Assignment 4 - Markov Decision Processes

Prithi Bhaskar
pbhaskar8@gatech.edu

1 Introduction

Two Markov Decision Processes were identified for the purpose of this assignment and some of the reinforcement learning techniques were then applied to solve those problems. The techniques include Model-based(Value Iteration and Policy Iteration) and Model-free(Q-learning) approaches. The results of the experiments and analyses have been presented in detail in the following sections.

2 MDPs

The following sections discuss the two MDPs in detail. The reason why they are interesting is that these problems allow customising of their sizes which helps to highlight the shortcomings and strengths of various RL techniques and their ability to scale according to the size of the problems. They are interesting also because they are familiar puzzles which are challenging to solve especially when there is a need to find the most optimal solution, given that there are n number of solutions.

2.1 Problem 1 (Grid world)- Frozenlake

Frozenlake is a simple grid problem. The objective is to train an agent to reach the goal state from start state by navigating through a lake(represented as a grid of varying dimensions) that is mostly frozen but has some holes in between. If the agent falls into any of the holes, it receives a reward of 0 and has to start over. On reaching the goal state, it receives a reward of 1. To make the problem more challenging, a measure of randomness was added to the problem. i.e. given a policy and after taking an action as per the policy, the agent will reach its desired state only with probability 0.33 and with probability 0.33 will reach any of its neighboring states other than the desired state. For the purpose of this assignment, grids of three sizes have been chosen to demonstrate the underpinnings of various RL approaches. The grid sizes are 4x4, 8x8 and 16x16.

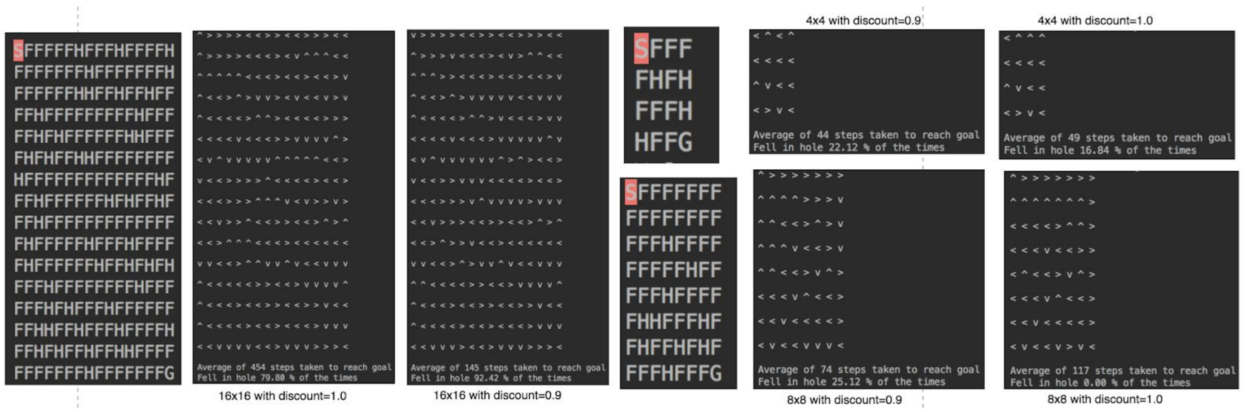
2.2 Problem 2(Non-grid world)- Towers of Hanoi

Towers of Hanoi is a mathematical puzzle consisting of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack to another rod, obeying the rules: 1)Only one disk can be moved at a time, 2) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod, 3)No larger disk may be placed on top of a smaller disk.

The minimal number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where n is the number of disks. For the purpose of this assignment, the number of disks to experiment with were chosen as 3,4 and 6 having 27, 81 and 729 states respectively.

3 Problem 1 - Analysis

3.1 Value Iteration



```
SFFFFFHFFFHFFFFH
FFFFFFFHFFFFFFFH
FFFFFFHHFFHFFHFF
FFHFFFFFFFFFFHFFF
FFHFHFFFFFFFHHFFF
FHFHFFHHFFFFFFFF
HFFFFFFFFFFFFFHF
FFHFFFFFFFHFHFFHF
FFHFFFFFFFFFFFFF
FHFFFFFHFFFHFFFF
FHFFFFFFHFFHFHFH
FFFHFFFFFFFFFHFFF
FFFHFHFFFHFFFFFF
FFHHFFHFFFHFFFFH
FFHFHFFHFFHHFFFF
FFFFFFFHFFFFFFFG
```

```
^ > > > > < < > > < < > > > < <
^ > > > > < < < > < v ^ ^ ^ < <
^ ^ ^ ^ ^ < < < > < < > < < > v
^ < < > ^ > v v > < v < < v > v
^ < < < < > ^ ^ > < < < < > > >
< < < v < < < > > v v v v ^ >
< v ^ v v v v v ^ ^ ^ ^ ^ < < >
v < < > > > > ^ < < < < > < < >
< < < > > > ^ ^ ^ v < v > > v >
< < v > > ^ < < > > < < > ^ > ^
< < > ^ ^ ^ < < < > < < < < < <
v v < < > ^ ^ v v ^ v < < v v v
^ < < < < < > > < < > v v v v ^
^ < < < > < < > > < < > > v < <
^ < < < < > < < > < < < > v v v
< < v v v v < < > v v v > > > <
Average of 454 steps taken to reach goal
Fell in hole 79.80 % of the times
```
16x16 with discount=1.0

```
v > > > > < < > > < < > > > < <
^ > > > v < < < > < v > ^ ^ < <
^ ^ ^ > > < < < > < < > < < > v
^ < < > ^ > v v v v v < < v v v
^ < < < < > ^ ^ > v < < < > v v
< < < v < < < > v v v v v ^ v
< v ^ v v v v v v ^ > ^ > < < >
v < < > > v v v < < < < > < < >
< < < > > > v ^ v v v v > v v v
< < v > > > < < > > < < > ^ > ^
< < > ^ > > v < < > < < < < < <
v v < < > ^ > v v ^ v < < v v v
^ ^ < < < < > > < < > v v v v ^
^ < < < > < < > > < < > > v < <
^ < < < < > < < > < < < > v v v
< < v v v > < < > v v v > > > <
Average of 145 steps taken to reach goal
Fell in hole 92.42 % of the times
```
16x16 with discount=0.9

```
SFFF
FHFH
FFFH
HFFG
```

