Assignment 4:
Reinforcement Learning: Markov Decision Processes

Xueyou Hu
huxy99@gatech.edu

Abstract
In this assignment, I tested two model-based planning algorithms (value iteration and policy iteration) and one model-free learning algorithm (Q-learning). To demonstrate the different properties of these algorithms, I randomly created two grid worlds MDPs and solve them using these three algorithms separately. The experiments showed that policy iteration has better performance with small size MDP problem whereas value iteration converge much faster than policy iteration when it comes to large MDP problem. Compared to value iteration or policy iteration, the experiments demonstrated that Q-learning takes much longer time to find a solution for both small size and large size MDPs.

1. Introduction

We have learnt reinforcement learning and Markov Decision Process (MDP). MDP provides a mathematical framework for modeling decision making in situations where outcomes are partially random and partially under the control of a decision maker. MDPs are used in a wide range of disciplines including robotics, economics, and manufacturing (WIKIPEDIA). MDPs can be solved by linear programming and dynamic programming.

The purpose of this assignment is to explore some algorithms for MDPs. We are required to choose two interesting MDPs and solve them with different algorithms. One MDP has "small" number of states while the other MDP has "larger" number of states. To keep things simple, I created two maze games for study. One maze game has 7x7 grids with less states and the other maze has 25x25 grids with much more states.

The required algorithms are value iteration and policy iteration. For the freely choosing algorithm, I pick my favorite reinforcement learning algorithm Q-learning. All these algorithms are provided by the Brown-UMBC Reinforcement Learning And Planning (BURLAP) java library (http://burlap.cs.brown.edu/index.html).

1.1 The Problems

In the field of MDP, the canonical example is Grid World. The agent lives in a grid world and the walls block the agent's path. The agent gets penalty for each movement/action and the reward

comes at the end. The agent's actions do not always go as planned direction. The grid-world MDPs have many properties that make them interesting and useful for this assignment.

- The grid-world MDPs have limited states and are easy to check if the solution is optimal.
- The process is easy to be visualized, which helps to understand the algorithms.
- It can extend the problem size easily which helps to compare the algorithms.
- The grid-world MDPs represent many real world problems such as Cleaning Robot and Shortest Path Problems.

In Figure 1, I created two grid-world MDPs for this assignment. The circle in cyan color represents the agent at the start point; the black grids represent the walls; the grid in blue represents the terminal location.

1) **Maze Game 7x7**: this maze has 7x7 grids and it randomly produces 30% wall grids as shown in Figure 1 (Left).
2) **Maze Game 25x25**: this maze has 25x25 grids and it randomly produces 15% wall grids as shown in Figure 1 (Right).

Both MDPs have an initiate state at the left-bottom corner while have a terminal state at the top-right corner. The penalty for each step is set as -1 and the terminal location will be given a positive reward 100. For 80% of the time, the agent goes with the aimed direction while there is 20% possibility with other directions.

1.2 The Methods

Value Iteration – also called backward induction (Bellman, 1957), which is a method of computing an optimal MDP policy and its value. Value iteration starts at an arbitrary V0 and then works backward, refining and updating an estimate of either Q* or V* (the expected sum of accumulated rewards).

Policy Iteration – in policy iteration (Howard, 1960), it starts with an arbitrary policy $\pi 0$ and iteratively improves it. There are two critical components in policy iteration: policy evaluation which determines $V^{\pi}_{i}(s)$ and policy improvement which chooses $\pi_{i+1}(s) = \text{argmax}_a Q^{\pi}_{i}(s, a)$. The algorithm stops if there is no change in the policy.

Q-Learning – a model-free reinforcement learning algorithm. Q-learning can find an optimal policy for any given finite MDP. It works by learning an action-value function that ultimately ends up with the expected utility of taking a given action in a given state and following the optimal policy thereafter (WIKIPEDIA).

1.3 Software and packages

The Brown-UMBC Reinforcement Learning And Planning (BURLAP) java library was used in this study. BURLAP java code library is for the use and development of single or multi-agent planning

and learning algorithms (http://burlap.cs.brown.edu/). I download the BURLAP library and developed the codes for this study using IDE Eclipse.

3. Results and Discussion

3.1 Maze Game

As required by our assignment 4, I randomly produced two maze games with different sizes. The maze on the left has 7x7 grids which has 42 states; while the maze on the right has 25x25 grids which has 543 states. With these settings, we can start to analyze these two MDPs using different algorithms that we mentioned in the introduction.

