The Taxi V3 environment

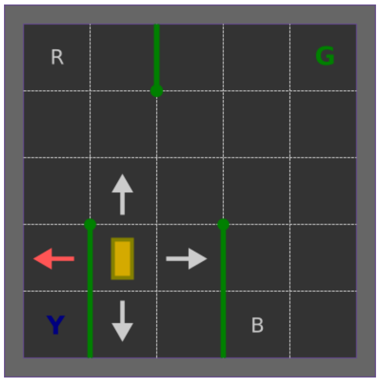
SMDP, Intro-Q learning, and more

V S S Anirudh Sharma and Tanzir Silar

Taxi is one of many environments available at OpenAI Gym. These environments are used to develop and benchmark reinforcement learning algorithms.

The goal of Taxi is to pick-up passengers and drop them off at the destination in the least amount of moves.

For Taxi-v3, the minimum previous 100 scores averaged for 'solved' is 9.7



Fully Hardcoded

We first hand coded policies for each of the 4 options:

1. Go to R
2. Go to G
3. Go to Y
4. Go to B

Based on the state, we also decide whether to pick or drop the passenger.

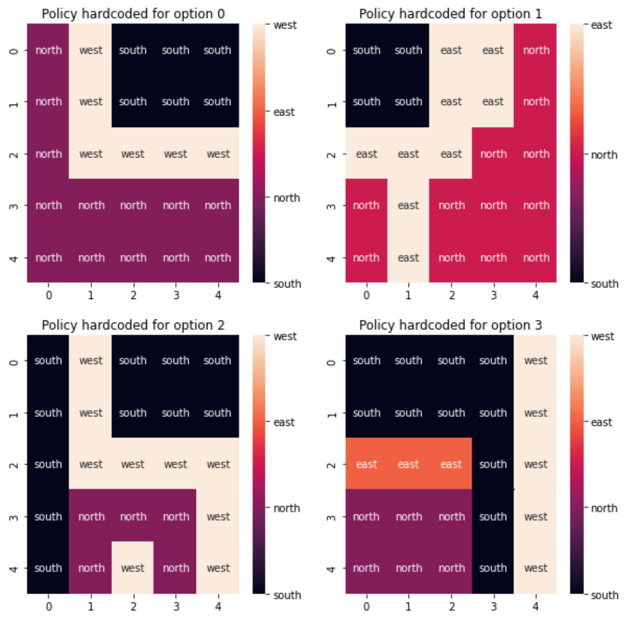


Figure 1: Fully hard coded option policies

Hard coded option selection with Q-Learning over option policies

Here, the 4 options are (with no other action/option):

1. Go to R, pick/drop according to the state
2. Go to G, pick/drop according to the state
3. Go to Y, pick/drop according to the state
4. Go to B, pick/drop according to the state

The sequence of option selection would be:

1. Based on state, find passenger location and select option to that location
2. Based on state, find drop location and select option to that location

