CS 747: Programming Assignment 1

Name: Ansh Khurana
Roll Number 170050035

Assumptions:

1. Number of initial pulls to start the algorithms: INIT_PULLS global variable (set to 3) for each arm - I have not done such initial pulls in thompson-sampling-with-hint which starts with a uniform distribution over arms.
2. Tie breaking: np.argmax does automatic tie breaking
3. randomization: using np.random.RandomState(args.randomSeed)
4. All plots are on log scale ('x' axis)

Choice of hyperparameters:

INIT_PULLS = 3
rng = np.random.RandomState(args.randomSeed)
KL_UCB_HYP = 3
KL_PRECISION = 1e-6

T1 & T2:

Observations:

Note that this number can be negative (and might especially turn up so on small horizons—why?).

Ans:

maximum **expected** cumulative reward possible is under expectation and is not the maximum reward ever possible. This maximum reward is equal to the horizon but has a low probability of occurrence. Even if prob is low, it might happen for some seed values and shorter horizons that we obtain a reward that is > p^* * horizon.

Implementation details:

`class global_arms:`

a global class that pulls the arm and returns the reward given index of the arm while hides the true p’s from the algorithms.

initial-pulls - pull in round robin manner until INIT_PULLS*num_arms pulls happen or you exhaust the horizon in just initial pulls.

epsilon-greedy - implemented in eps_greedy() function which upon explore pulls rng.choice() arm and on exploit pulls arm with best empirical mean.

ucb - implemented in ucb() - pulls the arm which has highest UCB value (returned by get_ucb()) function

kl-ucb - With c set to 3 (KL_UCB_HYP), we try to find the max q value that satisfies the property using binary search from p_hat (empirical mean) to 1. Search terminates within precision of KL_PRECISION global variable. The interval of search (upper_bound-lower_bound) < KL_PRECISION. Arm with highest q is pulled.

thompson-sampling - Using rng.beta() to get required samples from Beta(s+1, f+1) and then pulls arm with highest sample.

thompson-sampling-with-hint

Following the theoretical justification of the thompson-sampling algorithm given in lecture 3, we observe how in thompson-sampling a belief of the world (W) is represented using the beta distribution where the true means are distributed over [0,1] with probability given by Beta(s+1, f+1) distribution for an arm. (where s=#successes, f=#failures)

Following this motivation, for Thompson Sampling with Hint we model the Belief in the following way -

The true probabilities of the arms are known (upto a permutation).

Let the number of arms be ‘n’, each arm has it’s underlying probability as one of the true sorted probabilities.

let sorted probs be given by **TP = [tp_1, tp_2,.... tp_n]** and arm probs are given by **AP = [ap_1, ap_2,.... ap_n].**

For every arm, we model a discrete PMF over the true probability values to represent our belief. Thus the belief distribution would be a nxn matrix of probability where $P_{i,j}$ = prob(ap_i=tp_j).

We initially assume that every arm has parameter ap uniformly distributed over the true sorted probabilities ($P_{i,j}$ = 1/n).

Deciding which arm to pull:

Given the current belief matrix, at time t:

The arm which has the maximum probability according to the belief of being the best arm is pulled. That is the arm which has highest probability value in the last column of the belief probability matrix.

After pulling the arm, it's row will be updated using the update rule given below:

We update the belief using Bayes rule:

Say at time t, if arm a_i is pulled,

If we get reward of 1:

$$\text{Posterior}(ap_i=tp_j \mid r = 1) \propto P(r=1 \mid ap_i=tp_j) * \text{prior}(ap_i=tp_j)$$

$$\text{Posterior}(ap_i=tp_j \mid r = 1) \propto tp_j * \text{prior}(ap_i=tp_j)$$

If we get reward of 0:

$$\text{Posterior}(ap_i=tp_j \mid r = 0) \propto P(r=0 \mid ap_i=tp_j) * \text{prior}(ap_i=tp_j)$$

$$\text{Posterior}(ap_i=tp_j \mid r = 0) \propto (1- tp_j) * \text{prior}(ap_i=tp_j)$$

We calculate this product for each j and then normalize it to get the updated belief which is used in the next iteration. Thus at every time step t, one row of the belief matrix will be updated.

