Programming Assignment 1 : CS 747

Bhaskar Gharu, 180050023

September 2020

Implementation details and assumptions for Task 1 (T1)

Epsilon-greedy ($\epsilon - 3$)

- The algorithm explores with probability **epsilon** and exploits with probability **1-epsilon**
- Each arm is sampled once, before starting the algorithm
- Samples an arm uniformly from given instance of n arms with equal probability for exploration
- Samples an arm with best empirical mean at the current step for exploitation
- For exploitation all ties are broken based on the smaller index of arm in instance

UCB

- Samples an arm based on the maximum **Upper confidence Bound** value of the instance
- UCB value is evaluated according to $ucb_t = p_t + \sqrt{2 * \ln t / u_t}$ where p_t is the empirical mean and u_t is number of times arm has been pulled
- Each arm is sampled once, before starting the algorithm
- Ties are broken based on the smaller index of arm in instance

KL-UCB

- Samples an arm based on the maximum **KL-UCB** value of the instance
- KL-UCB value is evaluated as follows for each arm
  $\max_q\{u_t * KL(p_t, q) \leq \ln t + 3 * \ln\ln t\}$ here $q \in [p_t, 1]$
  where p_t is the empirical mean u_t is number of times arm has been pulled
- Each arm is sampled once, before starting the algorithm
- Ties are broken based on the smaller index of arm in instance

Thompson Sampling

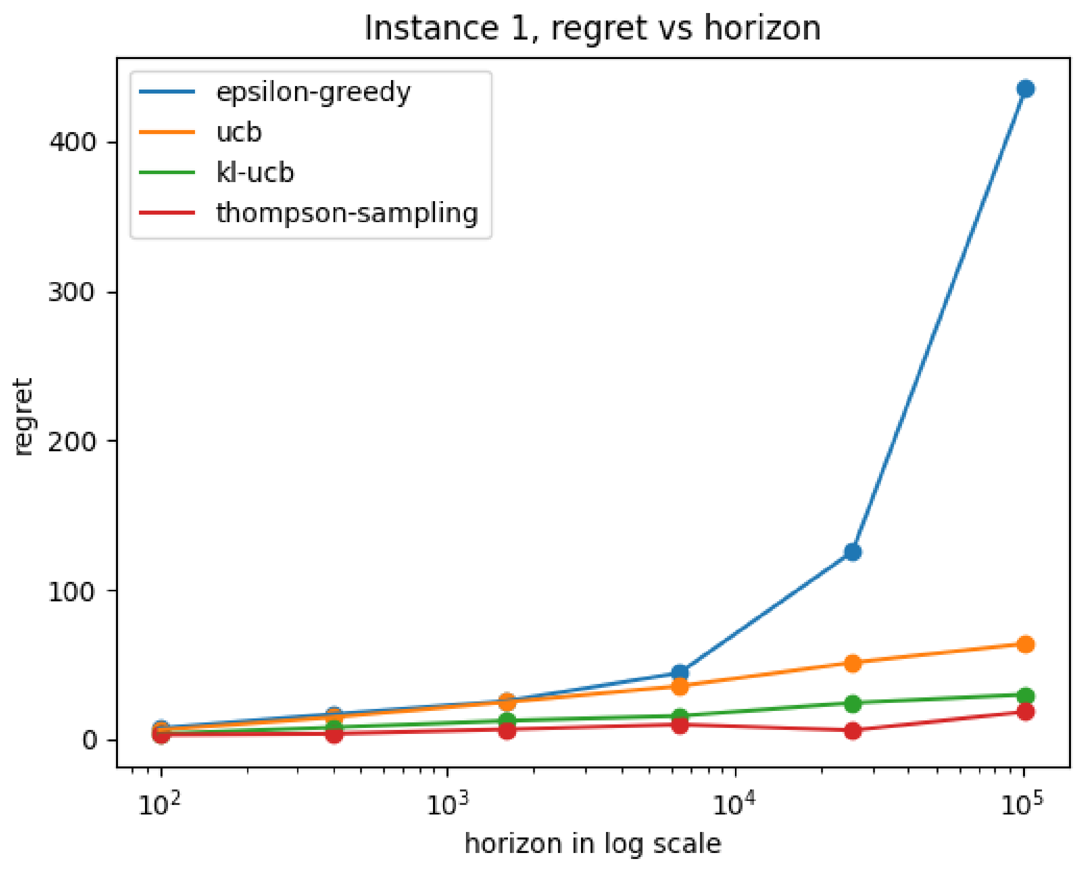
- At every step each arm generates a sample from the ∼beta($s_t + 1, f_t + 1$) where s_t and f_t are success and failure count for corresponding arm
- The arm with the maximum value of the above mentioned sample is chosen for sampling
- Ties are broken based on the smaller index of arm in instance

