Joseph Cantrell

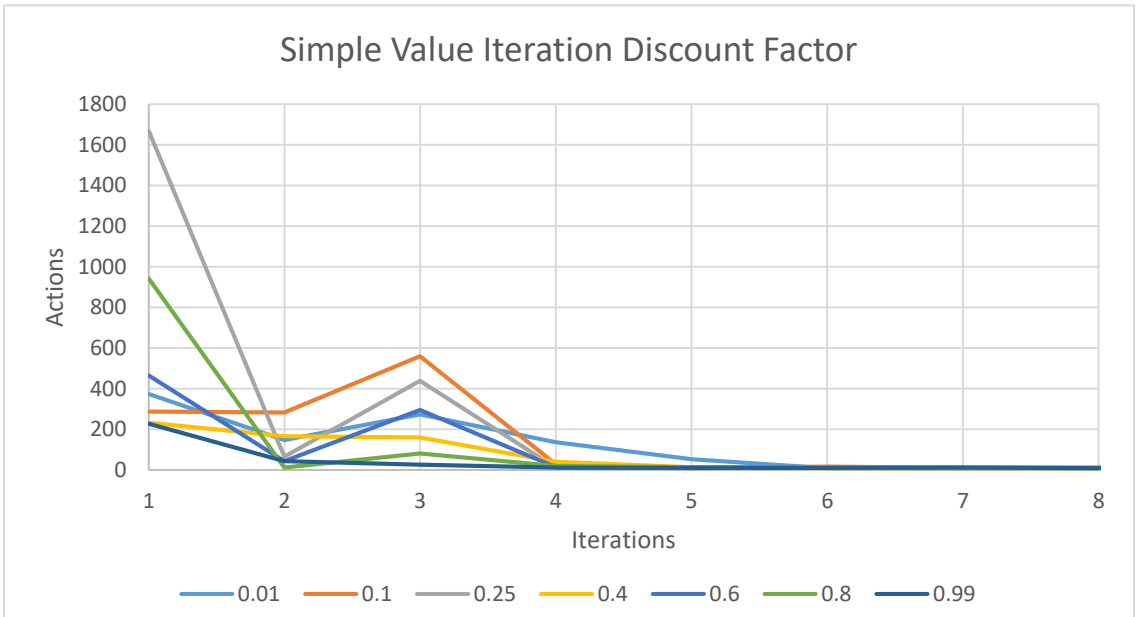
CS 4641

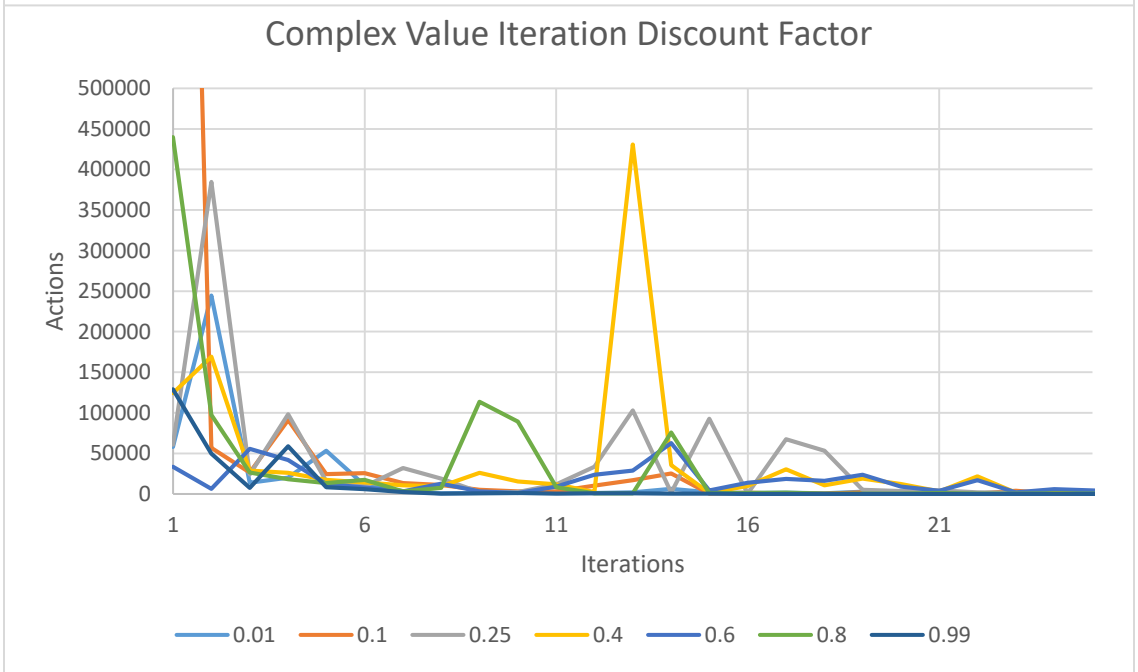
Markov Decision Process Algorithm Comparison