The implementation can be found in the thompson_discrete() function where prob_matrix is being updated as specified above. The algorithm is derived theoretically from the thompson-sampling algorithm with a discrete space of probabilities each arm can take from TP.

T3:

Observation: Since epsilon controls the explore-exploit tradeoff, having very low and very high values of epsilon would be adverse and are likely to perform worse. A value in the middle is suitable for having good exploration and exploitation. High epsilon -> very low exploitation, low epsilon -> very low exploration.

For instance i-1.txt

Epsilon	Regret
0.000001	1661.22
0.02	377.92
0.9	18456.9

For instance i-2.txt

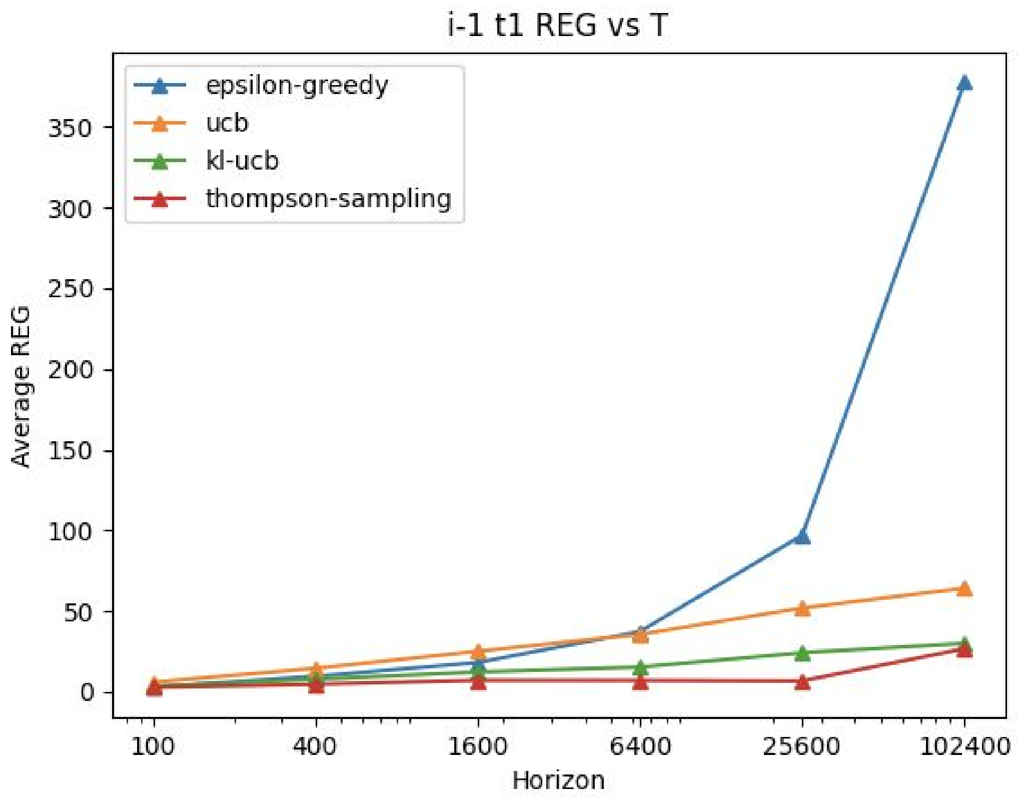
Epsilon	Regret
0.000001	3912.06
0.02	560.22
0.9	18416.7

