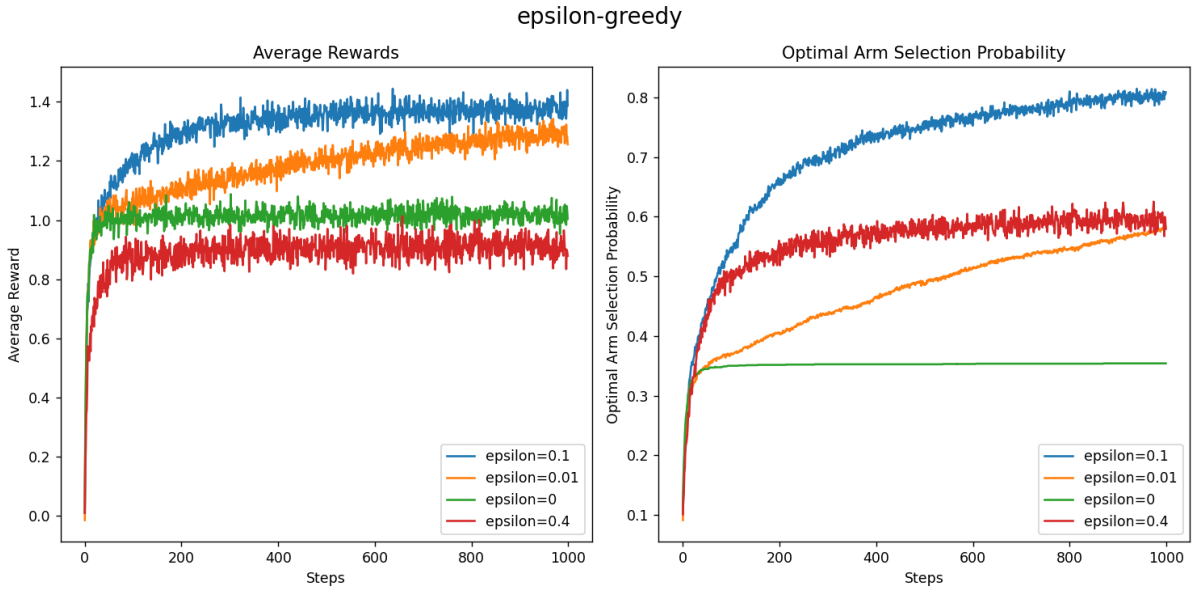
Ayush Mishra 2020A7PS1715G

Question 1

Multi-armed Bandit

1) Epsilon Greedy Action Selection



Different epsilon values impact the algorithm's behaviour:

1) **A) Epsilon = 0 (Greedy Strategy):** When epsilon is set to 0, the algorithm becomes purely greedy. It always chooses the action with the highest estimated Q-value and never explores other actions.
   B) Average rewards Initially are higher but after some time steps it might converges to some non-optimal solution and gives less Average rewards of around 1.
   C) The purely greedy strategy tends to get stuck with a suboptimal arm, resulting in a low percentage of **optimal arm selections** of around 35%.

2) **A) Epsilon = 0.01 (Low Epsilon):** When epsilon is set to 0.01, The algorithm does 1% exploration and 99% exploitations. It converges to a final value very slowly.
   B) Since the algorithm does not aggressively explore it gives less rewards in the beginning but overtime there is a steady increase in the **average Rewards** value as the algorithm learns more.
   C) Overtime the **optimal Arm selection** Probability increases slowly. In long enough time it will converge to 99.1% optimal arm selection.

3) **A) Epsilon = 0.1 (Moderate Epsilon):** A moderate value of 0.1 balances exploration and exploitation more effectively. As depicted in the graph above. the strategy

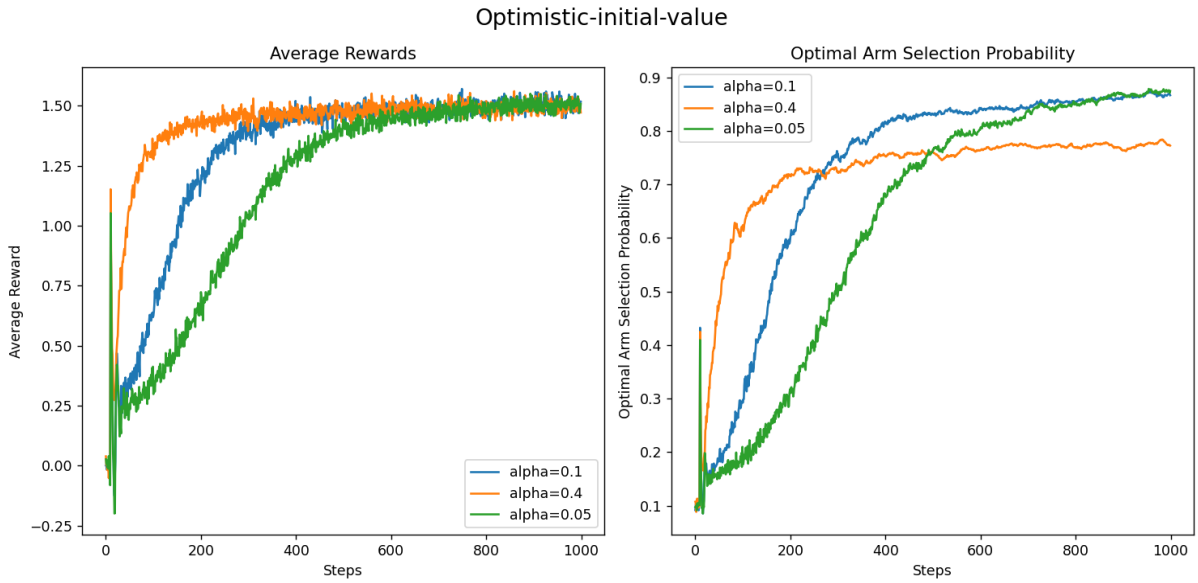
explores new arms with reasonable frequency of around 10% and 90% exploitation.
B) This allows it to discover the optimal arm while maintaining a relatively high **average reward** throughout the 1000th time steps. For this test-bed best average reward was shown by moderate Epsilon.
C) Initially It explored enough and allowed it to **select optimal arm** with high percentage about 80% in just 1000 steps. After infinity time step it will Converge to a value of 90.5% optimal arm selection.

