CS6700 Reinforcement Learning
Programming Assignment 3

Keerthi Vasan M (ED21B034), Gokul M K (ED21B026)

Please find the code in this GitHub repository: GitHub Repository

1 Introduction: Environment

The taxi domain environment provides a dynamic and challenging scenario for reinforcement learning tasks. It is represented as a 5x5 grid, where each cell corresponds to a position that the taxi can occupy. The environment features a single passenger who can be picked up or dropped off, or is currently being transported by the taxi. There are four designated locations in the grid world, each indicated by a distinct color: Red (R), Green (G), Yellow (Y), and Blue (B).

The environment encompasses a total of 500 discrete states, determined by the 25 possible taxi positions, 5 possible passenger locations (including the case when the passenger is in the taxi), and 4 designated destination locations. However, only 404 states are reachable during an episode, excluding situations where the passenger is already at their destination or immediately after a successful episode where both the passenger and the taxi are at the destination.

1.1 Passenger Locations

- 0: Red (R)
- 1: Green (G)
- 2: Yellow (Y)
- 3: Blue (B)
- 4: In the taxi

1.2 Destinations

- 0: Red (R)
- 1: Green (G)
- 2: Yellow (Y)
- 3: Blue (B)

1.3 Rewards

- A penalty of -1 per step, unless another reward is triggered.
- A reward of +20 for successfully delivering the passenger to their destination.
- A penalty of -10 for executing "pickup" and "drop-off" actions illegally.

The environment features a discount factor of $\gamma = 0.9$.

1.4 Actions

The environment offers six discrete deterministic actions:

- 0: Move south
- 1: Move north
- 2: Move east
- 3: Move west
- 4: Pick up passenger
- 5: Drop off passenger

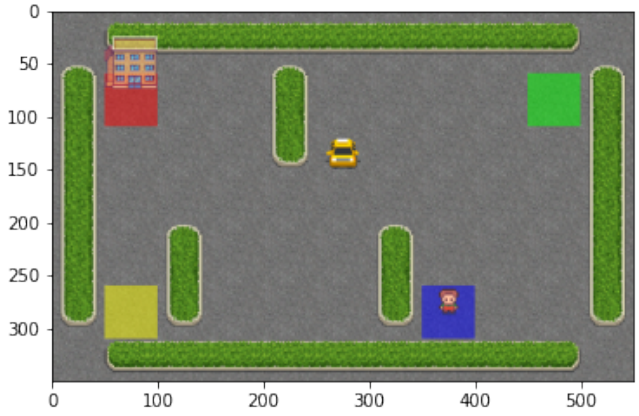

Figure 1: Taxi Environment V3

2 Hierarchial Reinforcement Learning

2.1 Options

An Option is a tuple <I,π,β>. I contains the initiation states where the option starts. π is the policy to be followed when the initiation states are encountered till the terminal state(of the option).β is the termination probability of the state which in this environment is deterministic (1 or 0). A true reinforcement learning agent should be able to find the inititation states and termination probabilities. Two commonly used methods to learn options are:

- 1: SMDP Q learning
- 2: Intra Option Q learning

The above two learning methods are used to solve the taxi problem in this report.

2.1.1 Options used in the report

Options to move the taxi to each of the four designated locations, executable when the taxi is not already there

3 SMDP Q Learning

The state value function in a SMDP is extended to accomodate macro actions, go to 4 colours(R,G,Y,B).If primitive action was selected in a state , the value of the state action values are updated via regular q learning. In our report , we only learnt option to solve the

environment and not the primitive actions.If the agent selects an option, not state action values are updated untill option terminates. The cumulative discounted return is used to update the value of the option in the state S in which the option was initiated. The SMDP Q-learning update rule can be represented as follows:

$$Q(s,o) = Q(s,o) + \alpha \left[R + \gamma^k \max_{o'} Q(s',o') - Q(s,o) \right]$$

where:

$$R = \sum_{k=0}^{T} \gamma^k r_{t+1+k}$$

```
Q_smdp[dropgridno_old, action] += alpha*(reward_bar+(gamma**k)*np.max(
    Q_smdp[dropgridno_new, :])-Q_smdp[dropgridno_old, action])
taxi_row, taxi_col, passenger_location_id, drop_location_id = env.decode(
    state)
```

Figure 2: SMDP Update Step

3.1 Methodology

There 5x4 substates (for 5 passenger locations and 4 drop locations). According to this state an option which is learned is triggered. The 4 options defined earlier terminate either when it reaches the pickup location or drop location.When it reaches the pickup location or drop locations , the action to pick or drop is chosen deterministically and not learned. Each option table has 25 states corresponding to the cells of the environment(5x5).

