CS747 : Foundations of Intelligent & Learning Agents

Programming Assignment 1

REPORT

Vinit P. Doke
190260018[1]

[1] Email: 190260018@iitb.ac.in

September 9, 2021

Content

1 Task 1

1.1 Implementation Details

1.1.1 Epsilon Greedy Algorithm

For given ϵ, we explore or exploit based on a random sample drawn from a Uniform Distribution bounded in $[0, 1]$. We also initialize the empirical means for each arm by sampling each arm once at the start and then proceeding according to the epsilon greedy algorithm.

1.1.2 Upper Confidence Bound Algorithm

At first, empirical means of each arm are initialised by sampling/pulling each bandit arm once. Then, UCB Algorithm is followed upto the horizon. The bound is calculated as follows:

$$ucb_a^t = \hat{p_a}^t + \sqrt{\frac{2 \cdot ln(t)}{u_a^t}}$$

here, ucb_a^t is the Upper Confidence Bound of arm a at the t^{th} pull. $\hat{p_a}^t$ is the empirical mean of arm a. u_a^t is the number of times arm a was pulled so far in the run. Here, $c = 2$

1.1.3 KL-UCB Algorithm

At the start, each arm is pulled once to have a non-zero value for the number of times arm is pulled and also to initialize the empirical means of each arm. Thereafter, KL-UCB Algorithm is followed with the bound q given by solving a binary search on the inequality given as follows:

$$\text{KL}(\hat{p_a}^t, q) \leq \frac{[ln(t) + 3 \cdot ln(ln(t))]}{u_a^t}$$

The precision taken is 0.001

1.1.4 Thompson Sampling

We maintain two values s_i & f_i for each arm i and pull random samples from a beta distribution written as Beta($\alpha = s_i + 1$, $\beta = f_i + 1$) and pull the arm for which we received the highest sample. The same is implemented.

Regret Calculation (for all of the above)

Let P^* be the mean of the arm with the highest mean (optimal arm). Then for T pulls, Regret R_T is given as:

$$R_T = P^* \times T - \text{sum of cumulative rewards from each arm}$$

1.2 Plots & Observations

1.2.1 Instance 1

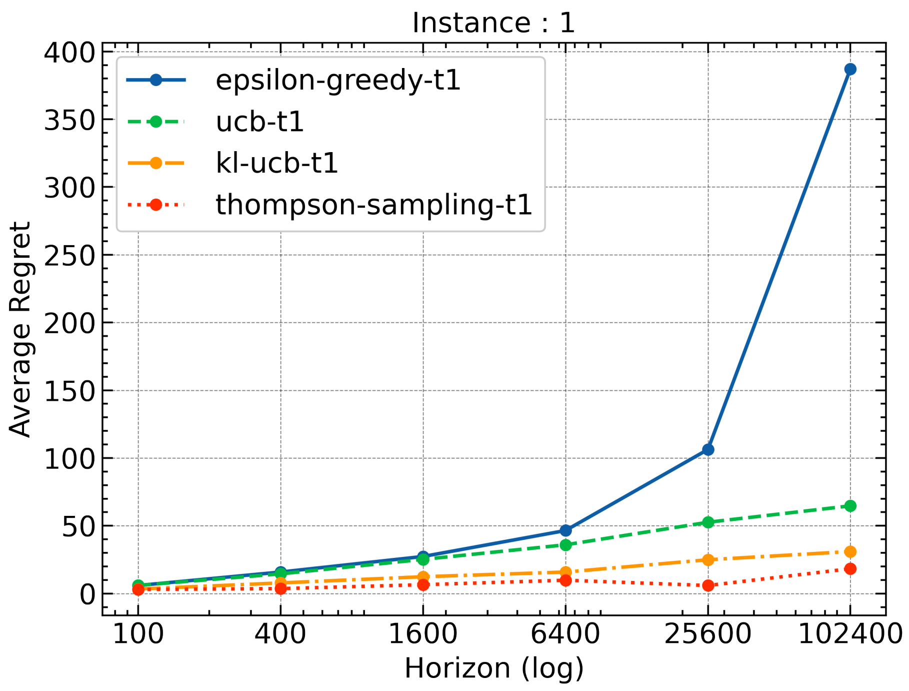

Figure 1: Task 1 Instance 1

1.2.2 Instance 2

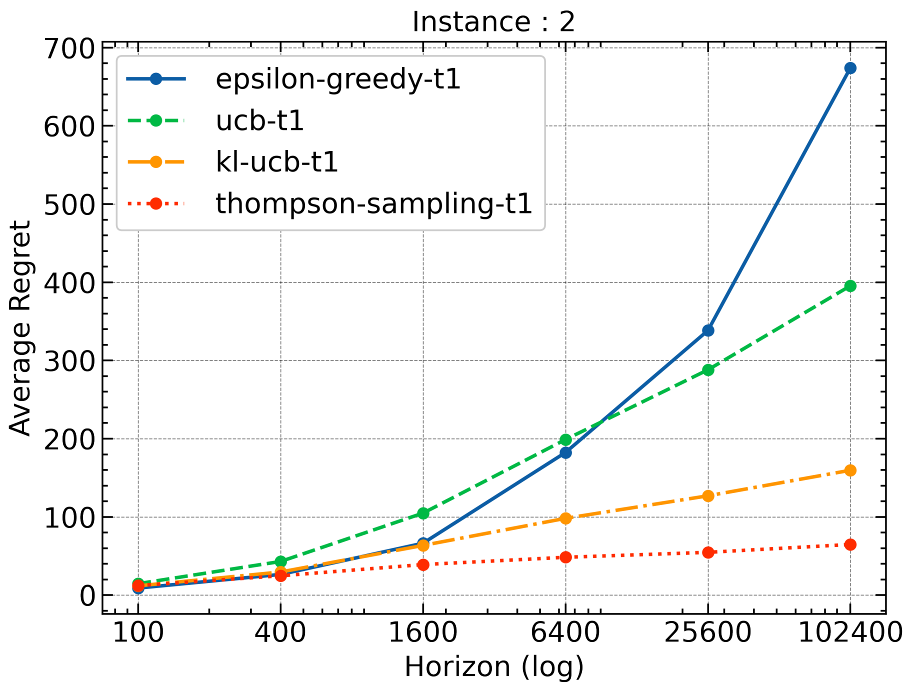

Figure 2: Task 1 Instance 2

1.2.3 Instance 3

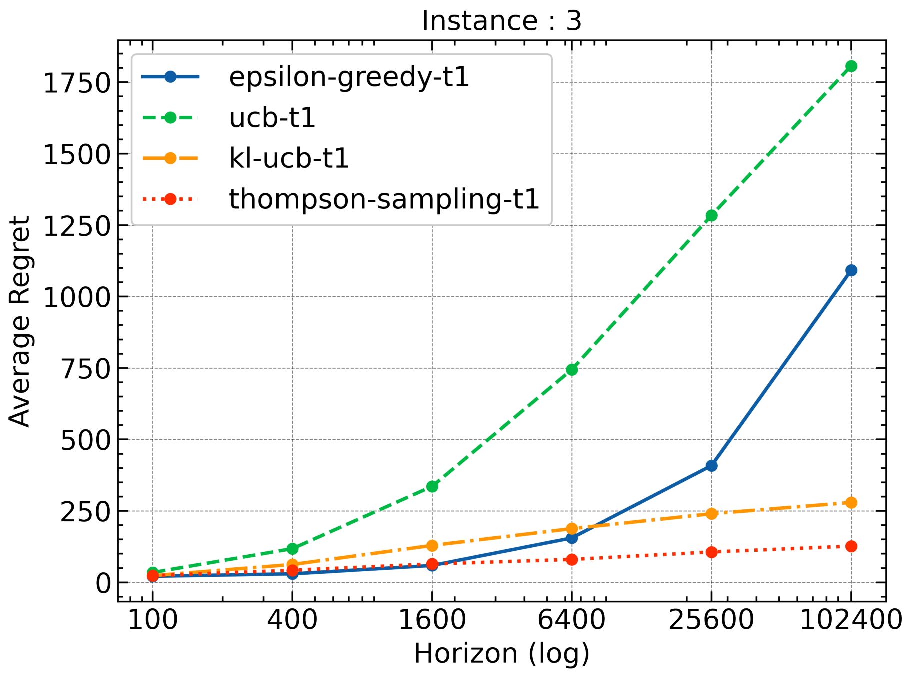

Figure 3: Task 1 Instance 3

Observations

1. For Instance 1:
    - Regret-wise : [epsilon greedy > UCB > KL-UCB > Thompson-sampling]
    - In log scale : [epsilon greedy : non-linear ; UCB : Linear ; KL-UCB : Linear ; Thompson-sampling : Linear]
2. For Instance 2:
    - Regret-wise : [epsilon greedy > UCB > KL-UCB > Thompson-sampling]
    - In log scale : [epsilon greedy : non-linear ; UCB : Linear ; KL-UCB : Linear ; Thompson-sampling : Linear]
    - Epsilon Greedy approach was better in short term than UCB in terms of Regret but later became worse. The initial higher regret than Epsilon Greedy approach for UCB Algorithm can be attributed to its Exploration phase
3. For Instance 3:
    - Regret-wise : [UCB > Epsilon Greedy > KL-UCB > Thompson-sampling]
    - In log scale : [epsilon greedy : non-linear ; UCB : Linear ; KL-UCB : Linear ; Thompson-sampling : Linear]
    - UCB is higher in regret than Epsilon Greedy though it can be extrapolated that it would do better than the latter for an even larger horizon due to the fact that UCB has a linear increase while Epsilon Greedy has a non-linear/polynomial increase in regret over time in log scale.

