RL Programming Assignment #3

ee21b124 Shivam, ee21b144 Sujal

May 2025

Note

1. Python version used is 3.11.9
2. Reward function used in each environment is same as what given its documentation.
3. Modifying the reward function to incorporate additional factors such as position, velocity, or other state variables can enhance the learning process by minimizing regret.
4. 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).
5. This helps us reduce the complexity of the model and speed up the learning without affecting final results.
6. Deterministic options were used for the alternate set of options part instead of epsilon greedy
7. All the code files used are shared on github, a zipped folder is submitted on moodle and a pdf report is submiited on gradescope.
8. Github link: `https://github.com/Sujal-Burad/DA6400_PA2`

1 SMDP Q-learning

1.1 Reward Curve

Following is the plot of Reward averaged over 100 runs with episodes for given Taxi Environment:

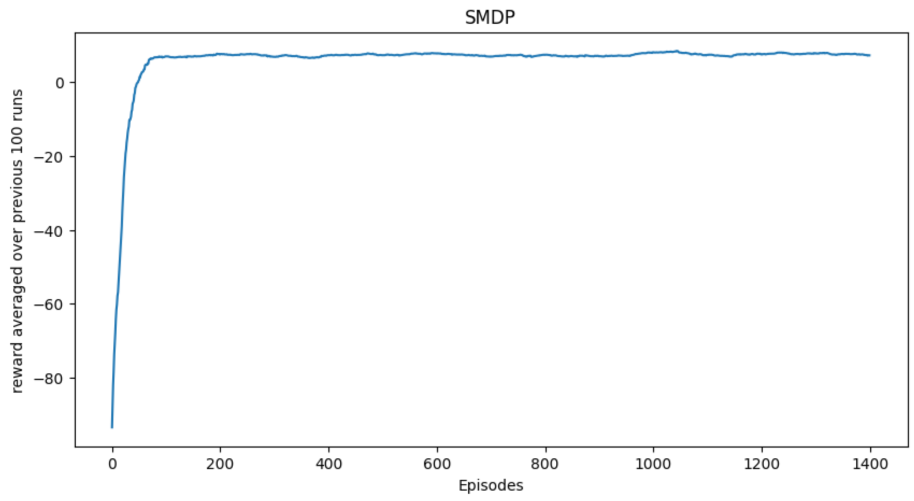

Figure 1: Reward Curve for SMDP Q-learning

1.2 Visualizing learned Q-values

Following figure shows learned Q-values for every action for each option:

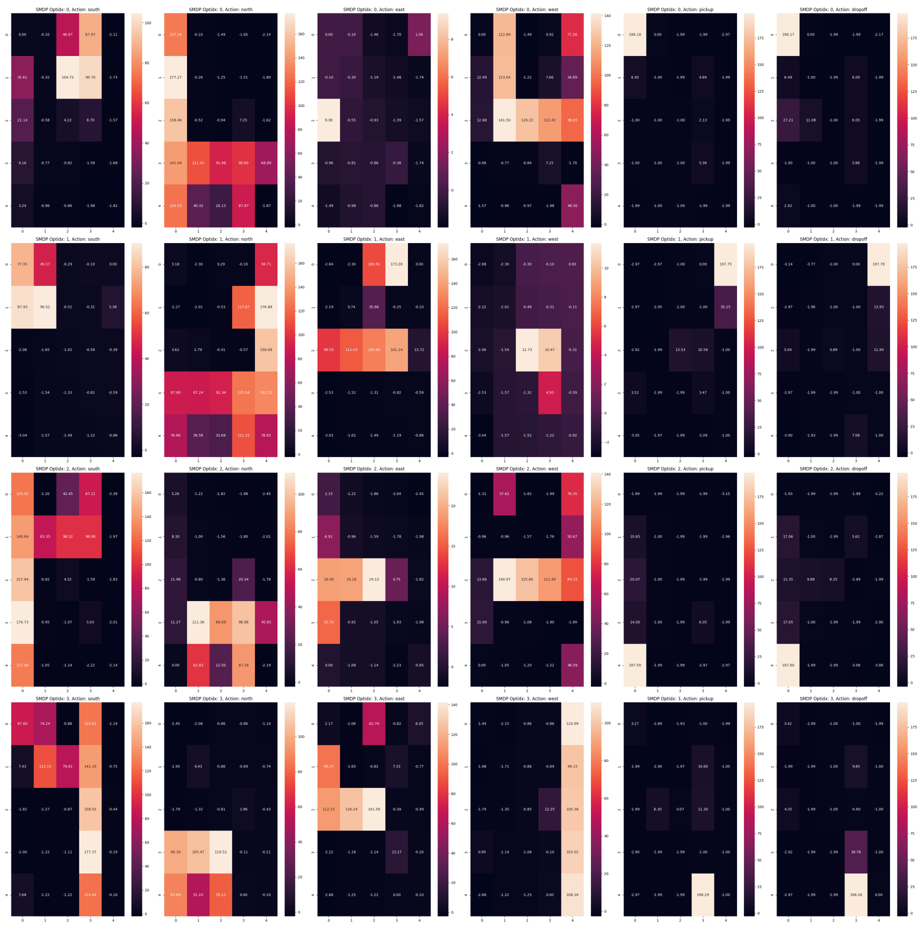

Figure 2: Learned Q-values by SMDP

1.3 Policies Learnt

Following is the heatmap illustrating policy to take for a given drop-off point which are the defined options:

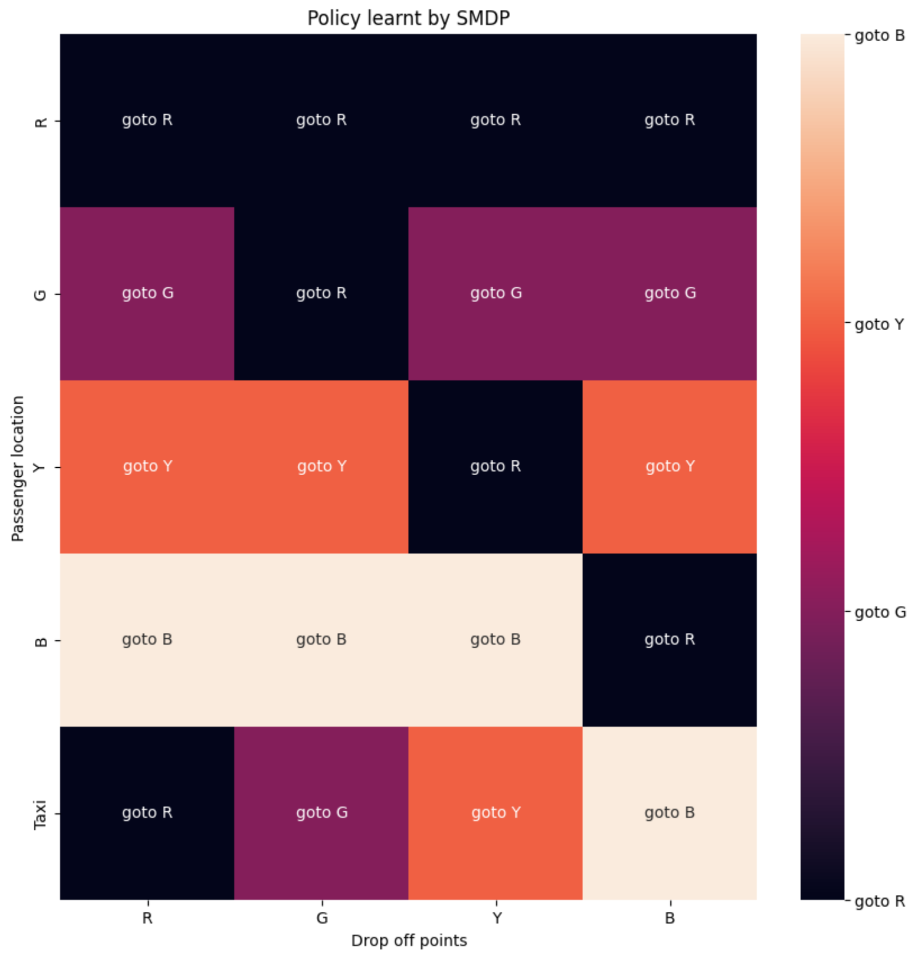

Figure 3: Policy Learnt by SMDP

And, following plots the policy of primitive action to be taken for each state, for a given option. Here,

$$0 \Rightarrow \text{Location Red}$$

$$1 \Rightarrow \text{Location Green}$$

$$2 \Rightarrow \text{Location Yellow}$$

$$3 \Rightarrow \text{Location Blue}$$

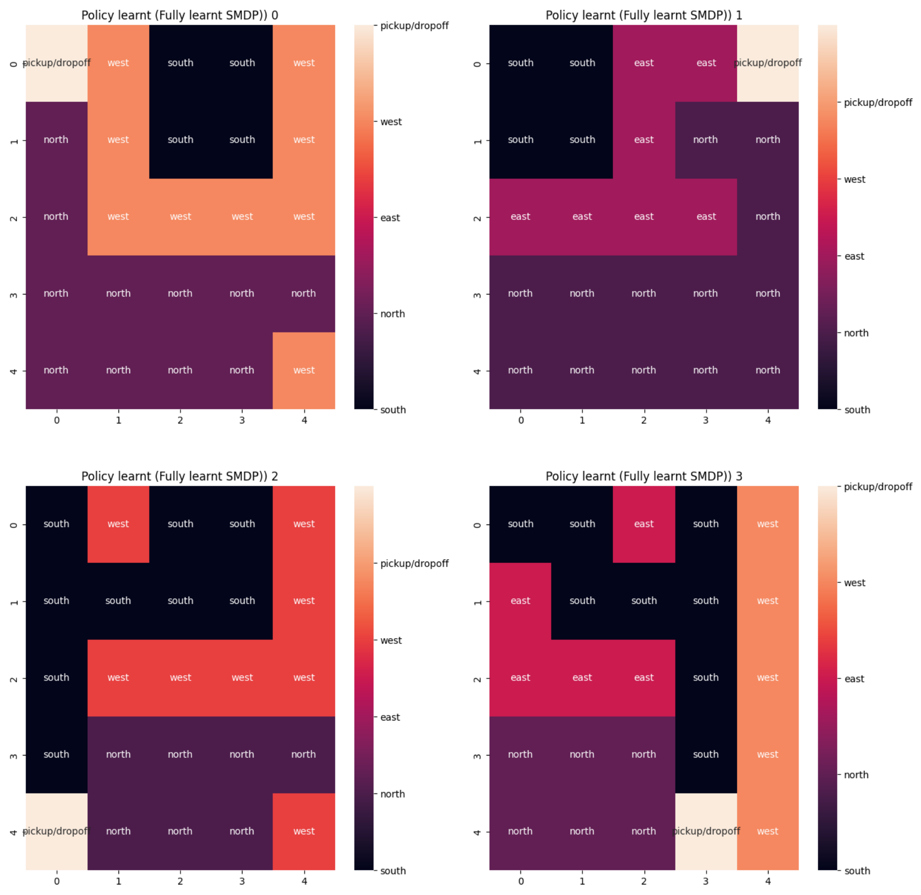

Figure 4: Fully learnt policy for each option by SMDP

SMDP Q-learning learns an effective policy in structured environments like Taxi-v3 due to the following reasons:

1. **Learning over Options (Temporally Extended Actions):**
   Instead of updating Q-values for primitive actions, SMDP Q-learning learns Q-values for *options*, which are sequences of actions leading to meaningful subgoals (e.g., navigate to a landmark, pick up the passenger). This abstraction enables efficient learning and planning in structured environments.

2. **SMDP Bellman Update:**
   The Q-value is updated based on the cumulative reward and duration k of executing the option:

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

where:

- s is the current state
- o is the selected option
- r is the total reward accumulated during the execution of option o
- k is the number of steps taken by the option
- s' is the resulting state after option termination

This allows the agent to value long-term outcomes of entire options.

3. **Effective Exploration:**
   The use of ε-greedy exploration ensures that the agent samples a wide variety of options early on, gradually favoring the best-known options. This promotes sufficient coverage of the state-option space.

