CS747 Assignment - 1

Name : Agulla Surya Bharath **Roll.No : 17D070055**

TASK - 1 :

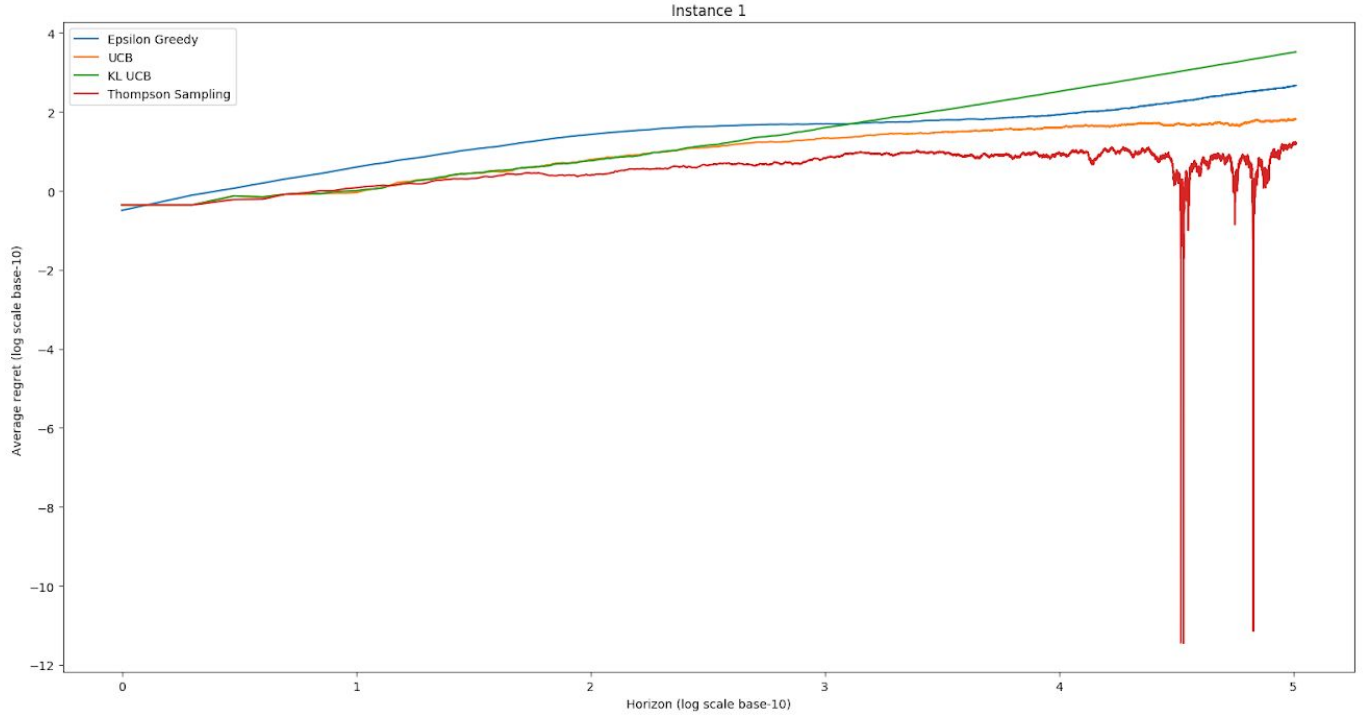
First few pulls information :

1. **UCB : I have pulled each arm exactly twice before following the algorithm**
2. **KL-UCB : I have pulled each arm exactly twice before following the algorithm & used {c=3} for the bound , just as mentioned in the class {c >= 3} theoretically in the main paper too.**
3. **Thompson Sampling : I have pulled each arm exactly once before following the algorithm**
4. **Thompson Sampling with hint : I have pulled each arm exactly once before following the algorithm**

Expect the above mentioned things, all others in T1 are followed as said in class

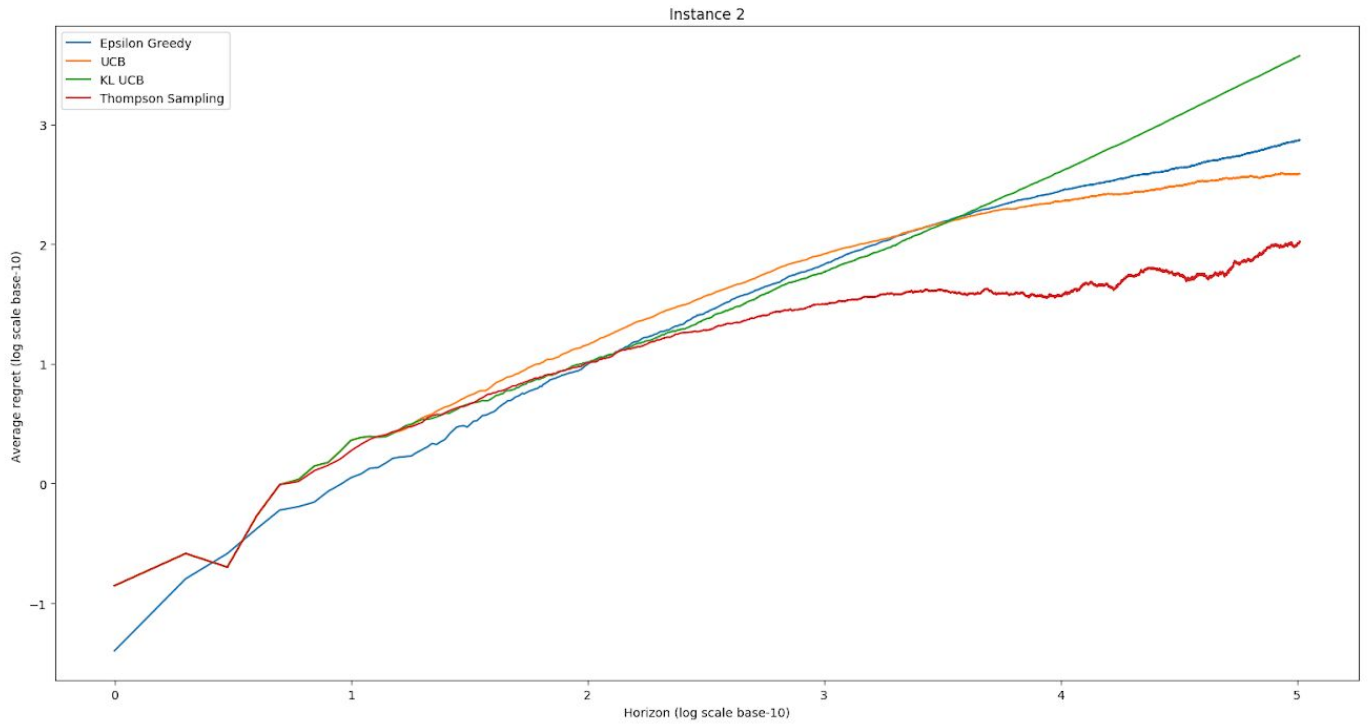
In each of the following three plots (corresponding to 3 instances) the x- axis represent no.of pulls(horizon) on a log scale and Y-axis represents the Average Regret(over seeds)