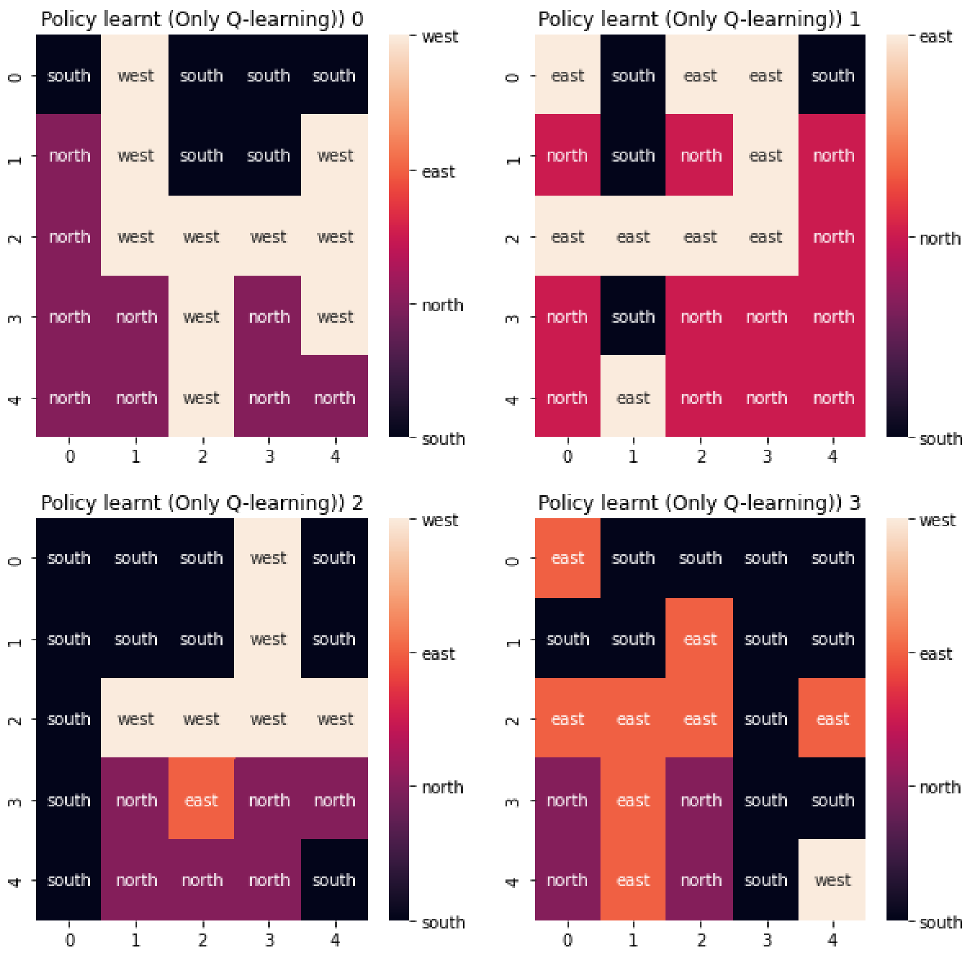


Fig 2: Option policies for hard-coded option selection

SMDP Q-Learning

Here, the 4 options are

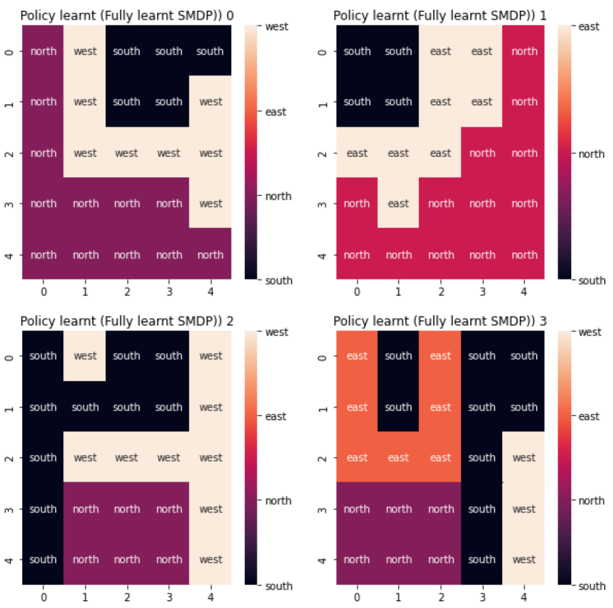
1. Go to R, pick/drop according to the state
2. Go to G, pick/drop according to the state
3. Go to Y, pick/drop according to the state
4. Go to B, pick/drop according to the state

No other actions/options are considered. Each of the option policies was learnt through Q-Learning

Our state space for this model is NOT the default 500 size state space. Instead, considering the options and the required state variables to make choices, we have taken a subspace of this default state space, which is of size 20. This new subspace represents the 5 x 4 combinations of <Passenger position> x <Drop location>.

```
def Sub(state, nPas = nPas, nDrop = nDrop):
  _,_,pas,drop = env.decode(state)
  subState = nDrop*pas+drop
  return subState
```

This helps us reduce the complexity of the model and speed up the learning. The overall option selection policy learnt by SMDP is given in figure 4.