2 Task 2

2.1 Implementation Details

Same as that of "ucb-t1" except that **c** is provided as a parameter

2.2 Plot, Results & Observations

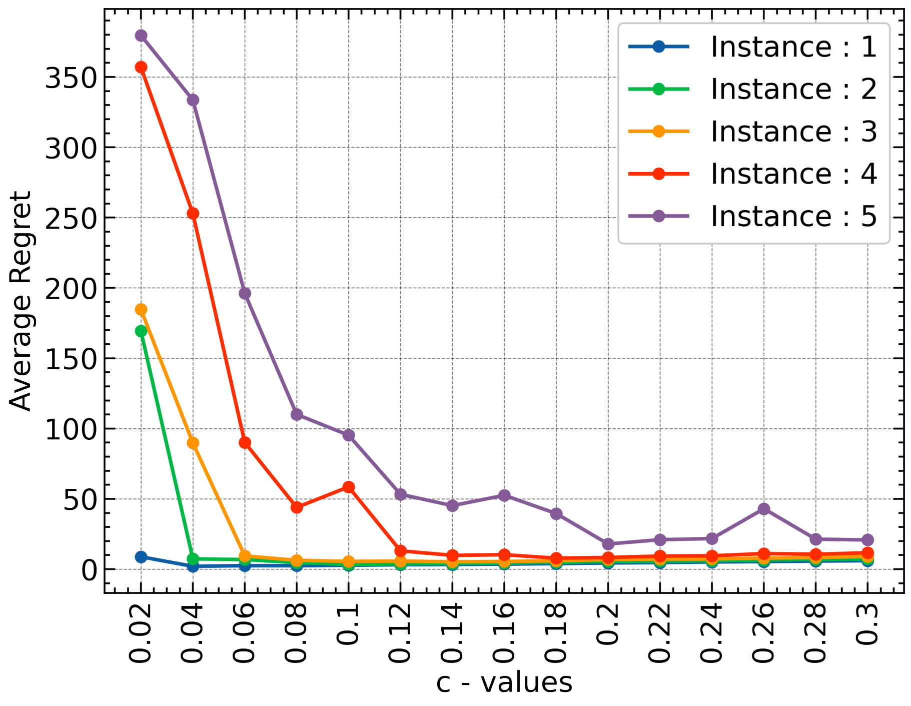

Figure 4: Task 2

Instance	"Optimised" c
1	0.04
2	0.10
3	0.14
4	0.18
5	0.20

Explanation

We see that optimal *c* values increase from Instance 1 to Instance 5. This can be attributed to the mean values of the instances. Each instance in Task 2 has 2 arms with different means. The trend is that the difference between the means decreases from Instance 1 to Instance 5. This means that Instance 1 has a clear distinction between the best arm and the worst arm (where the means are 0.7 (BEST) & 0.2 (WORST)) while it is not the case for Instance 5 (where the means are 0.7 & 0.6 (not a huge difference)). Therefore, to find the optimal arm, larger *c* values are better for exploration where the difference in means of arms is not significant.

3 Task 3

3.1 Implementation Details

Sampling Algorithm

The implementation is based on generalizing the Thompson Sampling method on Bernoulli Bandit instances to instances having rewards bounded in $[0,1]$. In the former case, we utilized a Beta Distribution while in this case, I have used a **Dirichlet Distribution**. The algorithm was referred from the following paper : *"Bandit Algorithms Based on Thompson Sampling for Bounded Reward Distributions" by Charles Riou and Junya Honda* (Ref : 1).The algorithm is given as follows:

Algorithm :=

For given Horizon $T \geq 1$, Number of arms $K \geq 1$, support size of arm distributions $M \geq 1$; we set $\alpha_m^k := 1$ for $k \in [K]$ and $m \in \{0, ...M\}$

```
    for $t = 1, 2, 3, ..., T$ :
        - for $k = 1...K$ :
            - - Sample $L_k \sim Dir(\alpha_0^k, \alpha_1^k, \alpha_2^k, ...., \alpha_M^k)$
        - $I(t) = argmax_{k \in \{1,...,K\}} \left\{ \left(0, \frac{1}{M}, \frac{2}{M}, ...1\right)^T L_k \right\}$
        - Pull arm $I(t)$ and observe reward $r_t = \frac{m}{M}$
        - Update $\alpha_m^{I(t)} := \alpha_m^{I(t)} + 1$
```

Regret Calculation

Let the reward vector be $\mathbf{R} = [r_1 \; r_2 \; ... \; r_M]_{1 \times M}$ for M total rewards bounded in $[0,1]$. Let Matrix $\mathbf{P} = [p_{nm}]_{N \times M}$ for N total arms. Here, p_{nm} refers to probability of getting reward r_m when arm $n \in \{1, 2...N\}$ is pulled. The highest expected reward on a single pull of the best arm is as follows :

$$r_{max} = \max_{1 \leq n \leq N} \sum_{m=1}^{M} r_m p_{nm}$$

Correspondingly, maximum expected cumulative reward $r_{max}^{cumulative}$ for Horizon T is:

$$r_{max}^{cumulative} = r_{max} \times T$$

and regret R_T is :

$$R_T = r_{max}^{cumulative} - (\text{sum of rewards obtained for T pulls})$$

3.2 Plots & Observations

3.2.1 Instance 1

Observation : The Average Regret over time on a log scale is found to be almost linear