As we know , Performance of an algorithm is better if the average regret is less.



Performance :

{ Thomson Sampling > UCB > Epsilon greedy > KL-UCB } for larger horizons .

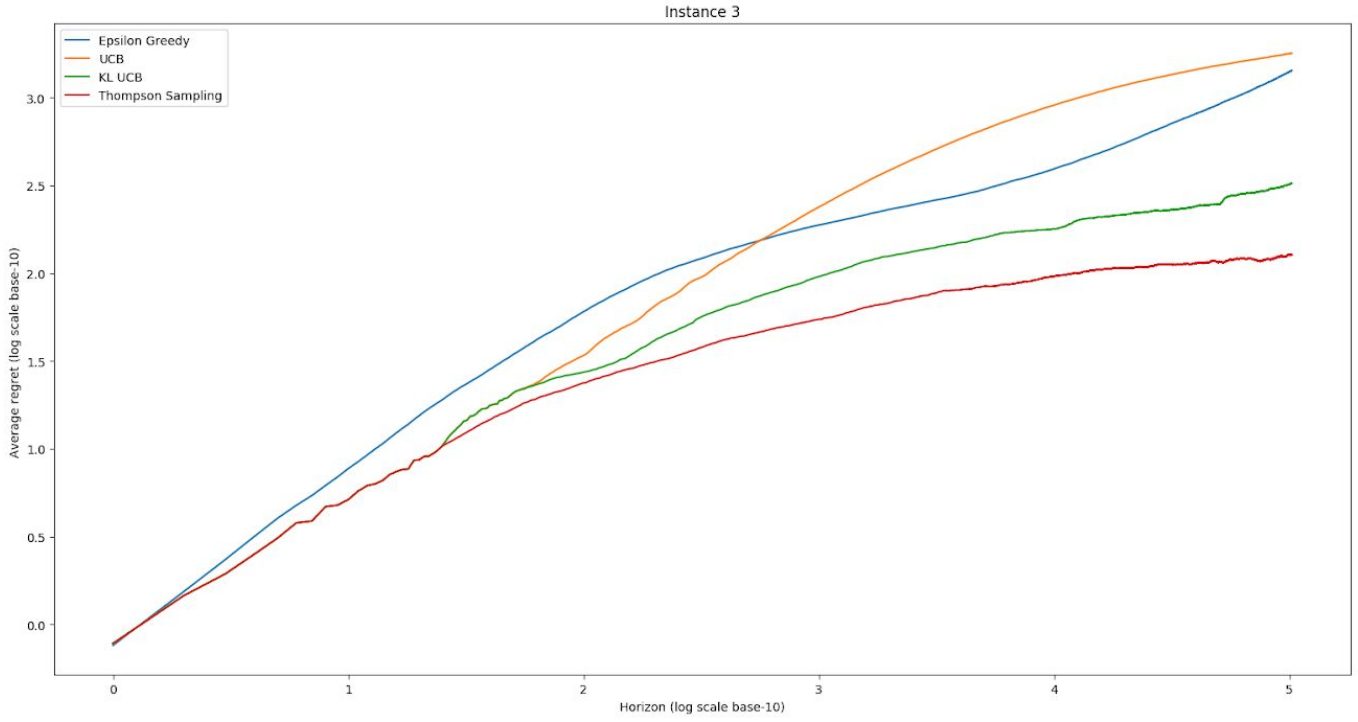


Performance :

{ Epsilon greedy > Thompson Sampling } for small horizons(<100) .

{Thomson Sampling > KL-UCB > Epsilon Greedy > UCB } for intermediate horizons.

{ Thomson Sampling > UCB > Epsilon greedy > KL-UCB } for larger horizons .

