CS-747

Foundations of Intelligent and Learning Agents

Adarsh Raj - 190050004

Assignment-1

September 2021

Task - 1

Implementation of Sampling Algorithms:

Sampling Function: Function `rand_samp(p)`, return 1 if random value is less than p, else return 0.

1. **Epsilon Greedy:** For each iteration in horizon, random float was sampled, if less than epsilon, explore (i.e choose arm randomly), else exploit (choose arm with maximum empirical mean).

   Assumption: For tie breaking between different arms with same maximum empirical mean, **first arm** was taken from the instance with max empirical mean, using `np.argmax`.

2. **UCB Algorithm:** First, each arm was chosen once, and corresponding reward was stored in numpy array. This was done to make ucb_a^t calculable. Now, for each iteration of remaining choices i.e (horizon - number of arms), ucb for all arms was calculated and arm was chosen with maximum ucb value.

   Assumption: For tie breaking, **first arm** with maximum ucb value was taken, using `np.argmax`.

3. **KL-UCB Algorithm:** Implementation was done in same function of UCB Algorithm, except the calculation of UCB. To call kl-ucb algorithm, one needs to set the flag argument to "1". **Binary search** was used to find q for $ucb-kl_a^t$ calculation.

   Assumption: For tie breaking, **first arm** with maximum ucb-kl value was taken. Also, in binary search, **error precision limit of 0.001** was set to find the value of q. This is because there is a possibility that the binary search can take a large amount of time if we can't find the exact mid value that is equal to KL value (mainly because of precision errors).

4. **Thompson Sampling Algorithm:** Success and Failure numpy arrays are maintained to store successes and failures for each arm. For each iteration, random samples for each arm were sampled from $Beta\ (succ + 1, fail + 1)$, and the arm with maximum sampled value is chosen.

   Assumption: For tie breaking, **first arm** with maximum sampled value was chosen, using `np.argmax`.

Note: Step-by-step implementation details is well commented in the code file `bandit.py` itself.

Plots were generated for each instance, with regret values as mean of all regret values across random seeds from 0 to 49.

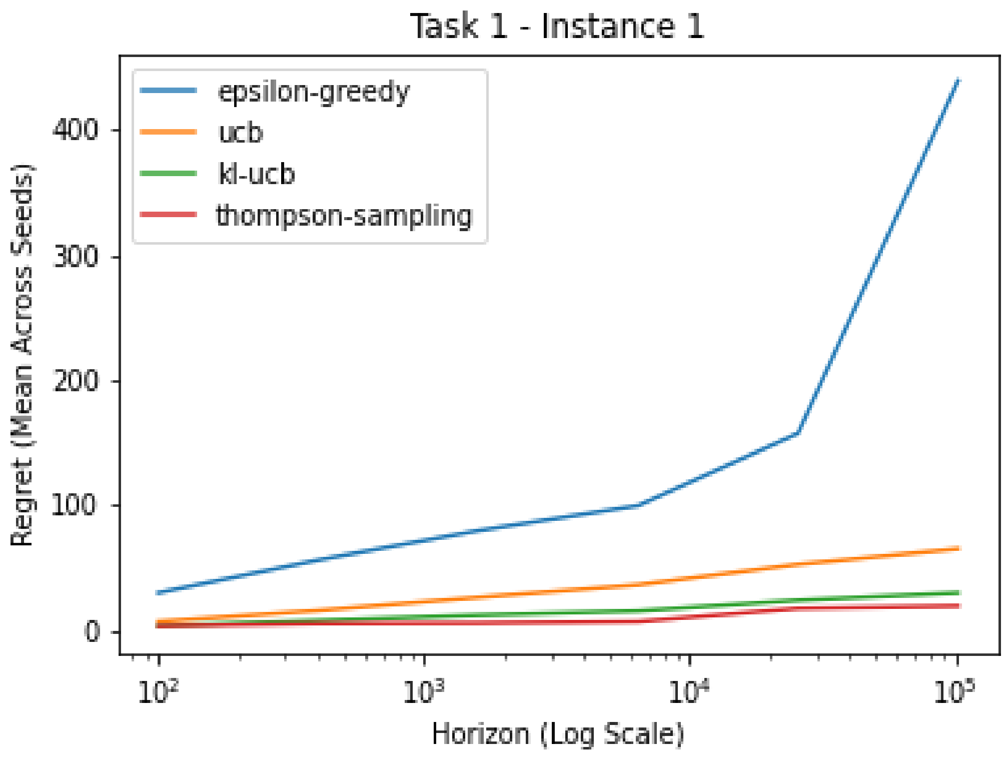

Figure 1: Regret vs Horizon (for different algorithms) for Instance 1

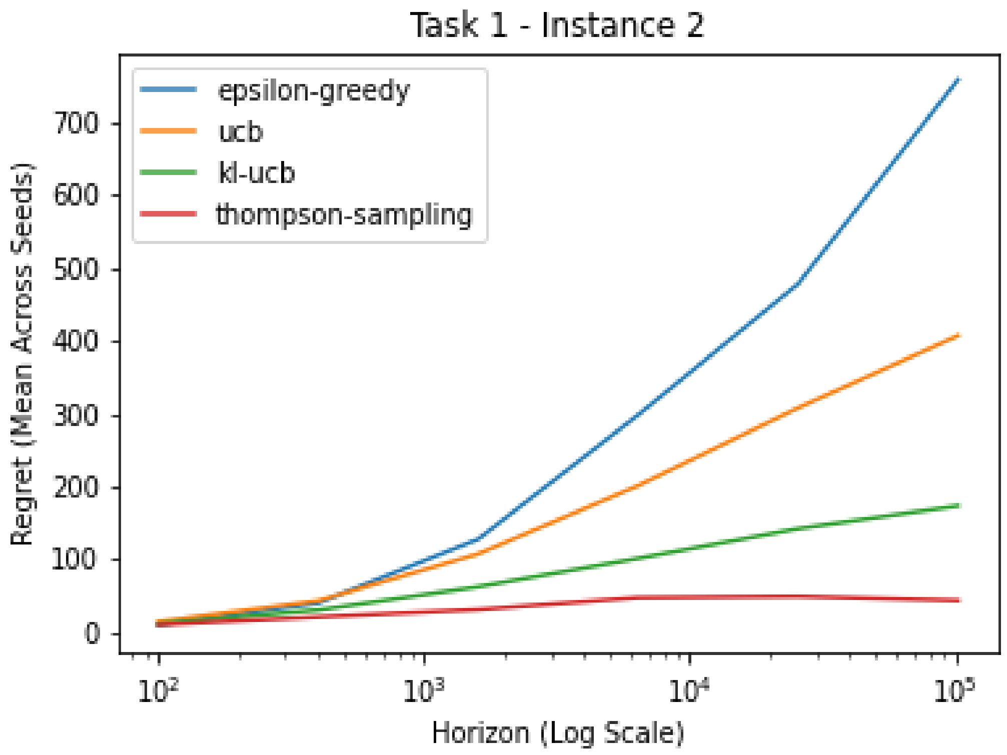

Figure 2: Regret vs Horizon (for different algorithms) for Instance 2

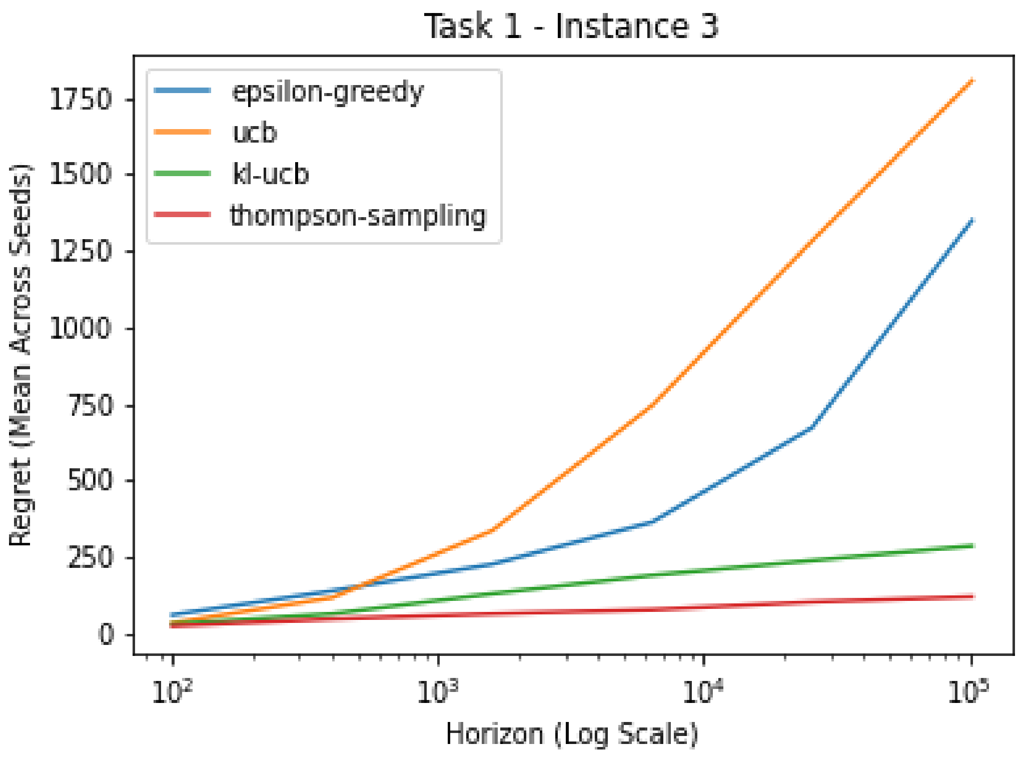

Figure 3: Regret vs Horizon (for different algorithms) for Instance 3

As we can observe from the plots that `kl-ucb` and `thompson-sampling` algorithms give less regret compared to other algorithms (thompson being superior than kl-ucb). This was expected because thompson sampling achieves optimal lower bound and kl-ucb also achieves it asymptotically.

For smaller horizons (< 1000) and small number of arms (for example: instance 1), `epsilon-greedy` and `ucb` algorithms perform quite well. But for higher horizon values and large number of arms (instance 2 and instance 3), we obtain much higher regret values compared to `kl-ucb` and `thompson` algorithms.

Task - 2

UCB-T2

Implementation Details:

To implement the functionality of `scale` parameter, same function of `ucb-t1` algorithm was used with additional flag argument and scale argument. To call `ucb-t2`, one needs to set the flag parameter as "2" in the function `ucb(instance, horizon, flag, scale)` and also provide the given scale. Additional details are well commented in the code file `bandit.py` itself.

Based, on the data generated for task 2, for each instance and each scale, regret values were calculated (average across all seeds). The scale parameter which gave lowest regret values for each instance are as follows:

Instance	Scale	Regret Value (Avg. across seeds)
1	0.08	2.74
2	0.1	3.8
3	0.08	4.98
4	0.16	8.88
5	0.2	17.92

Plot:

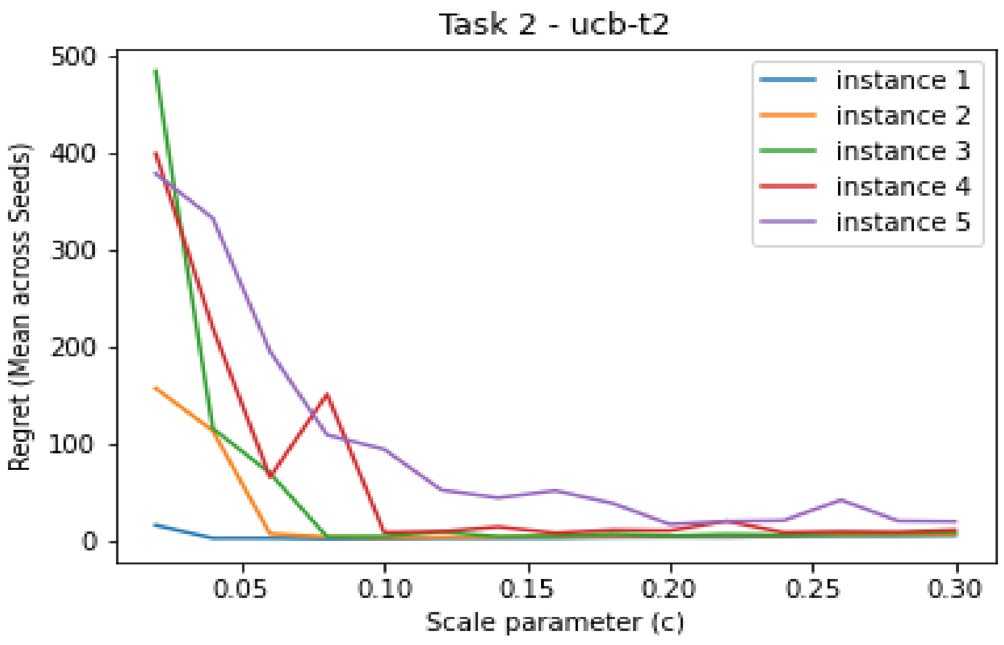

Figure 4: Regret vs Horizon (For different instances) for Task 2

Interpretation:

It is observed that on increasing the scale parameter, regret values become lower and lower. Based on the plot, at scale = 0.3 all instances have similar regret values. It can be concluded that if we increase scale parameter further, then, the regret values will come more closer for all instances. However, it can be observed that for small scale parameters, regret (average across all seeds) is very high.

For instances it can be observed from the plot, that the regret given by the algorithm is lower for Instance i than Instance j, for $(i < j)$. This can be due to the fact that number of arms of bandit is increasing with the instance number.

Task - 3

Algorithm used: **Modified Thompson Sampling**

For this task, since, the rewards were possible from a finite support and were not bernoulli, new definitions for success and failure for each arm in Thompson algorithm were taken and implemented.

For a sampled arm a, $succ_a$ is incremented by reward obtained and $fail_a$ is incremented by (1- reward obtained), after choosing maximum from beta random samples (**Assumption:** Tie breaking - First Arm with Maximum was chosen). Note that for bernoulli, we will have $succ_a += 1$, if reward is 1, else $fail_a += 1$. This is also covered by above definition.

For example in modified thompson algorithm, support = [0, 0.25, 0.5, 0.75] and probability distribution for an arm a is p = [0.3, 0.2, 0.3, 0.2]. Let's say, if on weighted sampling from support using `np.random.choice(support, p)`, we get 0.75. Then, we update $succ_a = succ_a + 0.75$ and $fail_a += fail_a + (1 - 0.75) = fail_a + 0.25$. And in next iteration we sample from Beta(succ, fail) for all arms, similar to Thompson Algorithm.

Plots:

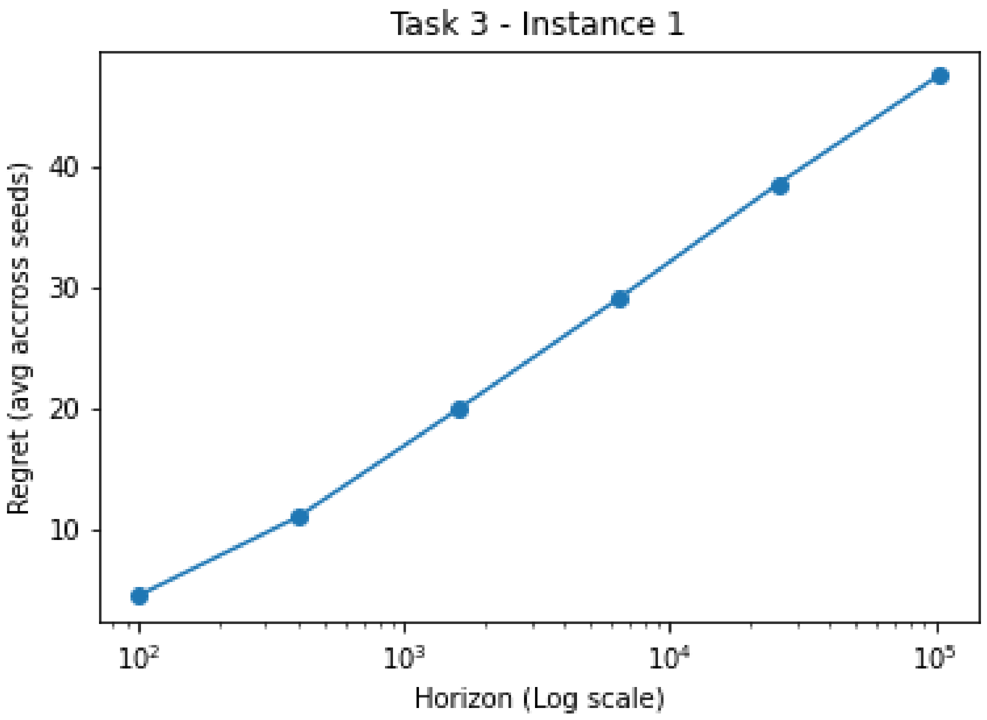

Figure 5: Regret vs Horizon (Instance 1)

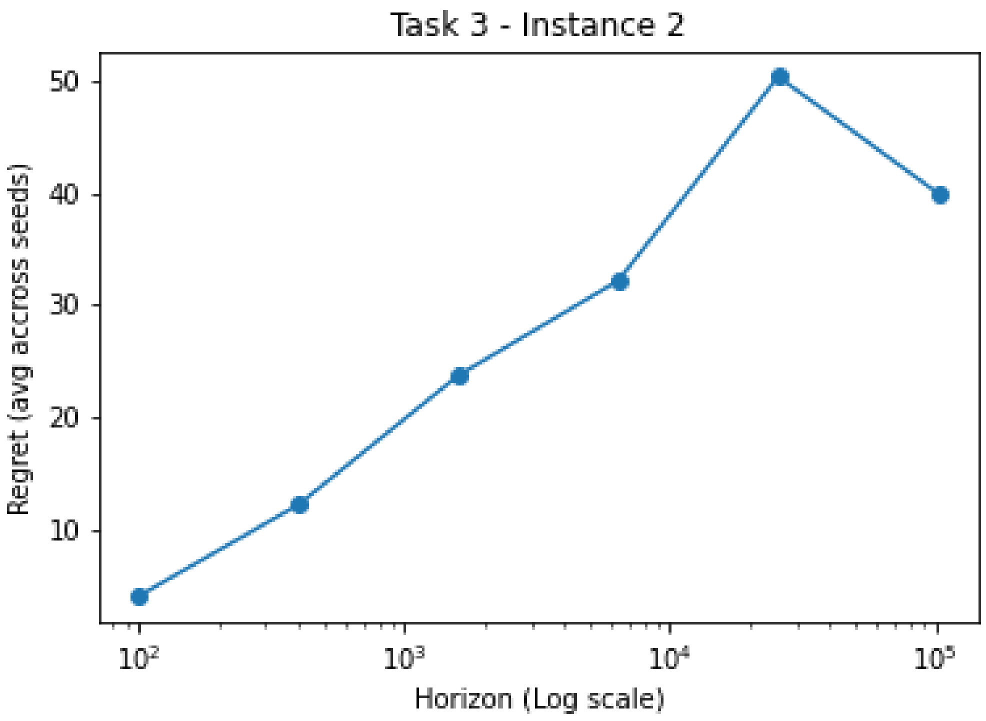

Figure 6: Regret vs Horizon (Instance 2)

Algorithm Justification and Interpretation:

I chose Thompson Algorithm and modified the definitions of success and failures due to the fact that Thompson Algorithm **achieves optimal regret** for Bernoulli distributed arms. Since, for the given instances of task 3, on success of an arm, we are not getting "1" as an addition to our cumulative reward, instead we are getting a float value in [0, 1] from the support stated in instance. Hence, instead of defining success and fail having a boolean behaviour, we can define success and fail as event of "the amount of reward gained and the amount of reward lost" respectively.

As for Bernoulli, total reward that can be obtained for a single pull is 1 (neglecting the case of 0 probability for success), the reward gained for modified definition would be the value from the support that was sampled using weights given and reward lost will be the value (1-obtained). After this, sampling for maximum beta random value was done and then that arm was pulled similar to Thompson.

For multiple range of instances given, the algorithm works quite well. The regret values are approximately linear with respect to log(horizon), which can be confirmed from the plot.

Task - 4

Algorithm used: **Modified Thompson Sampling**

For this task, since, the rewards were possible from a finite support and were not bernoulli, but threshold was given, which can lead to two choices, either the choice chosen is less than threshold or more than it. This behaviour was used to model this algorithm similar to Thompson Sampling.

For a sampled arm a, $succ_a$ is incremented by 1 and HIGH was incremented by 1, if chosen reward is greater than threshold (HIGH), else $fail_a$ is incremented by 1 (LOW). Note that, this makes the algorithm exact replica of thompson algorithm. After this we followed same steps as of thompson sampling.

Assumption: Tie breaking - First Arm with Maximum Beta Random value was chosen.

Plots:

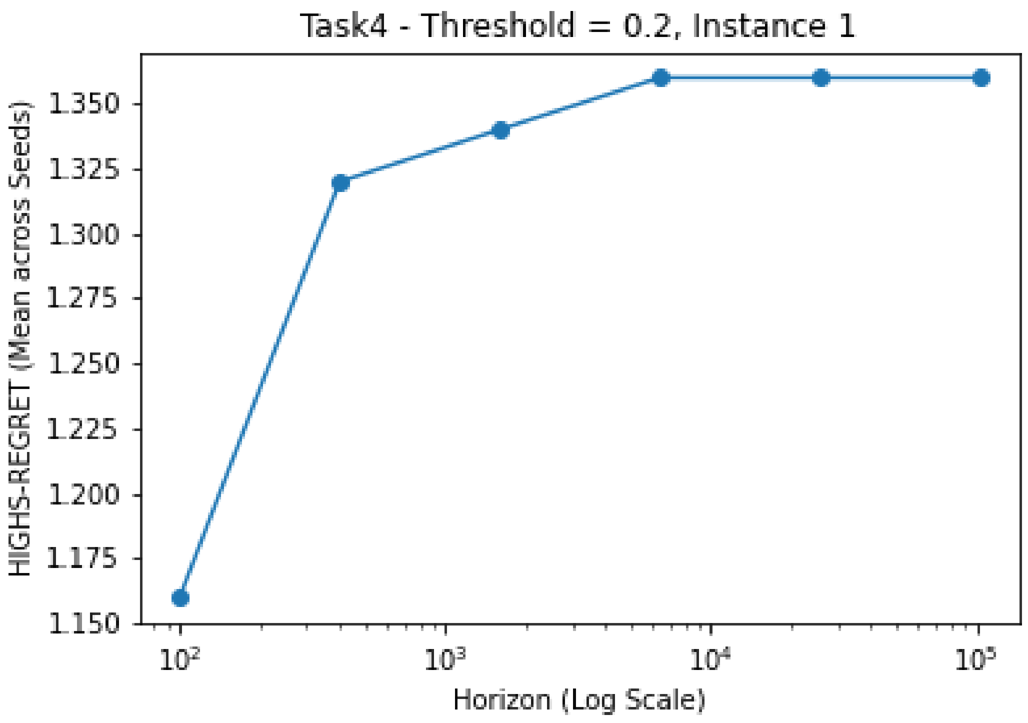

Figure 7: Highs-Regret vs Horizon (Threshold = 0.2. Instance 1)

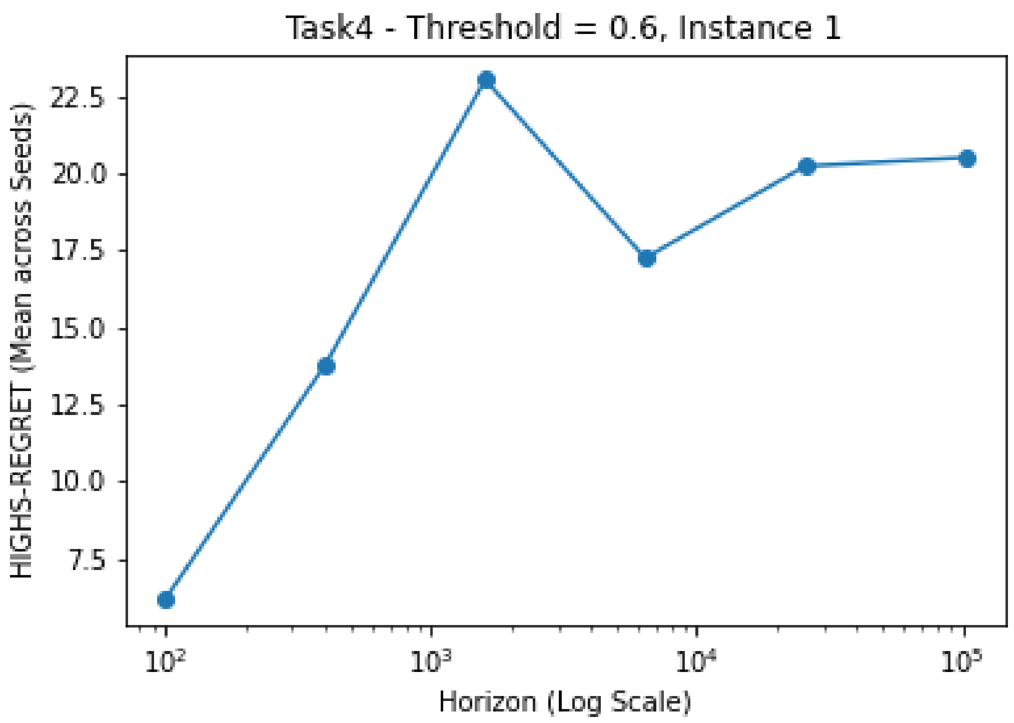

Figure 8: Highs-Regret vs Horizon (Threshold = 0.6, Instance 1)

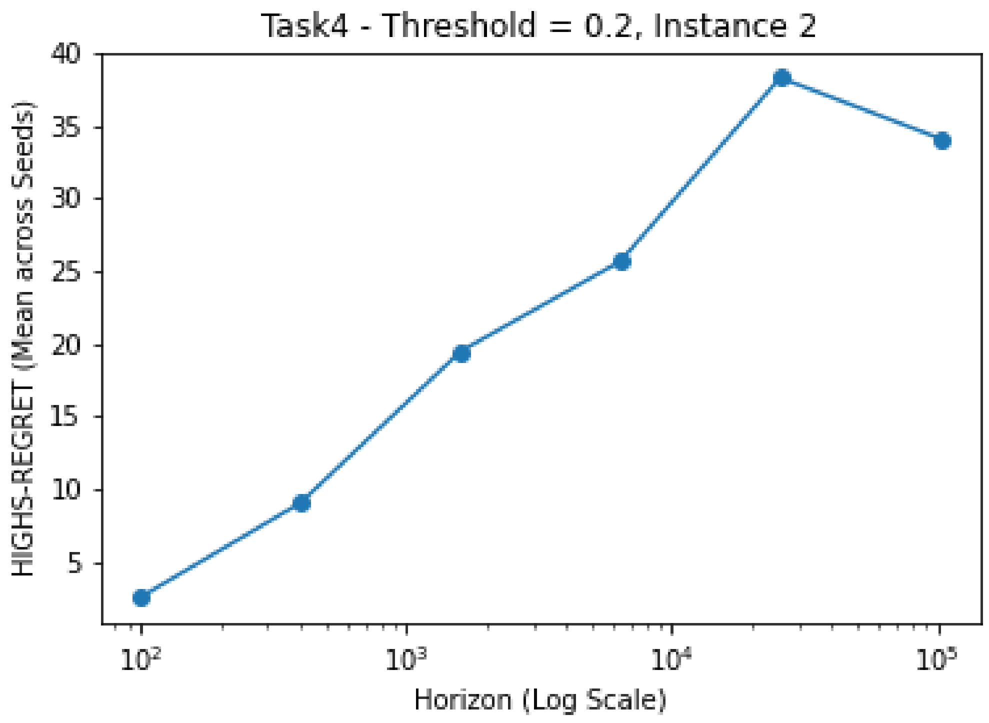

Figure 9: Highs-Regret vs Horizon (Threshold = 0.2. Instance 2)

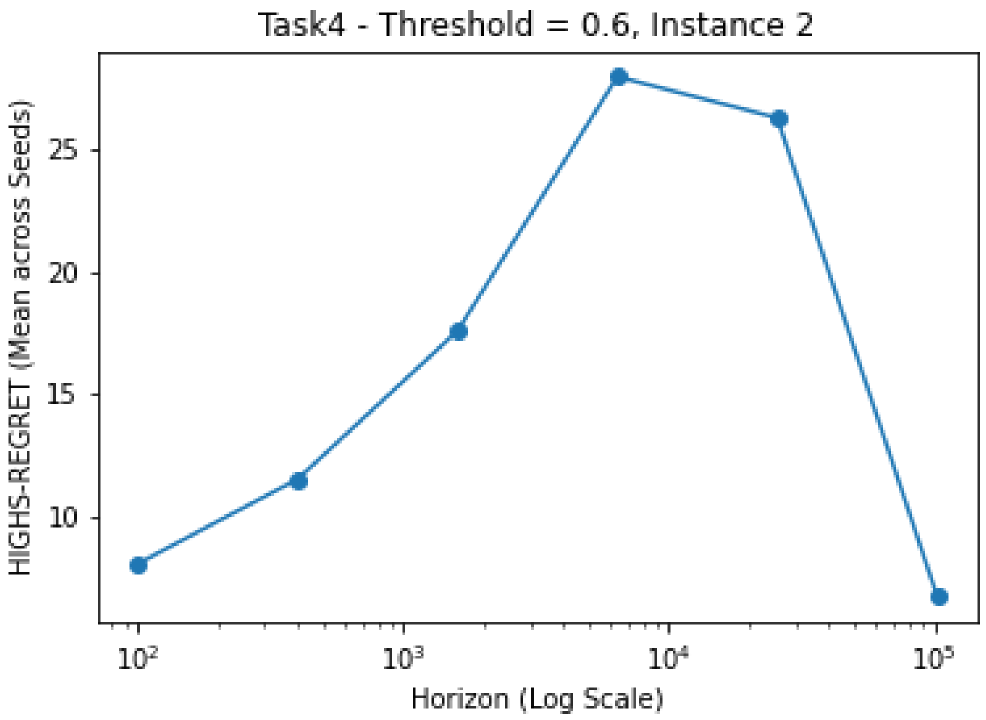

Figure 10: Highs-Regret vs Horizon (Threshold = 0.6, Instance 2)

Interpretation:
For threshold = 0.2, the HIGHS-REGRET values are approximately linear with respect to log(Horizon). Also, for Instance 1, the values (mean across seeds) are around 1. For instance 2, the values range between (0,40).

Similarly, for threshold = 0.6, linear behaviour can be observed with slight variations and the values of HIGHS-REGRET are in between (0, 30) for both Instance 1 and 2.