4x4 with discount=0.9
```
< ^ < ^
< < < <
^ v < <
< > v <
Average of 44 steps taken to reach goal
Fell in hole 22.12 % of the times
```

4x4 with discount=1.0
```
< ^ ^ ^
< < < <
^ v < <
< > v <
Average of 49 steps taken to reach goal
Fell in hole 16.84 % of the times
```

```
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```

```
^ > > > > > > > >
^ ^ ^ ^ > > > v
^ ^ < < > ^ > v
^ ^ ^ v < < > v
^ ^ < < > v ^ >
< < < v ^ < < >
< < v < < < < >
< v < < v v v <
Average of 74 steps taken to reach goal
Fell in hole 25.12 % of the times
```
8x8 with discount=0.9

```
^ > > > > > > > >
^ ^ ^ ^ ^ ^ ^ >
< < < < > ^ ^ >
< < < v < < > >
< ^ < < > v ^ >
< < < v ^ < < >
< < v < < < < >
< v < < v > v <
Average of 117 steps taken to reach goal
Fell in hole 0.00 % of the times
```
8x8 with discount=1.0

Figure 1— Visualisation of the Fozenlake grid and the respective policies obtained from value iteration(left) 16x16 and policy for 16x16(topright) 4x4 and policy for 4x4(bottomright) 8x8 and policy for 8x8

Figure 1 shows the grids of various sizes chosen for experiments. S corresponds to the start state, F corresponds to frozen states, H corresponds to holes in the lake and G is the Goal. Since the problem is episodic i.e. not continuous giving more weight to future rewards yields the most optimal policy with a very high success rate(about 84% of times for 4x4 and almost 100% for 8x8). A factor of 0.9 yields a policy that's successful for about 75% of the times for 4x4 and 8x8. The downside of using 1.0 as a discount factor is that it takes a lot of iterations(about 80-100 times the number of iterations using 0.9 as discount) to converge. For value iteration, the difference between the sum of all the state values between two iterations(delta) was analysed to study convergence. For illustration purposes, the delta values chosen were 0.001 and 0.0001. The plots have been shown below.

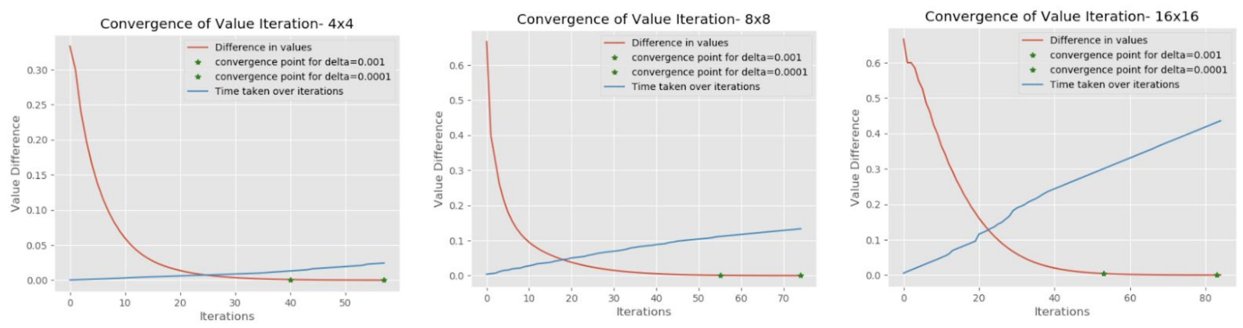


Figure 2— Convergence plots of value iteration for various sizes of Frozenlake(discount factor=0.9)

As shown in Figure 2, for both values of delta, value iteration converges after a finite number of iterations. Also, the policies obtained are always the same for a particular grid world showing that they are optimal policies. Albeit, this does not guarantee landing in the goal state because of the stochastic nature of the world. For 16x16, even though the policy seems to suggest

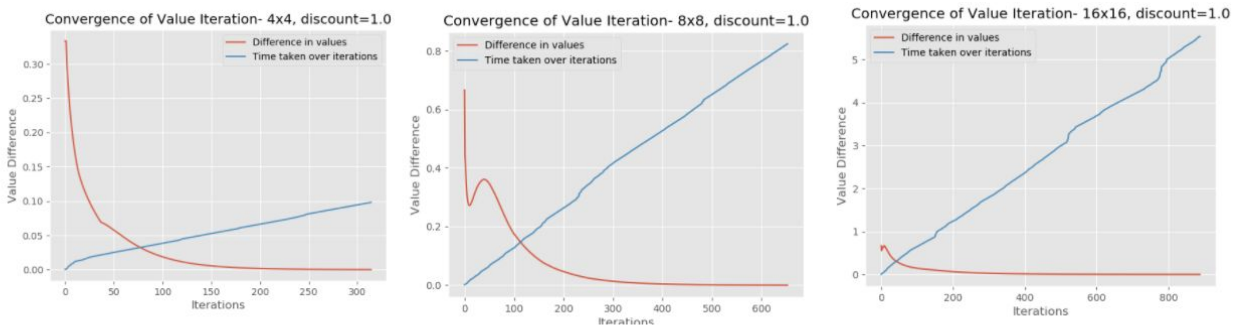
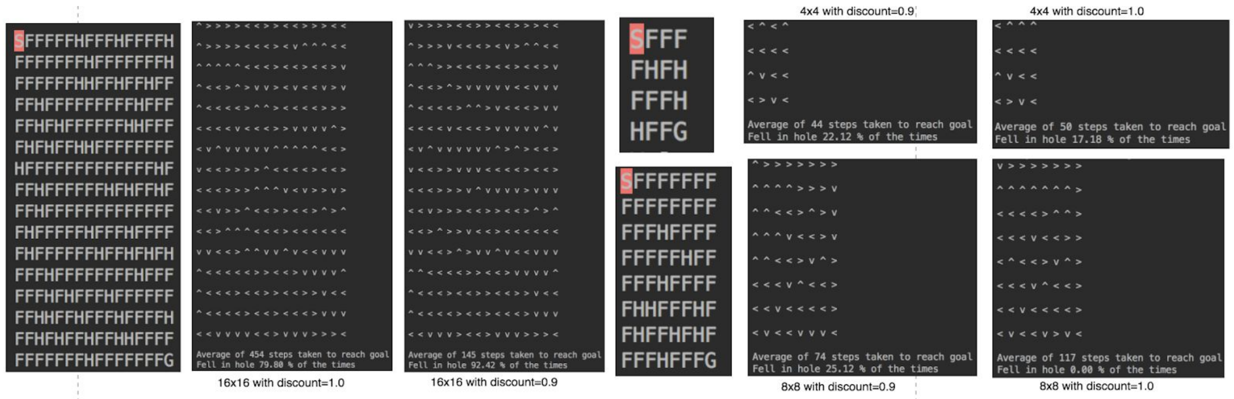


Figure 3— Convergence plots of value iteration for various sizes of Frozenlake(discount factor=0.1)

moving away from holes in states next to holes, the agent still ends up in a hole about 80% of the time on 10000 successive iterations.

3.2 Policy Iteration