4. **Theoretical Convergence Guarantee:**
   SMDP Q-learning is proven to converge to the optimal Q-values under standard conditions:

   - Finite state and option space
   - Decaying or sufficiently small constant learning rate α
   - Sufficient exploration (e.g., ε-greedy)

2 Intra-option Q-Learning

2.1 Reward Curve

Reward averaged over 100 runs v/s episodes is as follows:

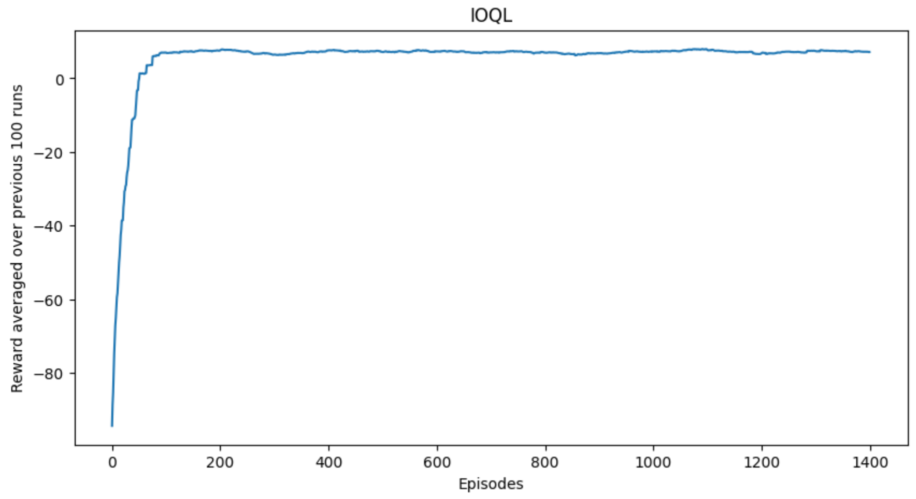

Figure 5: Reward Curve for Intra-option Q-Learning

2.2 Visualizing learned Q-values

Learned Q-values for every action for each option is:

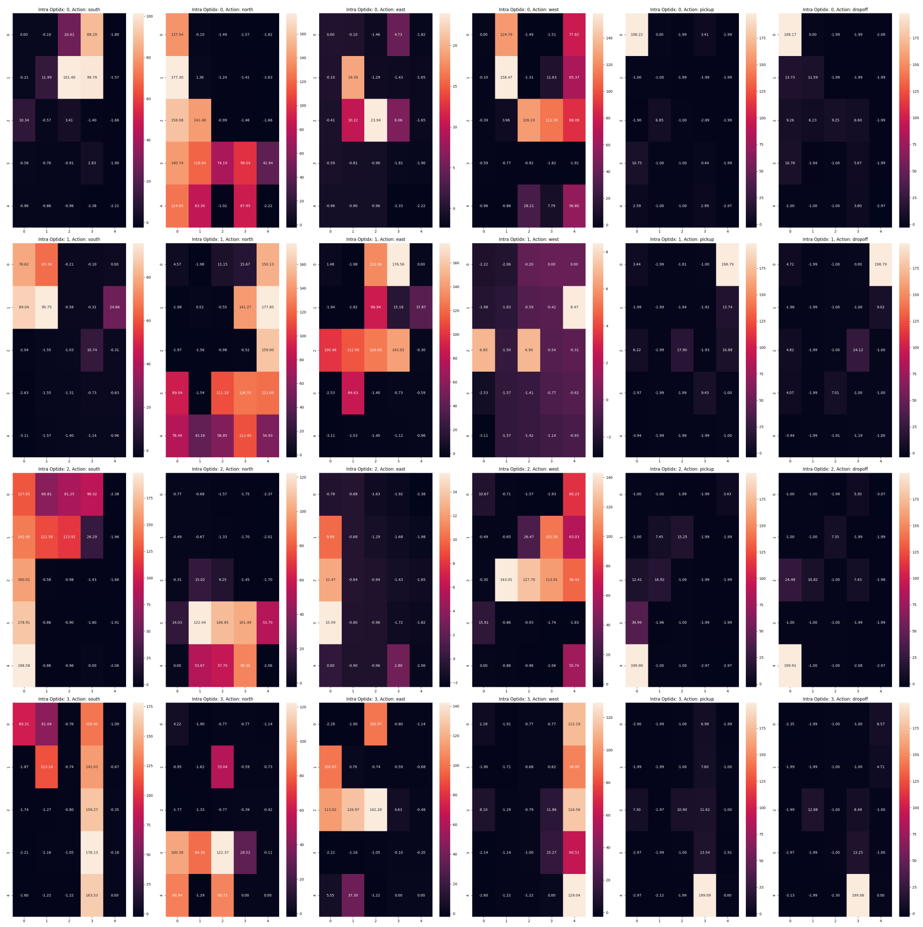

Figure 6: Learned Q-values by Intra-option Q-Learning

2.3 Policies Learnt

Policy for options defined as 4 destination locations:

