OLA project: Pricing and Advertising

Lorenzo Ferretti	lorenzo2.ferretti@mail.polimi.it	10713719
Nicolò Fontana	nicolo.fontana@mail.polimi.it	10581197
Carlo Sgaravatti	carlo.sgaravatti@mail.polimi.it	10660072
Marco Venere	marco.venere@mail.polimi.it	10865088
Enrico Zardi	enrico.zardi@mail.polimi.it	10659549

Professors:
Castiglioni Matteo,
Gatti Nicola,
Bernasconi de Luca Martino

A.Y. 2022/2023

The Problem

- Setting: an **e-commerce** website sells a **product** and can control both the **price** and the **advertising strategy**.
- Users have **two binary features** that can be observed by the advertising platforms: **F1** and **F2**
- Users can belong to **three different classes** according to such features: **C1**, **C2**, and **C3**
- Each class is described by two functions:
    1. the **number of daily clicks** as the bid varies
    2. the **cumulative daily cost of the clicks** as the bid varies
- Each class presents a different **purchase conversion rate**
- The time horizon is **365** rounds long
- The reward is given as $R = CR(p) \cdot n_{clicks}(bid) \cdot (p - c) - c_{cost}(bid)$ where p is the chosen prince, b is the chosen bid, and c the unit cost of the product. Indeed, $p - c$ is the margin.

Step 0 : Motivations and Environment design

- The object sold by the website is a **brand watch**.
- The prices space from 200 € to 400 € with intervals of 50 €.
- Each product has a **unit cost** set to 150$
- The E-commerce can observe two different features
  - F1 : **income level**
  - F2: **lifestyle** (fancy or sober)
- Users in class C1 are characterized by a **medium income level** and they are independent from the second feature.
- Users in class C2 are usually **high-income** and **sober lifestyle**
- Users in class C3 are **high-income** but with **fancy lifestyle**

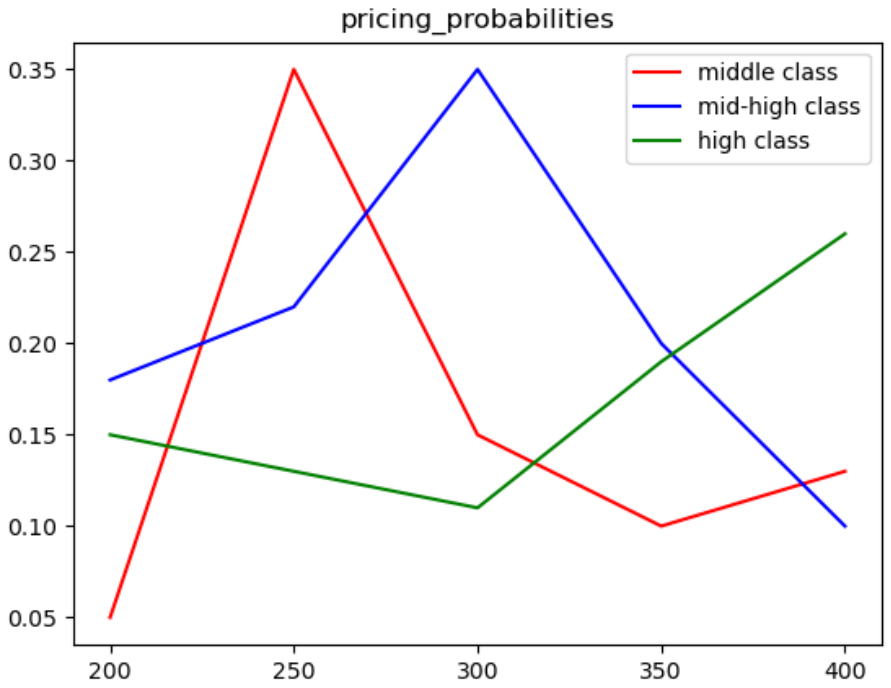
Step 0 : Environment design – Conversion Rates

User are divided in three classes having different conversion rates per prices:

- C1 : buyers of **middle level of income**.
  - Their conversion rate function **monotonically decreases** with the price except for the lowest price at which the conversion rate is very small
  - Users in class C1 are characterized by a medium income level and they are independent of the other feature.

- C2: buyers of **medium-high level of income**
  - Their conversion rate function **peaks** at 300$.
  - Users in class C2 are usually high income and sober lifestyle

- C3: buyers with the **highest income level**
  - Their conversion rate **increases** with the price as it is expected for luxury items.
  - The lowest price still has a **slightly higher** conversion rate than the second price.
  - Users in class C3 are high income but with fancy lifestyle

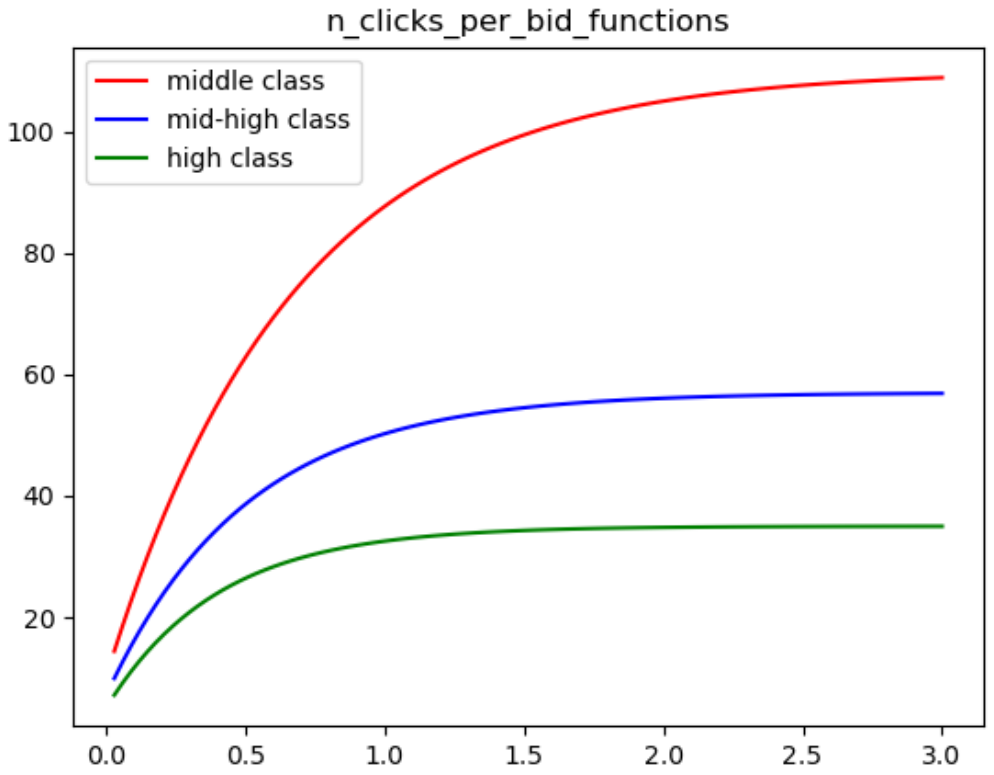


Step 0 : Environment design – Number of Clicks

We devised the following functions for the number of clicks:

- For C1, $n_{click}(bid) = 100\left(1 - e^{-1.5\,bid}\right) + 10$
  - Indeed, C1, the middle class, contains the **highest number of users**, thus the highest multiplicative factor in the function. Its saturation point is the rightmost one.
- For C2, $n_{click}(bid) = 50\left(1 - e^{-2\,bid}\right) + 7$
  - Indeed, C2 is the mid-high class, thus its multiplicative factor is **intermediate** between C1 and C3, as well as its saturation point.
- For C3, $n_{click}(bid) = 30\left(1 - e^{-2.5\,bid}\right) + 5$
  - Indeed, C3 is the high class, with the lowest cardinality, thus its multiplicative factor is **the lowest one**, and the saturation point is the leftmost one.

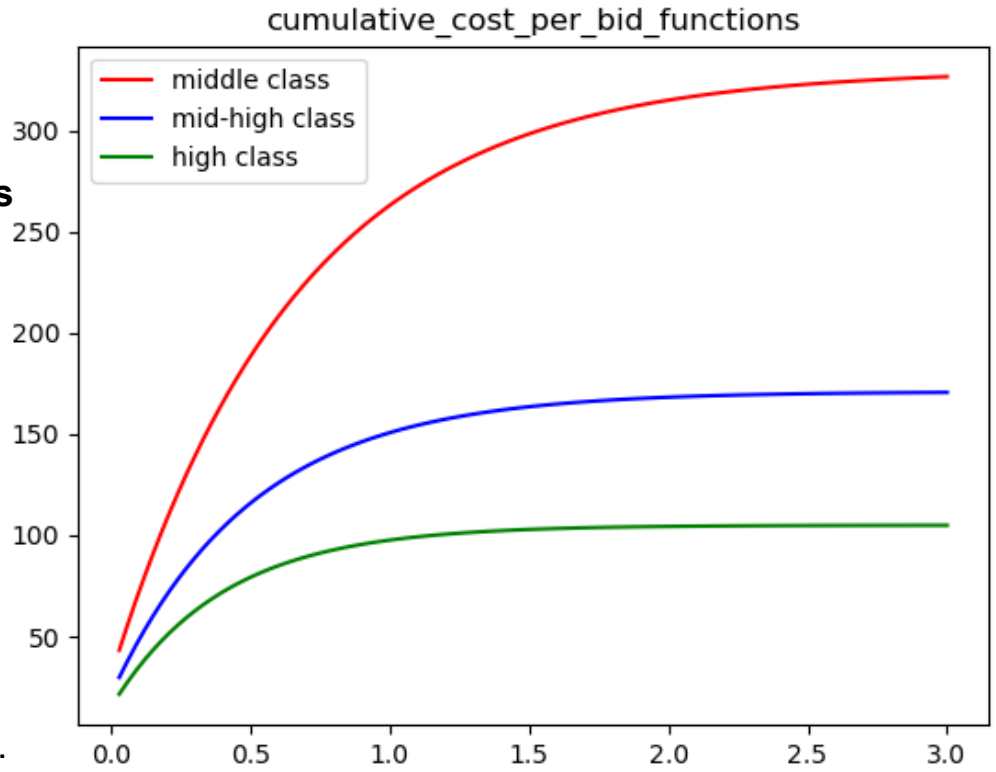
The additive factors are used to achieve realistic distributions



Step 0 : Environment design – Cumulative Cost

We devised the following functions for the cumulative cost:

- For C1, $c_{cost}(bid) = 300\left(1 - e^{-1.5\,bid}\right) + 30$
  - Indeed, C1, the middle class, performs the **highest number of clicks** and therefore the highest cost, thus the highest multiplicative factor in the function is the **highest** one, as well as its saturation point is the **rightmost** one.
- For C2, $c_{cost}(bid) = 150\left(1 - e^{-2\,bid}\right) + 21$
  - Indeed, C2 is the mid-high class, thus its multiplicative factor is **intermediate** between C1 and C3, as well as its saturation point.
- For C3, $c_{cost}(bid) = 90\left(1 - e^{-2.5\,bid}\right) + 15$
  - Indeed, C3 is the high class, with **the lowest cardinality**, and so is the corresponding cumulated cost. Therefore, its multiplicative factor is the **lowest** one, and the saturation point is the **leftmost** one.



The additive factors are used to achieve realistic distributions

Step1 : Learning for pricing - Request

The scenario is the following:

- Consider all the users belonging to class **C1**
- The curves related to the advertising part of the problem are **known**
- The curve related to the pricing problem is **unknown**

The request is to:

- Apply the **UCB1 algorithm** to estimate the conversion rates
- Apply the **TS algorithm** to estimate the conversion rates
- Plot the average value and standard deviation of the **cumulative regret**, **cumulative reward**, **instantaneous regret**, and **instantaneous reward**

Step1 : Learning for pricing - Solution

- Since the solution to the advertising problem is known, the bid b is chosen such that the associated number of clicks and the cumulative cost maximize the reward, given the chosen price, and the margin associated to that price.
- At each round, the two regret minimizers determine the price to choose by maximizing the product between the estimated conversion rate and the margin associated to that price:

$$p^* \in \arg\max \widetilde{CR}_p \cdot (p - c)$$

where $\widetilde{CR}_p$ is the estimated conversion rate for the arm associated with price p.

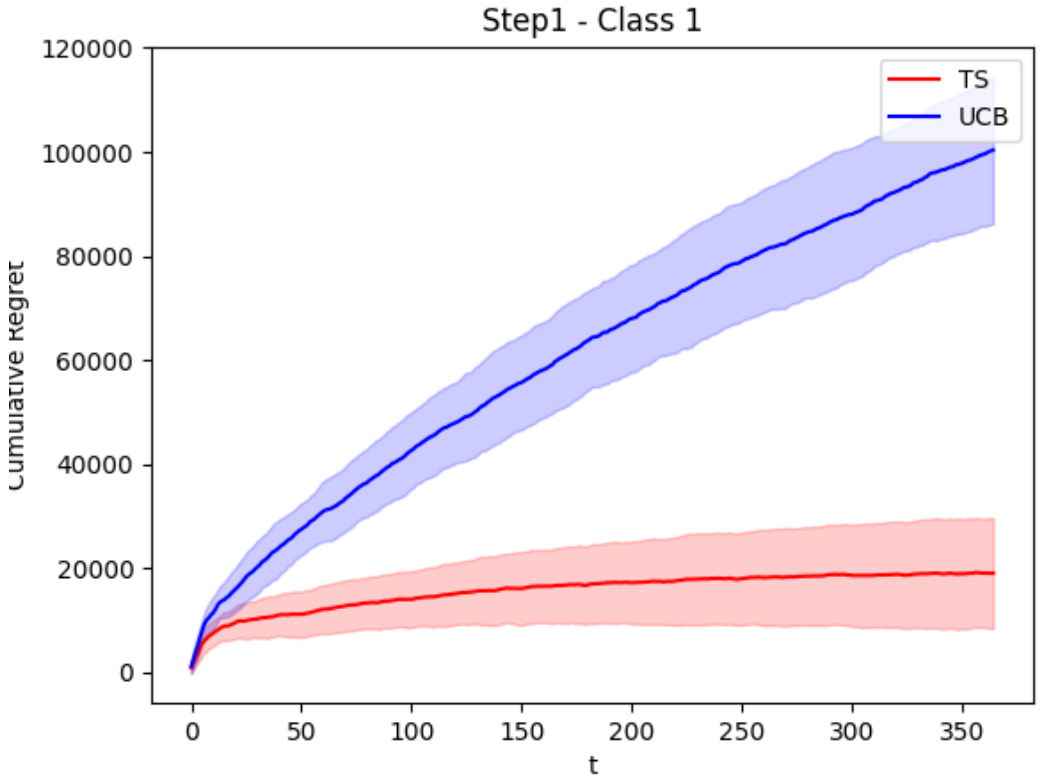
- The update rules for the two algorithms are:
  1. For TS algorithm: $(\alpha_{a_t}, \beta_{a_t}) \leftarrow (\alpha_{a_t}, \beta_{a_t}) + (c_{p,t}, c_{n,t})$
  2. For UCB1 algorithm: $n_{a_t}(t+1) \leftarrow n_{a_t}(t) + c_{p,t} + c_{n,t}$

  where $c_{p,t}$ is the number of positive conversions, $c_{n,t}$ is the number of negative conversion, and their sum is the number of clicks of that round

Step1 : Learning for pricing – Results: Cum. Regret

We run 100 experiments for the previously described process.