```
SFFFFFHFFFHFFFFH
FFFFFFFHFFFFFFFH
FFFFFFHHFFHFFHFF
FFHFFFFFFFFFFHFFF
FFHFHFFFFFFFHHFFF
FHFHFFHHFFFFFFFF
HFFFFFFFFFFFFFHF
FFHFFFFFFHFHFFHF
FFHFFFFFFFFFFFFF
FHFFFFFHFFFHFFFF
FHFFFFFFHFFHFHFH
FFFHFFFFFFFFFHFFF
FFFHFHFFFHFFFFFF
FFHHFFHFFFHFFFFH
FFHFHFFHFFHHFFFF
FFFFFFFHFFFFFFFG
```

```
^ > > > > < < > > < < > > > < <
^ > > > > < < < > < v ^ ^ ^ < <
^ ^ ^ ^ ^ < < < > < < > < < > v
^ < < > ^ > v v > < v < < v > v
^ < < < < > ^ ^ > < < < < > > >
< < < < v < < < > > v v v v ^ >
< v ^ v v v v v ^ ^ ^ ^ ^ < < >
v < < > > > > ^ < < < < > < < >
< < < > > > ^ ^ ^ v < v > > v >
< < v > > ^ < < > > < < > ^ > ^
< < > ^ ^ ^ < < < > < < < < < <
v v < < > ^ ^ v v ^ v < < v v v
^ < < < < < > > < < > v v v v ^
^ < < < > < < > > < < > > v < <
^ < < < < > < < > < < < > v v v
< < v v v v < < > v v v > > > <
Average of 454 steps taken to reach goal
Fell in hole 79.80 % of the times
```
16x16 with discount=1.0

```
v > > > > < < > > < < > > > < <
^ > > > v < < < > < v > ^ ^ < <
^ ^ ^ > > < < < > < < > < < > v
^ < < > ^ > v v v v v < < v v v
^ < < < < > ^ ^ > v < < < > v v
< < < < v < < < > v v v v v ^ v
< v ^ v v v v v v ^ > ^ > < < >
v < < > > v v v < < < < > < < >
< < < > > > v ^ v v v v > v v v
< < v > > > < < > > < < > ^ > ^
< < > ^ > > v < < > < < < < < <
v v < < > ^ > v v ^ v < < v v v
^ ^ < < < < > > < < > v v v v ^
^ < < < > < < > > < < > > v < <
^ < < < < > < < > < < < > v v v
< < v v v > < < > v v v > > > <
Average of 145 steps taken to reach goal
Fell in hole 92.42 % of the times
```
16x16 with discount=0.9

```
SFFF
FHFH
FFFH
HFFG
```

```
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```

4x4 with discount=0.9
```
< ^ < ^
< < < <
^ v < <
< > v <
Average of 44 steps taken to reach goal
Fell in hole 22.12 % of the times
```

4x4 with discount=1.0
```
< ^ ^ ^
< < < <
^ v < <
< > v <
Average of 50 steps taken to reach goal
Fell in hole 17.18 % of the times
```

```
^ > > > > > > >
^ ^ ^ ^ > > > v
^ ^ < < > ^ > v
^ ^ ^ v < < > v
^ ^ < < > v ^ >
< < < v ^ < < >
< < v < < < < >
< v < < v v v <
Average of 74 steps taken to reach goal
Fell in hole 25.12 % of the times
```
8x8 with discount=0.9

```
v > > > > > > >
^ ^ ^ ^ ^ ^ ^ >
< < < < > ^ ^ >
< < < v < < > >
< ^ < < > v ^ >
< < < v ^ < < >
< < v < < < < >
< v < < v > v <
Average of 117 steps taken to reach goal
Fell in hole 0.00 % of the times
```
8x8 with discount=1.0

Figure 4— Visualisation of the Fozenlake grid and the respective policies obtained from policy iteration(left) 16x16 and policy for 16x16(topright) 4x4 and policy for 4x4(bottomright) 8x8 and policy for 8x8

As shown in Figure 4, the policies obtained from policy iteration for all the dimensions of the grid world are the same ones as obtained from value iteration. Convergence for policy iteration has been defined in terms of the difference in actions prescribed by the policies obtained from successive iterations. Since a policy is assumed arbitrarily at the beginning, the number of iterations differ between multiple runs of policy iteration depending on the chosen initial policy. However, the number of iterations taken by policy iteration is always lesser than the number taken by value iteration. As can be observed from Figure 2 and Figure 5, there is a significant difference in the time taken by the algorithms to converge as the size of the grid increases. Policy iteration converges faster for all grid sizes and also identifies the optimal policy thus making it the most efficient for the Frozenlake problem.

