Pricing-Advertising project

Online learning application – Professor Nicola Gatti

Arrigoni Francesca, Enrico Brunetti, Stefano Ferrara,
Davide Gesualdi, Stefano Vighini

Step 0

We want to find an optimal pricing and advertising strategy for a backpack

Prices ∈ {10€, 20€, 30€, 40€, 50€}
Bids ∈ [0€, 2€], 100 bids
T=365 giorni

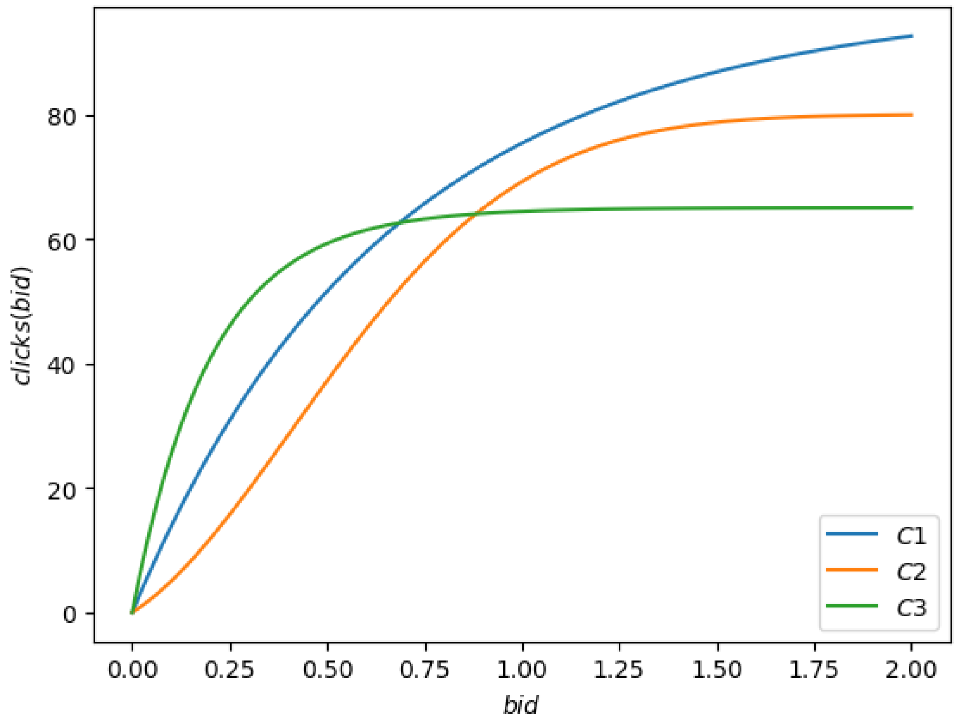
Features of the customers:

- Employment
- Residence

- C1: students
- C2: commuter workers
- C3: resident workers

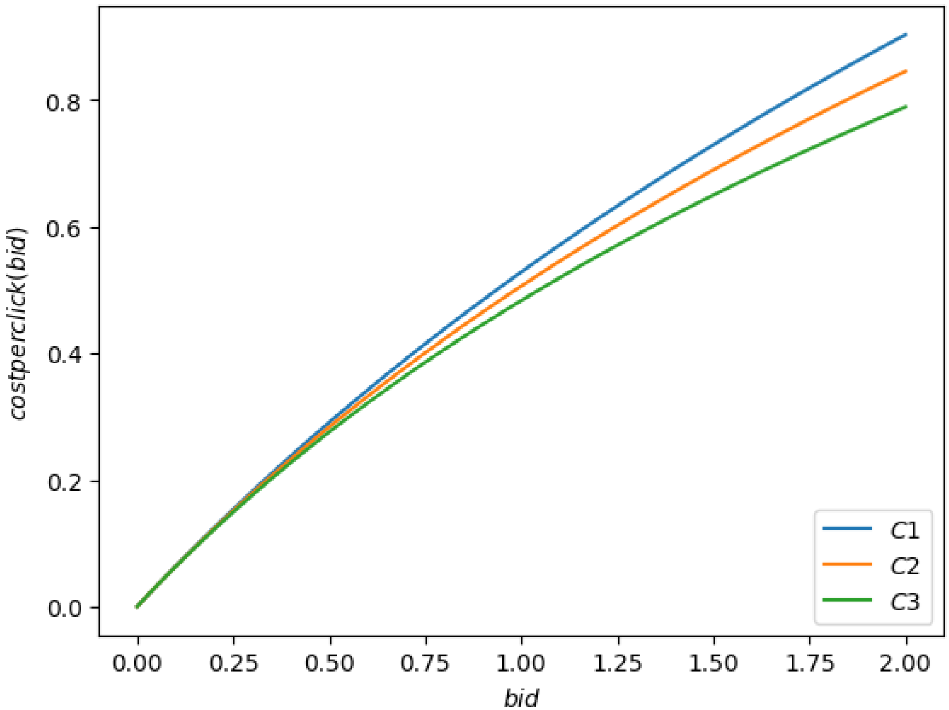
Step 0
Number of clicks

$$
\begin{cases}
100(1 - exp\{-1.5 \cdot bid - 0.5 \cdot bid^2\}) & \text{for C1} \\
80(1 - exp\{-0.5 \cdot bid - 1.5 \cdot bid^2\}) & \text{for C2} \\
65(1 - exp\{5 \cdot bid + 0.3 \cdot bid^2\}) & \text{for C3}
\end{cases}
$$

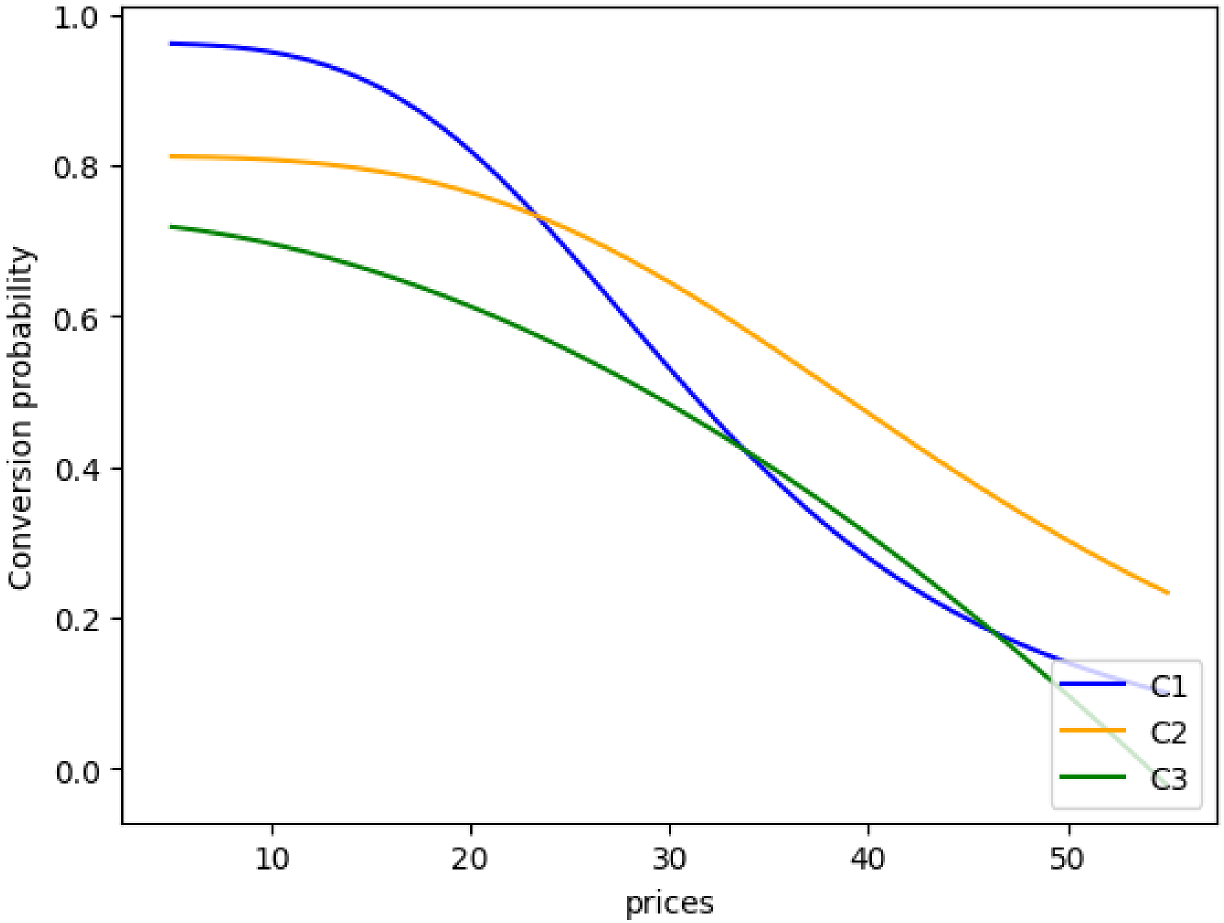


Step 0
Daily click cost

$$
\begin{cases}
1.5 \cdot 2 \cdot \log(1 + bid/2) & \text{for C1} \\
1.5 \cdot 1.6 \cdot \log(1 + bid/1.6) & \text{for C2} \\
1.5 \cdot 1.3 \cdot \log(1 + bid/1.3) & \text{for C3}
\end{cases}
$$