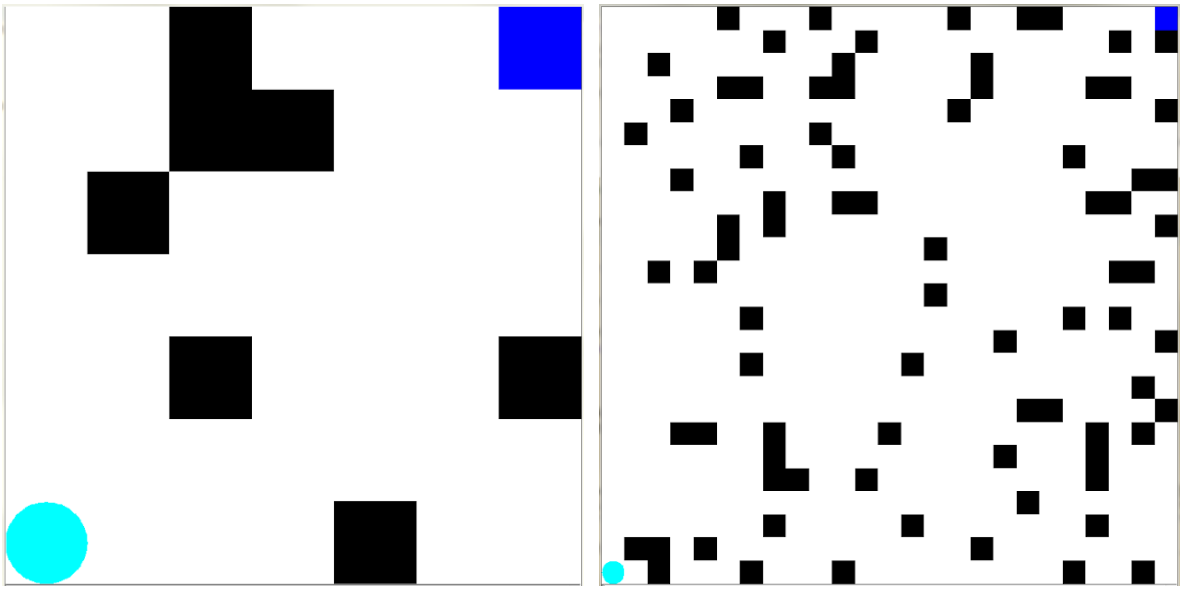

Figure 1. Left: Maze Game 7x7 with 30% walls; Right: Maze Game 25x25 with 15% walls

3.2 Solve MDPs: Value Iteration and Policy Iteration

Firstly I solve the two MDPs using value iteration and policy iteration. The results are shown in Table 1 and Figures 2 and 3. As we can see from Table 1, the performances of these two algorithms are quite different. The algorithm value iteration takes more iterations to converge both for 7x7 and 25x25 maze (25 and 51 vs 4 and 8). While both value iteration and policy iteration take same amount of time for 7x7 maze game (0.344s vs 0.390s), policy iteration takes much longer time to converge than value iteration does (13.214s vs 4.371s).

In spite of these different performances, these two algorithms converge similarly as shown in Figure 2 and Figure 3. We can see that both the algorithms come up with the very similar optimal policy even though the optimal values are little different from grids to grids.

Value iteration is a method of computing an optimal MDP policy and its value. Policy iteration starts with a policy and iteratively improves it. Value iteration calculates all the values for all the possible policies of a certain state and search for the optimal policy. Policy iteration only calculates the values for a certain policy and improves it. Thus policy iteration can be faster than value iteration in this case with small size problem.

When having large number of states (larger problems), however, policy iteration can take much longer time compared to value iteration because it needs to solve the large linear equations for the optimal policy. The maze game 25x25 is a good example of this situation.