Plots explanations for task 1 (T1)

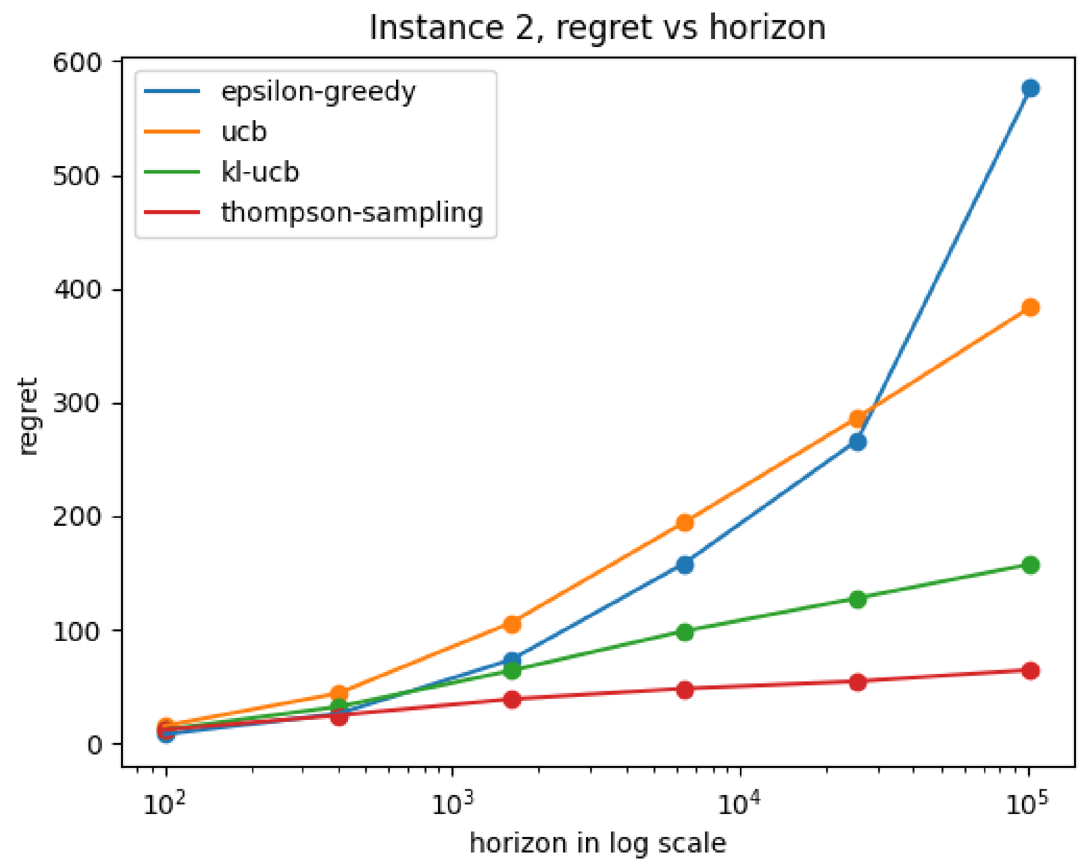
- for instance 1 and 2 the result is as expected, epsilon greedy shows exponential regret w.r.t log scale horizon and rest of the algorithms are showing linear regret w.r.t log scale horizon which agrees with Lai and Robbins bound on regret