3.2 Results and Inference

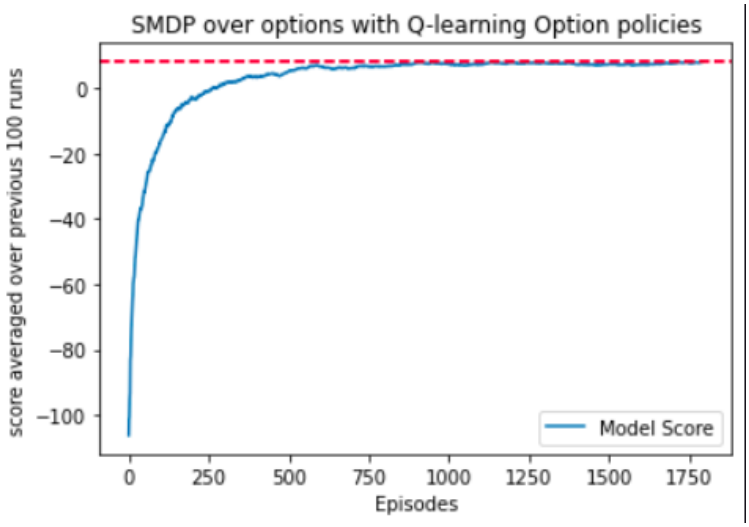

Figure 3: Learning Curve for SMDP

α is the paramter used for this training which is 0.1, we can see that it reaches close to the reward threshold around 800 episodes. This is bit delayed as compared to the intra option q learning which we will come across later. It always executes the optimal policy after training. The delay is due to the fact that the update for the smdp q values happen only when the option is finished and not before that. The final learnt policy is as follows:

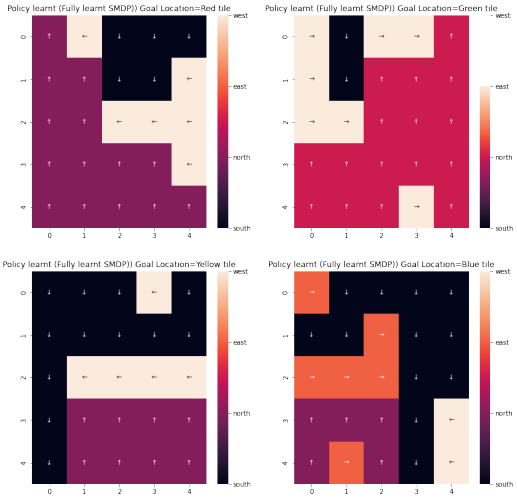

Figure 4: Option policy for each R,G,Y,B goal states

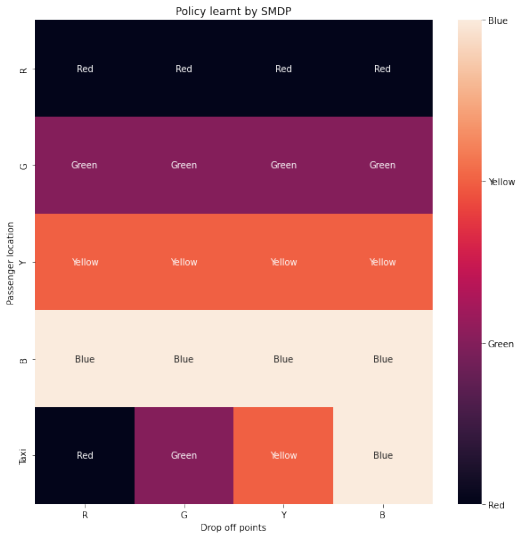

Figure 5: Policy learnt for each passenger and drop location

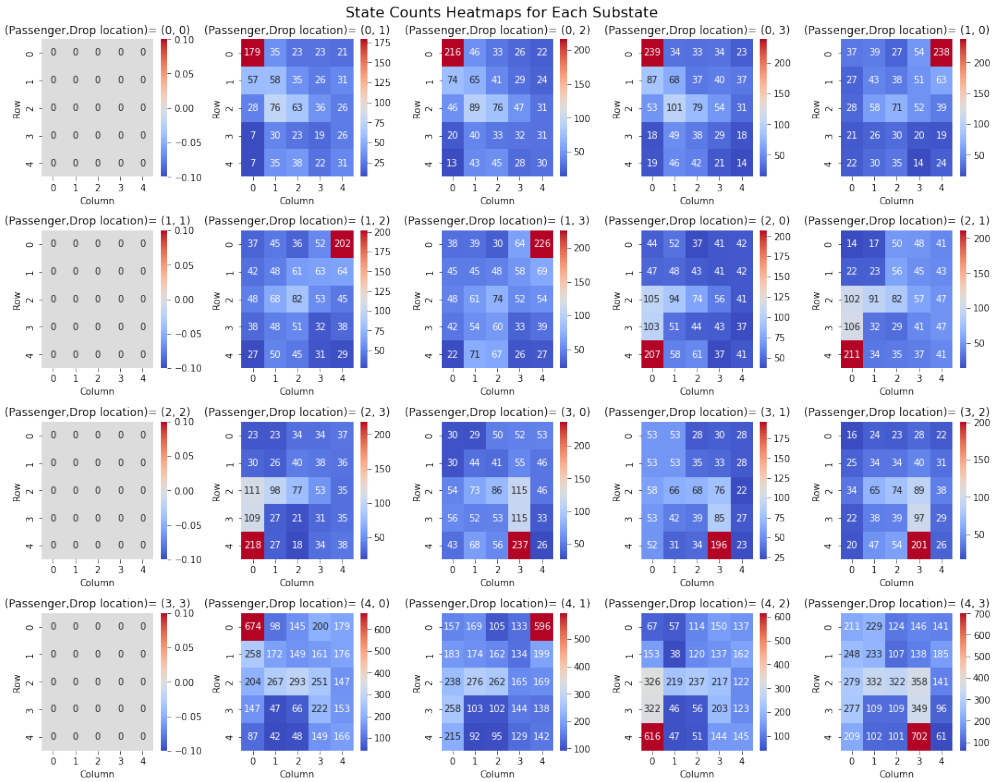

Figure 6: State Visits

The arrows in the policy diagram shows the policy the agent will follow to reach respective goal location. The SMDP Q learning table shows for what passenger location and goal location which option will be followed. The state counts shows how many times a give state is encountered(not same as the state updates).