Table 1. Value Iteration VS Policy Iteration

Algorithm	MDP	Gamma	Penalty	Reward	Iterations	Time (s)
Value	7 x 7	0.99	-1	100	25	0.344
	25x25	0.99	-1	100	51	4.371
Policy	7 x 7	0.99	-1	100	4	0.390
	25x25	0.99	-1	100	8	13.214

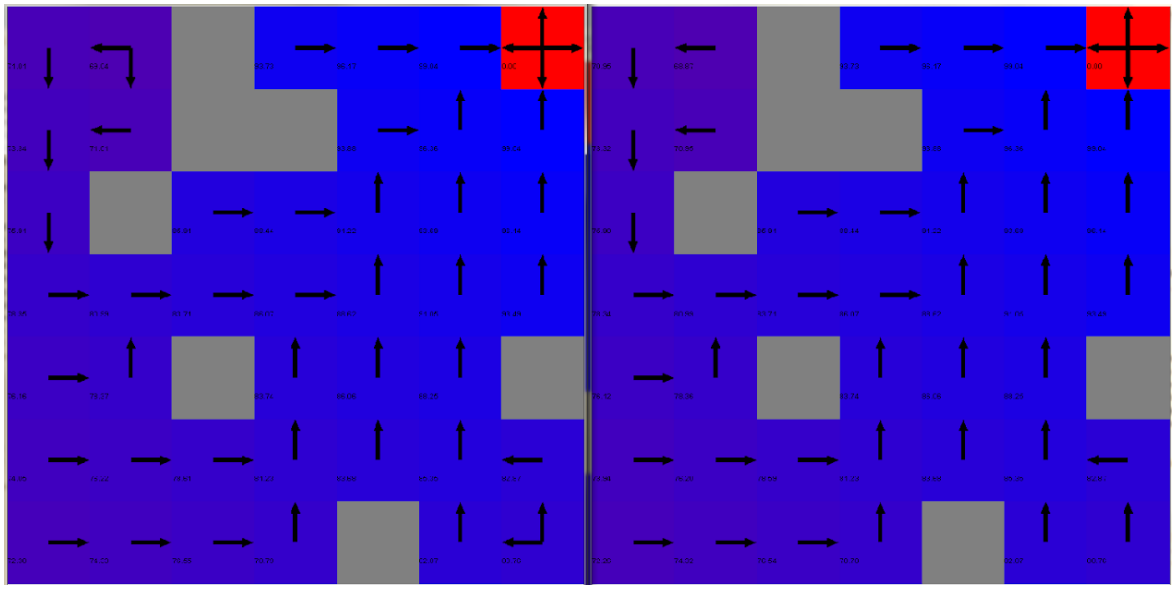

Figure 2. Optimal values and policies for 7x7 maze game
(Left: value iteration; Right: policy iteration)

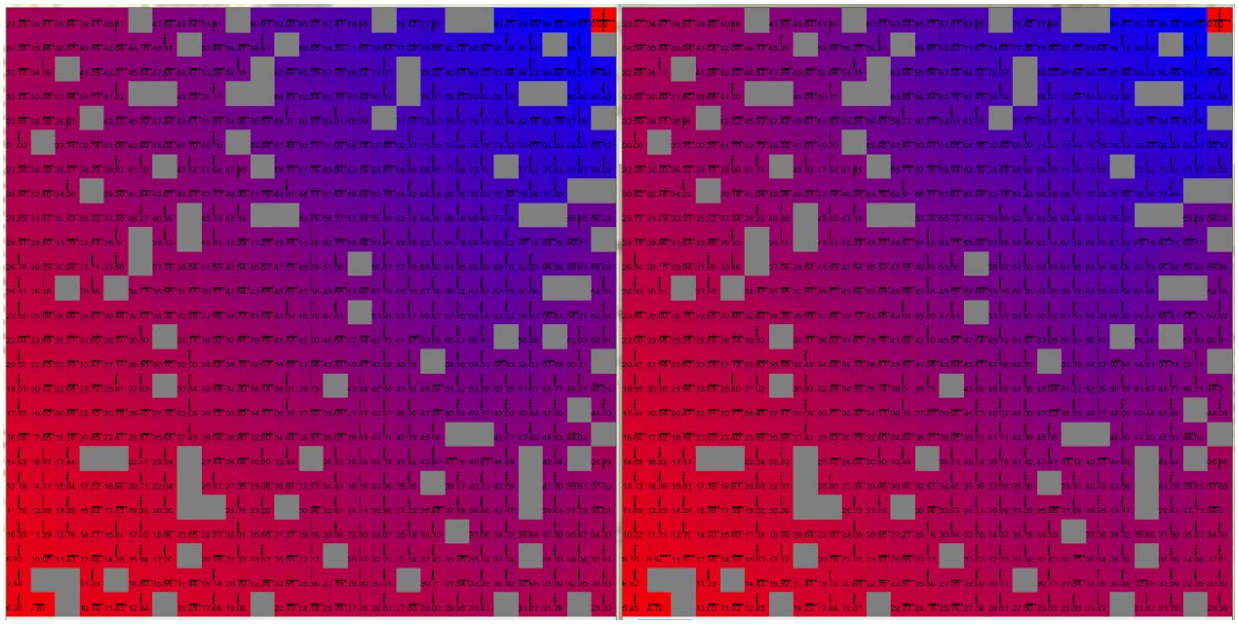

Figure 3. Optimal values and policies for 25x25 maze game
(Left: value iteration; Right: policy iteration)

3.3 The Effect of the Number of States

To explore the effect of the number of states, I studied the iterations and time changes while changing the number of states by changing the percentage of the walls in the maze game 25x25. As we can see from the Table 2, when we reduce the percentage of the walls the number of states decreases whereas the iterations increases for value iteration algorithm. However, it seems that the time was not affected by the number of states. One the other hand, the number of states has no effects on iterations and time for policy iteration algorithms.

I repeated this experiment for five times and the data presented here are the averages of these experiments. The data is also attached to the submission in an Excel file (state-vs-time.xlsx).

3.4 The Effect of Discount Factor Gamma

Besides the effect of number of states, I also explored the effect of discount factor gamma. The discount factor gamma determines the importance of future rewards. A factor of 0 will make the agent only consider the current rewards; whereas a factor close to 1 will make the agent strive for the long-term rewards. I read some online materials which said that gamma can affect the behaviors of value iteration and policy iteration.

Table 2. The effect of the number of states to the iterations and time

Algorithm	Walls	States	Gamma	Penalty	Reward	Iterations	Time (s)
Value	1%	618	0.99	-1	100	47	3.961
	5%	598	0.99	-1	100	48	4.425
	10%	569	0.99	-1	100	49	3.706
	15%	543	0.99	-1	100	51	3.552
	20%	513	0.99	-1	100	52	3.559
	25%	457	0.99	-1	100	55	3.509
Policy	1%	618	0.99	-1	100	8	15.349
	5%	598	0.99	-1	100	11	21.464
	10%	569	0.99	-1	100	9	15.087
	15%	543	0.99	-1	100	9	19.243
	20%	513	0.99	-1	100	9	16.926
	25%	457	0.99	-1	100	9	15.767

For both maze games 7x7 and 25x25, I compared the performance of value iteration and policy iteration with different gamma values 0.09, 0.39, 0.69, and 0.99. The results are presented in Table 3 (maze 7x7) and Table 4 (maze 25x25).

From Table 3 and 4, we can see that both the iterations and time increase when discount factor gamma goes bigger from 0.09 to 0.99 because it cares about more about future reward and it needs do more calculations. Interestingly we can see that value iteration and policy iteration are comparable when the gamma value is low. And we can also see that value iteration has some advantages when the gamma value is high. These features are similar to the effect of problem size as we saw in previous experiments.

Table 3. Maze 7x7: Value Iteration VS Policy Iteration with Gamma Effect

Algorithm	MDP	Gamma	Penalty	Reward	Iterations	Time (s)
Value	7x7	0.09	-1	100	3	0.146
	7x7	0.39	-1	100	8	0.163
	7x7	0.69	-1	100	13	0.187
	7x7	0.99	-1	100	25	0.218
Policy	7x7	0.09	-1	100	3	0.098
	7x7	0.39	-1	100	4	0.168
	7x7	0.69	-1	100	6	0.272
	7x7	0.99	-1	100	6	0.991

Table 4. Maze 25x25: Value Iteration VS Policy Iteration with Gamma Effect

Algorithm	MDP	Gamma	Penalty	Reward	Iterations	Time (s)
Value	25x25	0.09	-1	100	4	0.552
	25x25	0.39	-1	100	8	0.819
	25x25	0.69	-1	100	19	1.532
	25x25	0.99	-1	100	51	3.978
Policy	25x25	0.09	-1	100	3	1.585
	25x25	0.39	-1	100	4	2.295
	25x25	0.69	-1	100	6	5.708
	25x25	0.99	-1	100	10	33.691

3.5 Solve MDPs with Q Learning Algorithm

Both value iteration and policy iteration are planning algorithms which have access to a model of the world and thus is model-based algorithms. In this section, we will solve the MDPs using a learning algorithm, namely Q Learning. Learning algorithms are model-free algorithms, which means that the agent does not know how the world works and must learn how to behave from direct experience with the world.