This experiment tests the performance of three different algorithms on two different Markov Decision Process problems using the BURLAP Java library. The problems are two mazes differing in size complexity. The first is a five by five grid with sixteen possible states and the second is a twenty-three by twenty-three grid with three hundred fifty-nine states. Each of these states is defined by its grid position. The complex world has over twenty times the number of states as the simple world. In these worlds, an agent must find the optimal route from the start state to the goal state given four possible actions: up, down, left, and right. The grid world is stochastic in that each action the agent attempts to take can result in a random move instead. The chance this random move will occur is marked by an error value called epsilon. Each action the agent chooses yields a reward. These rewards are lessened by a discount factor gamma proportional to the time the action was taken; if an action is chosen farther in the future, it will yield less reward. The optimal solution is a combination of these four actions (a policy) that reaches the goal state while maximizing the total reward the agent receives.

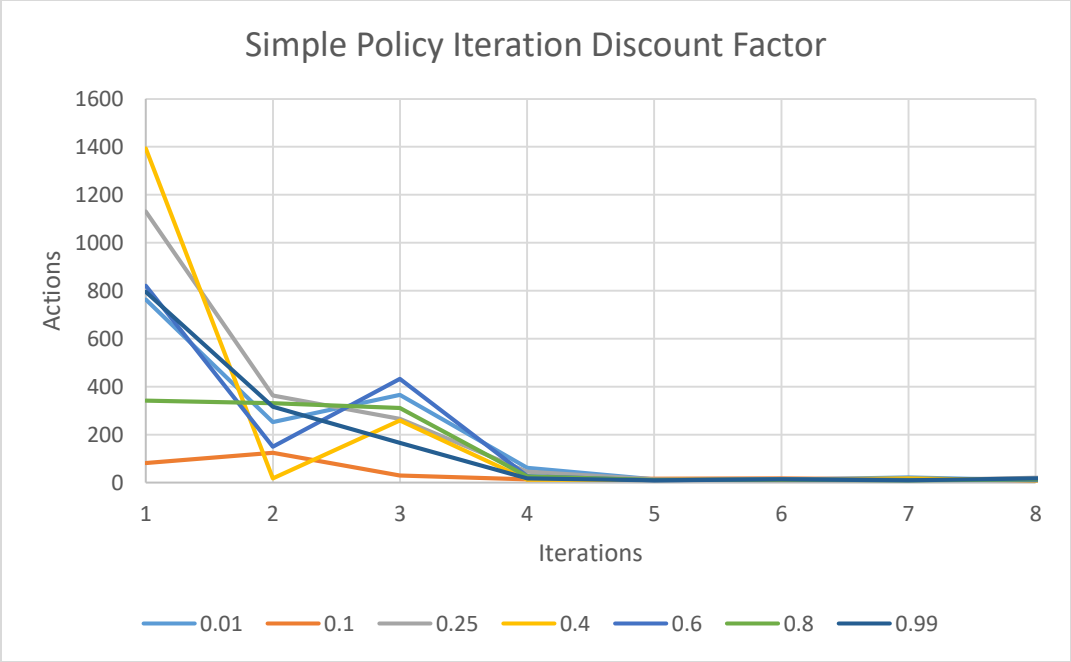
The three algorithms tested are value iteration, policy iteration, and Q Learning. Value iteration looks at each state and calculates a total reward based on the reward of the current state as well as the discounted rewards of that state's neighbors. The algorithm continues until it converges within a certain threshold. The discount factor was varied to test its effect on the convergence of value iteration. The graphs below show the results of this test with the number

