Machine Learning Assignment 4

Reinforcement Learning

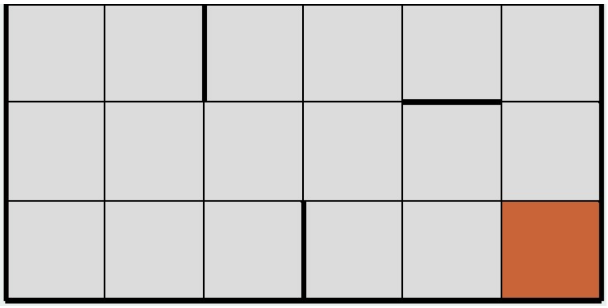
Keertana Subramani

Choice of Markov Decision Problems

In this assignment we have chosen to work with mazes – one small and one large – as our Markov Decision problems. Each of the mazes has several states i.e., boxes that the agent can be in, and a "goal" or "sink" which the agent has to find a way to. Mazes are interesting to study and can be used understand both MDPs and reinforcement learning techniques because they provide a clear distinction of states, actions and rewards. Moreover, many problems in the real world can be modeled as mazes. For instance, when we want to walk from an origin to a destination point and hope to take the shortest routes, but we cannot walk through buildings, we can model each distance unit as a state and buildings and walls on the way as obstacles. This is similar to a maze where the destination is the goal state with high reward and every step has an attached penalty (thus motivating us to minimize distance).

In this analysis, we run MDPs through doing Value Iteration and Policy iteration on the mazes. We also implement a reinforcement learning algorithm called Q-learning which uses a combination of exploration and exploitation to find to the best optimum in an previously unknown environment of the mazes. These analyses have been run using Carnegie Mellon University's RL_Sim package.

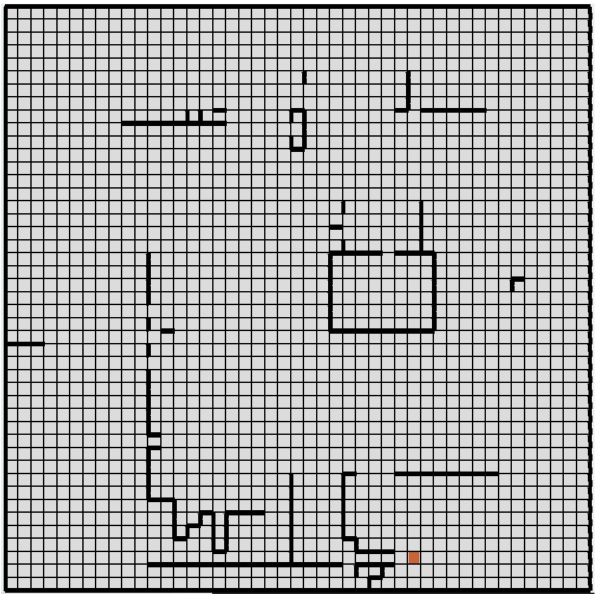
Small Maze:

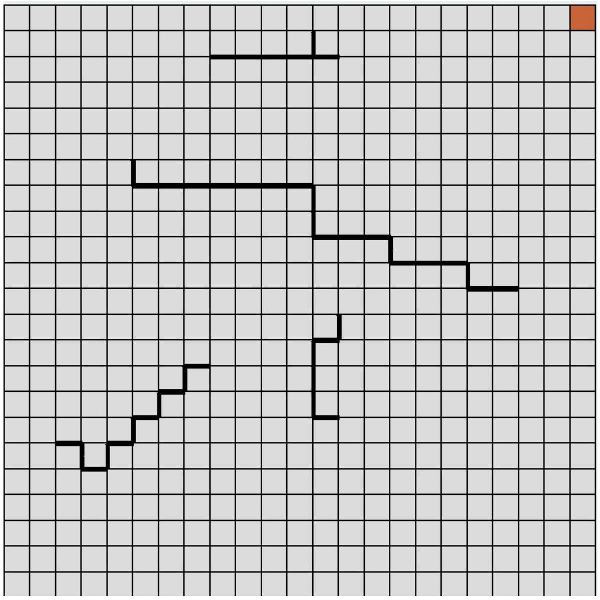


The small maze contains 18 states, 3 walls and a sink (or goal). An action that traverses every extra step adds a penalty to the total value function of the agent. The agent tries to minimize the number of steps (hence total penalty) to get to the goal (sink) as soon as possible.

Large Maze:

The large maze contains 45*45 states i.e., a total of 2025 states. There are, again a few walls and a sink, (but in this maze it will be much more difficult to learn the optimal policy to enter the sink because there are so many possible combinations of steps that can be taken to reach the final path). Value iteration and Policy iteration have been run on this maze. However, since the maze is too large for running the Q-learning algorithm with the limited processing power/memory on my computer, for exploring reinforcement learning I've created another maze of size 23*23 with 529 states. These big and medium big mazes are shown below:





Markov Decision Problems and Reinforcement Learning

Markov Decision Problems: A Markov decision process is a discrete time stochastic control process. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. It is defined by a set of possible states, actions and a reward function and assumes the Markovian property which states that the effects of actions taken in a state don't depend on any history of how we got to the state. In this assignment, we define two MDPs, the small and large maze, and run policy and value iterations on them to find the optimal policies.