Q-Learning is a model-free reinforcement learning algorithm. It can find the optimal policy for any finite MDP. Q-learning learns estimates of the optimal Q-values of an MDP and the agent's behavior is dictated by taking actions greedily based on the learned Q-values.

Because Q-learning algorithm needs first to learn the Q-values and then take action based on Q-values of current state, it will take longer time to find the optimal policy especially when the problem size is big such as the maze 25x25. Overall for both maze games, Q-learning takes much longer time to find a solution (Table 5) compared to value iteration and policy iteration (Table 1).

And again we see that with lower gamma values, the MDP problem size does not affects the performance much but it make big difference when the gamma values become high (Table 5). The data presented in Table 5 is the average of 4 experiments and the data is also attached as Excel file (gamma-effect. xlsx).

Table 5. Performance of Q-learning on the MDPs and Gamma Effect

Algorithm	MDP	Gamma	Penalty	Reward	Iterations	Time (s)
Q-learning	7x7	0.09	-1	100	3	0.653
	7x7	0.39	-1	100	4	0.632
	7x7	0.69	-1	100	6	0.896
	7x7	0.99	-1	100	6	2.485
Q-learning	25x25	0.09	-1	100	3	1.698
	25x25	0.39	-1	100	4	2.349
	25x25	0.69	-1	100	6	6.576
	25x25	0.99	-1	100	10	32.796

To see the effect of problem size (number of states) on the cumulative reward, I also plot the steps of learning for both maze games (Figure 4 and 5). Figure 4 for maze 7x7 shows that the cumulative reward starting to increase at around 1000 steps; whereas Figure 5 for maze 25x25 shows that the cumulative reward starting to increase at around 2000 steps. These figures explain why large problem (maze 25x25) takes longer time to find an optimal solution.