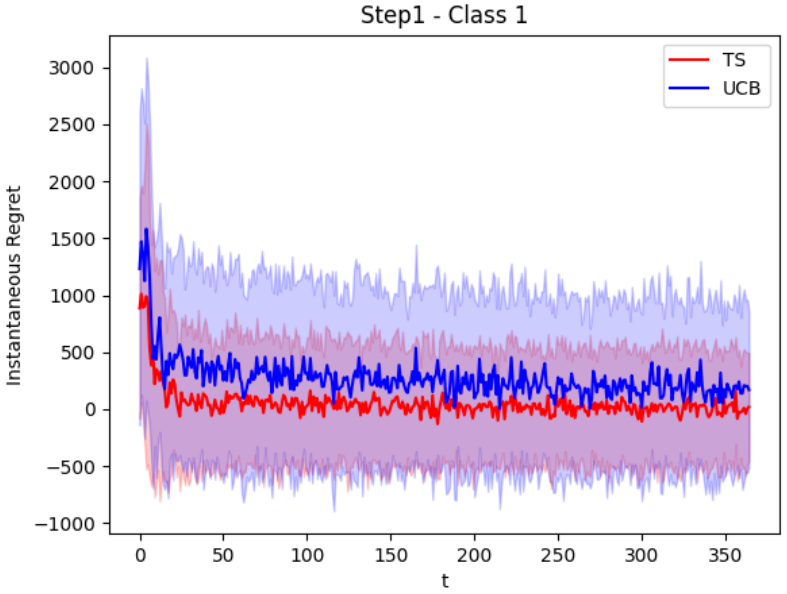
- As expected, both Thompson Sampling and UCB achieved a sub-linear regret, in line with the theoretical guarantees of the methods.
- It is clearly visible that Thompson Sampling performed much better than UCB, reaching the optimal solution in a smaller time than UCB and thus achieving a small regret
- The reason for this might be connected to the structure of the real conversion rates for class C1.

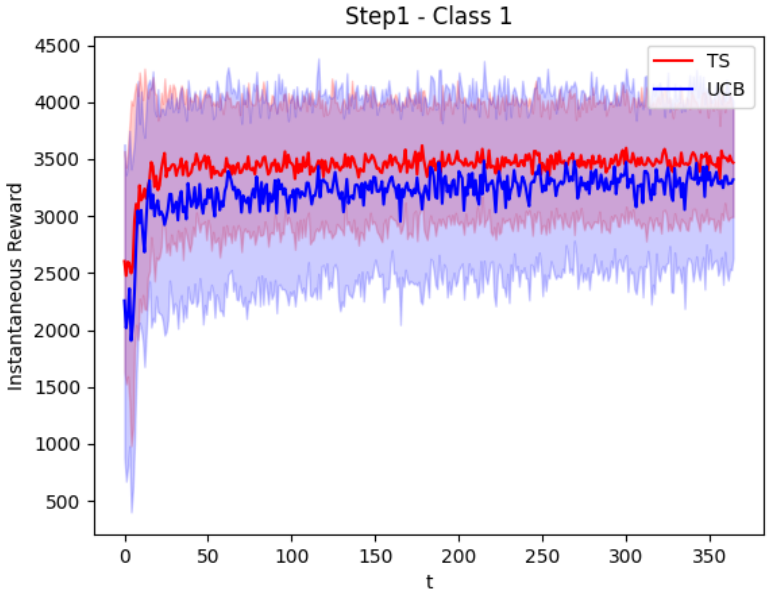


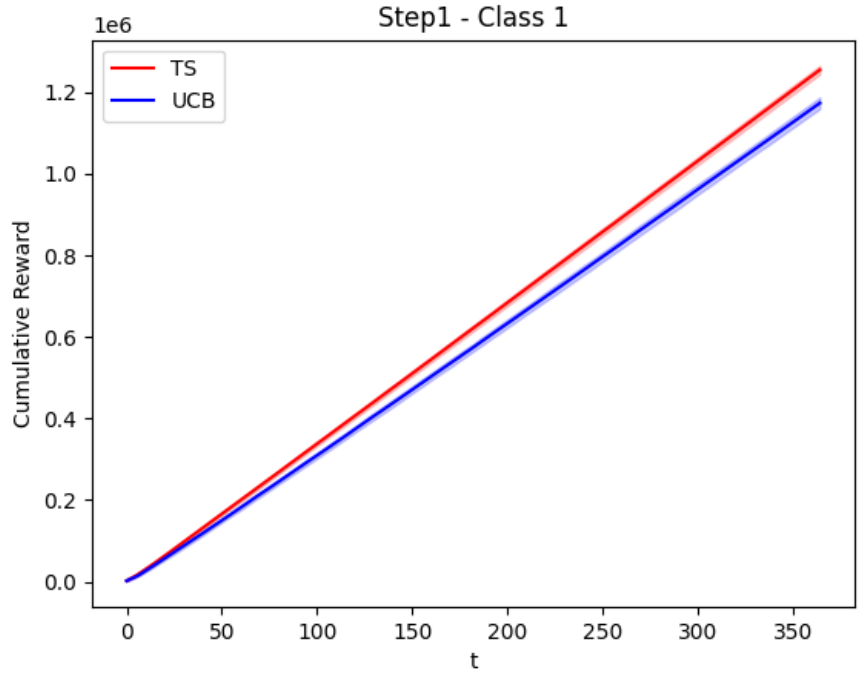
Step1 : Learning for pricing – Results: other plots

We run 100 experiments for the previously described process.

- Clearly, the cumulative reward is in agreement with the trends of the cumulative regret, being higher for TS with respect to UCB1 algorithms.
- Also the trends for the instantaneous regret and reward are as expected.

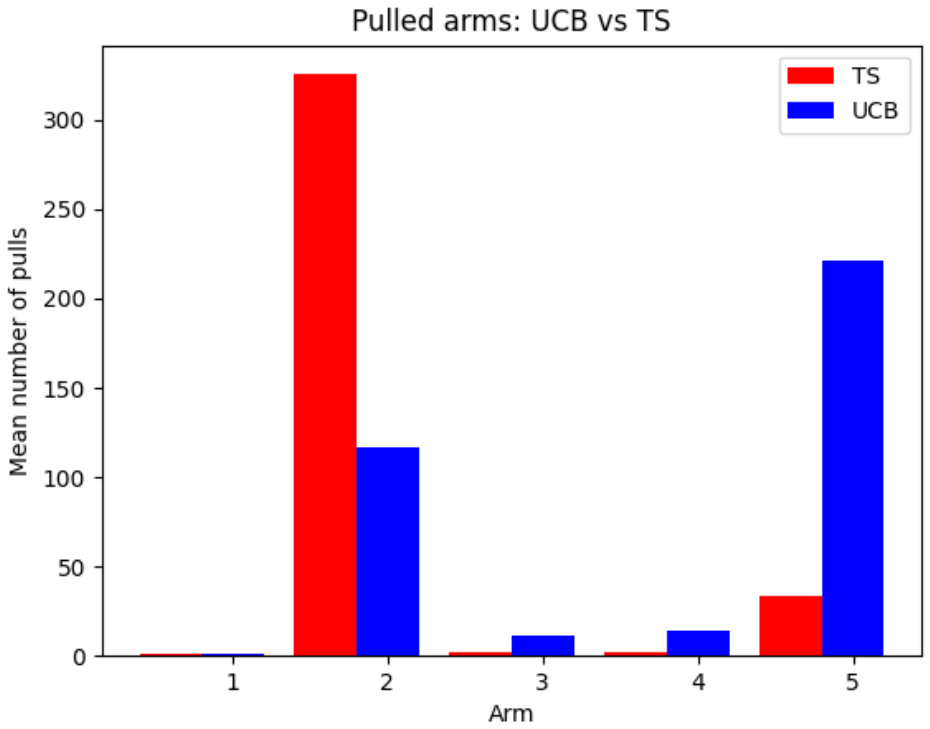






Step1 : Learning for pricing - Observations

- Considering only the conversion rates, the second arm, with 0.35, seems much better than the others.
- Considering the product between the conversion rate and the margin, the second arm is still the best one, providing an expected margin of 35€, but the fifth arm, which has a conversion rate of 0.13 and a price of 400 €, is close since it provides an expected margin of 32.5 €.

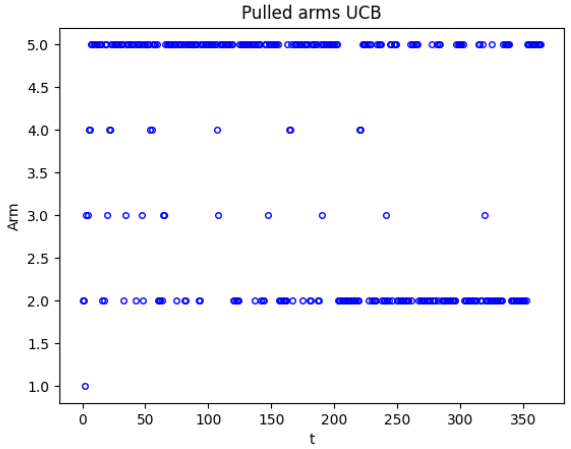


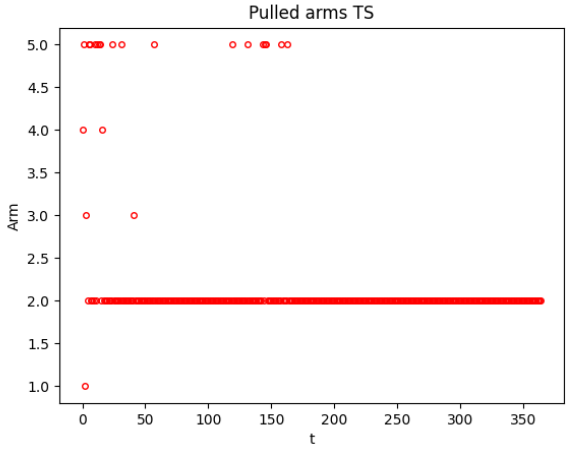
Step1 : Learning for pricing - Observations

- The fact that UCB uses an upper confidence bound for the estimated conversion rate makes the choice of the two arms more difficult since it requires much more exploration due to the presence of the confidence bound: the number of samples of the second arm must not be much more than the ones of the fifth arm, otherwise, even a slightly higher confidence bound in the fifth arm could make it more promising in terms of expected margin.

- On the other hand, Thompson's Sampling adopts a sampling process: as the number of samples is increased, the variance of the Beta distribution is much smaller, thus the sampling becomes much more accurate.

- Since at each round, hundreds of clicks (samples) are observed, it is sufficient to play an arm just for a few rounds in order to have a very low variance in the sampling process, thus making the sample usually really close to the empirical mean of the arm (which should be a good estimate of real the conversion rate, since the number of samples is high)





Step2 : Learning for advertising - Request

The scenario is the following:

- Consider all the users belonging to class **C1**
- The curve related to the pricing problem is **known**
- The curves related to the advertising part of the problem are un**known**

The request is to:

- Apply the **GP-UCB algorithm** to estimate the curves of the advertising problem
- Apply the **GP-TS algorithm** to estimate the curves of the advertising problem
- Plot the average value and standard deviation of the **cumulative regret**, **cumulative reward**, **instantaneous regret**, and **instantaneous reward**.

Step2 : Learning for advertising - Solution

- The **price** is fixed to be the one giving the **best** result in terms of **conversion rate multiplied by the margin**

- Both regret minimizers use **Gaussian processes** to store information about the functions describing, respectively, the number of clicks and the cumulative cost with respect to the bids.

- To use the Gaussian processes, we assume a **correlation exists between arms** near each other. This means that the functions describing the number of clicks and the cumulative cost must be sufficiently **smooth**.

- The main advantage of using Gaussian processes is to obtain a measure of the **uncertainty** over the prediction for each regret minimizer:
  - In the UCB algorithm, this measure is used to compute the **confidence bound** around the predicted mean.
  - In the TS algorithm, it is used as the **standard deviation** of the Gaussian distribution with the predicted mean

Step2 : Learning for advertising - Solution

- For the TS algorithm, a **Gaussian distribution** is used, instead of a Beta one like in the case of the pricing problem, because now the observations on which the update is performed are not Bernoulli distributed anymore, but they represent the means used by the Gaussian processes.

- For the TS case, it uses the means and standard deviations of both regret minimizers to get the samples from the **Gaussian distributions** of the number of clicks and the cumulative cost.

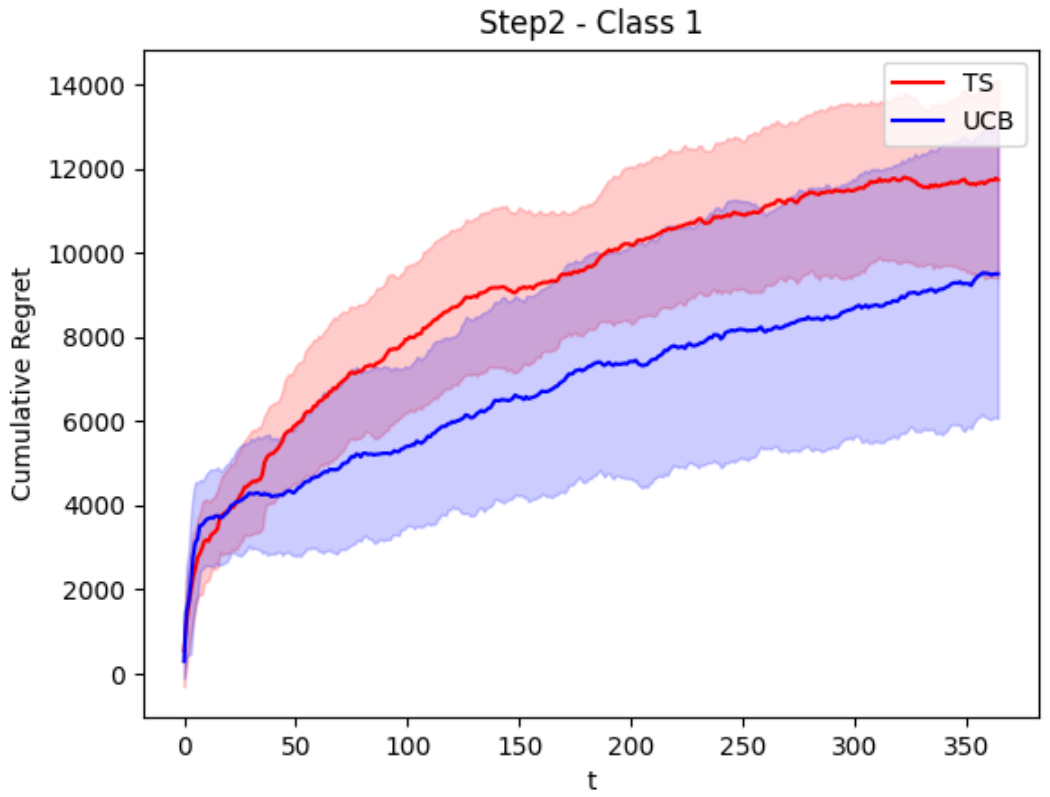
- For the UCB case, it uses means and standard deviations of both regret minimizers to get the **bounds**.

- Finally, the bid to be chosen maximizes the **reward** with respect to the optimal price (known), its corresponding conversion rate and the distributions or bounds for the number of clicks and the cumulative cost.

Step2 : Learning for advertising – Results: Cum. Regret

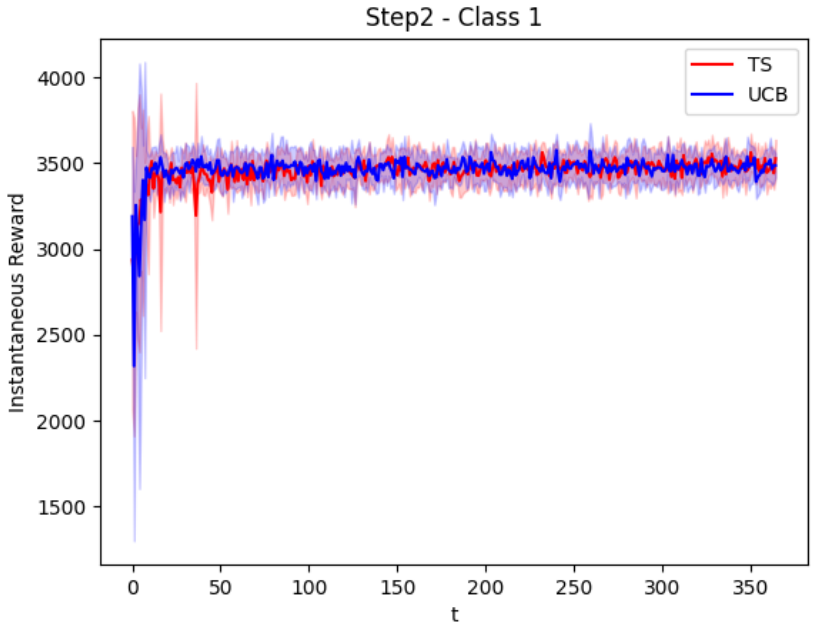
We run 100 experiments for the previously described process.