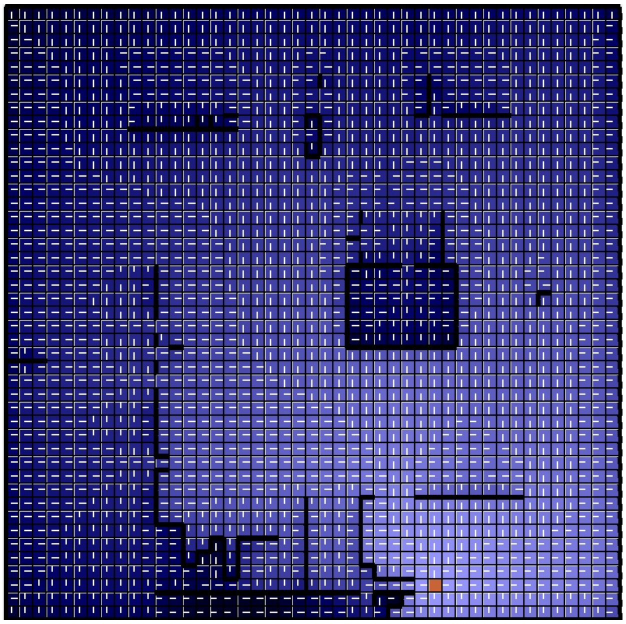
Reinforcement Learning : is a machine learning approach which assumes a Markov decision process whose model and reward function we don't know but we run the algorithm repeatedly to learn about. RL studies how agents should take actions so that discounted sum of cumulative rewards is maximized, or penalties minimized. In this analysis, I implement Q-learning exploration techniques for reinforcement learning.

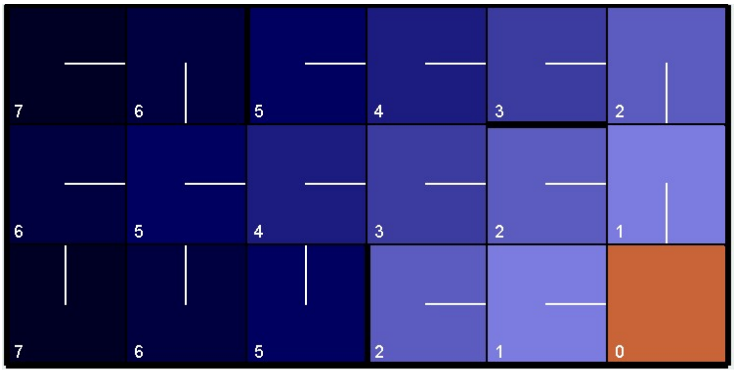
Running and analyzing algorithms

Value Iteration: Value function is a state-action pair that determines how good it is to perform a given action given a particular state. The goal would be to minimize the discounted present value of the value function (where the values corresponding to each step represent penalties) represented by the *Bellman equation* and reach the goal with the least number of steps possible. In Value iteration, the optimal value function is computed at every stage and is used to determine the optimal policy.

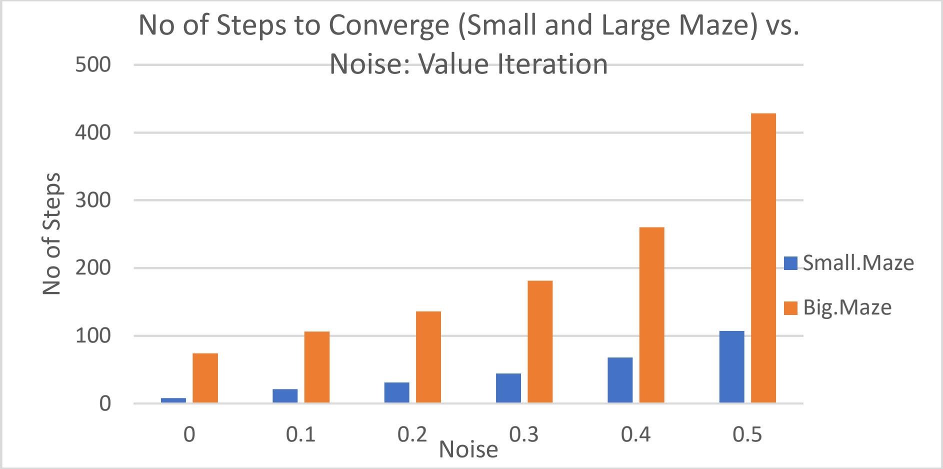
The two main parameters in our value iteration algorithm are the PJOG and Precision parameters. PJOG is a number between 0 and 1 and specifies the level of noise i.e., the probability that the agent will indeed take the specified optimal action from the current state as intended is (1-PJOG). The agent may deviate and take other actions, i.e., it may move in the other 3 directions besides the optimal one with a probability of PJOG/3.

When the value of PJOG is very small, for both value and policy iterations, it leads to the algorithm moving successfully towards the optimal policy without much stochastic deviation (and noise). Shown below are the optimal value iterations policy when PJOG = 0. We can see that the policies presented in all the states successfully point towards the optimum for both mazes. As the noise (PJOG) becomes bigger and bigger, there is a greater chance that the algorithm will stray away from the optimal policy path, and the longer time it takes to run to finally reach this optimal policy.