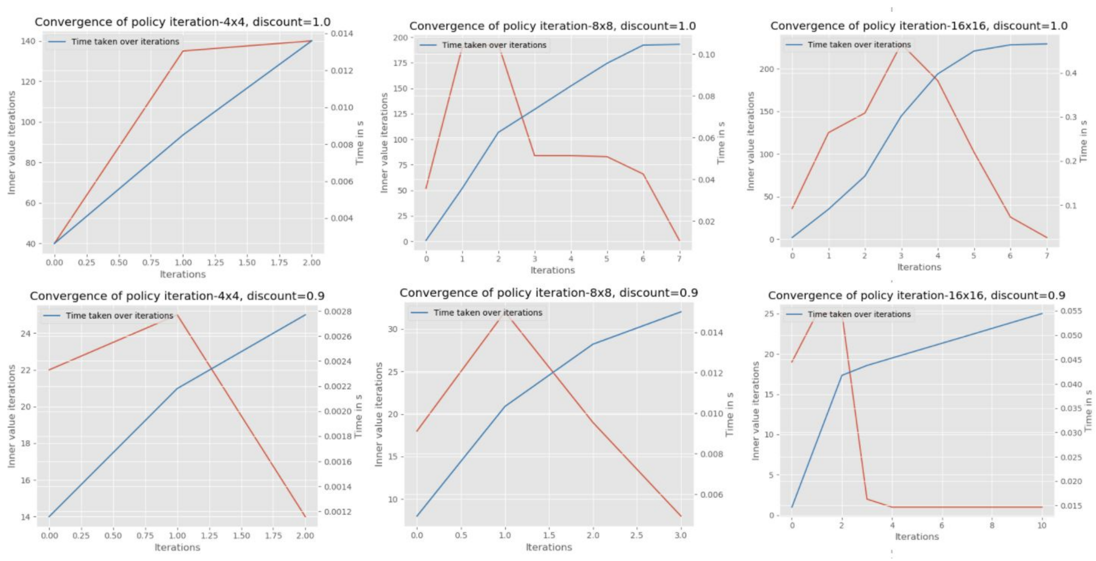
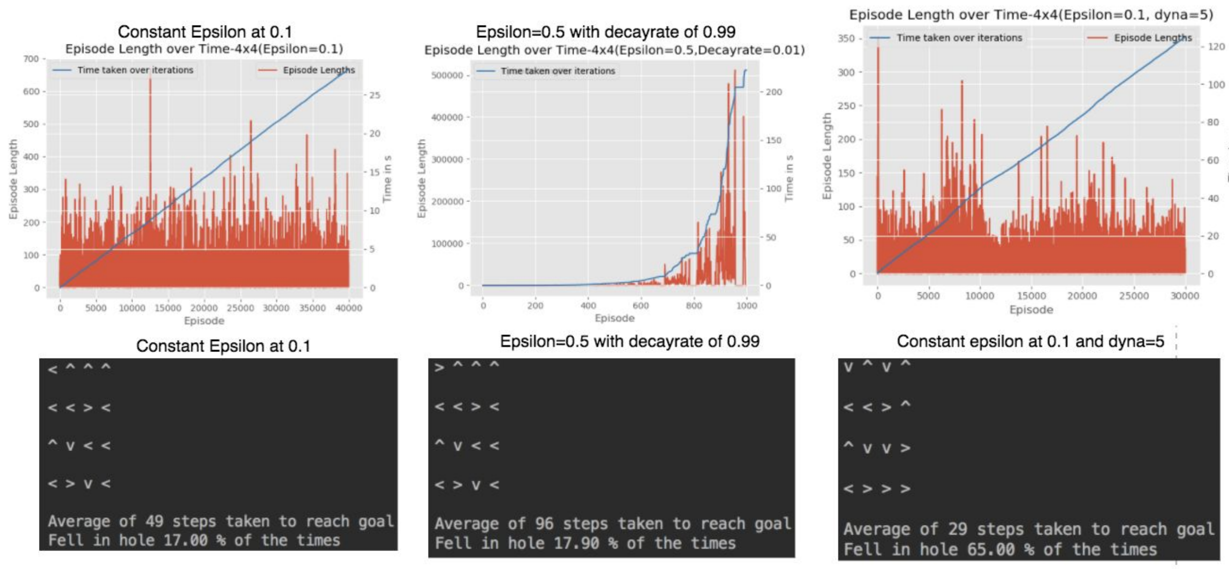


Figure 5— Convergence plots of policy iteration for various sizes of Frozenlake

Figure 5 plots the number of steps taken in each iteration of the policy iteration algorithm to bring the difference in values of the states between successive iterations in the policy evaluation step to a minimum value. As the values get filled up based on the true utility of the states at successive iterations, the number of steps in the policy evaluation step decreases and eventually the algorithm converges.

3.3 Q-Learning



```
< ^ ^ ^

< < > <

^ v < <

< > v <

Average of 49 steps taken to reach goal
Fell in hole 17.00 % of the times
```

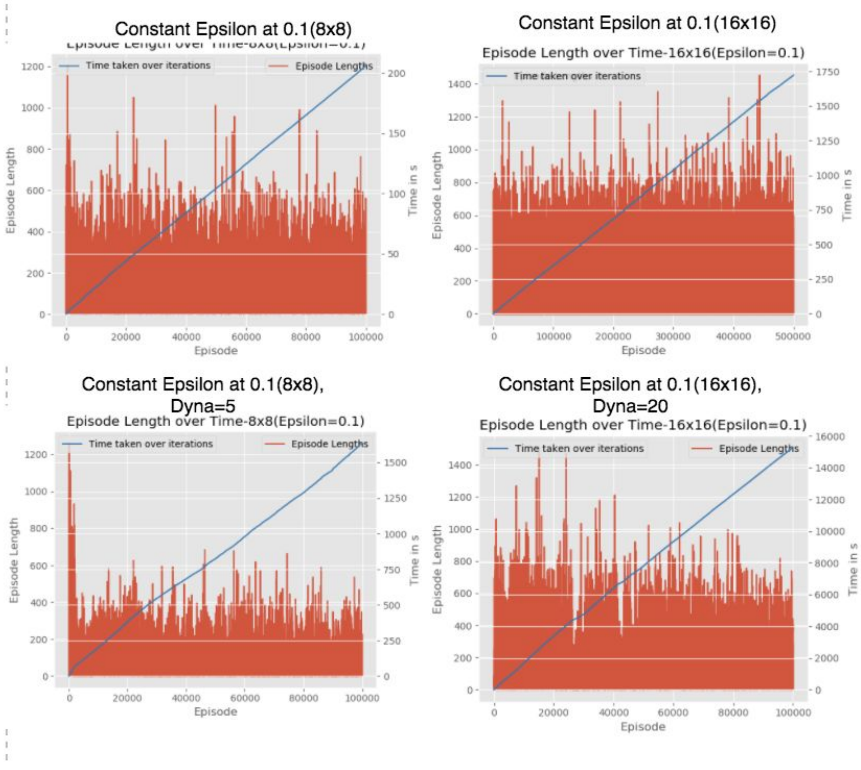
```
> ^ ^ ^

< < > <

^ v < <

< > v <

Average of 96 steps taken to reach goal
Fell in hole 17.90 % of the times
```