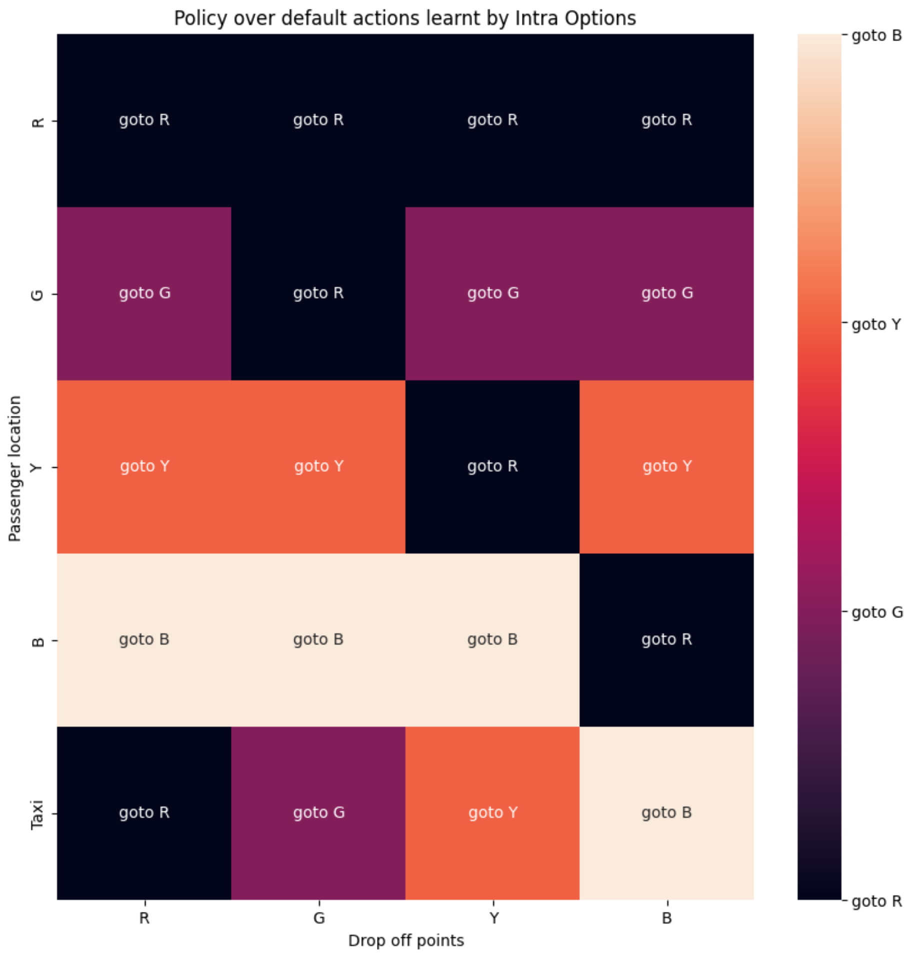

Figure 7: Policy Learnt by Intra-option Q-Learning

And, policy of primitive actions to be followed for each state is:

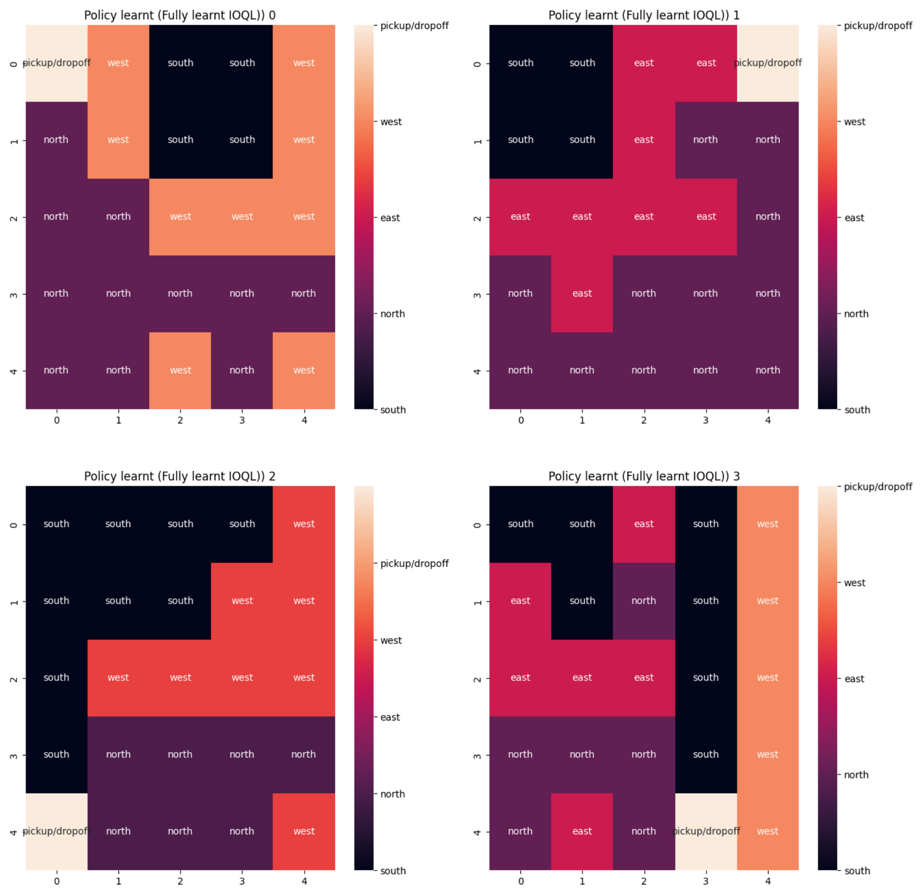

Figure 8: Fully learnt policy for each option by IOQL

Here, as before, 0,1,2 and 3 represents destination locations of R, G, Y and B respectively.

Reasoning behind why the IOQL algorithm learns the policy:

1. **Learning Over Options with Shared Initiation States:**
   Intra-option Q-learning maintains Q-values for each state-option pair $Q(s, o)$. When multiple options can be initiated from the same state, learning updates can be shared, allowing more efficient use of experience.

2. **Efficient Updates Using Option Policies:**
   When the agent takes action a in state s and transitions to s', all options o for which $\pi_o(s) = a$ (i.e., the option prescribes action a in state s) can be updated using the intra-option Bellman update:

$$Q(s, o) \leftarrow Q(s, o) + \alpha \left[r + \gamma Q(s', o) - Q(s, o)\right]$$

This enables simultaneous learning across multiple options, improving sample efficiency and accelerating convergence.

3. **Convergence to the Optimal Policy Over Options:**
Intra-option Q-learning is an off-policy algorithm with theoretical convergence guarantees, provided:

   - The MDP is finite,
   - All state-option pairs are visited infinitely often, and
   - The learning rate α decays appropriately.

   Under these conditions, the algorithm converges to the optimal Q-values, enabling derivation of an optimal meta-policy over options.

4. **Exploration via Option Policies:**
Exploration is supported both at the level of option selection (e.g., using ε-greedy selection of options) and within-option execution via stochastic option policies. This dual-layered exploration leads to robust and diverse policy learning.

