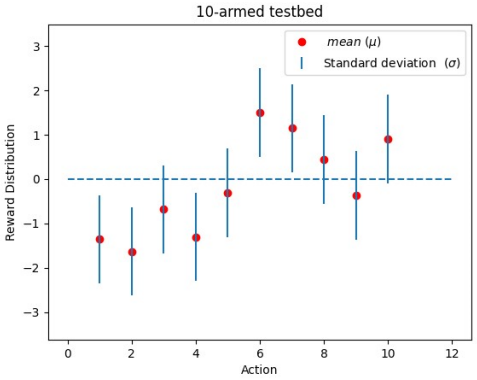
Multi Armed Bandit Problem

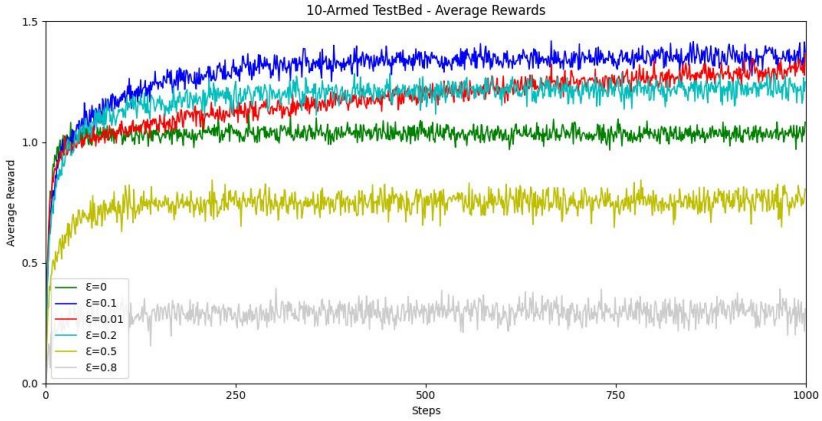
- Recreating 10 armed test bed as described in the textbook, I got these plots:

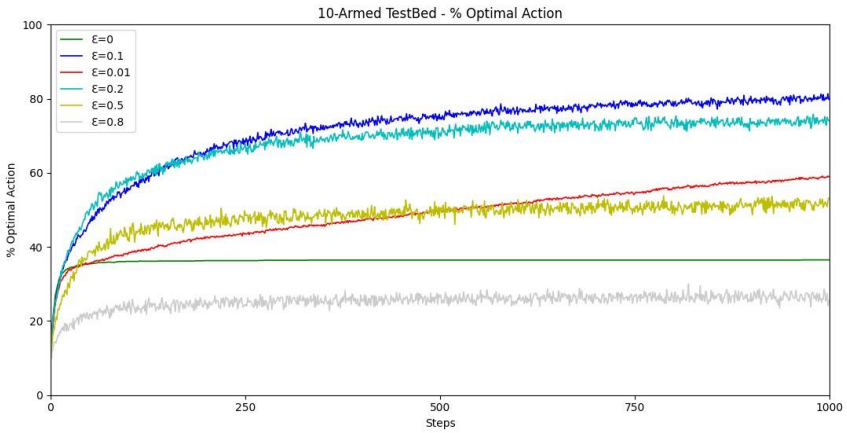


I'm performing 2000 independent runs of 1000 time steps each with k = 10 bandits. The q* values were generated using a normal distribution with mean = 0 and variance = 1.

i) Ɛ Greedy Action Selection

To compare the results from different epsilon values while using the epsilon greedy method, I plotted Average Rewards and %Optimal Action against the time steps for 6 epsilon values: 0 (greedy) , 0.01, 0.1, 0.2, 0.5 and 0.8.