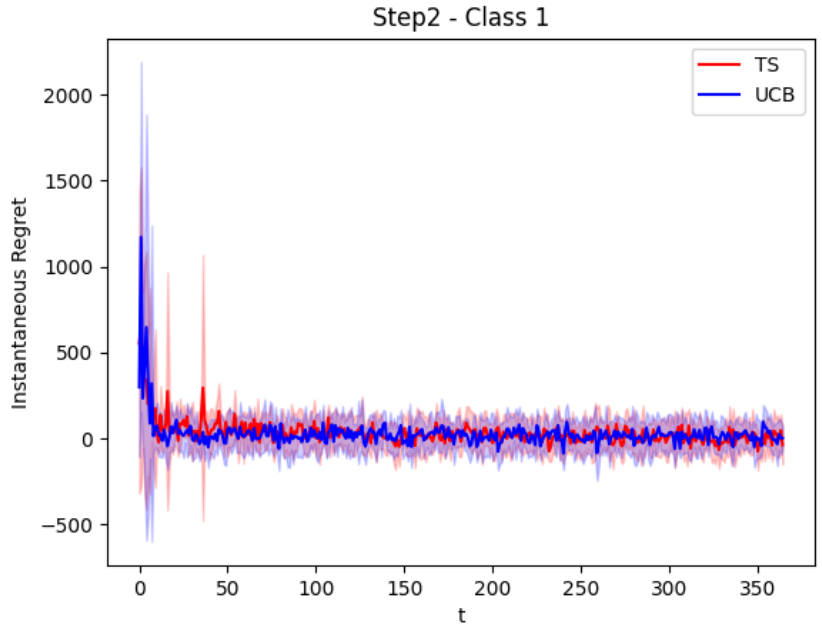
- Both algorithms achieve a sub-linear regret, in accordance with the theoretical guarantees and they do it with a high speed of convergence
- As expected, the performance of the UCB algorithm is better than the one of the TS algorithm.
- They have an high speed of convergence, which matches the assumptions made about the impact of the large amount of samples available in each round and about the simplicity of the functions to be learned.

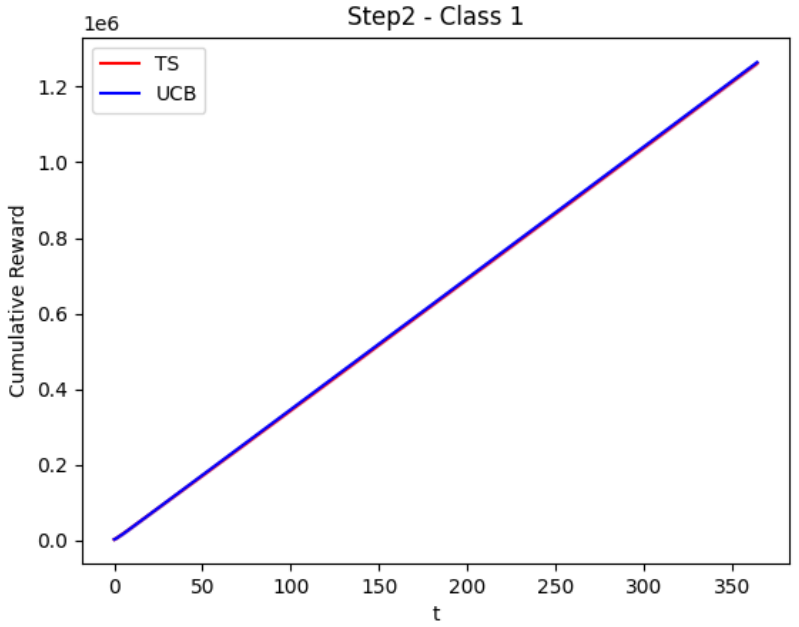


Step2 : Learning for advertising – Results: other plots

- Differently from the plot of the cumulative regret, where the difference between the two regret is more noticeable, in the other plots it is quite difficult to distinguish which algorithm has better results.







Step3 : Learning for joint pricing and advertising – Request

The scenario is the following:

- Consider all the users belonging to class **C1**
- The curves related to the advertising part of the problem are **unknown**
- The curve related to the pricing problem is **unknown**

The request is to:

- Apply the **GP-UCB algorithm** to estimate the curves of the advertising problem
- Apply the **GP-TS algorithm** to estimate the curves of the advertising problem
- Plot the average value and standard deviation of the **cumulative regret**, **cumulative reward**, **instantaneous regret**, and **instantaneous reward**

Step3 : Learning for joint pricing and advertising – Solution

- We adopt a **two-level optimization** approach.
- The first step consists in solve the pricing problem. Once the regret maximizer suggests a price, we then **optimize** the advertising problem using the suggested price and the empirical conversion rate.
- Since no constraint was given on how to solve the **pricing problem** we decided, for simplicity, to adopt the classic **Thompson Sampling algorithm**: the best-performing algorithm in Step1.
- **Reward**: $R = CR(p) \cdot n_{clicks}(bid) \cdot (p - c) - c_{cost}(bid)$
- It can be noted that the optimizations for the best price (and the conversion rate associated) and for the best bid are in principle to be performed **together**. That means a **joint optimization** of 2 variables. This is due to the advertising part in the reward function. Subtracting the cumulative cost makes in principle the reward function **not monotonically increasing** with the bid. However, in our case, with large values of revenue and small cumulative cost, performing a split optimization still guarantees learning the optimal solution.
- Therefore, having a two-level optimization is what is usually done in this case because it is easier **mathematically**, **computationally**, and often retrieves the optimal solution.

Step3 : Learning for joint pricing and advertising – Solution

- The optimization process is as follows:
  - At each round we need to **choose the best arm** to pull for the conversion rate and the best bid to pull.
  - According to the TS learner it is selected the **highest draw** coming from the Beta distributions associated with each arm.
  - The arm selected allows us to have $\widetilde{CR_p}$, the **estimated conversion rate** (the drawn value) and P_d, the **price** associated with that arm.
  - Then we use those values to select the best-estimated bid b^* which is the one which **maximizes** the reward:
    - $b^* = \arg\min_b \widetilde{CR_p} \cdot \ n_{clicks}(b) \ \cdot \ (p \ - c) \ - \ c_{cost}(b)$
      where $n_{clicks}(b)$ and $c_{cost}(b)$ are estimated by the Gaussian process
  - Then, the selected prices and bids are played and the environment returns the **effective conversion rates**, **number of clicks** and **cumulative cost**, which are used to upgrade a **TS learner** and the **two Gaussian processes** associated with the number of clicks and cumulative cost functions.

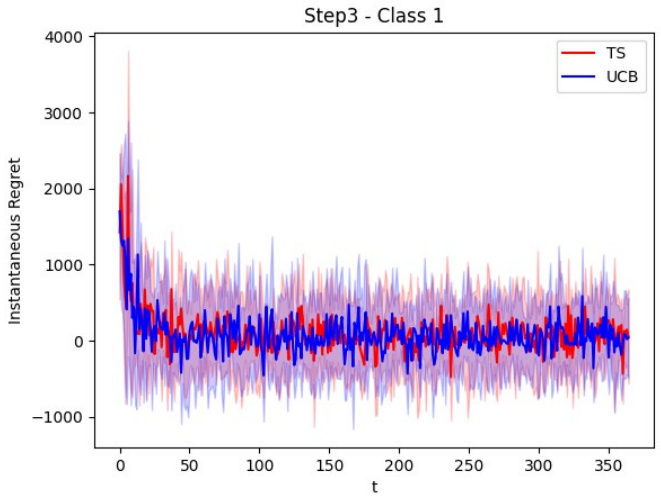
Step3 : Learning for joint pricing and advertising – Results

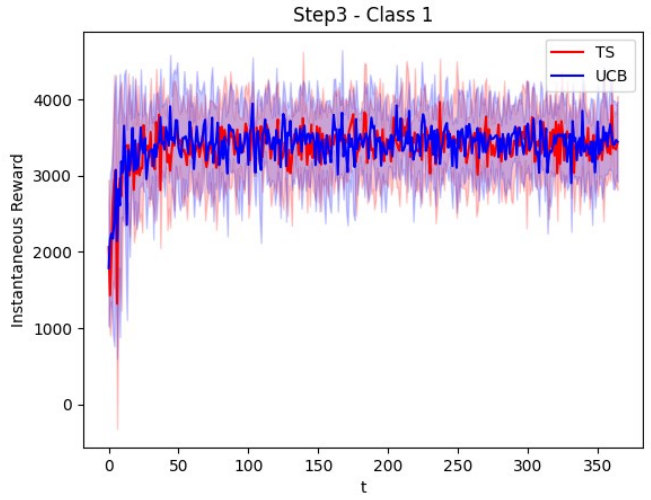
We run 10 experiments for the previously described process.

- By looking at 10 simulations it can be noted that the performance of TS + GPs is **comparable** to the one of Step2 where the prices were known and GPTS and GPUCB were used

- The instantaneous reward reaches 3500€ at limit value as in Step2. This is **expected** because also in that case we used the **optimal** price to perform the calculations of the reward.

- The same can observed for the instantaneous regret which approaches 0 as in Step2. In this scenario the learning process of the conversion rates in addition to the advertising functions does not slow down by much the overall learning of the best joint reward. In fact, by comparing the results of Step1, Step2 and Step3 it can be said that for **all the three scenarios** the learning phase is around 20 days.



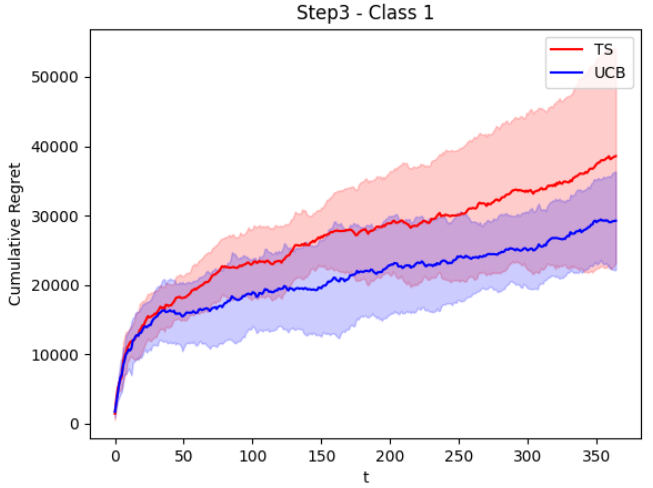


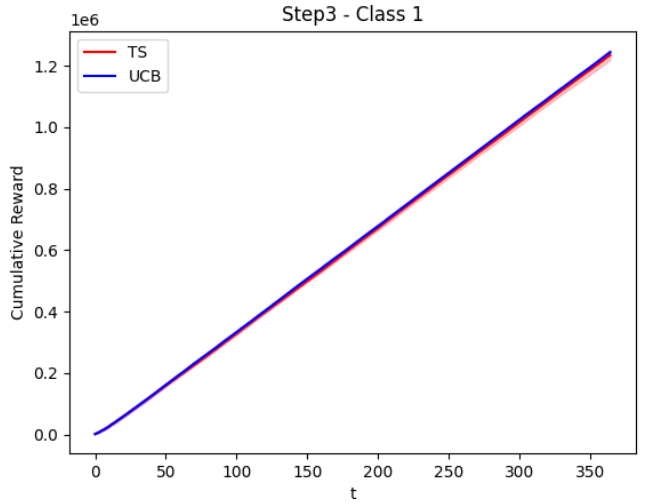
Step3 : Learning for joint pricing and advertising – Results

- Even for Step3 this is not a surprise because we perform a **splitting optimization** of the best price and of the best bid independently. As shown in Step1 and Step2 also there it takes around 20 days individually to learn the optimal values. The variances of the results in Step3 are **larger** than Step1 and Step2 since there is variability in both prices and advertising.

- The cumulative regret can be described as **sublinear** even though the lines for both TS and UCB approaches are **slightly linear**.

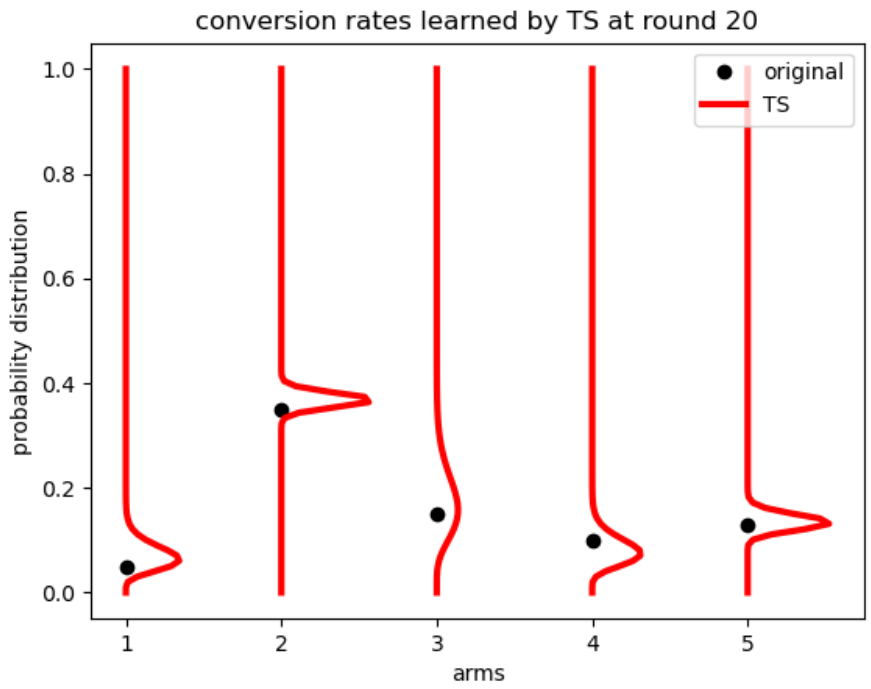
- The reason is that the optimal estimate is computed with the **exact functions** however the number of clicks and cumulative costs functions have a **white noise** added by the environment.

