CS6700 - RL

SMDP & Intra - Options Q-learning

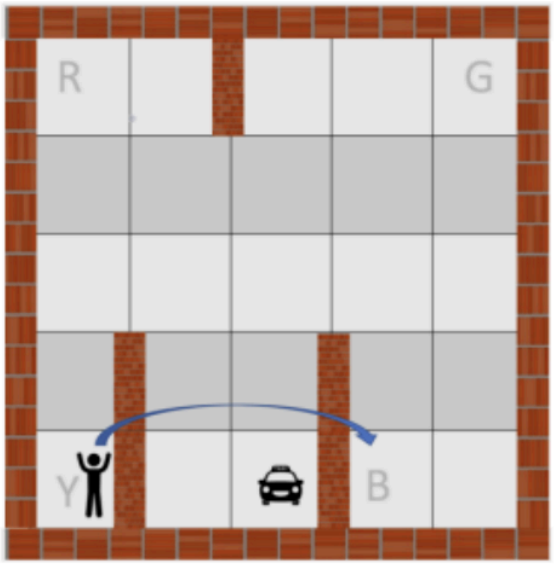
Name => Shaunak Mujumdar

Roll number => CH21B062

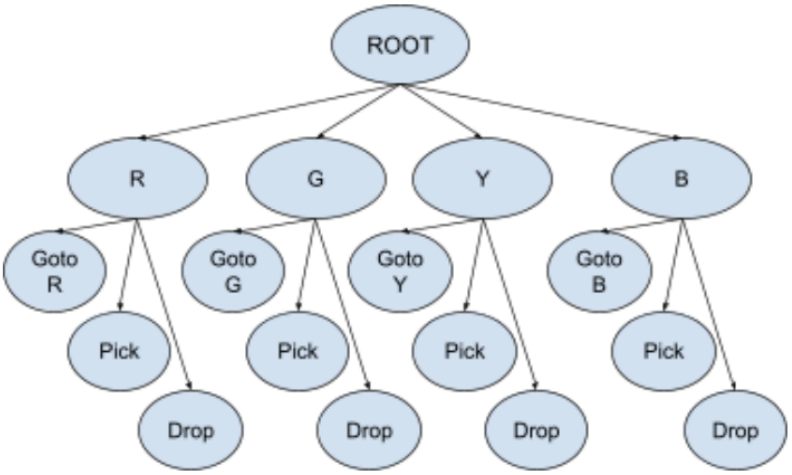
Taxi-v3 environment :

We are trying to solve the OpenAI gym's Taxi-v3 environment using SMDP and Intra Options Q -learning algorithms.

The goal of the Taxi-v3 environment is to pick-up passengers and drop them off at the destination in the least amount of moves.



Options :



Options provide a way to deal with complex tasks by breaking them down into smaller, more manageable units. As mentioned in the paper , this is the general structure of options that we want to use in this problem. Since the destinations are completely different, the option set is completely mutually exclusive.

We will define 4 options (with no other primitive actions) :

- Option 0 : Go to R -
- Option 1: Go to G
- Option 2: Go to Y
- Option 3: Go to B

SMDP Q learning :

SMDP is an **MDP + options** algorithm where execution of option **o** which starts in some state **s** belonging to initialization state set, follows option policy π and jumps to the state **s"** in which **o** terminates. Based on this experience, the **one-step SMDP Q-learning** updates option-value function **Q(s, o)** by

$$Q(s,o) = Q(s,o) + \alpha[r + \gamma^k \max_{o' \in \mathcal{O}_{s'}} Q(s',o') - Q(s,o)]$$

where **k** denotes the number of time steps elapsing between **s** and **s"** , **r** denotes the cumulative discounted reward over this time.

The update mentioned above is basically done at the end of elapsed time steps. We use **reward_bar** to keep track of the total discounted reward. This will be used as the variable r in the q-update.