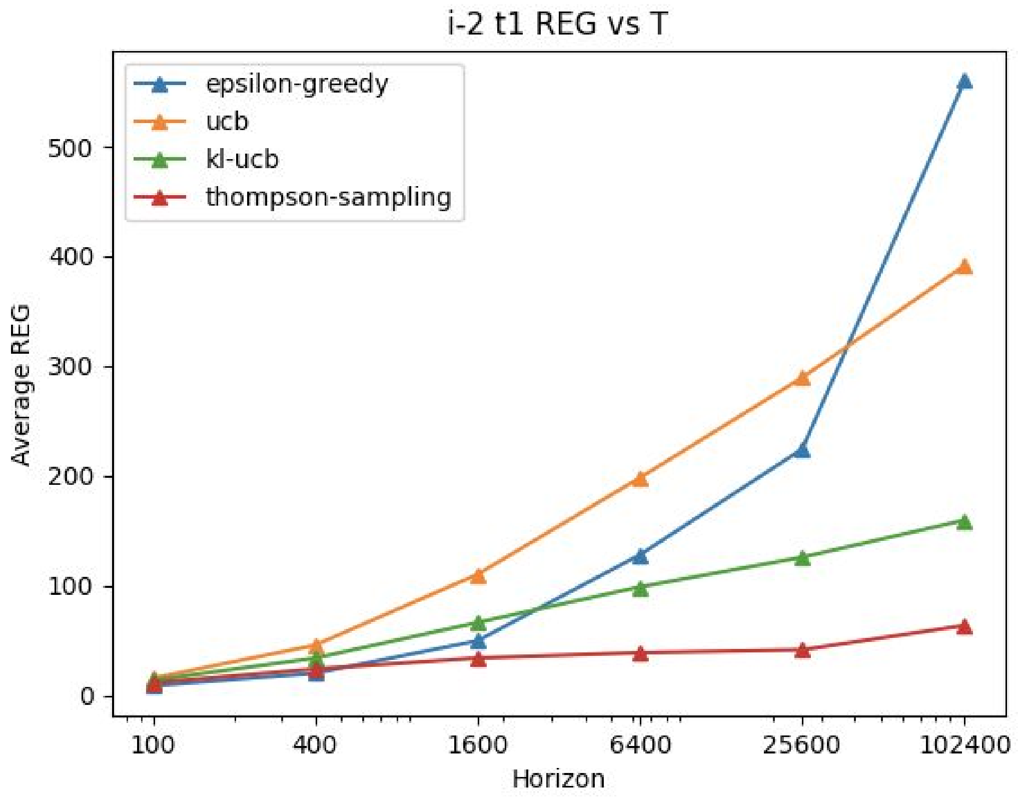
For instance i-3.txt

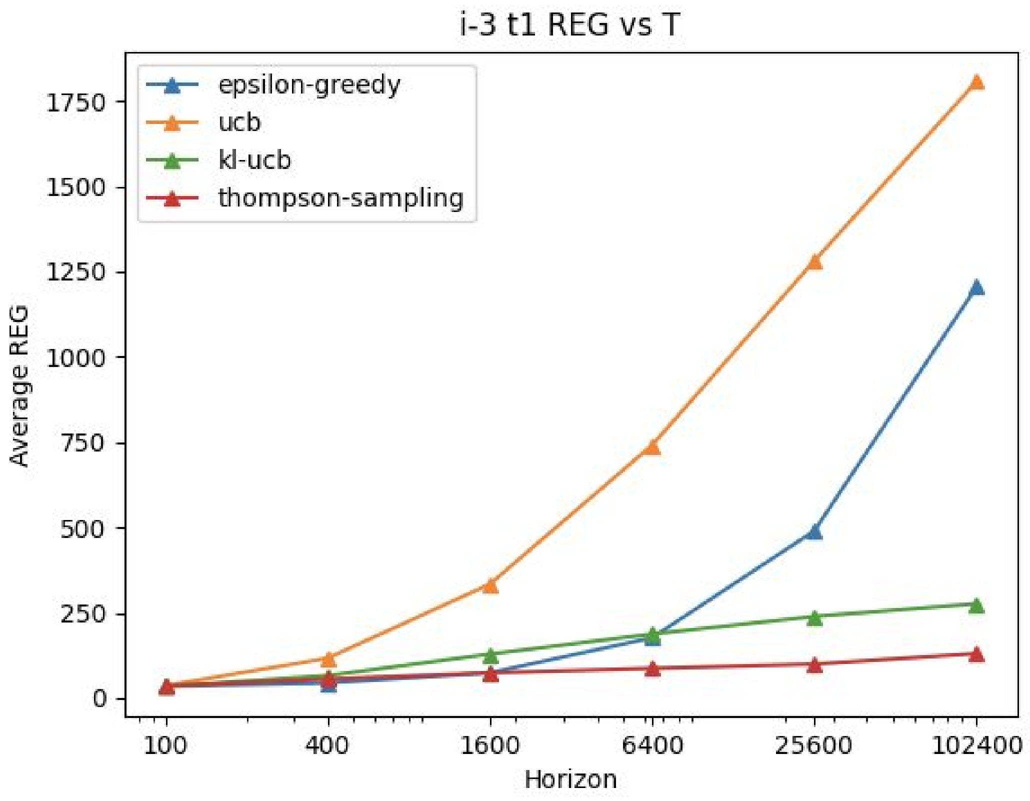
Epsilon	Regret
0.000001	1344.5
0.02	1205.58
0.9	41896.9