- for instance 3 the plot regret curves are in accordance with Lai and Robbins but UCB gives greater regret compared to epsilon greedy. One of the contributing factor for this can be that we have number of means closer to optimal mean is more for this instance leading to overlap in confidence intervals of arms which in turn reduces the likeliness for optimal arm to be picked

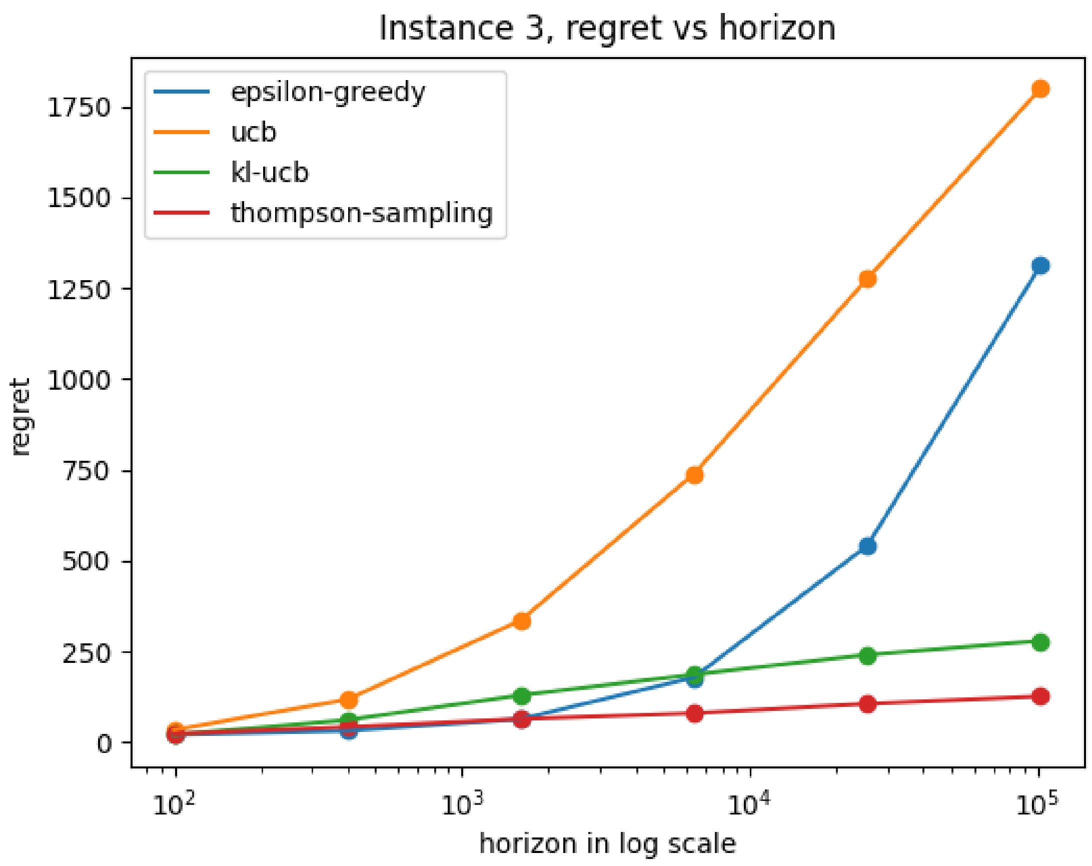
Plots for Task 1 (T1) Instance 1



Plots for Task 1 (T1) Instance 2



Plots for Task 1 (T1) Instance 3

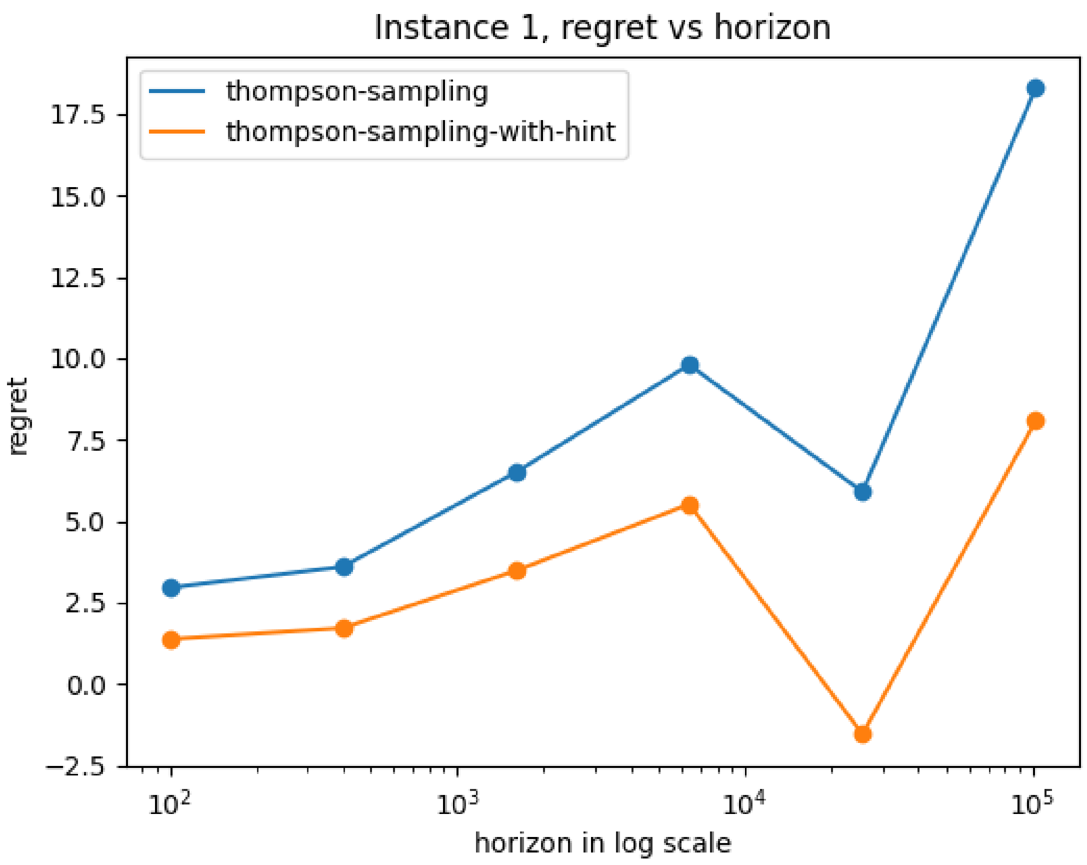


Implementation details and assumptions for Task 2 (T2)