As you can see from the above update equation, SMDP methods execute an option to its termination before they can learn about it. Because of this, they can only be applied to one option at a time. To overcome this limitation and to learn about an option before the option terminates, intra-option methods are used.

Intra Options Q-learning :

Intra-option methods fall under the category of off-policy learning. The agent learns about options while potentially following a different policy from the optimal one. These methods enable learning about multiple options concurrently, even if only one option is actively being used.

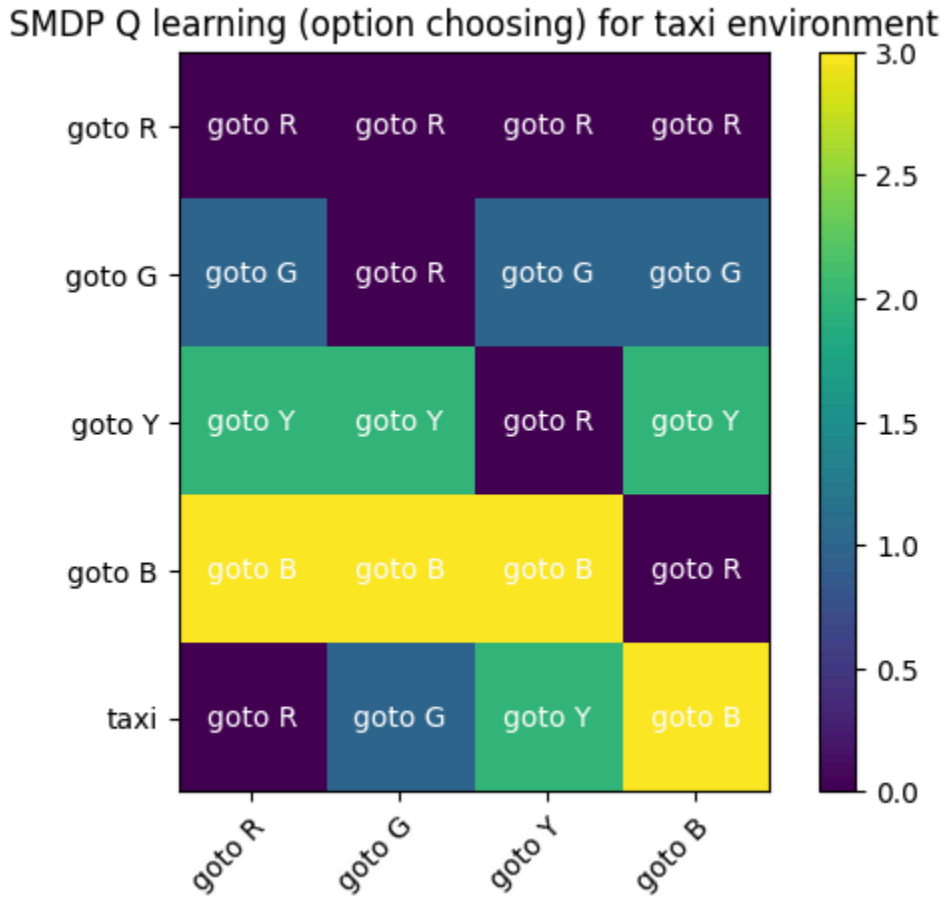
Any experience **(s1,o,s2,r1)** can be used for the following update equations :

$$\begin{aligned} Q(s_1,o) &= Q(s_1,o) + \alpha[r_1 + \gamma Q(s_2,o) - Q(s_1,o)] && \text{If not terminating at s2} \\ &= Q(s_1,o) + \alpha\left[r_1 + \gamma \max_a Q(s_2,a) - Q(s_1,o)\right] && \text{If terminating at s2} \end{aligned}$$

For these reasons, intra options q-learning are potentially more efficient than SMDP methods because they extract more training examples from the same experience