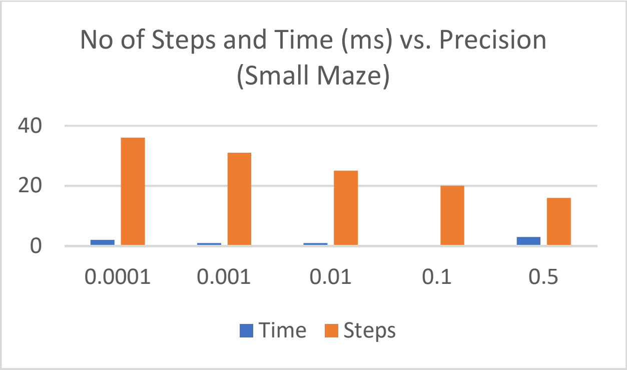


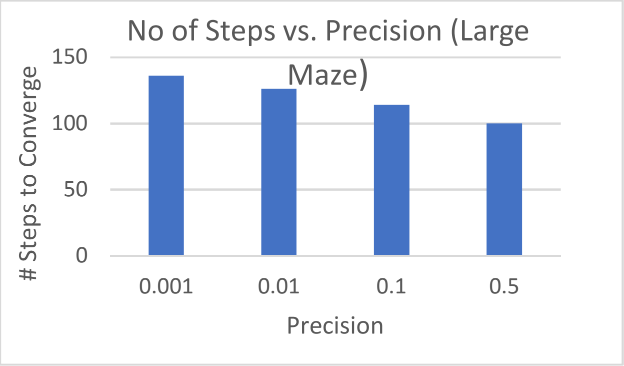


As we keep increasing noise, the number of steps taken to converge increases as shown below, and is a higher number for the big maze than the small one.



Precision, on the other hand, specifies the precision level at which the algorithm converges. As precision level increases, the number of steps taken by the algorithm to converge does down as can be seen in the case of the small maze as well as the large maze through the plot below. The relative drop in of steps with precision is smaller for the large maze because even with high precision, due to its size, it needs a comparatively large absolute number of steps to converge. Unlike number of steps, the time taken by the algorithm to converge, however doesn't seem dependent on precision. However, the time taken to converge for the large maze is so much larger (3k – 17k ms), simply because the MDP has to iterate through a much larger number of states before reaching the optimal value policy. Because times for large maze were so high relative to # steps to convergence, I've only plotted the latter for the large maze below.

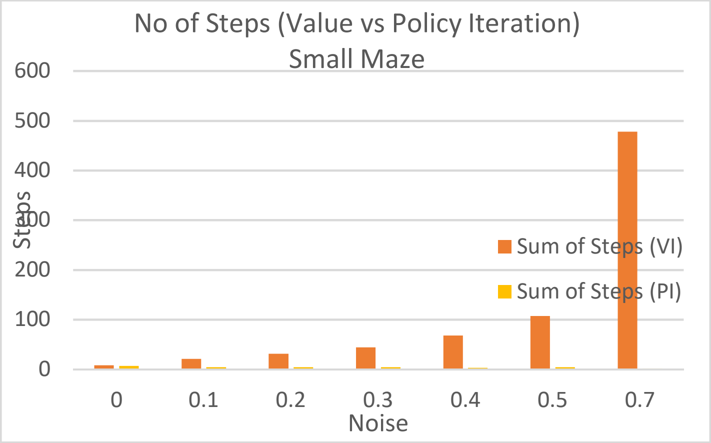


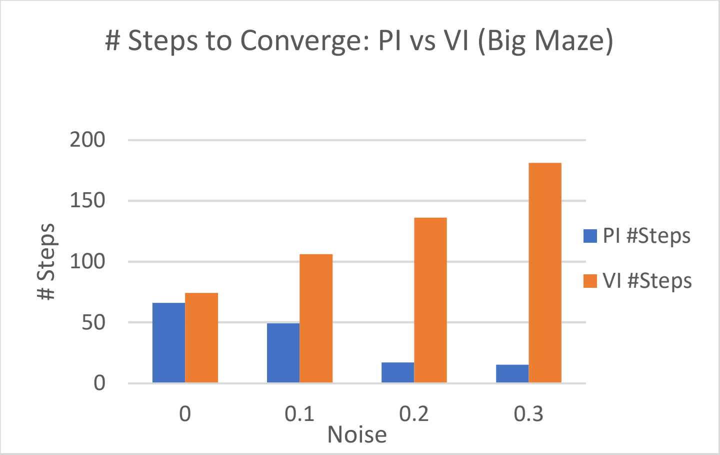


Policy Iteration: A policy is a mapping from states to actions π : S → A (can also define stochastic policies). In policy iteration, we repeat policy evaluation and improvement repeatedly until the algorithm converges.

Policy iterations usually takes far fewer steps than value iteration to converge – as shown in the graph below for the small maze. This could be because the optimal policy only prescribes the direction in which the agent should move (in other words, the optimal action to be taken given a particular state) and converges when there is no further change of policy required. On the other hand, given an optimal policy, value iteration works to iterate enough number of times to maximize the expected present value of future rewards which make take more steps to converge. Overall, the relative performance of these algorithms depend on the ratio of number of actions to the number of states. For policy iteration, higher this ratio better will be performance.

