CS747 Assignment 1

Gurparkash Singh
160050112

September 3, 2019

Q1. Regret can be negative, especially on small horizons. Why?
A1. Regret is the difference between maximum expected reward and the observed value of reward for the given value of horizon. If we consider a large amount of time, and keep sampling the best arm, we would expect the regret to be 0. However, for a small horizon, we might get lucky and sample an arm and get more reward than is expected. For example if we have 2 arms of probabilities 0.5 each, and we have a horizon of 10, the maximum expected reward will be 5. However, it may so happen that on randomly choosing the arms and pulling them, we end up having a total reward of 6 or 7. This will result in a negative reward. However, for a large number of samples, because of the law of large numbers, we will, on an expectation get only 0.5t reward. This will make the regret = 0. Furthermore, we are not always sampling the best arm either. Therefore, in most cases the regret should be positive. However, as explained before, for smaller horizons, we might end up having a negative value of regret.

Explanations and Assumptions

1. Round Robin Algorithm

In the Round Robin Algorithm, I'm starting the sampling from the first arm and going sequentially over all the arms before wrapping back onto the first and repeating the process all over again.

2. Epsilon Greedy Algorithm

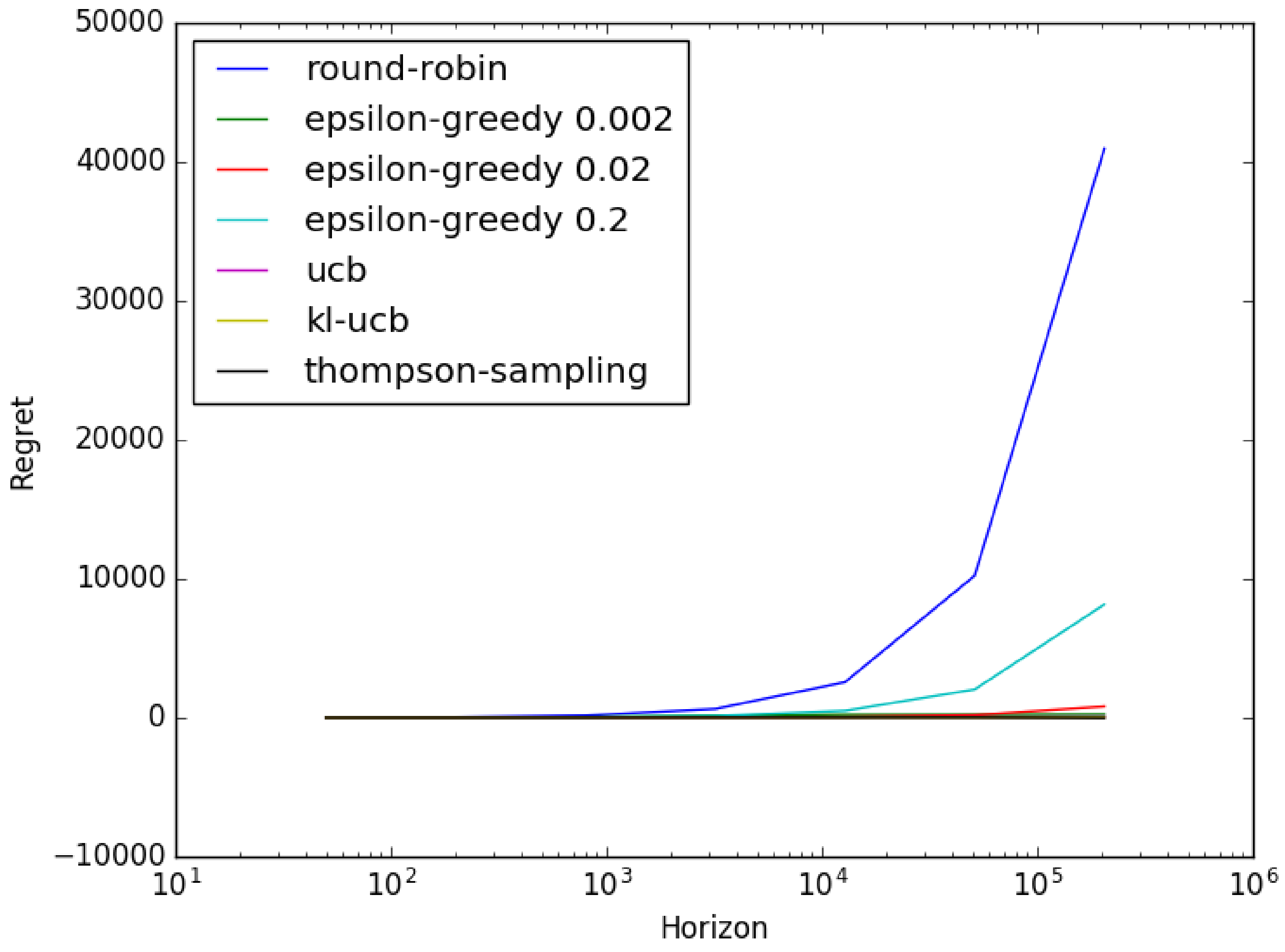
In the Epsilon Greedy Algorithm, while choosing the argmax over the empirical probabilities, if an arm has not been pulled before, I'm considering it's empirical probability to be equal to 0.5. This probability will be appropriately changed once the arm has been sampled.