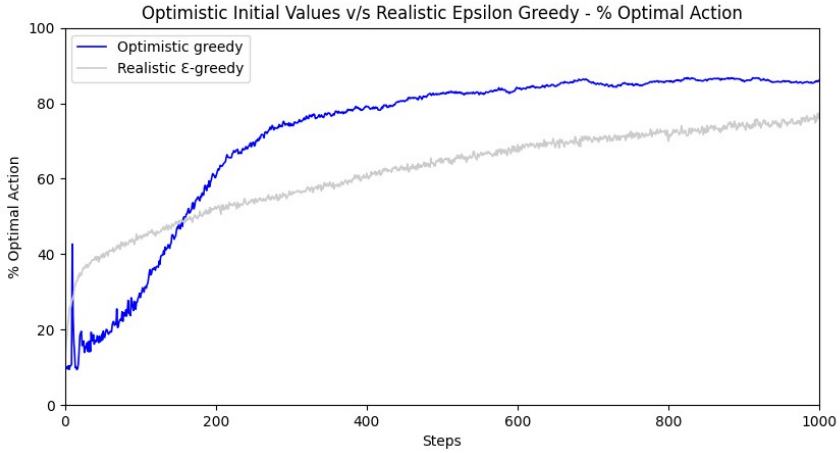


Analysis:

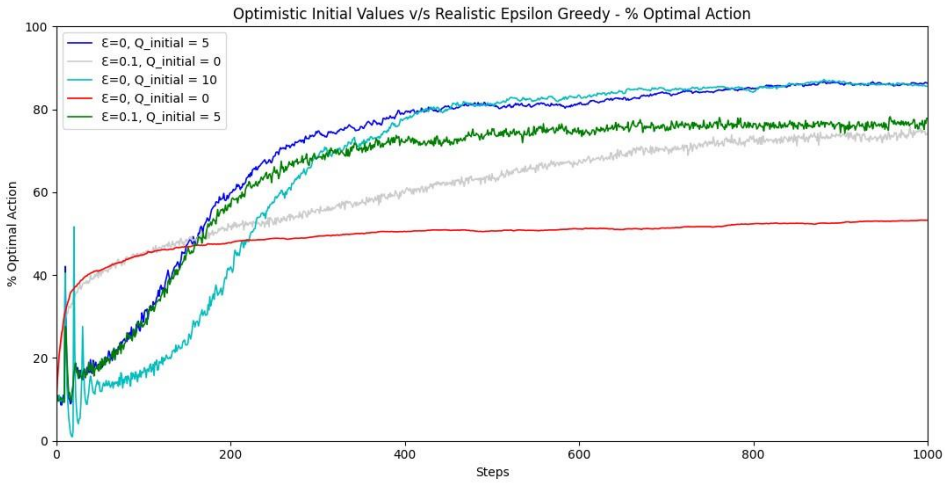
The greedy method improved the fastest but hits a plateau with almost no improvement and average reward and %Optimal Action. Overall, we can say that Ɛ = 0.1 performed the best showing the maximum values in both parameters with a decent growth rate to reach to that point. Ɛ = 0.8 performed the poorest, with performance declining from Ɛ = 0.5 itself. This is a clear indicator that high epsilon values are not beneficial to generate useful results. A possible reasoning for this is that for 10 bandits we get a good enough estimate of rewards for each bandit and thus, should move towards performing greedy actions via exploiting rather than continuing to explore and thus choose less profitable bandits. Ɛ = 0.01 showed slower improvement than Ɛ = 0.1 and 0.2 as it's exploratory power is much slower, but overtakes both in average return over time. Even though 0.2 found the optimal action faster than 0.1, 0.1 shows a better average reward than 0.2 consistently as it chooses the optimal action more often.

ii) Optimistic Initial Value

As done in the textbook, I compared Optimistic Greedy using initial value = 5 & Ɛ = 0 and realistic epsilon greedy using initial value = 0 & Ɛ = 0.1. Alpha in both cases was a constant = 0.1