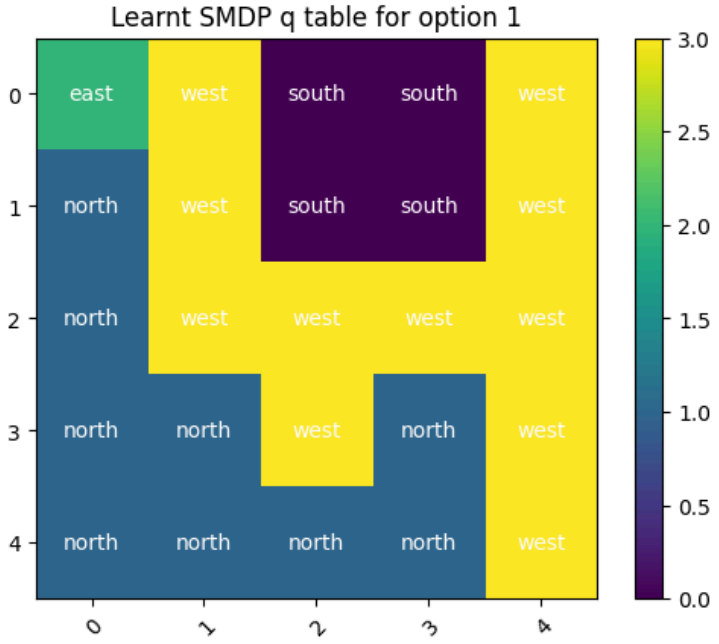
SMDP Q-learning Results :

1. **Passenger location - destination Q values :**



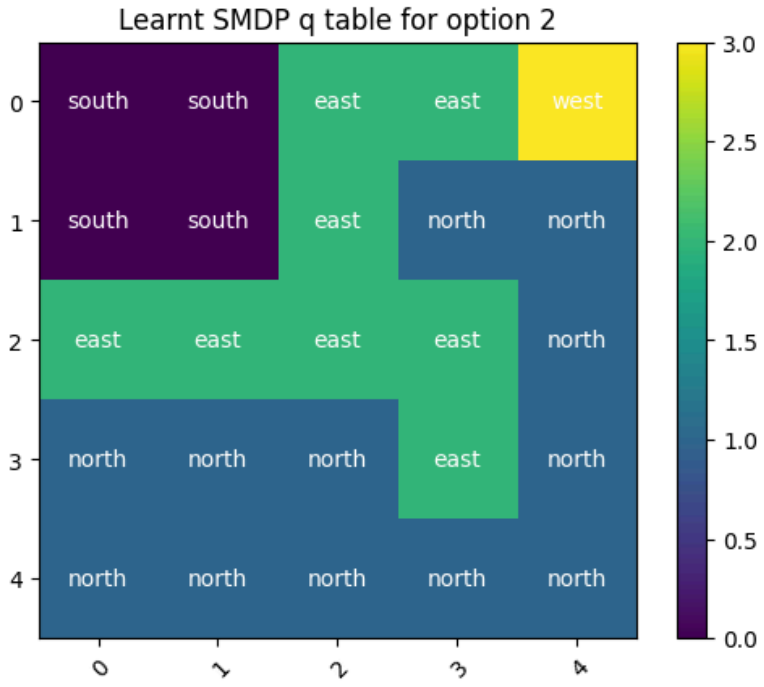
2. **Q-table for Go to R :**

   Here we expect the option to learn an optimal path to R from any location in the grid. The Q table visualized by the policy as shown below.