3. KL-UCB Algorithm

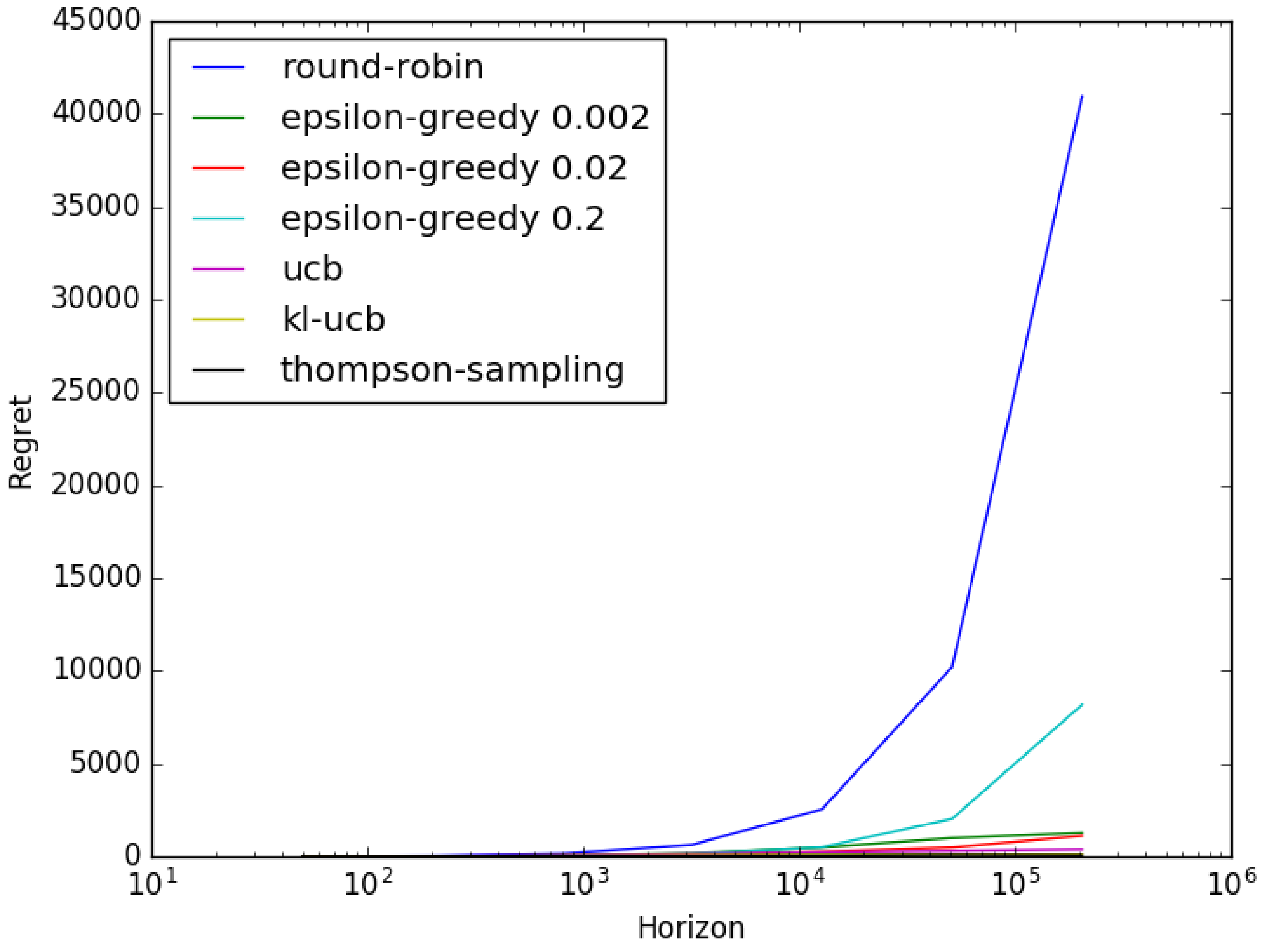
In the KL-UCB Algorithm, we need to find the highest value q between p and 1, such that the value of another function is maximized. This function happens to be monotonically increasing (or decreasing, based on how you pose it). Therefore, we just need to solve an equation to get that value. Various numerical analysis techniques such as Bisection Method or Newton Raphson Method can be used. I'm using the Bisection Method since I find it more elegant to implement. I'm using the precision value $(\epsilon) = 10^{-4}$. I believe this will give an answer sufficiently close to the true value.