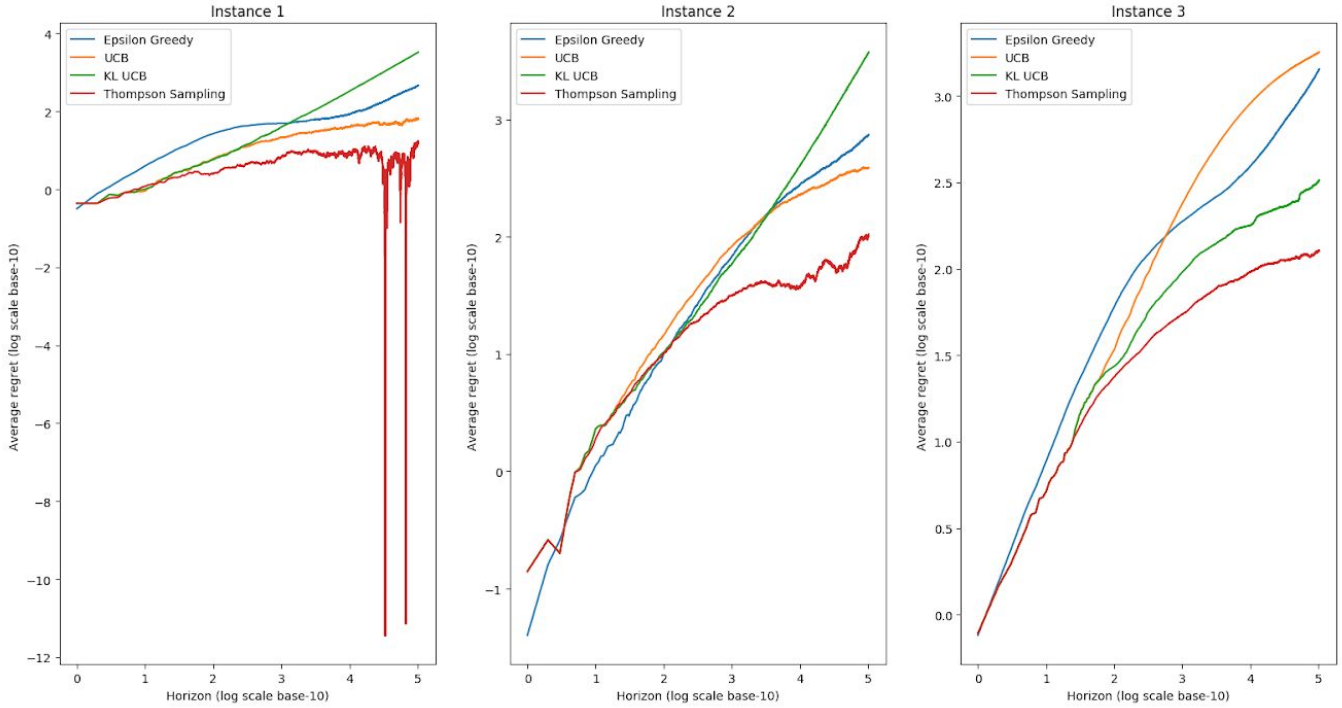


Performance :

{ Thomson Sampling > KL-UCB > UCB > Epsilon greedy } for the Small horizons .

{ Thomson Sampling > KL-UCB > Epsilon greedy > UCB } for most of the Large horizons . From the slope it looks like UCB is going to eventually perform better than epsilon greedyfor extremely large horizons.

Comparative Performance analysis :



1. **Thompson Sampling is working best among all 4 , for all instances for reasonably large horizons(>50-100).**
2. **KL-UCB is working better than UCB for instances with larger number of arms. In the counter case their relative performance is dependent on horizon.**
3. **Epsilon Greedy is working comparatively good for Large horizons.**

Task 2 :

Thompson Sampling with Hint Implementation :

The main aim of this algorithm design is to get a better performance than the original Thompson sampling , when a hint of sorted list of true means of the bandit instance is provided to us before doing the experiment..

Step 1 : As the main motto is perform better than Thompson sample, my idea is to follow original Thompson sampling for the first half of horizon(irrespective of horizon value) and thereafter try to tweak the confidences generated by Thompson sampling in the next half of the horizon.

Step 2: After performing the original Thompson Sampling for the first half of the horizon, we are left with a list of empirical means of arms. For next pull, first compute the *optimality confidence list of arms* (by sampling from beta distribution as per original Thompson sampling).Then compute the *distance list* which contains the absolute value of the between the empirical mean of arm and maximum value in the sorted Hint means list(i.e true optimal arm mean)

Step 3 : Calculate the rank of each arm in the *distance list* and say this is ranks list to be Rank1 (Small distance gives small rank for each arm) .
Then calculate the rank of each arm in the *optimality confidence list* and say this is ranks list to be Rank2 (large confidence value gives small rank for each arm).

Step 4 : Compute a list called Rank12 = Rank1 + Rank2 . Then generate another list called **ranking list**(of size = no.of arms) which contains the rank of each arm in the Rank12 list.

Step 5 : Create a list (Sorted emp mean list) containing empirical arms , sorted according to their rank in ranking list. Thereafter calculate the absolute value of difference between the 'Sorted emp mean list' and 'Hint means list'. Let this list be "Difference List"

Step 6: Now , take the arms (a) with **ranking** <= log(no.of arms) . Replace the original confidence (sampled from beta distribution) of these arms with new confidence .

New confidence(a) = (Original Confidence(a)) * (Difference List(a) / distance list(a))

For other arms (a) with **ranking** > log(no.of arms), the New Confidence is same as the original confidence generated by sampling from beta distribution….

Then pull the arm which has the maximum value of New confidence...

Justifications for above steps :

Ideally we want an arm with more original confidence and least distant from the maximum optimal mean given as a hint. So , I ranked the arms by adding the Rank1 and Rank2 and then sorting them. I believed the original Thompson sampling idea(i.e beta distribution sampling) to a small extent .So, I only wanted to tweak the first few arms of higher original confidence .