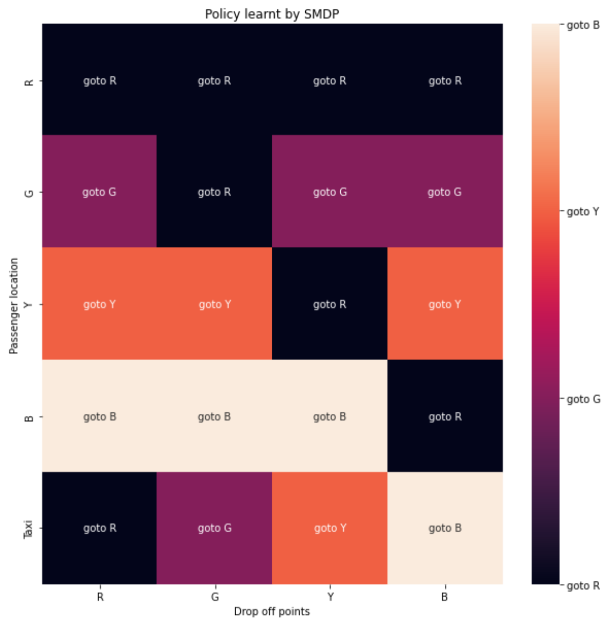
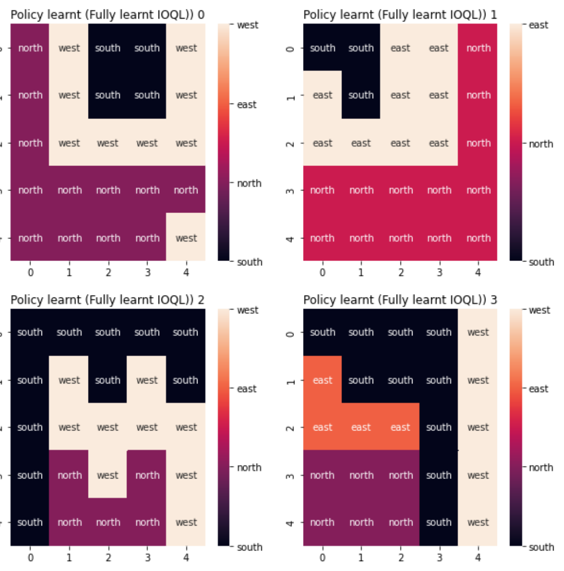


Fig 3: Option policies for SMDP and *Figure 4: Option selection policies of SMDP*

Intra-option Q-Learning

With the policy learning state space and options only configuration similar to SMDP Q-Learning, we get the following results



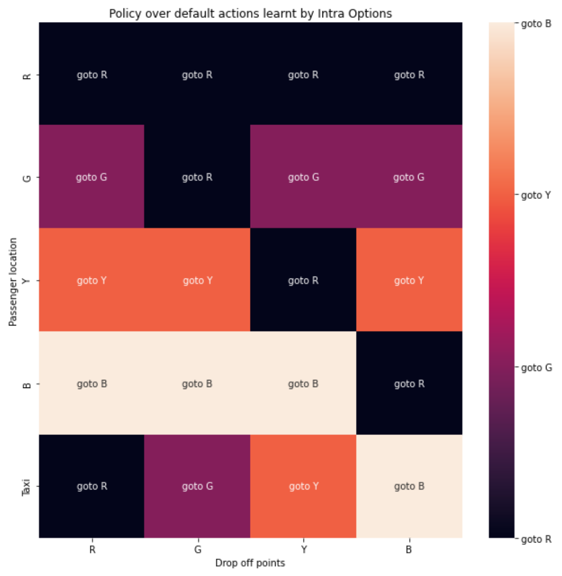


Fig 5: Option policies for IOQL and *Figure 6: Option selection policies of IOQL*