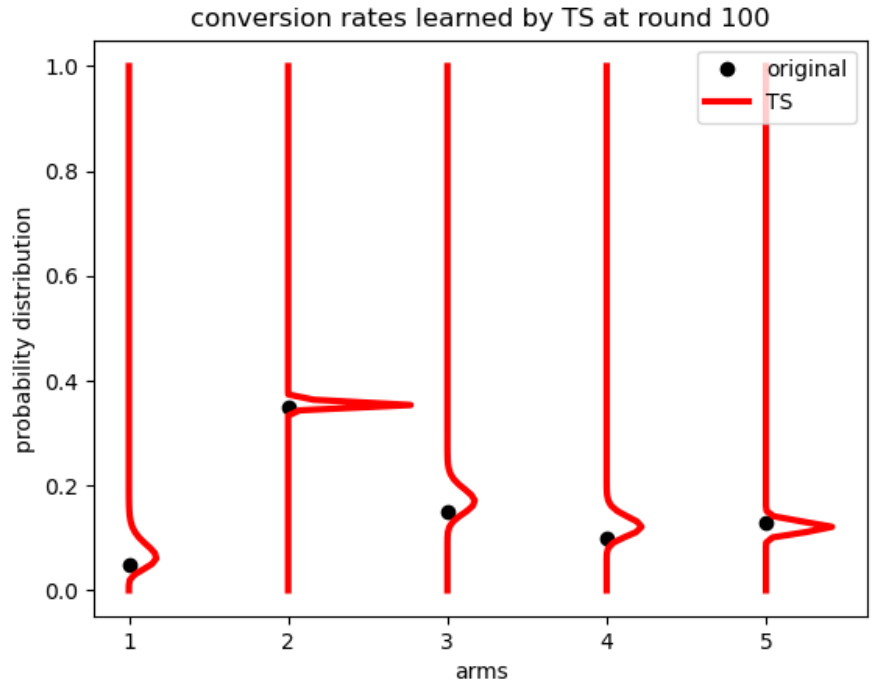




Step3 : Learning for joint pricing and advertising – Results

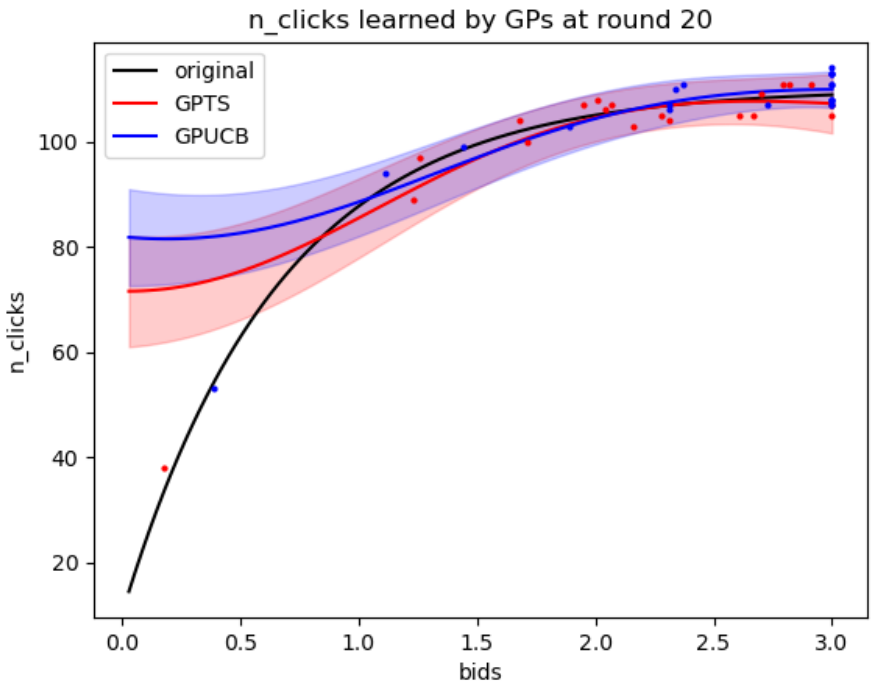
- Here we can notice how the uncertainty of TS algorithm over the conversion rates varies from round 20 to round 100.
  As it can be seen, after 100 rounds the distributions are **sharper**, and the most convenient arms have **very strong peaks**.

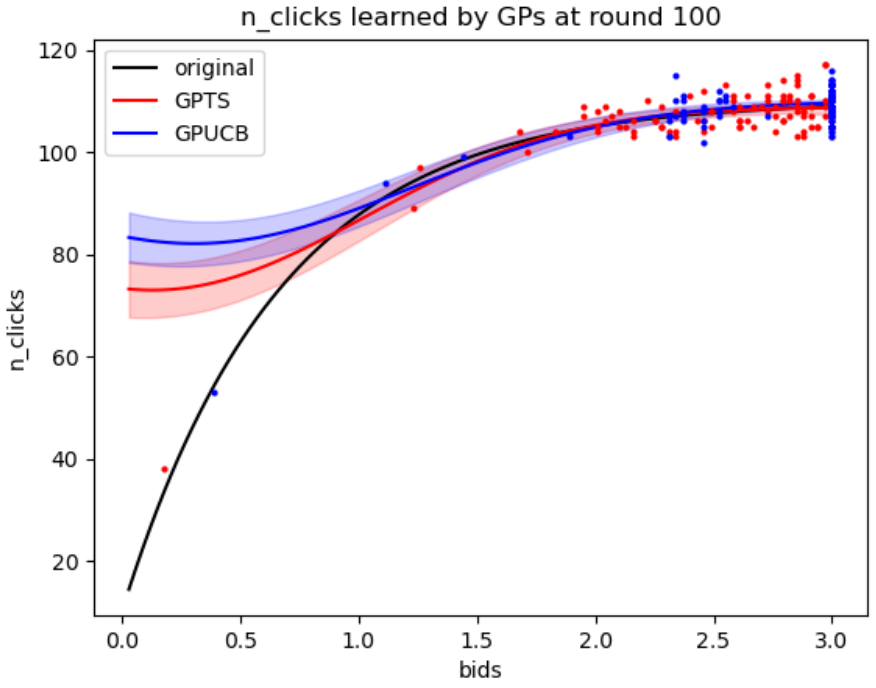




Step3 : Learning for joint pricing and advertising – Results

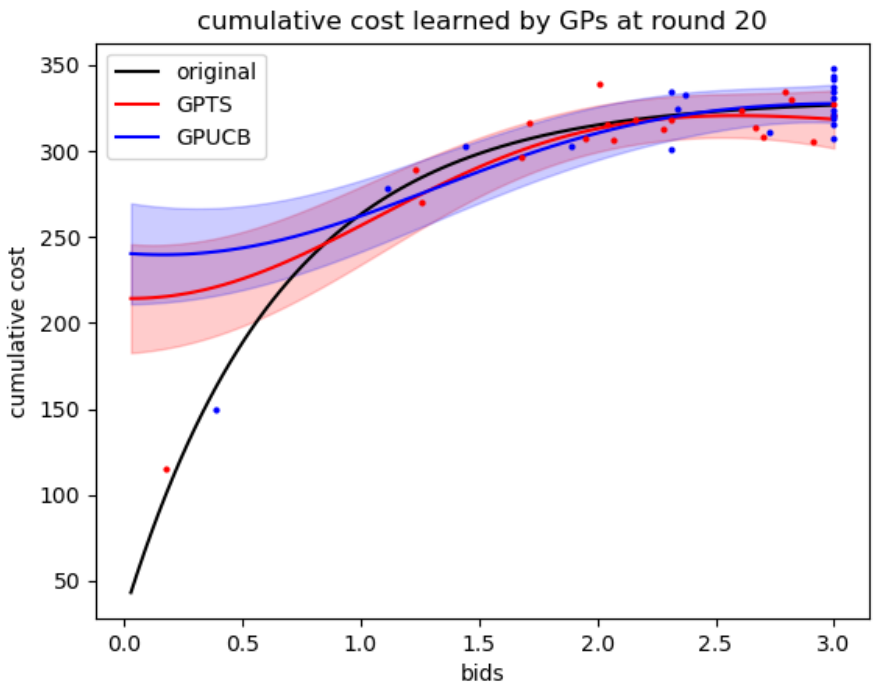
- The same observations can be done with the Gaussian Processes, and their **uncertainty** over the learned curve for the number of clicks. As it can be seen, at round 20, the uncertainty is **higher** with respect to round 100. After 100 rounds, the bids that are more convenient to play are associated with **lower** variance with respect to the others.

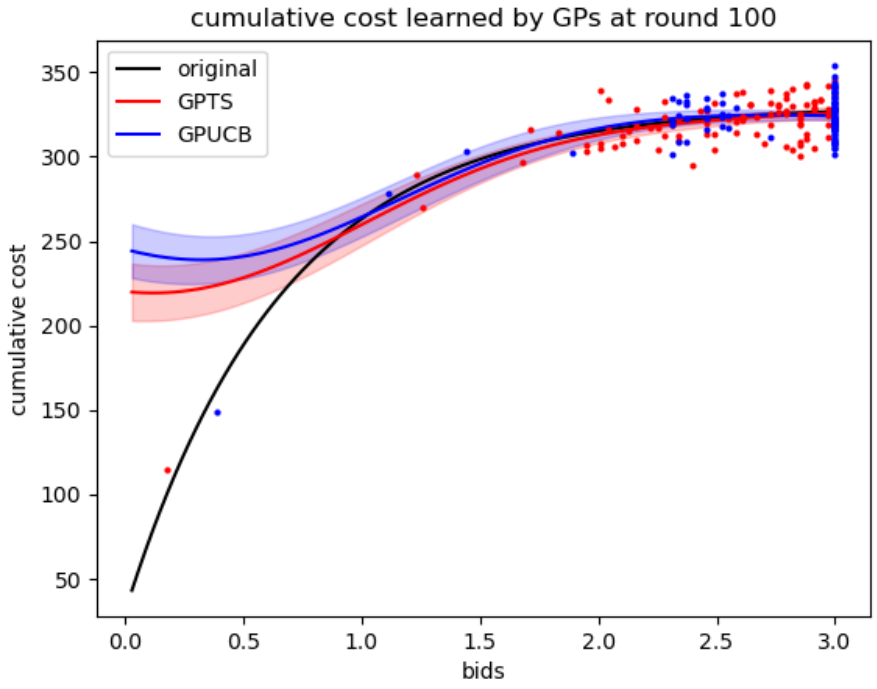




Step3 : Learning for joint pricing and advertising – Results

- Equivalent considerations hold for the learned curve of the cumulative cost. At round 20, the uncertainty is **higher** with respect to round 100, and After 100 rounds, the bids that are more convenient to play have **less uncertainty** than the others.





Step4 : Contexts and their generation– Request

The setting is the following:

- Consider three classes of users: **C1**, **C2**, and **C3**.
- The curve related to the pricing problem is **unknown**
- The curves related to the advertising part of the problem are **unknown**

The request is to consider two scenarios:

1. The structure of the context is **known beforehand**
    - Apply the **GP-UCB** and **GP-TS algorithms** to estimate the curves of the **advertising problem**, and report the related plots.

Step4 : Contexts and their generation– Request

2. The structure of the context is **not known beforehand** and needs to be learnt from data
    - The learner does not know how many contexts there are, while it can only observe the features and data associated with the features.
    - Apply the **GP-UCB** and **GP-TS** algorithms when using GPs to model the two advertising curves paired with a context generation algorithm.
    - Apply the **context generation algorithms** every two weeks of the simulation.
    - Run the GP-UCB and GP-TS algorithms **without context generation**, therefore forcing the context to be only one for the entire time horizon, and **compare their performance** with the performance of the previous algorithms used for the second scenario

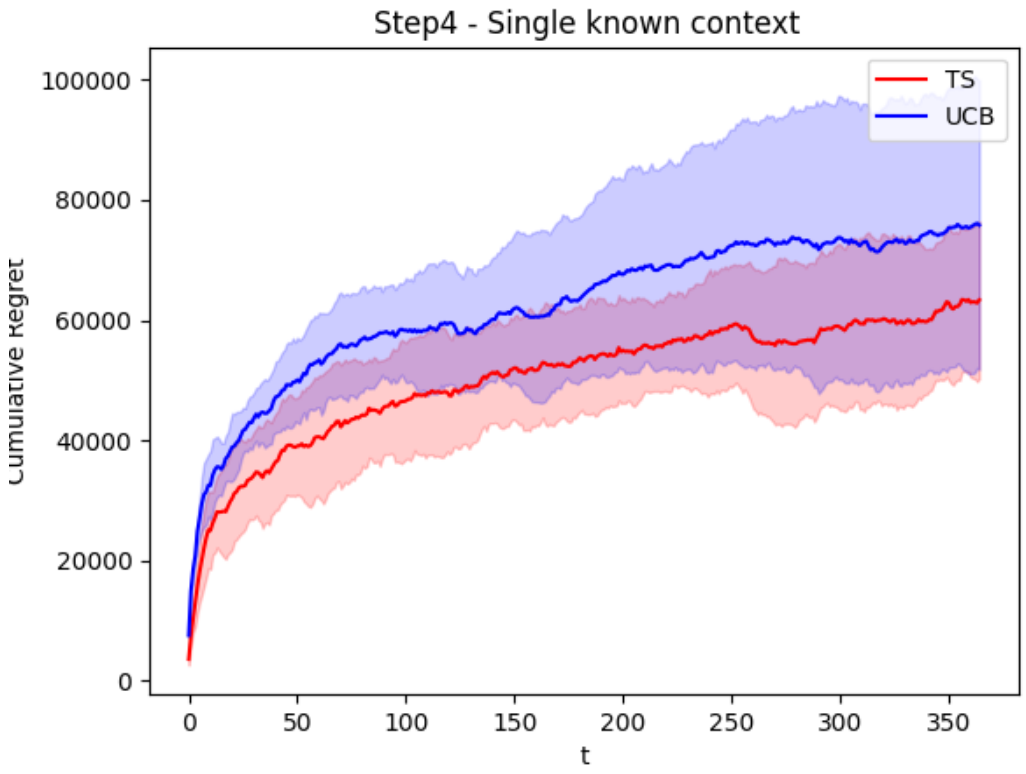
Step4 : Contexts and their generation– Solution Scenario 1

- We assigned a **regret minimizer** to **each class** of users, optimizing each class independently, pulling at each round a price and a bid for each class of users

- The regret is then defined as the **sum of the regrets** of the three regret minimizers.

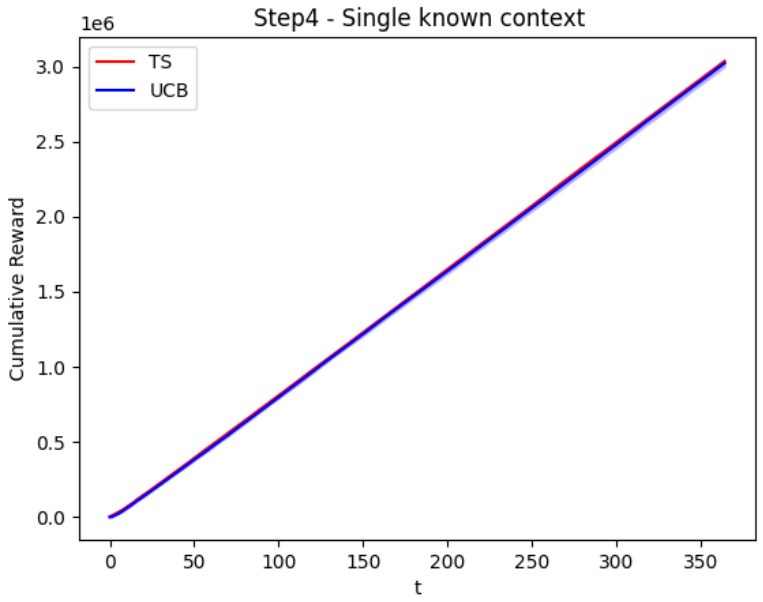
Step4 : Contexts and their generation– Results Scenario 1

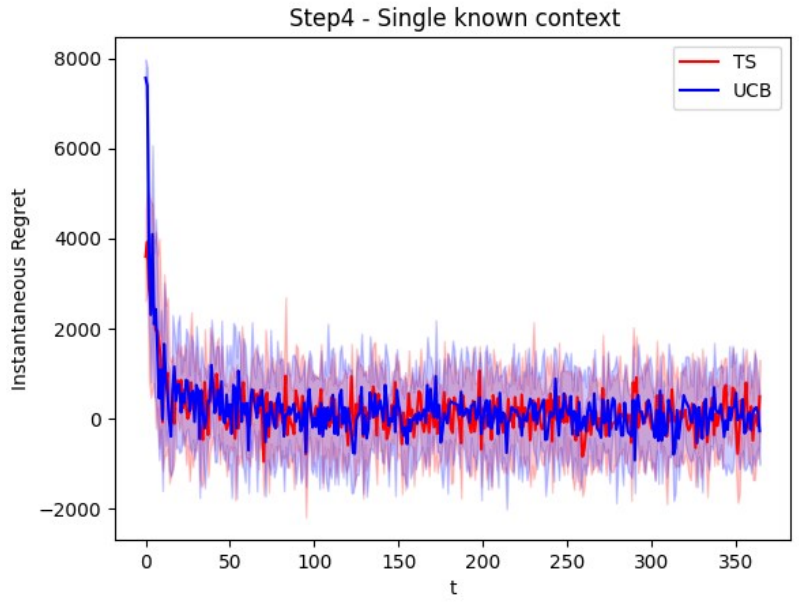
- A **sublinear** regret is obtained for both GP-TS and GP-UCB, thus the results are in line with the ones observed in step 3.
- The figure shows that here Thompson Sampling is performing **better** than UCB, while in the previous two steps, it was the opposite.
- The reason why this happens is that optimizing classes C2 and C3 give **better performances** from the TS side, while for class C1 UCB performs better. Thus, the total regret for the three classes is **lower** for TS.