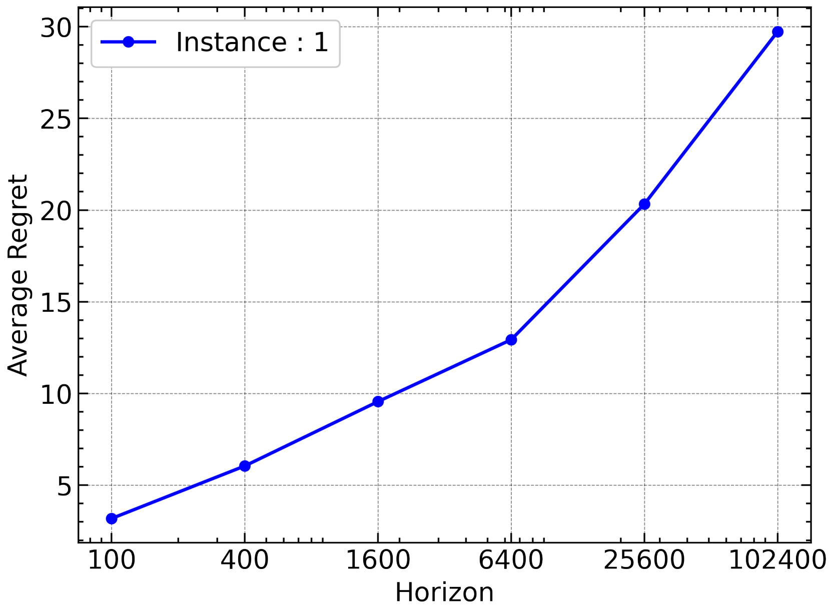

Figure 5: Task 3 Instance 1

3.2.2 Instance 2

Observation : The Average Regret over time on a log scale is found to be almost linear. We also see that average regret drops at higher horizons. This is likely because of the rewards being received from the optimal arm are higher than the mean reward of the same arm.

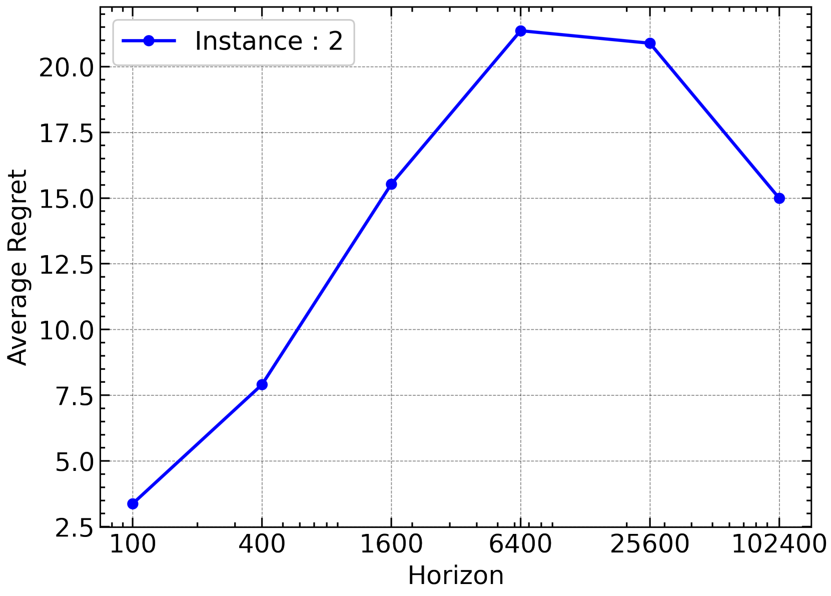

Figure 6: Task 3 Instance 2

4 Task 4

4.1 Implementation Details

Algorithm

We redefine success s and failure f depending on the threshold value t and the reward r obtained on pulling an arm as follows :

- success (HIGH) **if** *reward* $>$ *threshold*
- failure (LOW) **if** *reward* $\leq$ *threshold*

This formalism lends itself to solution by the standard Thompson Sampling Algorithm on Bernoulli Bandits. Consequently, we maintain variables S (Successes) and F (Failures) for each arm of the bandit instance and take random samples from a Beta distribution with parameters $\alpha = (S_i + 1)$ $\beta = (F_i + 1)$ for $i \in \{1, 2, ...N\}$ where N is number of arms.

Regret Calculation

$$\text{HIGHS REGRET} = \text{MAX EXPECTED HIGHS} - \text{HIGHS}$$

$$\text{HIGHS} = \sum_{i=1}^{N} S_i \text{ (Sum of successes of all arms)}$$

To calculate MAX HIGHS, we first obtain the probability that the reward obtained from an arm is HIGH which is equivalent to the sum of probabilities of obtaining rewards greater than the threshold. Let p_{nm} denote probability of receiving a reward m bounded in $[0, 1]$ when arm n is pulled. Then, probability of success associated with an arm is:

$$P_n = \sum_{m} p_{mn} \; s.t. \; m > t$$

$$P^* = \max_{1 \leq n \leq N}(P_n)$$

$$\text{MAX EXPECTED HIGHS} = P^* \times T$$

4.2 Plots & Observations

4.2.1 Instance 1

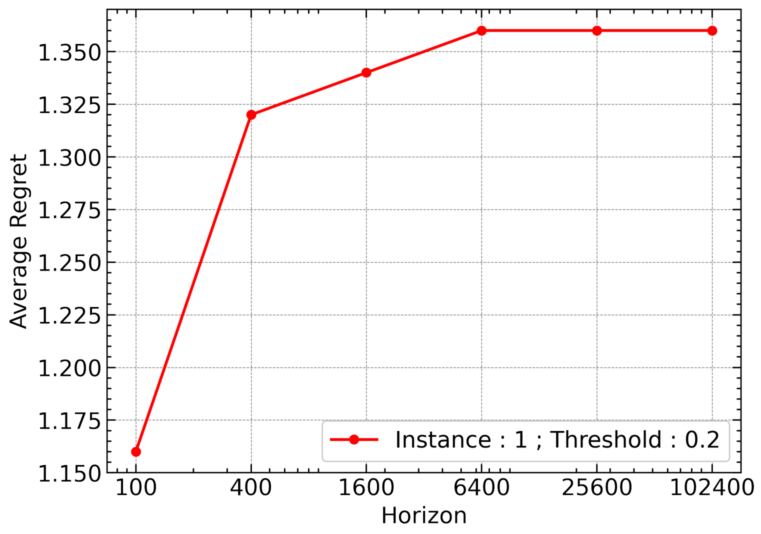

Figure 7: Task 4 Instance 1 Threshold 0.2

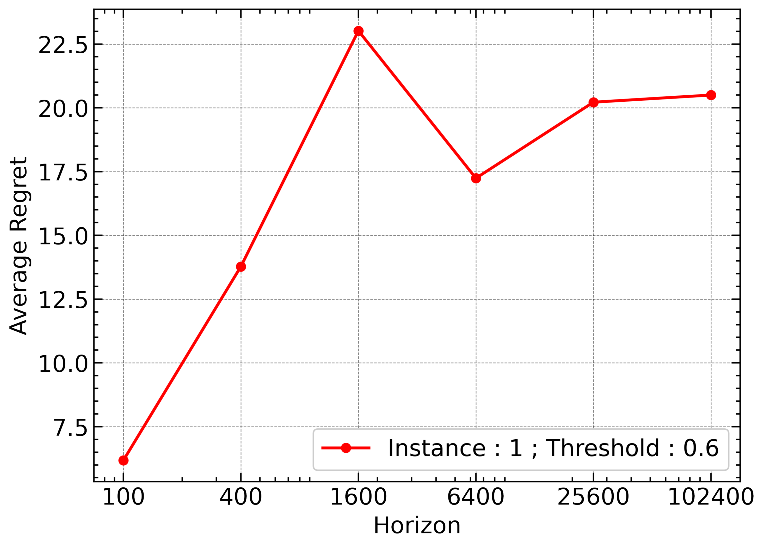

Figure 8: Task 4 Instance 1 Threshold 0.6

Observations

The regrets seem to saturate at higher horizons. This can be attributed to the fact that once the optimal arm is singled out, the success is determined only by its probability of giving a reward greater than the threshold. In the first case, where threshold = 0.2, since Instance 1 has 0 probability of giving a 0 reward & all other rewards are greater than the threshold, we see that the regret stagnates.

4.2.2 Instance 2

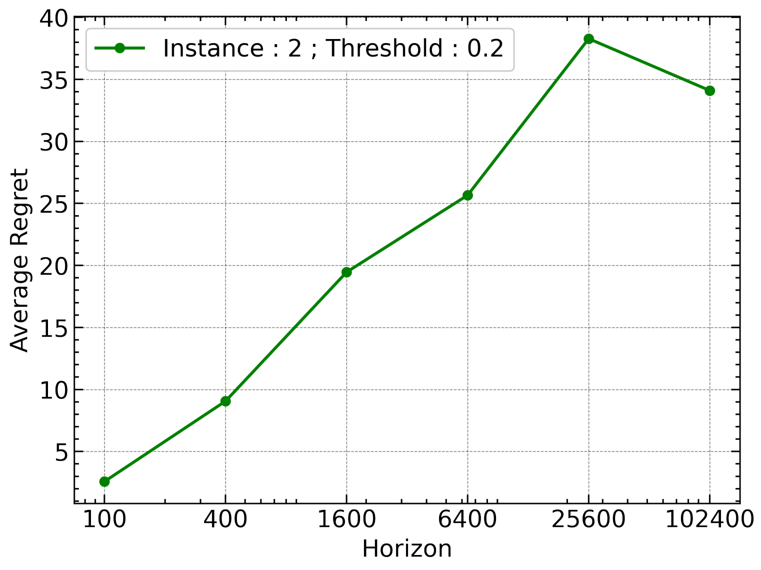

Figure 9: Task 4 Instance 2 Threshold 0.2

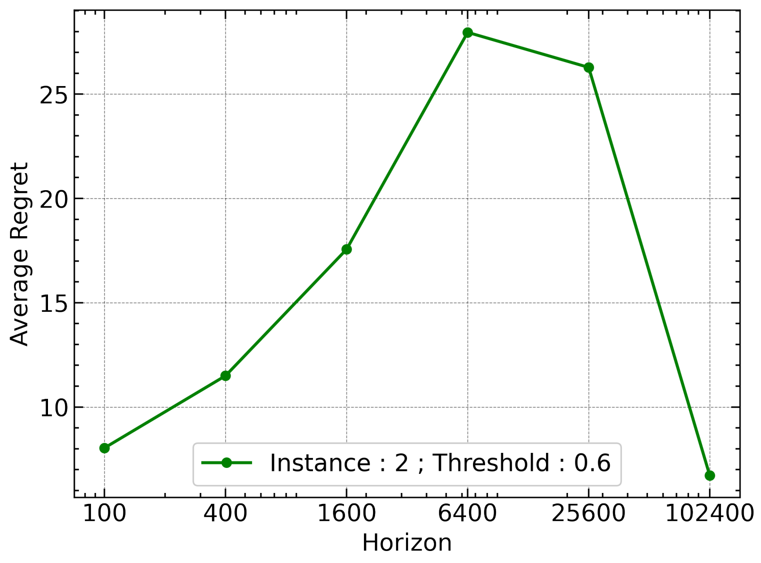

Figure 10: Task 4 Instance 2 Threshold 0.6

Observations

The regrets seem to increase linearly at first (log scale) and then achieve a plateau. This can correspond to the algorithm narrowing down to the optimal arm. The fluctuations later on can be attributed to the probability of the optimal arm producing HIGH rewards.

References

1. **Bandit Algorithms Based on Thompson Sampling for Bounded Reward Distributions** by Charles Riou & Junya Honda [Link]

2. argparse module
https://docs.python.org/3.8/library/argparse.html#module-argparse

3. numpy.random module
https://numpy.org/doc/stable/reference/random/index.html

4. KL Divergence Definition
https://math.stackexchange.com/questions/2604566/kl-divergence-between-two-multivariate-bernoulli-distribution

List of Figures