Step 0
Conversion rate



Step 0
Clairvoyant algorithm

Pricing problem:

$$p = \text{price chosen}$$
$$c = 8 = \text{cost of production of one backpack}$$
$$conv = \text{conversion rate function}$$
$$gain(p) = (p - c) \cdot conv(p)$$

Advertising problem:

$$bp = \text{best price found maximizing the gain}$$
$$c_{ad} = \text{cost per click of the ad}$$
$$n_{ad} = \text{number of clicks}$$
$$reward(bid) = n_{ad}(bid) \cdot (gain(bp) - c_{ad}(bid))$$

Step 1 : Pricing

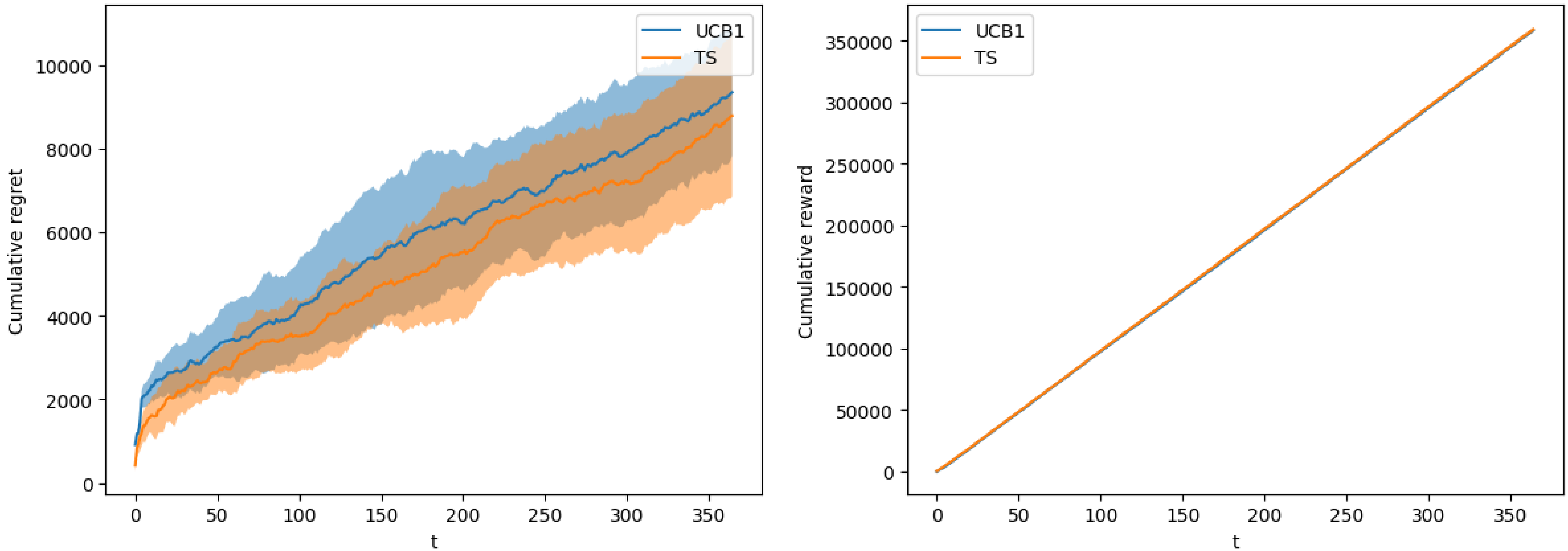
- All customers belong to C1
- The bid is fixed at 2€ for each ad
- The number of customers that enter the website each day is 92 people

We employ two different algorithm to maximise the reward:

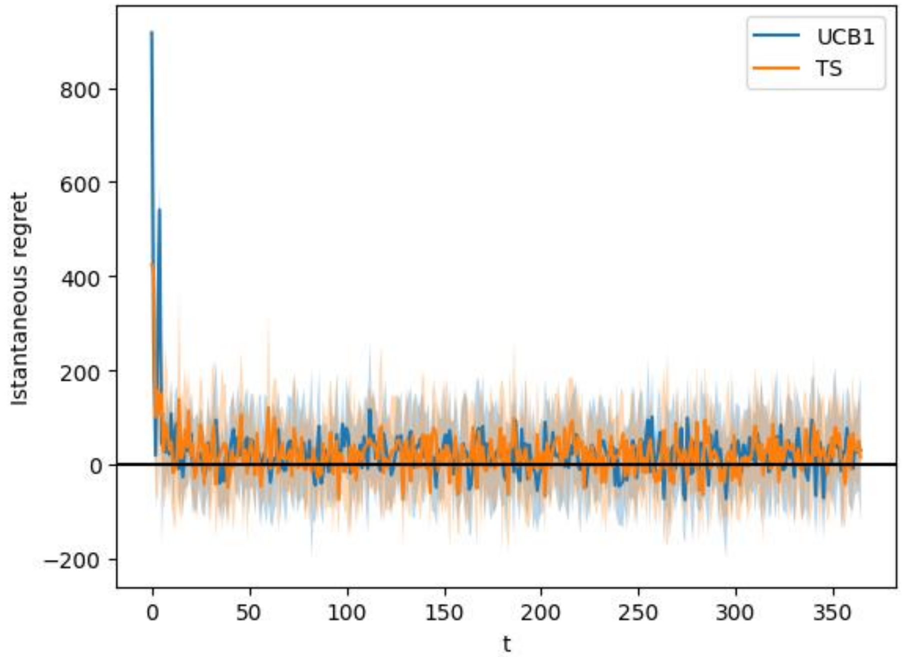
- UCB-1: $$ucb = \bar{x} + \sqrt{\frac{2 \cdot \log(t)}{n_{a_t}(t-1)}}$$

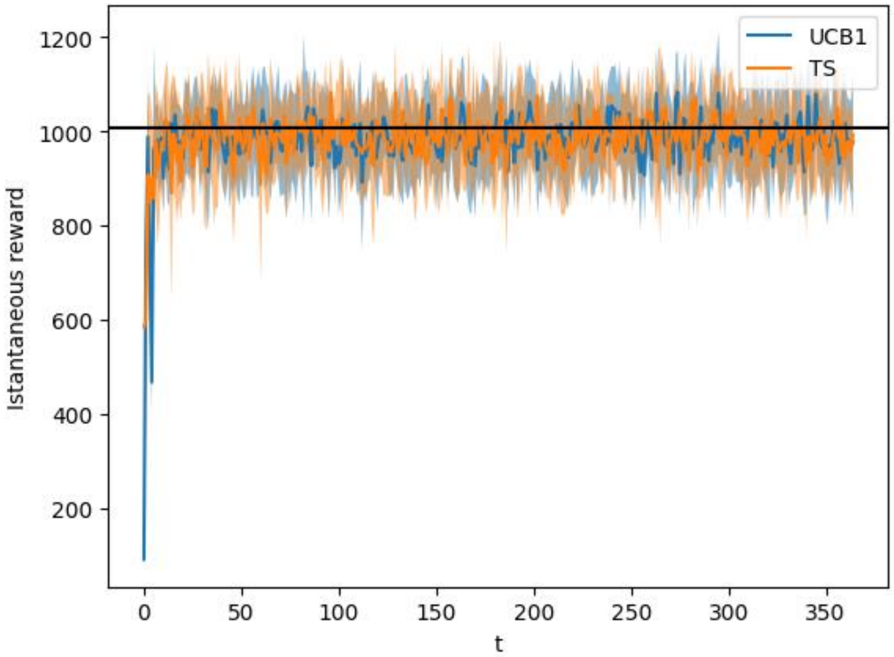
- Thompson Sampling: $$(\alpha_{a_{t+1}}, \beta_{a_{t+1}}) \longleftarrow (\alpha_{a_t}, \beta_{a_t}) + (s_t, f_t)$$

Step 1 : Results



Step 1 : Results





Step 2 : Advertising

- All customers belong to C1
- The price is fixed at 30€

We employ two Gaussian Processes algorithm to maximize the reward:

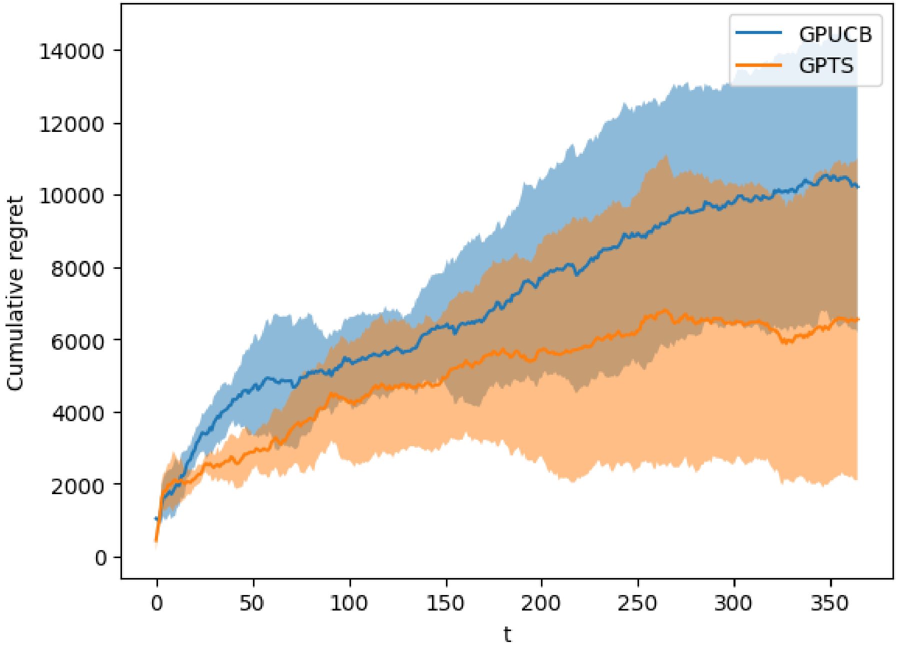
- GPUCB:

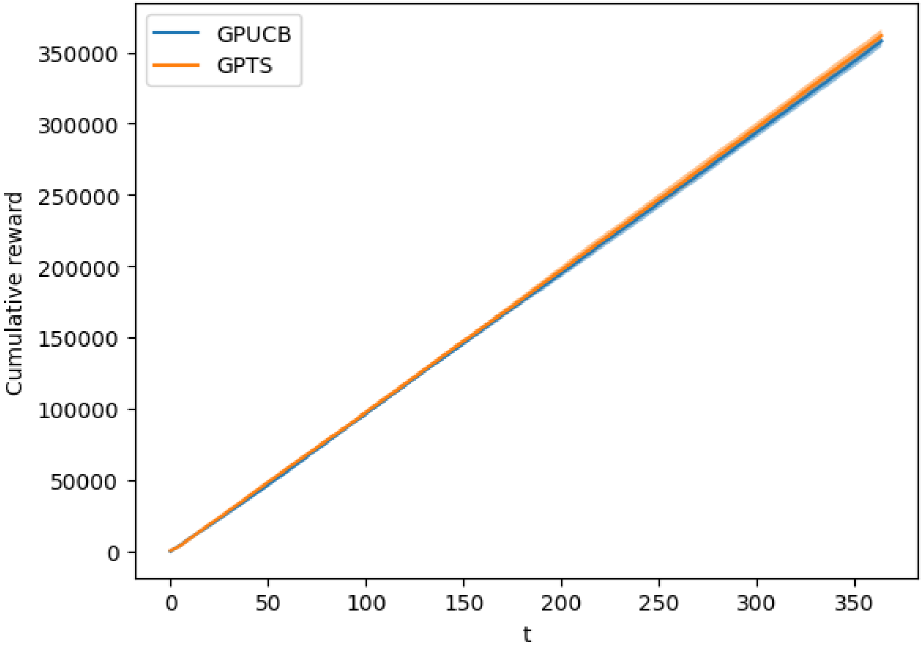
$$ucb = \bar{x} + \sigma \cdot \sqrt{\beta}$$

$$\beta = 2\log(\frac{N \cdot t^2 \cdot \pi^2}{6 \cdot \delta})$$

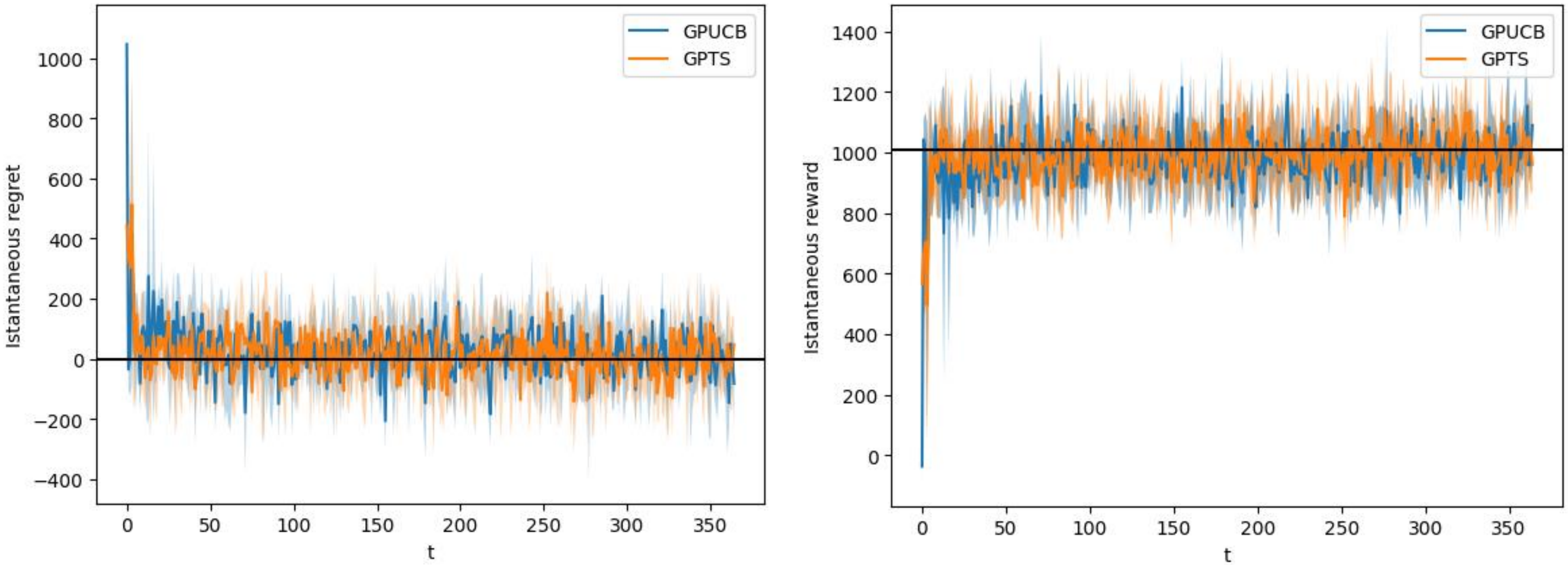
- GPTS: The posterior distribution of the Gaussian process is distributed as a Gaussian in which we can update the mean and standard deviation