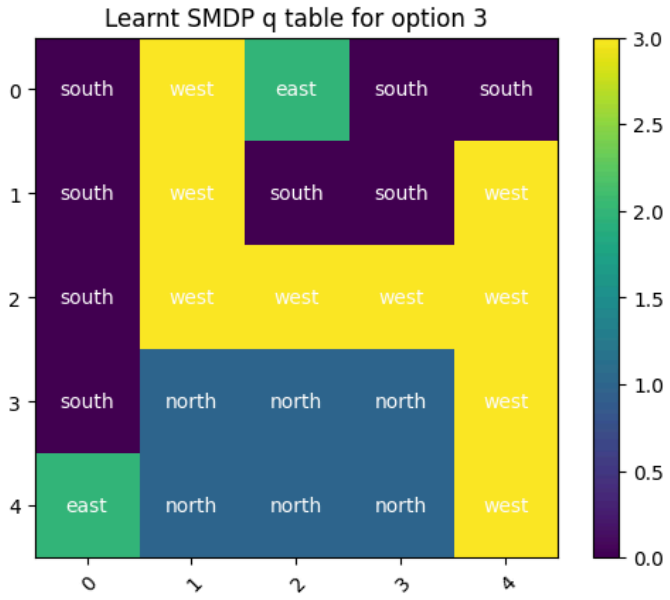
3. Q-table for policy of choosing option 2 => Go to G :

Here we expect the option to learn an optimal path to G from any location in the grid. The Q table visualized by the policy as shown below.



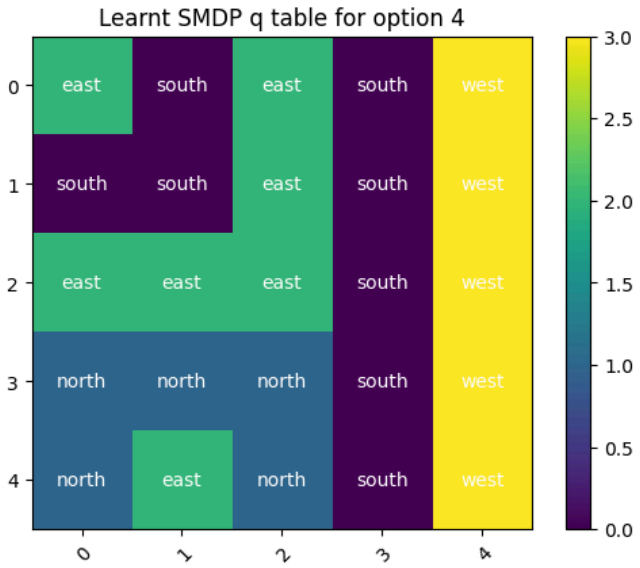
4. **Q-table for policy of choosing option 3 => Go to B :**

   Here we expect the option to learn an optimal path to B from any location in the grid. The Q table visualized by the policy as shown below.



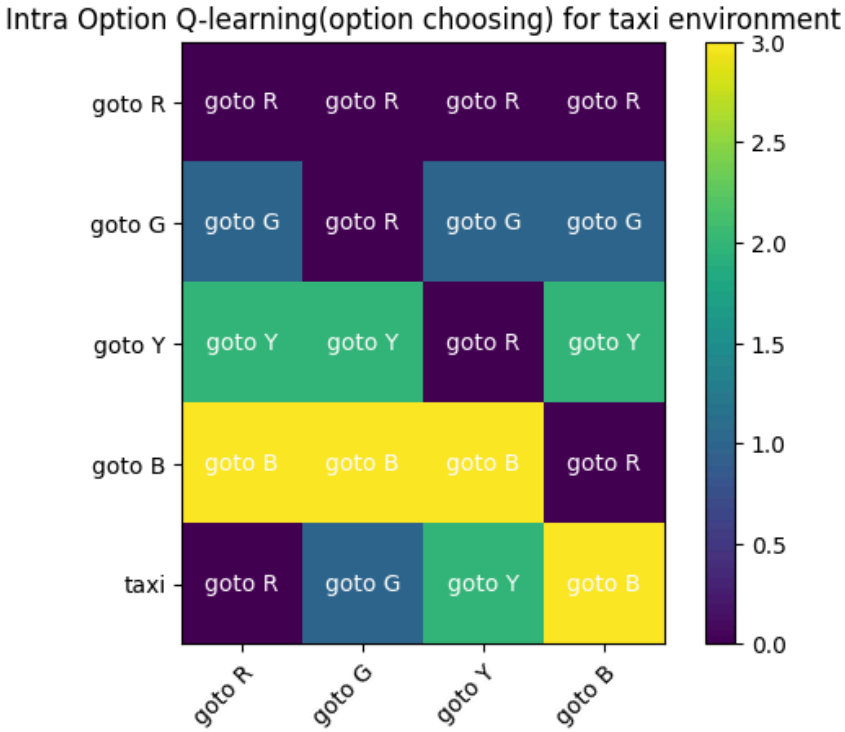
5. **Q-table for policy of choosing option 4 => Go to Y :**