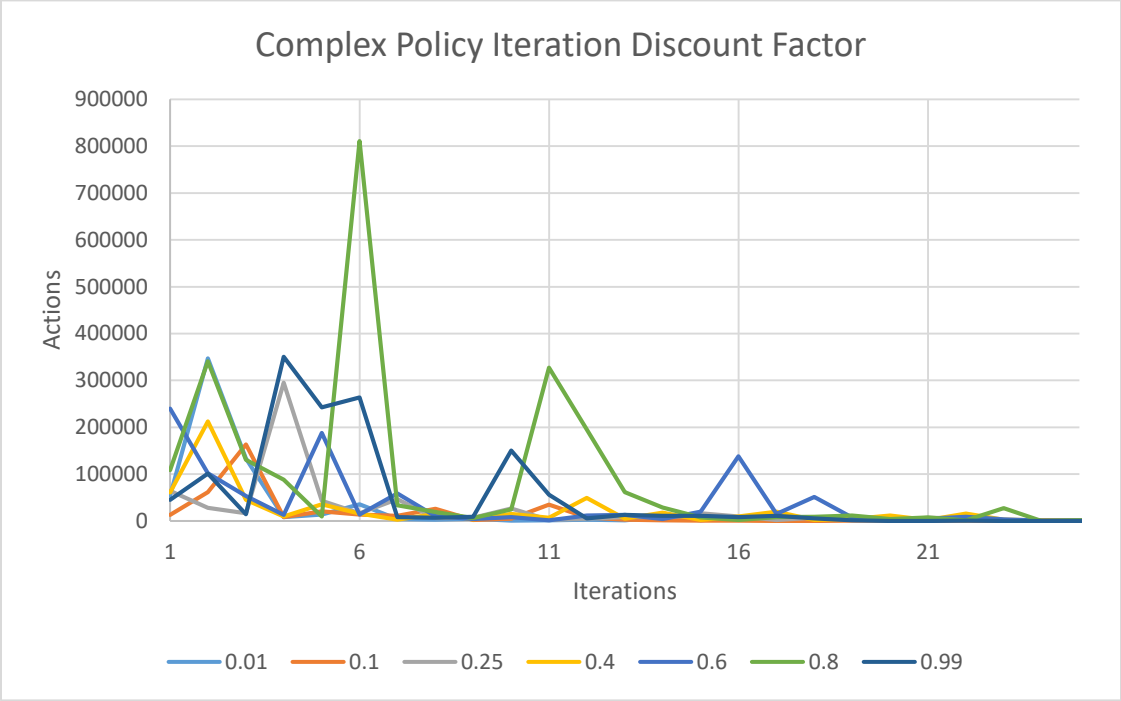
of iterations on the horizontal axis and the number of actions to find a solution on the vertical axis. As shown in the graphs, value iteration tends to converge more quickly with a larger discount factor in the case of both problems. Larger discount factors also tend to give less variance in the amount of actions required to find a solution. .99 gave the best performance of all the discount factors tested. This value is very close to one, so future rewards are not discounted very much.