Step 2 : Results



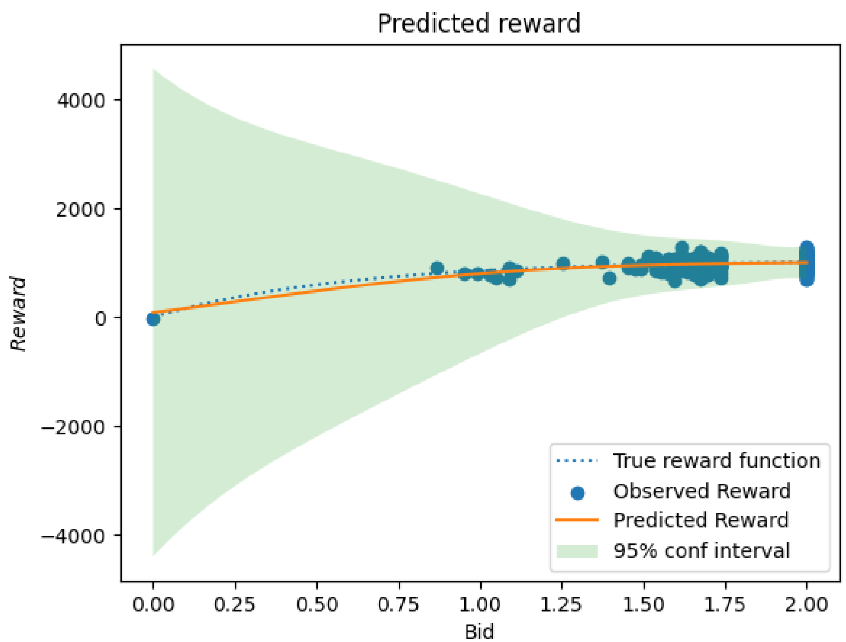


Step 2 : Results

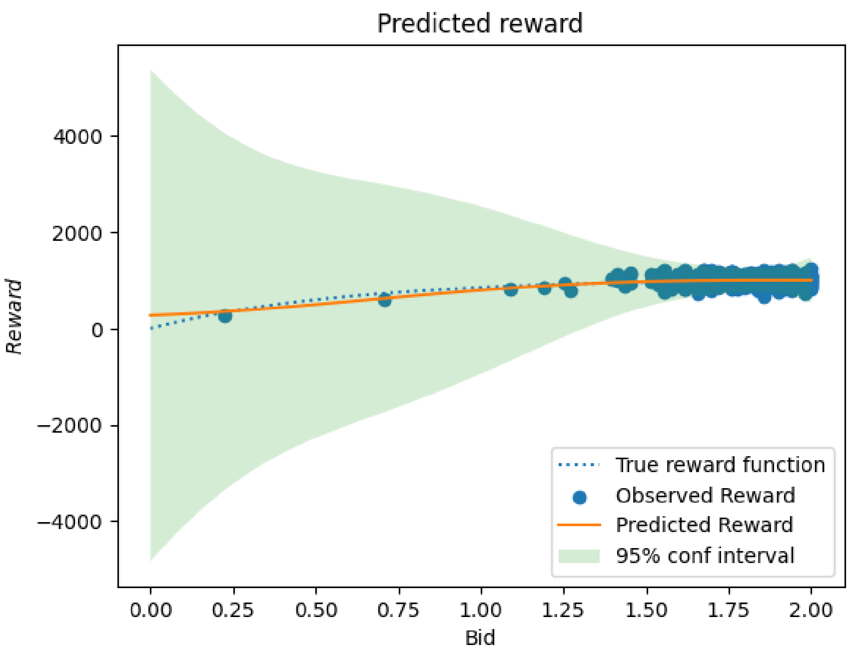


Step 2 : Results

GPUCB



GPTS



Step 3 : Pricing + Advertising

- All customers belong to C1

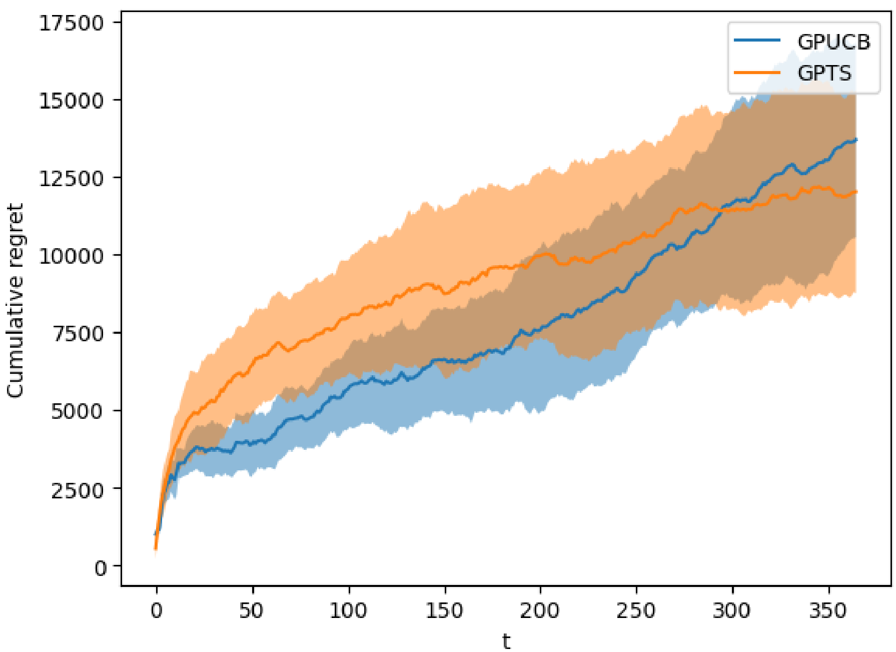
We employ a single environment that first samples from the advertising curves and uses the results to sample from the pricing curve.

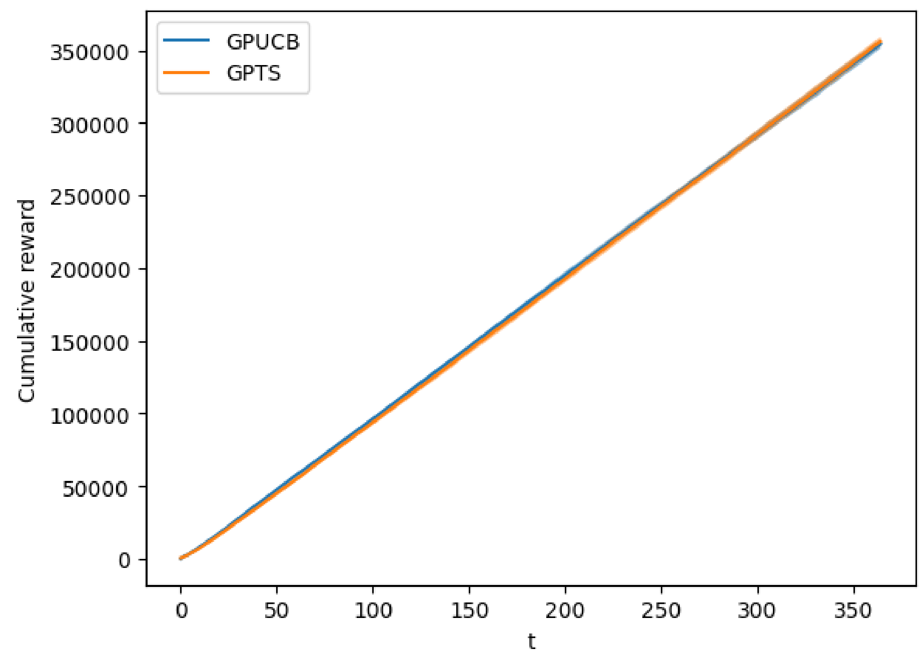
The algorithms employed are:

- UCB-1 for the pricing part
  GPUCB for the advertising part

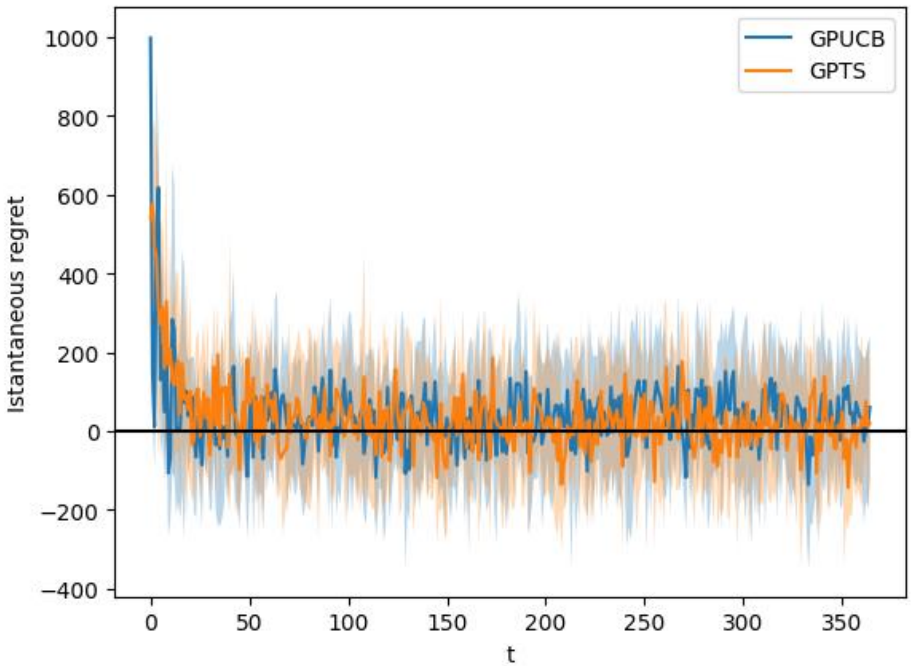
- Thompson Sampling for the pricing part
  GPTS for the advertising part

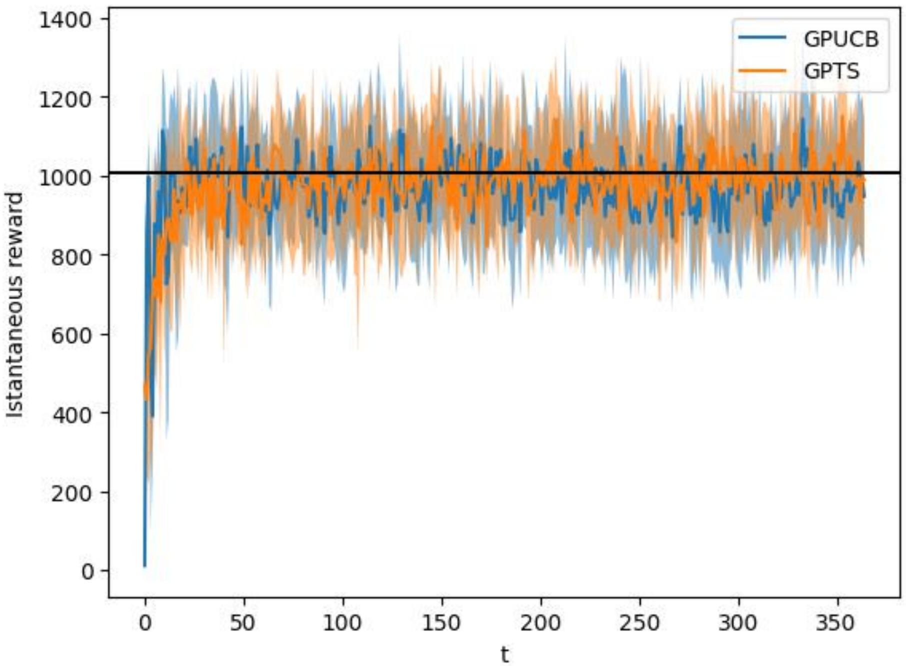
Step 3 : Results





Step 3 : Results

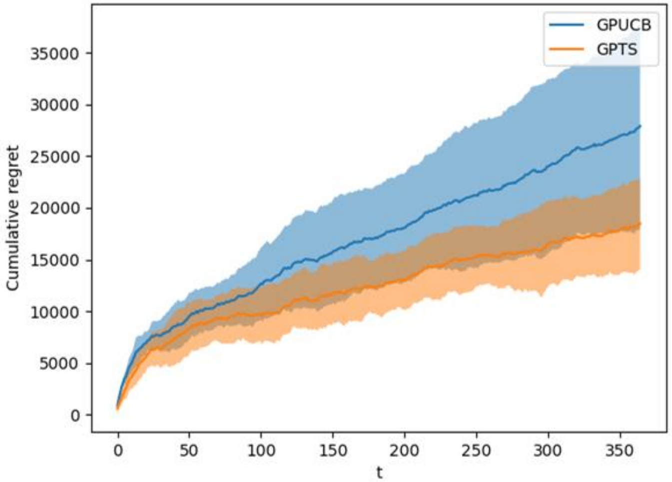




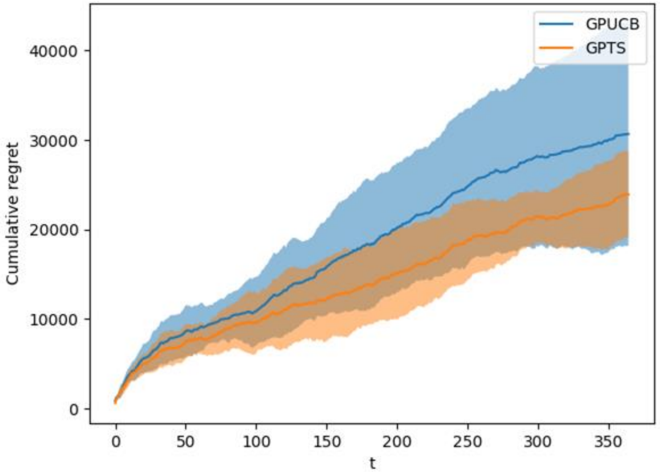
Step 4: Multiple customers

- C1, C2, and C3 get pulled with the same probability every round
- Pull feature first, then pull arm from a context
- Analize reward in 3 scenarios:
  - Not using any context
  - Using known contexts
  - Generating the context (every 14 rounds)

