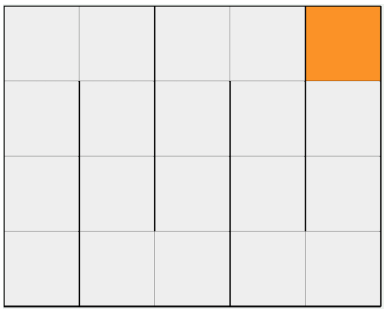
Kaley Findley
4/23/17
CS 4641 Assignment 4

Markov Decision Processes and Reinforcement Learning

Introduction

For this assignment I will be exploring the behavior of Markov Decision Processes (MDP) and using Reinforcement Learning to solve these types of problems. I chose two different MDP's in terms of the number of states and overall layout. They were custom GridWorld mazes constructed using CMU's Reinforcement Learning Simulator.

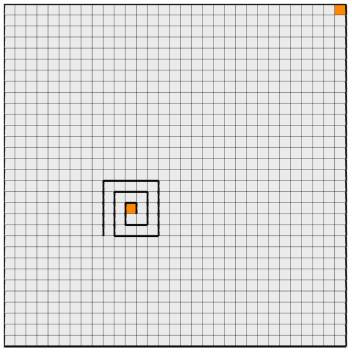
The maze setup is fairly straightforward. Components of the maze include walls, spaces (states), the starting point, and goal points. The actions available to the agent are the four movement directions: up, down, left, right. If the agent moves in a direction obstructed by a wall, then the agent will receive a set wall penalty. If the agent moves into a non-goal cell, the agent will receive a penalty of 1 point. The game comes to an end when the agent touches any of the goal states. The transition model for the domain is determined by the PJOG variable which indicates the probability that the attempted direction will be replaced by one of the other three directions uniformly.



Small-Snake Maze

The first MDP is what I call the 'small-snake MDP.' It is a 5x4 grid space with a snake-like pattern with the goal in the top right corner. The black lines are walls and those walls deal out some sort of reward, whether positive or negative. This small maze illustrates an interesting situation where the agent is adjacent to a wall at every state; in other words, it is always in a state where it could get some neg or pos reward at any time. This leads to a very interesting situation for policy and value iteration and Q-Learning algorithm.

The second MDP, which I called 'big spiral,' is a much larger maze at 31x31 with a spiral made out of walls in the lower left quadrant of the maze. It has a goal inside of this spiral as well as one in the top right corner. This MDP is interesting due to the fact that there is a lot of free space in-between the two goals and the spiral leads to some interesting behavior in those states that are surrounded by walls. Later in the analysis, each algorithm shows different policies for those states that reside almost equidistant to the two goals; it will lead to some interesting decisions based on the metrics of the algorithms.



Big Spiral

The conditions and penalties of the maze are set up in a way that incentivizes the agent to find the nearest goal state without running into walls. The introduction of wall penalties and allowing the wall penalties to be configured makes these problems particularly interesting. Rather than doing reward-based movements, the problem focuses on a positive punishment-based approach to teach the agent to react appropriately towards bumping into walls. This

encourages the agent to reach the goal not by clumsily running into walls but by navigating around obstacles.

