Reinforcement Learning: Assignment 1
DSCI-6650

Soroush Baghernezhad
sbaghernezha@mun.ca

June 9, 2024

Abstract

In this assignment, we investigate different approaches to solving the 10-armed bandit problem in both stationary and non-stationary environments. Our results show how these algorithms can gradually learn to achieve higher rewards. Finally, we compare the different methods to determine which one performs better.

1 Part One

1.1 greedy with non-optimistic initial values

In this section, we initialized the action value estimates to 0 and used the incremental implementation of the simple average method. The incremental update formula for action values is given by:

$$Q_{n+1}(a) = Q_n(a) + \frac{1}{N_n(a) + 1} \left(R_{n+1} - Q_n(a)\right)$$

Where:

- $Q_{n+1}(a)$ is the updated action value after the $(n+1)$-th reward.
- $Q_n(a)$ is the current action value before the update.
- $N_n(a)$ is the number of times action a has been selected before the $(n+1)$-th selection.
- R_{n+1} is the reward received after the $(n+1)$-th selection of action a.

1.2 epsilon-greedy with different choices of epsilon

The epsilon-greedy algorithm is a simple yet effective approach used in multi-armed bandit problems to balance exploration and exploitation. At each time step, with probability ϵ, a random action is selected to explore the action space; with probability $1 - \epsilon$, the action with the highest estimated value is chosen to exploit the known information. This method ensures that the algorithm occasionally explores suboptimal actions to discover their potential rewards, preventing the premature convergence to a suboptimal action. The value of ϵ is typically small (e.g., 0.1), allowing for sufficient exploration while predominantly exploiting the best-known actions.
In this part, we use ϵ greedy method. For choosing the best epsilon, we did a pilot run over a set of different epsilons.
$\epsilon \in \{0.01, 0.1, 0.2, 0.3\}$

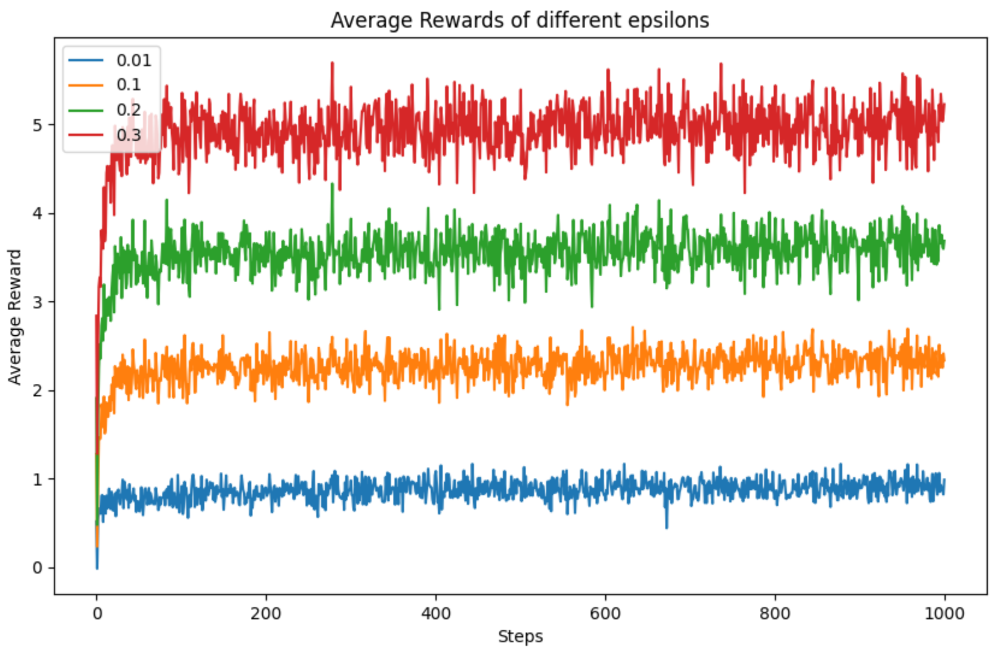

Figure 1: average reward of different epsilons

1.3 Optimistic Starting Values with a Greedy Approach

In this section, we initialized Q_{estimate} using the mean of our action values, as specified in the project description. This optimism allows us to explore more actions since the initial values are higher than the actual Q value.

1.4 gradient bandit algorithm

The gradient bandit algorithm is an approach used in reinforcement learning to optimize the selection of actions in multi-armed bandit problems by using gradient ascent methods. Unlike traditional methods that estimate action values, the gradient bandit algorithm maintains a preference for each action, which is updated based on the received rewards. The preference for an action a at time t, denoted as $H_t(a)$, is adjusted by an amount proportional to the difference between the received reward and the average reward, scaled by a step-size parameter α. This update rule can be expressed as $H_{t+1}(a) = H_t(a)+\alpha(R_t-\bar{R}_t)(1-\pi_t(a))$ for the selected action, and $H_{t+1}(a) = H_t(a)-\alpha(R_t-\bar{R}_t)\pi_t(a)$ for all other actions, where $\pi_t(a)$ is the probability of selecting action a at time t. The probabilities are derived from the preferences using a softmax distribution: $\pi_t(a) = \frac{\exp(H_t(a))}{\sum_b \exp(H_t(b))}$. This method effectively balances exploration and exploitation by dynamically adjusting the action preferences based on the observed rewards.