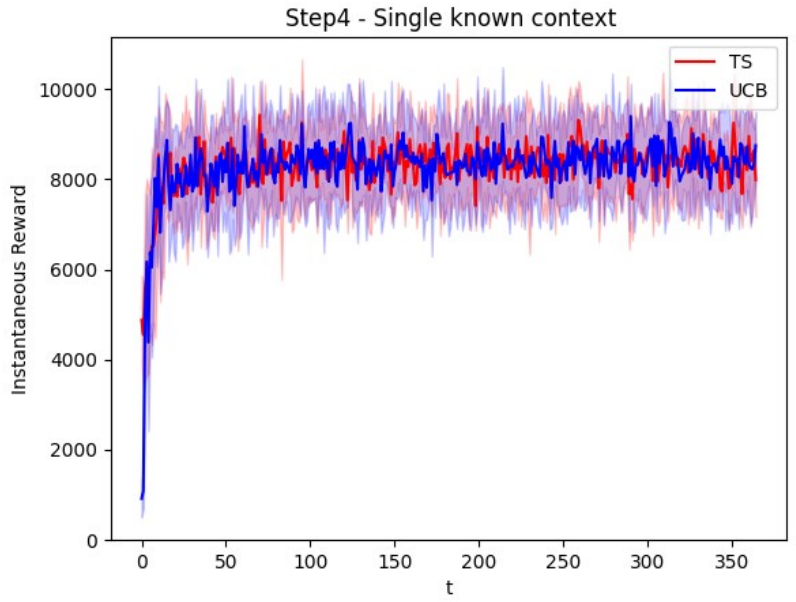


Step4 : Contexts and their generation– Results Scenario 1

- These other plots show the cumulative reward, the instantaneous regret and the instantaneous rewards. These plot show the same trend, yet the cumulative regret made it **more clear**.







Step4 : Contexts and their generation– Solution Scenario 2

- In the second scenario, we start by working on an **aggregated** model, by assigning each combination of features to the same context, i.e. we start with one single regret minimizer.

- At the end of every two weeks, we run the **context generation algorithm**, which will decide to build the new context structure from zero (i.e. a completely disaggregated structure can, in principle, even return to a completely aggregated structure or to something in the middle, and vice versa) based on all the data collected from the beginning.

- The context generation algorithm will use not only the data collected in the previous two weeks but **all the samples** collected from time t = 0.

Step4 : Contexts and their generation– Solution Scenario 2

- Thus, at each round and for each combination of features, we **save the pulled arms** (price and bid), the **number of clicks**, the **cumulative cost**, and the **number of positive conversions** associated with that combination of features at that round.

- We have chosen to use the **greedy algorithm** since we have binary features. The algorithm builds a feature tree based on the following split condition: $\underline{p}_{c1}\underline{\mu}_{a_{c1}*,c1} + \underline{p}_{c2}\underline{\mu}_{a_{c2}*,c2} > \underline{\mu}_{a_{c0}*,c0}$
  - C_0 is the **current context structure** in a node of the features tree
  - $\{C_1, C_2\}$ is the context structure that will be obtained by **splitting** on a certain feature
  - $\underline{p}_c$ is the **lower bound** of the **probability** of occurrence of a user that belongs to the context c
  - $\underline{\mu}_{a_c*,c}$ is the **lower bound** of the **reward** of the optimal arm of context *c*

Step4 : Contexts and their generation– Solution Scenario 2

- In order to obtain the bound on the reward of the optimal arm for a certain context, we reasoned with a **two-step procedure** by computing first the optimal price and then the optimal bid :

  1. The lower bound on the conversion rate for a price p at time t for a context c can be obtained using the **Hoeffding bound** for a **Bernoulli** distribution:

$$CR_p = \frac{\sum_{k=1}^{t} nconv_{k,c}}{\sum_{k=1}^{t} nclicks_{k,c}} - \sqrt{-\frac{\log \delta}{2\sum_{k=1}^{t} nclicks_{k,c}}}$$

where:
$nconv_{k,c}$ = number of positive conversions at time *k*.
$nclicks_{,c}$ = number of clicks obtained at time *k*.
$\delta \in [0,1]$ = confidence bound
The optimal price is the one with the **highest product** beween the price and the lower bound of the conversion rate .

$$p^* \in \arg\max_p CR_p \cdot (p - cost)$$

Step4 : Contexts and their generation– Solution Scenario 2

2. For the bid, we have two parameters to estimate: the **number of clicks** and the **cumulative cost**:

   - We have considered the **lower bound** of the number of clicks and the upper bound of the cumulative cost (since the cumulative cost has a minus in the formula of the reward).

   - Both are assumed to be Gaussian variables, thus we have obtained them using the bounds of **95% confidence interval** for a **Gaussian** process.

   - The **lower bound** of cumulative cost is defined as: $NC_b = \frac{1}{t}\sum_{k=1}^{t} nclicks_{k,c} - 1{,}96\frac{\sigma_{nclicks_c}}{\sqrt{t}}$

   - The **upper bound** of cumulative cost is defined as : $CO_b = \frac{1}{t}\sum_{k=1}^{t} cumcost_{k,c} + 1{,}96\frac{\sigma_{cumcost_c}}{\sqrt{t}}$

Step4 : Contexts and their generation– Solution Scenario 2

- Finally the optimal bid is defined as the one that **maximises** the total reward per click :

$$b^* \in \arg\max_p \frac{1}{NC_b}[CR_{p^*} NC_b(p^* - cost) - CO_b]$$

- While the lower bound for the optimal reward per click for context c is given by :

$$\underline{\mu}_{a_{c^*},c} = \frac{1}{NC_{b^*}}[CR_{p^*} NC_{b^*}(p^* - cost) - CO_{b^*}]$$

- In order to compute the lower bound on the probability of the context c_1 (or c_2), we used the **Hoeffding bound**, where the empirical mean is defined as the **fraction of clicks** belonging to that context among the ones of the aggregated context c_0 (equal to the sum of the clicks belonging to c_1 and c_2):

$$\underline{p}_{c_1} = \frac{\sum_{k=1}^{t} nclicks_{k,c_1}}{\sum_{k=1}^{t} nclicks_{k,c_1} + \sum_{k=1}^{t} nclicks_{k,c_2}} - \sqrt{-\frac{\log \delta}{2\sum_{k=1}^{t} nclicks_{k,c_1}}}$$

Step4 : Contexts and their generation– Solution Scenario 2

- After the execution of the **greedy algorithm**, we assign to each context a regret minimizer: if the context is new, a new **regret minimizer** is assigned to it, otherwise the **previous regret minimizer** will continue to be associated with it.

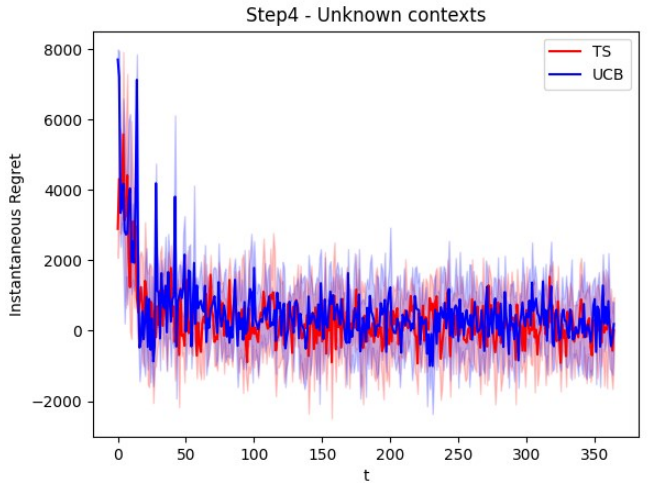
- If a contest is not present in the new structure its regret minimizer will be **deleted**. The new regret minimizers will **not** start from zero; indeed, we initialize them with all the past samples belonging to the associated context, performing a **bulk update** after creating them. We do this in order to not have a useless initial exploration since we can **exploit** past information to initialize the estimates.

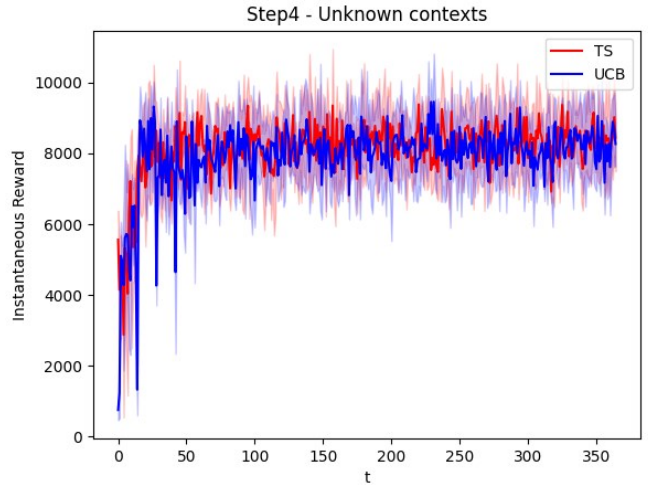
Step4 : Contexts and their generation – Results

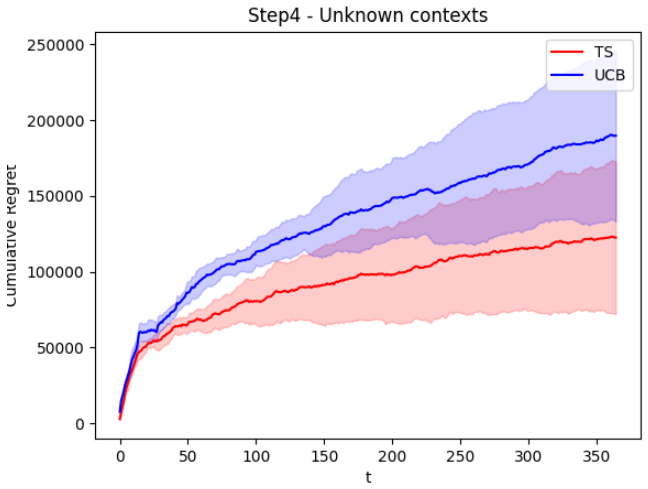
- The effect that the context generation algorithm has is **clear**. Considering the first rounds, the cumulative regret has a **very high slope** and the instantaneous regret is very high.

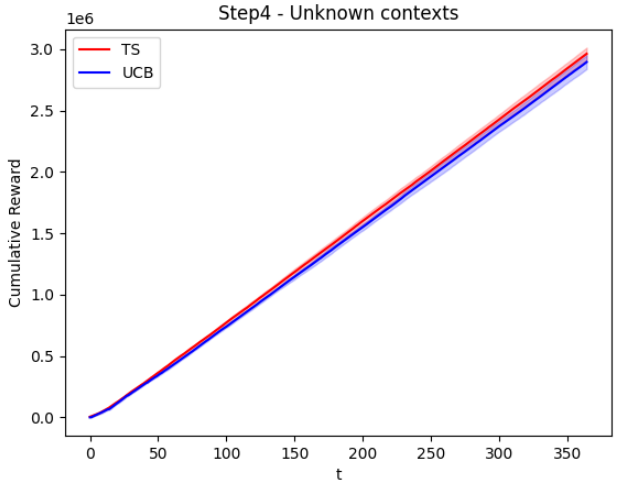
- After a couple of rounds (approximately after two weeks), it is possible to notice a **big increase** (decrease) in the instantaneous **reward** (regret). This means that the context generation algorithm has found a **new context structure** that is better than the aggregated structure and is at least **closer** to the optimal one.

- Starting from that point, the regret is always **very small**, with a bit of oscillation (especially for UCB, which bounces from **almost zero regret** to more than 1000$ of instantaneous regret) until more or less 100 rounds.



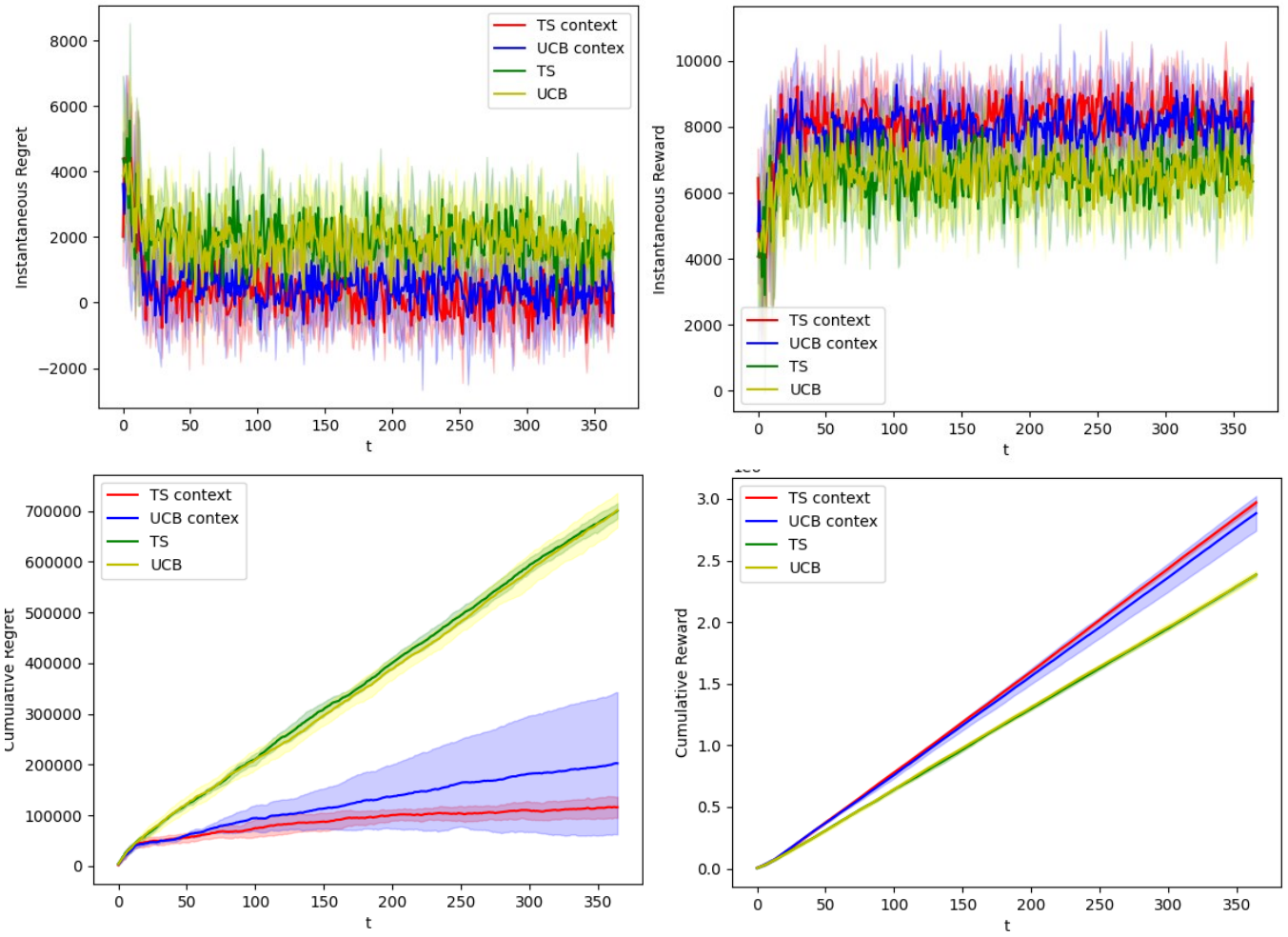






Step4 : Contexts and their generation – Observations

- After this period, the regret is **more stable** around zero, meaning that the context generation algorithm is able to **converge** to the optimal context structure much more times.
- The fact that the context structure is **unknown** makes the problem **more difficult** to optimize.
- It is possible to notice that the instantaneous regret (and reward) has a **much higher variance** in this case, with respect to the previous scenario with a known context structure.
- It can also be noticed that the cumulative regret at the end of the time horizon is **much larger** than in the previous scenario. Indeed, here more than 100k $ of regret has been obtained by TS, while previously the regret was around 60k $



Step5 : Dealing with non-stationary environments with two abrupt changes - Request

The setting is the following:

- Consider all users belonging to class **C1**.
- The curves related to the advertising part of the problem are **known**
- The curve related to the pricing problem is **unknown**, **non-stationary** and have **three different phases**.