To alter confidence I looked at two parameters.

1. Difference List : If this value is low for an arm it says that the current rank of the arm might be close to its rank in the list of true means.
2. Distance list : Closeness of empirical mean of arm to optimal mean in given hint.

So if the Distance list(arm) is less then we look at the difference list(arm) and it difference list is also more then we try to bring it up in the rank (by updating confidence by large value) as the probability of that arm being optimal is large . This is what is implemented in the last step.

The average reward (over 50 seeds) obtained for each instance are :

TS : Original Thompson Sampling .
TS hint is : Thompson Sampling with hint

----------> For instance 1 <----------
for horizon 102400 TS avg reward = 81905.18 TS hint avg reward = 81905.66
for horizon 25600 TS avg reward = 20472.42 TS hint avg reward = 20471.78
for horizon 6400 TS avg reward = 5113.18 TS hint avg reward = 5114.16
for horizon 1600 TS avg reward = 1271.16 TS hint avg reward = 1271.44
for horizon 400 TS avg reward = 315.24 TS hint avg reward = 316.6
for horizon 100 TS avg reward = 77.56 TS hint avg reward = 77.88

----------> For instance 2 <----------
for horizon 102400 TS avg reward = 51094.66 TS hint avg reward = 51106.9
for horizon 25600 TS avg reward = 12739.5 TS hint avg reward = 12747.38
for horizon 6400 TS avg reward = 3161.34 TS hint avg reward = 3165.06
for horizon 1600 TS avg reward = 763.84 TS hint avg reward = 770.48
for horizon 400 TS avg reward = 178.22 TS hint avg reward = 183.06
for horizon 100 TS avg reward = 39.88 TS hint avg reward = 41.62

----------> For instance 3 <----------
for horizon 102400 TS avg reward = 98176.58 TS hint avg reward = 98191.9
for horizon 25600 TS avg reward = 24466.44 TS hint avg reward = 24480.34
for horizon 6400 TS avg reward = 6057.66 TS hint avg reward = 6062.88
for horizon 1600 TS avg reward = 1471.94 TS hint avg reward = 1481.1
for horizon 400 TS avg reward = 342.16 TS hint avg reward = 349.76
for horizon 100 TS avg reward = 72.26 TS hint avg reward = 74.68

Thompson Sampling with hint is performing BETTER than Original Thomson sampling..

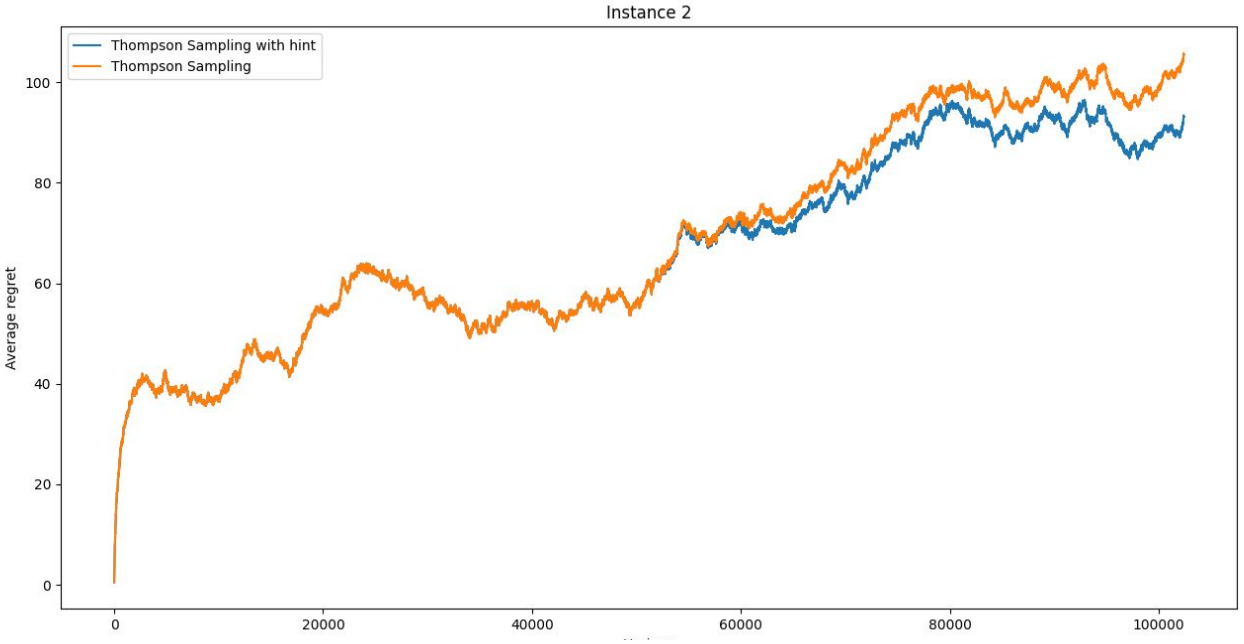
NOTE : Since my method for Thompson sampling with hint is horizon dependent (first half pulls, second half pulls) .It would be very difficult to generate plot of Avg Regret vs Horizon, because for each horizon point I have to run algorithm for 50 times and get avg regret => algorithm should run {102400*50} times and also time taken increases with the horizon .(where as in other algorithms above by calculating for horizon = 102400 we get all other