4) **A) Epsilon = 0.4 (Large Epsilon):** When epsilon is set to 0.4, the strategy explores extensively, often choosing arms with lower estimated rewards. Exploration is around 40% and 60% exploitations.
B) This leads to a more fluctuating average reward as the strategy frequently explores suboptimal arms this results in lower **average reward** because nearly 4 actions it takes out of 10 are for exploring in the process it is not able to make the average high.
C) This value of epsilon quickly converges because it explores very aggressively from the starting. Percentage of **optimal arm selection** to settles around 60% in 1000th time steps which is very near to the theoretically calculated value.

In Conclusion, the choice of Epsilon in the Epsilon-greedy strategy significantly influences its performance in the 10-arm bandit problem. **A moderate epsilon value, such as 0.1, appears to strike a balance between exploration and exploitation, leading to higher average rewards and a quick increase in the percentage of optimal arm selections**.
A purely greedy strategy (Epsilon = 0) quickly plateaus in terms of average reward and also the optimal arm selection percentage.
Smaller epsilon value (Epsilon = 0.01) is not able to explore enough to give its best performance in only 1000time steps.
However, a larger Epsilon value (Epsilon = 0.4) results in more exploration, causing greater fluctuations in the average reward and Percentage of optimal arm selection.
These findings highlight the importance of selecting an appropriate Epsilon value based on the problem's characteristics and the desired exploration-exploitation trade-off.
In this problem best performance was possible for moderate epsilon but in some other problem best performance might be given by setting it too small or large or even purely greedy.

Gradually reducing the value of epsilon over time steps is a beneficial strategy. Initially, it encourages more exploration to discover options, but as time progresses, it shifts towards exploiting known solutions effectively.

2) Optimistic Initial Value



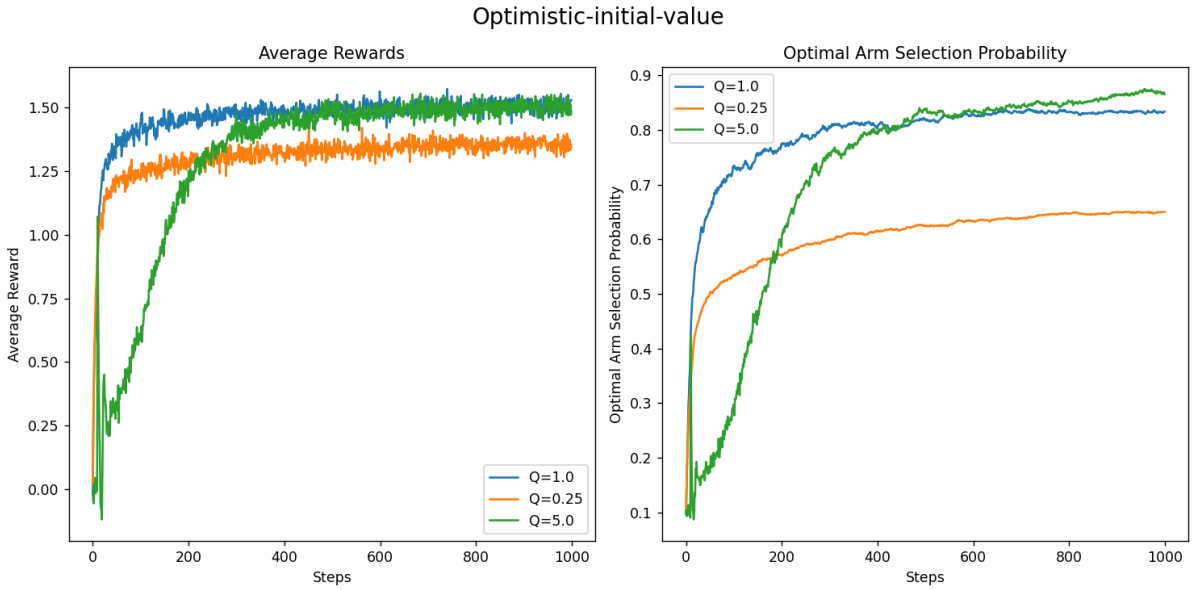
Different alpha values impact the algorithm's behaviour:

1) **A) Alpha = 0.05(Small alpha):** Leads to slow convergence, often requiring more time steps for learning. For a small alpha the initial value of Q decreases slowly as compared to alpha allowing it to explore more.
   B) The **average reward** for a small value of alpha is low in the initial time steps because it is still exploring and have not converged to a good solution but as time steps increases average rewards increases.
   C) Initially the **Optimal arm selection** probability is low but with time is increases as it explores enough to converge on a optimal arm giving the best % for optimal arm selection around 90%.