2 Part Two

In this section, we deal with a non-stationary problem in which rewards can change both gradually or abruptly. gradual changes follow this formula:
drift: $\mu_t = \mu_{t-1} + \epsilon_t$
revert: $\mu_t = \kappa\mu_{t-1} + \epsilon_t$ where $\kappa = 0.5$
abrupt changes permute action values indices by probability of 0.005
Since reward distribution is changing, plotting the average rewards per steps will not be meaningful, therefore, we run many steps, e.g. 10,000 and 20,000 and then draw the box plot of average rewards over many problems. We also plotted the optimal action selection percentage for the three methods over different reward changing strategies.

Apart from setups mentioned above, our functions and method we are used are same as part one.

3 Results: Part One

Comparing different algorithms based on their average accumulated rewards depicted in figure 2 we can see that the Greedy algorithm performed the worst. This is mainly because it completely sticks to the best action based on the current Q value. If it finds an action that is slightly better than the others according to Q, it chooses only that action for the rest of the steps (assuming we initialize all Q values with 0 initially).

Epsilon-greedy, on the other hand, attempts to address the aforementioned issue by randomly exploring different actions, considering the value of ϵ (which was determined through pilot runs).

Optimistic initialization yields better average rewards in the first few steps since the $Q_{\text{estimates}}$ tend to be higher than their actual values, thereby encouraging exploration. However, due to following a greedy action selection afterward, it takes time to find the optimal action.
Optimistic initialization where we set Q values as follow:

$$Q = \max(\text{action\_values\_means}) + 1$$

The Gradient Bandit algorithm performed the best among the others. Initially, it may seem slower to find the optimal actions and achieve better average rewards. However, over time, due to its gradient characteristics, it outperforms other methods in terms of the total average reward obtained in the long run.

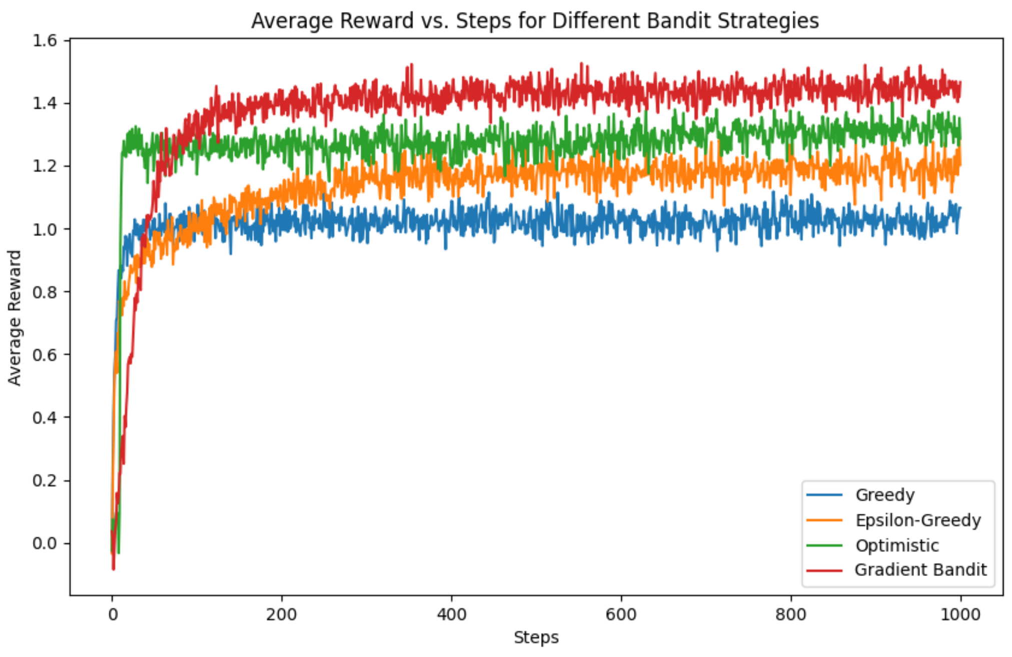

Figure 2: average rewards of different bandit algorithm over 1000 problems.

Figure 3 shows interesting insights over algorithms. The constant rate of choosing the optimal action for greedy method explains why we call it greedy. On the other hand, $\epsilon greedy$ moderately choose optimal actions after some steps which is the result of exploration and finding out that other actions might have better values even though their action values estimation is not as good as other actions in the current time step. Optimistic initialization results shows the slow and steady improve of choosing optimal actions till it converge and choose the optimal action.
Gradient method with the use of action preferences chooses the most optimal actions between other methods.