   Here we expect the option to learn an optimal path to Y from any location in the grid. The Q table visualized by the policy as shown below.



Intra Option Q-learning Results :

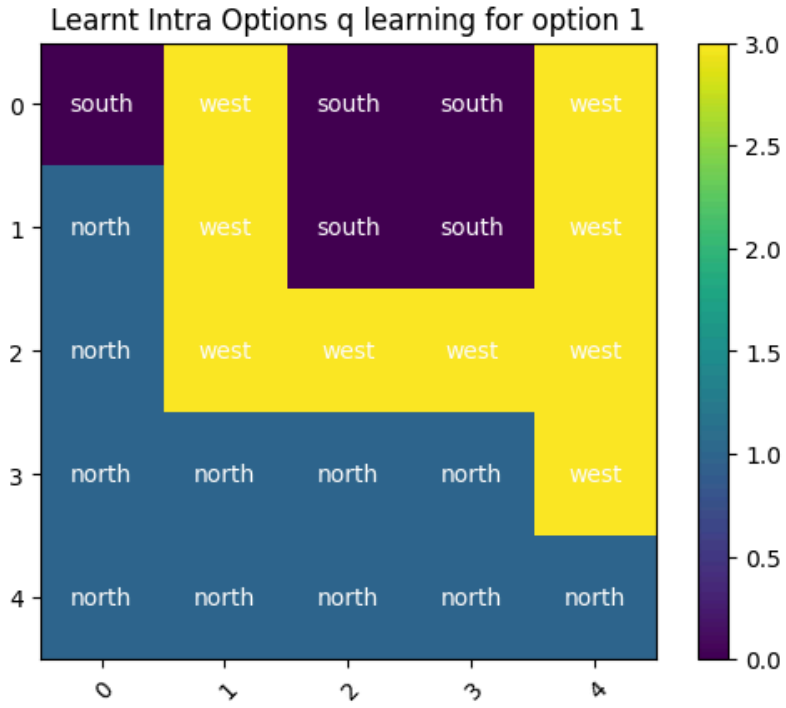
1. **Passenger location - destination Q values :**



a.