2) **A) Alpha = 0.1 (Moderate Alpha):** Moderate alpha offers a balanced convergence rate over time steps Moderate alpha strikes a balance between stability and adaptation speed.
   B) Average Reward in moderate Alpha is higher as compared to small value of alpha as it converges faster to it final value.
   C) Optimal arm selection is higher than smaller value of alpha because of the same reason but at the end smaller value of alpha overtakes moderate alpha. It is around 90%.

3) **A) Alpha =0.4 (Large Alpha):** Large alpha results in a rapid initial increase but may exhibit more fluctuations.
   B) Average rewards are high initially but over time alpha with lower value overtakes it because using large alpha convergence is to a non-optimal solution
   C) Due to quick convergence to a non-optimal solution initially high percentage of **optimal arm selection** action is seen but in the end it flattened around 75%.

In conclusion we can say overall **moderate value performed better** than small value of alpha and that performed better than large value of alpha for this problem but the performance may vary from problem to problem. Selection of alpha should be done according to the particular problem.



Different Initial Q value values impact the algorithm's behaviour:

1) **A) Q = 0.25(Small Initial Value):** Leads to fast convergence, often Converging on a non-optimal action. For a small Initial Q the value of Q quickly comes in range of actual values giving it less time to explore hence non-optimal convergence.
   B) The **average reward** for a small value of Initial value Quickly stabilizes on a very low average of around 1.25 and increases very slightly.
   C) Optimal arm selection probability quickly shoots up to 60% but after that it doesn't change because it is no longer exploring.

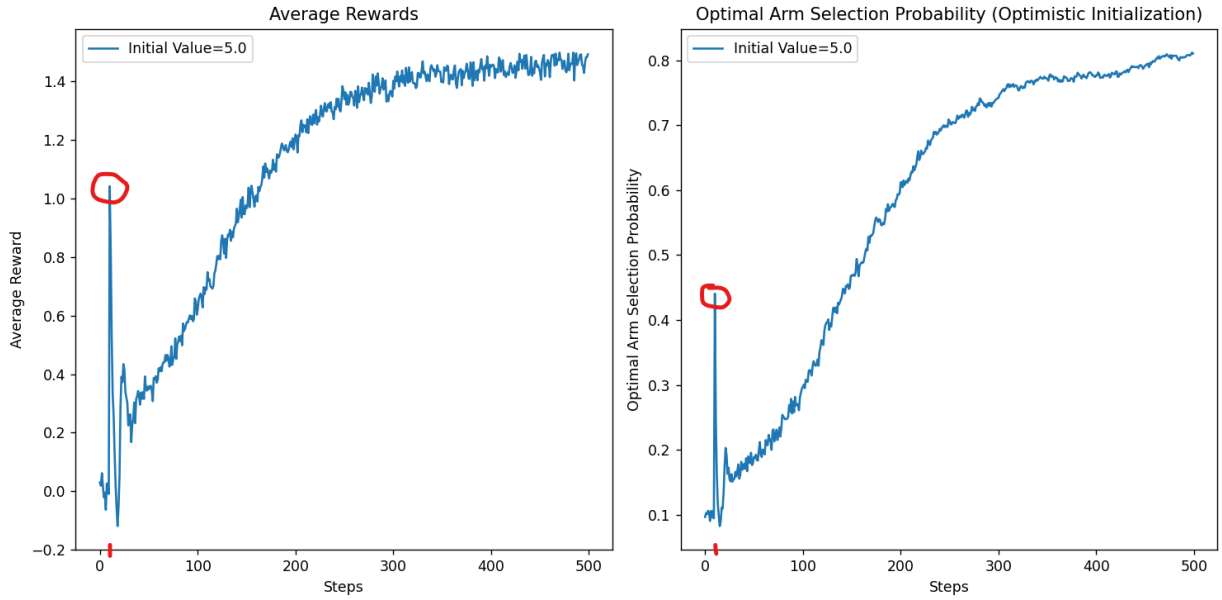
2) **A) Q = 1.0 (Moderate Initial Value):** Moderate Initial value offers a balanced convergence rate over time steps. Moderate Initial value strikes a balance between stability and adaptation speed.
   B) Average Reward in moderate Initial value is higher as compared to small Initial value as it is allowed to explore more than smaller initial value and reaches a better optimal action.
   C) Optimal arm selection is higher than smaller Initial value because it is allowed to explore more than smaller initial value and reaches a better optimal action. It is around 80%.

3) **A) Q = 5.0 (Large Initial Value):** Large Initial value results in a slower Convergence to the optimal solution and it also experiences more spikes in the beginning of the run will be explained later.