Rewards obtained

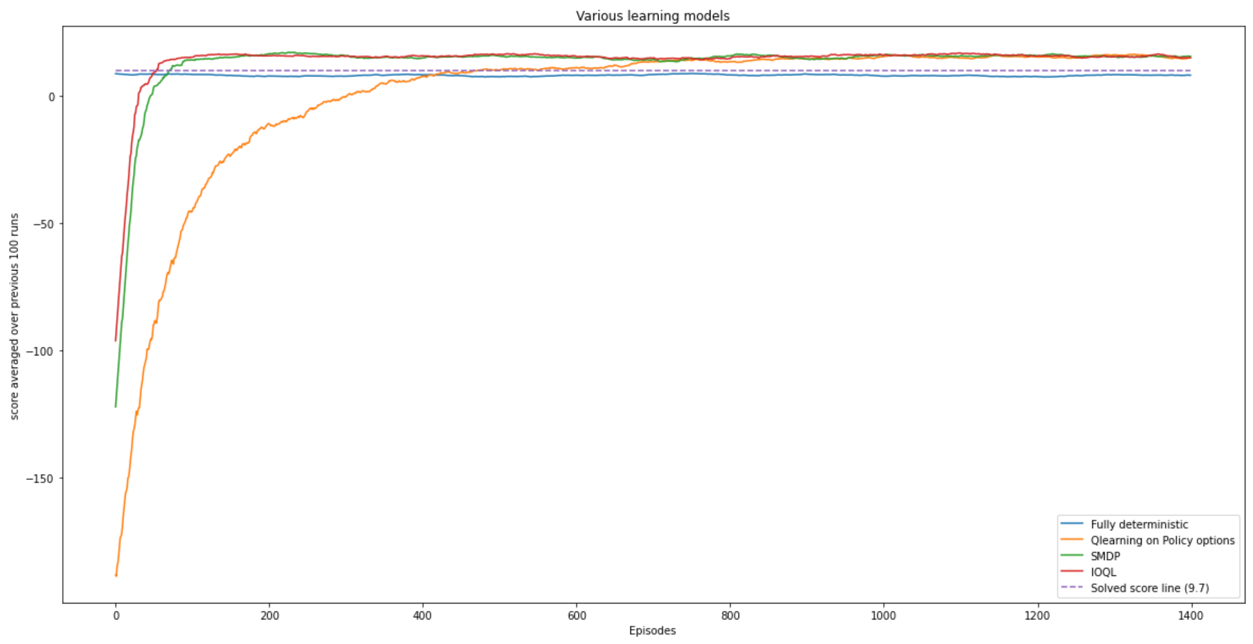


Fig 7: Rewards averaged over 100 episodes for different models

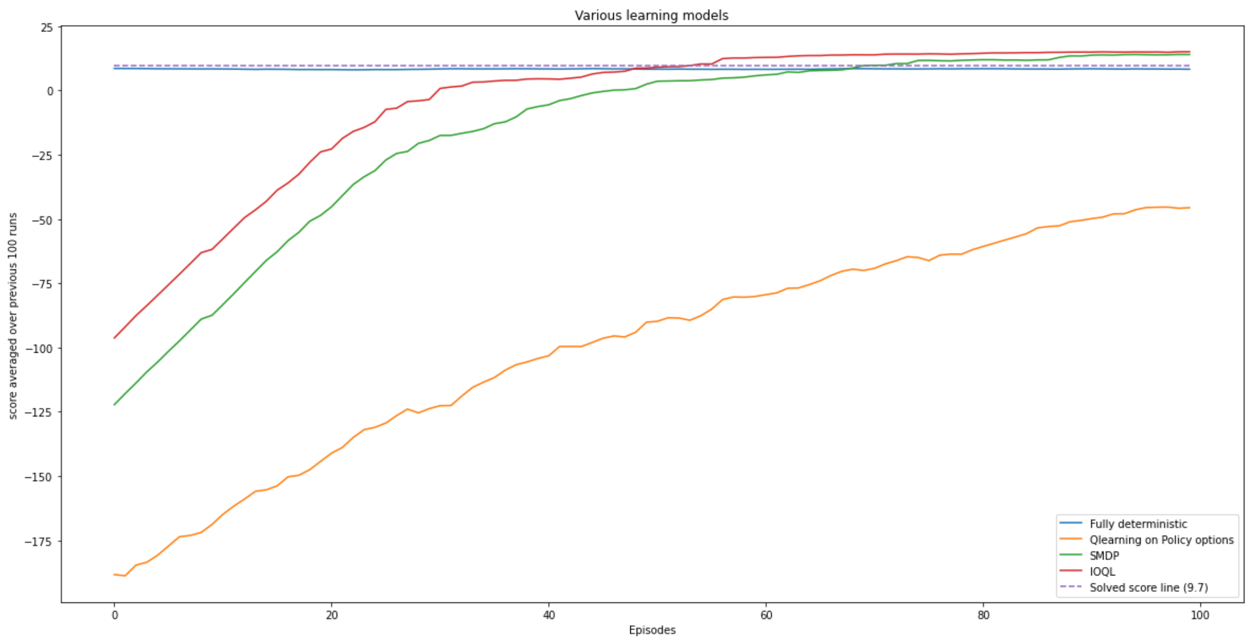


Fig 8: Rewards averaged over 