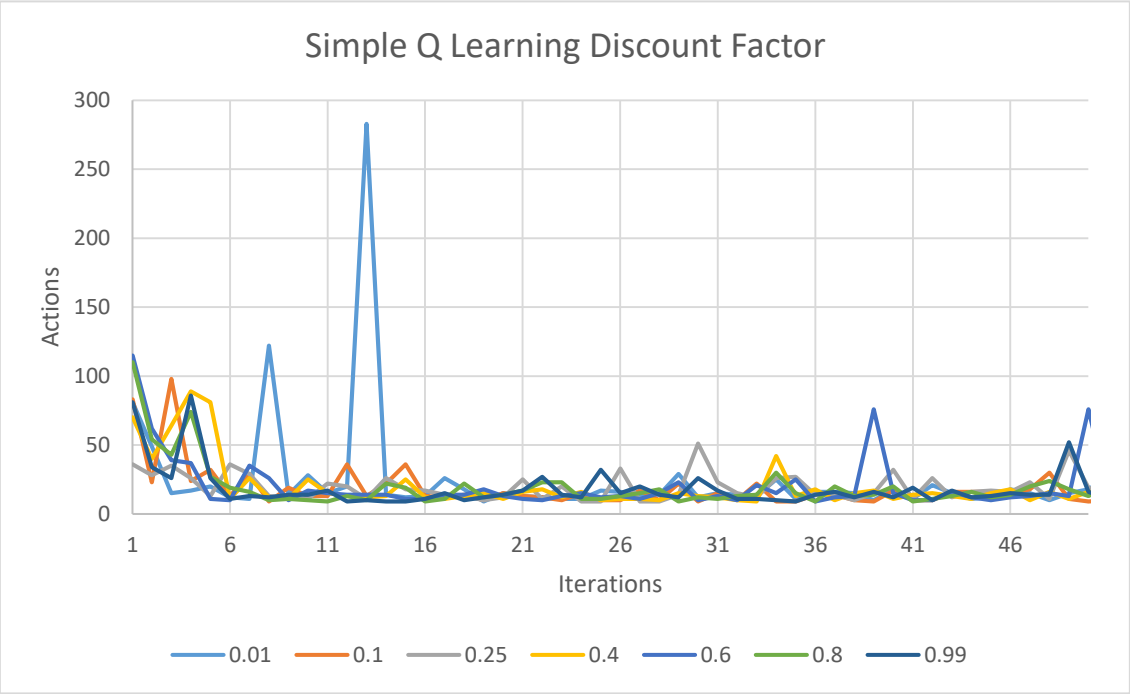


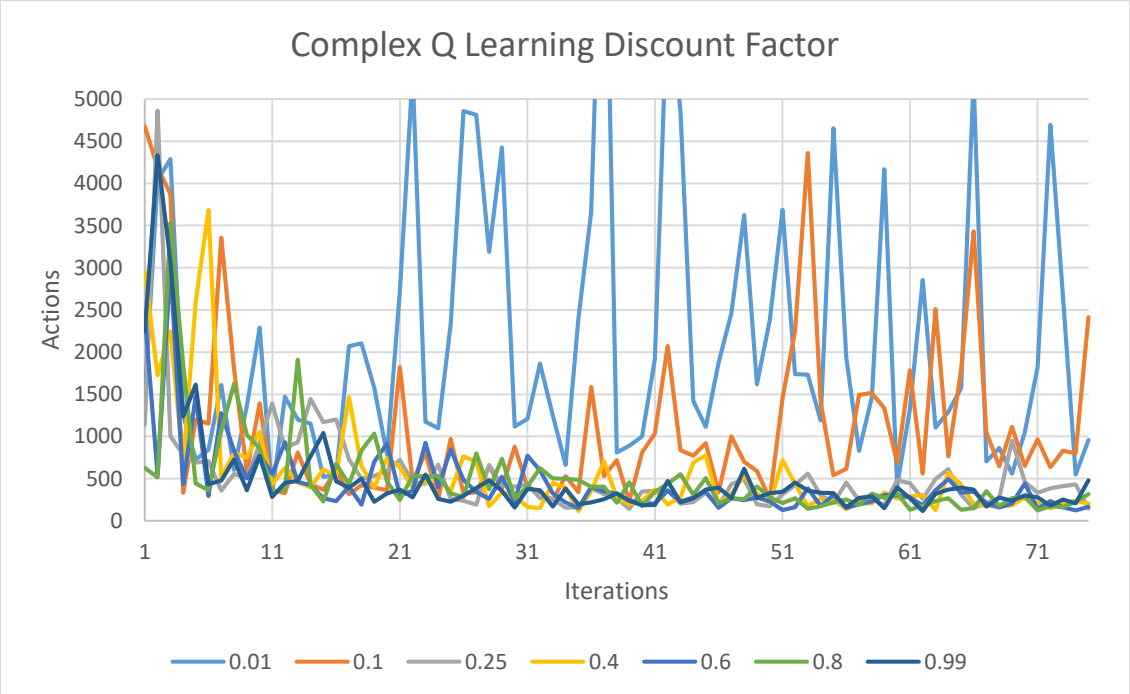
Policy iteration iterates through the search space of policies. It starts with a random policy and changes the true reward of a state based on this policy. With these new rewards, the algorithm picks the optimal action at each state and repeats until no action changes from the given policy. The discount factor was also tested with policy iteration. This did not affect convergence much for the simple problem; each test converged at five iterations. The complex problem showed a great deal more variety between different discount factors. Here, lower discount factors tend to converge more quickly and display less response variance. Because there are many more states in the complex problem, a lower discount factor discounts future bad actions more and prevent the algorithm from choosing longer but less rewarding paths to the goal.