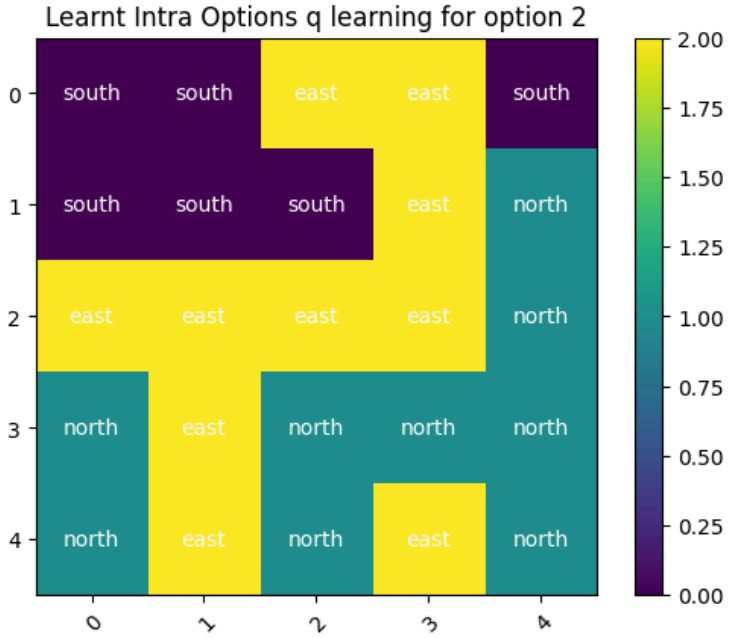
2. **Q-table for Go to R :**

Here we expect the option to learn an optimal path to R from any location in the grid. The Q table visualized by the policy as shown below.



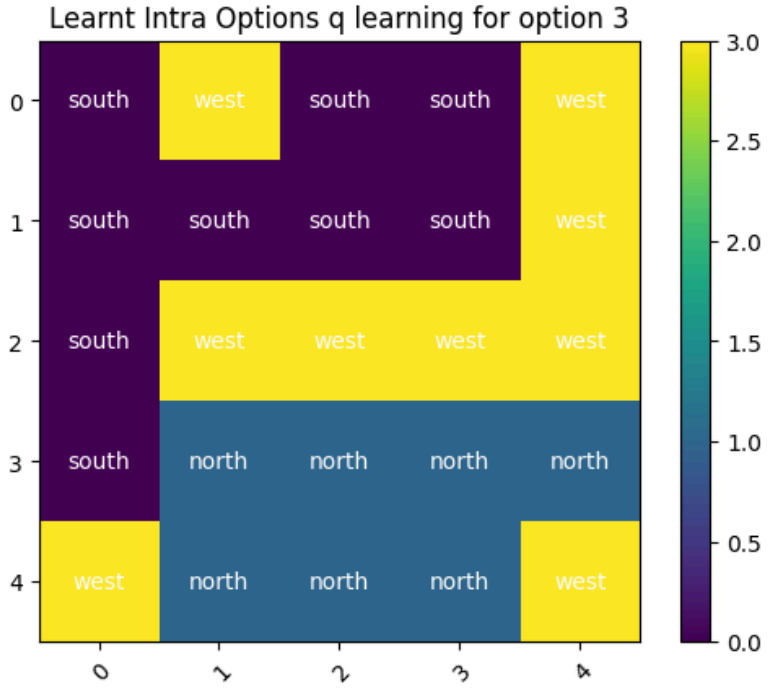
3. **Q-table for Go to G :**

Here we expect the option to learn an optimal path to G from any location in the grid. The Q table visualized by the policy as shown below.



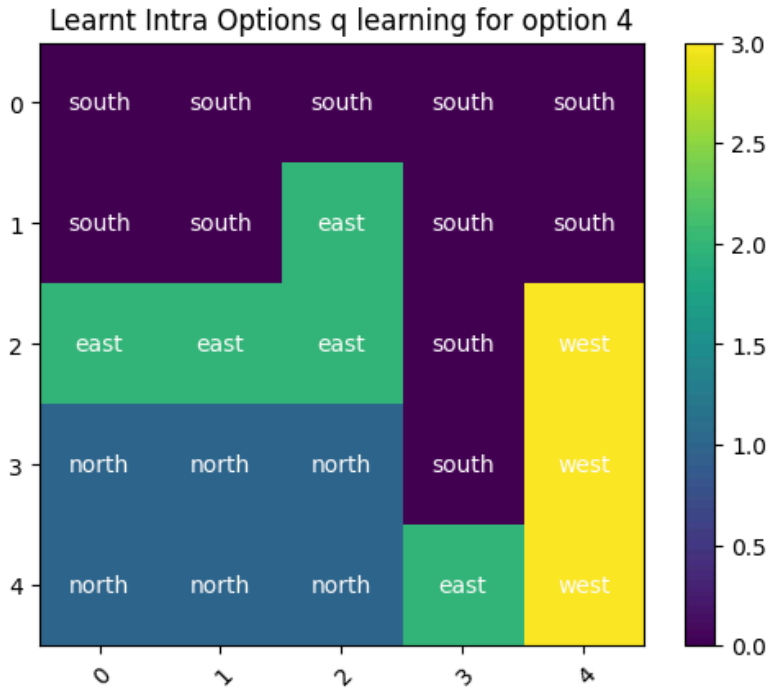
4. Q-table for Go to B :

Here we expect the option to learn an optimal path to B from any location in the grid. The Q table visualized by the policy as shown below.