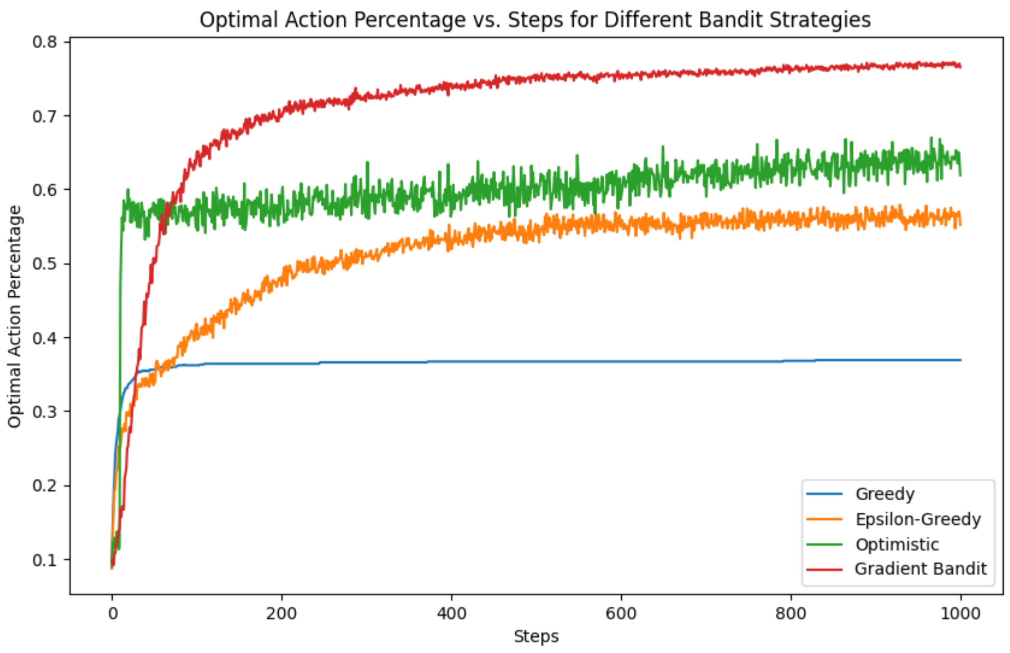

Figure 3: optimal action selection percentage

To tune greedy method, we can add optimistic Q values initialization and ϵ to further explore actions.

4 Result: Part Two

Figure 4 and Figure 5 show how our three methods work under gradual changes in the reward distribution. As before, the optimistic greedy approach gets more average rewards in the long run. The ϵ-greedy method with a decreasing step size performs slightly better in 10k steps because the algorithm tends to be greedier towards the end. This result can also be concluded by looking at Figure 10 and 13 in which the optimistic greedy method chooses more optimal actions compared to the other two.

In the long run (20k steps), the average reward range is much higher because the mean of the reward distribution is gradually drifting to higher values.

For the gradual revert setup, where $\kappa = 0.5$, the mean of the reward distribution converges to zero. Therefore, regardless of the method followed, the reward you get is nearly zero. Figure 6 and 7.
In the abrupt changes setting, all three methods under experiment could not achieve good average rewards because the action values' indices are subject to permutations. Figure8 and 9

Figure 15 illustrates this concept, showing that the algorithms experience wide fluctuations in selecting an optimal action. This instability is due to the abrupt changes causing the action values' indices to permute frequently.