100 episodes for different models: ***Initial stage of learning***

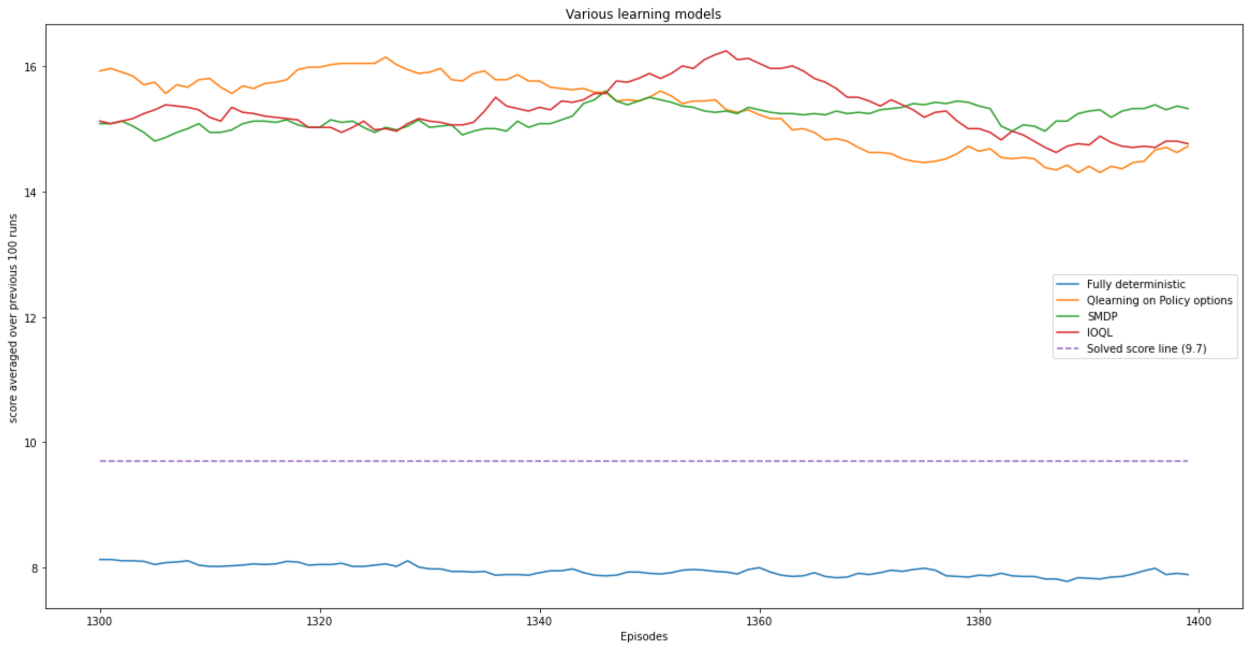


Fig 9: ***Settled values****: rewards averaged over 100 episodes for different models*