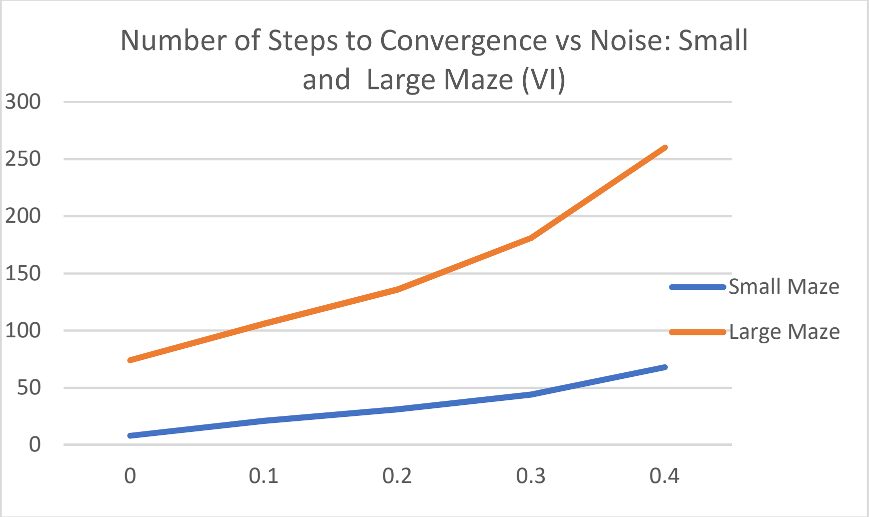
As long as the maximum number of iterations is set as greater than equal to the number of steps the problem will take to converge, the max. iterations parameter does not influence the algorithm's performance much.

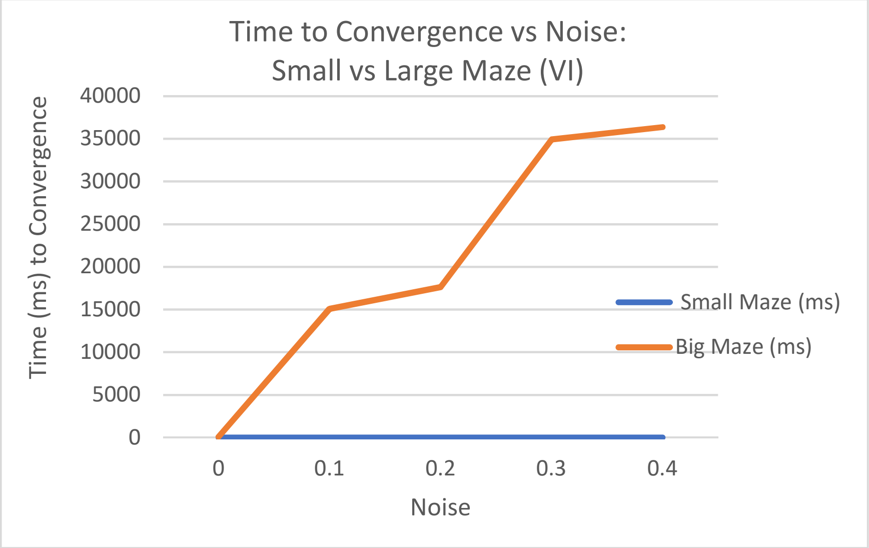




Small Maze vs Large Maze

Since it has a much higher number of states (2025 instead of 18), the algorithms (both MDP and reinforcement learning algorithms) have to go through several more states before reaching the goal, or learning about the environment. As a result of this much-larger search space, the time taken by the value and policy iterations to converge to an optimal solution under the MDPs is much, much higher for the big maze as compared the small one. The number of steps taken by value or policy iterations to converge is also much higher for the large maze, however, the increase in the number of steps with respect to that of the small maze is much smaller than the increase in time taken for convergence in the big maze.





The time taken for convergence especially increases in the big maze as the noise level increases, as shown below. This could be because, when the PJOG parameter has a finite value, there is a positive probability that the agent will stray from the intended path in moving towards the optimum, at every time step. In a much bigger maze, there is a much higher probability therefore, that the agent strays away in multiple steps during the policy, thus moving far away from the optimum, which will then require it to spend a lot of time in trying to come back on track, given the consistent noise. In fact for the big maze, for noise levels higher than PJOG – 0.6, the program doesn't converge given the computational memory and time limitations. It kept running for a very, very long time and I had to terminate the program. Because there are so many states, a high probability of straying away can easily put the agent very far off from the goal, and repeatedly do so, making it hard even for MDPs to direct it towards the goal. This time increase for the big maze is shown in the plot above.

Q learning: Q-learning works by estimating the value of state-action pair, called Q value, which, in our mazes, is the expected sum of future reinforcement (penalty) obtained by taking an action from a state and following an optimal policy thereafter. Once these values have been learned, the optimal action from a state is the one with the minimal Q-value. For every new observation we update Q-value using the Learning rate (between 0 and 1) which determines how much current Q-value is changed.