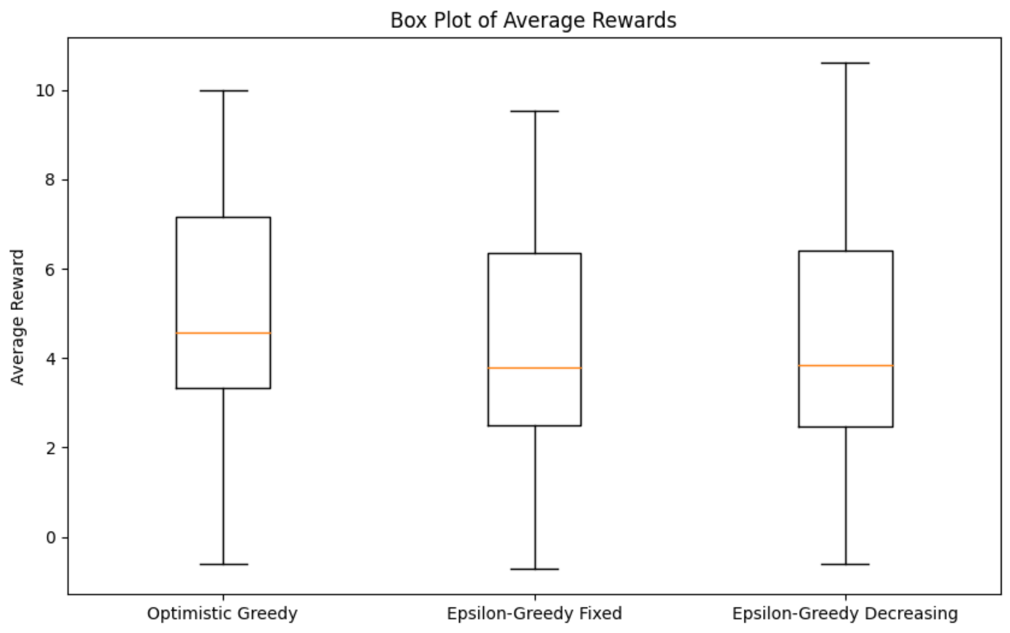

Figure 4: average rewards for gradual drift after 10k steps over 1k problems

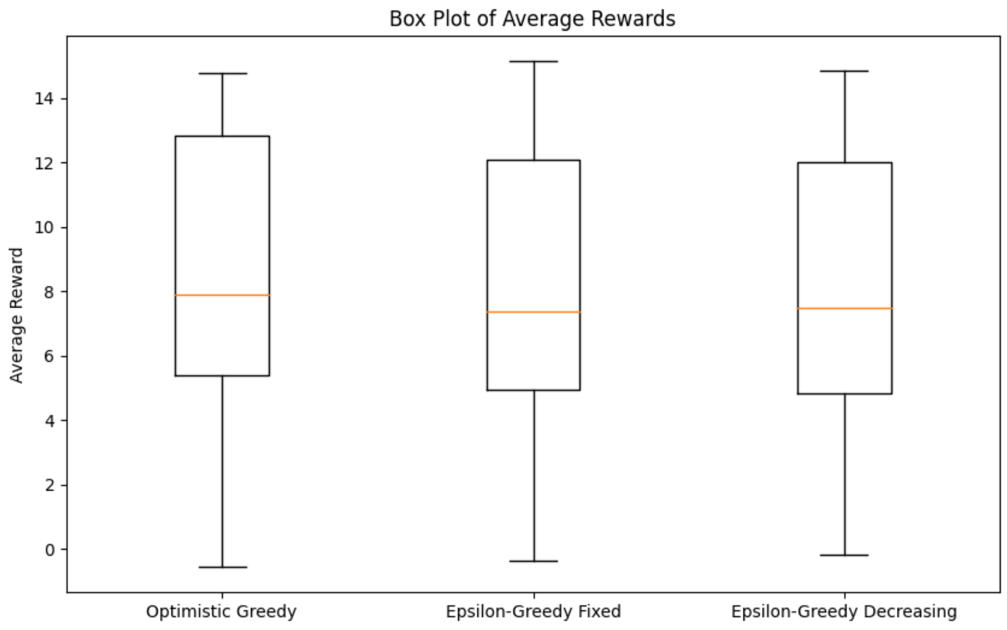

Figure 5: average rewards for gradual drift after 20k steps over 1k problems

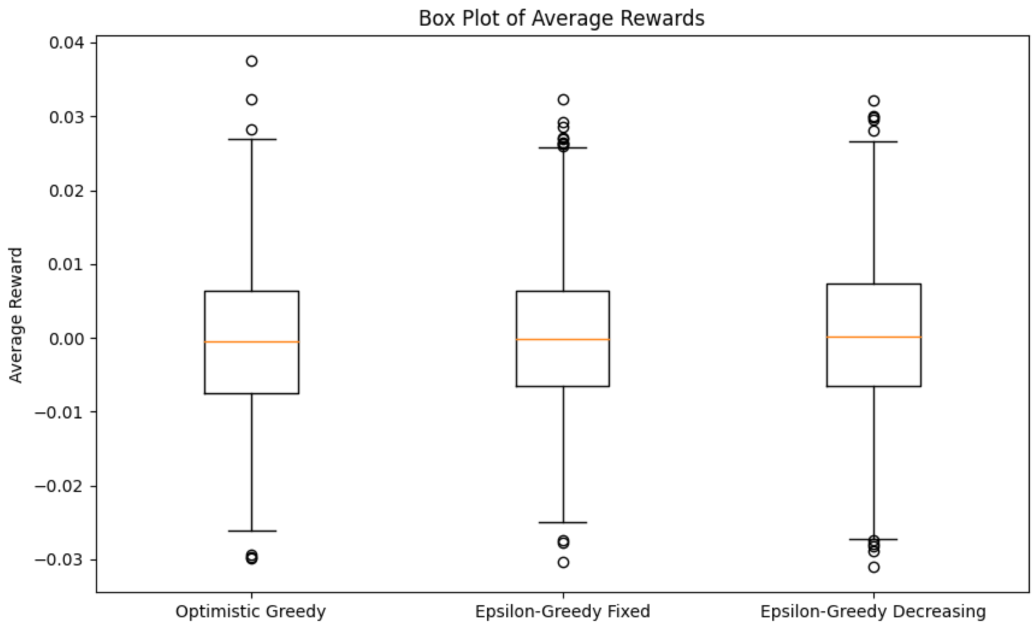