B) Average rewards are Low initially but over time because of the high exploration but later on it catches with the moderate initial value because using large initial value the final convergence is at a better solution.
C) Due to slow convergence to a optimal solution initially low percentage of **optimal arm selection** is seen but in the end it reaches to around 85 percent.

In conclusion, Lower Initial value of Q doesn't support exploration. Large Initial value leads to higher exploration. **Here moderate as well as Large initial value performed similar on the average reward.** But Selection of initial value should be on basis of problem-to-problem basis. **If you have very long time then try to use Large Initial value else use a moderate value**.

Spikes



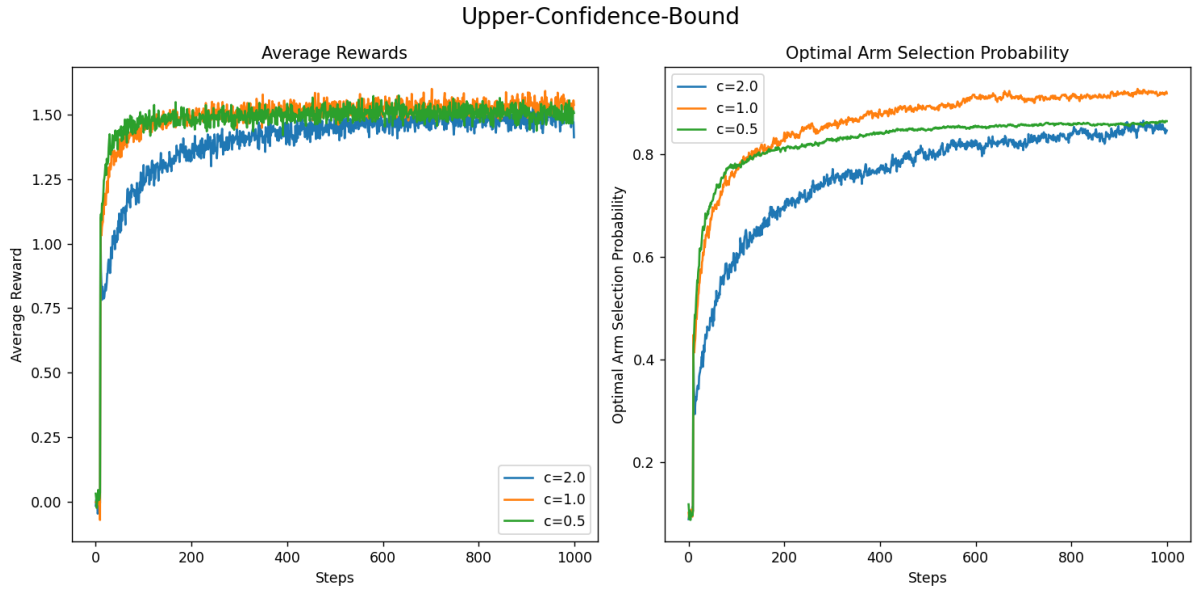
Spikes in the initial part of Optimistic Initialization:

The oscillations and spikes in the early part of the curve for the optimistic method, despite being averages over 2000 tasks, can be attributed to the initial optimistic estimates. In this method, each action's initial estimated value is set optimistically high, encouraging exploration.

However, during the initial steps, the algorithm choses the maximum value of Q for exploration and decreases the value for each action selected. Once each action is selected once and value of Q is decreased for each the very next action to be select has a high probability to be the optimal action because the Q value will be decreased least for this action.

As more steps are taken, the algorithm converges to better estimates, and these initial fluctuations tend to smooth out, resulting in a more reliable long-term performance.

3) Upper-Confidence-Bound Action selection



Different C value values impact the algorithm's behaviour:

1) **A) C=0.5 (Small C):** When C is small, the UCB algorithm places less emphasis on exploration and focuses more on exploiting the actions with the highest estimated values.
   B) The **average reward** may increase rapidly in the short term because the algorithm quickly identifies and exploits what it believes to be the best actions. However, in the long run small C can lead to suboptimal performance because it doesn't explore enough to discover better actions.
   C) Small C Prioritizes exploiting perceived best arms, leading to initially high **optimal arm selection**. However, long-term performance can suffer due to limited exploration and potentially inaccurate estimates.