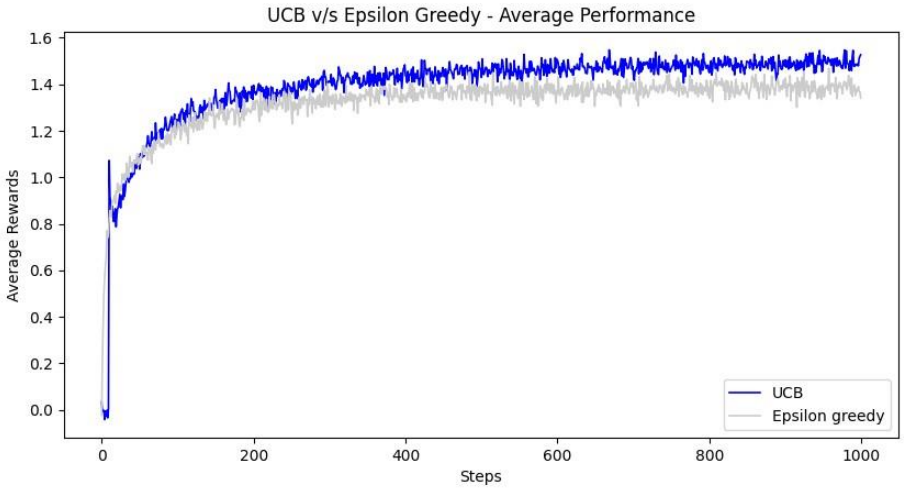
Optimistic Initial Value is a method that encourages exploration without changing the epsilon value. At the beginning it performs worse as it's busy exploring, but in the long run it performs better as exploration decreases with time and it found a high reward value which gives good returns in its exploration phase. I tried tuning the parameters further:

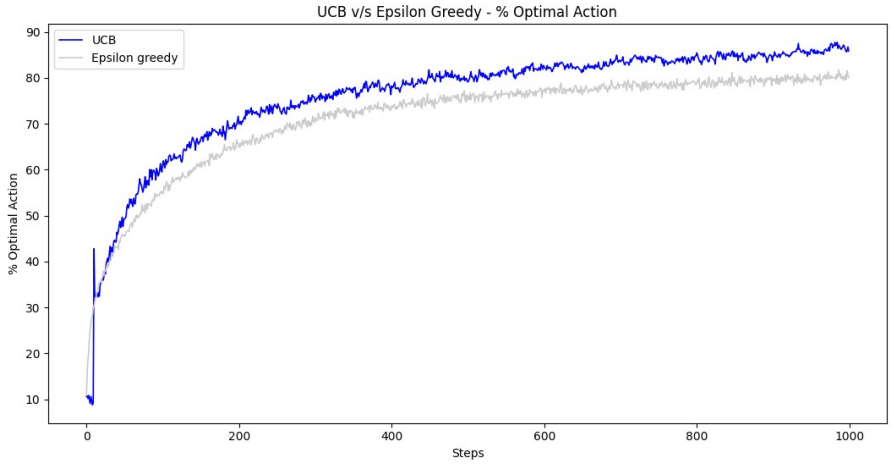


Analysis: When we tried increasing the initial value from 5 to 10, we see faster growth at the beginning, owed to the fact that it is more biased towards exploration due to higher initial value. After some time has passed, the effect of the initial value wears off and both graphs somewhat converge. There are significant spikes in both these graphs, especially when the initial value is high. This is owed to the fact that in the beginning, we start with overly optimistic estimates for all actions so we randomly pick actions without much success, resulting in low probability of picking the best one. As we continue, we'll see improvement because our estimates are becoming more accurate and less overly optimistic. When comparing between two models with the same epsilon but with different initial values, we see that for around the first 150 runs, no optimistic initial value finds the optimal action with much more frequency than when initial value = 5. The second model starts performing better eventually but the advantage wears off after 1000 time steps and they converge. An interesting thing to note is that models with $\varepsilon = 0$ and optimistic initial value outperform those with added exploratory power when $\varepsilon > 0$ in the long run. A possible explanation for this is that the initial bias helps the model find out actions which give better rewards, and once the bias wears off, the model greedily picks these better models, whereas when exploration happens through epsilon values, the model keeps exploring in some runs even once the optimal value has been found out, thus decreasing performance.