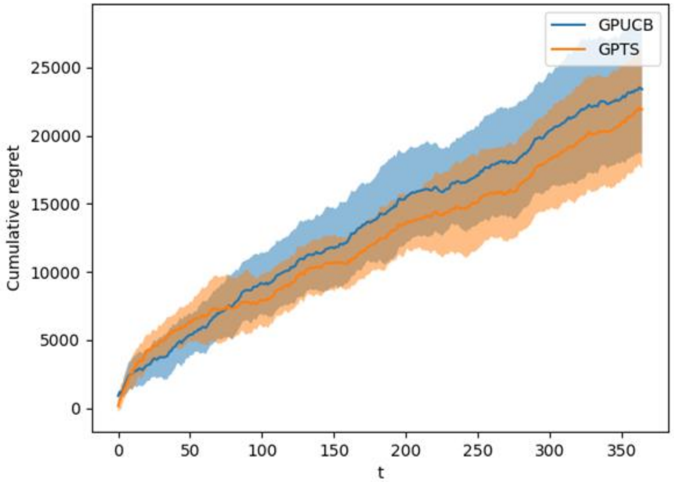
Step 4 : Results



Known Context

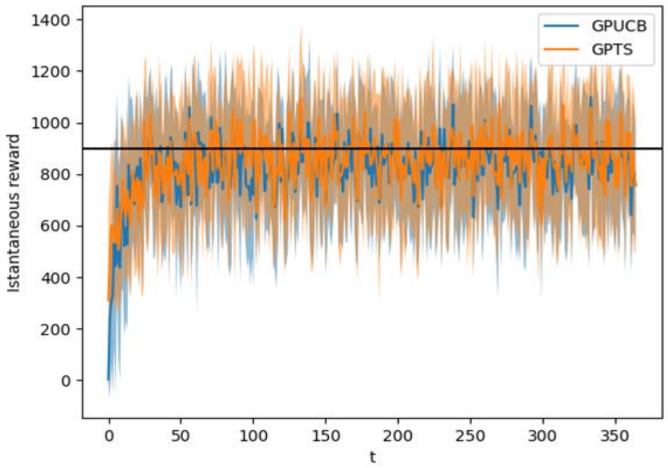


Generated Context

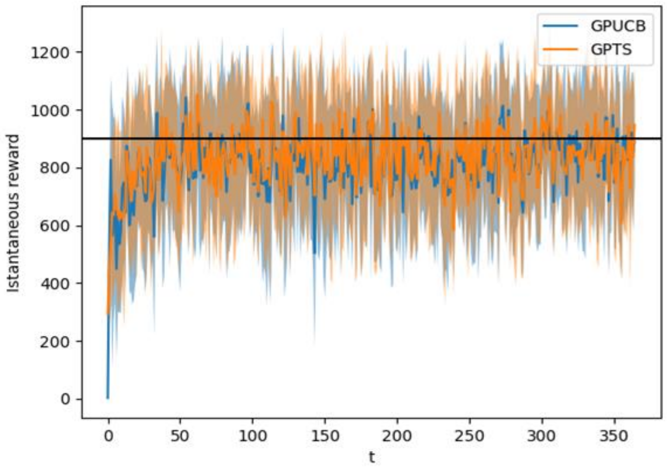


Without Context

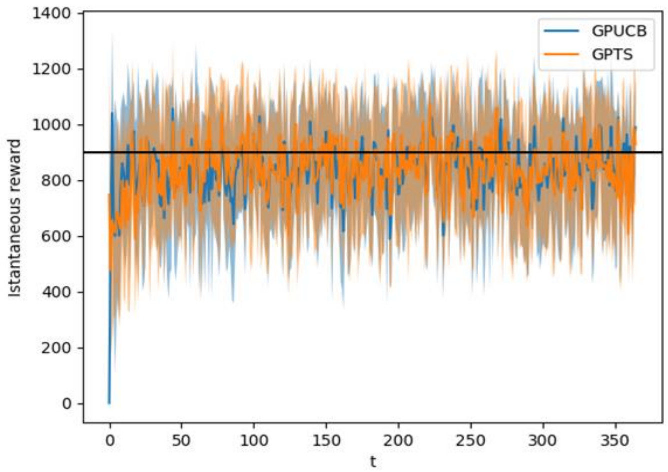
Step 4 : Results



Known Context



Generated Context



Without Context

Step 4 : Unexpected behaviour

- Convergence speed
- Reward difference between context and non context
- Not applicable, too small differences (only 30 €)

Step 5: Setting

Non Stationary(Pricing) Environment.

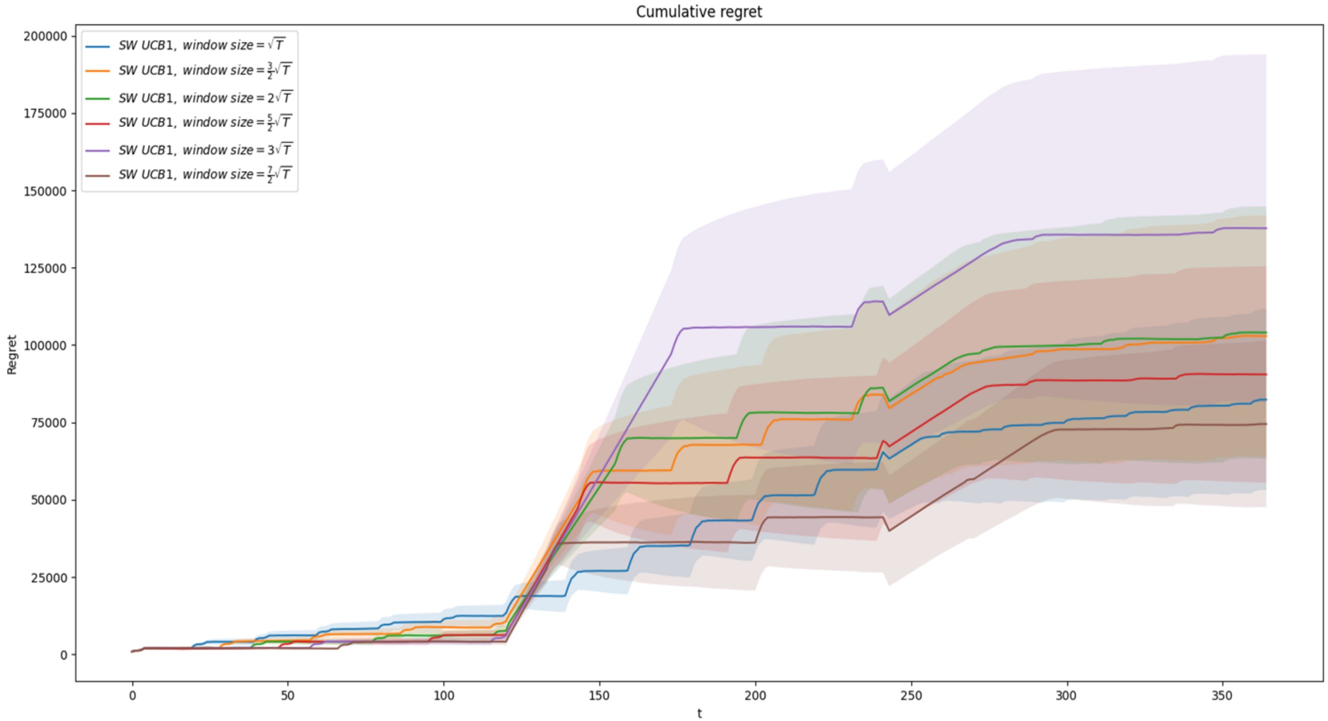
- All users belong to C1.
- 3 phases during the year:
  - July to October, students buy new backpacks for the upcoming school year → higher conversion probabilities
  - November to February, Christmas holidays, people tend to spend more for presents → higher conversion probabilities for more expensive prices
  - March to June, regular purchasing behaviour, less need for new backpacks in general → lower conversion probabilities.

Step 5 : Algorithms

- Stationary UCB1
- Sliding Window UCB1(passive approach): employs a sliding window of fixed size to store samples, computing the upper confidence bound only over those samples
- CUSUM UCB1(active approach): employs the CUSUM procedure to detect a change in the underlying distributions, dependent on four parameters(M,ε,h,α)

Step 5 : Sliding Window UCB1

From theory, $\tau \propto \sqrt{T}$, so we tried six different multiplying constants: 1, 1.5 , 2, 2.5 , 3 , 3.5.