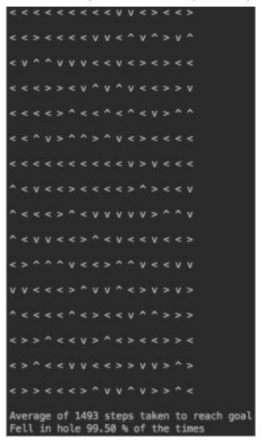
```
v ^ v ^

< < > ^

^ v v >

< > > >

Average of 29 steps taken to reach goal
Fell in hole 65.00 % of the times
```

Figure 6— Convergence plots of model free Learning(Q Learning) for 4x4 grid. (bottom) policies to which the approaches converge and the percentage of times the agent reached the goal on 1000 runs

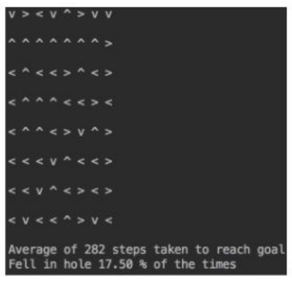
Figure 6 shows the convergence plots of Q-learning for 4x4 world. The 'notion' of convergence for Q-learning differs from the other approaches in that there is no reduction of the number of steps in each episode(in this case the difference in number of steps between two successive iterations), as noted for others. This is due to the stochastic nature of the world and at initial iterations, the difference in length of episodes is low because the agent falls into holes as it explores the space. There are spikes as the agent learns the space but the magnitude of the spikes decrease with increase in number of iterations as the agent learns the optimal path to the goal. The graphs in Figure 6 plot the difference in lengths of the successive iterations as a function of iterations. The approaches used with Q-learning are: 1.a constant rate of epsilon (0.1) for all the iterations, 2.starting with an epsilon and decaying it slowly over iterations(1% of the epsilon is retained after every iteration),3.Dyna Q model where at each iteration after taking an action, the q-table is randomly updated based on the new value calculated. This is normally done when taking a step in the environment is considered costly. The intuition is that the agent learns more than it does from a single step than the conventional methods.

Due to the non-deterministic nature of the problem, a random action throughout the process of learning helps in faster convergence and exploration of the surface. Dyna Q model helps in faster convergence as well but not as fast as the constant epsilon approach. Epsilon decay approach converges to an optimum with only about 12% failure rate but takes almost twice the time taken by a constant epsilon approach. Dyna Q does not converge to an optimal policy even after a high number of iterations as shown in the figure(a failure rate of 65%).





```
< < < < < < < < < v v < > < < >
< < > < < < < v v < ^ v ^ > v ^
< v ^ ^ v v v < < v < > < > < <
< < < > > < v ^ v ^ v < < > > v
< < < < > ^ < < ^ < ^ < v > ^ ^
< < ^ v > ^ ^ > ^ v < > < < < <
< < < < < < < < < < v > v < < <
^ < v < < > < < < < > ^ > < < v
^ < < < > ^ < v v v v v > ^ ^ v
^ < v v < < > ^ < v < < v < < >
< > ^ ^ ^ v < < > ^ ^ v < < v v
v v < < < > ^ v v ^ < > v > v >
^ < < < < ^ < > < < v ^ ^ > > >
< > > ^ < < v > ^ < > < < > > <
< > ^ < < v v < < > > v v > ^ >
< > > < < < > ^ v v ^ v > > ^ <

Average of 1493 steps taken to reach goal
Fell in hole 99.50 % of the times
```



```
v > < v ^ > v v
^ ^ ^ ^ ^ ^ ^ >
< ^ < < > ^ < >
< ^ ^ ^ < < > <
< ^ ^ < > v ^ >
< < < v ^ < < >
< < v ^ < > < >
< v < < ^ > v <

Average of 282 steps taken to reach goal
Fell in hole 17.50 % of the times
```

Figure 7— (Left) Convergence plots of Q Learning for 8x8 and 16x16 grid. (Right) policies to which the approaches converge and the percentage of times the agent reached the goal on 1000 runs

As shown in Figure 6, the Q-learning agent is able to find an optimal path for 8x8 world with a constant epsilon approach and the policy is the same as obtained from VI/PI. But for 16x16, the agent is not able to converge to an optimal policy even with a large number of iterations(in the order of 1million). This might be because of the placement of the holes and the percentage of holes in the grid space. The grid space considered for 16x16 contains 20%holes. The agent might exhibit the same behaviour as for 8x8 with even higher iterations(order of 10 or 1billion) but the model-based approaches are better by a huge factor(an order of 10000) in terms of speed. Figure 8 provides a summary of comparison of the model-based and model-free approaches in terms of both time and number of iterations.