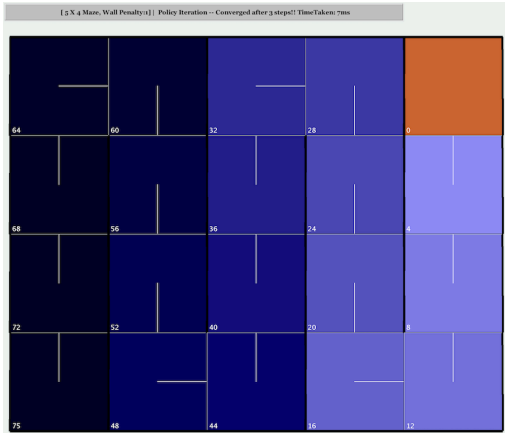
Markov Decision Processes

Small-Snake

Policy and Value Iteration

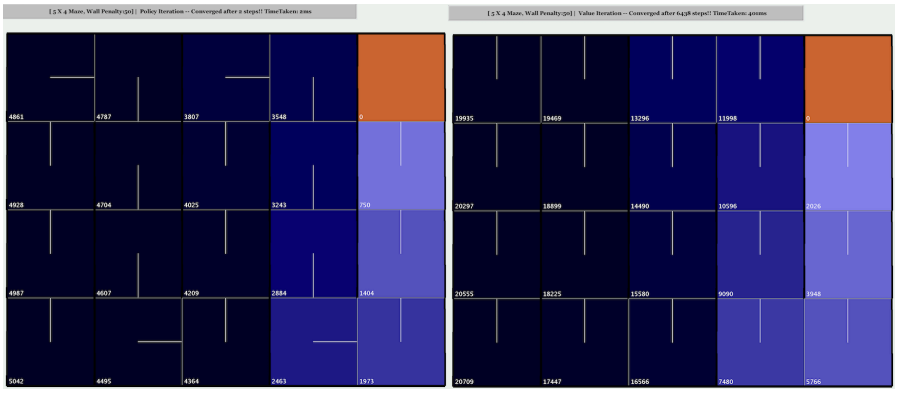
In the 'small-snake MDP' there is a path one unit in width from the starting point (bottom left corner) to the goal state. The expectation for this map was to be able to observe the effect of it being a simple MDP with few total states, the effect of it being one-unit narrow, and the effect of it having multiple sharp turns. I expected that all of these aspects to form a distinct policy, as there should be strictly one solution.

I manipulated the values for the PJOG and the precision parameter for policy and value iteration. I gradually increased the PJOG value from 0 to 1. I wanted to see how the policy changed as the uncertainty increased. For policy and value iteration, with penalties of 1 and 50, they converged to the same policy for a relatively low uncertainty (around 0 to 0.5) as shown below. The precision parameter simply dictated how sensitive to minor changes the algorithm was. The only major difference between the different values of PJOG or precision, was the actual values of the states. The values decreased as the agent got closer to the goal and as the uncertainty increased these values started off much larger towards the beginning. This behavior was expected because the agent was far from the goal and was very likely to hit one of the two to three walls around it.



Policy Iteration PJOG: 0.1 Penalty: 1

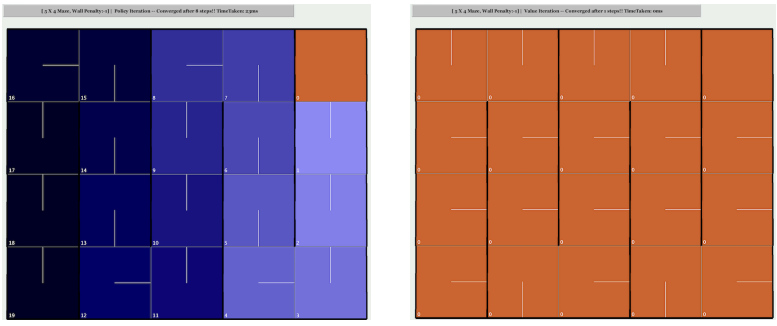
Overall, policy iteration converged faster and on less iterations than value iterations. Due to the nature of each of these processes that is to be expected. Value iteration took several minutes to converge in the cases where PJOG was 0.75 whereas policy iteration would only take a few seconds as shown in the figures below. In the case of a penalty of 50, the policy for value iteration with PJOG of 0.75 was very bizarre. No matter what state you were in the policy was to go to in the upward direction, regardless of whether it was a wall or goal. The point I wanted to make was the massive difference in iteration and time between the two algorithms.



