Programming Assignment 1: CS 747

Md Kamran

Due: 9 September 2021

Contents

Task 1

For all the 4 algorithms, ie. epsilon-greedy, UCB, KL-UCB, and thompson-sampling, in the initialization step, I pulled all the arms once one-by-one so that the number of pulls of each arm becomes equal to 1. This is to ensure that the term for $\#pulls$ that comes in the denominator is non-zero.

Again for all the 4 algorithms listed above, the breaking of tie is done by sampling one of the arms uniformly from all the arms which are a part of the tie breaker. Like for epsilon-greedy, if two arms have same highest empirical mean, then one of the arms is sampled uniformly at random from those having the highest empirical mean. Similarly for UCB, KL-UCB, thompson-sampling, if two arms have same highest ucb-value, kl-ucb-value, beta-sample-value respectively, then one of the arms is sampled uniformly at random from those which have the highest respective values.

The same initialization step and tie breaker step is also included in the tasks 2, 3, and 4.

Setting of algorithm specific parameters is done as taught in class. For epsilon-greedy, a random number e is generated and compared with the ϵ value. If $e < \epsilon$, then arms are sampled uniformly, else, arm with the highest empirical mean is chosen and the mean value for the chosen arm is updates. In UCB and KL-UCB, the ucb-values and kl-ucb-values is updated for each arm after an arm is chosen and the reward is generated. In thompson-sampling, the samples from beta distribution are generated after every time an arm is chosen and the reward is generated.

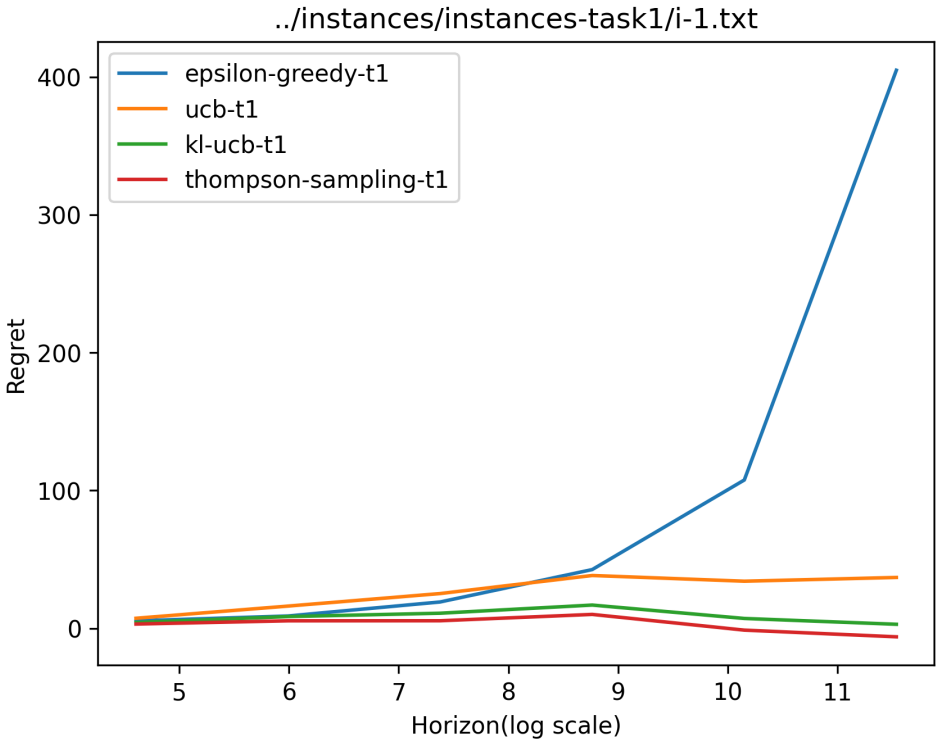

Figure 1: Task-1 Instance-1

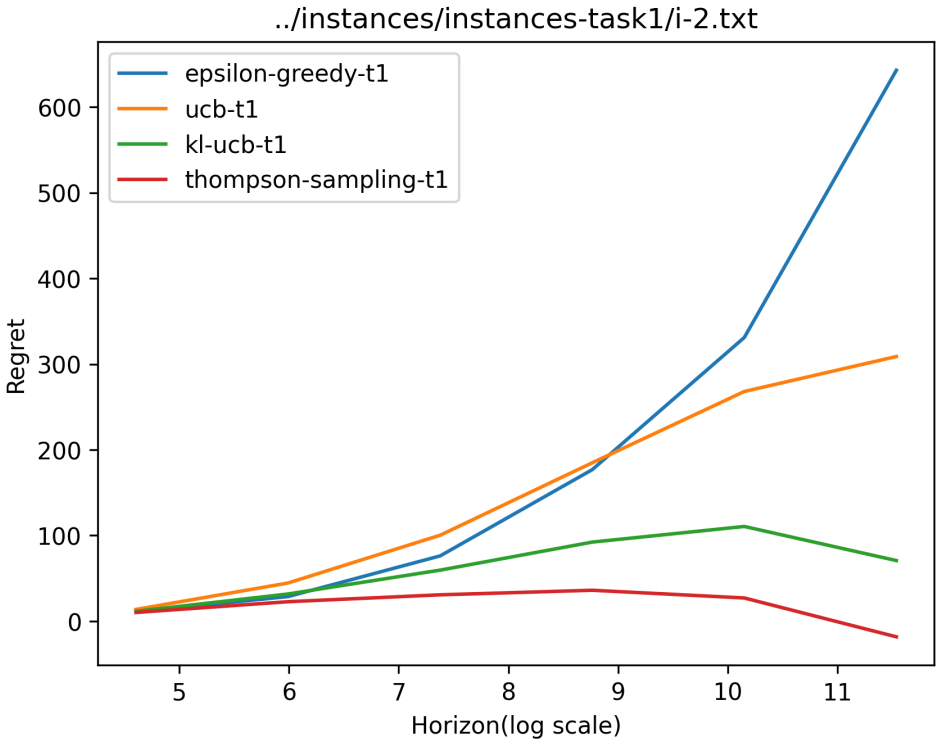

Figure 2: Task-1 Instance-2

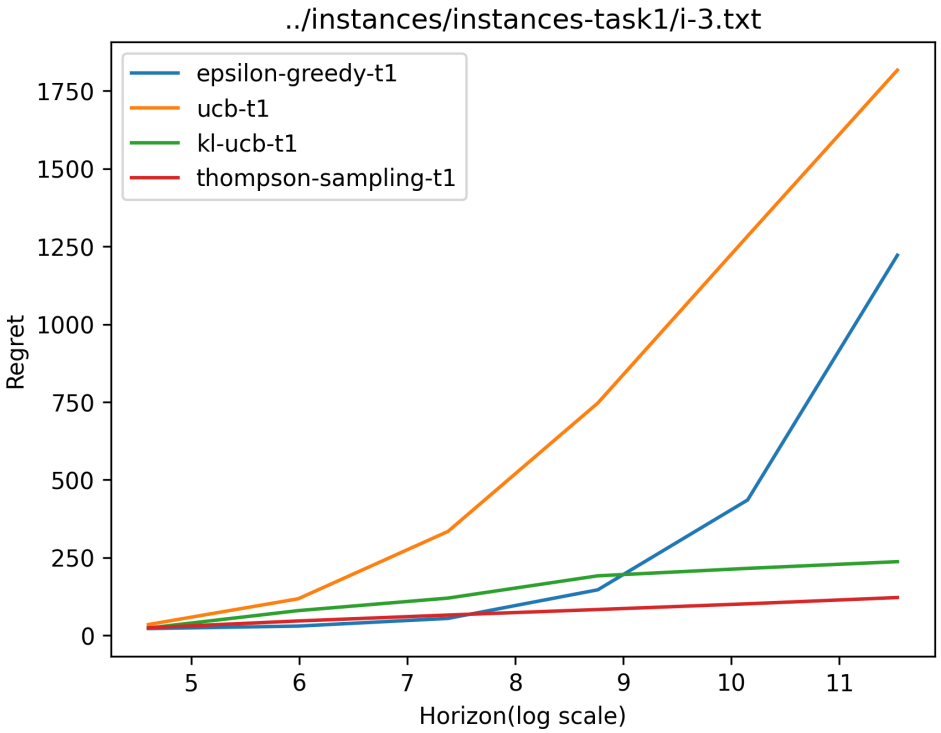

Figure 3: Task-1 Instance-3

Observations for task 1:-

(i) For instance 1 with 2 arms, epsilon greedy performs very poorly while the other 3 are a lot better with UCB coming at third, KL-UCB at second and thompson sampling coming at first in terms of ranking based on regret.
(ii) For instance 2 with 5 arms, epsilon greedy again performs very poorly although UCB also gives poor results while the other 2 are still a lot better with KL-UCB coming at second and thompson sampling at first in terms of ranking based on regret.
(iii) For instance 3 with 25 arms, UCB is observed to perform the worst although epsilon greedy also gives poor results and is closely worse enough while the other 2 are still observed to give good enough results with again KL-UCB coming at second and thompson sampling at first in terms of ranking based on regret.
(iv) In instance 1 and 2, thompson sampling is performs so well that even for larger horizon values, it ends up giving negative regret values.

Task 2

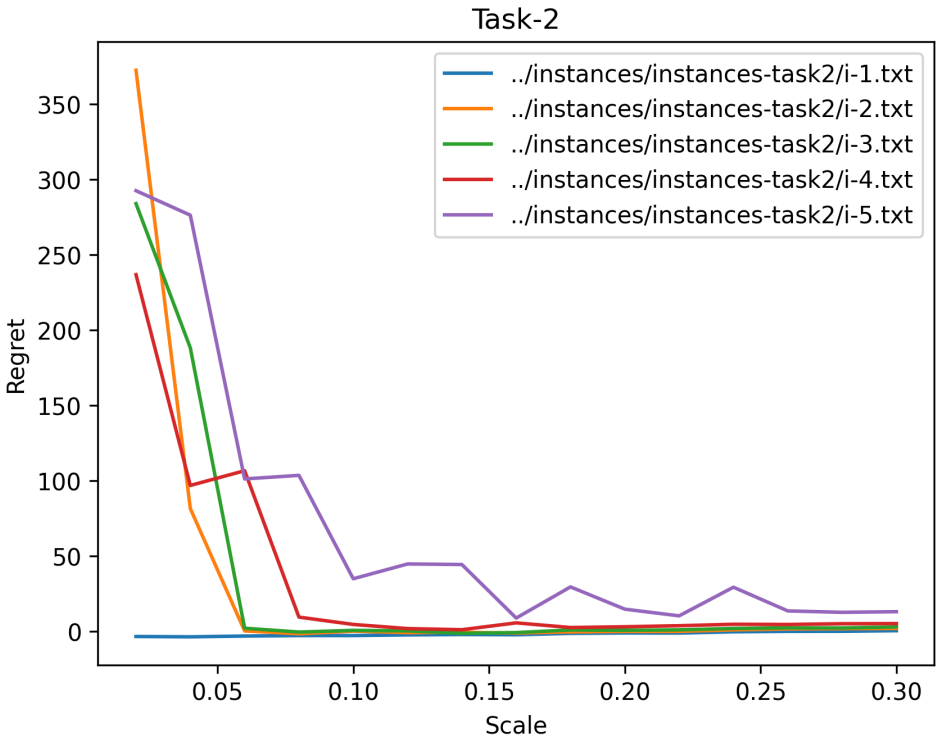

Figure 4: Task-2

(i) The value of c that gives the lowest regret for each of the 5 instances:
Instance-1: c=0.04
Instance-2: c=0.08
Instance-3: c=0.14
Instance-4: c=0.14
Instance-5: c=0.16

(ii) The trend that I observe across the instances from 1 to 5 is that the c values are in a non decreasing order. This is because when we look at the instance file, the difference of probabilities of the two arms are in decreasing order of 0.5, 0.4, 0.3, 0.2, 0.1 from instances 1 to 5.
So if the difference is already large as in instance 1, then we don't need to explore a lot as the empirical mean term can easily differentiate between the more optimal arm. But if the difference is small as in instance 5, then we will need more exploration to carefully determine which one of the two is the optimal arm. And since the exploration factor is controlled by c, therefore for larger difference in probabilities of arms, c value will be smaller and vice versa.

Task 3

I used thompson sampling for task 3. Since the main difference in task 3 from task 1 is generation of rewards from a distribution with a finite support, therefore the method for sampling of arms is similar to task 1. The approach used by me can be described by the following example.

Suppose the support is 0.0, 0.25, 0.5, 0.75, 1.0 and the probabilities are 0.2, 0.2, 0.2, 0.2, 0.2. Then firstly, a random number r is generated and compared with the cumulative sum of the probabilities to generate the reward. The cumulative sum for this case will be 0.2, 0.4, 0.6, 0.8, 1.0. So if $r < 0.2$, then the reward is 0.0, else if $r >= 0.2$ but $r < 0.4$, then the reward is 0.25, else if $r >= 0.4$ but $r < 0.6$, then the reward is 0.5, else if $r >= 0.6$ but $r < 0.8$, then the reward is 0.75, else if $r >= 0.8$ but $r < 1.0$, then the reward is 1.0.

In this case, since the reward generated will be non-integral, therefore the normal notion for success and failure will not work in thompson sampling. Therefore, I reparameterized the inputs for the beta distribution to be equal to $\alpha = rewards_a + 1$ and $\beta = pulls_a - rewards_a + 1$ and then continued with the normal thompson sampling for choosing the arms.

To calculate the regret value for task 3, weighted mean reward is computed for each arm from the support and the highest of them is multiplied by horizon to get the highest expected reward value subtracting from which the obtained reward value gives the regret.

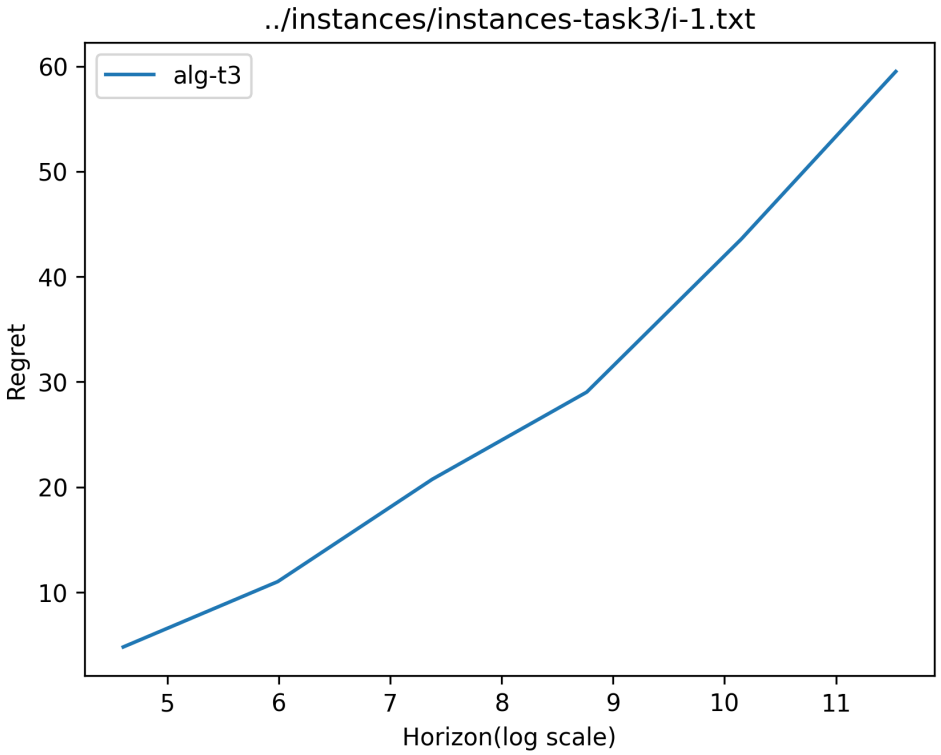

Figure 5: Task-3 Instance-1

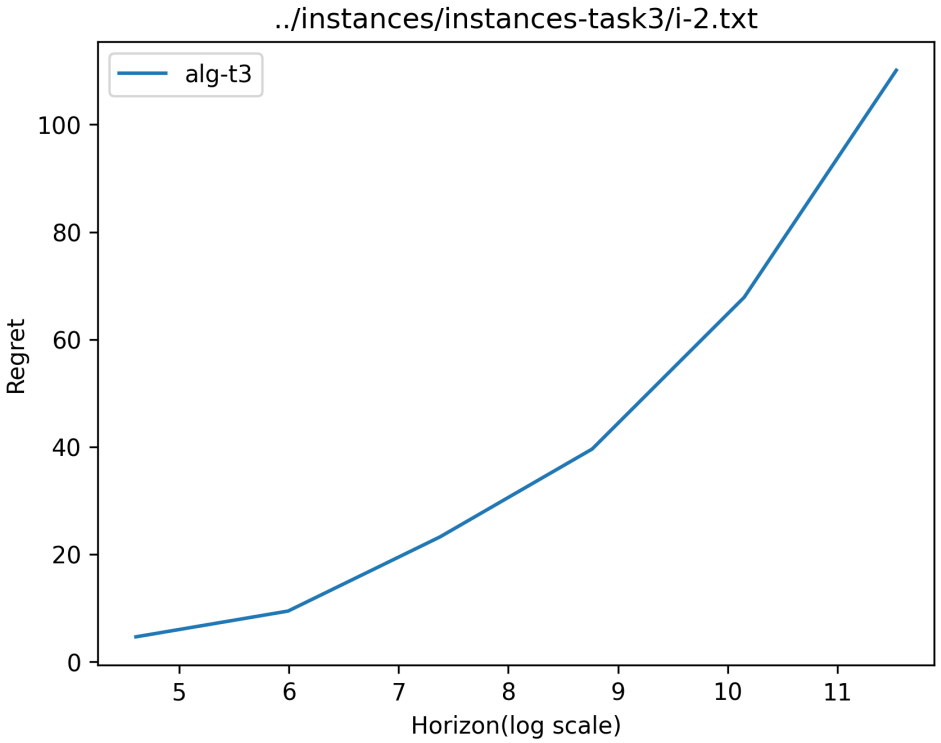

Figure 6: Task-3 Instance-2

Observations for task 3:-

(i) The regret values increases with horizon as it should.
(ii) Since, thompson sampling is used to solve task 3, therefore the regret values obtained are considered good enough.

Task 4

In task 4 also, I used thompson sampling. Since the step of reward generation in task 4 is same as that in task 3 therefore the rewards are generated in similar manner, where the generated random number r is compared with the cumulative sum of the probabilities to generate the reward. The main difference in task 3 and task 4 is the concept of highs.

So after a reward is generated as in task 3, it is compared with the threshold value t. If $reward > t$, then $high = 1$, else $high = 0$ and $low = 1$. Again we can use the general notion of success and failures for the thompson sampling where a success is defined to be equal to the $\#highs$ and failure is defined to be equal to the $\#lows$. So the parameters of beta distribution becomes $= \#highs + 1$ and $\beta = \#lows + 1$.

To calculate the max-highs, the sum of probabilities of those support elements are considered for each arm whose value is greater than the threshold and the highest such sum times the horizon is equal to max-highs. The highs-regret is then equal to the difference of max-highs and obtained highs.

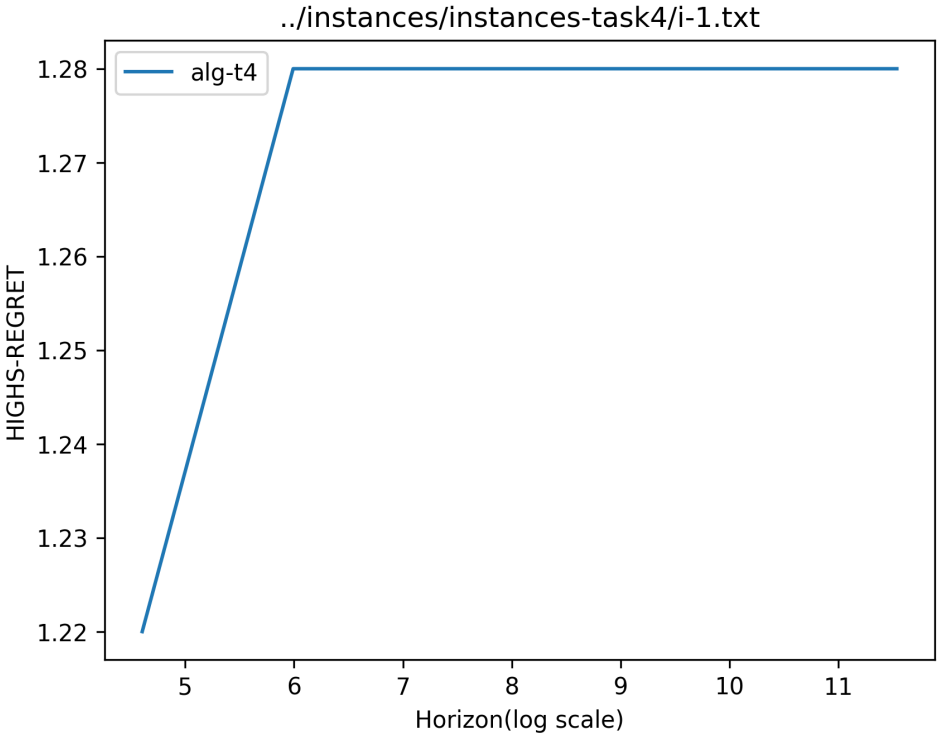

Figure 7: Task-4 Instance-1 t=0.2

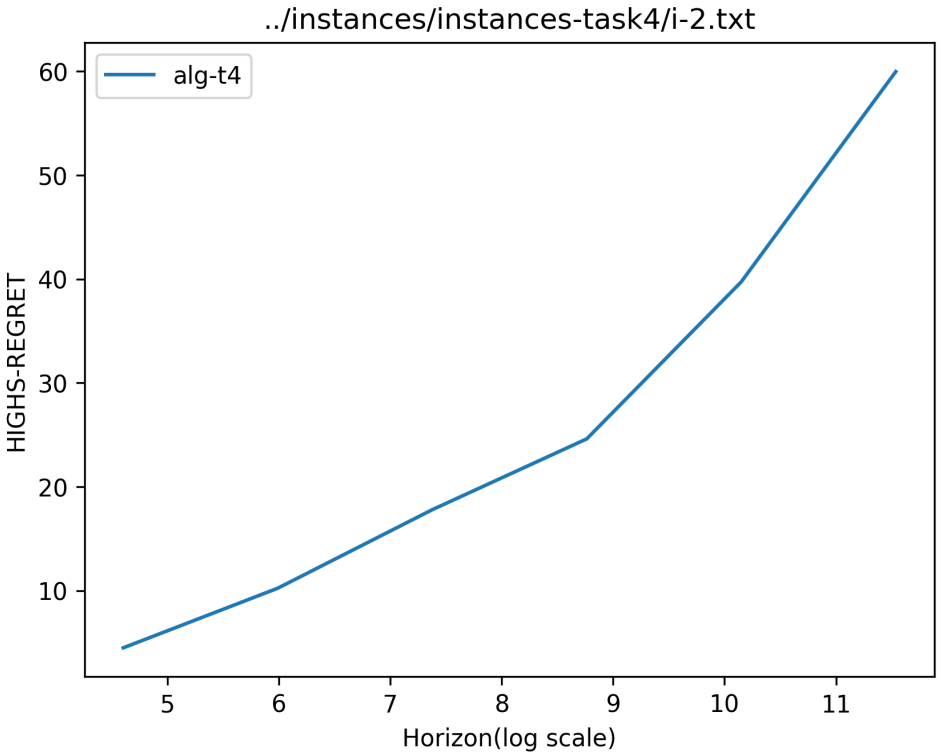

Figure 8: Task-4 Instance-2 t=0.2

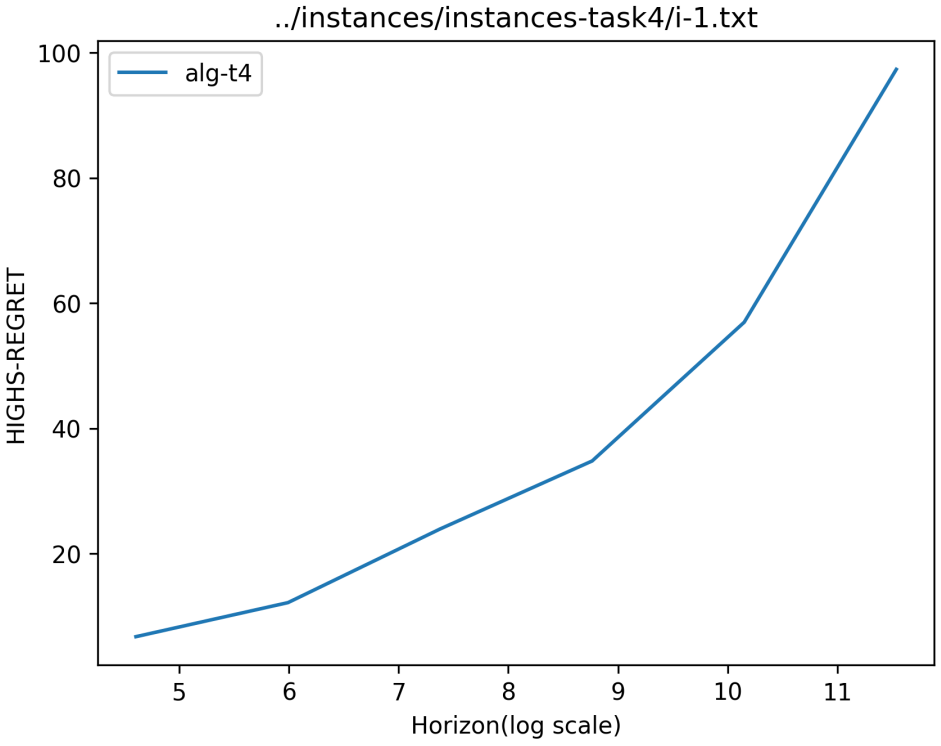

Figure 9: Task-4 Instance-1 t=0.6

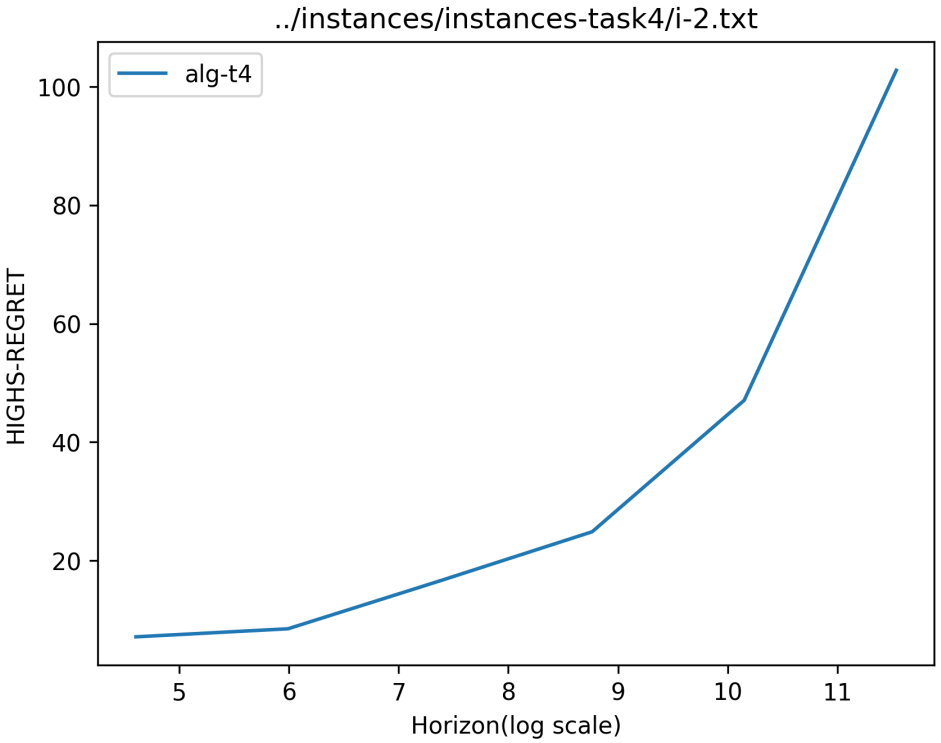

Figure 10: Task-4 Instance-2 t=0.6

Observations for task 4:-

(i) For instance 1 with $t = 0.2$, the regret is almost negligible even for higher horizon values. This is because there are 2 out of 3 arms which are optimal for the given threshold and will always give high value equal to 1 and since we are using thompson sampling, the expected result came out great.
(ii) For other 3 cases, the regret values increases with horizon as it should.
(iii) For $t = 0.2$, the regret is smaller than that for $t = 0.6$ because more number of rewards are expected to exceed the threshold as the threshold is fixed at a smaller value.