The request is to:

- Apply the **UCB1 algorithm with sliding windows** to estimate the curve of the princing problem
- Apply the **UCB1 algorithm with change detection** to estimate the curve of the princing problem
- Provide a **sensitivity analysis** for the parameters of the algorithm.
- Plot the average value and standard deviation of the **cumulative regret**, **cumulative reward**, **instantaneous regret**, and **instantaneous reward**

Step5 : Dealing with non-stationary environments with two abrupt changes - Solution

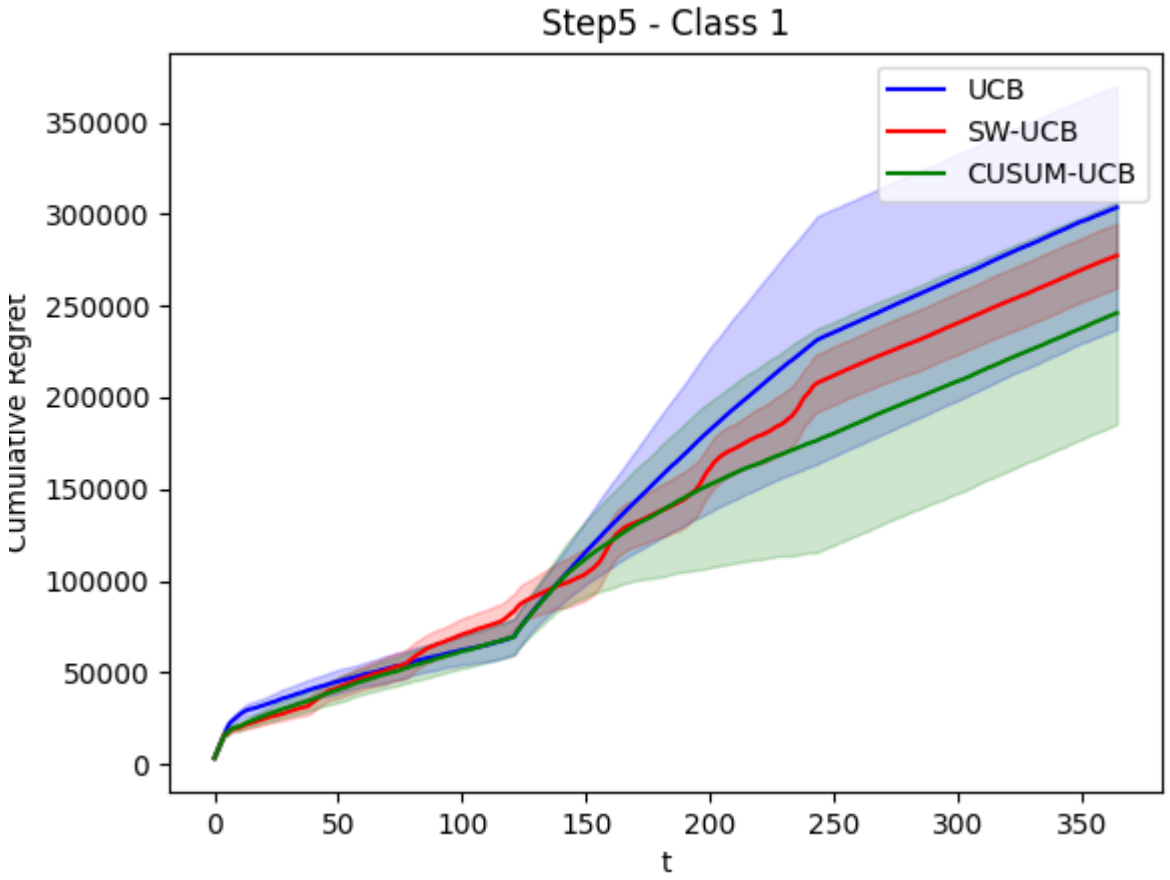
- Since the price is known, we used an optimizer to learn the price giving the **highest** reward computed using the **real values** of the number of clicks and cumulative cost given by the bid related to the chosen price
- We associated each phase with a different time of the year, having a time horizon of 365 days starting from May.
    1. The **first** phase , May- September, is characterized by **regular** purchasing behavior where the preferred price (for middle-class customers) is the **second cheapest**
    2. The **second** phase, around September, people are more willing to purchase even if at a **higher** cost. The consequences are that the chosen price of the second phase is the **middle** one (higher than the regular one) and every price meets an **increase** in their conversion rate.
    3. Finally, the **third** phase is the spring one, characterized by **discounted** sales. During this period, the cheapest price is the preferred one with a huge peak in its conversion rate, while the others suffer a **reduction** in their conversion rate, the cheaper ones because they are **too similar** to the regular one (representing little to no saving to the customers' eyes) and the more expensive ones because in **contrast** with the trend of the whole phase.

Step5 : Dealing with non-stationary environments with two abrupt changes - Solution

- To choose the parameters of the two variants of UCB1 we relied on theoretical suggestion:
  - For the **sliding window size**, we opted for a value directly proportional to $\sqrt{T}$
  - For the active change detection variant we used **CUSUM:**
    - change detection parameter : $\approx logT$
    - Exploration parameter $\approx \sqrt{\frac{logT}{T}}$
- We did not detach from the **theoretical suggestions**, and we did not perform any kind of tuning because both algorithms are deployed over a **single** case of functions; in other words, there is no variety or randomicity in the functions modelling the behaviours of customers (apart for the noise added when receiving the reward). This **limitation** in the variety of cases analyzed could have been the cause of serious **overfitting** if we decided to tune the parameters of the algorithms with respect to the results achieved in our experiments.

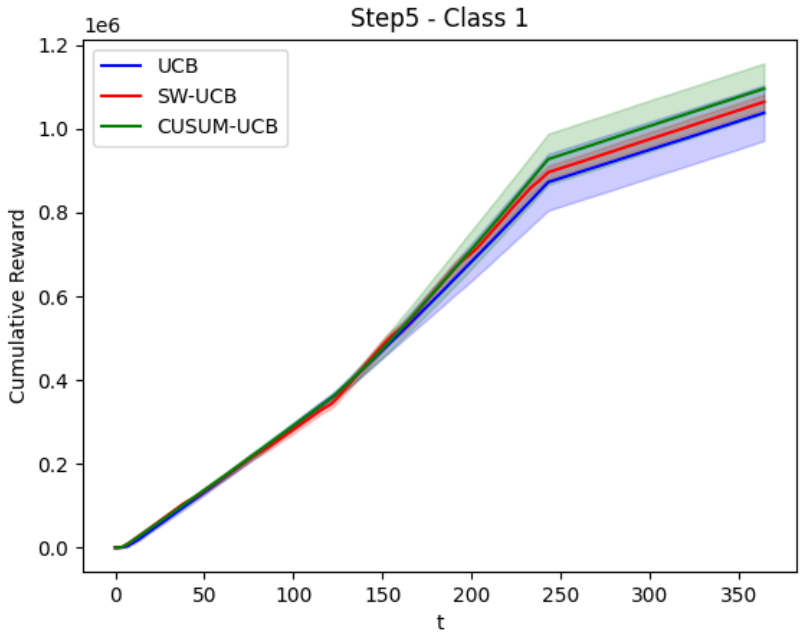
Step5 : Dealing with non-stationary environments with two abrupt changes - Results: Cum. Regret

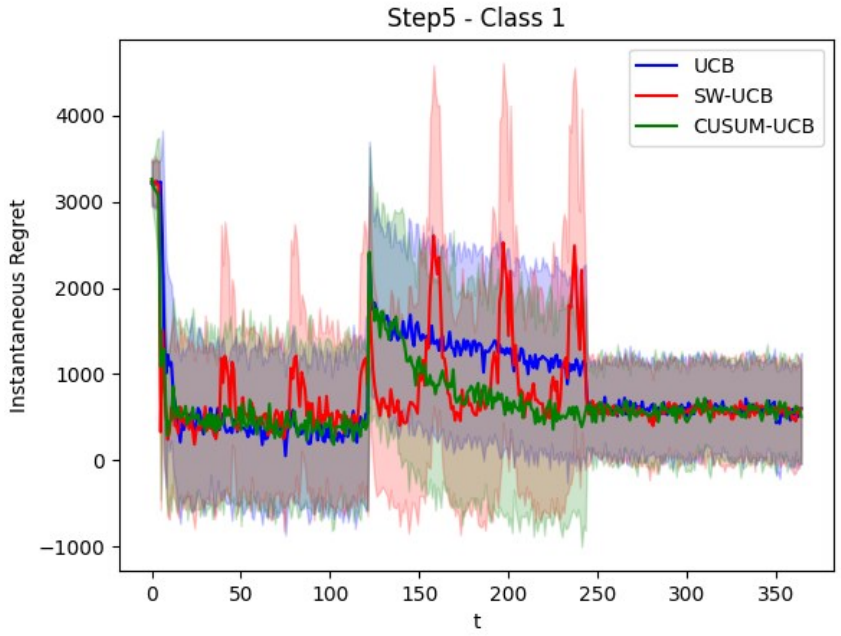
- As expected, the algorithm achieving the **best regret** is CUSUM, with sliding window being the **second best** and standard UCB the **worst** one.
- It is interesting to notice that the sliding-window version is affected by some **spikes** with **decreased reward** and **increased regret**.
- We can assume that these peaks are caused by the **worst arms** which are pulled just once for each window; indeed, when the time step at which their samples were obtained exits the window of validity those arms have no valid samples anymore and, thus, they get **infinite confidence bound** and, as a consequence, the algorithm is forced to suggest those arms for the next pulls. Once they are pulled and a poor reward is observed SW-UCB can suggest again **better** arms.

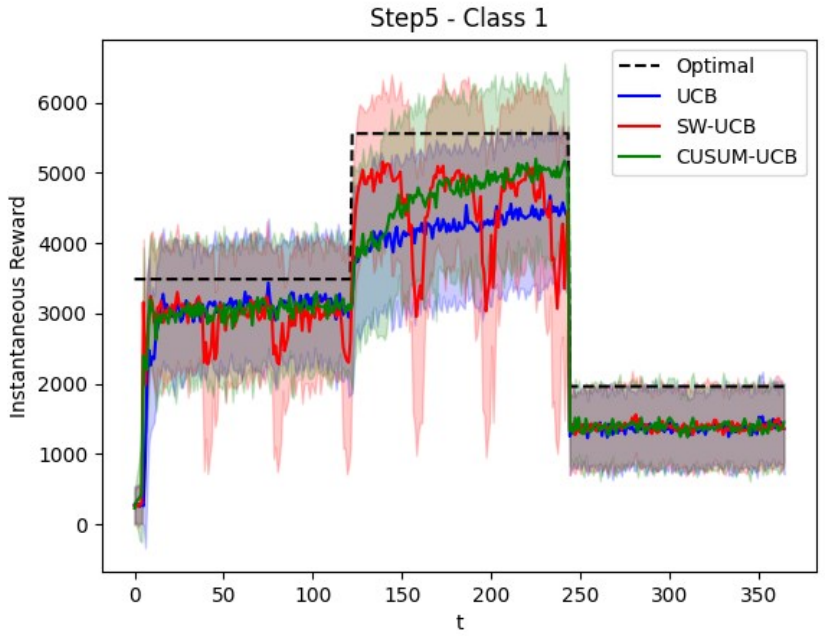


Step5 : Dealing with non-stationary environments with two abrupt changes - Results: Other plots

Here we show the plots for the **cumulative reward** and the **instantaneous regret** and **reward**.
As it can be noticed, also in these plots it is possible to see the **peaks** related to UCB with sliding windows







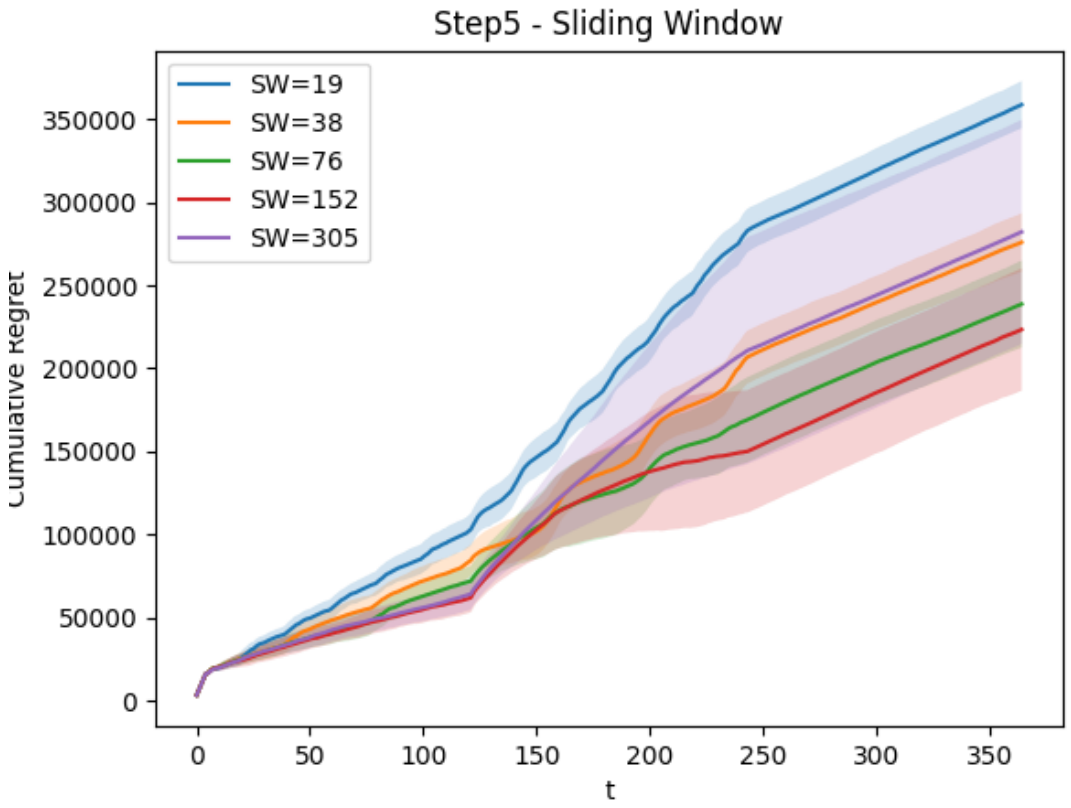
Step5 : Dealing with non-stationary environments with two abrupt changes - Sensitivity analysis

For the **sensitivity analysis**, we considered the following parameters:

- UCB sliding window parameters tested:
  - **Window Size**
- UCB cusum parameters tested:
  - **M**: Number of samples to be used to compute the empirical mean that will be used as a reference for the current behavior in the actual change detection phase.
  - ***h:*** Threshold to detect any change in the customers' behavior.
  - $\boldsymbol{\varepsilon}$*:* Critical level used to adjust the sensitivity of the change.
  - $\boldsymbol{\alpha}$*:* Amount of exploration of the algorithm