Policy Iteration: Penalty: 1 PJOG: 0.75 Value Iteration: Penalty: 1 PJOG: 0.75

The figure on the left is policy iteration and it only took 2 iterations to find the expected optimal policy, whereas the figure on the right is for value iteration. It took 6438 iterations to find a suboptimal policy. This is due to how PJOG works. When PJOG is 75%, then there is a 25% chance to move in the desired direction, and a 25% chance to move in any other direction. This means that the actions are uniformly random. The agent has no control over what action to choose given a particular state, which explains why the policy could not be improved. On the hand, due to the greedy nature of value iteration, it makes sense that the policy would point in the direction of the calculated, in this problem, maximum negative expected value of the next state. With lower PJOG values both processes produced the same policy, but policy iteration would still outperform value iteration in terms of number of iterations.

The MDP's with a penalty performed as I expected, however when I changed the wall penalty to a negative penalty (positive reward), the policies that were found were very different from the ones found above. For example, I set the PJOG to zero to see how the algorithms acted with no uncertainty in the world (one can dream). With the wall penalty set to negative 1 and PJOG to 0, the policies found for value and policy iteration were vastly different, as shown below.

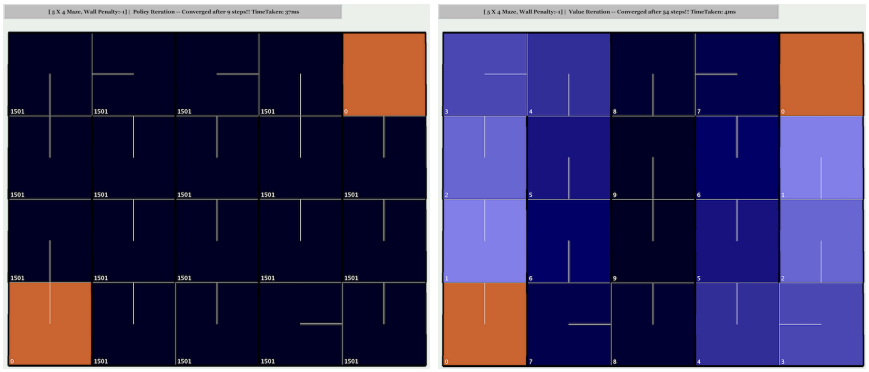


Policy Iter: Penalty: -1 PJOG: 0 Policy Iter: Penalty: -1 PJOG: 0

Policy iteration found the optimal policy in 8 iterations. It is obviously a lot more consistent in this problem because it works directly with the policies. If it can't find a better overall policy, it converges on the latest one it found. However, value iteration has a more local view. It works in a similar sense as hill climbing in regard to it's greedy nature. It moves in the direction that yields the maximum value for the next state as shown above. Since the wall penalty actually gives you a positive reward of 1, hitting the wall cancels out the transition penalty of 1, so the agent would hit the wall no matter where it was. Even if it's in the state next

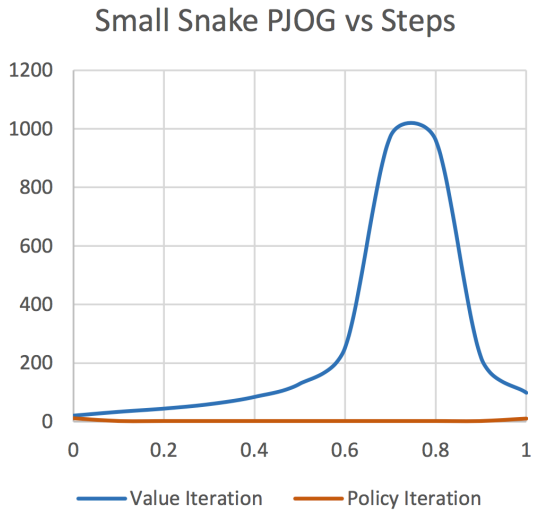
to the goal, it would rather keep going towards the wall and zero out its reward instead of accumulating a negative reward and then ending the game (since there's no goal reward).