Figure 6: average rewards for gradual revert after 10k steps over 1k problems

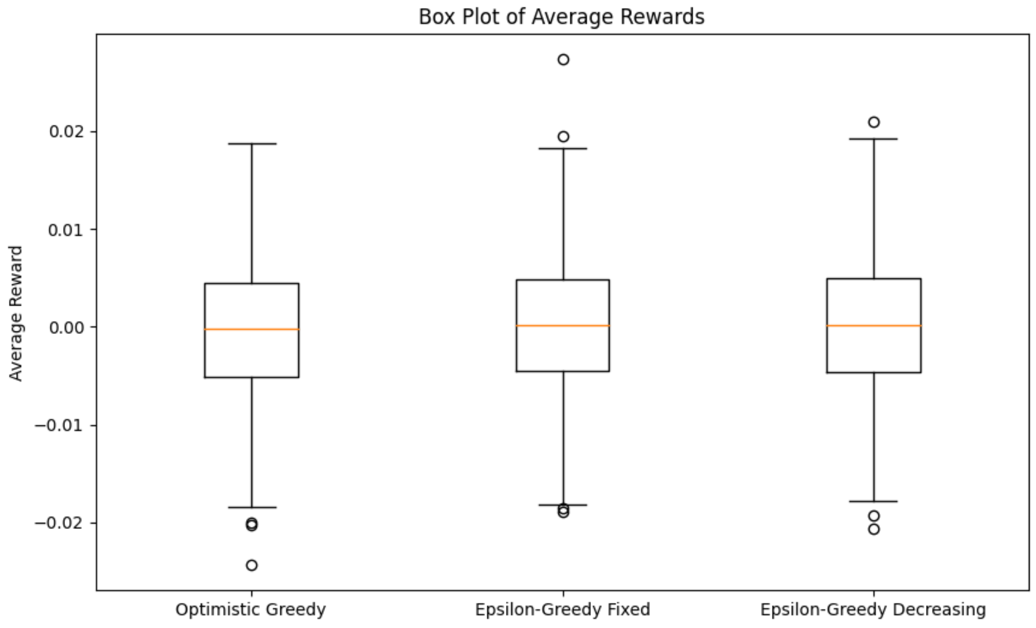

Figure 7: average rewards for gradual revert after 20k steps over 1k problems

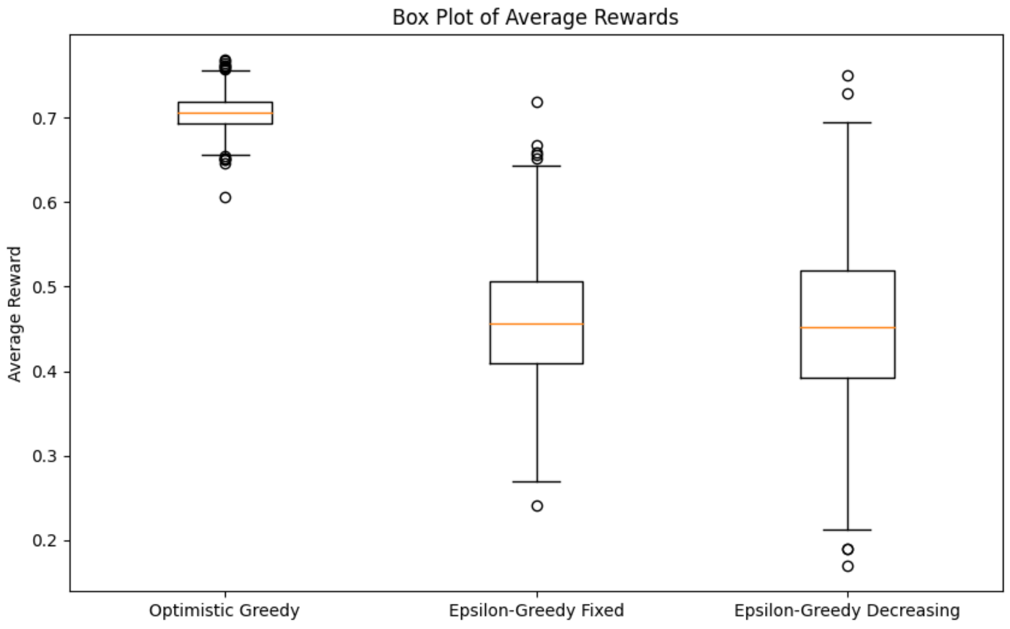

Figure 8: average rewards for abrupt change after 10k steps over 1k problems