The above are examples of the cases when epsilon values $ε1 < ε2 < ε3$ for each instance such the regret of ε2 is less compared to the other two.

T4: Plots and Observations







T1 - epsilon-greedy, ucb, kl-ucb, thompson-sampling

Here we can see that for shorter horizon values the difference between the performance of each of the algorithms is quite low. However, one can see that the algorithms perform as expected in theory where plots of ucb, kl-ucb, thompson-sampling are linear on the log(T) scale while the eps-greedy algorithms looks to have an exponential plot on the log(T) scale. This provides confirmation to the results that eps-greedy algorithm is not sublinear while ucb, kl-ucb, thompson-sampling attain the Lai and Robbins bound asymptotically and their regret is O(log(T)). Further, ucb has a higher constant than kl-ucb in the bound.

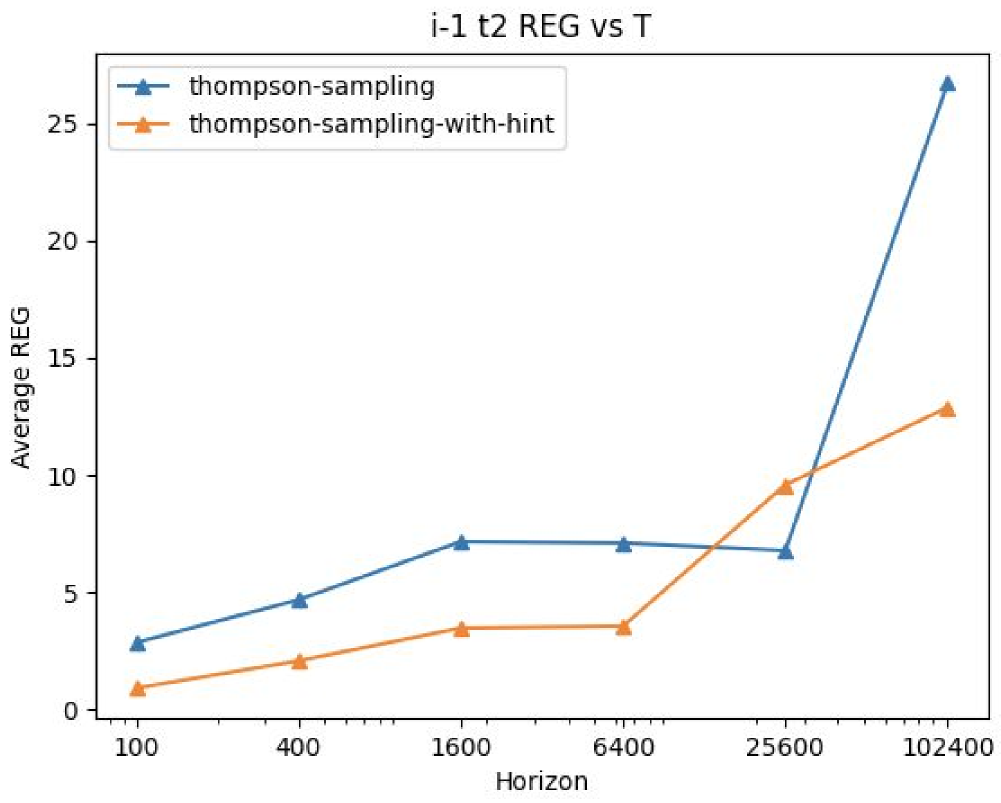
General trend in performance on the basis of (lower) average regret:

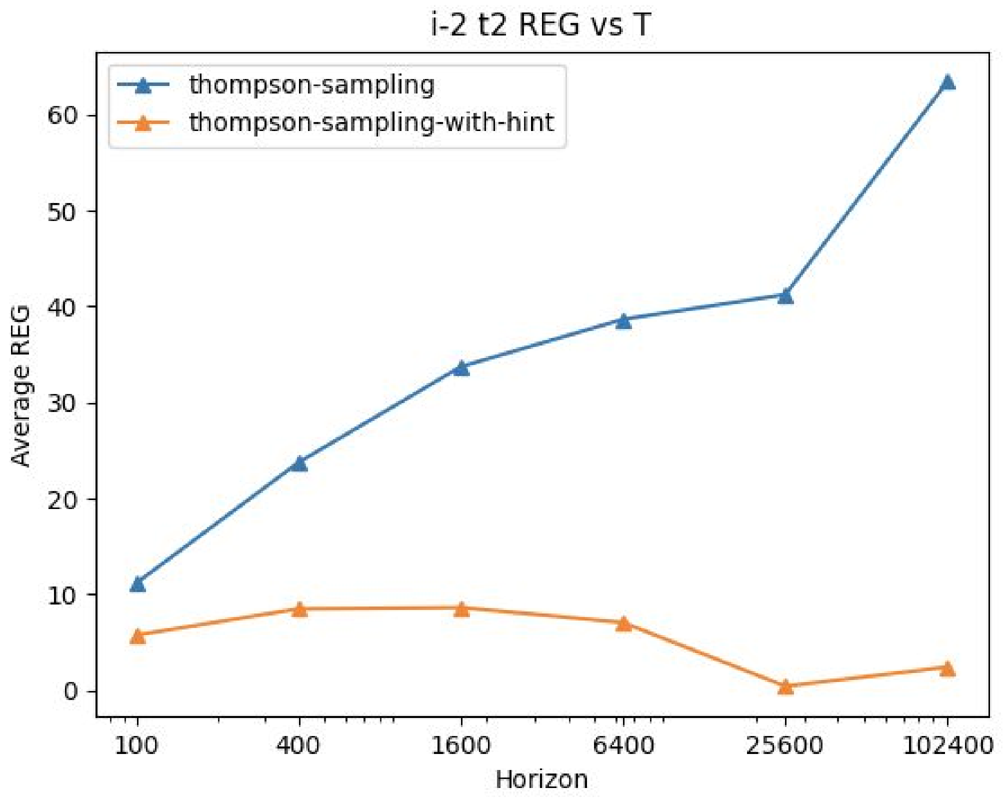
thompson-sampling > kl-ucb > ucb > epsilon-greedy

