
# Online Learning Applications

## Pricing and Advertising

**A.Y. 2022-2023**

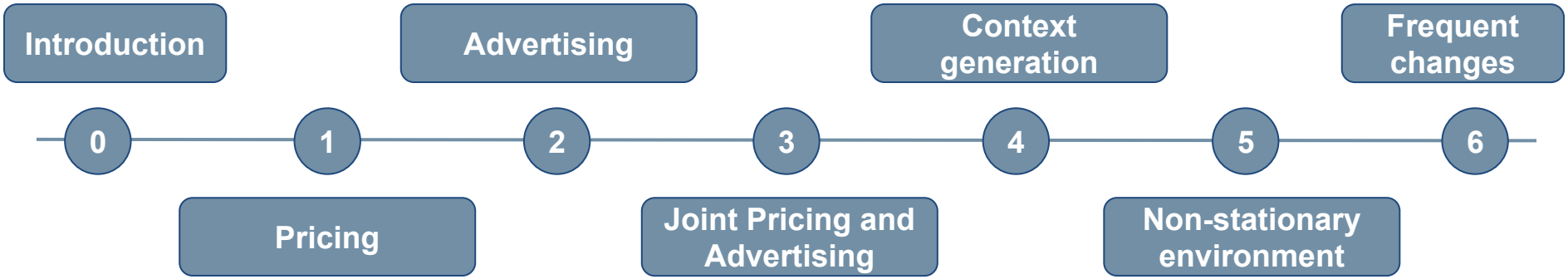
Professors Castiglioni Matteo, Gatti Nicola, Bernasconi de Luca Martino

POLITECNICO
MILANO 1863

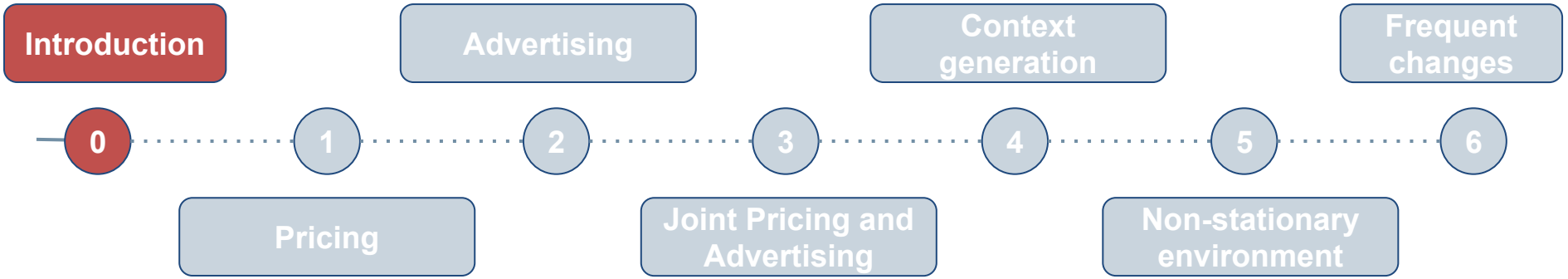

| | |
|---|---|
| **Pertino Paolo** | **10729600** |
| **Pesce Leonardo** | **10659489** |
| **Sandri Alberto** | **10698469** |
| **Simionato Enrico** | **10698193** |
| **Vitali Michael** | **10730463** |

Professors Castiglioni Matteo, Gatti Nicola, Bernasconi de Luca Martino

# Roadmap


Introduction
Advertising
Context generation
Frequent changes
0
1
2
3
4
5
6
Pricing
Joint Pricing and Advertising
Non-stationary environment

# Roadmap


Introduction
Advertising
Context generation
Frequent changes
0
1
2
3
4
5
6
Pricing
Joint Pricing and Advertising
Non-stationary environment

# Step 0: Motivations and environment design
# Introducing iPear: e-commerce smartphone debut

*iPear*, a **burgeoning e-commerce company**, is ready to make waves with the launch of its very first smartphone. With a **strong focus on technology** and **innovation**, *iPear* aims to redefine the **smartphone market**.