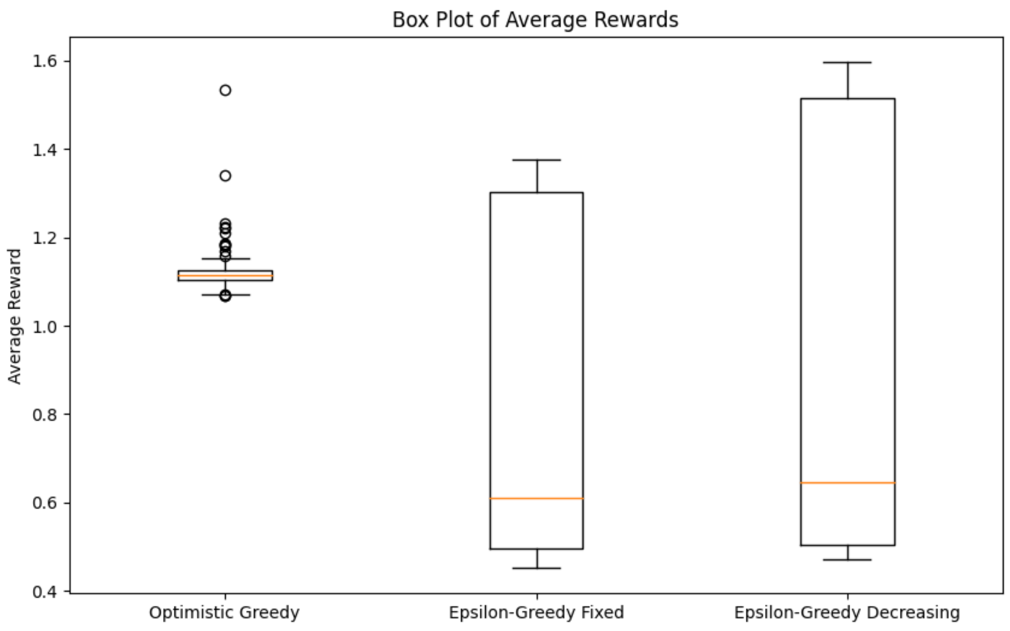

Figure 9: average rewards for abrupt change after 20k steps over 1k problems

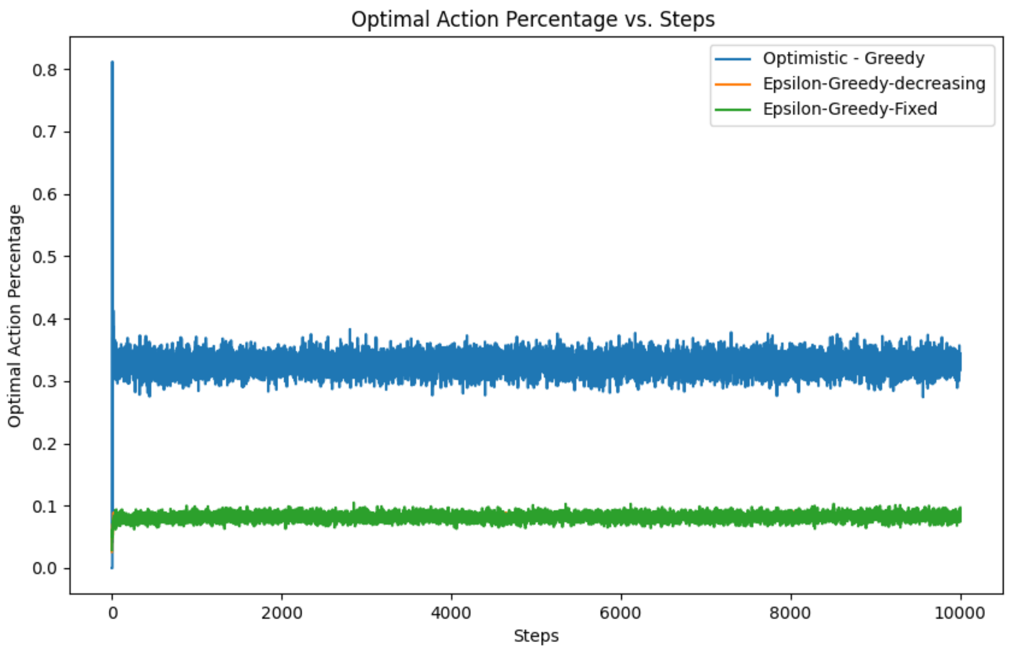

Figure 10: Optimal action selection after 10k steps over 1k problems with gradual drift