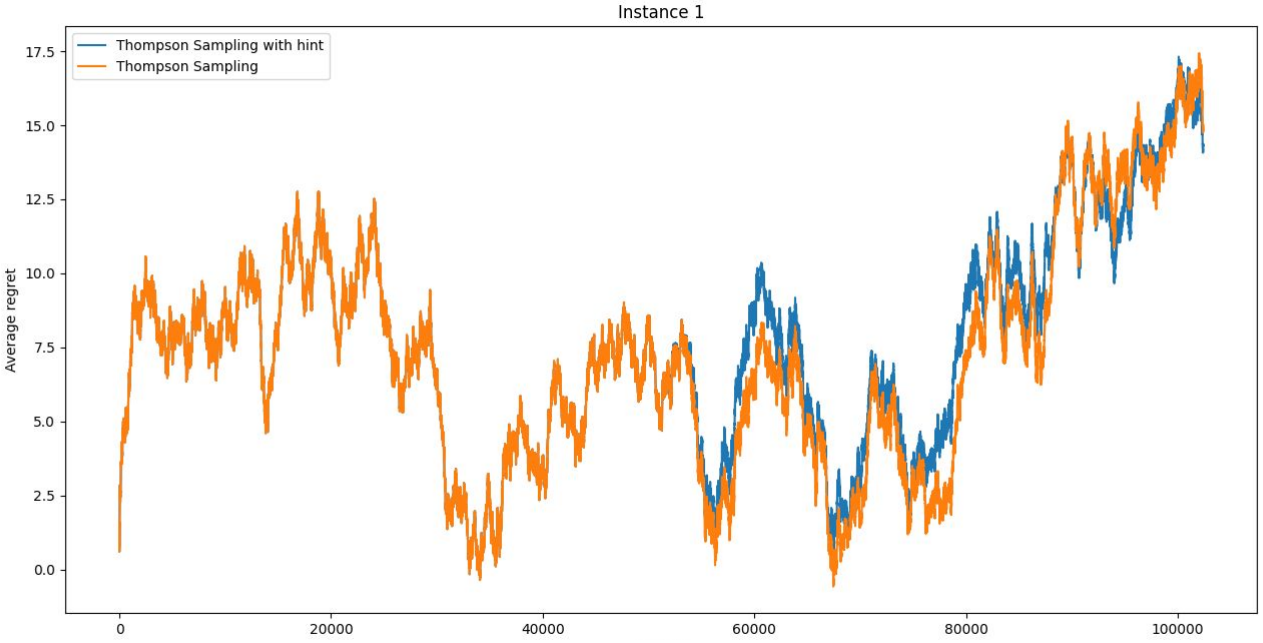
lesser horizon regret too. So running algorithm 50 times with horizon = 102400 would suffice there)

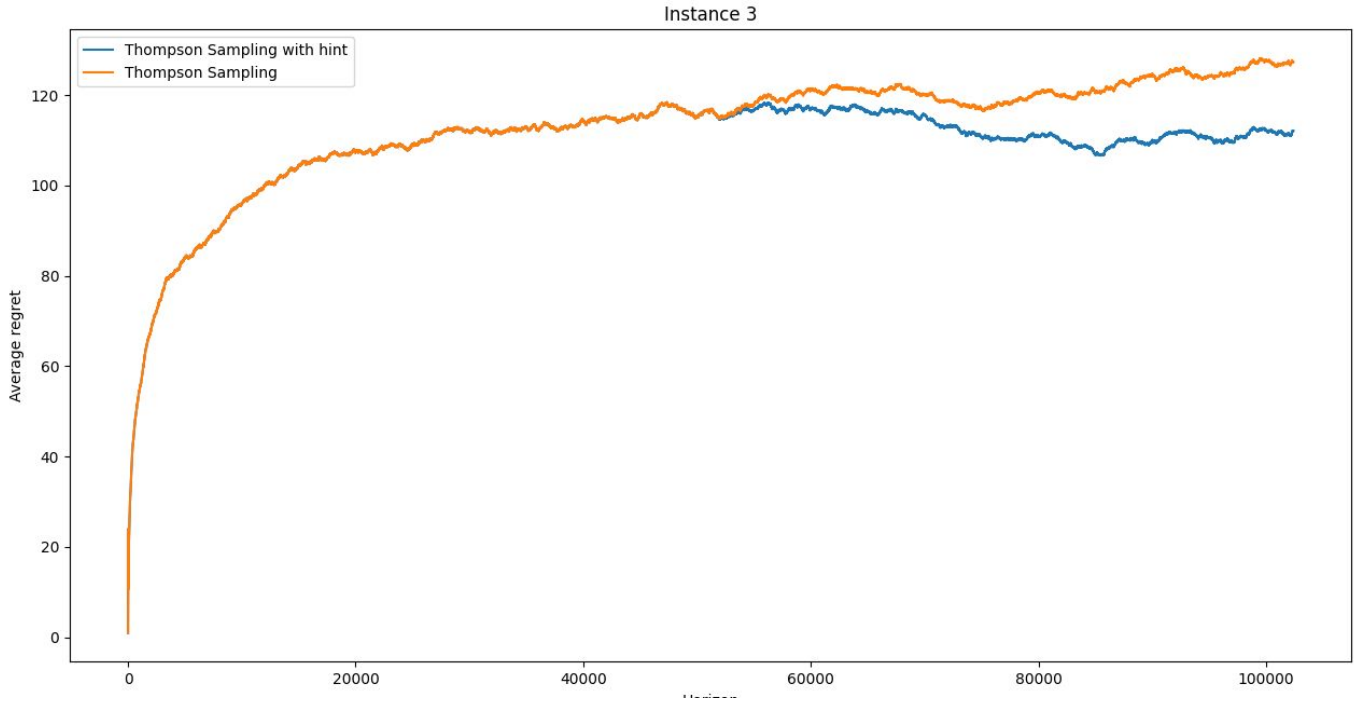
So, what I can guarantee you is that my algorithm performs atleast equal or better than thompson sampling for almost all of the pulls . Thus I am showing the plots for horizons 102400, 25600,6400, 1600, 400,100