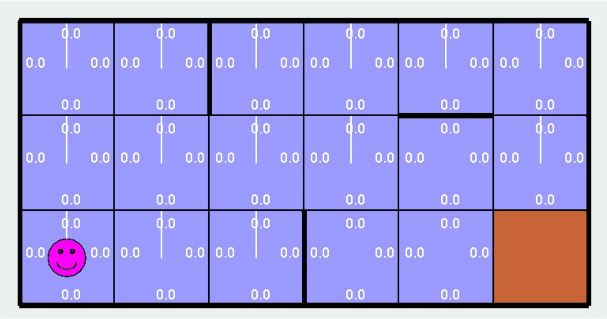
In noisy environment, high learning rate would not allow the system to stabilize and low learning rate makes learning very slow. We use a decaying learning rate to solve this conundrum. In our Q learning implementation, we take E (Epsilon) as the probability that the algorithm chooses a random action in a given state (instead of the optimal action) i.e., that it explores. The learning rate decays exponentially in the algorithms I have run. Also, artificially high virtual rewards are given for states and actions that haven't been explored for a long time.

We see that Q learning is easily outperformed by Value and policy iteration in the cases of both the small and the large maze because unlike Value/Policy iterations which are Markov decision processes with perfect information about their environment, Q learning has to using exploration and exploitation techniques to iteratively learn about its environment. Through exploration, the algorithm takes actions that are necessarily not the best, but help it to maximize knowledge gain about the unknown portions of its environment. Though exploration may waste computing time by making suboptimal moves, it helps discover effective behaviors that may help the algorithm converge to the optimal solution faster. In combination with exploitation, where the algorithm taps into the knowledge it has already gained to make decisions, exploration techniques can optimize the performance of the Q learning algorithm.

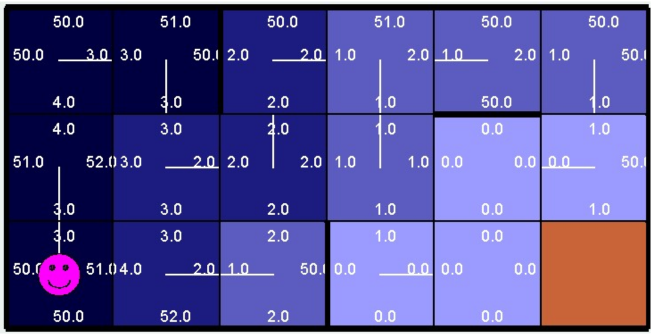
According to Q-exploration techniques, the agent interacts with the environment with a very high learning rate initially so that it can quickly settle down to a near optimal policy. As the agents interactions with the environment increase the learning rate decays and the agent now makes only minor modifications to its near optimal policy to obtain the optimal policy. This scheme is guaranteed to asymptotically converge to the optimal policy.

As the initial learning rate increases, it takes longer and longer to run the Q learning algorithm. This could be because a higher learning rate encourages the algorithm to explore new areas, often making wasteful moves that aren't directed towards reaching the optimal solution. This causes the algorithm to lose a lot of time in exploration instead of quickly and directly moving by the optimal policy.

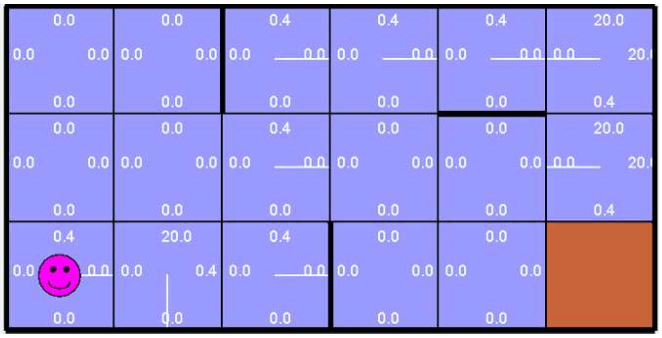
At very low Q values the algorithm doesn't learn anything or explore. As a result, it runs for a very long time, moving randomly without any prior knowledge of states or rewards. It ends at some arbitrary state but all states remain un-updated, no better than when the algorithm started running. The policy suggested also points away from the goal. This is shown below:



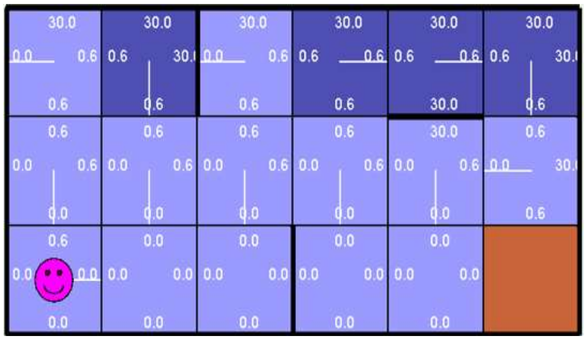
When we set the learning rate to a very high value, like 0.9 or 1 , the algorithms take a very long time to run and instead of converging, end up at a very suboptimal policy with high penalty scores, as shown below with the small maze, for LR = 1.0. Moreover the optimal policy in the sink state points away from the goal.



Therefore, it is between these two extremes of learning parameter values that we find the optimal result – with the Q learning algorithm pointing accurately towards the goal at the end of the algorithm run, as shown below, for a learning parameter value = 0.4.