4 Intra Option Q Learning

Intra Option is an example of off-policy learning methods because they learn about the consequences of one policy while behaving according to another policy. Intra Option simultaneously learn about many different options from the same experience

4.1 Drawbacks of SMDP

SMDP Learning need to execute an option to termination before they can learn about it. Due to this, only one option can be applied at a time, the one that is being executed. This led to the motivation for intra-option learning. Intra Option learning is potentially more efficient than SMDP learning because they extract more training examples from the same experience.

4.2 Intra Option Update

Suppose primitive action a_t is taken in s_t, and s_{t+1}, r_{t+1} are the next state and immediate reward

For every Markov Option o $= < I, \pi, \beta >$, the update is:

$$Q_{k+1}(S_t, o) = (1 - \alpha)Q_k(s, o) + \alpha[r_{t+1} + \gamma U_k(s_t, o)]$$

where:

$$U_k(s, o) = (1 - \beta(s))Q_k(s, o) + \beta(s) \max_{o'} Q_k(s, o')$$

```
if temp_optact == optact:
    u = (1-temp_optdone)*q_values_option[Sub(next_state), i] + (temp_optdone)*np.max(q_values_option[Sub(next_state), :])
    q_values_option[Sub(state), i] = (1-alpha)*q_values_option[Sub(state), i] + alpha*(reward+gamma*u)
    state_count_option[Sub(state), i]+=1
```

Figure 7: Intra Option Update Step

In case of deterministic Option Policies, this update is applied to all options whose policy selects action a_t in s_t, whether that option was executing or not.

If all the Options are deterministic and Markov, then for every option, the function converges to optimal Q with probability 1, in the limit every primitve action is executed infinitely often in every state.

4.3 Methodology

The Taxi-v3 environment is used to learn the Options, earlier with SMDP Learning, now with Intra-Option Learning. The Option is to pick for every state action epsilon greedily which is being decayed to ensure exploration at the beginning to exploitation gradually. While at the Goal States, which are the R, G, Y, B locations, depending on the environment state decoding, the passenger is either picked or dropped deterministically, or perform a random primitive taxi movement out of the 4.

4.4 Results and Inference

In the update process, it is evident that intra-option updates the state-action value dynamically during learning, showcasing the significant efficiency of intra-option compared to SMDP, as indicated by the graph provided. The learning process utilizes a step size parameter of $\alpha = 0.1$ and a discount factor of $\gamma = 0.9$. The Environment's Reward Threshold is achieved rapidly, typically within approximately 150 episodes. Additionally, provided below is the state visit probability for each option implemented through intra-option learning. The heatmap closely mirrors the probability distribution that a typical planning algorithm would follow to reach either the passenger or drop-off location.