- larger window sizes associated to lower regret
- in particular, $3.5\sqrt{T}$ is the best one

Step 5: CUSUM UCB1

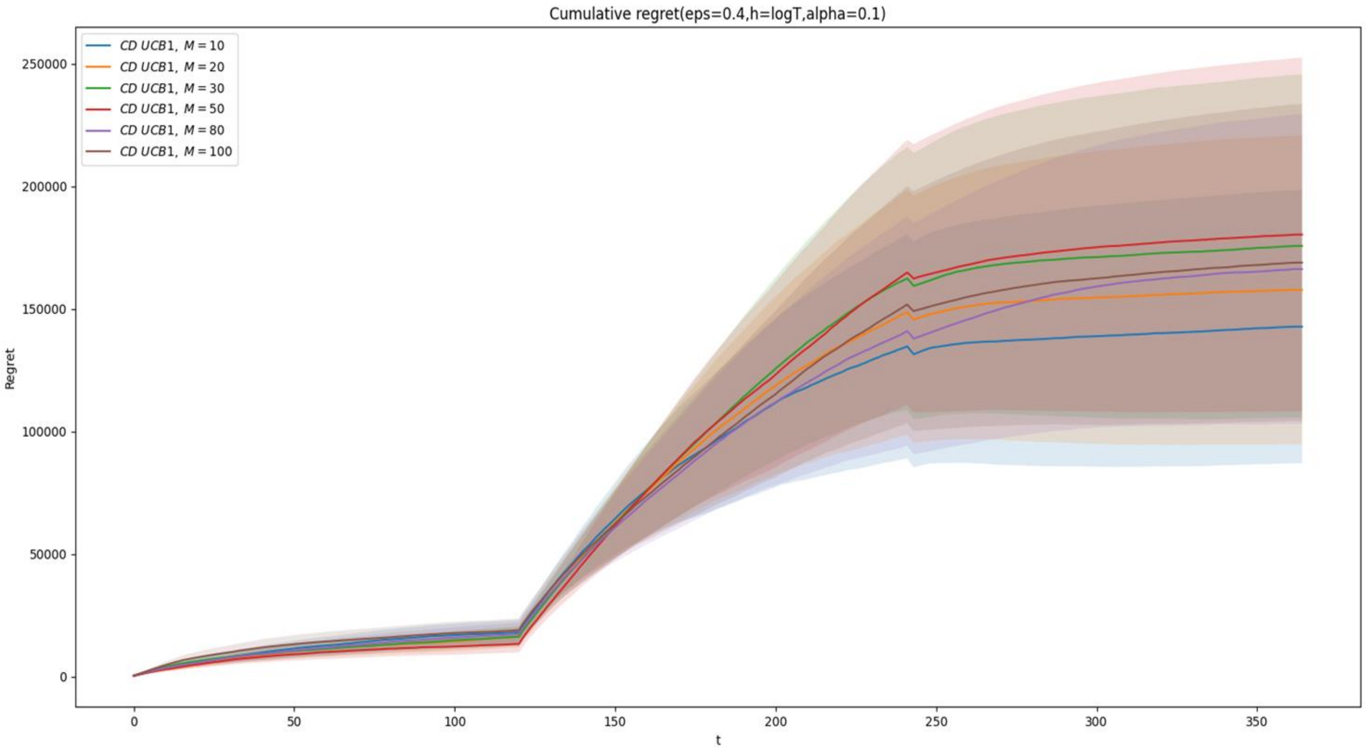
CUSUM values for every arm representing deviation from the expected reward → when a threshold is exceeded a change is detected

4 parameters to be selected:

- M, number of samples over which the mean is computed
- ε, adjusting the difference between current mean and new samples
- h, threshold to be surpassed in order to trigger a change
- α, probability of pulling a random arm at each round

Step 5: Analysis of M

Starting from fixed values for the other parameters, we tested six possible choices for M: 10,20,30,50,80,100.

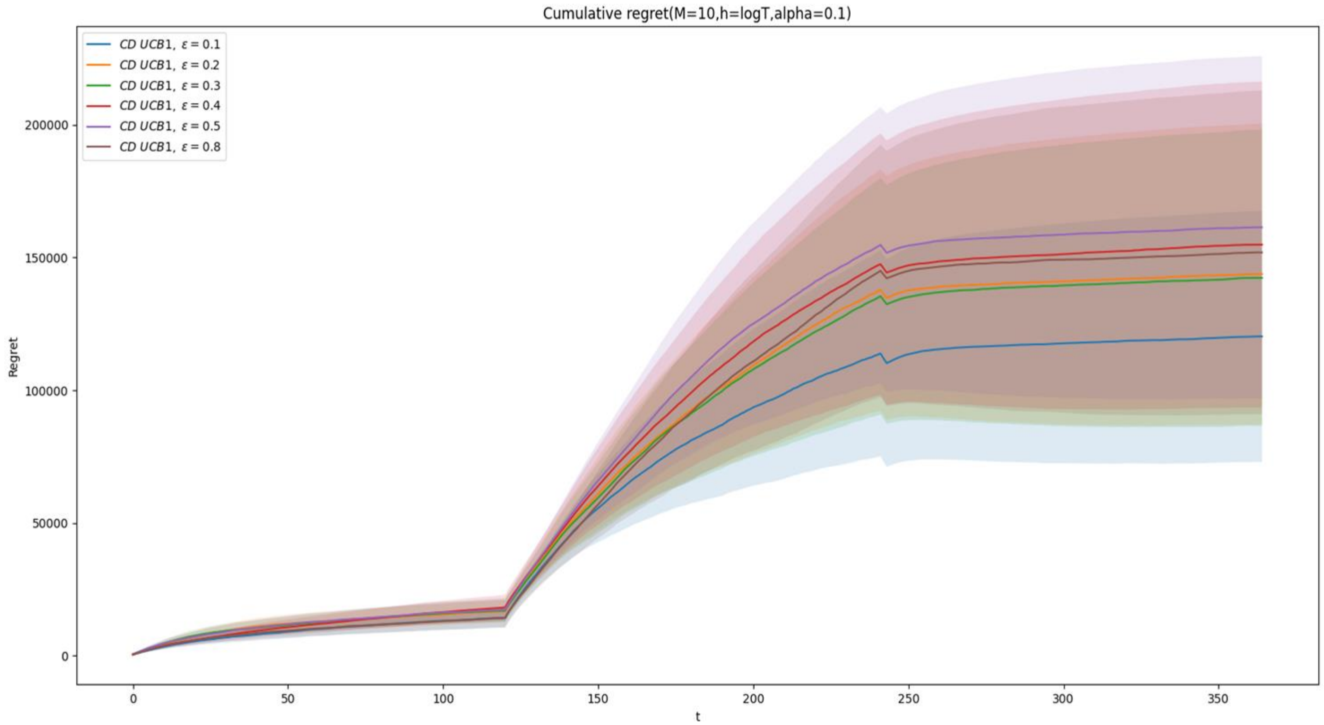


smaller window size is preferred, we set M=10.

Step 5: Analysis of ε

Normalized rewards → values in [0,1]

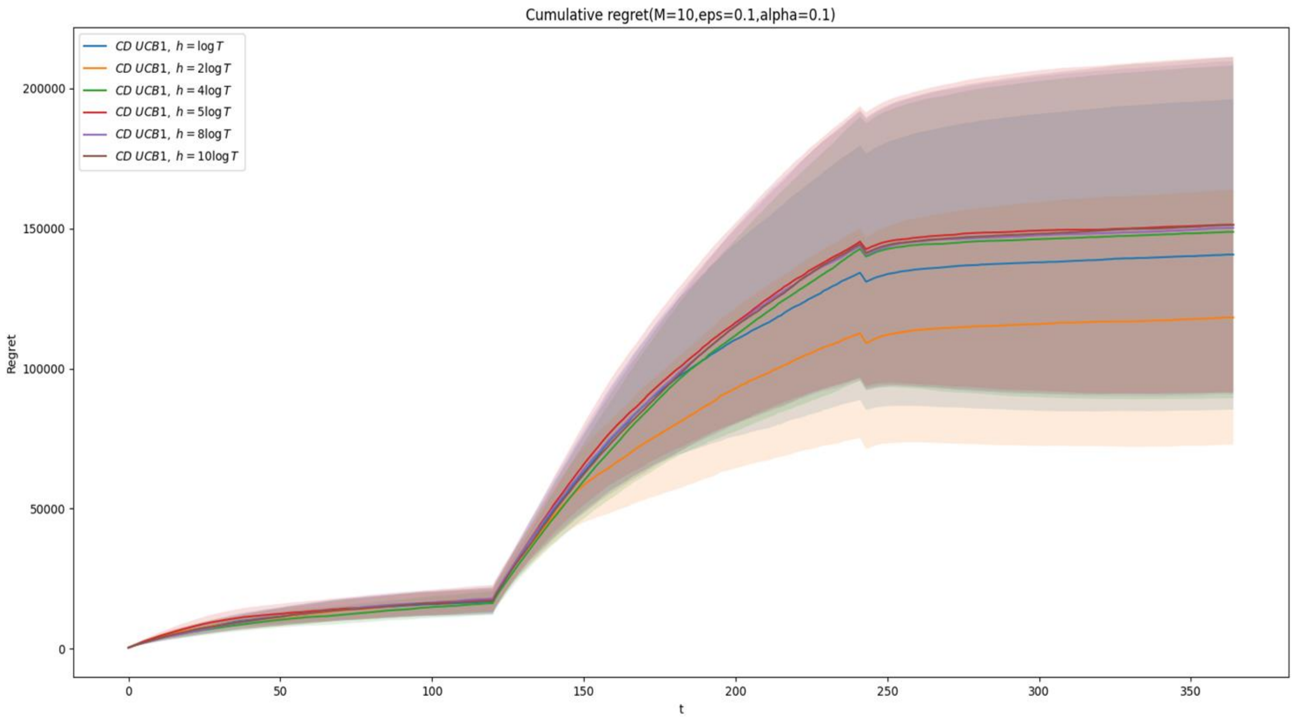
We tested six possibilities: 0.1,0.2,0.3,0.4,0.5,0.8



no clear best value, but ε=0.1 was the most frequent among multiple runs and with biggest difference in single ones.