Temporal abstraction

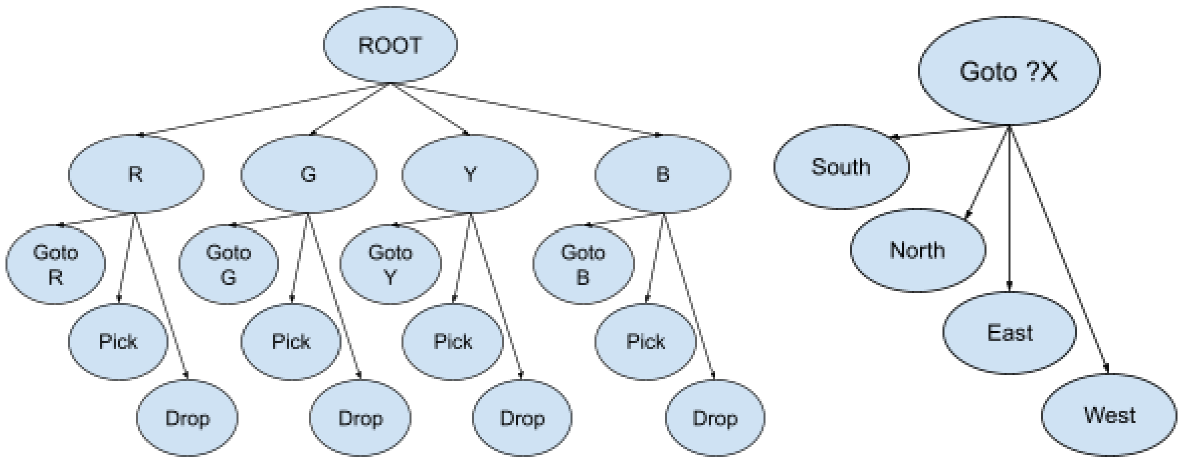


Figure 10: The Temporal Abstraction followed in our approach.

The temporal abstraction used here for solving the given environment hierarchically is given in Figure 10. This is the basic temporal abstraction that can be used. It is certain from the graph that a set of options which are mutually exclusive to the current one is impossible to build so as to solve the environment. Any set of options we choose, some of them are going to have the same termination states same with the ones used here. So, it wouldn't be possible to have a mutually exclusive set of options.

Comparison between SMDP Q-Learning and Intra Option Q-Learning:

SMDP Q-Learning:

- One Q-Learning update is done to the state and option pair at the end of execution of an option.
- A cumulative discounted reward is used for updation and a multi-step updation is done.

Intra Option Q-Learning:

- A Q-Learning update is done on each primitive step of the option and different updates upon termination of the option and intra- option update.
- An update is performed for each option that would have chosen the same action with its policy as the action chosen by the current executing option.
- In a way, it is ensured that multi-step updates are avoided and updates happen frequently for options which choose the same action at a given step.

Insights and Inferences

Figures 3, 4, 5, 6 show the visualizations of learned Q-values and the plots of the reward (averaged over 100 episodes) curves is given in figures 7, 8, 9.

The Best Model

We observe the best performance with Intra Option Q-Learning. The rate of convergence is faster than SMDP Q-Learning. This happens because of the intra option update rule, where updates are performed for every primitive step of the option and multiple options are updated at a time based on their policy selecting the same action as the current executing policy.