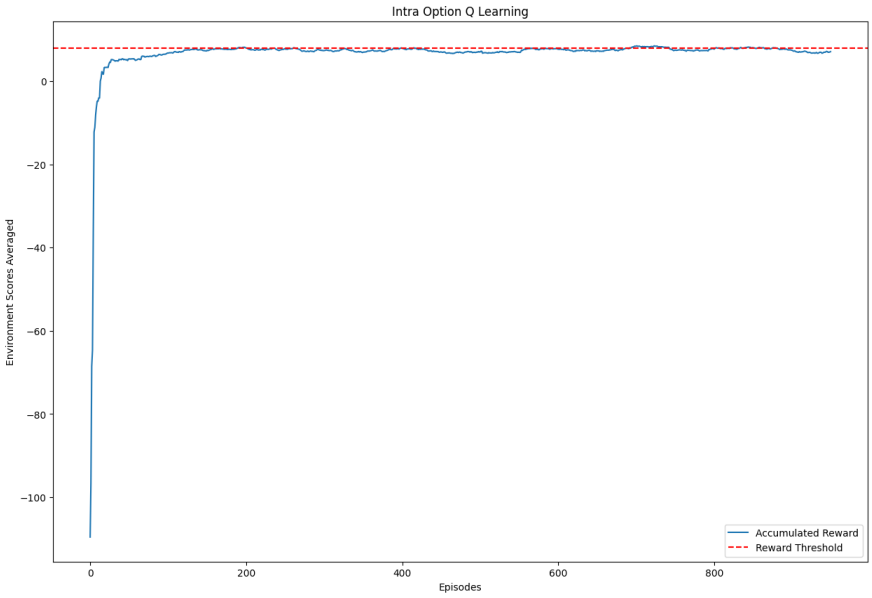

Figure 8: Intra-Option Learning Curve

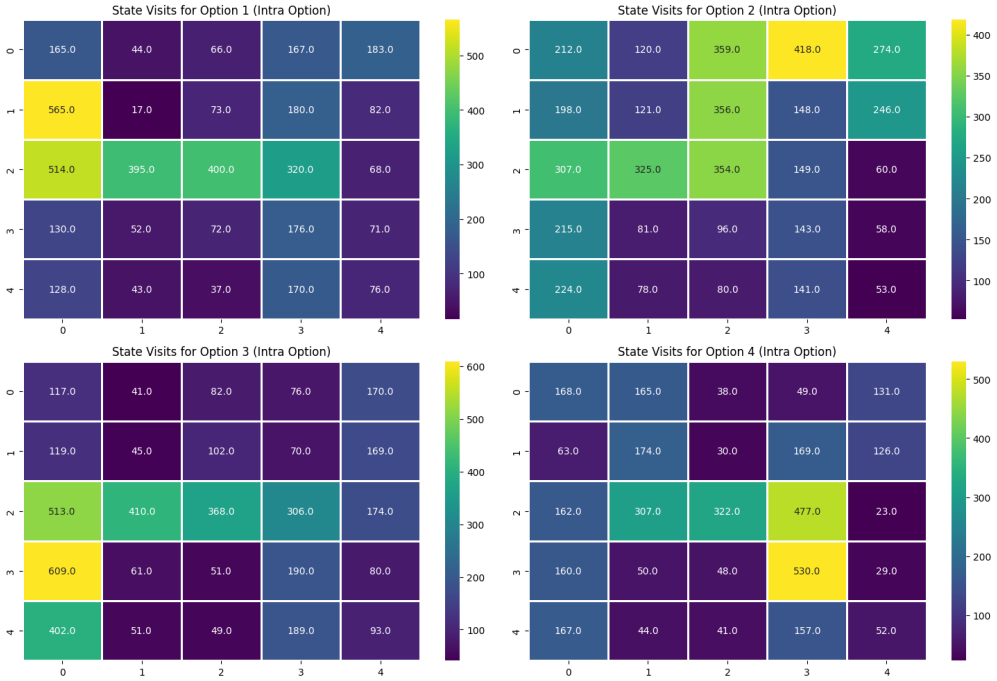

Figure 9: State Visit Probability (Intra Option)

4.5 Comparison (Hardcoded states vs Intra Option Learning)

The heatmap presented below contrasts the actions executed by the Agent acquired through intra-option learning against a predefined state-to-action mapping derived from intuition for each state and its associated option. Upon examination, a strong correlation was observed

between the majority of states and their corresponding actions, with a few exceptions. These discrepancies could potentially influence the optimal action during Option Termination.

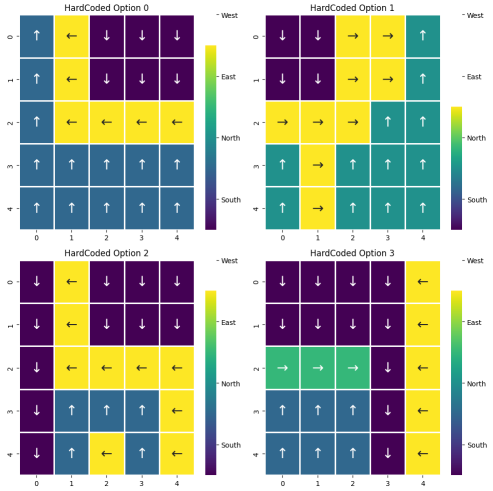

Figure 10: Hardcoded State Action Map

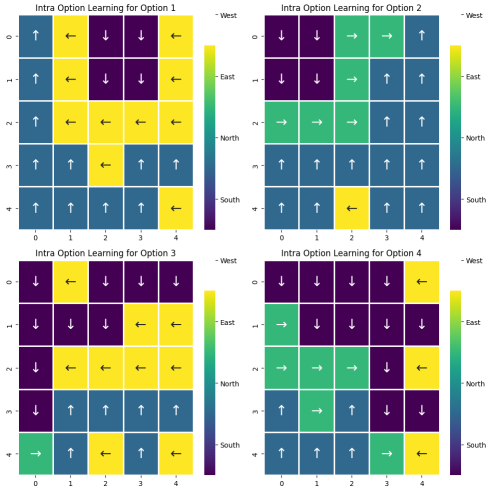

Figure 11: Agent's Action Map (Intra-Option)

4.6 Alternate option 1

The given option(In the question) is the best option for this environment. Some options come close to solving it. One such option we have defined is the move N,S,E,W two steps instead of one step($\alpha = 0.1$and decay=0.9 for SMDP and $\alpha = 0.5$ and decay=0.9 with $\epsilon = 0.1$ in both cases. This is disjoint from the primitive actions, since we dont learn primitive actions and if the taxi moves it will be by 2 steps. This indeed solves for some substates but not all it has high probability of finding a solution but it may not be optimum(around 96% of the time solution is found).The maximum reward we could get is -25 , this is due to fact that we are not able to control the explorations since option ends after 2 steps or when the goal is reached or a passenger is picked. The environment is 5x5 which is odd, and we are traversing by 2 steps and for some cases it may demand us to move 2+2+1, where one is not possible. This prevents the agent from learning is certain states.