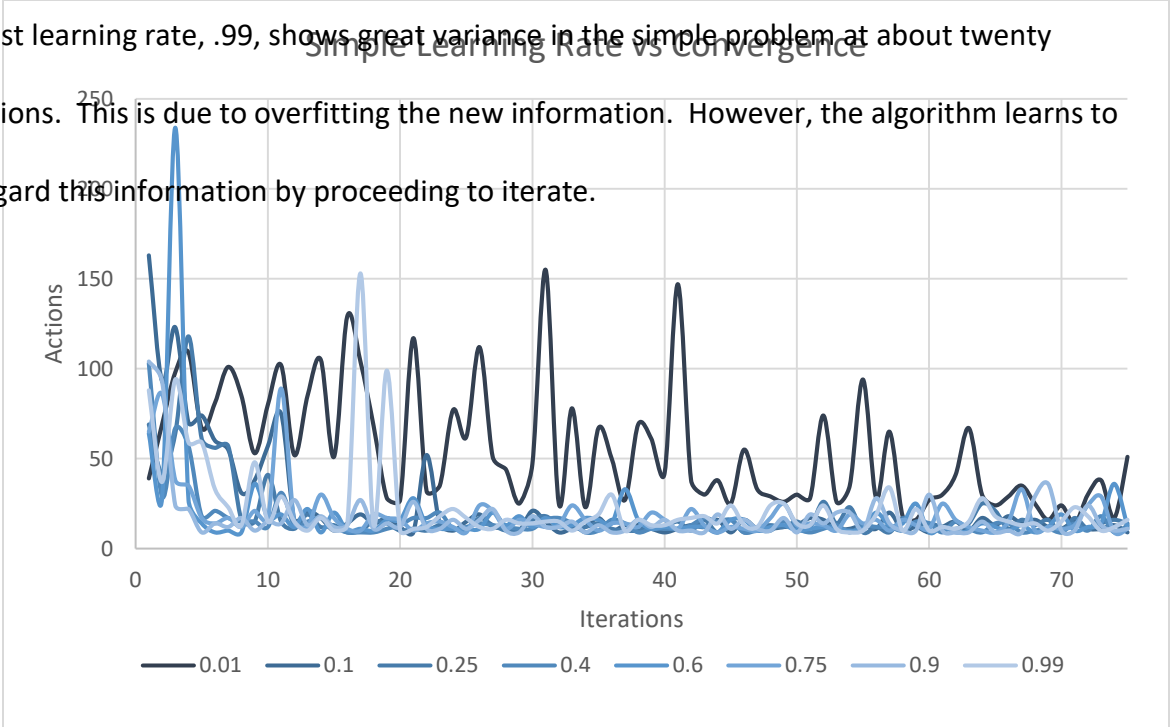


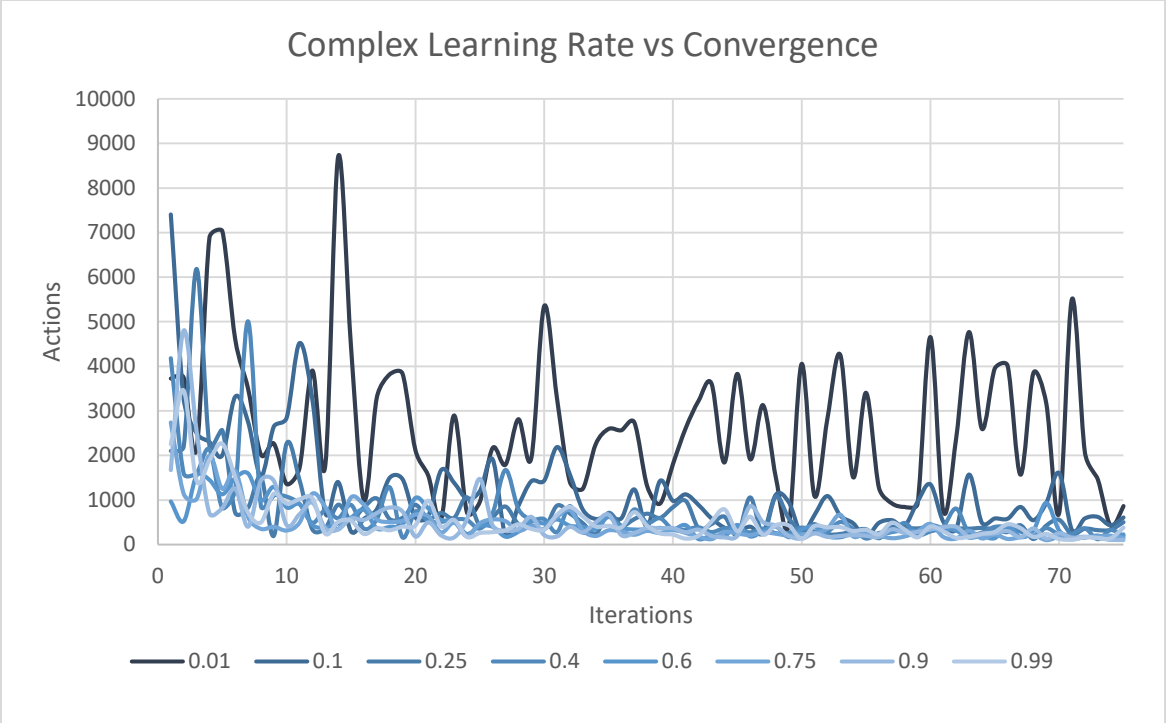
Q Learning is a reinforcement algorithm that learns the optimal action for each state based on its total reward for this action and all discounted future actions. Each action the agent selects will yield a new reward which gives more information about the total reward of previous states. The exploration strategy employs stochastic movement where the agent greedily picks the best move but has a chance to make a random move. The discount factor was varied to find the best value for the fastest algorithm convergence. All discount rates performed similarly for the simple problem. The lowest factor, .01, gave a large amount of random variance. This effect is better seen in the complex problem, where the two lowest discount factors do not cause convergence within seventy-five iterations. The larger discount factors performed better by causing the algorithm to converge more quickly and yielding less response variance.