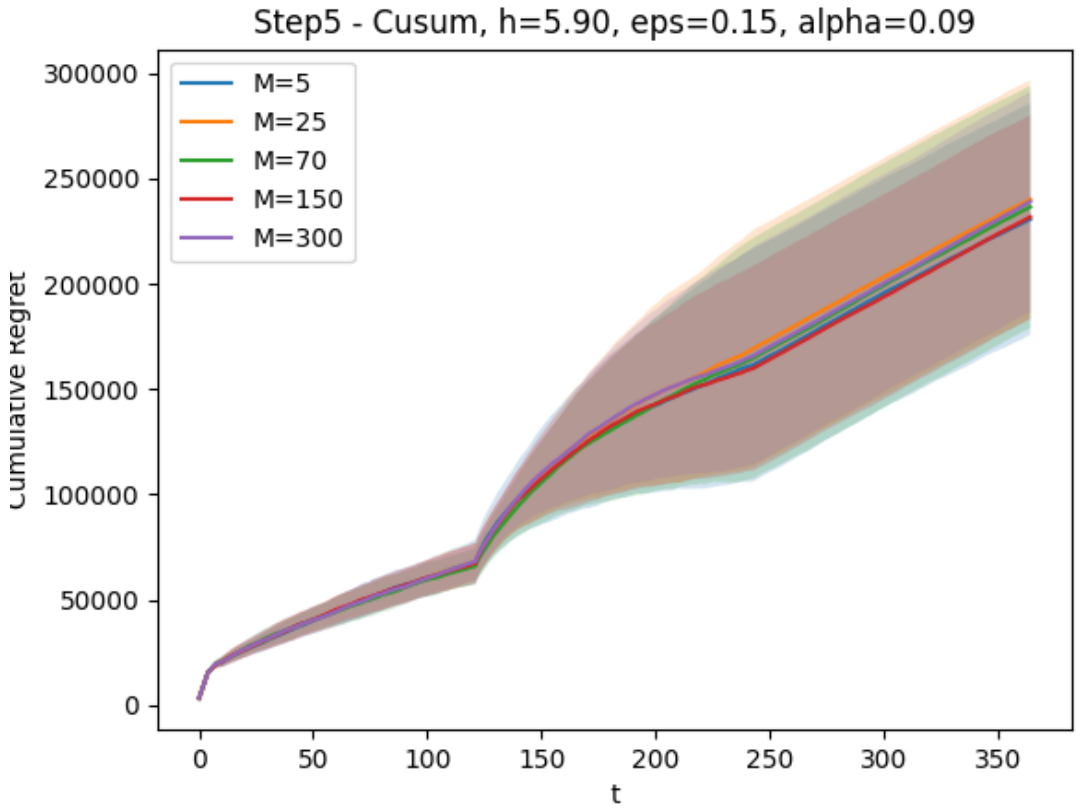
Step5 : Dealing with non-stationary environments with two abrupt changes - Sensitivity analysis

Sliding Window: Window size analysis.



- For sliding window UCB, we tested **5 different values** of window size, all of them computed proportionally to $\sqrt{T}$ but with different multiplying factors (respectively: 1, 2, 4, 8, 16)
- With small window, the number of samples used to learn is **limited**, thus more **exploration** is performed (because it is more probable for an arm to have no valid samples).
- The consequence of a small window size is that for the algorithm to be effective the changes in the environment behavior must be **frequent** and more **abrupt**.
- With a big window allows using more samples to learn, but this is **counterproductive** in the case in which the changes are too frequent because too much history of an arm is taken into consideration, even samples coming from a previous, different, behaviour.

Step5 : Dealing with non-stationary environments with two abrupt changes - Sensitivity analysis

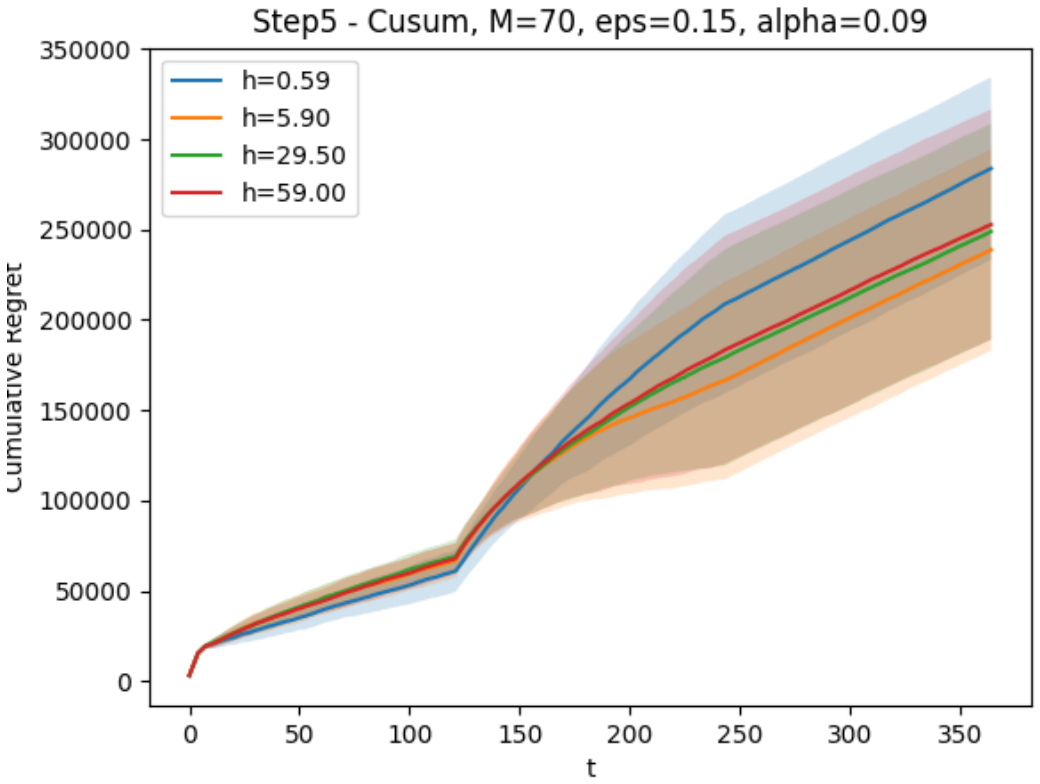


CUSUM: M

- We tested 5 different values for M, the number of samples to be used to compute the empirical mean, all of them of the **same order of magnitude** of the time horizon
- The **higher** the value of M the **higher** the number of samples used to compute the mean. This means that the mean obtained is more indicative of the **real behavior**.
- The downside of bigger values of M is that the detection test is not performed until all the M samples are collected, thus if the value is too high it could be that a change in the behavior is detected too late or even not detected at all.
- The performance of the algorithm in terms of cumulative regret is quite similar even for very different values of M.
- Our hypotheses for this result are that **other parameters** have a **much more important impact**, is relevant to remember that this is a variation of UCB, so there is still a learning algorithm behind the change detection add-on. Hence it makes sense that the performances do not change too much after the variation of a **less influential** parameter.

Step5 : Dealing with non-stationary environments with two abrupt changes - Sensitivity analysis

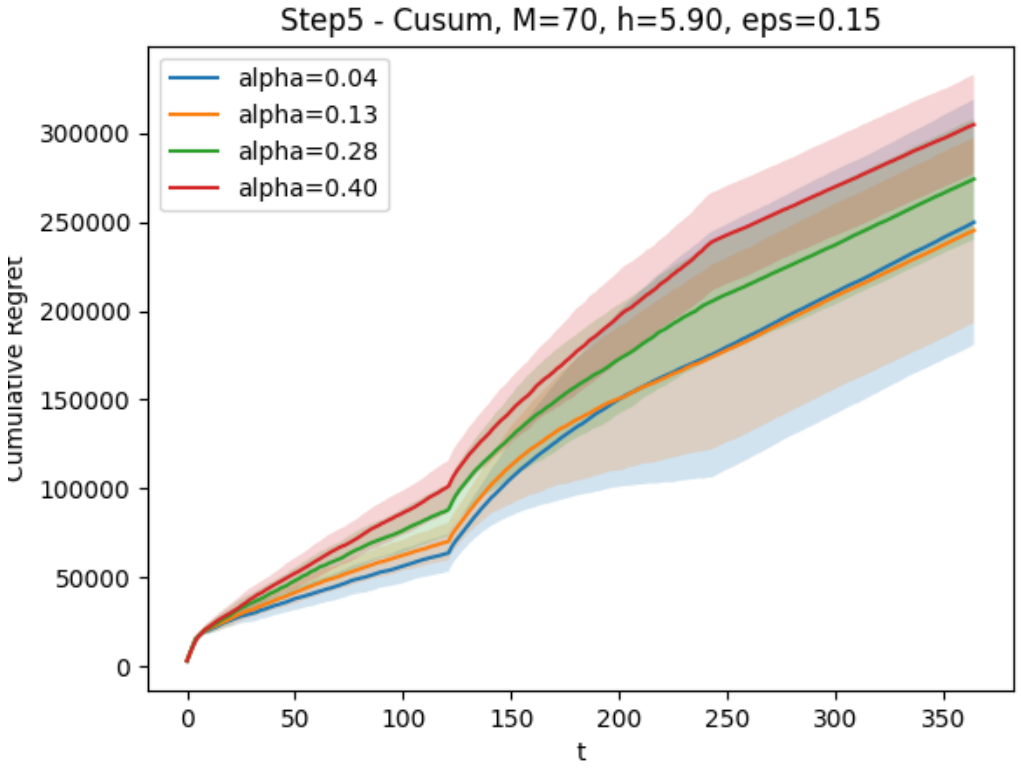
CUSUM: *h*



- *H* is a threshold to **detect any change** in the customers' behaviour. This means that **bigger** values of it require the changes to be more evident (i.e. more abrupt), while smaller ones represent a **lower** threshold to be passed to raise a change detection
- We tried 4 different values of *h* each one proportional to *log(*T) respectively with multiplying factors of 0.1, 1, 5 and 10.
- Both too-small and too-big values achieved **high regret** compared with a middle-way one, but also that the **lowest** one suffered much more regret than the higher ones (always with respect to the median one).

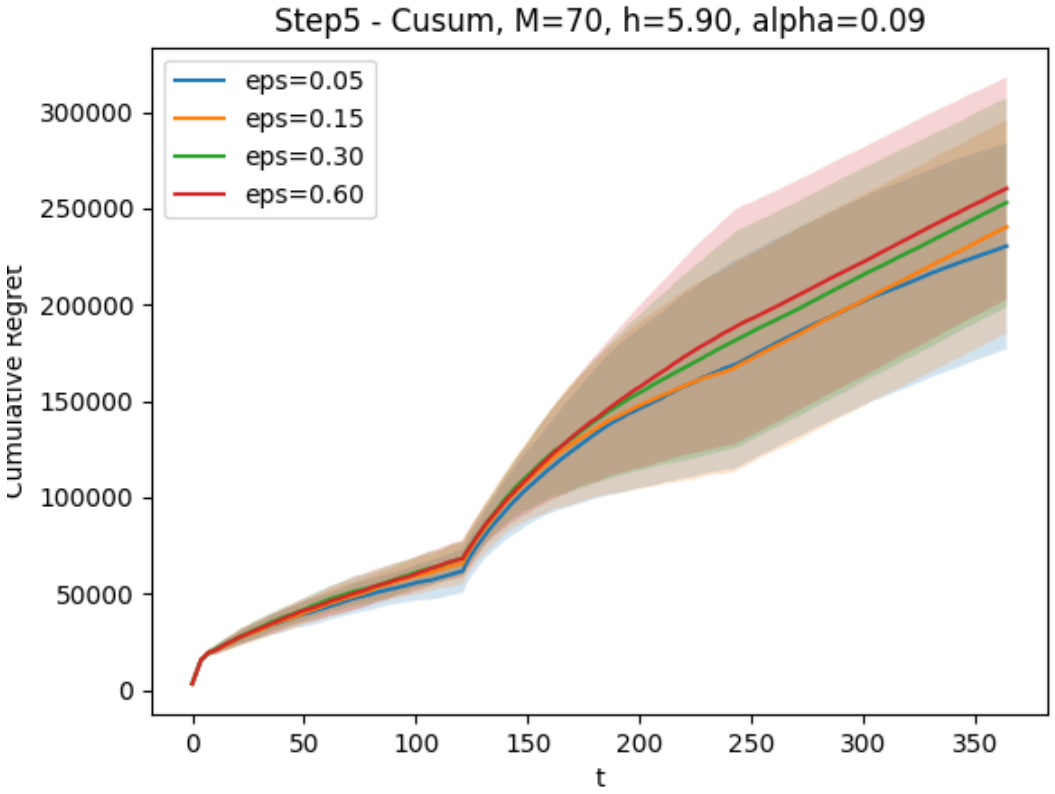
Step5 : Dealing with non-stationary environments with two abrupt changes - Sensitivity analysis

CUSUM: α



- α is the parameter of CUSUM directly responsible for the amount of **exploration** of the algorithm, indeed it represents the **probability of pulling a random arm** instead of the one suggested by the UCB check over the confidence bounds.
- Since it represents the exploration tendency of CUSUM, **higher values** allow us to detect changes happening at **higher frequency** but also increase the risk of pulling high-regret arms in place of the optimal one.
- We performed our tests over 4 different values for α , all proportional to $\sqrt{\frac{logT}{T}}$ and with various multiplying factors (specifically: $\sqrt{0.1}$, 1, $\sqrt{5}$, $\sqrt{10}$)
- We obtained that under a certain value, small α (0.04 and 0.13 for us) give all almost the **same regret**, while if they are increased (0.28 and 0.4), CUSUM accumulates **more regret**, as displayed in the subsequent chart.
- This matches what we expected from theory because the **higher values** of α force **more frequently** the pulling of random (and thus potentially sub-optimal) arms.

Step5 : Dealing with non-stationary environments with two abrupt changes - Sensitivity analysis



CUSUM: ε

- ε represents the **critical level** used to adjust the sensitivity of the change
- It is the value used to **adjust** the difference between the empirical mean and a new sample before it is added to the bound used for the change detection test
- Bigger values of ε mean **lower sensitivity** to the changes, that is **more samples** or samples with a **bigger difference** with respect to the empirical mean are needed to put the bounds over the threshold *h*
- The values we tried (0.05, 0.15, 0.3 and 0.6) were all proportional to what we expected to be the range of difference between a new sample (binary, 0 or 1) and the empirical mean (oscillating between 0 and 1
- The values we tried (0.05, 0.15, 0.3 and 0.6) were **all proportional** to what we expected to be the range of difference between a new sample (binary, 0 or 1) and the empirical mean (oscillating between 0 and 1)

Step6 : Dealing with non-stationary environments with many abrupt changes - Request

Two settings:

1. In the first one, the environment is **non-stationary**. It is a simplified version of Step5, with the **bid fixed**.

- Develop and apply the **EXP3 algorithm**, and verify that it performs **worse** than the two non-stationary versions of UCB1.

1. In the second one, the environment has a **higher degree of non-stationarity** with 5 phases that frequently change.

- Apply **EXP3**, **UCB1**, and the **two non-stationary flavors** of UCB1. Verify that EXP3 **outperforms** the non-stationary version of UCB1.

Step6 : Dealing with non-stationary environments with many abrupt changes - Solution

- A scenario with **many abrupt changes** poses **harder obstacles** in the learning process. In the setting proposed the bid is fixed and therefore we select a value of the bid which is the optimal value of 2.79€ computed in the previous steps. This choice allows us to have **comparisons** with previous results. The parameters we need to estimate to **maximize** the monetary reward are the **conversion rates** which are unknown and rapidly changing.
- Such rapidly changing scenario is called an **adversarial bandit**. It differs from the classical bandit problem for the changes in the quantities that for a classic bandit are fixed and learned over time. In fact in an adversarial bandit setting we **cannot** expect to learn quantities changing over time. We can only try to play a good arm without the hope to learn anything useful from its reward.