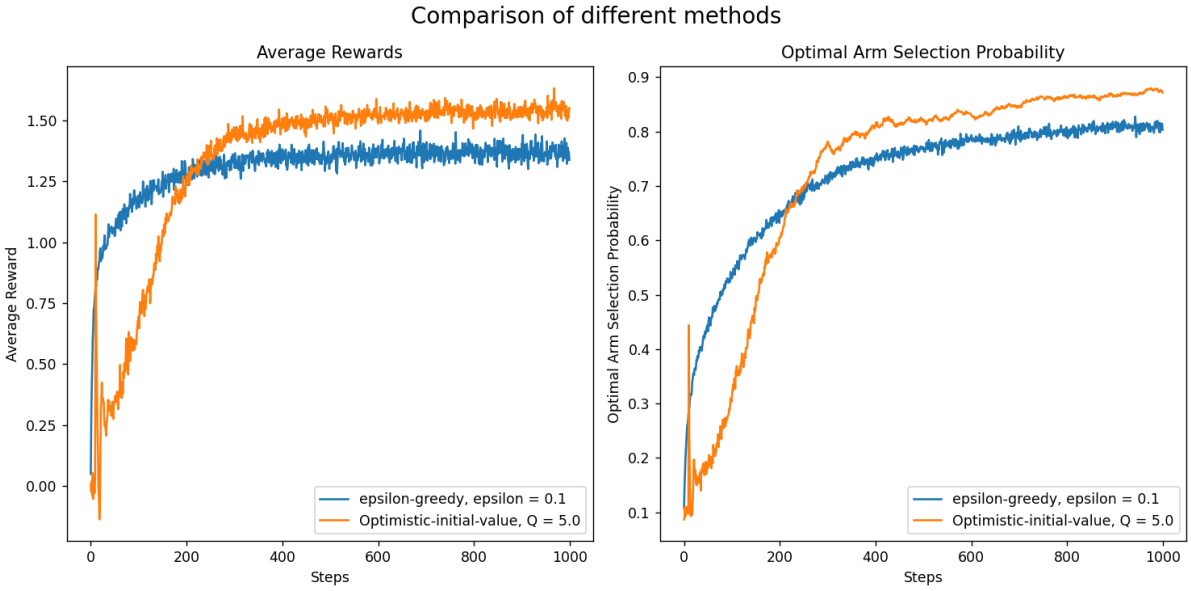
2) **A) C=1.0 (Moderate C):** The algorithm explores a bit more than with a small C which can lead to discovering better actions while still exploiting the ones with high estimated values.
   B) The **average reward** may show a reasonably steady increase over 1000 time steps as the algorithm effectively balances exploration and exploitation.
   C) Small C Prioritizes exploiting perceived best arms, leading to initially high **optimal arm selection.** However, long-term performance can suffer due to limited exploration and potentially inaccurate estimates.

3) **A) C=2.0 (Large C):** Large value of C prioritizes exploration. the UCB algorithm places a higher priority on exploration, even for actions with lower estimated values.
   B) Initially the average is quiet lower because of the exploration but the algorithm is likely to explore extensively and might discover the optimal action, potentially leading to higher **average rewards** in the long run.

C) Large C Emphasizes exploration, leading to lower probability of the **optimal arm selection** in the short term. Over time, it may discover optimal arms and give the best results over long time interval even better than moderate values of C.

In summary, the choice of C in UCB influences the balance between exploration and exploitation, which, in turn, impacts the optimal arm selection probability over 1000-time steps. Smaller C values tend to focus on exploitation, moderate C values balance both, and larger C values emphasize exploration. For this Problem **Moderate value of C performed better than other Values.**

4) Comparison Between Epsilon greedy and Optimistic initial value



1) Optimistic initialization performance is not good at the beginning as it explores a lot early. Optimistic initial values promote more aggressive exploration as compared to epsilon greedy. quickly identifying the optimal arm, while Epsilon-Greedy explores less initially and may take longer to converge to the best action.

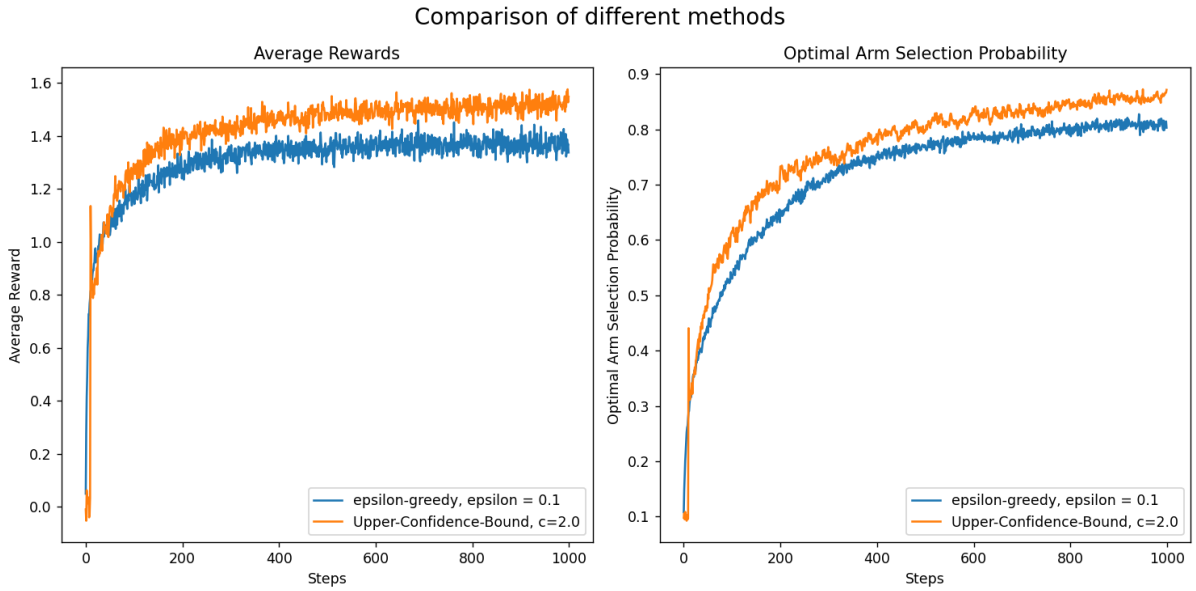
2) That is why initially the **average reward** is more for epsilon greedy. But after sufficient steps optimistic initial value converges to a solution and thus increasing the average reward. After nearly 300 steps Optimistic initial value overtakes epsilon greedy and stays above in the average reward graph after that.