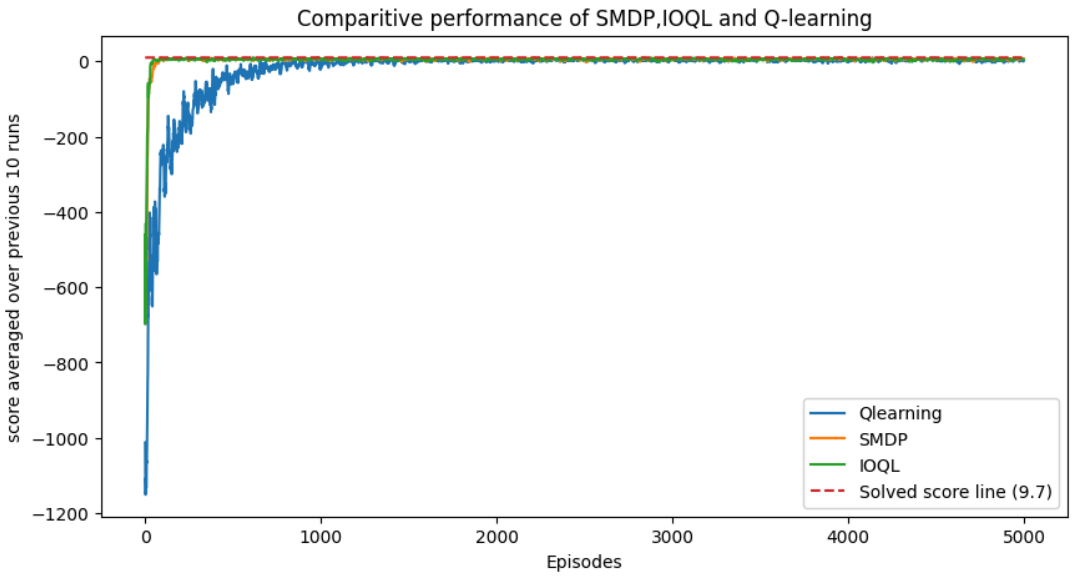
5. **Q-table for Go to Y :**

Here we expect the option to learn an optimal path to Y from any location in the grid. The Q table visualized by the policy as shown below