*iPear*'s debut **smartphone** promises to be a game-changer, merging technology, innovation, and user-centric design.
Its journey is one worth following closely, showcasing the transformative **potential of AI in the world of e-commerce** and **consumer electronics**.

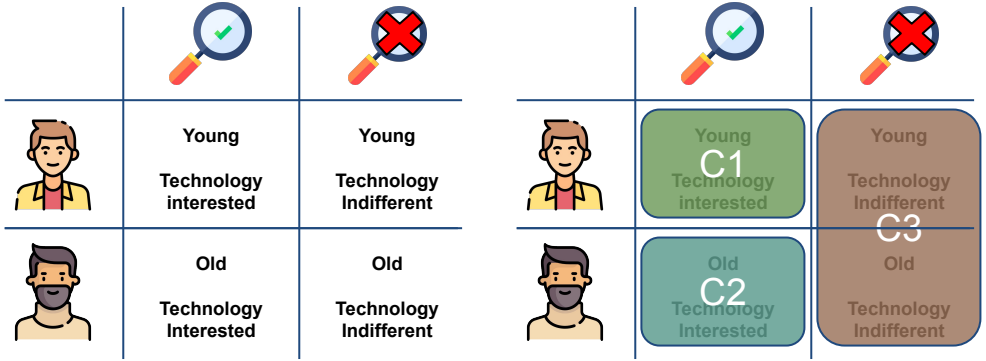
*iPear* team is strictly controlling both the **pricing** and **advertising strategies** for the upcoming year (365 days) after the launch of their brand new smartphone: ***iPear ArmOptima***.

The company is **strategically positioning its smartphone** in the **middle-level market segment**, aiming to strike a balance between affordability and advanced features to cater to a wide range of consumers. In fact, the **pricing schema** defined, ranges from **500€** to **700€** with equal spacing of **50€**.

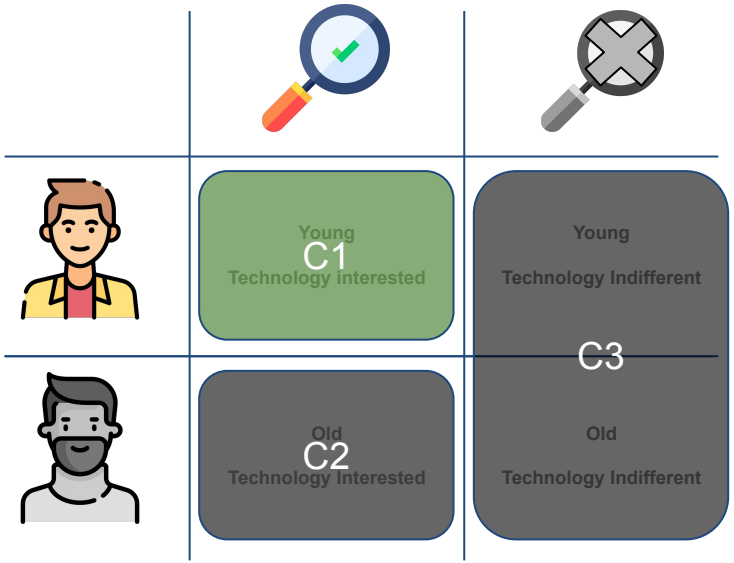
# Step 0: Motivations and environment design
# iPear - Introduction

The research team of the company is looking forward to launch an **advertising campaign** and has determined to **categorize users** into **three** distinct **classes** (**C1**, **C2**, **C3**) that differ in terms of both **click frequency** and **cumulative daily costs** as the bid varies.

In its first version, the advertising platform is able to **characterize users** by exploiting two **binary features F1** and **F2**. **F1** represents the **age of the user**, i.e. it explains whether the consumer is **old** (value 1) **or young** (value 0). **F2**, instead, indicates whether the **customer** has **recently looked for** a **technology-related item** on his/her browser (value 1) or not (value 0).