When I flip the PJOG to 1, meaning there is a 0% chance that the agent will move in the intended direction, value iteration's policy is to move to each end of the maze. In the center of the maze, the policy is to move towards the middle, which would actually make the agent move away from the center and closer to the end of the maze. Once at the end, by moving in the opposite direction of the wall, the agent is guaranteed to hit the wall, resulting in a maximal reward. However, policy iteration had a difficult time finding an optimal policy. As shown in the figures below, the values for each of the states are greater than all of the states in the value iteration figure combined.



Policy Iteration PJOG: 1 Penalty: -1 Value Iteration PJOG: 1 Penalty: -1

Comparing the number of steps for different PJOG values below, we first see that the number of steps in policy iteration is multitudes smaller than the number of steps for value iteration. This is because policy iteration runs through multiple value convergences in order to converge on policy. Secondly, we see there is a large spike in number of value iteration steps around PJOG of 0.75. This makes sense because at PJOG of 0.75, the transition function is uniformly random, which means it will require significantly more steps to progress towards an optimal policy.



According to the graph above, policy iteration is optimal for when the transition function is fully deterministic. It takes only a few iterations to find the optimal policy and for the most

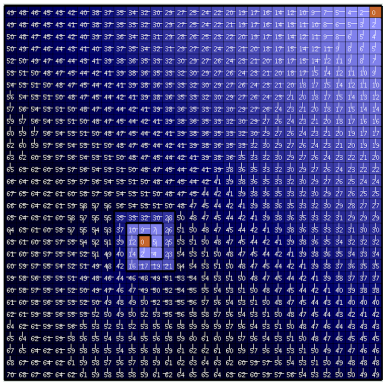
part, does not get caught up in some of the edge cases as value iteration did (i.e., when PJOG=0 and penalty=-1). There is spike around 0.75 for value iteration as expected due to the fact that policy caused the agent to hit more walls.

Big-Spiral

Policy and Value Iteration

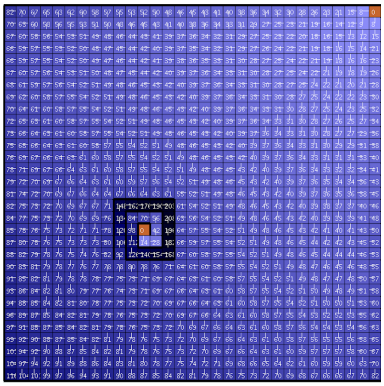
As mentioned in the introduction, the big-spiral MDP is much larger than the small-snake MDP and has two goals instead of one. One goal is inside the spiral and the other is in the top right corner and has the same initial starting point (bottom left corner). The goal inside the spiral requires a minimum of 42 steps to reach and the other goal requires at least 60 steps to reach. This maze will test the agent's 'courage' to approach the walls in this maze. It should be interesting to see what metrics causes the agent to enter the spiral or go around it.

To begin this analysis, I began with a relatively small PJOG value of 0.25. Both policy and value iteration converged on a similar policy. Both algorithms did not avoid the spiral; they both took the risk of entering the spiral to reach the goal. The only difference between the two was the number of steps taken to converge. As in the small-snake MDP, policy iteration took only a few iterations (8 steps) to converge while value iteration took 100 steps.



Policy/Value Iteration Penalty: 1 PJOG

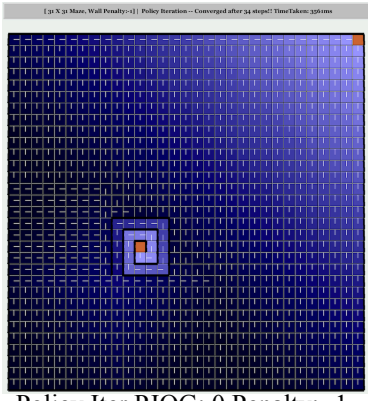
However, when I increased the penalty to 50, that is where I began to see some differences in the policy. Both policy and value converged on the same policy when PJOG was 0.25 again, but the policy was to avoid the spiral at all costs. Even the first 3 states in the spiral, the policy was to get out of it. The policy was to go to the goal in the top right corner. The states inside of the spiral are a darker blue meaning their values are very high in relation to the other states (shown below).