Simple Q Learning Discount Factor
Actions
300
250
200
150
100
50
0
1
6
11
16
21
26
31
36
41
46
Iterations
0.01
0.1
0.25
0.4
0.6
0.8
0.99

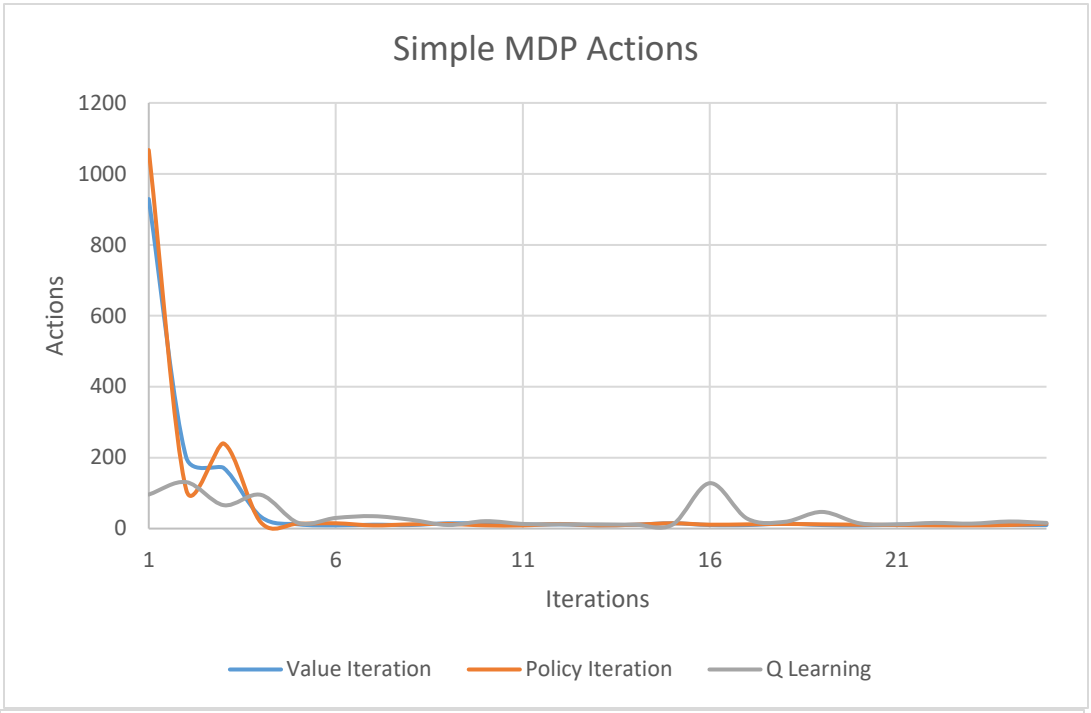

Complex Q Learning Discount Factor
Actions
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1
11
21
31
41
51
61
71
Iterations
0.01
0.1
0.25
0.4
0.6
0.8
0.99

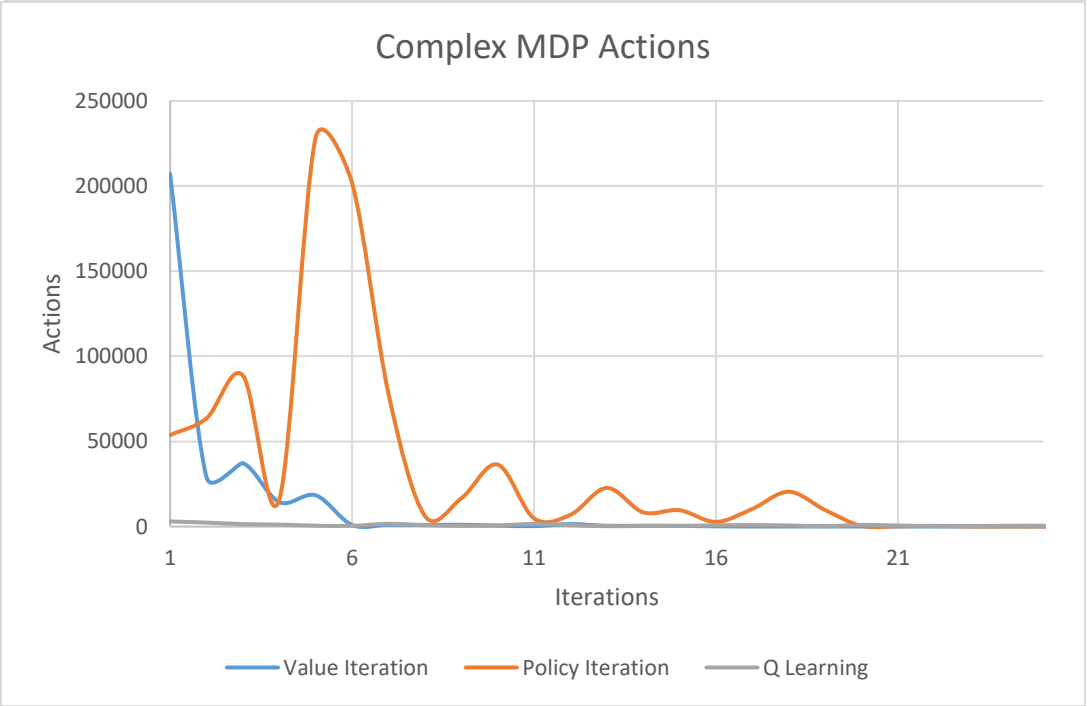
The learning rate determines what impact new information has on the agent's knowledge of total rewards for states. A higher learning rate will cause the agent to give new reward information higher priority than old information, while a lower learning rate will cause the agent to learn less from the newly available information. A higher learning rate caused Q Learning to converge more quickly when solving both the simple and complex problems. The highest learning rate, .99, shows great variance in the simple problem at about twenty iterations. This is due to overfitting the new information. However, the algorithm learns to disregard this information by proceeding to iterate.