3) The **Optimal arm selection** Probability for Optimistic Initial value is lower than epsilon greedy for first 300 time steps because Optimistic explores in the beginning Aggressively. After 300 time steps Optimistic Initial value over takes Epsilon Greedy

and stays above it for rest of time steps around 85% where epsilon greedy was at only 80%.

Here overall performance of Optimistic initial value was better than Epsilon greedy. How one method performs will depend on the value of the hyper-parameters and the vary for problem-to-problem for both the methods. Initially optimistic initial value focuses on exploration where-as epsilon greedy keeps the ratio of exploration to exploitation same throughout the run so this is why in initial part of the run epsilon greedy performs better than optimistic initial value but later optimistic initial value overtakes epsilon greedy.

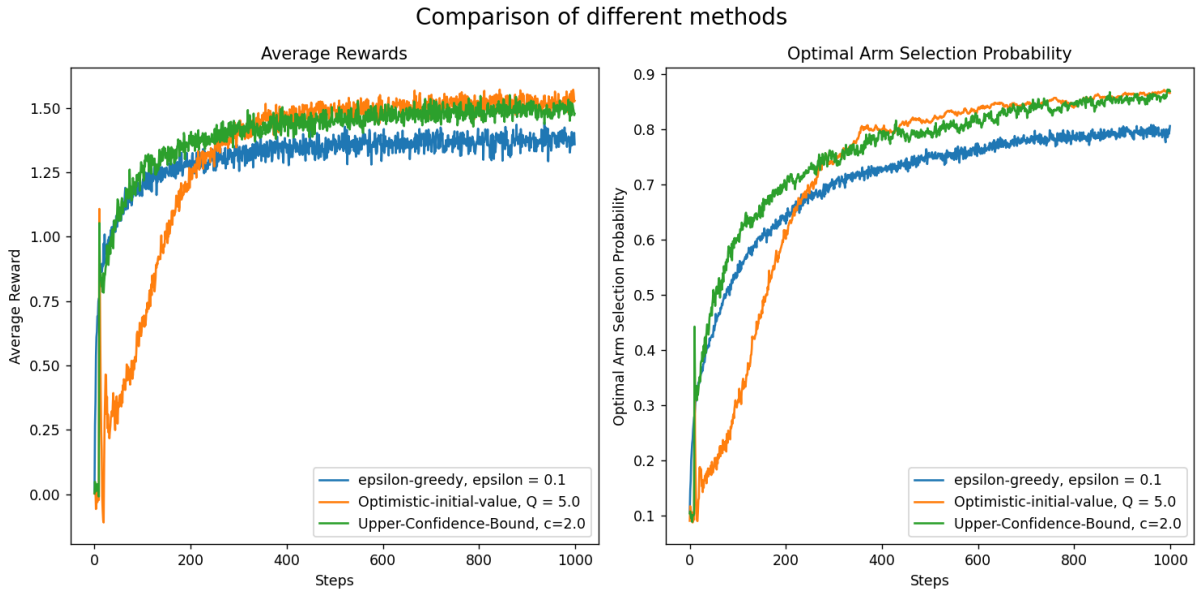
4) Comparison Between Epsilon greedy and Upper-Confidence-Bound



1) Upper Confidence Bound surpasses Epsilon-Greedy through adaptive exploration, maintaining a balance between exploration and exploitation by prioritizing actions with greater uncertainty. This strategic approach leads to efficient learning and ultimately results in superior long-term rewards, especially in dynamic environments. UCB dynamically adjusts its action selection strategy.
2) Which effectively improving both **average reward** progression Upper Confidence Bound method than Epsilon Greedy Initially both methods performance was same but at 1000 steps there is a significant difference in Average reward for both methods.
3) The probability of selecting the optimal action over time, leading to more effective exploration and exploitation compared to the fixed exploration rate of Epsilon-Greedy. At end of 1000 steps **Optimal arm selection** Probability of Upper Confidence bound action selection stayed 85% where Epsilon greedy was only at 80%.

In Conclusion, The overall performance of Upper Confidence Bound was better than Epsilon greedy. How one method performs will depend on the value of the hyper-parameters for both the methods. Epsilon-Greedy prioritizes exploration or exploitation based on a fixed parameter, while Upper Confidence Bound dynamically balances both, making UCB more effective in adaptive and uncertain environments.