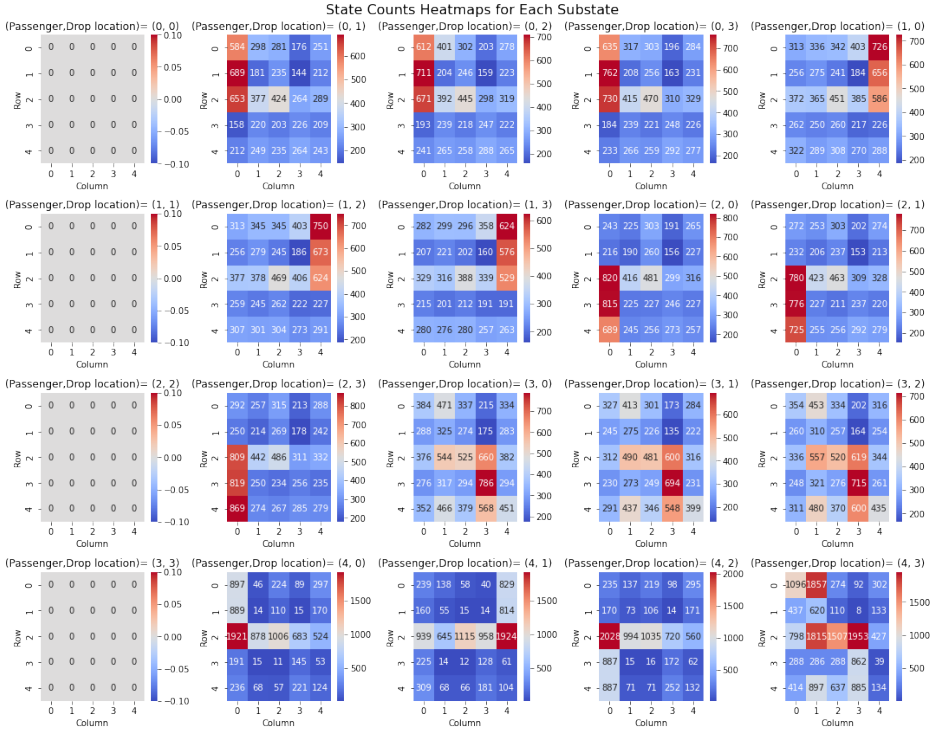

Figure 12: State Visits for SMDP

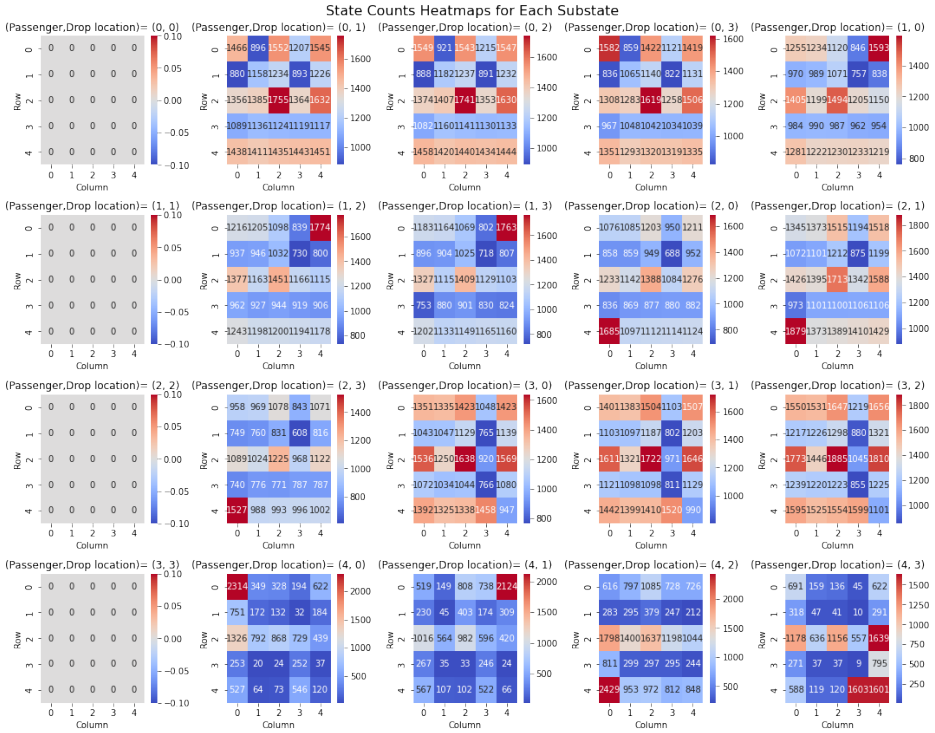

Figure 13: State Visits for Intra

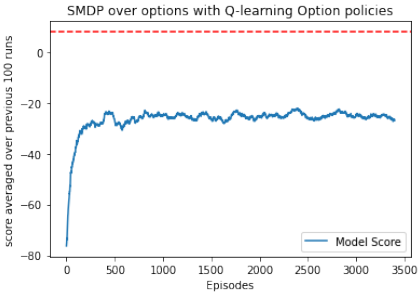

Figure 14: Training Curve for SMDP

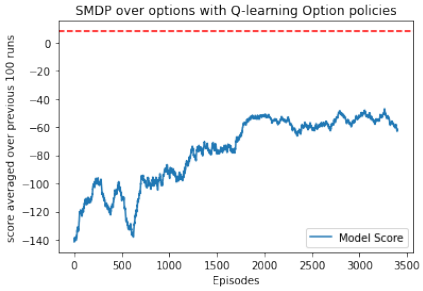

Figure 15: Training Curve for Intra

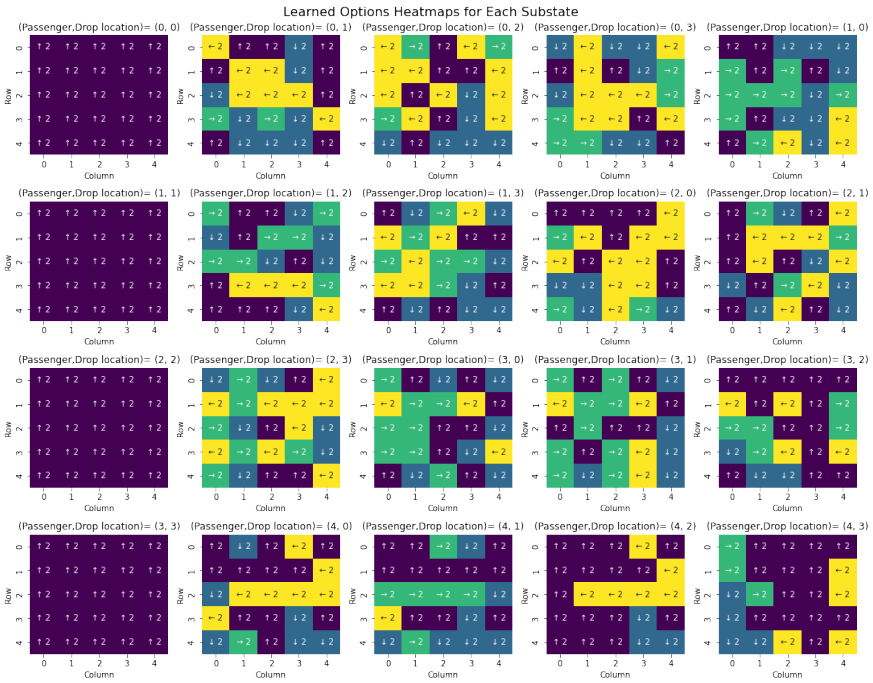

Figure 16: Policy learnt for SMDP

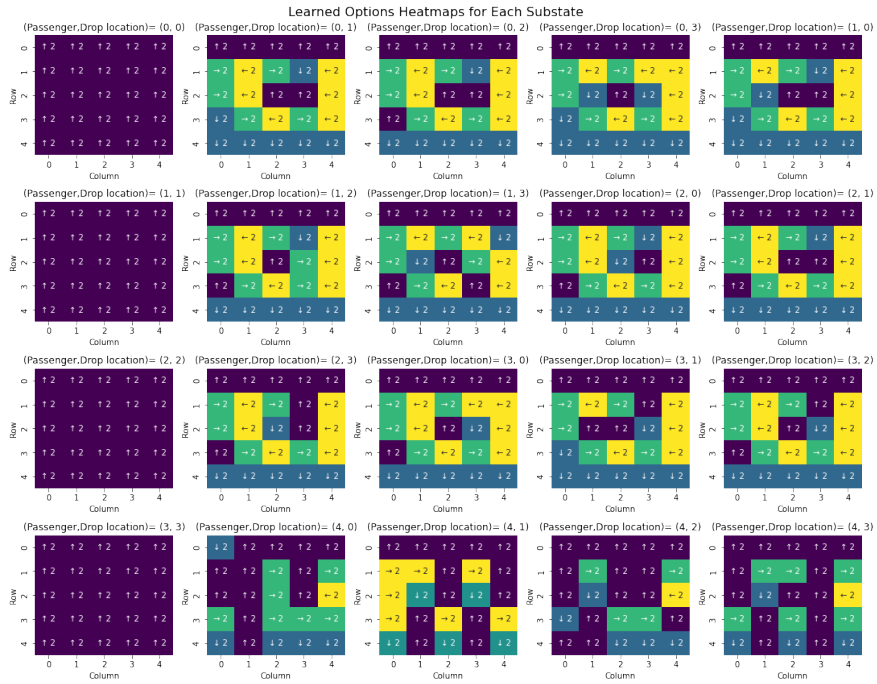

Figure 17: Policy learnt for Intra

From the above diagrams we can observe that intra update has a very high variance, this could be due to uneven exploration caused due to two step policy and the policy learnt is bad. We can see that a lot of states in the top part is not explored(different actions in them) that well due to which policy is not learnt properly.This is again due to 5x5 nature , and even 2 step movement. If we are able to make the steps dynamically varied we could learn a good policy which we could see from the learning curves. Use of bottlenecks could potentially improve the learnt policy in both cases(SMDP and Intra).