Step 5: Analysis of h

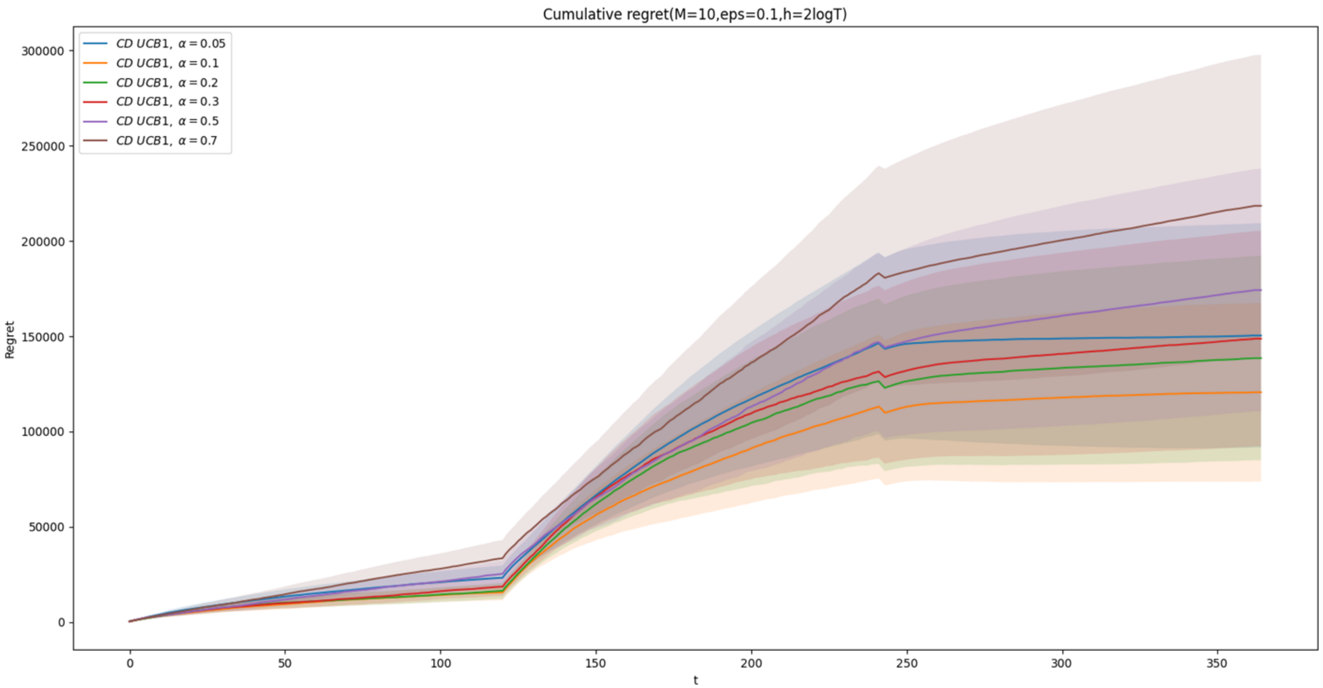
We tested different factors multiplying logT: 1,2,4,5,8,10



smaller values of the threshold achieve lower regret, in particular h= 2logT≃12 is the best choice.

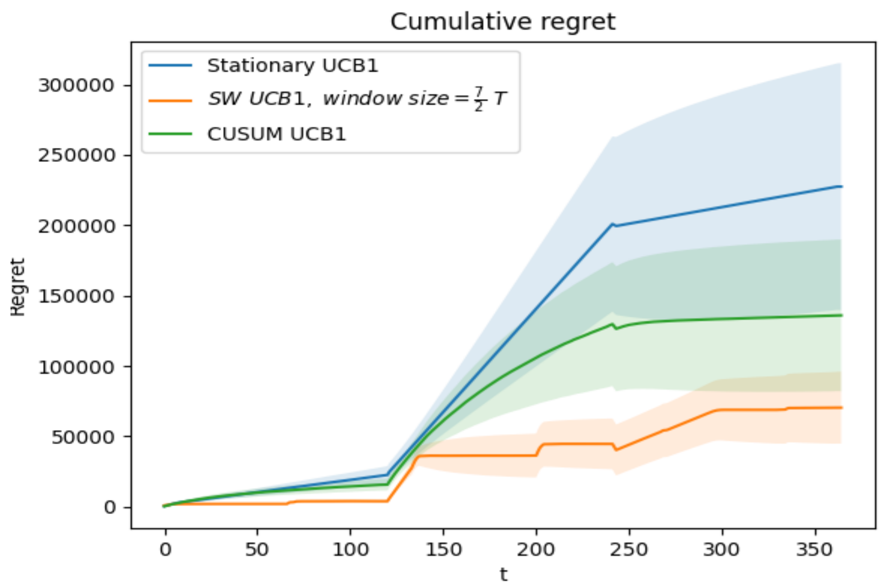
Step 5: Analysis of α

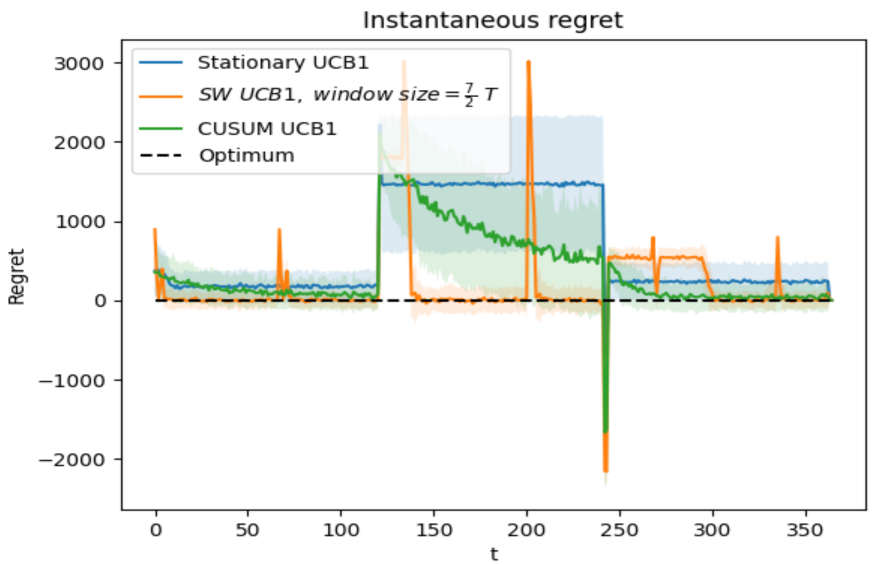
Since it's a probability we explore six values in (0,1): 0.05,0.1,0.2,0.3,0.5,0.7



smaller values around 0.1-0.2 seem like the best choice, providing a good balance between exploration and exploitation.

We compare the performances of the three algorithms in terms of regret, taking the "best" parameters obtained before for the SW-UCB and CUSUM-UCB algorithms.