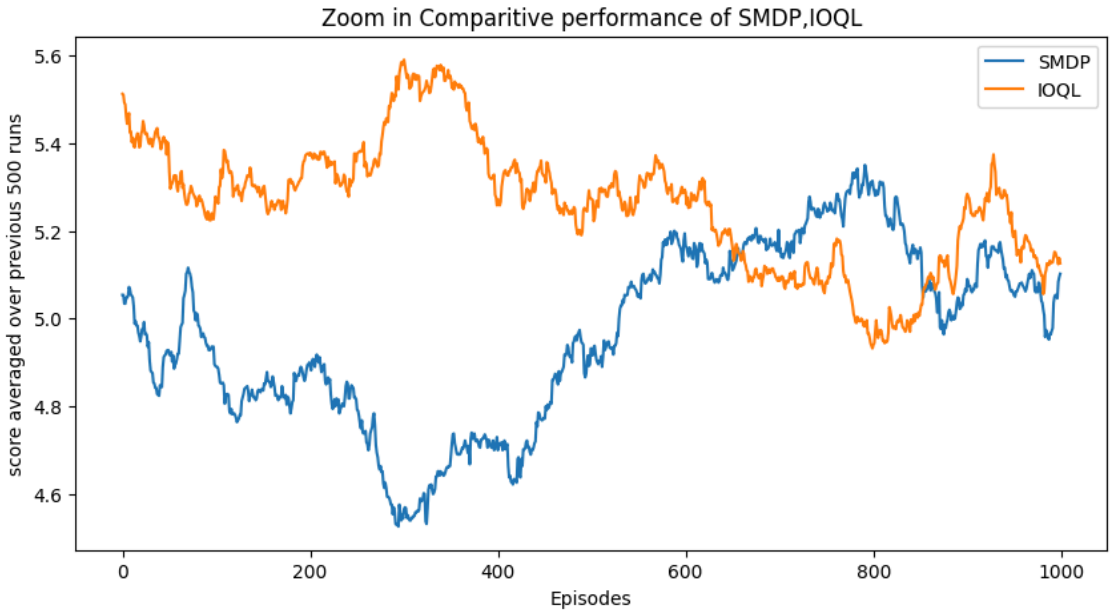


<u>Comparative performance between different methods</u> :

Reward curves :



Zoom in on a part of the reward curves :



Best Model :

As expected,the best performance was shown by Intra Option Q-Learning. The rate of convergence is faster than SMDP Q-Learning.

As mentioned earlier, this is because learning updates are done while the option is still being performed. Multiple options are updated at the same time. So intra-option methods can extract more training data from a single experience.

Also it can be observed that both option based algorithms - SMDP and IOQL outperformed the simple Q-learning algorithm. Both of them converge way faster than simple Q-learning algorithms too.

This shows the effectiveness of hierarchical RL with executable options.

Dividing the Search Space :

Taxi-3 is environment where we have total 500 states such that 25 taxi positions * 5 passenger positions*4 goal destinations.

In order to implement SMDP and IOQL, we basically wanted two different policies - one to choose the next best option which will require **(passenger_location,destination)** and one to take the taxi to that desired location in a minimum number of steps which will require **(x,y)** taxi positions . For that we decided to maintain multiple Q-tables with divided state search space.

So basically we will be dividing the search space of states into 2 different parts :

1. Range of (passenger_location,destination)
2. Range of (x,y) for each of the 4 options to reach the particular destination.

Description of Policies :

We can observe that :

- Both SMDP and Q-learning managed to learn the same choice of options for different passenger positions and their goal destinations. (which is optimal)
- As for the option policies learnt for each option, the policies learnt by SMDP are different from those in Intra option Q-Learning.

As mentioned before , for both SMDP and IOQL we are deploying two different policies - option choosing policy and taxi moving policy.

- **Option choosing policy :**

  From the "**Passenger location - destination Q-values heatmap"** that you can see above, SMDP has managed to learn the optimal option for all the passengers location and destination combinations. In the code, **q_values_smdp** and **q_values_IO** store the policy learnt here.

- **Taxi moving policy for each option :**

  For each of the 4 options, we define a Q-table for moving the taxi in the minimum number of steps.

  You can see this policy in the heatmap of "**Learnt SMDP Q-table for all options**" and "**Learnt Intra options Q-table for all options**" .

  After the taxi reaches the corresponding location the option terminates by choosing to pick up or drop off.

  In the code, **Q_hat_smdp** and **Q_hat_IO** are dictionaries to store the policy learnt here for each of the options.

Alternate set of Options :

The alternate set of options we can use to solve this environment are :

1. Go north till you hit the wall
2. Go south till you hit the wall
3. Go east till you hit the wall
4. Go west till you hit the wall

These options are mutually exclusive to the given options since the objectives of both of these option sets are different.

Code :

https://colab.research.google.com/drive/16ZDASuVFqwm_9LCS-sQwlyh8ViexToJ9?usp=sharing

Github link :

https://github.com/Gilgamesh60/PA3_CH21B062