

# Comparison of $\epsilon$ -greedy, decaying $\epsilon$ -greedy and UCB in a Multi Armed Bandit setting

Amirthalingam Rajasundar

January 6, 2026

## 1 Introduction

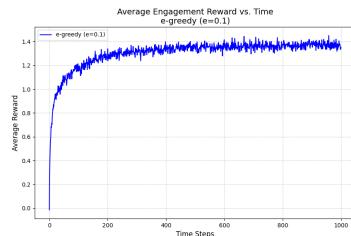
This report details an investigation into various multi-armed bandit strategies for ad selection, focusing on their performance in maximizing user engagement. We compare the  $\epsilon$ -greedy, decaying  $\epsilon$ -greedy, and Upper Confidence Bound (UCB) algorithms. The performance of each strategy is evaluated based on the average engagement reward over time and the frequency of selecting the optimal ad.

## 2 Code Implementation

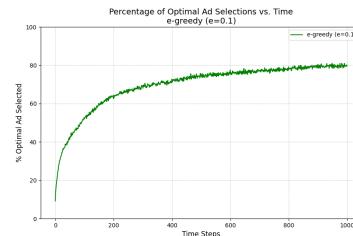
Refer main.py for the environment definition and solver implementations.

## 3 Plots

### 3.1 $\epsilon$ -greedy ( $\epsilon = 0.1$ )



(a) Average reward over time



(b) Optimal ad selection frequency

Figure 1: Performance of  $\epsilon$ -greedy with  $\epsilon = 0.1$

### 3.2 $\epsilon$ -greedy ( $\epsilon = 0.2$ )

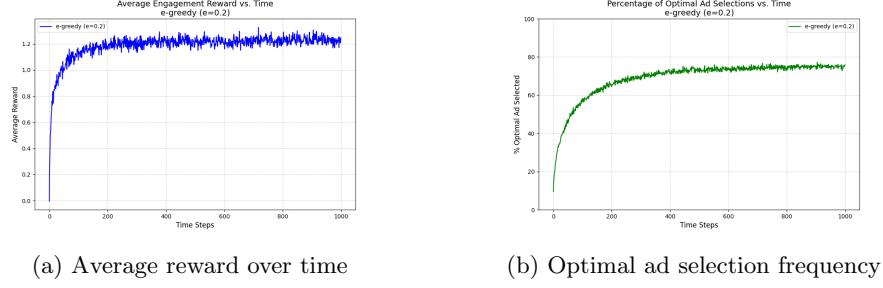


Figure 2: Performance of  $\epsilon$ -greedy with  $\epsilon = 0.2$

### 3.3 Decaying $\epsilon$ -greedy

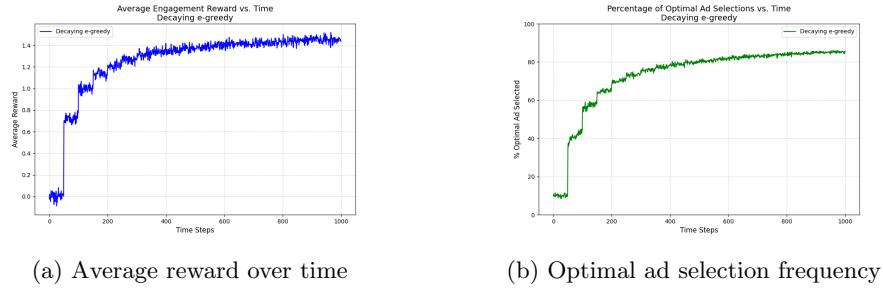


Figure 3: Performance of decaying  $\epsilon$ -greedy

### 3.4 UCB ( $c=1$ )

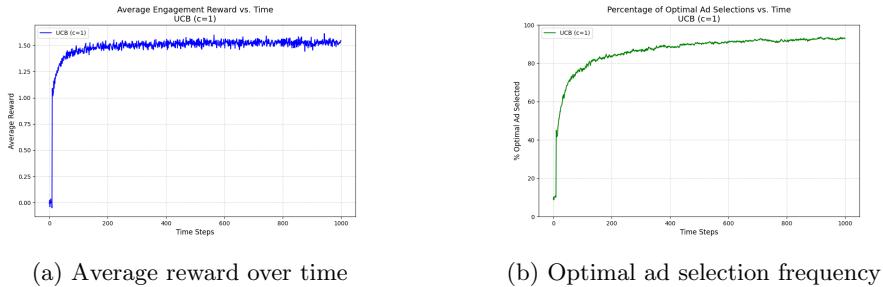


Figure 4: Performance of UCB with  $c=1$

### 3.5 UCB ( $c=10$ )

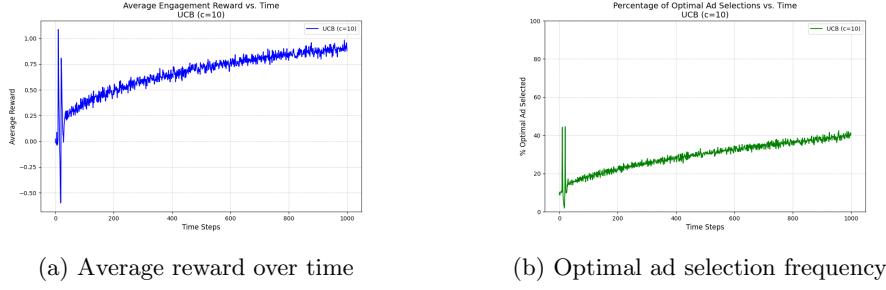


Figure 5: Performance of UCB with  $c=10$

## 4 Discussion and Comparison

- **Average reward over time - initial reward performance:** At the 200th time step, we can observe that e-greedy (0.1) is at around 1.3, e-greedy (0.2) is at around 1.2, decaying e-greedy is at around 1.1, ucb ( $c=1$ ) is at around 1.5, UCB( $c=10$ ) is at around 0.5. So, UCB( $c=1$ ) achieves better rewards faster resulting in a steeper curve. Between  $e=0.1$  and  $e=0.2$ , lower  $e$  value seems to perform better as it exploits more often. Decaying greedy also has a poor initial performance as it explores more in the initial time steps. Between  $c=1$  and  $c=10$ ,  $c=1$  seems to have better initial performance as a higher value of  $c$ , gives more weight to exploration.