4.7 Alternate Option 2

We have explored alternative option definitions to replicate option learning. Through experimentation, the options learned have proven to be optimal and closely resemble the intuitively ideal actions for each state. Within this alternative option framework, states have been categorized accordingly.

$$\textbf{Bottleneck States:}[2,0],[2,3],[2,4](Red)$$

$$\textbf{Crucial States:}[0,2],[1,2],[4,1],[3,1],[4,2],[3,2](Green)$$

$$\textbf{Middle States:}[2,1],[2,2](Purple)$$

In these states, actions are selected using epsilon-greedy methods based on two possible actions for each state. For these states, actions are taken epsilon greedily on two actions based the states. For example, in state [3, 1] which is a crucial state (because of its proximity to the wall), the best action would be to move south, similarly in [2, 1] which is a middle state, the best action would be to move either west or east depending on passenger or drop location.

With $\epsilon = 0.1$, the average reward earned is -9.4 which is very close to the Reward Threshold for this environment.

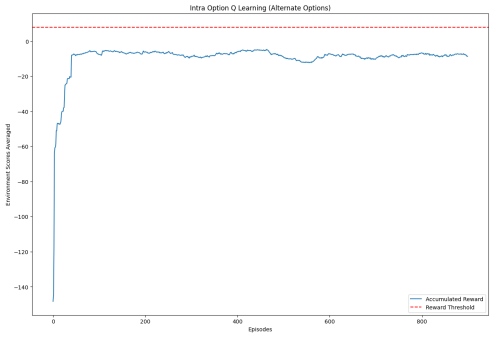

Figure 18: Intra Option Learning with Alternate Options

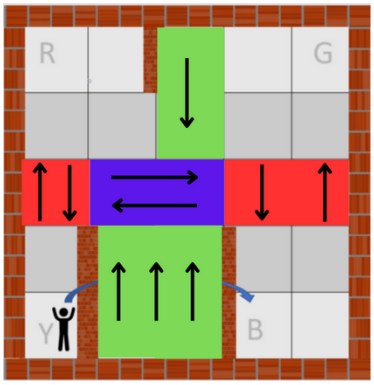

Figure 19: Alternate Options Grid

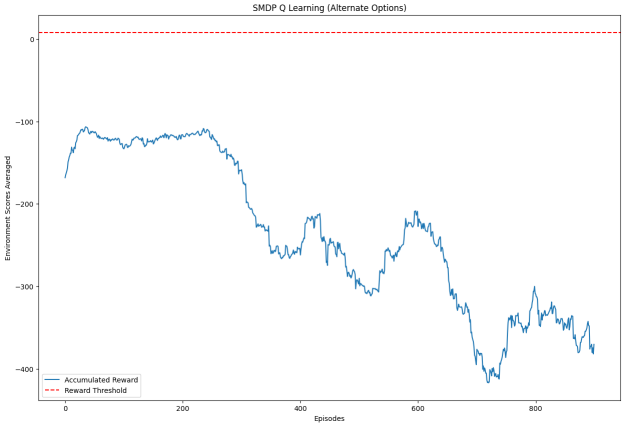

Figure 20: SMDP Option Learning with Alternate Options

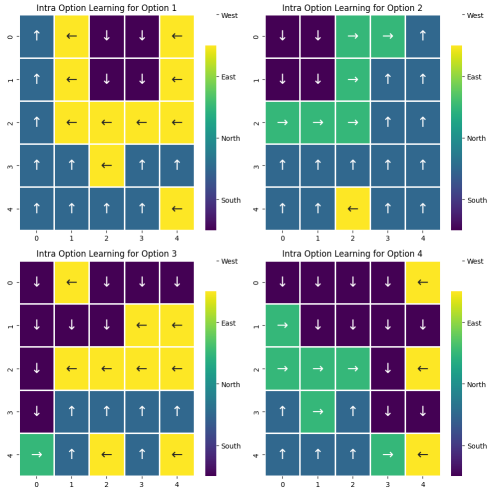

Figure 21: Agent's Action Map (Option Defined)

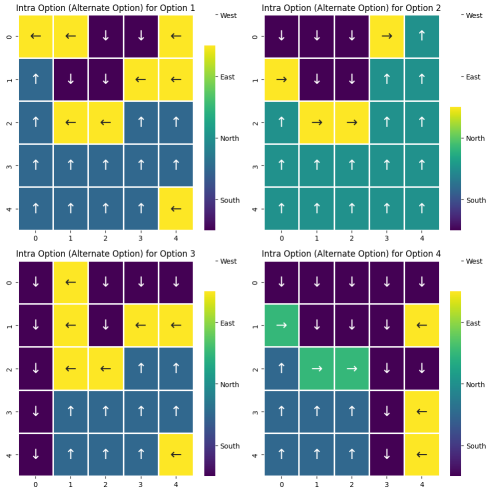

Figure 22: Agent's Action Map (Alternate Option)

5 Comparison (SMDP vs Intra-Option)

Experimental Setup: A comparative study was conducted to evaluate the efficiency of an Agent learning through SMDP against Intra-Option Learning. The step size parameter was set to $\alpha = 0.1$, and the discount factor was set to $\gamma = 0.9$ for both learning methods. The environment was subjected to learning for 1000 episodes, and a Reward Threshold of 8 was specified, in accordance with the Open AI Gym Taxi-v3 environment.