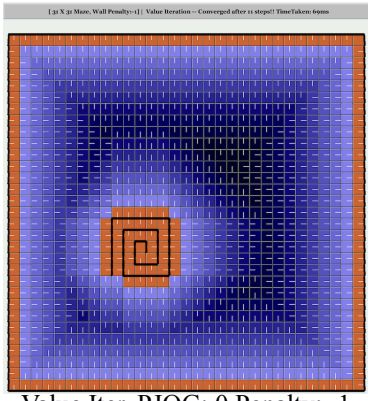
Policy/Value Iter. PJOG: 0.25 Penalty: 50

When I increased the PJOG values, the policy did not change from the one shown above, except that the individual state values would increase greatly. This is expected because the agent's actions would become increasingly inconsistent or random, therefore when the agent attempts to follow the policy, there is a greater chance that it would not end up in the expected state, therefore accumulating more cost along the way to the goal.

The optimal policy found by these two processes began to change a lot when I made the wall penalty negative 1, meaning the agent received a positive reward when hitting a wall. I expected the policy to lead the agent to hit more of the walls and to head to the spiral at all costs. I set the PJOG to 0 so the reward was mainly dictating the policy. As shown in the figures below, value iteration behaved in the way I hypothesized. Policy iteration still found the policy shown above. This is consistent in the behavior policy iteration as shown throughout this analysis since it estimates the optimal policy based on the overall values of the states as shown in the following the figures.



Policy Iter PJOG: 0 Penalty: -1

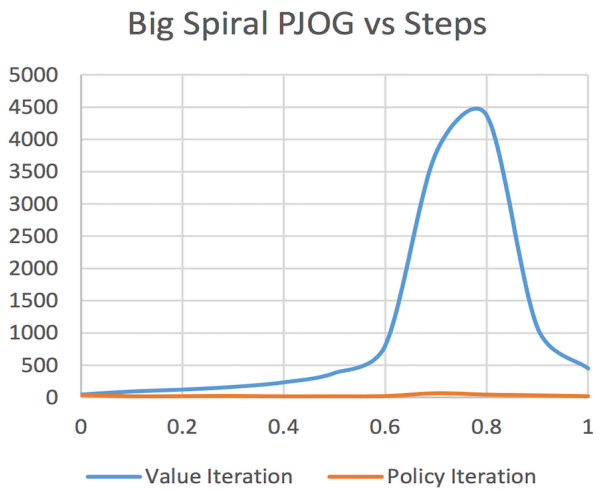


Value Iter. PJOG: 0 Penalty: -1

In the figure on the right (value iteration), new goal states appear around the walls. This is because hitting the walls would result in a maximal reward and with 0 stochasticity, it is guaranteed the agent's intended action will occur. The states close to the spiral head towards it while the others head to the nearest boundary wall. For policy iteration, the resulting policy is not what I expected. Inside the spiral the agent will head towards the goal. Also, besides some of the states near the left side of the spiral, the policy is to go in the upward direction and then finally towards the goal in the top right corner. The policy found by value iteration is more optimal than the one found by policy iteration because it prioritizes on optimizing the values over the entire space.

Next I tried a reward of 0.5 to see if the behavior would change from the figures above. It turns out that since the reward is so small compared to the transition penalty, it seeks to reach the closest goal state as soon as it can. The resulting policy for both algorithms look very similar to the figure in the beginning of this section when PJOG was 0.25 and the penalty was 1.

Similar to the small-maze MDP, we see that the value iteration takes on a parabolic structure, with the peak being when the transition function is uniformly random. The number of steps taken by value iteration is drastically bigger than the number of steps taken by policy iteration. Compared to the small-snake MDP, big-spiral has a much bigger state space, which leads to a greater number of steps to converge.