- **Average reward over time - final reward performance:** At the 1000th time step, we can observe that e-greedy (0.1) is at around 1.35, e-greedy (0.2) is at around 1.21, decaying e-greedy is at around 1.41, UCB( $c=1$ ) is at around 1.51, UCB( $c=10$ ) is at around 0.9. So, UCB( $c=1$ ) achieves better rewards towards the end, when compared to all other algorithms. Although, decaying e-greedy had poor initial performance, it achieved better rewards towards the end when compared to the other e-greedy algorithms without decay. This is because, the decaying e-greedy algorithm performs better exploration during the initial time steps and exploits the best performing ad more often towards the end. UCB ( $c=10$ ) might need more steps to converge. We can also observe that the UCB( $c=10$ ) curve shows frequent spikes and dips, reflecting its high exploration.
- **Optimal ad selection frequency:** Looking at the 'Optimal ad selection frequency graphs', we can observe that UCB( $c=1$ ) performs the best and selects the optimal ad 82 percent of the time as early as the 200th time step. Towards the end, it selects the optimal ad 92 percent of the time. UCB ( $c=10$ ) has the worst performance as it gives more weight to exploration and requires longer to converge. Among the e-greedy algorithms,

decaying  $\epsilon$ -greedy algorithm has the best performance as it exploits more often in the later time steps.  $\epsilon$ -greedy ( $\epsilon=0.2$ ) comparatively has a poor performance as it explores more often.

- **$\epsilon$ -greedy:** Simple to implement but suffers from a constant exploration rate. This is inefficient as it continues to explore even when a good policy has been found, limiting its overall performance.
- **Decaying  $\epsilon$ -greedy:** A significant improvement over the basic  $\epsilon$ -greedy. By decaying the exploration rate, it effectively transitions from a phase of learning to one of exploiting the learned knowledge, leading to better long-term performance.
- **UCB:** The most sophisticated of the three. It balances exploration and exploitation by considering both the mean rewards and the number of selections for each ad. This method is generally the most effective at identifying the optimal ad quickly and consistently, leading to superior performance in the long run.

In summary, while  $\epsilon$ -greedy is a good starting point, both decaying  $\epsilon$ -greedy and UCB provide much better performance. The UCB algorithm, in particular, is an excellent choice for real-world applications where maximizing reward over time is the primary objective.