Merit of state space compression

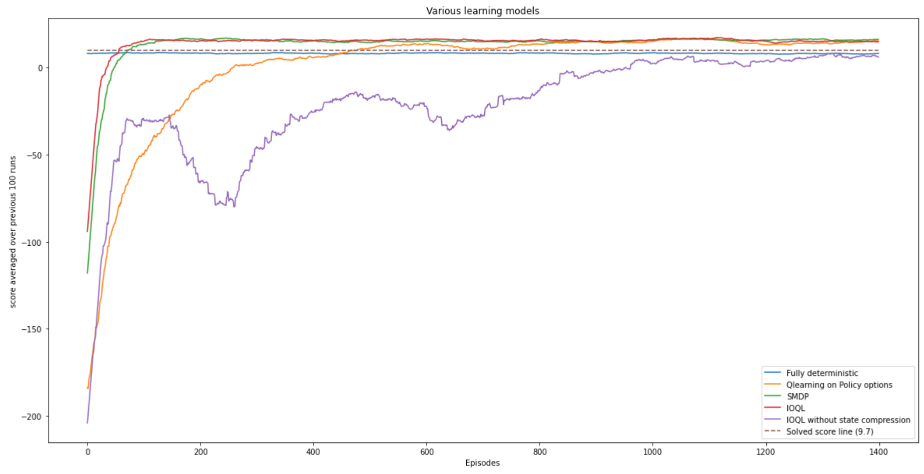


*Fig 11: Rewards averaged over 100 episodes (**uncompressed state space IOQL** included)*

Along with the aforementioned models, we also tried IOQL with 500 states. The worst performance shown by the best algorithm on this 500 size state space shows the merit of our decision to compress the state space to size 20 for the high level q-learning.

Understanding the learnt policies

Each of the states of the given environment encodes the information on taxi row, taxi column, passenger location, destination location. There are a total of 500 states ranging from 0 to 499.

We propose a hierarchical model with a high level policy to choose between options (Moving to a particular destination location). And each of these options have a policy based on which the actions are executed when using the option. The description of these policies for different learning methods (*SMDP Q-Learning, Intra Option Q-Learning)* are given below.

SMDP Q-LEARNING:

- High Level Policy
    1. From figure 4, it is evident that given a state the agent takes the correct next option, the passenger location and destination location are decoded from the state.
    2. Then a sequence of options - to go to Passenger Location, to pick up, to go to Destination Location and to drop off are taken, in order to minimize time.
    3. **For example**: if the passenger location is R and destination is B, then the sequence of options followed by the agent is - goto R (navigating to R and picking up), goto B (navigating to B and drop off).
- Option Policies:
    1. The options' policies are defined in such a way that each option is supposed to start at any position of taxi on the grid and take the taxi to a particular location. After the taxi reaches the corresponding location the option terminates by choosing to pick up or drop off..
    2. The policies learnt are in such a way to reach the desired location in a minimum number of steps. As it is evident from the Q-plots of the options in figure 3.

INTRA OPTION Q-LEARNING:

- High Level Policy:
    1. From Figure 6, which is similar to Figure 4, it is clear that the agent follows/takes the options that take the agent to the goal state in the least amount of time.
    2. **For example**: if the passenger location is Y and destination is G, then the sequence of options followed by the agent is - goto Y (navigating to Y and picking up), goto G (navigating to G and drop off).
- Options Policies:
    1. The options' policies are defined in such a way that each option is supposed to start at any position of taxi on the grid and take the taxi to a particular location. After the taxi reaches the corresponding location the option terminates by choosing to pick up or drop off.
    2. The policies learnt are in such a way to reach the desired location in a minimum number of steps. As it is evident from the Q-plots of the options in figure 5.

Conclusion

In this Assignment, we tried two different Learning Algorithms for Hierarchical RL on the Taxi-v3 environment from OpenAI gym. SMDP Q-Learning and Intra Option Q-Learning and contrasted them with two other methods that involve hardcoding based on human understanding. We conclude that the solutions learnt by machine are way superior than humans for this problem. Intra Option Q-Learning outperforms SMDP Q-Learning because of better usage of the SARS samples (similar to experience replay). Our algorithms even outperform the Hardcoded Agent. We also demonstrated and concluded the strong effectiveness of state compression on the model performance.