iii) Upper Confidence Bound (UCB) Action Selection

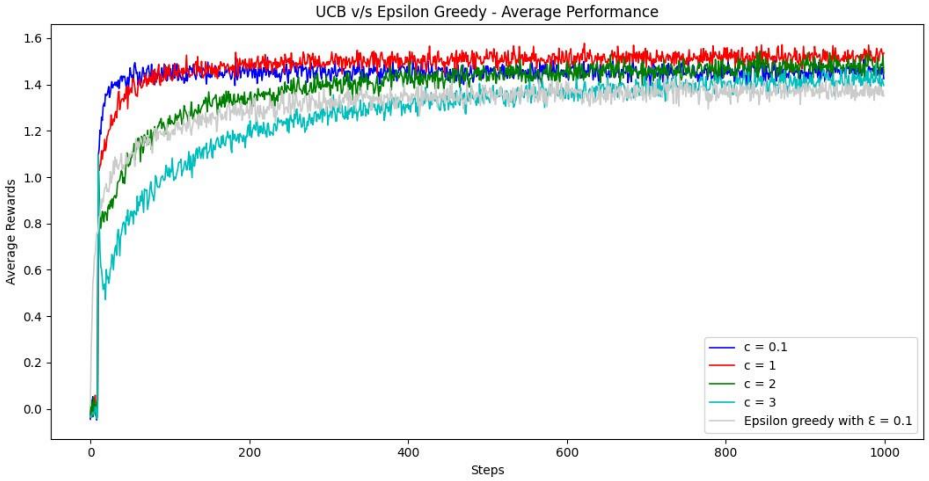
As done in the textbook, I compared UCB with epsilon greedy, taking c = 2 and & $\varepsilon = 0.1$.

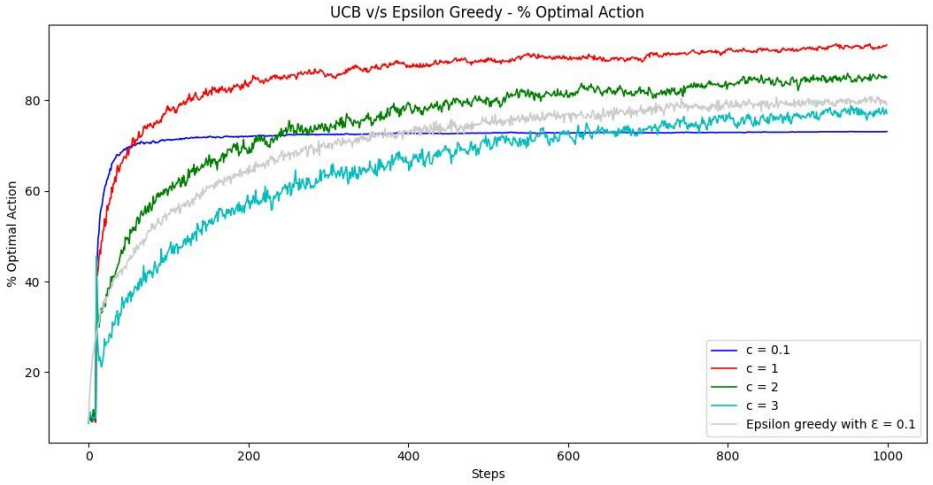




UCB outperforms epsilon-greedy because it learns to balance trying new actions and sticking with known good actions more effectively by considering uncertainty in action values, leading to smarter decision-making. Epsilon-greedy is random and doesn't adapt as well to the changing environment. We used this formula to adapt to the new information:

$$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right],$$





As visible, c = 1 outperforms the others in both the graphs. Greedy performs worse than c = 1 and c = 2 but better than c = 3. C = 0.1 finds out the optimal action much faster, but then immediately plateaus and after 400 runs has lesser optimal action% than greedy, but still has higher average returns. In terms of average returns, all of the UCB methods perform better than greedy in the long term.