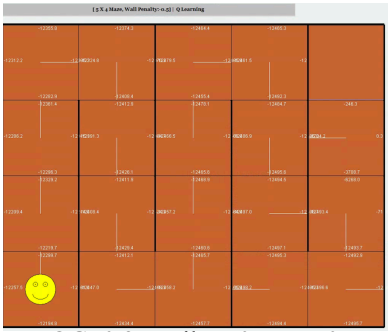
Reinforcement Learning

Q-Learning

After analyzing how policy and value iteration worked, I began to explore the process of Q-Learning on the small-snake MDP. I manipulated values for PJOG, epsilon and learning rate. I used the same values of PJOG as in the previous section (0, 0.1, 0.25, 0.5, 0.75, 0.9). I used epsilon values of 0.1, 0.5, and 0.9. Lastly, I used learning rate values of 0.2, 0.7, and 0.9. The epsilon value determines the probability of whether the agent should go to the next state with the estimated Q-value or act randomly. My hypothesis was that Q-Learning would take longer to converge than policy or value iteration, but it would still find the optimal policy as long as the epsilon value not close to 1 and the learning rate was a moderate level to allow the algorithm to learn the environment. The learning rate is crucial to the important trade-off between exploration and exploitation. The Q-Learning algorithm needs to be able to learn what actions to take at every state, but also use what it has already learned to make an optimal decision to lead to an optimal policy.

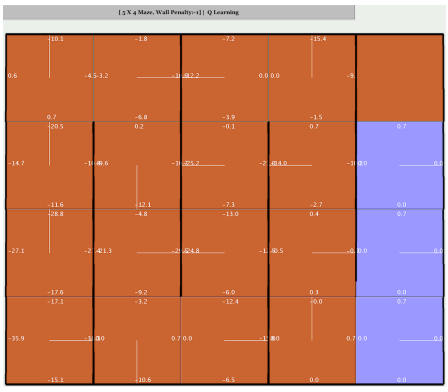
Small-Snake

I began by manipulating the epsilon values while keeping the learning rate at 0.7 and PJOG at the default value of 0.3. My intention was to see how the decisions changed (or didn't) based on the probability distribution of epsilon. The figure below shows the episodic progress of the infinite time horizon as a result of having a negative wall penalty. The only factors that will move the agent towards the goal are the epsilon exploration or the PJOG stochastic action. This particular situation was going to go on for an extremely long time.



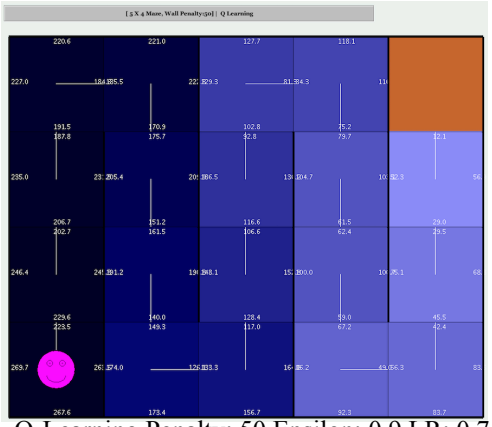
PJOG: 0.3 Epsilon: 0.5 LR: 0.7

The figure to the right represents any positive reward value greater than or equal to the transition penalty. This is consistent with the concept of Q-Learning. It will basically oscillate back and forth between going toward each of the walls surrounding the respective states. It is very similar to the policy found by value iteration. It sees hitting the wall as beneficial; since the agent gets a reward of 1 for hitting the wall, it means the agent will zero out the penalties on the way to the terminating states (goal). Because of this the agent is less likely to try different actions because they are worse for the agent (in terms of penalty). Moving along states means the agent accumulates penalties and why would an agent do that if hitting the wall results in a positive award. However, when the epsilon is high then the agent is more willing to try things that could potentially be suboptimal, but it means the agent could also find the goal faster. In the case where the award is 0.5, this becomes less of a problem because it takes a lot of wall hits to cancel out the score from moving from state to state. Then at some point in time (with some randomness) the agent needs to try to terminate.