5) Graph of all 3-methods epsilon greedy, Optimistic Initial value and Upper-Confidence-Bound Action selection

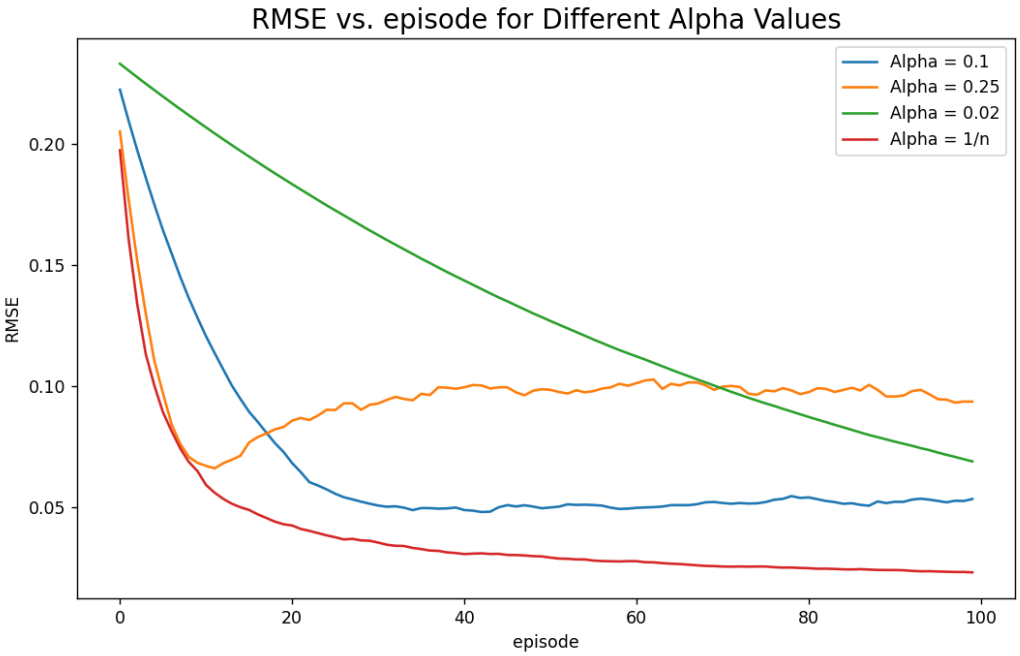


Conclusion:

In the context of the bandit problem, we have examined three fundamental strategies: Epsilon-Greedy, Optimistic Initial Value, and Upper Confidence Bound (UCB). Epsilon-Greedy offers a straightforward approach, but its performance can be sensitive to the choice of the exploration parameter (epsilon). Optimistic Initial Value encourages early exploration by starting with optimistic value estimates, often leading to swift learning.
However, the star of the comparison is UCB. UCB dynamically balances exploration and exploitation by considering uncertainty in action values. This adaptability makes it exceptionally effective, especially in situations with varying levels of uncertainty. UCB's ability to respond to the inherent unpredictability in real-world scenarios sets it apart from its counterparts.
In summary, the choice between these strategies hinges on the problem's complexity and the desired level of exploration. UCB stands out as the go-to option for environments where adaptability and robustness are essential, while Epsilon-Greedy and Optimistic Initial Value have their merits in simpler, well-understood scenarios.

Question 2

Markov reward process



Different Alpha value values impact the algorithm's behaviour:

1) **Alpha=0.25:** If the learning rate alpha is too high: This means that the updates to V(s) are large and can lead to oscillations or overshooting. In essence, it's like taking big steps in the direction of the new estimate. This might make the learning process unstable, and the algorithm might not converge to the correct values. It may bounce around without settling.

2) **Alpha =0.02:** If the learning rate alpha is too low: With a small learning rate, the updates to V(s) are tiny, and the algorithm makes very slow progress in adjusting its estimates. It might take an exceedingly long time to converge, and in this case it has not converged because the estimates are changing too slowly to reflect the true values in the environment.

3) **Alpha=0.1:** If the learning rate alpha is Moderate in TD(0) learning it strikes a balance between the extremes of too high and too low values. It enables the algorithm to adjust its value estimates reasonably, avoiding wild oscillations or overshooting while ensuring steady convergence.

Convergence of RMSE to 0 For constant Alpha?

The Root Mean Square Error for TD(0) with a constant learning rate does not converge to 0 because it can exhibit a form of stochastic behaviour and may not reach the optimal solution for this problems. due to fixed step size α, causing it to overshoot, oscillate, and potentially miss the optimal solution, preventing RMSE convergence to 0.