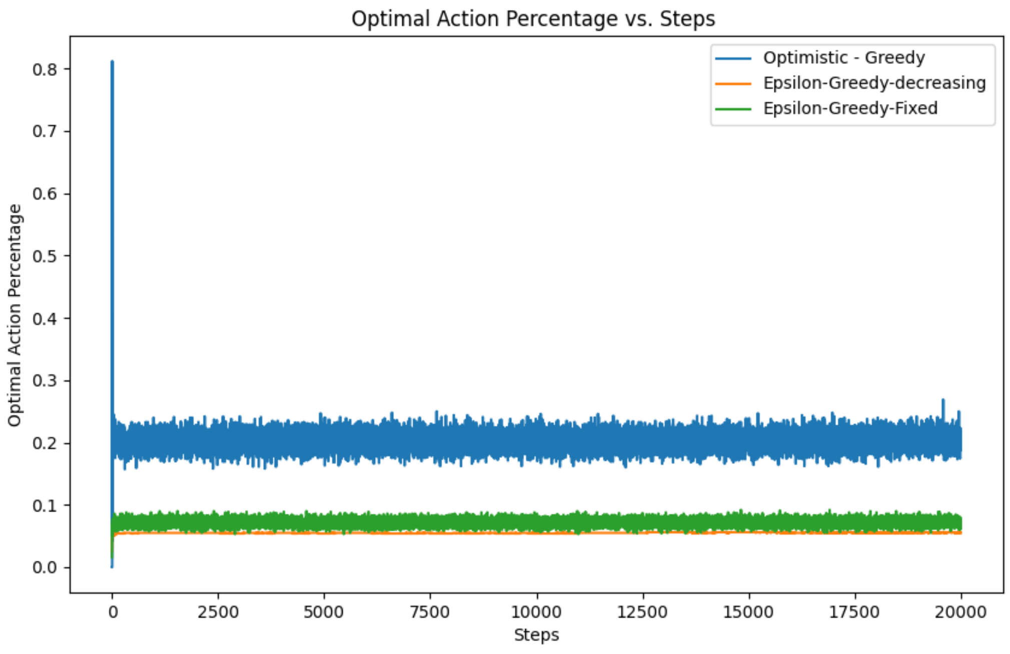

Figure 11: Optimal action selection after 20k steps over 1k problems with gradual drift

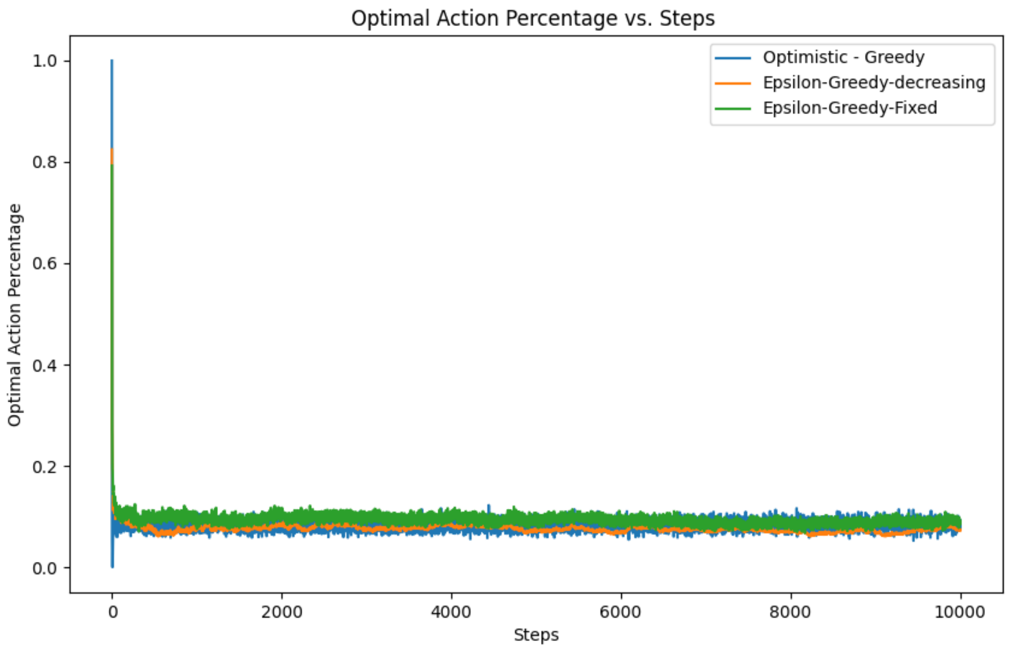

Figure 12: Optimal action selection after 10k steps over 1k problems with gradual revert

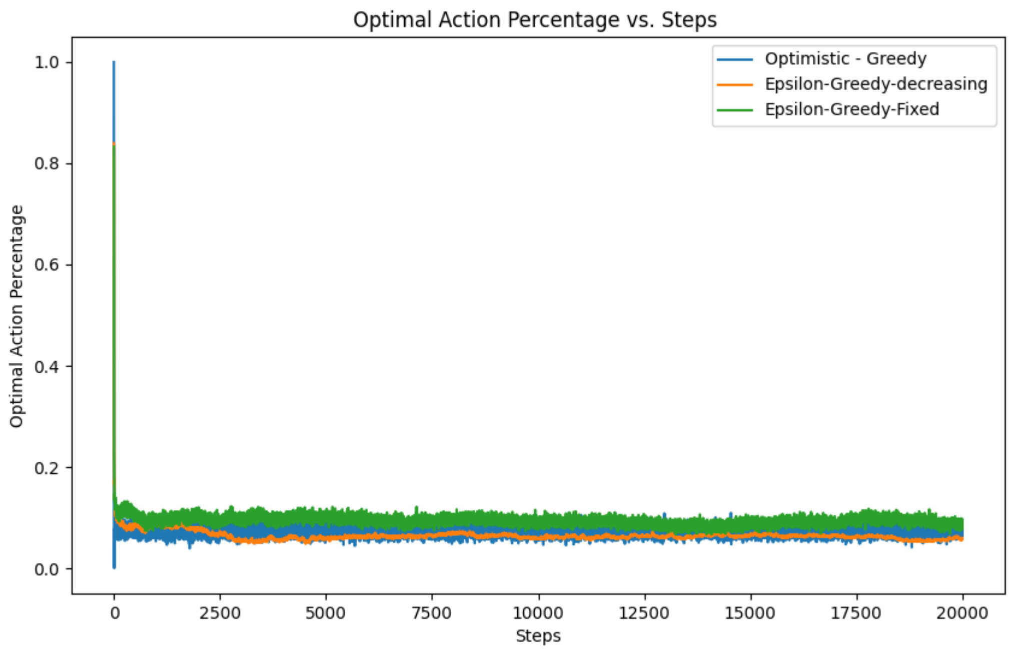

Figure 13: Optimal action selection after 20k steps over 1k problems with gradual revert