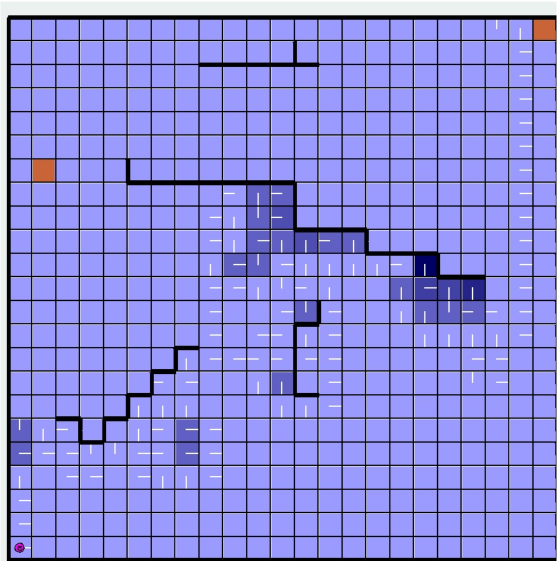


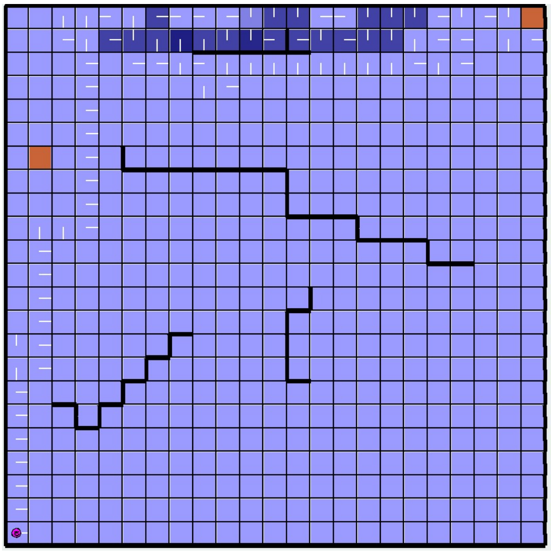
With regards to the PJOG parameter, the greater its value, the more the probability that the agent will wander to directions not intended for exploration or movement based on exploitation – therefore, with increasing noise, or PJOG, the running time increases and the likelihood of finding the most optimal policy decreases. When PJOG is 0 or 0.1, there is high confidence that the agent will move as intended. This results (as shown below) in an optimal policy with minimal penalty (zero) and optimal policy pointing towards the goal.



With the large maze, the Q learning algorithm takes a very long time and does not converge or do a very good job. To run all the algorithms and re-test performances on the big maze, I chose to shrink the size of the maze to 23*23 so that it now has 529 squares instead of the previous 45*45 maze.

Even when we run the Q learning algorithm (As shown below) on the big maze, the results are unsatisfactory. With a learning rate of 0.7 and noise of 0.3, the algorithm terminates at a sub-optimal state. The policies near the goal are suboptimal as well. The values (at nearly every state) is 0.0 meaning RL suggests that given how little it understands regarding reaching the goal state, it minimizes penalty to simply stay put. We can conclude that Q learning algorithm is more efficient with smaller MDPs with fewer number of states.



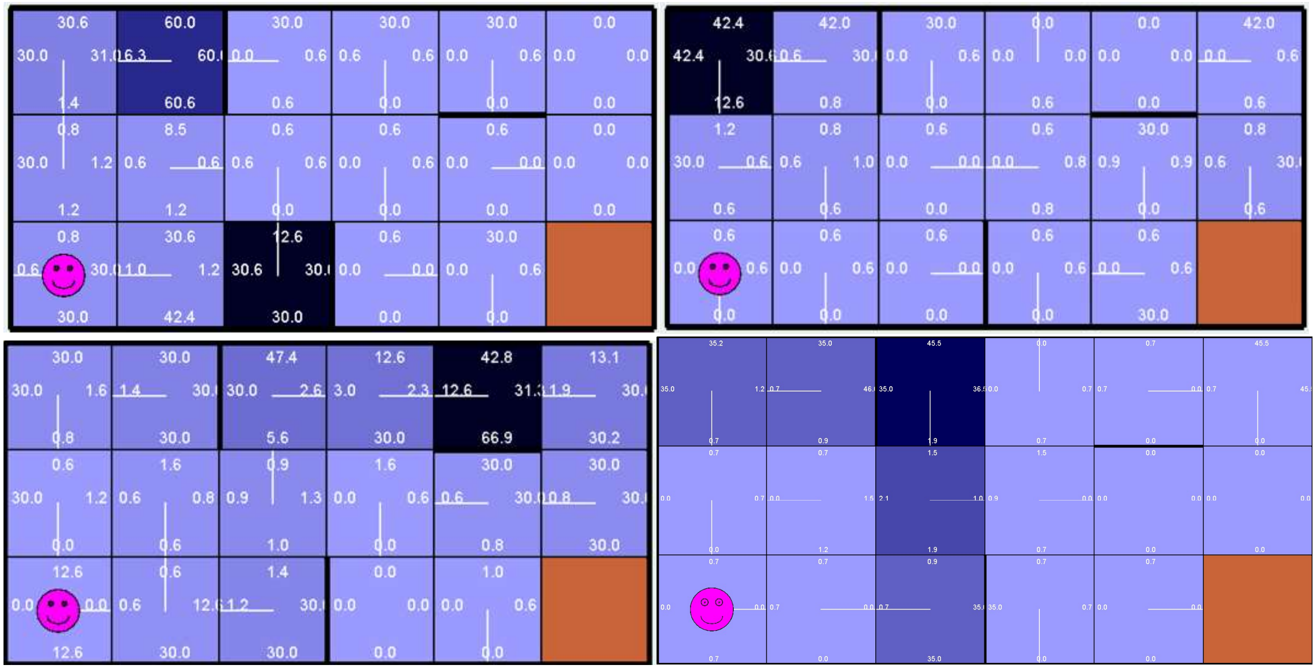


(a) with noise = 0.3 and (b) with noise = 0.1. More of the exploration has accurately moved towards the optimum (goal) when noise is low and intended actions based on exploitation can be actioned.