Step 5: Results

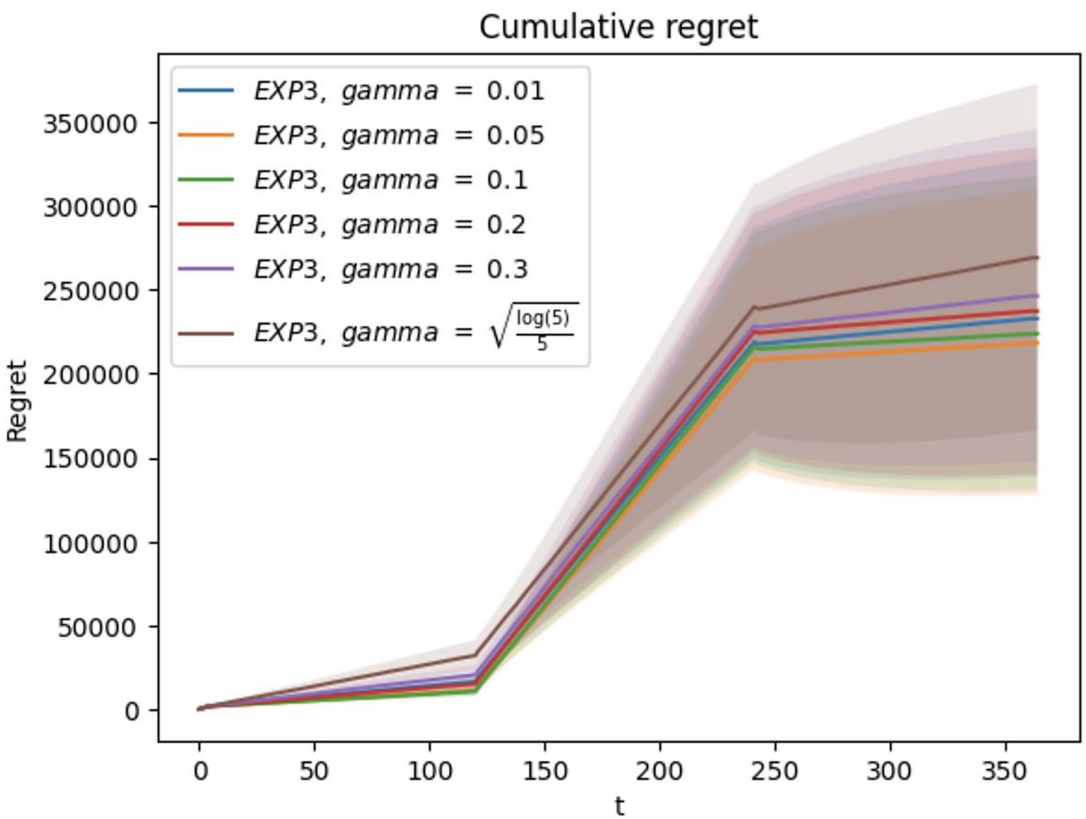
- Sublinear regret for each of the algorithms
- Worst performance is of the stationary UCB1 algorithm as expected
- SW-UCB performs better than CUSUM-UCB, reacts faster after abrupt changes
- Possible explanation due to the hard task of setting four parameters properly simultaneously.

Step 6 : Dealing with non-stationary environments with many abrupt changes

EXP3 algorithm applied in 2 different settings:

- Simplified version of step 5 with fixed bid (3 phases).
- A different non-stationary setting with a higher non-stationary degree (25 phases).
  - 5 different phases (different optimal price) repeated cyclically 5 times.

Step 6 : EXP3 algorithm



Based on importance-weighted sampling

- arms with higher rewards receive a higher importance weight

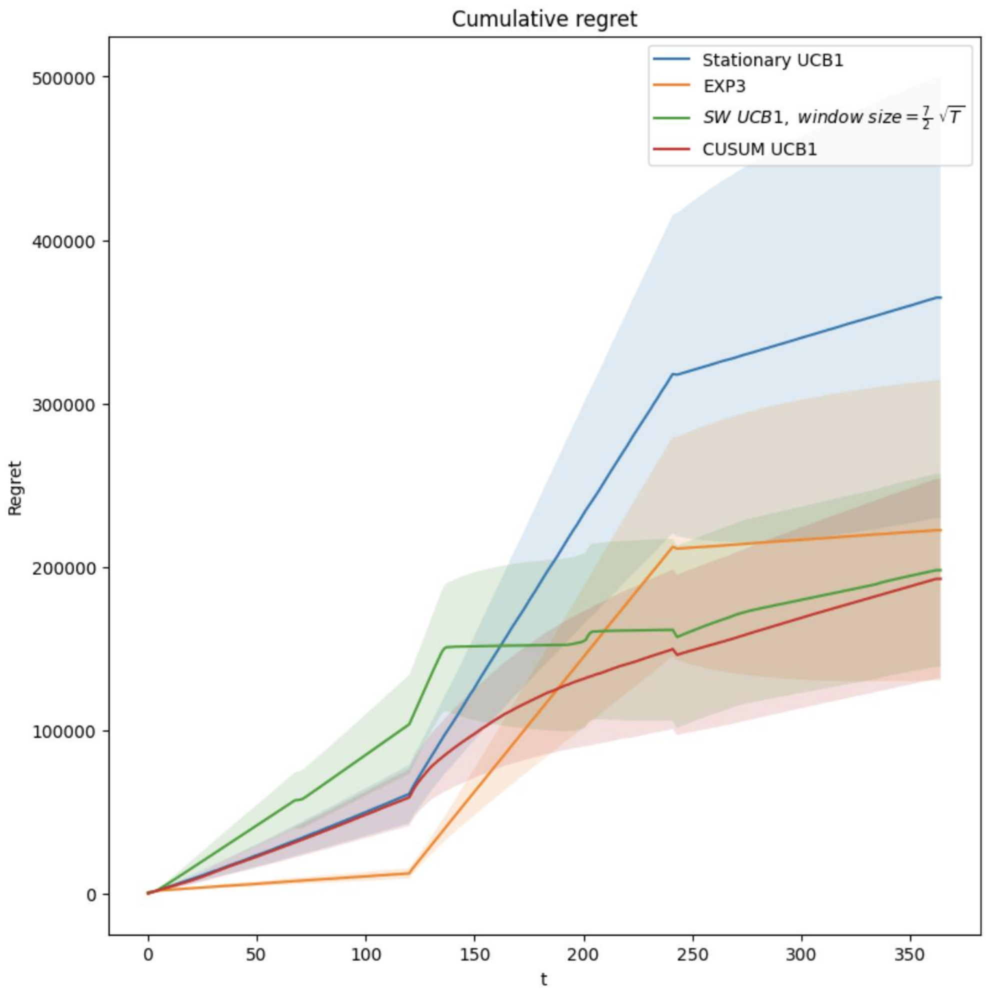
Exploration parameter $\boldsymbol{\gamma} \in [0,1]$:

- $\boldsymbol{\gamma} \to 0$: more probability according to the weights
- $\boldsymbol{\gamma} \to 1$: probability distribution over the arms tends to be uniform

Sensitivity analysis:

- Fixed $\boldsymbol{\gamma} = 0.05$

Step 6: Simplified version of step 5 setting (3 phases)



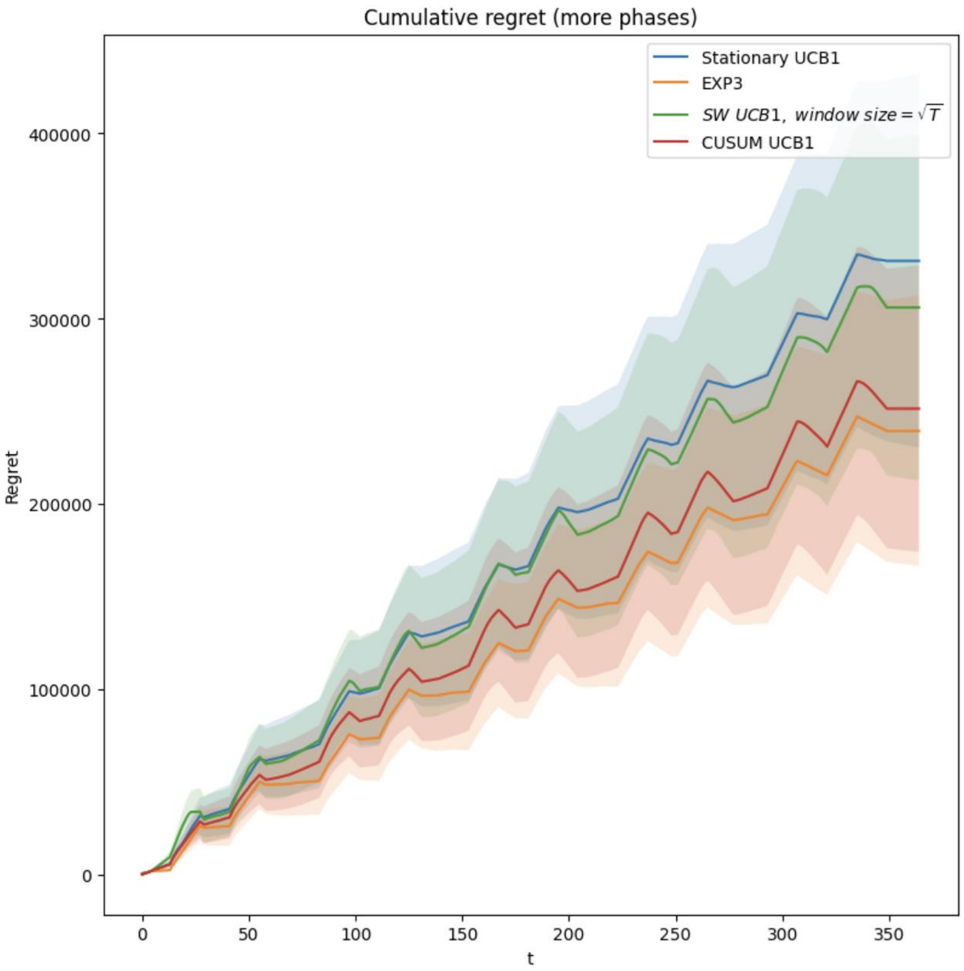
Sensitivity analysis:

- SW UCB1 and CUSUM UCB1 set with the same parameters of step 5.

EXP3 algorithm suffers from a linear regret due to its constant exploration

- Non-stationary flavors of UCB1 outperform EXP3

Step 6: more phases setting (25 phases)



Sensitivity analysis:

- window size of SW UCB1 changes only

EXP3 outperforms all versions of UCB1

- robustness to adversarial changes
- capable of adapting the learning rate based on the information collected over time

CUSUM UCB1 is the second best performing

- fixed methodology for calculating the threshold value and detecting out-of-control values

Thank you for your attention!