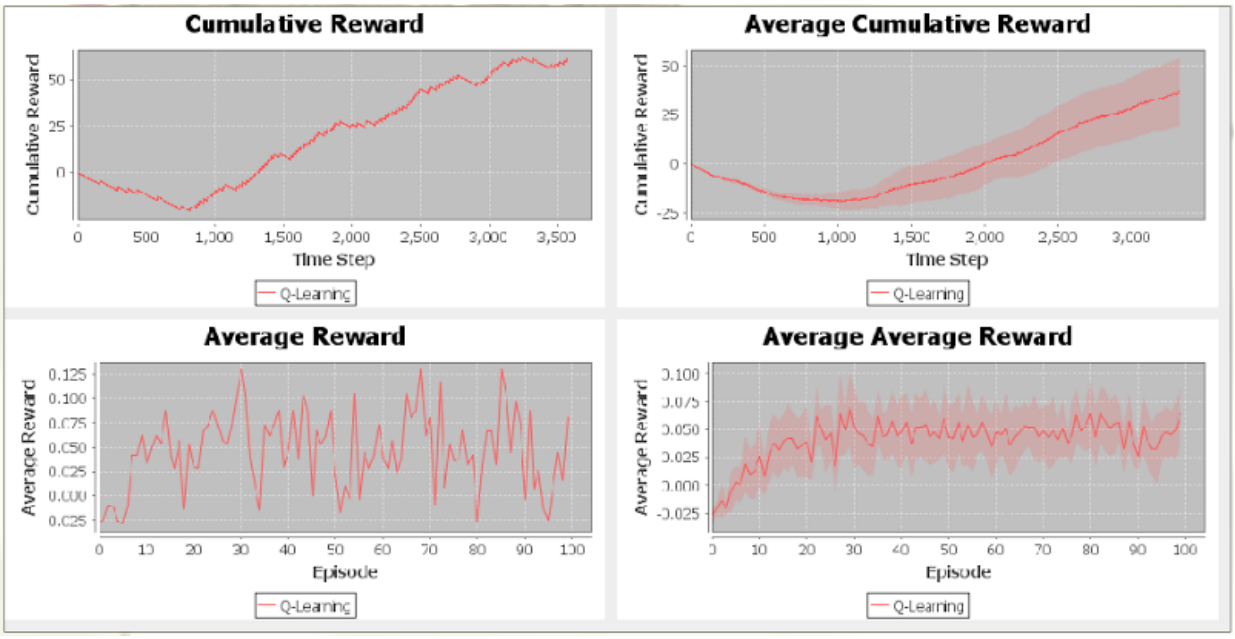

Figure 4. Maze Game 7x7: Rewards VS Steps & Episodes

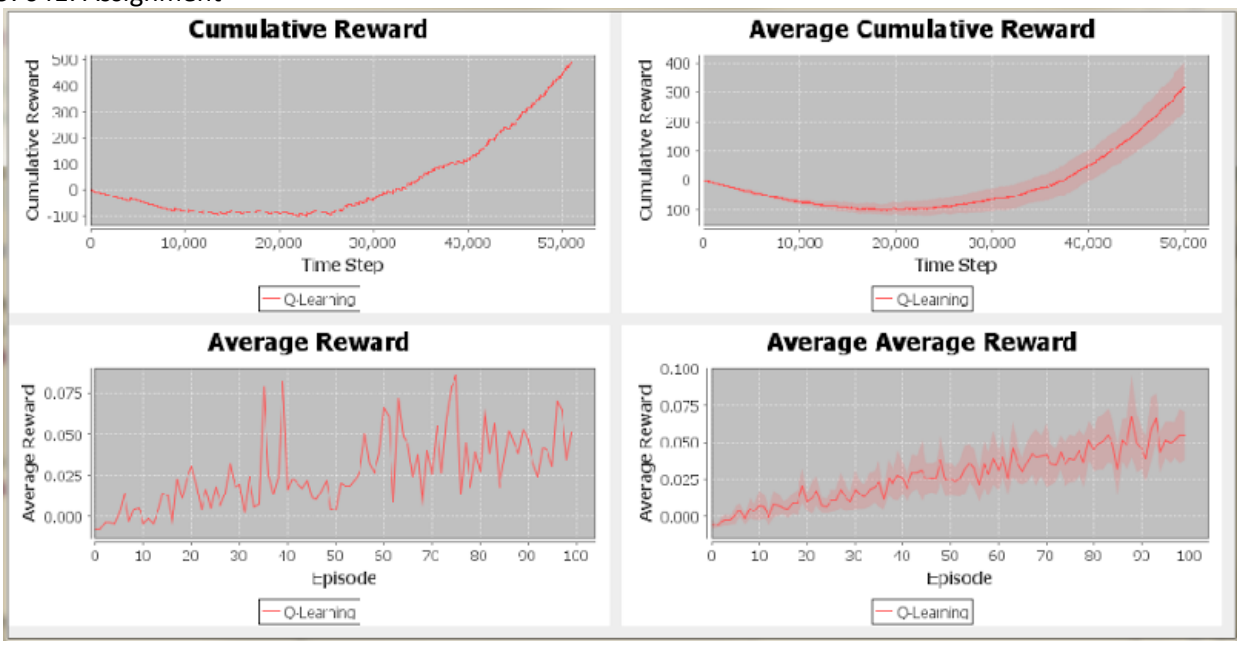

Figure 5. Maze Game 25x25: Rewards VS Steps & Episodes

4. Conclusions

In this assignment, I mainly compared two algorithms (value iteration and policy iteration) in solving two MDP problems. From the experiments that I did we can induce the following conclusions.

1) With large size problem, value iteration is usually faster than policy iteration; while policy iteration may have better performance with small size problem.
2) In spite of their different performances, value iteration and policy iteration converge at the same optimal values/policies for both small and large size of MDPs.
3) The number of states can affect the number of iterations for value iteration algorithm; but it did not have any effect for policy iteration algorithm.
4) The discount factor gamma has similar effect as problem size. At lower gamma values, policy iteration has comparable performance with value iteration; while at higher gamma values, value iteration has better performance.

Besides above two algorithms, I also applied model-free learning algorithm Q-Learning in these two MDPs. Overall, Q-learning takes longer time to solve the MDPs compared to both value iteration and policy iteration. Meanwhile, the problem size and discount factor have similar effect on Q-learning as in the other two model based algorithms.