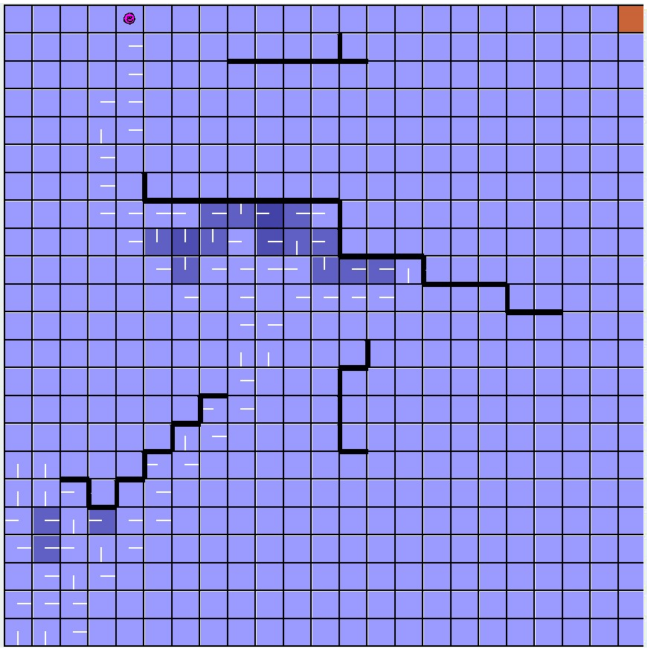
Exploration Strategies Used: Some of exploration policies used and varied for this experiment were:

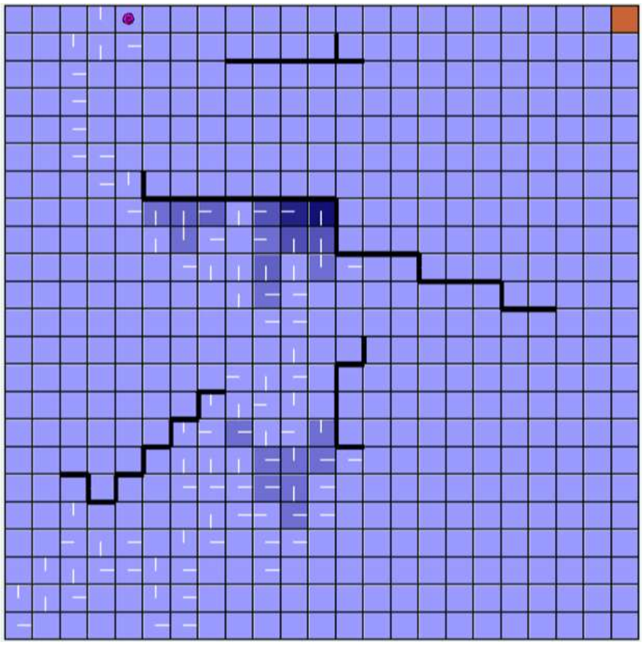
- E-greedy (Epsilon Greedy strategy): Select best action with probability 1 − E and choose random action with probability E.
- Changing start state: Once goal is reached, reset to random state for next iteration.
- Time-worn virtual reward: Artificially give high virtual rewards for state and action which have not been explored for a long time.

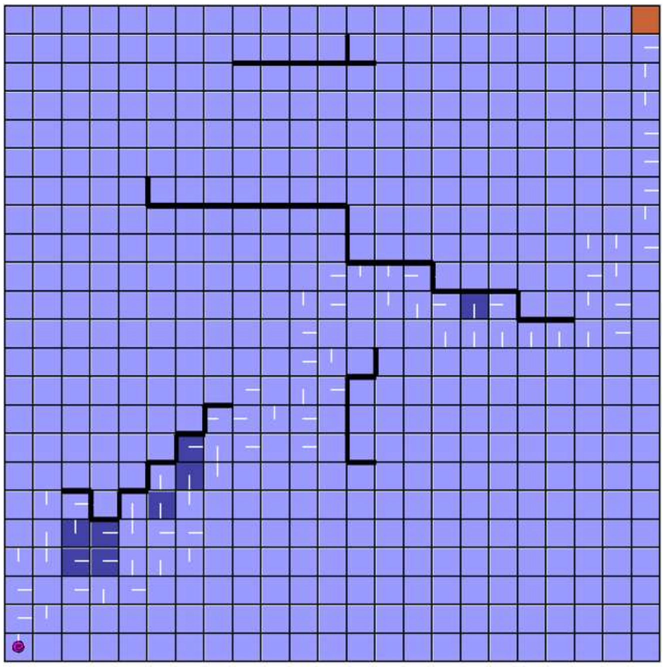
Small Maze: Below we can see the results of changing the Epsilon from 0.1, 0.3, 0.4, 0.6 . The results get better (i.e., at the end of the run, the penalties get lower and lower and the policy slowly move towards the goal when E=.4). When E becomes higher the algorithm is able to choose random new actions (explore) instead of being stuck to limited prior knowledge (gained through exploitation). This allows it to break out of locally optimum policies and seek better results.