Plots and Observations

1. Instance 1

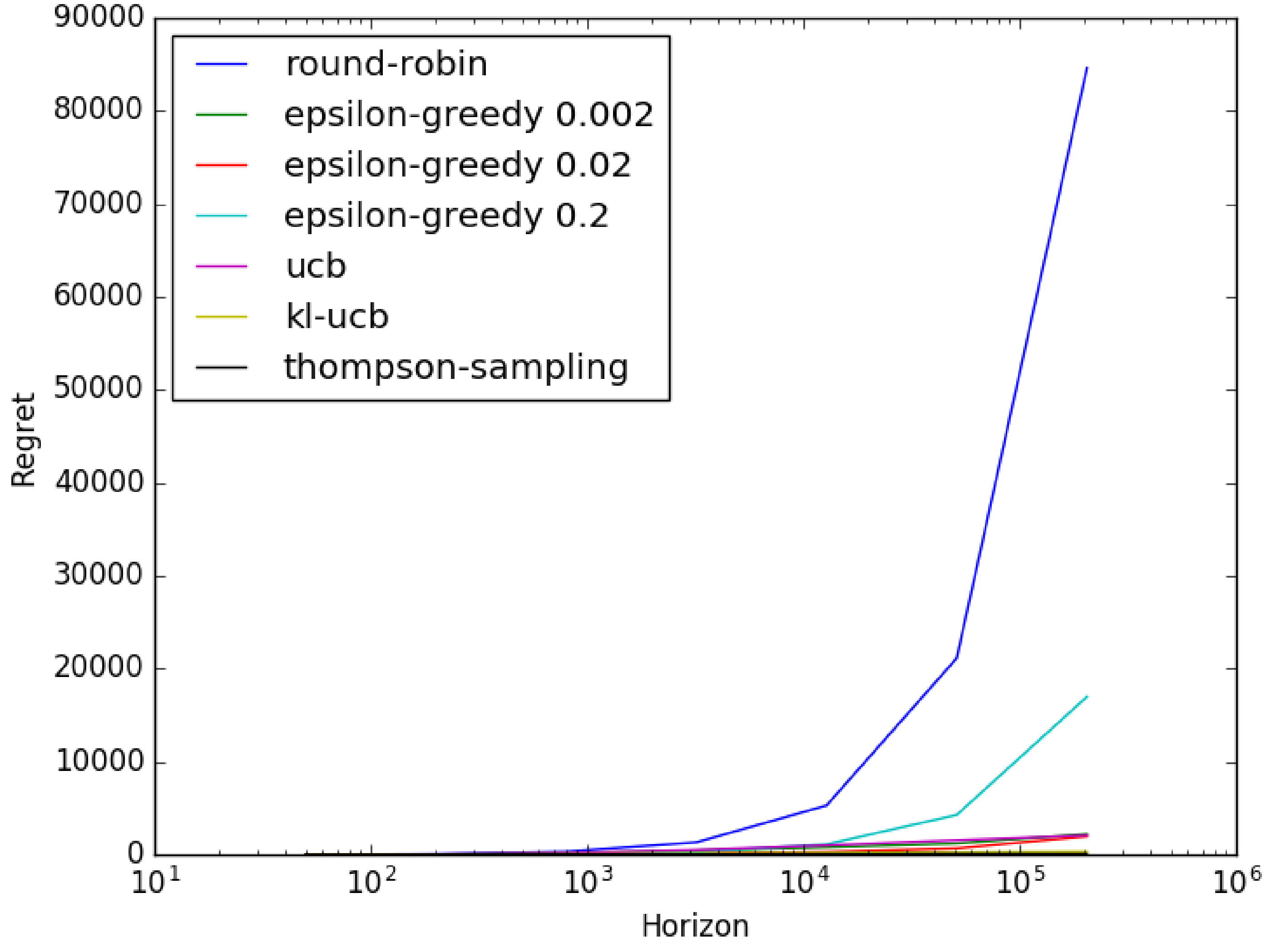


2. **Instance 2**



round-robin
epsilon-greedy 0.002
epsilon-greedy 0.02
epsilon-greedy 0.2
ucb
kl-ucb
thompson-sampling
Regret
Horizon
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
10^1
10^2
10^3
10^4
10^5
10^6

3. **Instance 3**



4. **General Observations**

We can clearly see that the Round Robin Algorithm performs the worse, since we are not using the learned information in any way. For large values of horizon, the Epsilon Greedy Algorithm with $\epsilon = 0.2$ works the worst amongst the 3 values since higher the value of epsilon, the more we are exploring and the less we are exploiting. For a large value of horizon, we tend to explore enough even for smaller values of epsilon, therefore the regret is lower for smaller values of epsilon. The best algorithm amongst the given turns out to be Thompson Sampling Algorithm, but the values aren't much different as compared to KL-UCB.