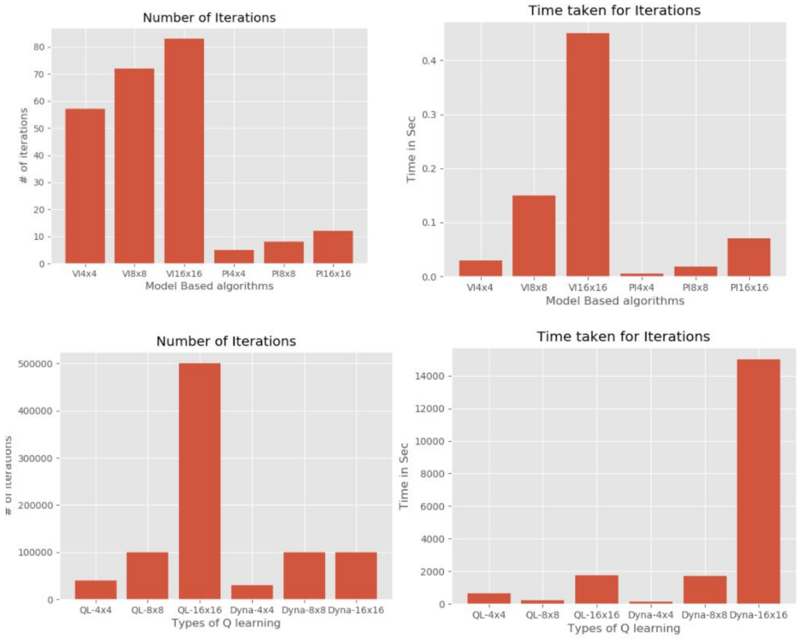
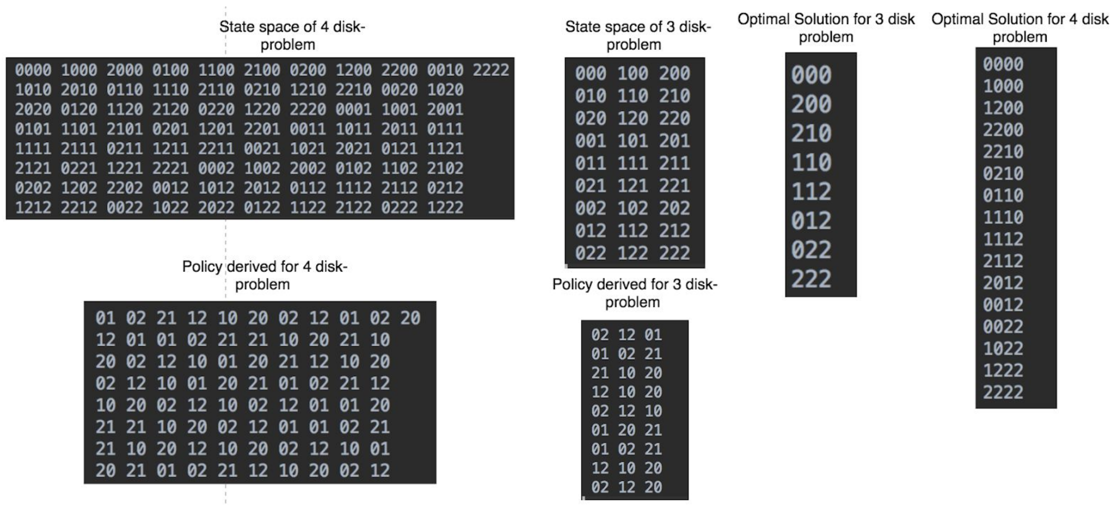


Figure 8— (TopLeft) Number of iterations of model-based approaches, (TopRight)Time taken by model-based approaches,(BottomLeft) Number of iterations of Q-learning, (BottomRight)Time taken by Q-learning

4 Problem 2 - Analysis

4.1 Value Iteration

States are represented as a string of numbers-0,1 and 2 representing the disks 0 1 and 2. The position of integers in the strings represents the disks. For example, state space 200 of a 3 disk problem represents a state where the first disk is in pole 2 and the second and third disks are in pole 0. The below figures show the policies and solutions obtained using VI.



State space of 4 disk-problem

```
0000 1000 2000 0100 1100 2100 0200 1200 2200 0010 2222
1010 2010 0110 1110 2110 0210 1210 2210 0020 1020
2020 0120 1120 2120 0220 1220 2220 0001 1001 2001
0101 1101 2101 0201 1201 2201 0011 1011 2011 0111
1111 2111 0211 1211 2211 0021 1021 2021 0121 1121
2121 0221 1221 2221 0002 1002 2002 0102 1102 2102
0202 1202 2202 0012 1012 2012 0112 1112 2112 0212
1212 2212 0022 1022 2022 0122 1122 2122 0222 1222
```

State space of 3 disk-problem

```
000 100 200
010 110 210
020 120 220
001 101 201
011 111 211
021 121 221
002 102 202
012 112 212
022 122 222
```

Optimal Solution for 3 disk problem

```
000
200
210
110
112
012
022
222
```

Optimal Solution for 4 disk problem

```
0000
1000
1200
2200
2210
0210
0110
1110
1112
2112
2012
0012
0022
1022
1222
2222
```

Policy derived for 4 disk-problem

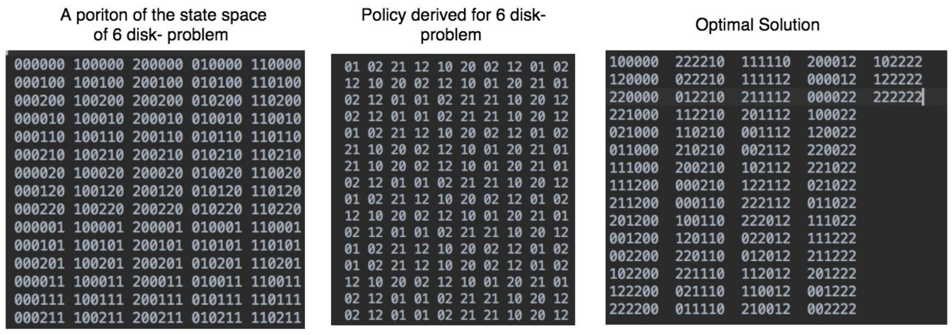
```
01 02 21 12 10 20 02 12 01 02 20
12 01 01 02 21 21 10 20 21 10
20 02 12 10 01 20 21 12 10 20
02 12 10 01 20 21 01 02 21 12
10 20 02 12 10 02 12 01 01 20
21 21 10 20 02 12 01 01 02 21
21 10 20 12 10 20 02 12 10 01
20 21 01 02 21 12 10 20 02 12
```

Policy derived for 3 disk-problem

```
02 12 01
01 02 21
21 10 20
12 10 20
02 12 10
01 20 21
01 02 21
12 10 20
02 12 20
```

Figure 9— State spaces, Policies derived and optimal solutions of Towers of Hanoi problem with 3 and 4 disks

Actions and thereby the policies are represented by a string consisting of a pair of integers from the set(0,1,2) where the first integer represents the pole from which the disk should be moved and the second integer represents the pole to which the disk should be moved. Optimal solution is the list of states taken by the algorithm from the initial state to reach the goal state. In general, the most optimal solution for towers of hanoi involves $2^n - 1$ steps, where n is the number of disks.