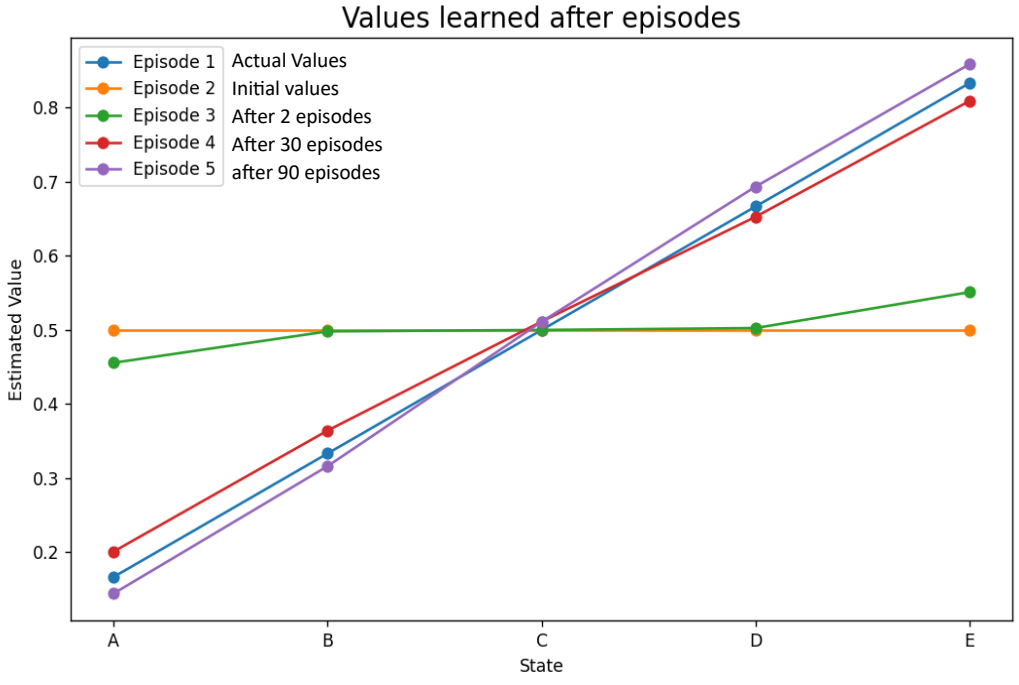
when the learning rate is set to 1/n we observe that the RMSE values are minimum and also for alpha =1/n it converges to 0.

Reason:

1) Constant Learning Rate alpha: When you use a constant learning rate alpha in TD(0) learning, the weight updates are determined solely by this fixed value. TD(0) is an online learning algorithm, which means it updates its estimates after each episode based on the observed rewards and predictions. If the learning rate is too high, it can lead to oscillations and overshooting of the optimal values. If it's too low, it can take a very long time to converge or may never converge at all. It's essential to choose an appropriate learning rate that balances exploration and exploitation. Due to these Oscillations for constant Alpha Convergence to 0 is not Possible.

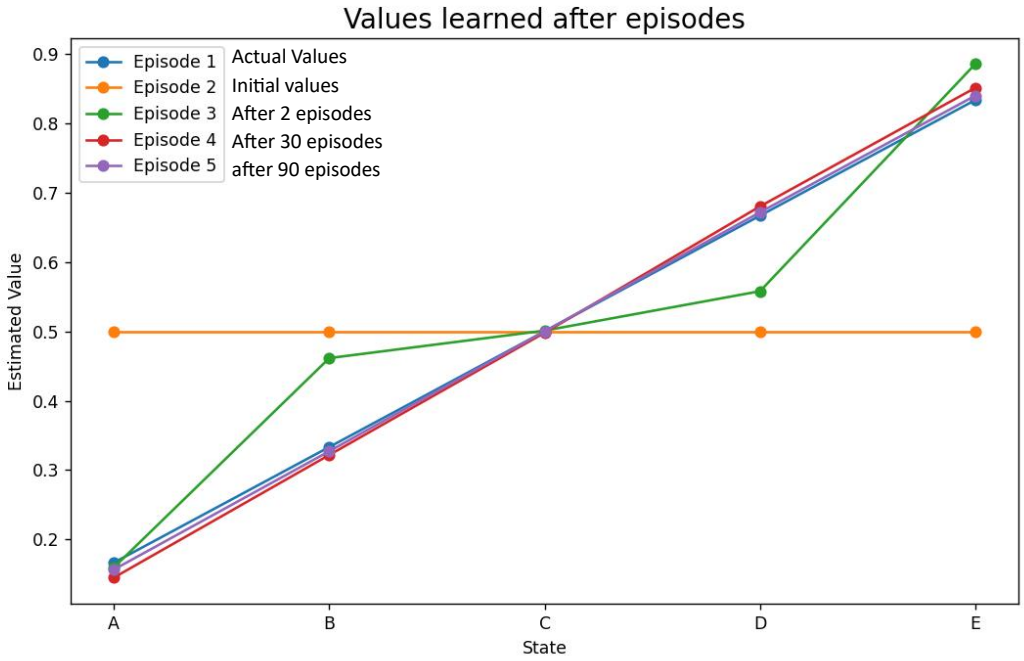
2) 1/n Learning Rate: When you set the learning rate to 1/n, where n is the number of times a state-action pair has been visited. This approach automatically adjusts the learning rate over time. Initially, when n is small, the learning rate is high, allowing the algorithm to explore and update its estimates more quickly. As n increases, the learning rate decreases, which helps to stabilize the learning process. In this case, this approach can guarantee convergence to the true value function under certain conditions.

Constant alpha=0.1



In The above graph we can see that after 30 episodes (red line) and after 90 episodes (purple line) episodes there is no difference in convergence to the actual value (Blue Line) this confirms the oscillating behaviour of the algorithm for a constant alpha

Variable alpha = 1/n



In this graph we can see that after 30 episodes (red line) and after 90 episodes (purple line) episodes we can observe that the 90 episodes (purple line) has converged more to the actual value as compared to the red line so we can say that after a long duration the estimated value will converge to the actual value using alpha =1/n.

Conclusion:

In Conclusion the choice of learning rate is crucial in reinforcement learning algorithms like TD(0). A constant learning rate may not converge to 0 due to issues like oscillations or slow convergence, while a 1/n learning rate adapts over time and can lead to convergence under certain conditions.