Q1:

Based on these results, and your understanding of the three algorithms used to produce them, please answer the following discussion questions. <br />

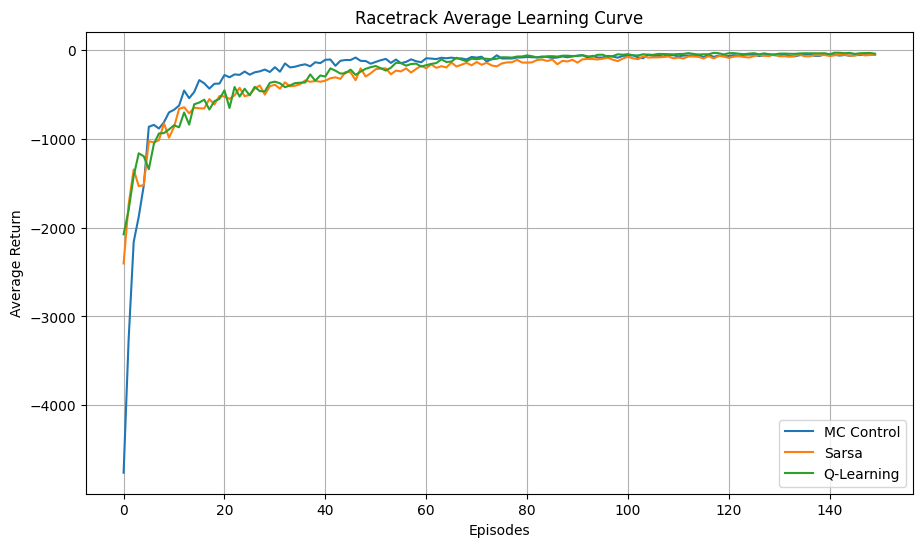
**Question 1:** Briefly compare the performance of each of the three agents.

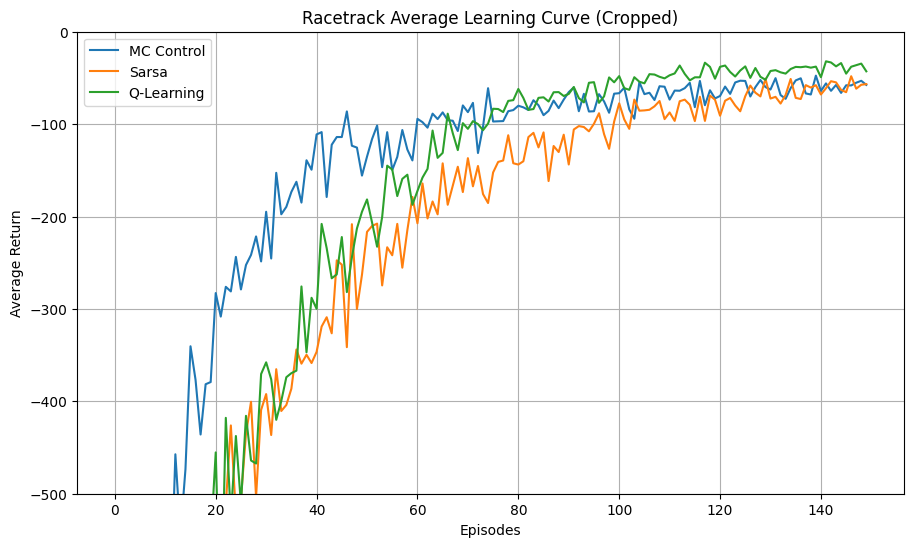
**Question 2:** Why do you think that your Monte Carlo and Temporal-Difference agents behaved differently?

**Question 3:** Does the performance of your Sarsa and Q-Learning agents meet your expectations? Why do you think that this is the case?

**Question 4:** What could be done to improve the performance of these agents?

Please do not exceed **60 words** for any of your answers.



****

1. Initially, Sarsa and Q-learning outperform Monte-Carlo (MC), which requires full episodes to learn, but MC catched up quickly. MC starts converging to an average return at 45 episodes compared to 80 and 100 for Q-learning and Sarsa respectively, with Q-learning ultimately reaching a higher average return of -45, compared to -55 for both MC and Sarsa.
2. Temporal-Difference (TD) agents update their policy after each step using current and previous estimates, which enables quicker learning adjustments compared to Monte Carlo (MC) methods, which only update after complete episodes. This difference in update frequency, with TD's bootstrapping from existing estimates, results in divergent learning behaviours, particularly noticeable in initial episodes.
3. The performance aligns with expectations as Q-Learning seeks the optimal policy through exploration, learning a behaviour policy for experience and an optimal policy for performance. Sarsa, being on-policy, learns a near-optimal policy directly through its actions. Q-Learning's dual-policy approach often achieves optimal results, while Sarsa's method ensures safer, albeit slightly less optimal, learning progression.
4. Implementing either an epsilon-greedy or epsilon -soft policy can improve agent performance by fine-tuning the balance between exploration and exploitation. Gradual reduction of epsilon allows a shift towards exploitation as the agent learns. This, coupled with a carefully adjusted learning rate, can significantly enhance the learning efficiency and policy quality of the agents.

Based on your results, and your understanding of the algorithm and modifications that you have implemented, please answer the following discussion questions. <br />

**\*\*Question 1:\*\*** What modifications did you make to your agent?

**\*\*Question 2:\*\*** What effect(s) did you expect your modifications to have on the performance of your agent?

**\*\*Question 3:\*\*** Did your modifications have the effect(s) you expected? Why do you think that this was the case?

**\*\*Question 4:\*\*** If you had more time, what would you do to further improve the performance of your agent?

Please do not exceed **\*\*60 words\*\*** for any of your answers.

Please note that **\*\*your implementation and discussion will be assessed jointly\*\***. This means that, in order to score highly, you will need to correctly implement appropriate modifications to your agent **\*\*AND\*\*** discuss them well.

(Q1)

1. The Dyna-Q+ learns directly and indirectly. The direct learning policy was changed to a epsilon-greedy policy incorporating epsilon-decay. A model was built for indirect learning using the state-action variable previously visited, allowing our agent to learn indirectly from the model, the state-action variable updates in the model with a bonus variable that increases every timestep it was not visited.
2. Due to the addition of the epsilon-decay in direct learning, and the exploration bonus in the indirect learning. I expected the agent to converge faster; due to the indirect learning, and at a higher reward than the basic Q-learning algorithm. This is due to the agent having a more balanced exploration to exploitation rate.
3. Yes, it did. The Dyna-Q+ agent converged in 20 episodes, faster than the basic agent's 120, thanks to indirect learning from 50 state-action pairs and an exploration bonus. This alongside the epsilon decay, made the agent explore more effectively, reaching a higher convergence value of around -5 versus -45.
4. I would firstly sweep the hyperparameters to find the optimal values for them. I would also run the model for indirect learning for a higher number of iterations which could prove a faster convergence rate and a higher convergence value.