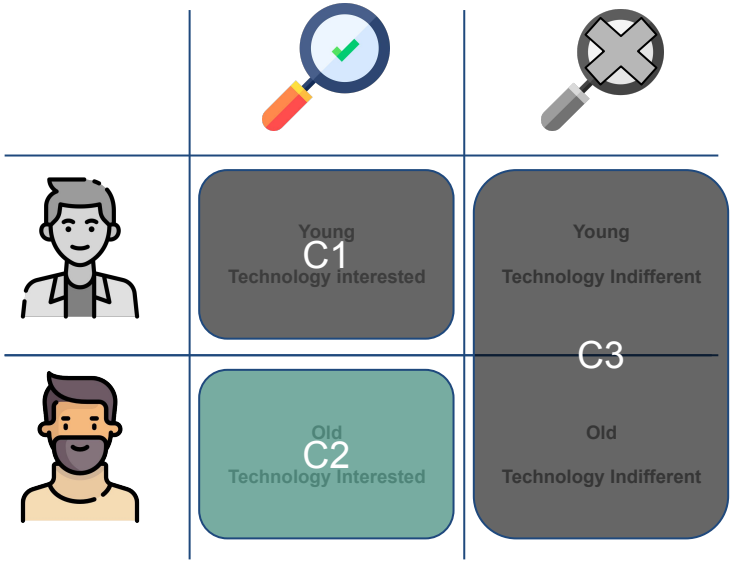
# Step 0: Motivations and environment design
# iPear - Users' classes - C1: Young Technology Enthusiasts




Users of **class C1**, primarily **comprising younger individuals** with **great interest in technology**, are expected to possess relatively **moderate purchasing power**. They are likely to allocate a significant portion of their income or savings toward the latest tech products and innovations, making them an attractive target for tech-related advertising campaigns.
Moreover, being **digitally passionate individuals**, they are **more likely to click on ads related to tech products** that align with their interests and preferences. Therefore, when targeting Class C1, one can **expect** a **relatively higher click-through rate compared to other user segments**.

With that being said, the **conversion rate** of this class **peaks at 600€**, while the number of clicks is expected to be the highest and to **saturate** at around **100 clicks**. Since the number of clicks is the highest, the **highest effort** in terms of **cumulative daily cost** is made.

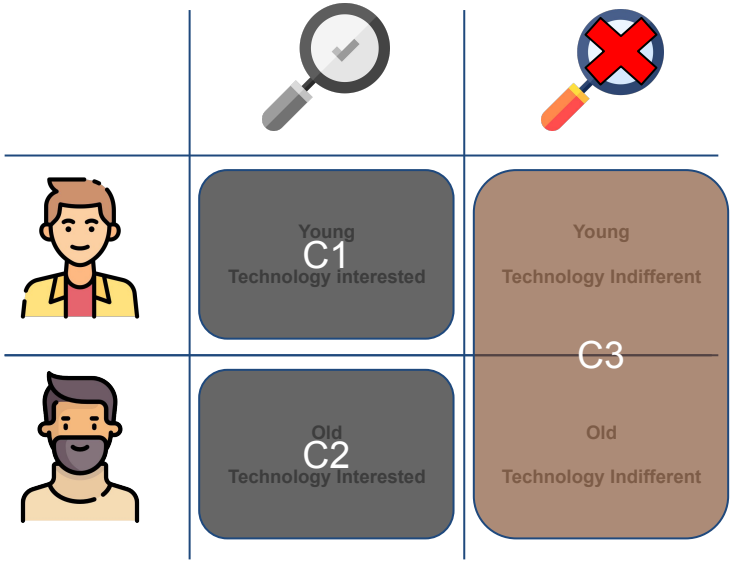
# Step 0: Motivations and environment design
# iPear - Users' classes - C2: Old Technology Enthusiasts




**C2 users**, who fall into the **older age group** and share an **enthusiasm for technology**, typically **exhibit stable** and **higher purchasing power** with respect to other classes. They are generally more discerning consumers who prioritize quality and reliability in their tech purchases.
Customers of this class may click on tech-related ads that align with their interests and preferences. However, the **number of clicks** from **Class C2** users is **generally moderate**, and usually **smaller compared** to the **young** more tech-oriented **individuals**.

Their **conversion rate peaks** at **700€**, while maintaining a quite high number of clicks as the bid varies; value that **saturates** at around **90 clicks**.
The research team thinks that it is still worthy investing in reaching Class C2 due to their high purchasing power and the potential for meaningful engagement.