Comparison Plots for each instance are displayed below :

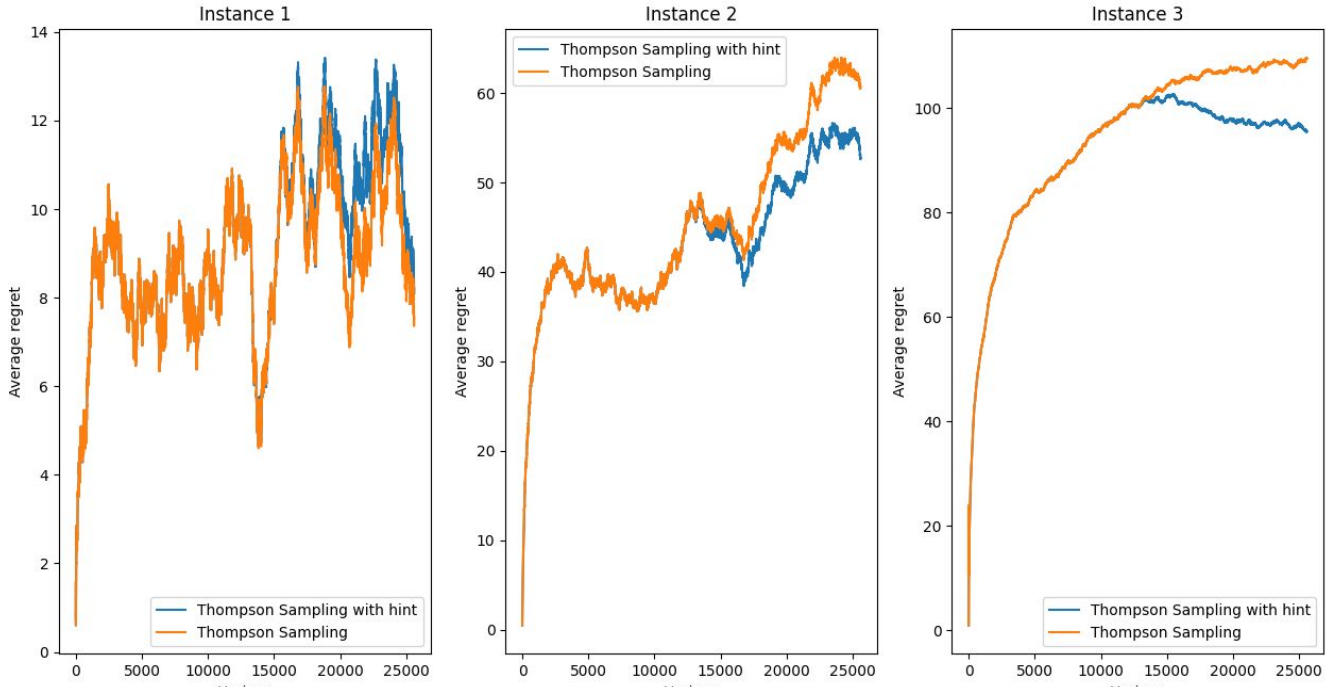
1 . For horizon = 102400 (X axis show no.of pulls)