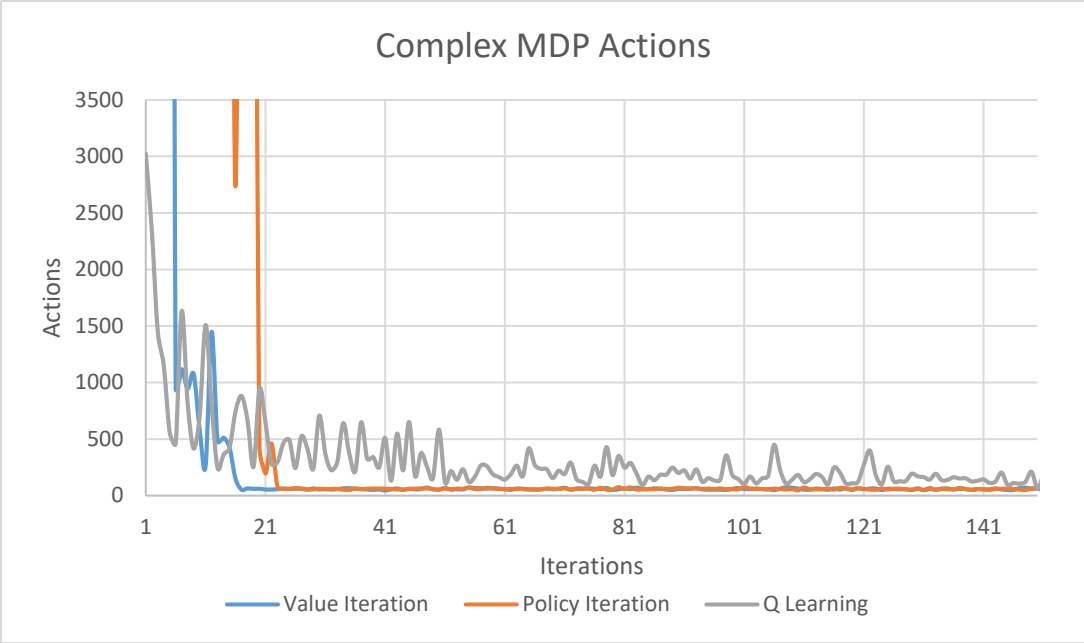




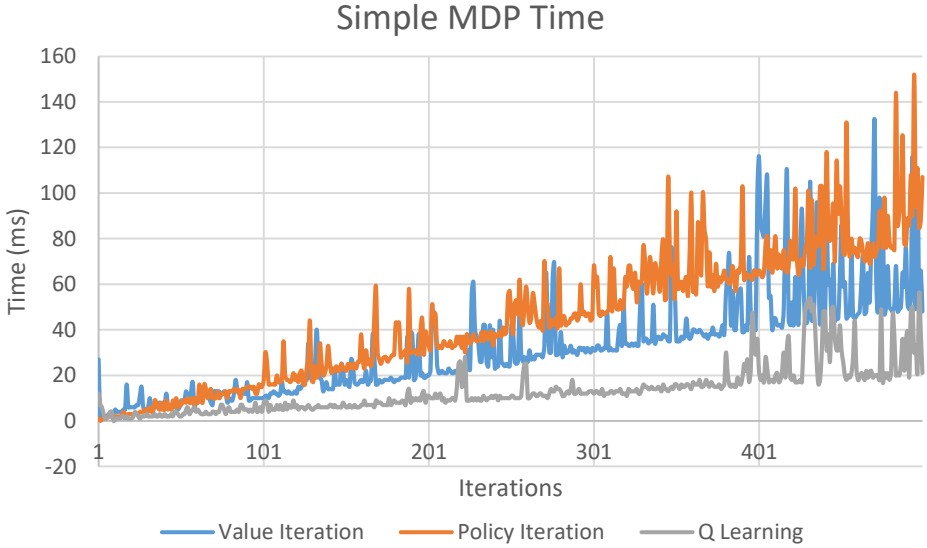
Each algorithm was then tested using a .99 discount factor, and Q Learning was tested using the best learning rate found, .99. For the simple problem, value and policy iteration converged at about five iterations. Q Learning showed a large variation at iteration sixteen but converged completely at twenty iterations. For the complex problem, value iteration converged fastest at six iterations. Policy iteration converged with twenty iterations, over three times the amount taken by value iteration. Q Learning took fifty iterations to converge, over eight times the amount needed by value iteration. However, Q Learning began closer to the optimal solution than value and policy iteration.