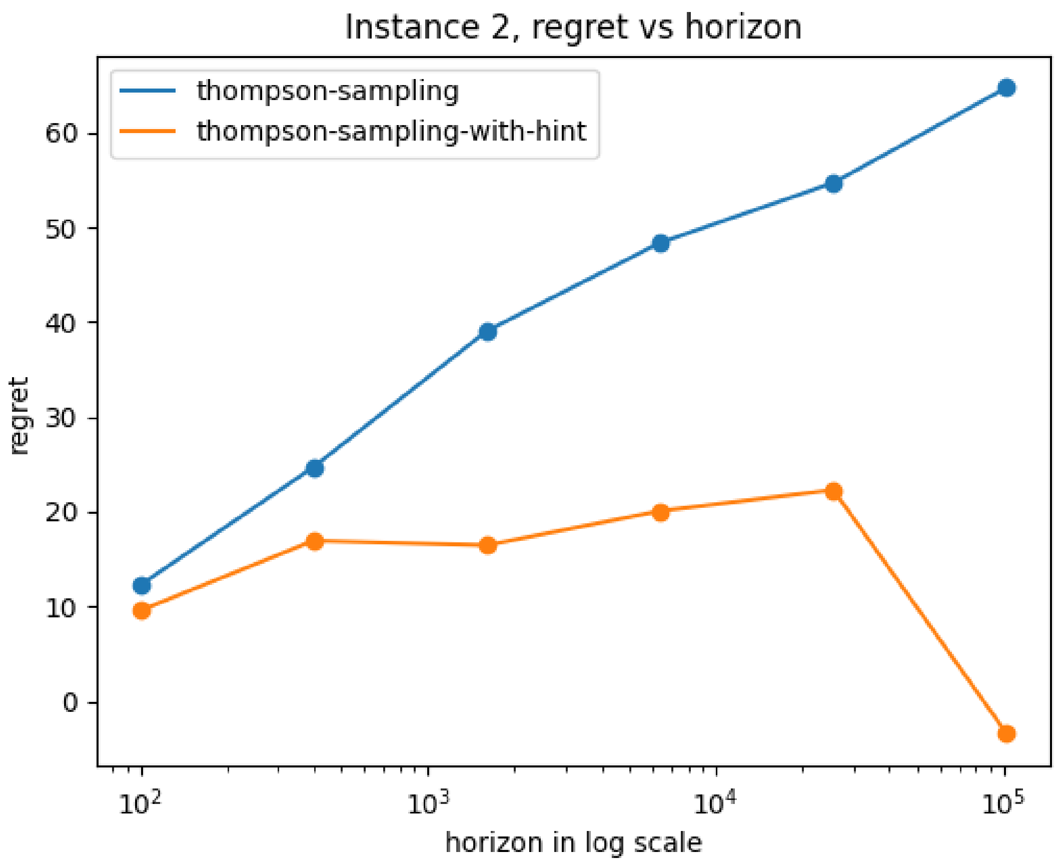
Thompson Sampling with hint

- For each arm maintained an custom probability distribution based on the given sorted permutation of true means, which represents the likeliness of the given arm to have the corresponding probability in the permutation
  - The given true probabilities of the instance are stored as a sorted list of distinct elements (i.e. duplicates are not considered in permutation) from hereafter denoted as **P**
  - For a given arm initially each mean is equally likely with the probability $frequency(p_j)/\sum_{j=1}^{j=k} frequency(p_j)$ where p_j is the jth mean in **P** ,$frequency(p_j)$ is the frequency of p_j in the input instance of algorithm and **k** is the total number of distinct true means in the instance (i.e length of **P**), Note that $\mathbf{k} \leq \mathbf{N} and \sum_{j=1}^{j=k} frequency(p_j) = \mathbf{N}$, where **N** is the number of arms in the instance
  - The above mentioned point is basically the initialisation of the custom distribution (**Prior**) for each arm
  - each arm is sampled once before the start of the algorithm and their corresponding distributions are also updated
  - Now when the algorithm starts after initial pulls, at a given step t, we sample a value for each arm based on their respective distributions (uniform sampling) and select the arm with the maximum value of the sample generated, thereafter we sample the selected arm at this step and based on the reward generated we update it's distribution (belief) as follows
    Distribution * P if reward was a success
    Distribution * (1-P) if reward was a failure
    Note that * denotes the element wise product of Distribution list and probability list also **Distribution[i]** represents the probability of this arm having true mean equal to **P[i]** and **1-P** represents a list such that each element is 1-**P[i]** for each element **P[i]** in **P**
  - after the updating of belief for selected arm we normalize its **Distribution**
  - Note that for selecting an arm to be pulled at a given step we generate sample from each arm's distribution until we get a clear unique maximum sample i.e. ties are not required to be handled in my implementation
- Observe that the above algorithm captures the essence of thompson sampling (**bayesian**) with minor modification that the domain of probability is now known as a finite discrete set unlike the continuous $[0, 1]$ interval , which lead to a discrete custom distribution of means for each arm unlike the continuous beta distribution in thompson sampling
- Note that above illustration of the algorithm is scaled down to 1 dimensional lists for simplicity in explanation but implementation is done using 2-D numpy arrays for time complexity optimizations