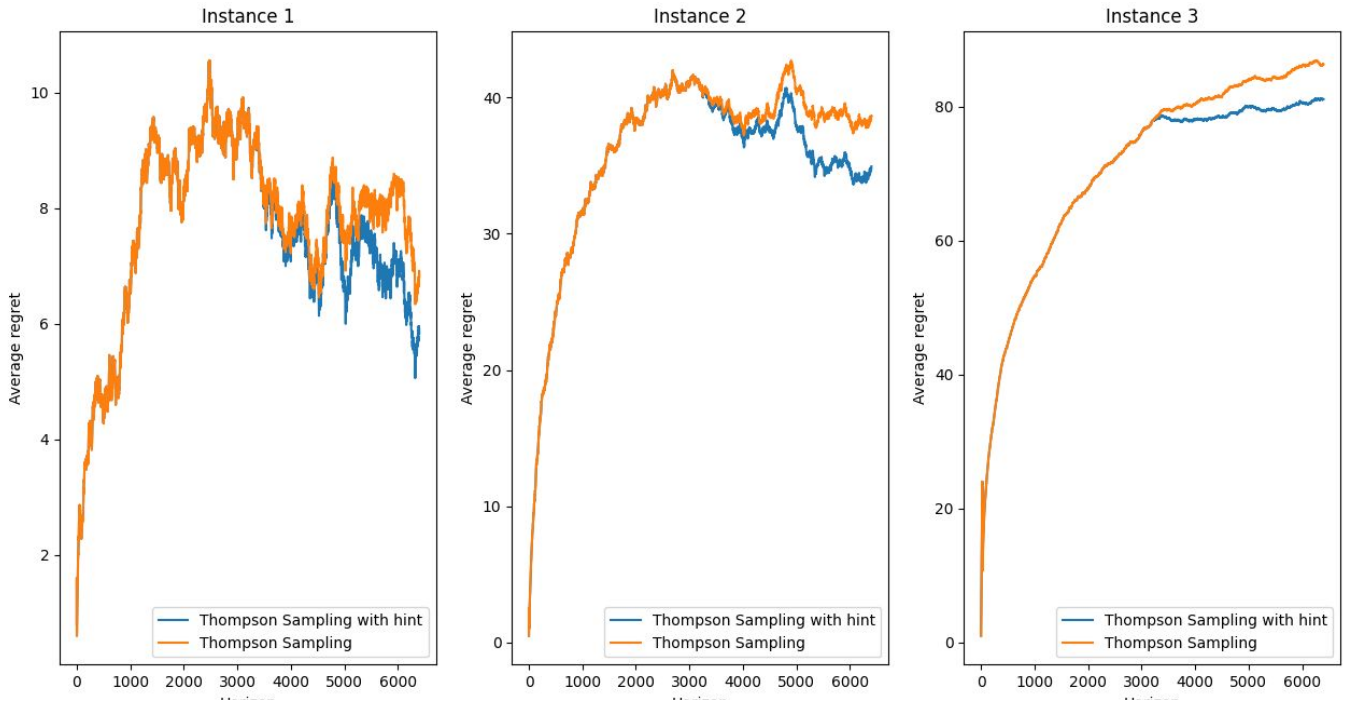




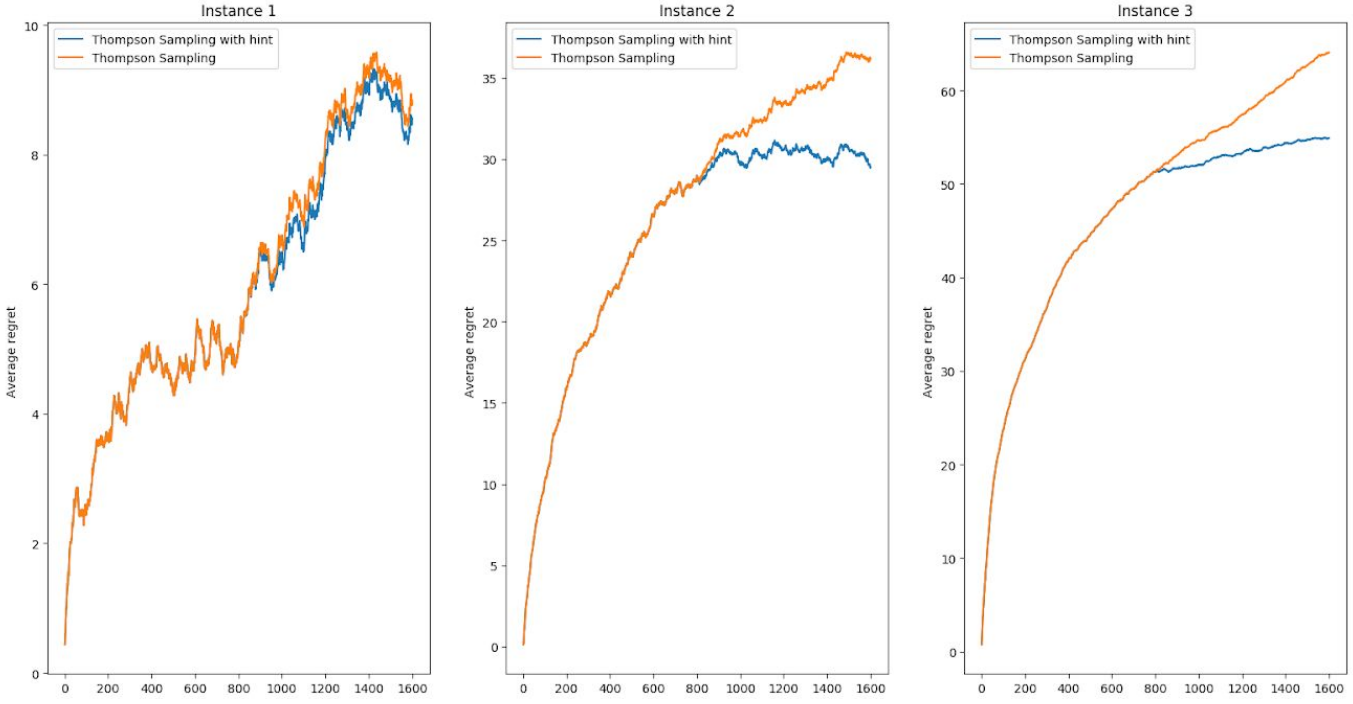
2. For Horizon = 25600 (X axis shows the no.of pulls)