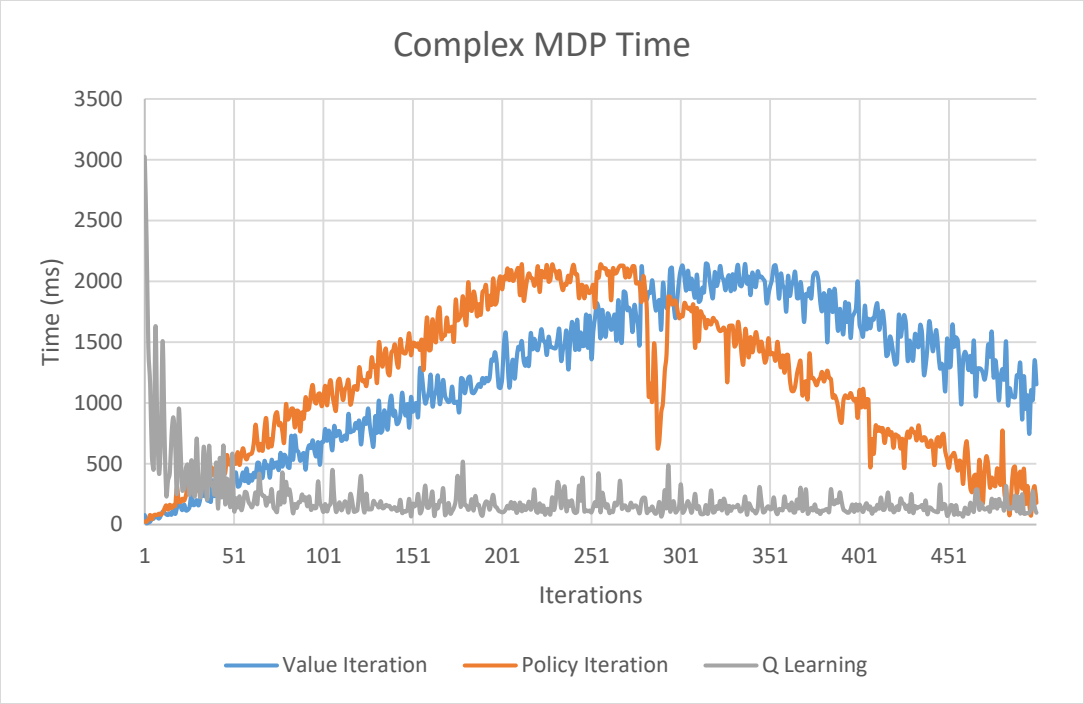

Simple MDP Actions
1200
1000
800
600
400
200
0
Actions
1
6
11
16
21
Iterations
Value Iteration
Policy Iteration
Q Learning


Complex MDP Actions
250000
200000
150000
100000
50000
0
Actions
1
6
11
16
21
Iterations
Value Iteration
Policy Iteration
Q Learning



For the simple problem, each algorithm took progressively longer to finish an iteration. On average, the time taken for each algorithm increased linearly. The complex problem showed more variation in algorithm time complexity. Value iteration and policy iteration started with low time but began to consistently need a large amount of time per iteration for many iterations. Value iteration began to need less time at three hundred iterations, but policy iteration needed three hundred iterations before its time per iteration decreased. Q Learning needed the least time for both problems, and its time needed was consistently decreasing on average.





This experiment showed that each algorithm has its uses. For simple problems, value iteration or policy iteration will converge much more quickly than Q Learning. However, for more complex problems, Q Learning will find the optimum solution much more quickly than value iteration or policy iteration because each iteration in Q Learning takes much less time. For both problems, value iteration converges with less iterations than policy iteration and Q Learning. However, for very complex problems needing a large amount of learning iterations, Q Learning can give a solution using much less time than value iteration and policy iteration.