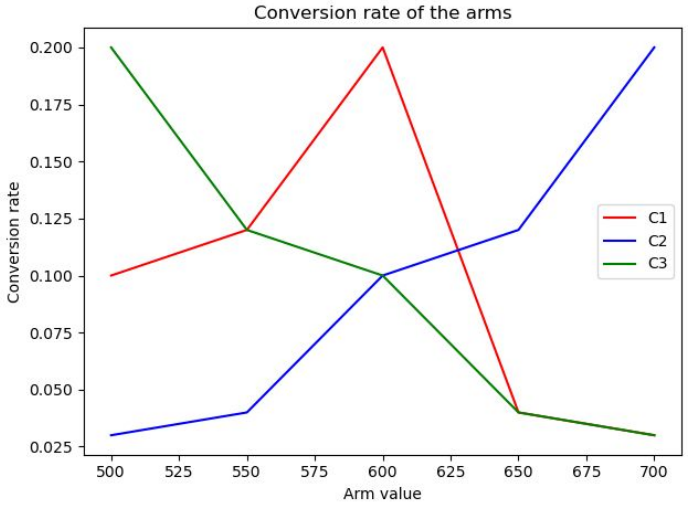
# Step 0: Motivations and environment design
# iPear - Users' classes - C3: Technology Apathetic



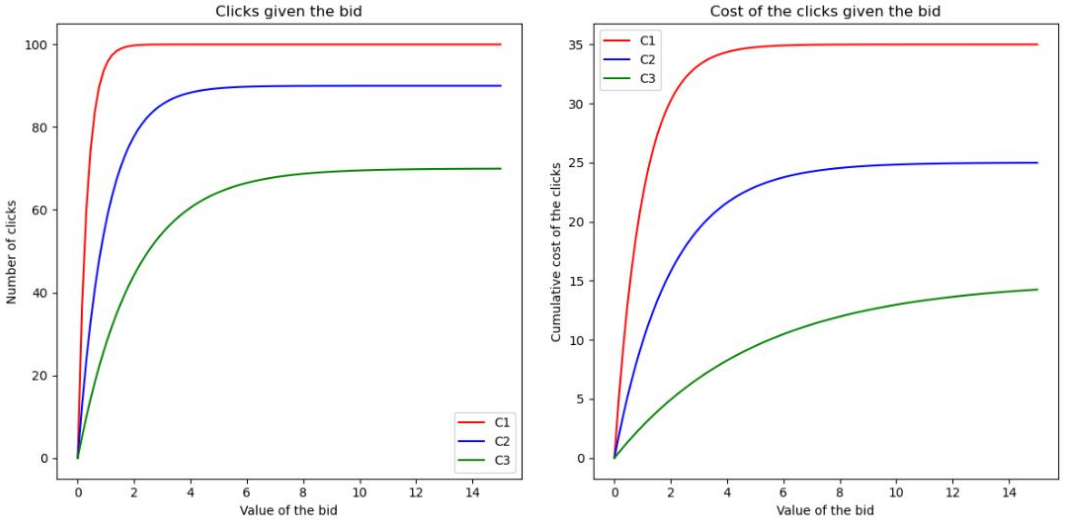

**Class C3** represents a group of individuals composed **both** by **young and old people** who generally **do not exhibit** a particular **interest in technology**.
Their **click-through rate** tends to be **more conservative** compared to tech enthusiasts. These users are less likely to actively engage with online advertisements related to technology products or services and thus **little effort** and **few resources** are **allocated** to this class.

Their **conversion rate peaks** at **500€**, and their **clicks** curve **saturate** at around **70 clicks**.