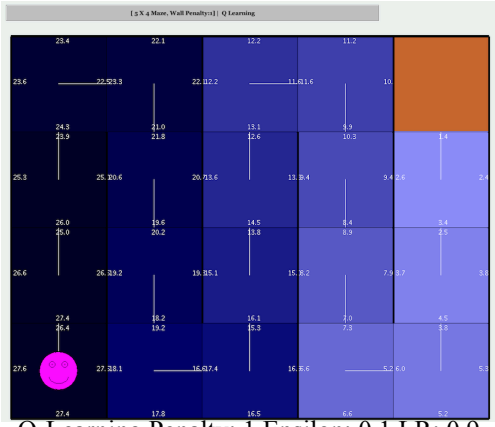


Q-Learning Penalty: -1 PJOG: 0.3 Epsilon: 0.1

When the wall penalty was greater than or equal to the transition penalty of 1, then Q-Learning found policies and state values similar to policy and value iteration processes. On average Q-Learning still took longer to converge because it has to visit every state, action pair in the grid, but it still found the expected optimal policy.



Q-Learning Penalty: 50 Epsilon: 0.9 LR: 0.7

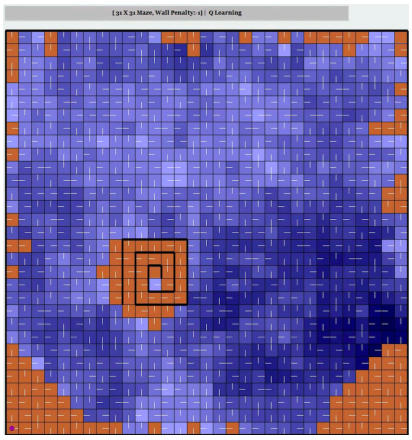


Q-Learning Penalty: 1 Epsilon: 0.1 LR: 0.9

It took a lot less time to find an optimal policy for the different penalty values rather than the different reward values. I believe this was due to the fact that it was easier to calculate the Q-value for each of the state, action pairs. In other words, it was clearer which state, action pair was the better choice, thus led to a faster convergence. In context of the exploration-exploitation tradeoff, it was easier to learn from the events in the grid world, exploration was more consistent, thus leading to more optimal decisions in the exploitation phase.

Big-Spiral

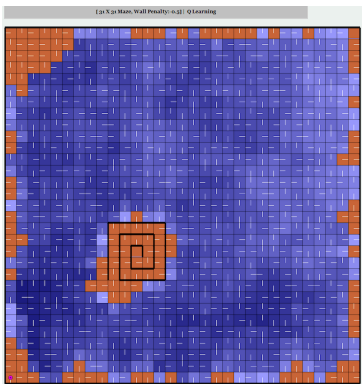
The size of this maze led to some time constraints with Q-Learning due to the amount of state, action pairs the algorithm needed to explore. Therefore a lot of the following figures are after about 5-10 episodes. Ideally, I would have liked to run about 1000 episodes to get a more optimal resulting policy. Regardless, even after 5-10 episodes, the trend of the what the Q-Learning algorithm was finding was relatively clear. The following figure illustrates the policy after 5 episodes with a wall reward of 1 with a learning rate of 0.7 and epsilon of 0.1.



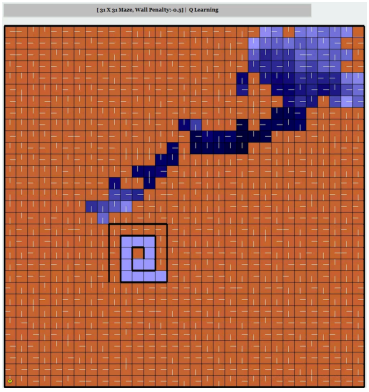
Q-Learning: Penalty: -1 Learning Rate: 0.7 Epsilon: 0.1

As you can see, the policy appears to be converging on a similar policy as value iteration in this scenario. The spaces adjacent to the walls look like goal states except for the fact that they aren't terminating conditions, but they cancel out the transition penalties the agent has accumulated. I believe with more episodes, the policy found by Q-Learning would look even more like the policy found in value iteration.