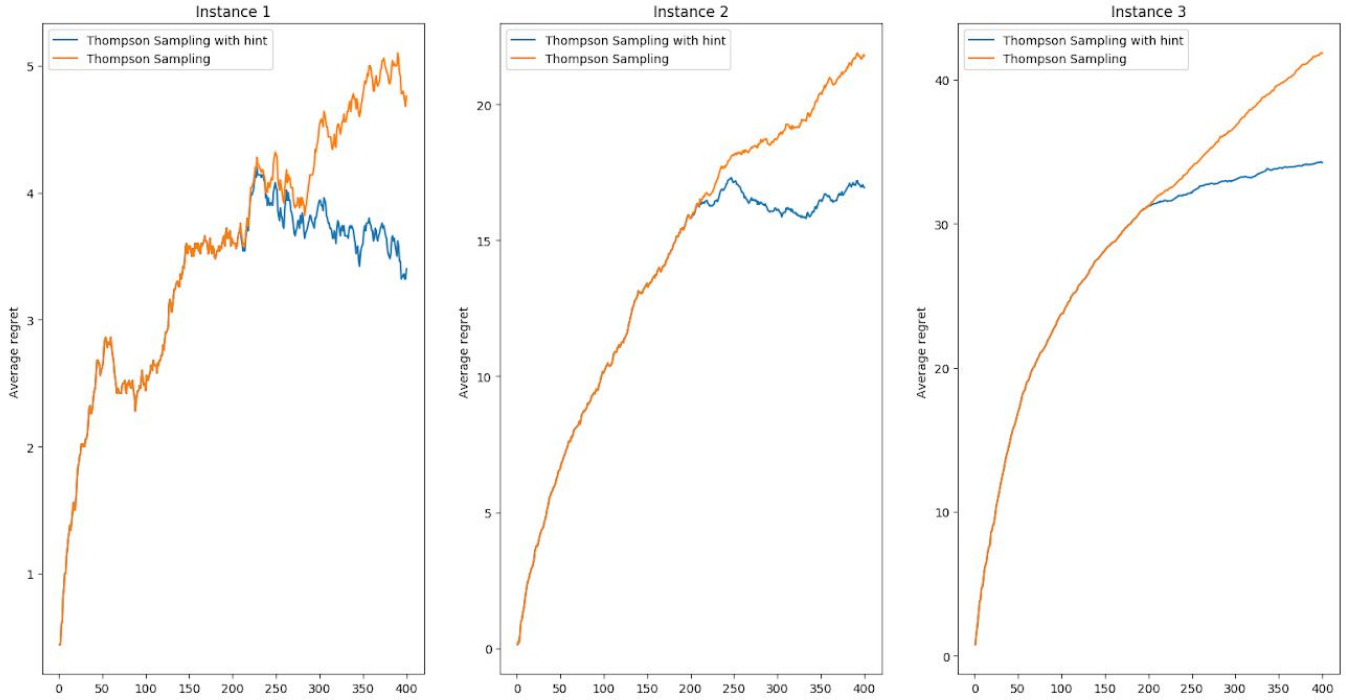
3.For Horizon = 6400 (X axis shows the no.of pulls)



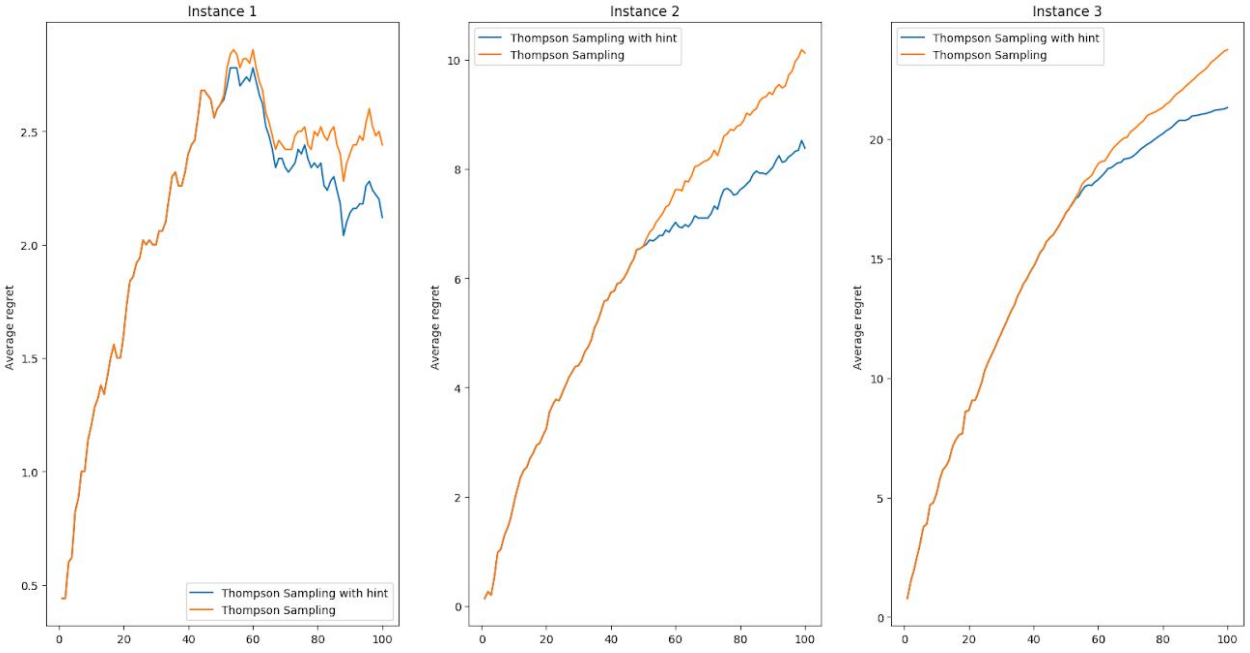
4. For Horizon = 1600 (X axis shows the no.of pulls)



5. For Horizon = 400 (X axis shows the no.of pulls)



6. For Horizon = 100 (X axis shows the no.of pulls)



Task 3 :

Instance - 1 :

```
ep-1 = 0001,   Average Regret =   18456.72
ep-2 = 0.4,    Average Regret =  4109.66
ep-3 = 0.9,    Average Regret =  8755.82
```

Instance - 2 :

```
ep-1 = 0.0001, Average Regret =   18416.1
ep-2 = 0.2,    Average Regret =  4126.64
ep-3 = 0.9,    Average Regret =  8349.38
```

Instance - 3 :

```
ep-1 = 0.0001, Average Regret =   38107.78
ep-2 = 0.1,    Average Regret =  4327.26
ep-3 = 0.9,    Average Regret =  27402.74
```
