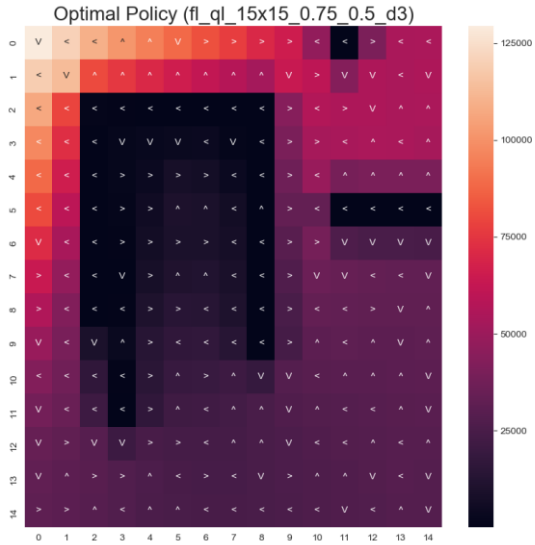


Fig. 15. 5x5 Frozen Lake Q-learning policy for gamma=0.75

alpha=0.01, 0.5, 0.99 (top to bottom) with exp. decay





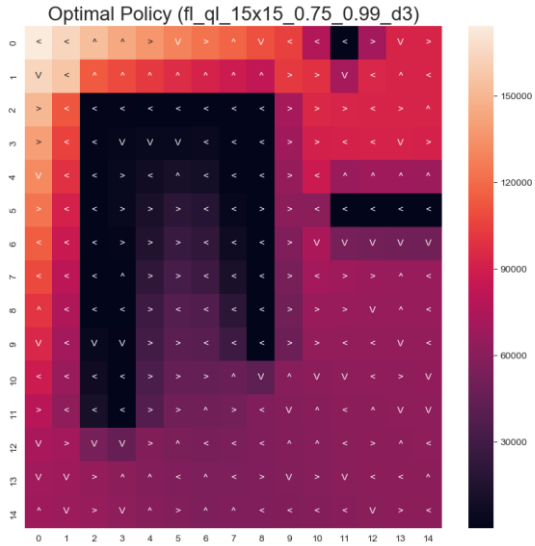


Fig. 16. 15x15 Frozen Lake Q-learning policy for gamma=0.75

alpha=0.01, 0.5, 0.99 (top to bottom) with exp. decay