3 Comparision between SMDP and IOQL

Following plot compares both algorithms:

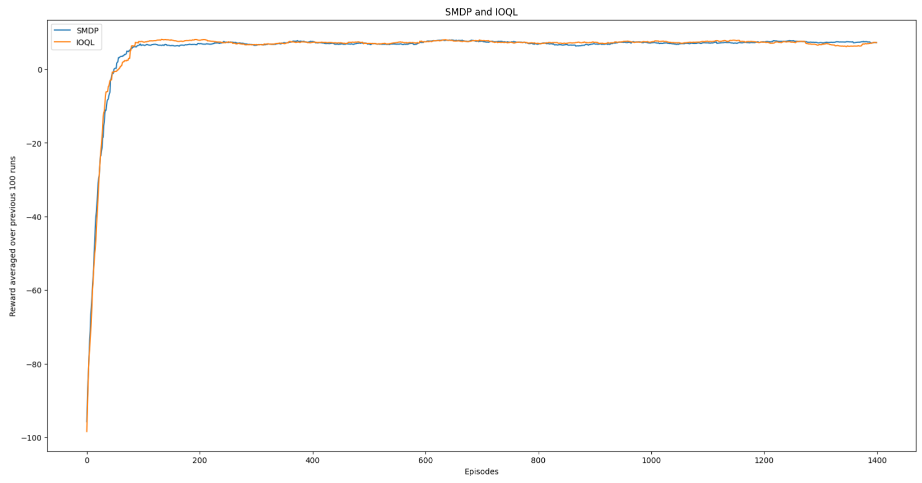

Figure 9: SMDP v/s IOQL

In above, plot both curves closely overlaps, so following are the zoomed versions of it:

1. Zooming saturation Region:

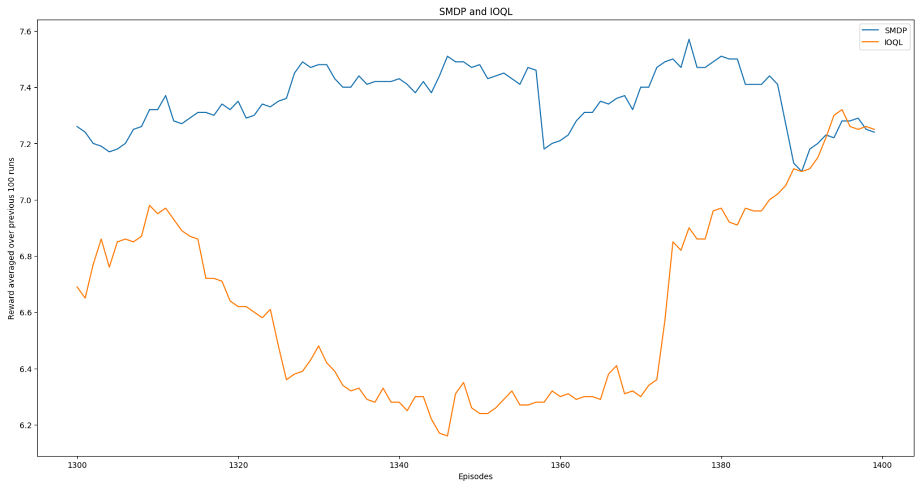

Figure 10: SMDP v/s IOQL around saturation region

2. Zooming initially:

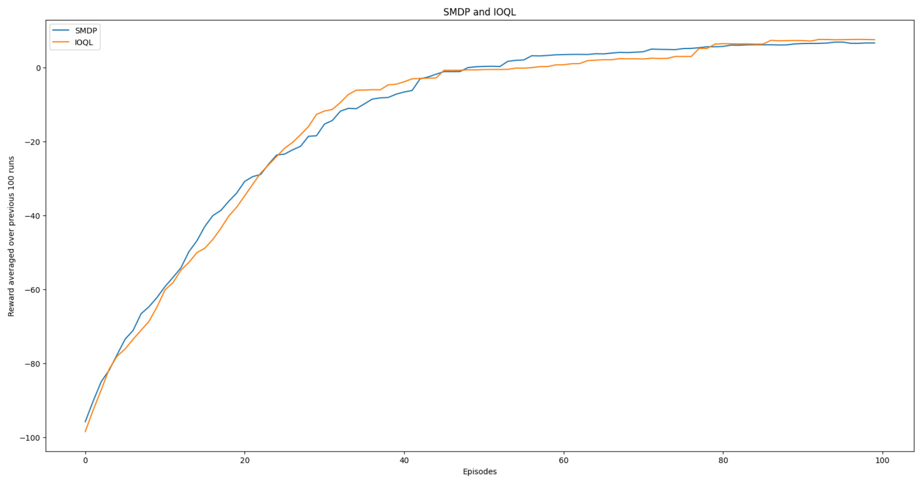

Figure 11: SMDP v/s IOQL initially

Conclusion:
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.

4 Alternate Set of Options

Till now we defined options using fixed landmarks (R, G, Y, B), regardless of the passenger or destination. But now, 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