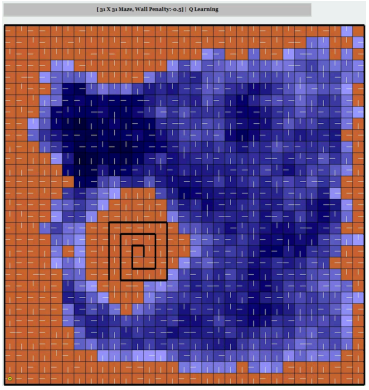
In this bigger MDP, the epsilon value was more impactful than it was in the small-snake MDP. As the epsilon value increased, the more drastic the policy would change. This is due to the increasing randomness in the policy. The agent is more and more likely to take a random action rather than the action corresponding to the estimated optimal Q-value. In the case where the reward was 0.5, the lower epsilon values corresponded to policies similar to the one described above, which is closer to the expected policy for this particular situation. However, as the epsilon increased then you start to see the randomness and how that impacted the policy in a suboptimal way. You can see this behavior in the following figures.



Penalty: -0.5 Learning Rate: 0.2
Epsilon: 0.1

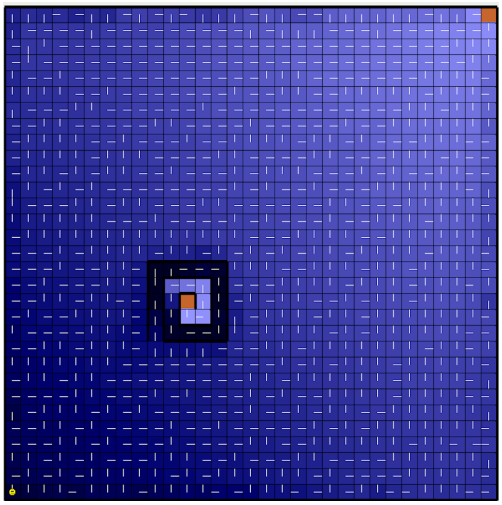


Penalty: -0.5 Learning Rate: 0.7
Epsilon: 0.5



Penalty: -0.5 Learning Rate: 0.7
Epsilon: 0.9

When the penalty was 50, Q-Learning found a similar policy to both policy and value iteration in the fact that it tried to avoid the spiral at all costs. The states inside of the spiral had a higher value than the other states outside of it. The policy was mainly to go to the goal in the top right corner. No matter what the epsilon value, the policy was about the same. I was also able to run Q-Learning on this scenario for 100 episodes, so the policy is a bit more developed than the previous cases.



Penalty: 50 Epsilon: 0.5 LR: 0.7

Conclusion

For the MDP analysis, I have concluded that policy iteration is the more optimal choice for relatively small, discrete state spaces. I believe it would be more difficult for policy iterations to estimate an overall policy for a very large, continuous state space; or it would take a lot longer to converge than the two problems I used in this assignment. Value iteration takes a lot longer in terms of iterations to converge. This is due to the fact that it has a more local approach and is more subject to minor changes in-between states. If time or energy was not a factor I would like to have tried an even larger space, or perhaps a continuous space, to see how these two algorithms would perform.

For the Reinforcement Learning analysis, I used Q-Learning. I liked the concept of this algorithm and the trade-off between exploration and exploitation. It was interesting to try to find that balance in the epsilon and learning rate values to make sure that the algorithm was indeed learning something from the agent's interaction with the world, and also using that information to continuously improve its decision making process. It was also interesting to see the drastic difference in convergence behavior between when the wall penalty was a true penalty and when it was actually a reward. The time taken to converge for the latter was extremely big. If time was not an issue I would have experimented with more epsilon values and learning rates and maybe went into more on how those two parameters affect one another.