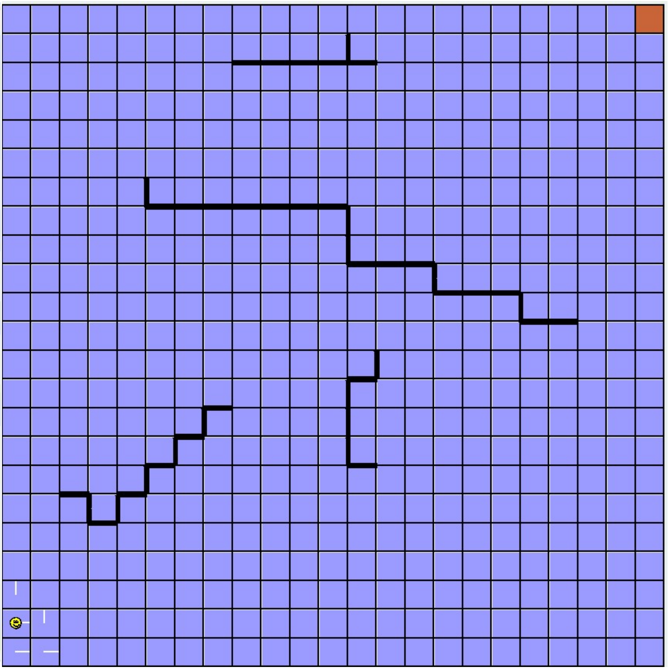


Big Maze: For the big maze, once again, I varied the Epsilon values and observed the resulting policies produced by Q-learning. Below are the results for epsilons 0, .2, .6 and .8. We observe that the optimal policy is much closer to the desired goal with a higher E (exploration) value. At lower values of E, the algorithm terminates when its going towards, but still far from the goal – a suboptimal policy. At a very high value, the algorithm terminates very far from the optimal solution because it is unable to use much known information (exploit) at all (as shown in the 4th figure). The optimal exploration technique is for the algorithm to use an epsilon of 0.6 – not too low, or too high.









In general, Q learning doesn't seem to work as well for larger mazes as compared to smaller mazes, primarily for 2 reasons. One is that the computing power and memory becomes a limiting factor to thoroughly perform reinforcement learning on large grid problems. Secondly, the number of states is so high that there are many possible combinations of actions defining an exponential number of policies of which only one is the optimal policy works towards the goal. Without the kind of certainty we have with MDPs, reaching exactly such an optimal policy becomes very improbable and challenging.

Conclusion

In this project, I have learned about understanding the performance of Markov Decision Processes through implementing Value and Policy iterations on grid problems of various sizes. By applying the Q-exploration Reinforcement learning technique, and different exploration strategies, I've also understood the comparative performance of an algorithm that uses a combination of exploration (of unknown territory in the state space) as well as exploitation (of known knowledge from previous learning) to learn the best possible approach to reach the optimum. Mazes – one small and one large – that have been chosen for this analysis are interesting because many real world situations, games and problems can be modelled as mazes having a penalty in each state, a large reward for the goal and obstacles on the way.

Within MDPs, where the states, rewards and goals are all known, we see that value iteration takes more steps to converge than policy iteration because policy iteration simply looks for the best actions to be taken in each state whereas value iteration also calculates and tries to optimize the value of gains in each state. We also find that as noise increases the number of steps taken to converge also increase since noise often leads the algorithm to deviate from the optimal route.

As precision increases, the number of steps taken by the algorithm to converge decreases for both the small and the big maze. This is because precision determines the algorithm of the algorithm as it moves towards the optimal policy. Time taken by the MDPs to converge is generally multitudes higher for the large maze as compared to the small one.

With regards to Q exploration, we find that the time taken for it to run on the large maze is extremely high and so is the memory required. This is because as a reinforcement learning technique, in each cycle, Q exploration has to use a combination of exploration and exploitation to reach the optimal result. We find that increasing the learning rate too much or keeping it too little both worsen the performance of the algorithm; we need a moderate rate. With lower noise (PJOG), we can again ensure that there's a higher probability that the policy suggested by Q exploration will be closer to optimum.

Several exploration strategies were used in this experiment. We tweaked the proportion of exploration to exploitation and found that for the best results, we have to use a moderate blend of both techniques (medium E value). Here again, we observe an a moderate Epsilon is optimal for reinforcement learning.

Overall, we can conclude that the size of the Markov decision problem, the algorithm an the exploration techniques used, and their parameters - all affect how effectively we can model and find optimal policies for the problem.

References

1. Reinforcement learning simulator: https://www.cs.cmu.edu/~awm/rlsim/

2. Reinforcement Learning Lecture: https://www.cc.gatech.edu/~bboots3/CS4641-Fall2018/Lecture20/20_RL1.pdf

3. https://www.quora.com/What-is-the-precise-connection-between-Reinforcement-Learning-and-Markov-Decision-Processes-MDP