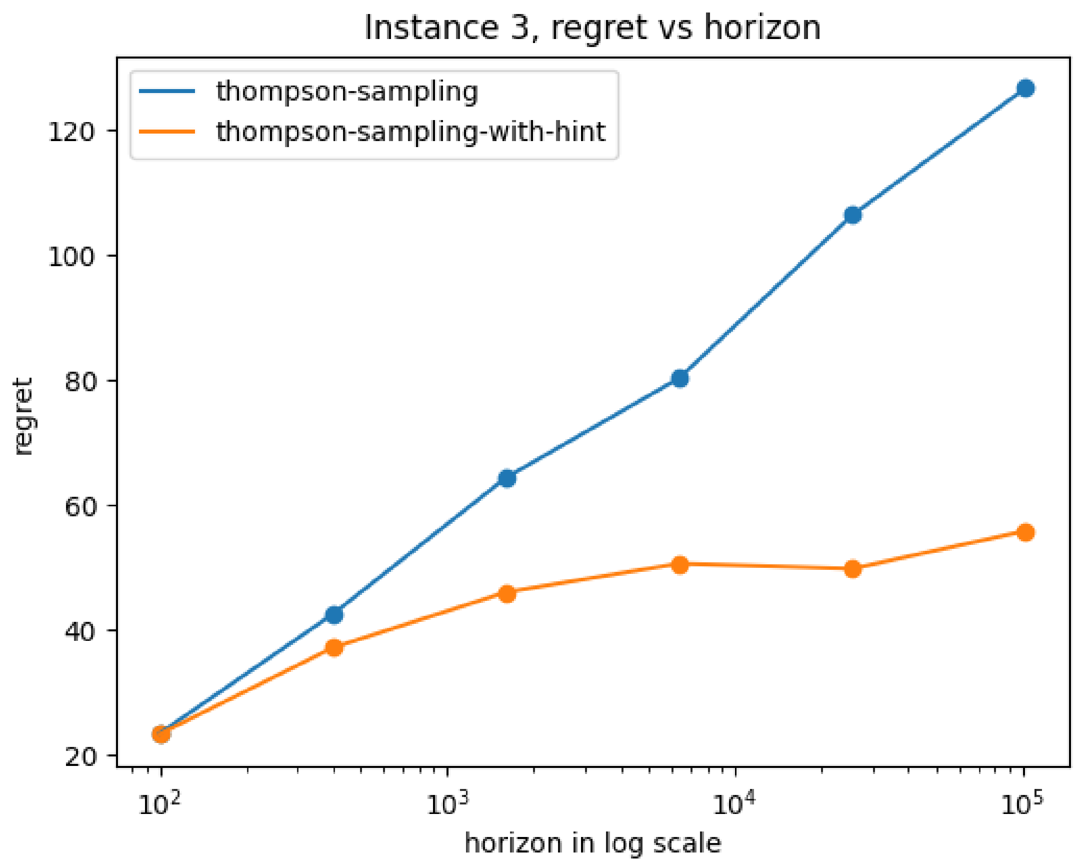
Plots for Task 2 (T2) Instance 1



Plots for Task 2 (T2) Instance 2



Plots for Task 2 (T2) Instance 3



Plots explanation for Task 2 (T2)

- The results are very pleasing as with a slight hint advantage to the thompson sampling algorithm we tremendously improved our regret
- since the assignment limited horizon to 102400 which is not large enough to check for asymptotic performance for the thompson-with-hint algorithm, So just to make sure my implemented algorithm is asymptotically better than traditional thompson sampling i verified the results for 1 million and 10 million horizon values which indicated thompson sampling with hint is better for large horizon also

Epsilon values for Task 3 (T3)

For all 3 instances $\epsilon - 1 = 0.00$, $\epsilon - 2 = 0.02$ and $\epsilon - 3 = 0.04$ are such that regret at $\epsilon - 2$ is less compared to $\epsilon - 1$ and $\epsilon - 3$