A poriton of the state space of 6 disk- problem

```
000000 100000 200000 010000 110000
000100 100100 200100 010100 110100
000200 100200 200200 010200 110200
000010 100010 200010 010010 110010
000110 100110 200110 010110 110110
000210 100210 200210 010210 110210
000020 100020 200020 010020 110020
000120 100120 200120 010120 110120
000220 100220 200220 010220 110220
000001 100001 200001 010001 110001
000101 100101 200101 010101 110101
000201 100201 200201 010201 110201
000011 100011 200011 010011 110011
000111 100111 200111 010111 110111
000211 100211 200211 010211 110211
```

Policy derived for 6 disk-problem

```
01 02 21 12 10 20 02 12 01 02
12 10 20 02 12 10 01 20 21 01
02 12 01 01 02 21 21 10 20 12
02 12 01 01 02 21 21 10 20 12
01 02 21 12 10 20 02 12 01 02
21 10 20 02 12 10 01 20 21 01
21 10 20 02 12 10 01 20 21 01
02 12 01 01 02 21 21 10 20 12
01 02 21 12 10 20 02 12 01 02
12 10 20 02 12 10 01 20 21 01
02 12 01 01 02 21 21 10 20 12
01 02 21 12 10 20 02 12 01 02
01 02 21 12 10 20 02 12 01 02
12 10 20 02 12 10 01 20 21 01
02 12 01 01 02 21 21 10 20 12
02 12 01 01 02 21 21 10 20 12
```

Optimal Solution

```
100000  222210  111110  200012  102222
120000  022210  111112  000012  122222
220000  012210  211112  000022  222222
221000  112210  201112  100022
021000  110210  001112  120022
011000  210210  002112  220022
111000  200210  102112  221022
111200  000210  122112  021022
211200  000110  222112  011022
201200  100110  222012  111022
001200  120110  022012  111222
002200  220110  012012  211222
102200  221110  112012  201222
122200  021110  110012  001222
222200  011110  210012  002222
```

Figure 10— State spaces, Policies derived and optimal solutions of Towers of Hanoi problem with 6 disks

As with problem 1, convergence of VI was measured based on the difference between successive iterations and if the difference was less than a threshold, the algorithm was considered as converged. As shown in the convergence plots in Figure 10, VI finds the most optimal solution(one involving $2^n - 1$ steps) in a finite number of iterations for all the grid sizes considered. Various delta values were passed to the algorithm and it was found that as delta increases the algorithm converges faster until a certain value of delta and increasing delta beyond that point leads to convergence to a policy that is not optimal. For this problem, even a delta value of 300 results in convergence to the optimal policy. This is because even though the values are not the true values of the states, they still represent the relative utility of the states with respect to each other and hence the argmax component remains the same irrespective of the states' values. The maximum value of delta for which the algorithm converged to the optimal policy was chosen for plotting.

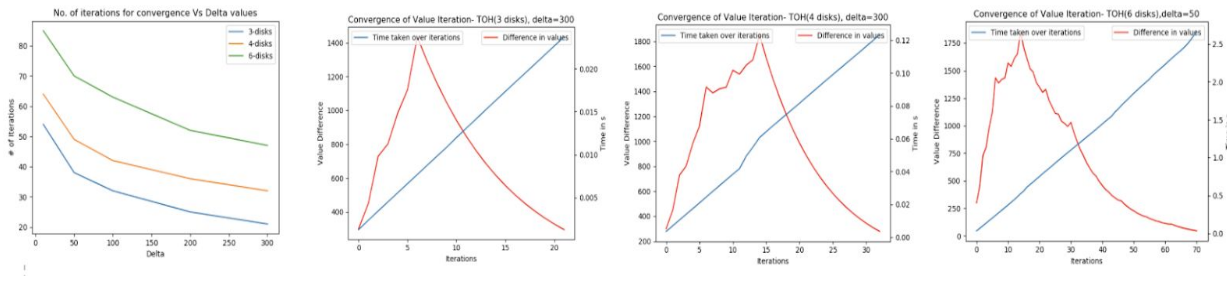


Figure 11— Convergence plots of VI for various sizes of TOH problem

4.2 Policy Iteration

Convergence for policy iteration has been defined in terms of the difference in actions prescribed by the policies obtained from successive iterations. So when the policies do not change over two consecutive iterations, the algorithm is marked as converged. Policies and optimal solution derived from PI are the same as obtained from VI(shown in Figures 8&9). Keeping the delta value for the policy evaluation step at a minimum(for this problem delta was set to 1), PI converges with a minimum number of iterations. Using the same number of delta values used in VI leads to an increased number of PI iterations as the individual values of the states do not converge to their true values with a large delta thus taking a lot of iterations for PI to converge. Hence the minimum value of delta for which the algorithm converged with minimum number of iterations was chosen for the purpose of plotting and comparisons.

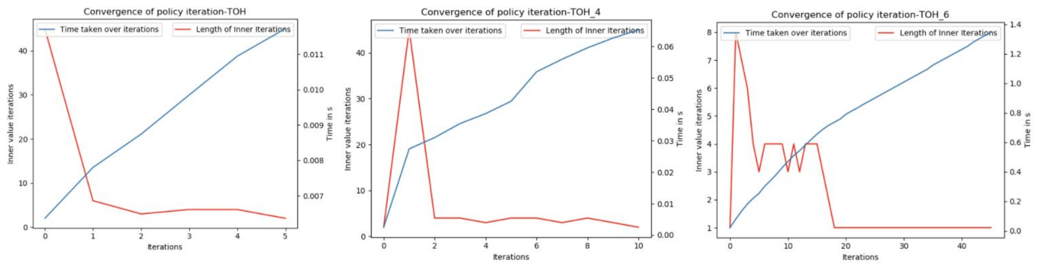


Figure 12— Convergence plots of PI for various sizes of TOH problem

It can be observed from figures 10 and 11 that PI converges to the optimal policy with lesser number of iterations and in lesser time as compared to VI. The actual convergence point of PI differs based on the initially assumed policy. However, it remains lesser than VI in all cases. The same pattern continues as the size of the problem is increased.

4.3 Q-Learning

The Three approaches used with problem 1 were taken for problem 2 as well. Different values of epsilon and learning rates were tried and it was observed that as learning rates are increased towards 1, the number of iterations to converge decreases. This is because of the deterministic nature of the problem and with a high learning rate the agent converges quickly. The convergence plots are shown below. The variation in step length between two successive iterations has been plotted to show convergence.