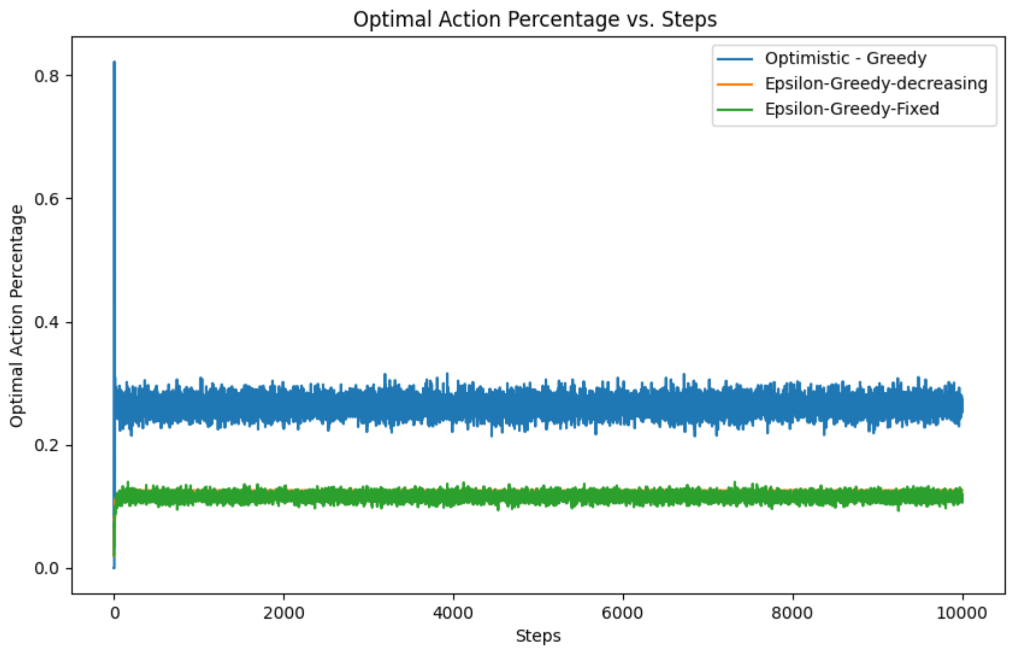

Figure 14: Optimal action selection after 10k steps over 1k problems with abrupt changes

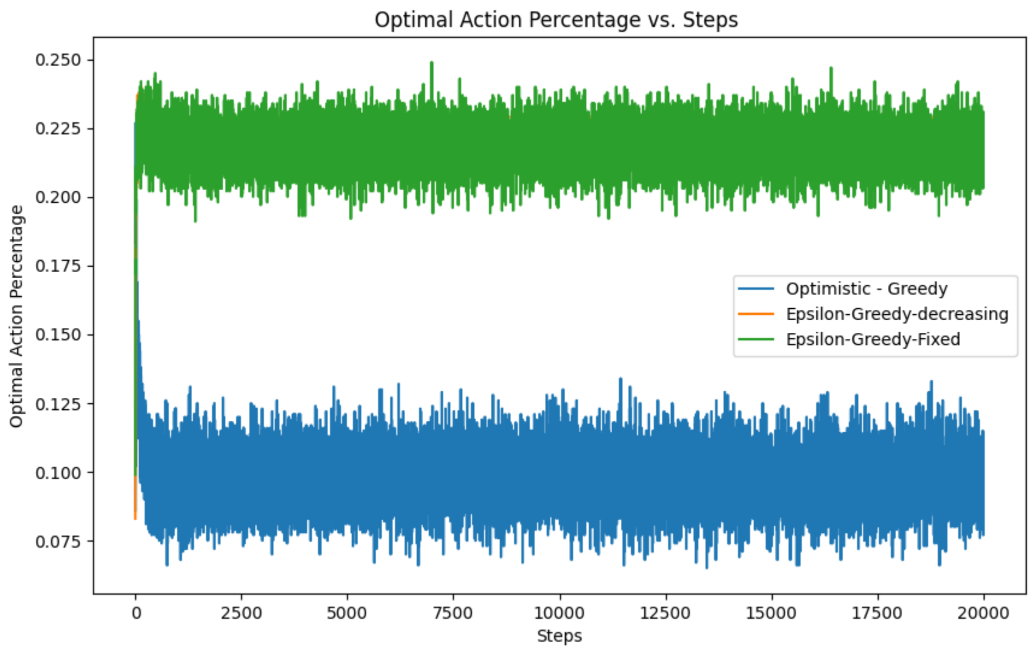

Figure 15: Optimal action selection after 20k steps over 1k problems with abrupt changes

References

ichard S. Sutton and Andrew G. Barto. 2018. Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA.