CS 747, IIT Bombay
Lab #1

Prof. Shivaram Kalyanakrishnan

Vibhav Aggarwal (190050128)

Initializations

- **Epsilon-Greedy:** The initial value of empirical means is set to 0.5.
- **UCB:** Each arm is pulled once.
- **KL-UCB:** 3 random pulls are made so that $\ln(t) + c\ln(\ln(t))$ becomes positive.
- **Thompson Sampling:** No initialization required.

Tie breaking

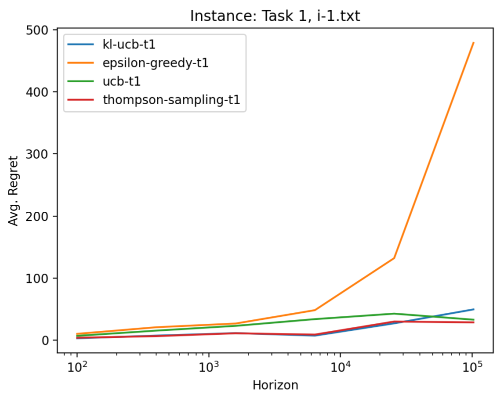
Tie breaking is done by the default behaviour of `numpy.argmax` which is to select the **first** max value.

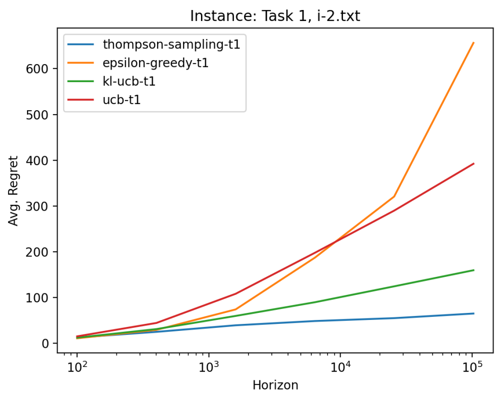
Scripts

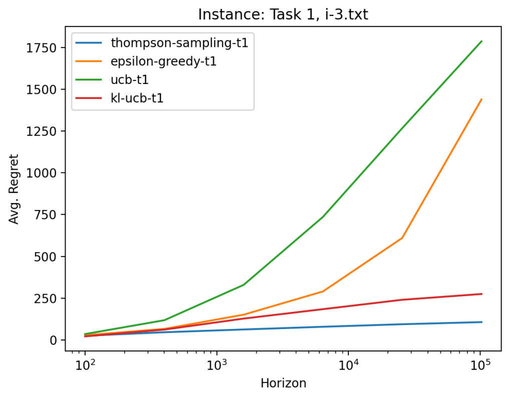
- `bandit.py`: All the arguments are required to run this script.
- `runner.py`: This is a wrapper script to run `bandit.py` and generate `outputData.txt` on all the combinations. Takes about 6.5hrs to run completely.
- `plots.py`: This script is to generate plots from the data. Run it as: `python3 plots.py --instance ../instances/instances-task1/i-1.txt --threshold 0`. For task 2, just use `task2` in the `--instance` argument.

Plots

Task 1





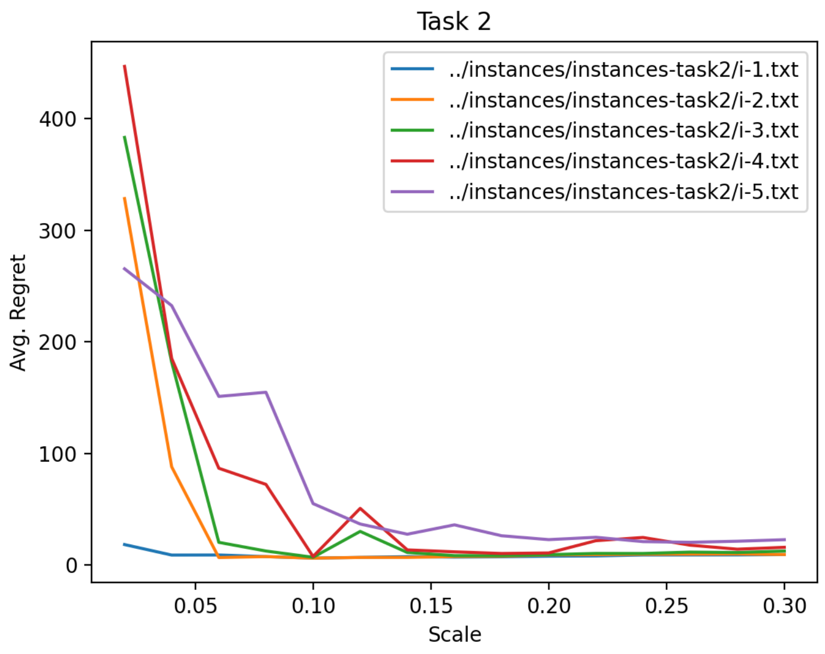


Task 1 continued on next page...

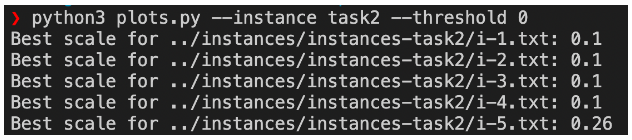
It can be seen that the average regret increases (most of the time) with increase in horizon for all the algorithms as one would expect. Also, Thompson Sampling and KL-UCB perform the best in all three instances with Thompson Sampling having some edge over KL-UCB.

An interesting observation is regarding the $\epsilon-$greedy vs. UCB algorithm. We know that UCB will give better regret eventually because its regret is logarithmic as opposed to linear. However, for instance 3, it gives higher regret for all the horizons tested and even for instance 2, it gives higher regret until horizon = 1e+4. This is because instance 2 has 5 arms and instance 3 has 25 arms. When there are more arms in the bandit, then the u_a^t values will remain small for longer duration of time and there will be a lot of exploration during that time. It is because of these explorations that we observe a high regret in the start. In the third graph, one can see that slope of UCB curve is lower than the $\epsilon-$greedy one towards the end. So if we run it for longer horizon, the two curves will indeed cross each other.

Task 2



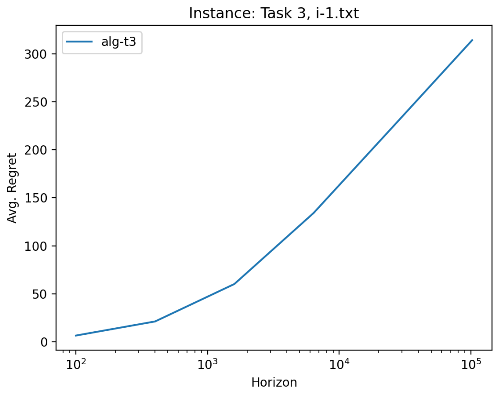
The best scale values found are:

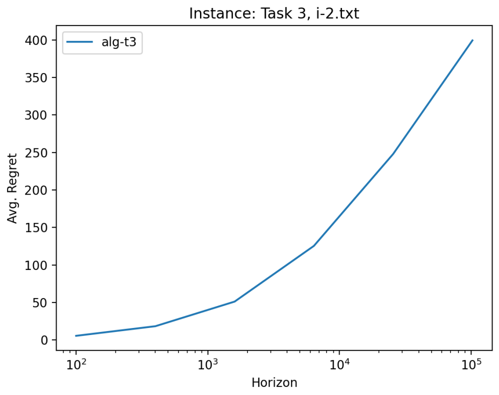


```
❯ python3 plots.py --instance task2 --threshold 0
Best scale for ../instances/instances-task2/i-1.txt: 0.1
Best scale for ../instances/instances-task2/i-2.txt: 0.1
Best scale for ../instances/instances-task2/i-3.txt: 0.1
Best scale for ../instances/instances-task2/i-4.txt: 0.1
Best scale for ../instances/instances-task2/i-5.txt: 0.26
```

We first see a sharp decreasing trend in regret which becomes almost constant for higher values. This is because at lower scale values, there is almost no exploration so the algorithm always pulls a sub-optimal arm (with high probability). As c increases, exploration increases and we get a better regret. If we continue to increase the scale however, the exploration will increase too much unnecessarily and will produce higher regret again.

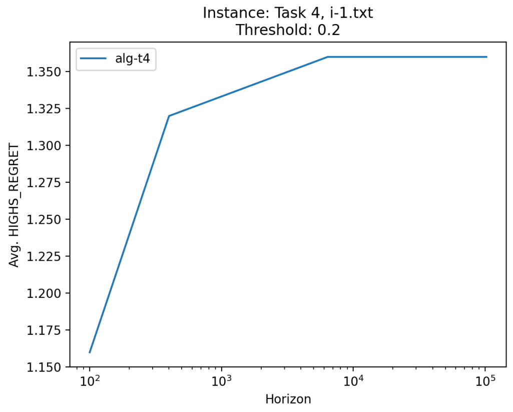
Task 3

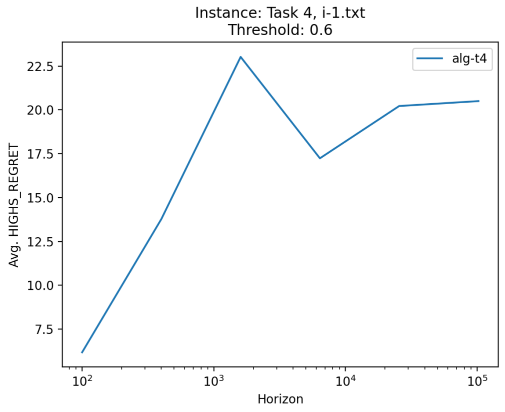


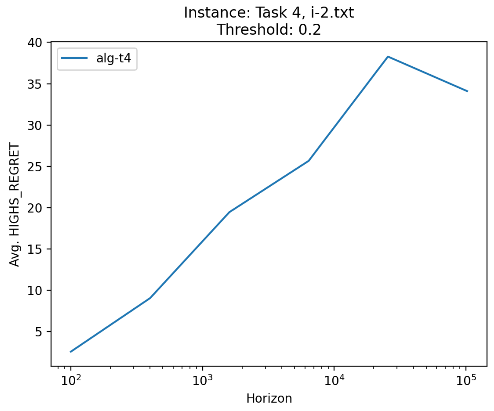


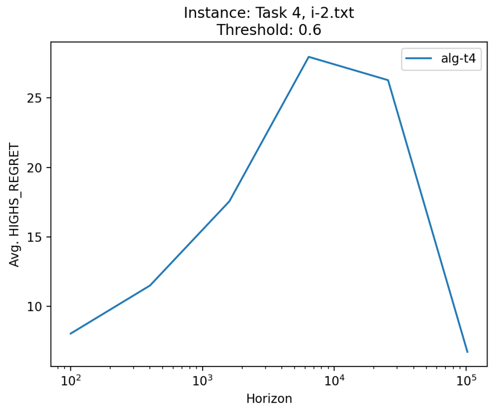
I have used the UCB algorithm in this task. This is because, in the derivation of UCB, we never made any assumptions on the distribution or support of each arm. KL-UCB could also be used would have probably given lower regret values but I did not use it because it is much slower as compared to UCB.

Task 4









Task 4 continued on next page. . .

This task can be reduced to a simple multi-arm bandit problem with bernoulli distribution. If the reward received is greater than threshold, then we consider it 1-reward, otherwise 0-reward. The probability of getting a 1-reward for some arm a is:

$$\hat{p}_a = \sum_{s \in S_a, s > th} p(s)$$

where S_a is the support set of the arm a. We wish to identify the arm a with the highest value of $\hat{p}_a$.

Since the distribution is bernoulli, I have used Thompson's Sampling algorithm for this task. Whenever an arm is pulled, if its reward is greater than the threshold, then it is considered as a success (and failure otherwise).