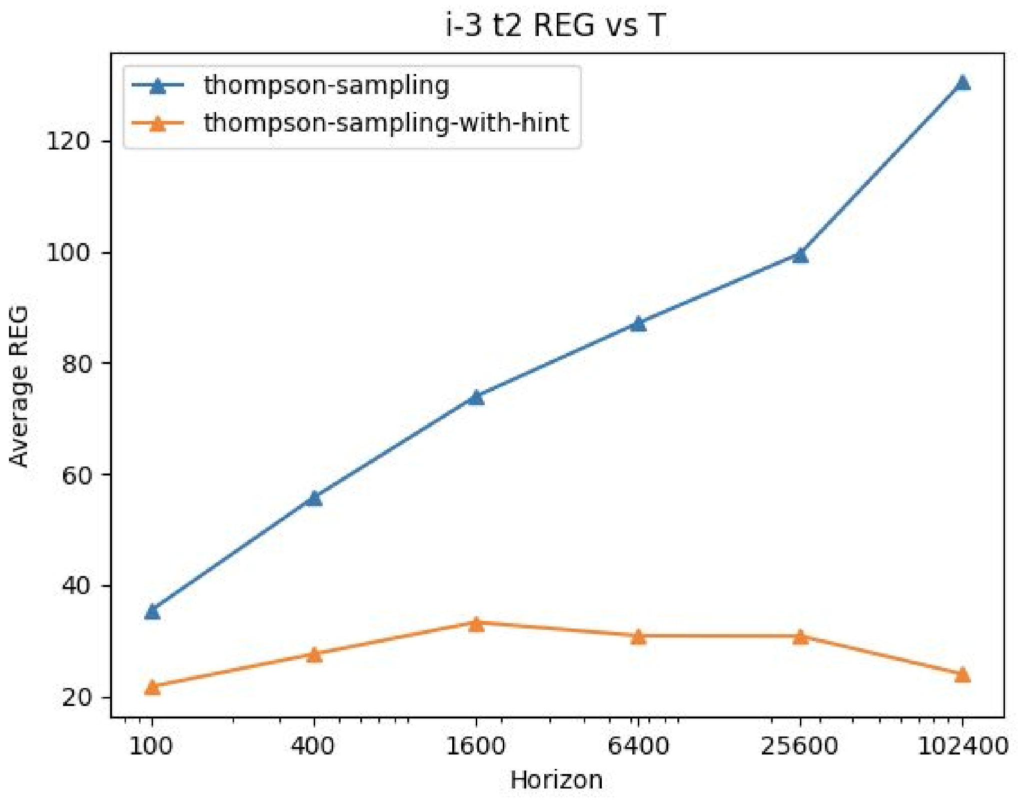
For some lower horizon values it can be the case that a “worse” algorithm in the above trend attains lower regret than a better one.

Surprisingly, for instance i-3 ucb is returning worse regret than epsilon-greedy algorithm. However, the plot for ucb algorithm still has linear slope while epsilon-greedy grows exponentially and will cross ucb at a larger horizon (see how it crosses in i-2). The slope trends still conform to the theoretical expectations. High constant even with log(T) bound might be causing this higher regret value.

T2: thompson-sampling vs thompson-sampling-with-hint




i-2 t2 REG vs T
thompson-sampling
thompson-sampling-with-hint
Average REG
Horizon
0
10
20
30
40
50
60
100
400
1600
6400
25600
102400


i-3 t2 REG vs T
thompson-sampling
thompson-sampling-with-hint
Average REG
Horizon
20
40
60
80
100
120
100
400
1600
6400
25600
102400

As expected, since in the thompson-sampling-with-hint algorithm we have the knowledge of the true probabilites (upto a permutation), it performs better than the regular thompson-sampling algorithm. For thompson-sampling algorithm, we needed to model a distribution over [0,1] domain and with the hints, we utilize the given information effectively to have a discrete distrubution over the true probabilities. Thus, we obtain the advantage in performance.

For the instance i-1 which had only 2 arms, there is some deviation at horizon=25600. This can be seen as an experimental anomaly. This happened only for the 2 arm instance (here the advantages over normal thompson-sampling are also the lowest). It could be the case that since we have to select only between one of the 2 arms (whose underlying true probabilities differ by a lot 0.4 / 0.8), thompson-sampling even without hint gets to a confident distribution based on s+1, f+1 and reaches close to the performance with hints on this very simple instance.