4.1 SMDP

4.1.1 Reward Curve

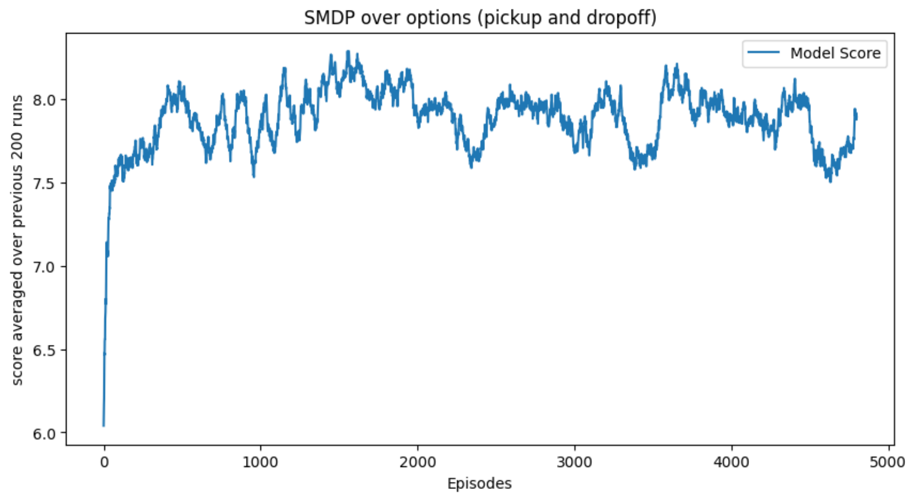

Figure 12: Reward Curve

4.1.2 Visualizing learned Q-values

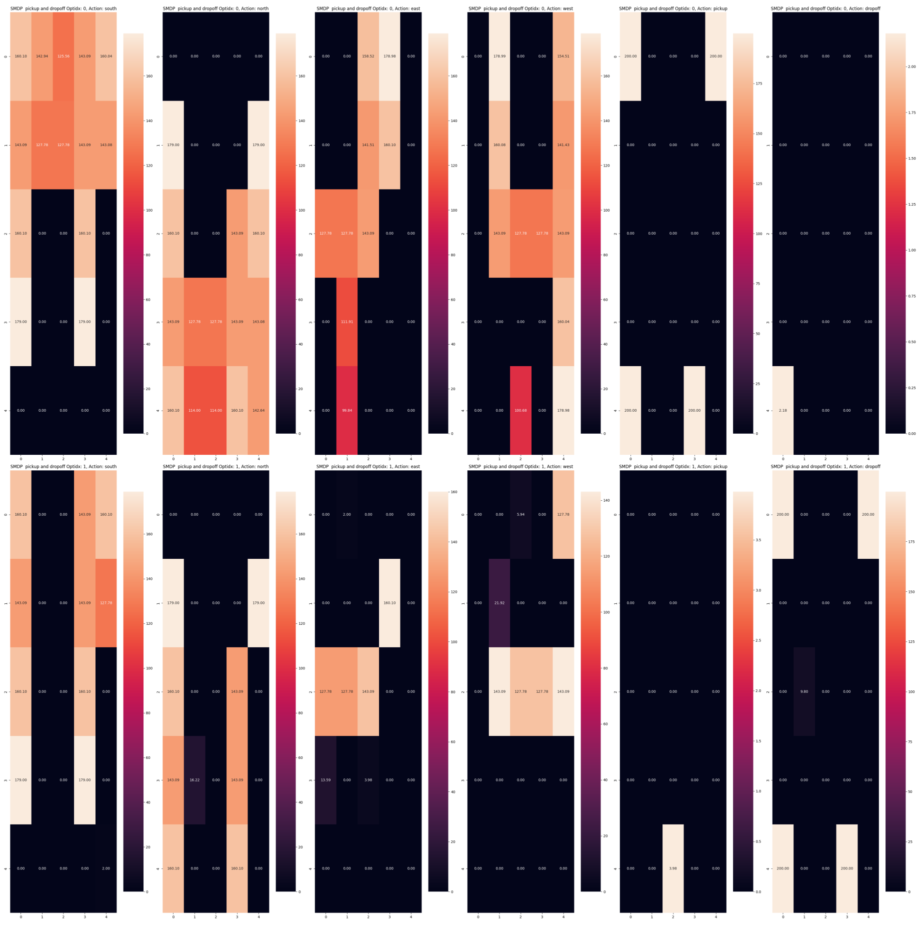
Figure 13: Learned Q-values

4.1.3 Policies Learnt

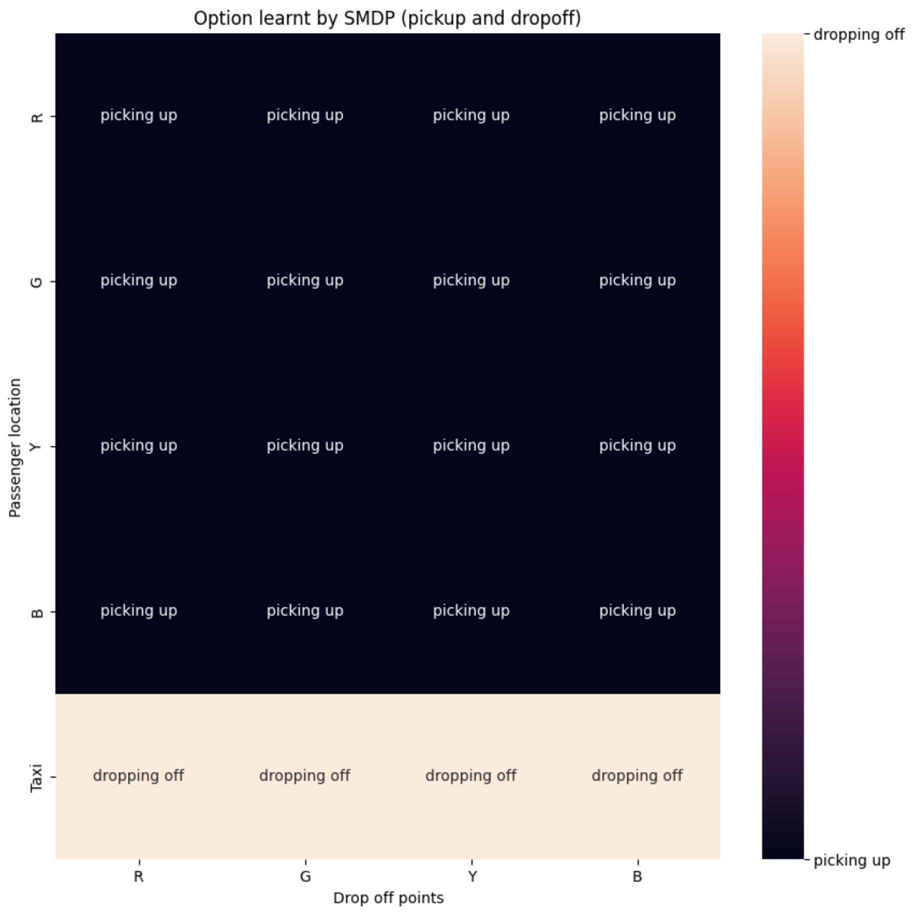

Figure 14: Policy Learnt

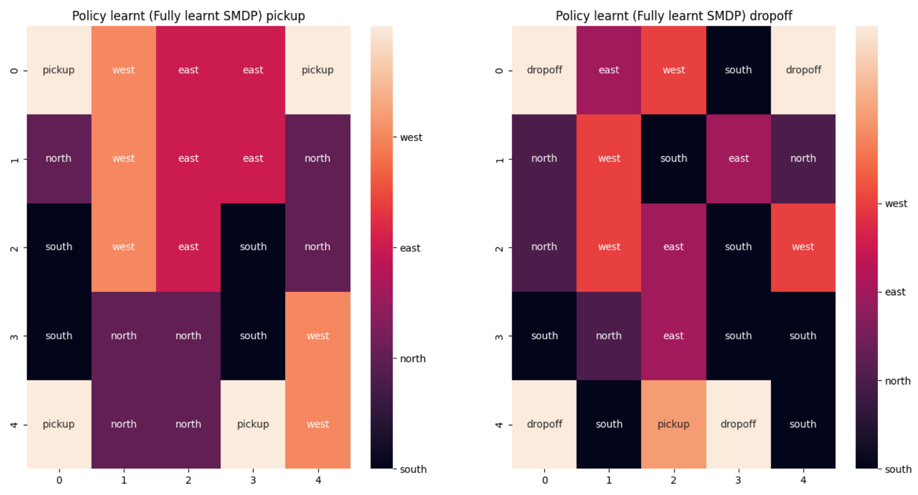

Figure 15: Fully learnt policy for each option

4.2 IOQL

4.2.1 Reward Curve

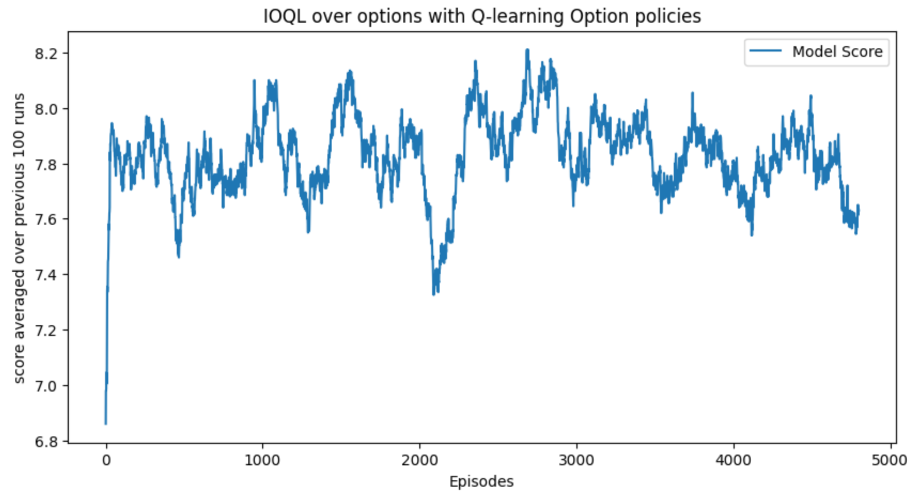

Figure 16: Reward Curve