Inference: Agent learning by Intra Option update method has improved characteristics over learnt via SMDP Learning. This can be inferered from the grap too, as it takes around 800 episodes to touch the Reward Threshold line for the first time, whereas it is around 150 for Intra Option learning method

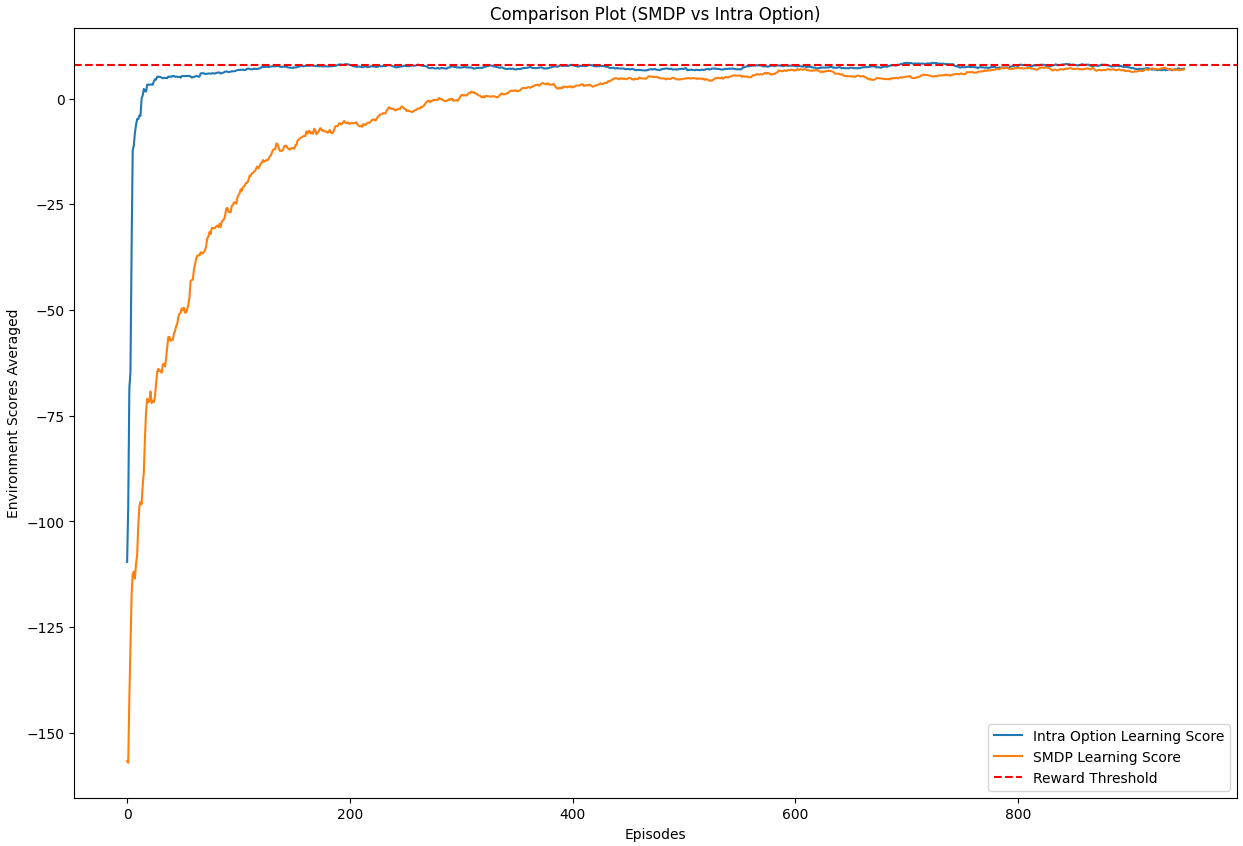

Figure 23: Comparison Plot (SMDP vs Intra Option Learning)

6 Final Inference:

1. The Agent learned through SMDP methods updates the Q values of the start state of the option using the terminated option state only after the option has terminated. This

update process can take a considerable amount of time, depending on the effectiveness of the option.

2. Intra-option learning updates the Q values while the option is being executed. This is achieved by including a condition on the termination probability.If the option is terminated, the max of the Q value over the Options is used in the update step, whereas while executing the option, the one-step Q update is used.This makes faster updates as soon as an action is taken in the state rather than waiting for the option to complete for the update.

3. Choice of options plays a very important role in finding the solution if the chosen option is bad both the methods will fail.Which can be clearly seen for the go to colour option and the different alternate options given in our report.

4. Another important thing to note is the exploraration tradeoff while using options. If an option doesnt traverse properly through required states it wont learn a good policy(which can be seen from the 2 step option case which misses out some state action pair update which even though has capability it doesnt learn good policy and fails in some cases and also doesnt find the optimal solution in most cases). So while learning an option this is to be taken note of.