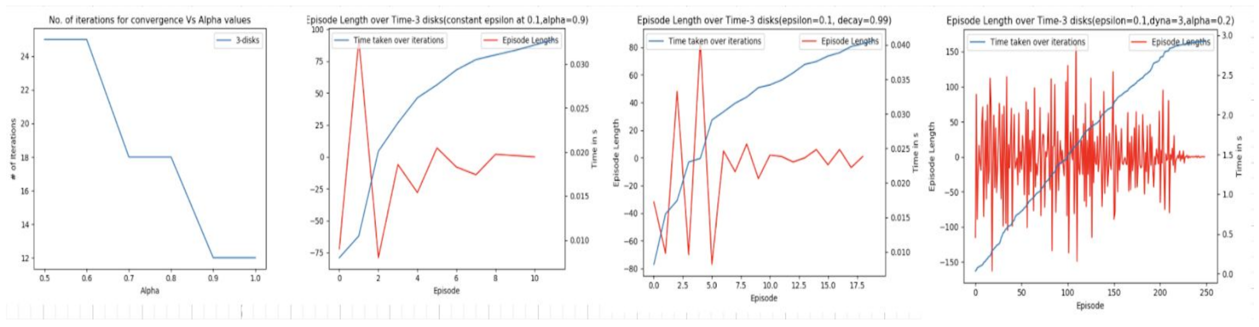


Figure 13— Convergence plots of 3-disks TOH problem using various Q-learning approaches

A high learning rate that results in the optimal solution was chosen to make the algorithm converge in a minimum number of iterations. As noted in Figure 13, a constant low value of epsilon results in the least number of iterations. This is because a random action helps when the Q-table has the same values for many states. Alpha decay approach exhibits almost similar performance in terms of the number of iterations. But dynaQ takes almost 12 times as much iterations as the other approaches to converge to the optimal policy. This is because the q-values are updated more rapidly based on the new values obtained and hence the learning rate cannot be so high as in the other methods. In general, dynaQ model is beneficial where the cost of taking a step in the actual environment is high. But for the TOH problem, because of its deterministic nature, the problem is rather straightforward and hence a high learning rate is beneficial. This explains the advantages of alpha decay methods over dynaQ model.

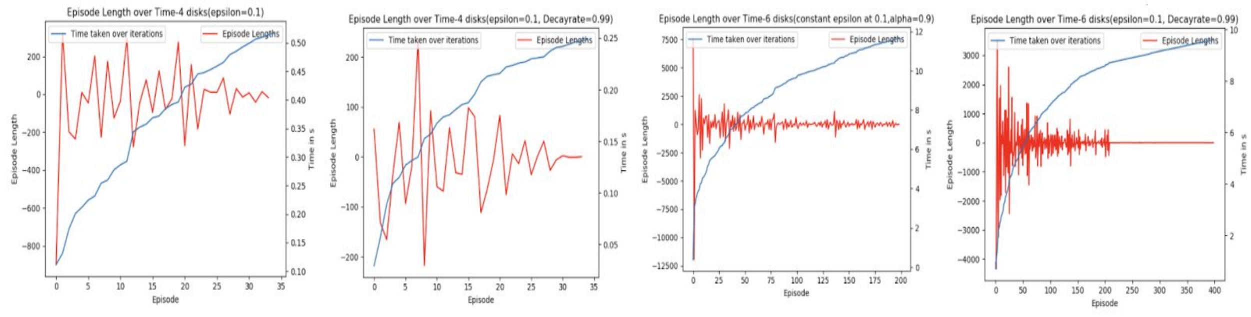


Figure 14— Convergence plots of 4-disks and 6-disks TOH problem using various Q-learning approaches

As with problem 1, both VI and PI converge in a lesser number of iterations to the optimal policy when compared to model-free methods. Also, as the size of state space increases, the time and number of iterations required for convergence increases more sharply in Q-learning than the model-based approaches. Among VI and PI, PI converges faster than VI in terms of both iterations and time irrespective of the problem size.

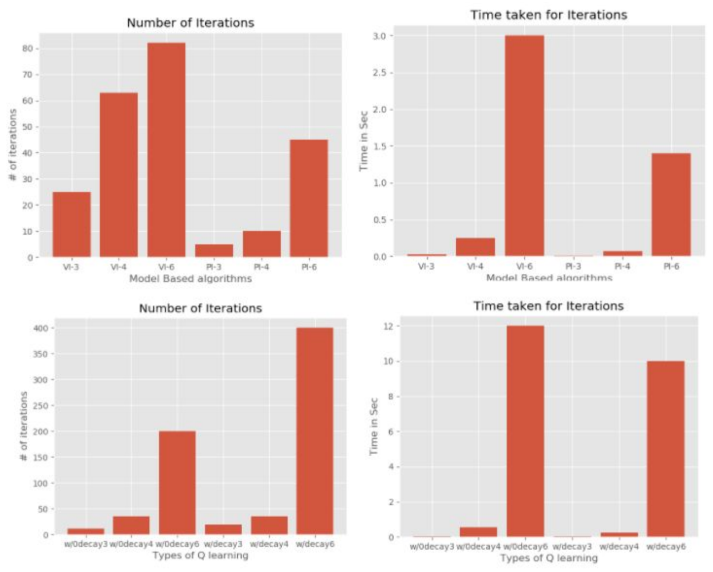


Figure 15— (TopLeft) Number of iterations of model-based approaches, (TopRight)Time taken by model-based approaches,(BottomLeft) Number of iterations of Q-learning, (BottomRight)Time taken by Q-learning

Figure 15 provides a summary of comparison of the model-based and model-free approaches in terms of both time and number of iterations for various sizes of the TOH problem.

4 References

1. Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake Vanderplas, Arnaud Joly, Brian Holt, Gaël Varoquaux, "API design for machine learning software: experiences from the scikit-learn project", *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases*, 2013.
2. Wikipedia contributors. (2020, April 3). Tower of Hanoi. In *Wikipedia, The Free Encyclopedia*. Retrieved 23:57, April 11, 2020, from https://en.wikipedia.org/w/index.php?title=Tower_of_Hanoi&oldid=948790909
3. https://github.com/RobertTLange/gym-hanoi/blob/master/gym_hanoi/envs/hanoi_env.py
4. Kaelbling P.L., Littman L.M., Moore W.A, Reinforcement Learning: A Survey, 1996
5. https://www.geeksforgeeks.org/q-learning-in-python/
6. https://medium.com/@annishared/searching-for-optimal-policies-in-python-an-intro-to-optimization-7182d6fe4dba