4.2.2 Visualizing learned Q-values

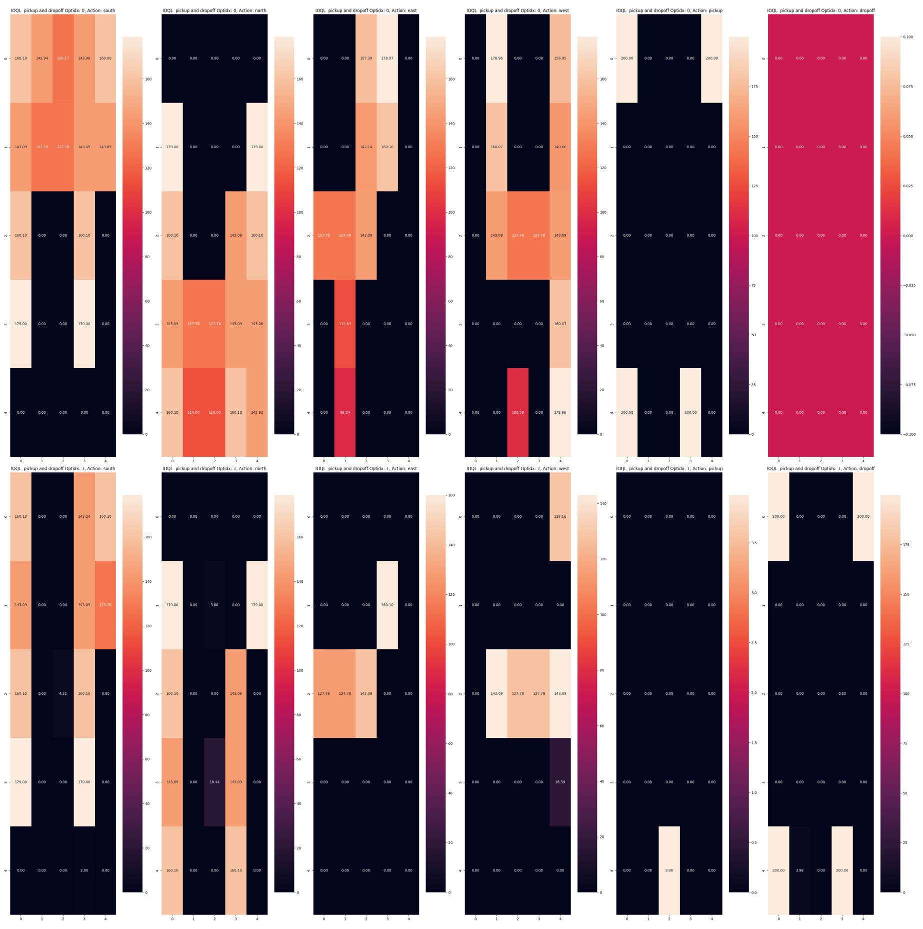

Figure 17: Learned Q-values

4.2.3 Policies Learnt

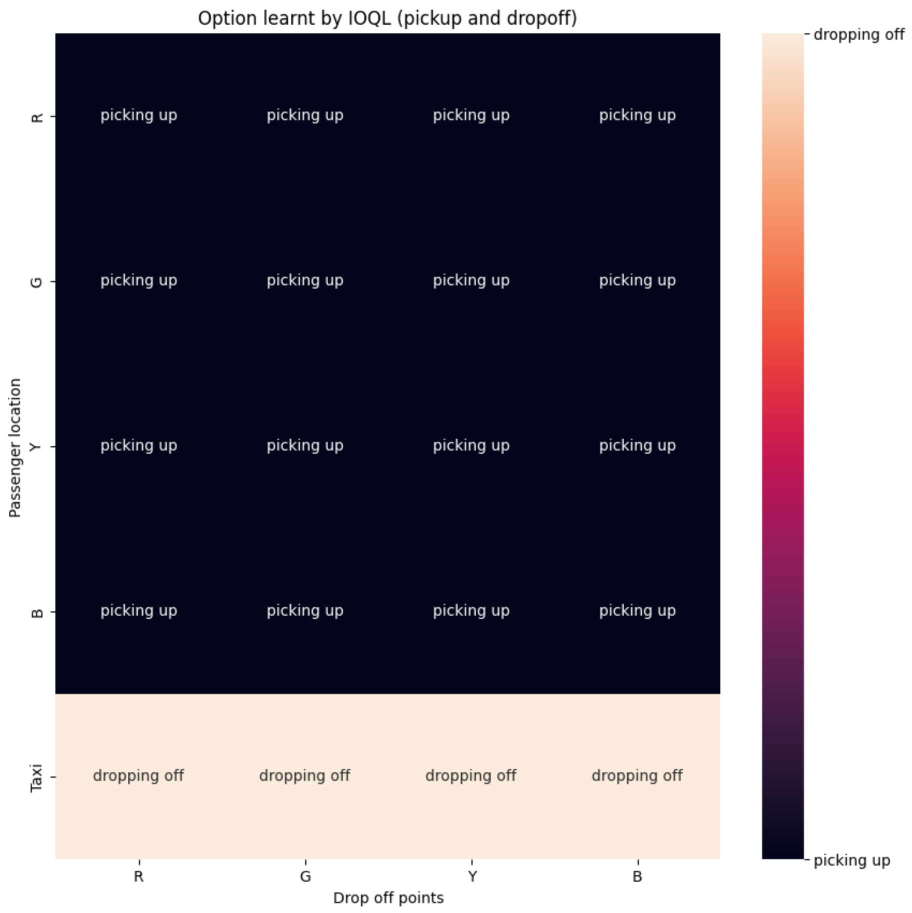

Figure 18: Policy Learnt

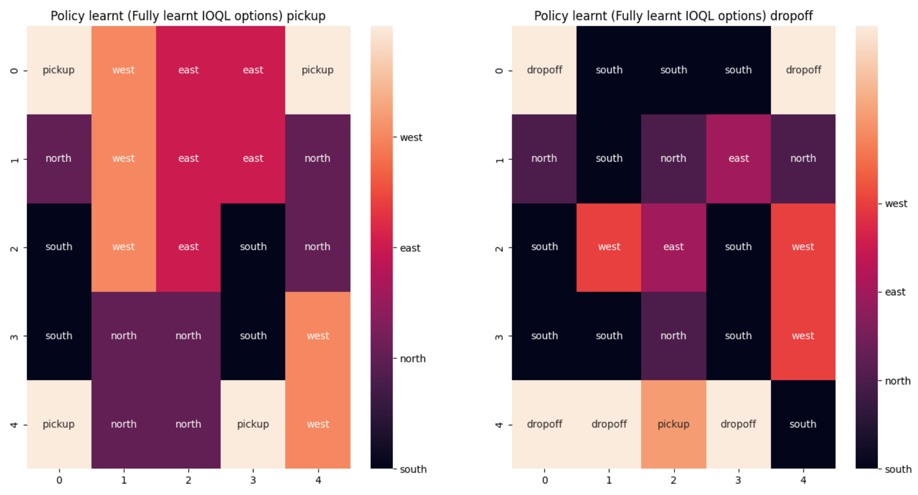

Figure 19: Fully learnt policy

4.3 SMDP v/s IOQL

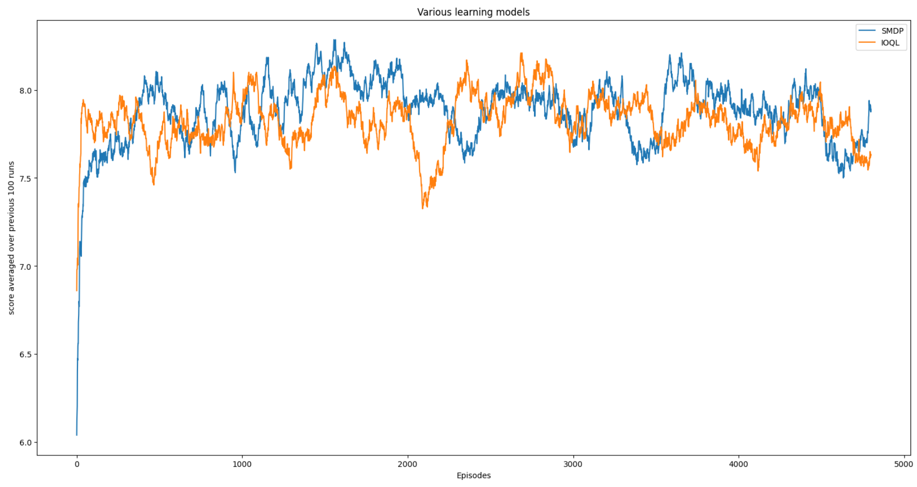

Figure 20: SMDP v/s IOQL

5 Comparing performance on both the options

The R,G,B,Y set options is landmark based option and pick-up drop-of based options is task-based. From the plots provided we can see that task-based options start with positive reward from the beginning unlike landmark based options and both having similar highest rewards. Thus task-based options is better because Task-based options leverage the structure of the Taxi problem, breaking it into meaningful subtasks. This improves both learning efficiency and performance, especially when combined with hierarchical methods like SMDP or intra-option learning.