Step6 : Dealing with non-stationary environments with many abrupt changes - Solution

- The EXP3 algorithm proposed is an algorithm designed to play in an **adversarial bandit setting**.
- At each round the arm to be played is selected by a **random draw** from a probability distribution over each arm computed at the previous step. The probability distribution of arm i at round t is computed by:

$$p_i(t) = (1 - \gamma)\frac{w_i(t)}{\sum_{j=1}^{k} w_j(t)} + \frac{\gamma}{K}$$

where *K* is the number of arms, $\gamma \in (0,1)$ is a **hyperparameter** to tune, and $w_i(t)$ is the weight associated with arm *i* at round *t* and computed as:

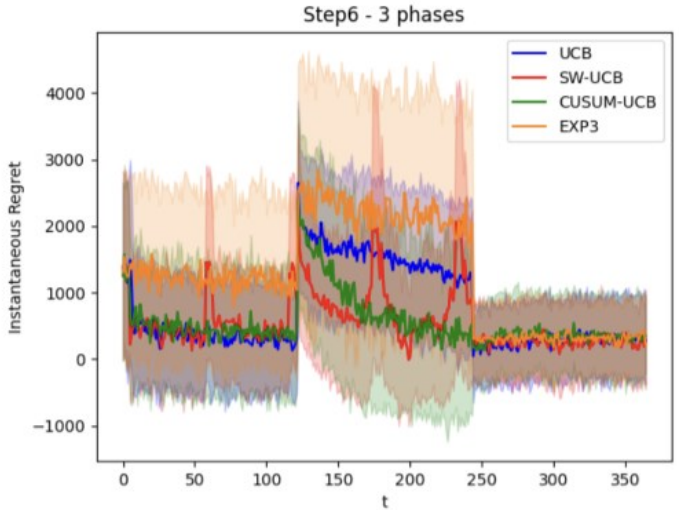
$$w_{i_t}(t+1) = w_{i_t}(t) \cdot e^{\frac{\gamma}{K}\hat{x}_{i_t}}$$

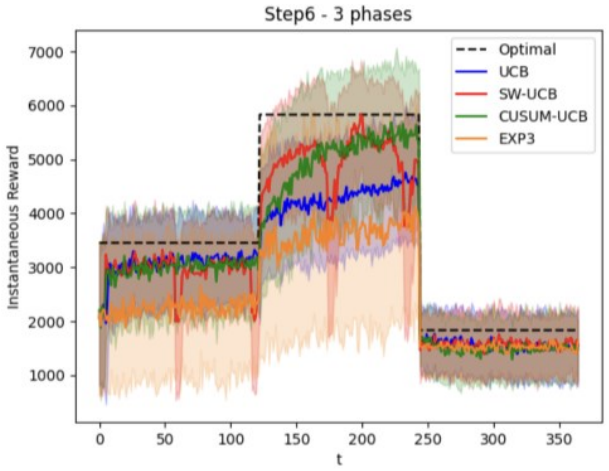
where i_t is the arm pulled at time *t* and $\hat{x}_{i_t}$ the expected reward obtained from that arm at time *t*: $\hat{x}_{i_t} = \frac{x_{i_t}}{p_{i_t}}$.

Step6 : Dealing with non-stationary environments with many abrupt changes - Solution

- Weights are **initialized** to 1 for each arms, and rewards are **scaled** to the interval [0, 1].
- γ must be **tuned**. If it is closer to 1, the arms tend to be uniform, while if it is closer to 0 the algorithm gives more probability according to the weights and therefore according to the rewards obtained from the game.
- EXP3 being an adversarial bandit algorithm presents, as said before, **learning inclinations**. In fact when an arm is selected its weight increases if the reward is high and by consequence the probability of drawing it is higher. Compared to classical bandit, such as TS or UCB, EXP3 explores more between all the arms. This behavior is needed to **compensate** for the lack of fixed rewards in time.

Step6 : Dealing with non-stationary environments with many abrupt changes - Results (Low Frequency)

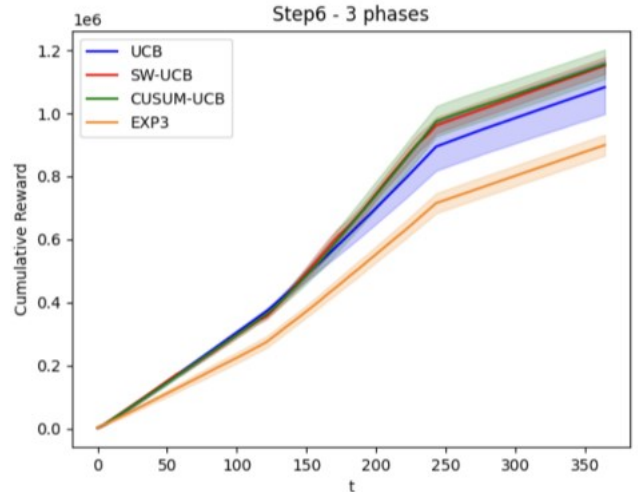


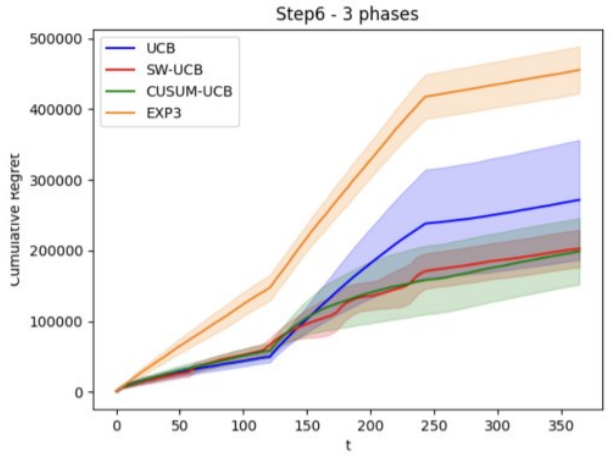


- We analyse the performance of EXP3 in the Step5 setting, where there are 2 **abrupt changes**.
- As we expect the EXP3 algorithm **does not perform well** in a setting where there are only 2 changes over 365 rounds
- In fact, the regret we observe is **linear** in all three phases. This is in line with the fact that EXP3 even in a single-phase scenario has chances to draw each arm even if a high weight is assigned to the best arm.

- By looking at the instantaneous reward and the instantaneous regret it can be noted that the learning velocity of EXP3 is very slow compared to UCB,SW-UCB and CUMSUM-UCB.

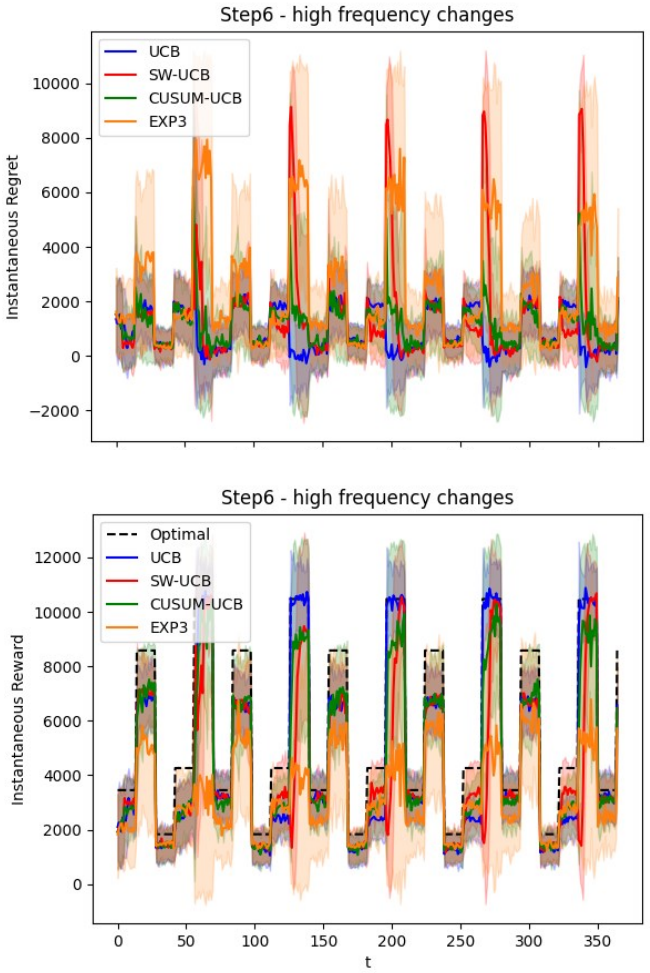
Step6 : Dealing with non-stationary environments with many abrupt changes - Results (Low Frequency)





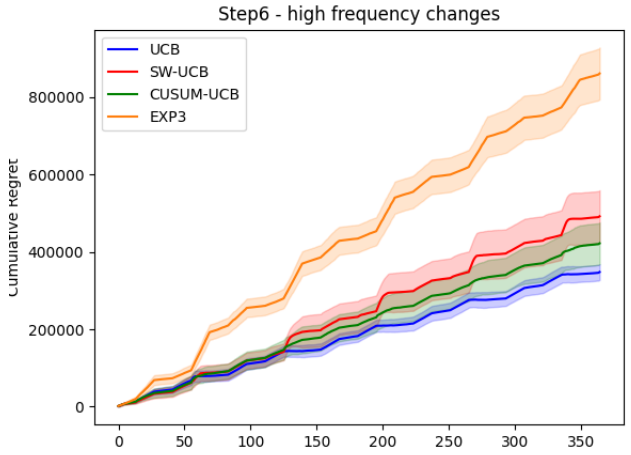
- By looking at the **probability distribution** assigned to the arms over the rounds we observed **it does not evolve much** from the uniform distribution assigned at the beginning, even for small values of gamma. With this knowledge, it can be explained a **slow learning rate** for EXP3.

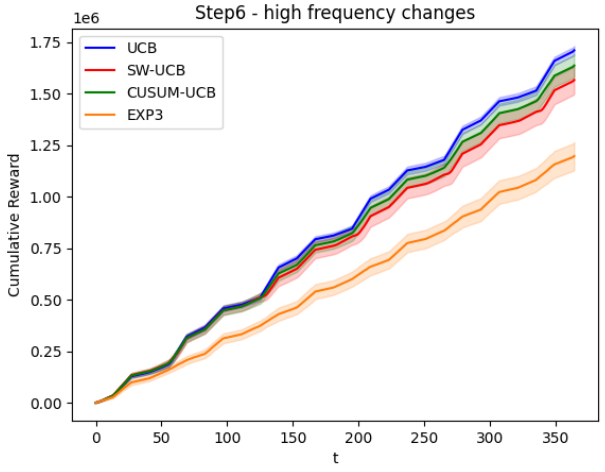
Step6 : Dealing with non-stationary environments with many abrupt changes - Results (High Frequency)



- We challenge EXP3 in a setting with **high-frequency changes**. We created 5 different conversion rates, 3 of which are the same as the previous scenario.
- Each phase has a **different** optimal arm.
- The results we found are that Exp3 **does not perform better** than the other methods. In fact, the EXP3 algorithm has theoretical good performance in expectation when the reward is between 0 and 1 with extreme values achievable in practice.
- This is **not our case** being the learning reward is the conversion rates times the selling price. To be able to use EXP3 algorithm we need to **normalize** the reward.
- To limit the reward between 0 and 1 the **rescale factor** takes into account a maximal theoretical reward of 1 for the conversion rate multiplied by the maximum for all prices, which means rescaling by 1 * 250

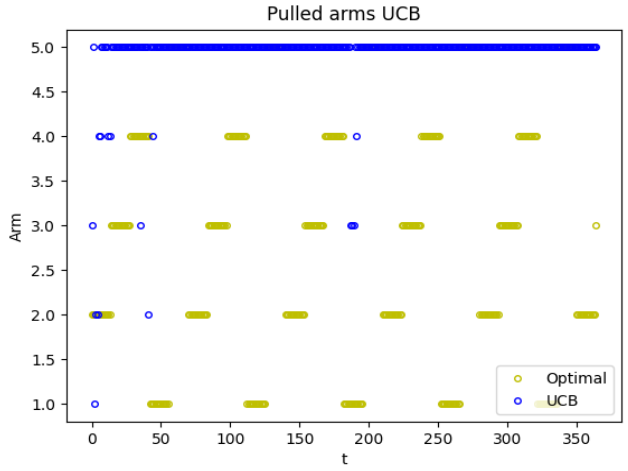
Step6 : Dealing with non-stationary environments with many abrupt changes - Results (High Frequency)

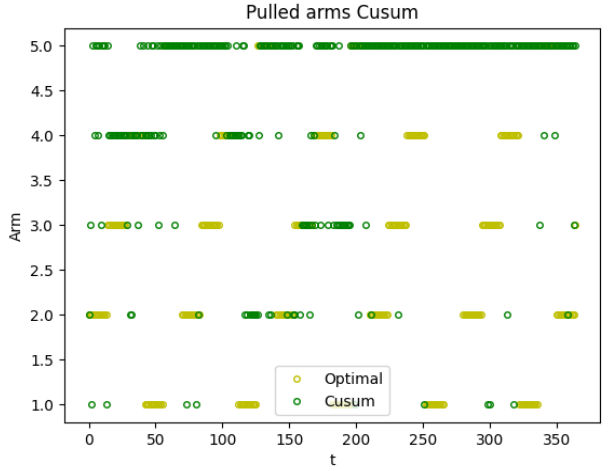


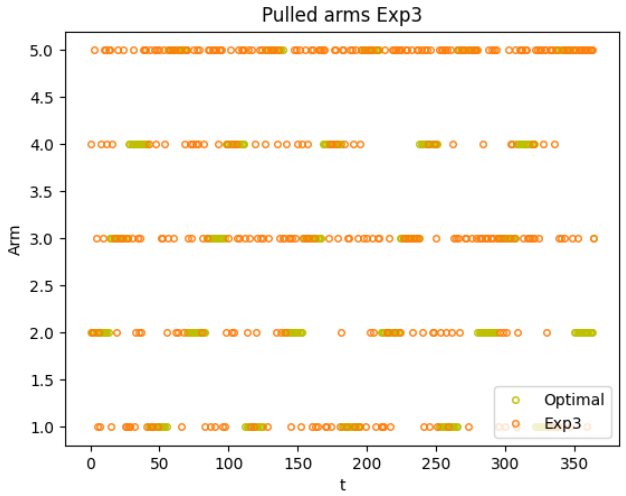


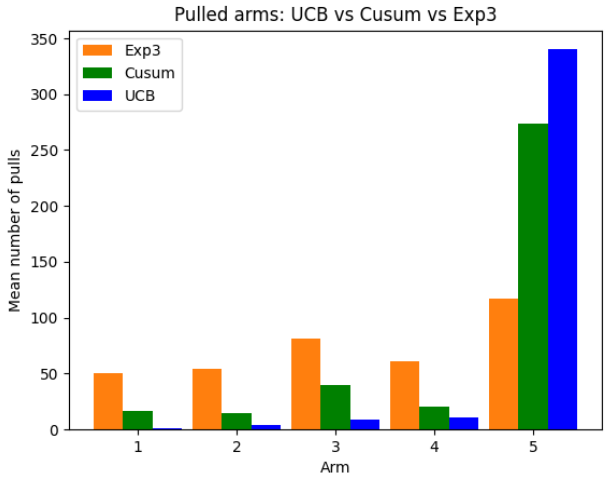
- We note also that the other algorithms have **comparable performances** in such scenario.
- **None** of the algorithms reaches a sublinear regret but linear as we would expect in such instances of high-frequency changes.
- By having a linear regret we can verify that in such a scenario the cumulative reward is of the order of magnitude of the cumulative regret.

Step6 : Dealing with non-stationary environments with many abrupt changes - Results (High Frequency)









Thanks for your attention!

Lorenzo Ferretti	lorenzo2.ferretti@mail.polimi.it	10713719
Nicolò Fontana	nicolo.fontana@mail.polimi.it	10581197
Carlo Sgaravatti	carlo.sgaravatti@mail.polimi.it	10660072
Marco Venere	marco.venere@mail.polimi.it	10865088
Enrico Zardi	enrico.zardi@mail.polimi.it	10659549

Professors:
Castiglioni Matteo,
Gatti Nicola,
Bernasconi de Luca Martino

A.Y. 2022/2023