# Step 0: Motivations and environment design
# iPear - Users' classes - Technicalities

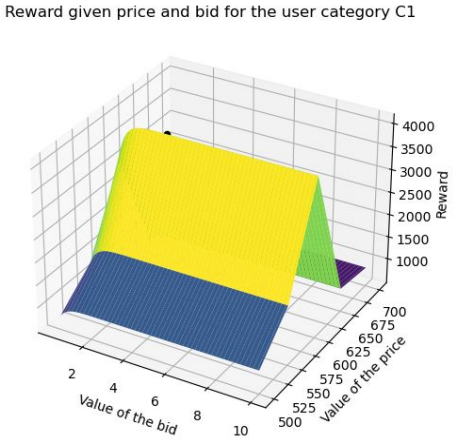


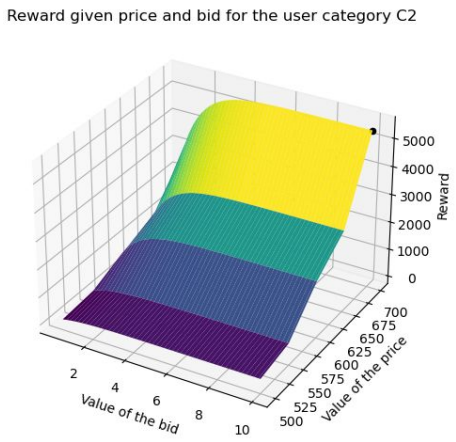

[A1] Conversion probabilities for the three classes

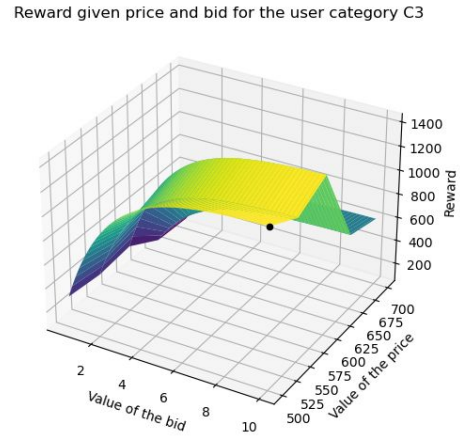


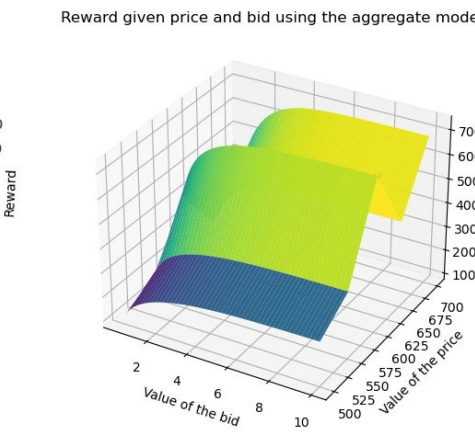

[A2] Clicks and cumulative daily costs curves

# Step 0: Motivations and environment design
# iPear - Users' classes - Technicalities cont.



# Step 0: Motivations and environment design
# iPear - Clairvoyant

The research team is **testing** the **advertising platform**, in the simulation described, by **comparing** its **results** in terms of **reward/regret against** the performances given by a **clairvoyant algorithm**, i.e. a perfect machine with **comprehensive knowledge** about both the **pricing** and **advertising curves**.

Due to its unmatched knowledge, the Clairvoyant **always plays** the **best possible price** and **bid**, thus it always **maximize perfectly** the **reward** (**profit**), which is given by:

$$reward = \#_{clicks} * \underbrace{\text{conversion probability} * (price - \text{other costs})}_{\textit{conversion\_times\_margin}} - \text{cumulative daily costs}$$

This algorithm works as follows:

***Input***: *category*

1. *best_price*, *conversion_times_margin* ← *maximization of reward from price given category*
2. *best_bid*, *reward* ← *maximization of reward from bid given category and conversion_times_margin*

***Output***: *best_price*, *best_bid*, *reward*

# Step 0: Motivations and environment design
# iPear - Users' classes - Clairvoyant cont.

***Input**: category*

1. *best_price, conversion_times_margin ← maximization of reward from price given category*
2. *best_bid, reward ← maximization of reward from bid given category and conversion_times_margin*

***Output**: best_price, best_bid, reward*

**1.**

**Exhaustive search over** the list of **prices** that **returns** the **price** which **maximize** the

***conversion probability * (price - other costs)***

quantity (note that the conversion probability is known since the clairvoyant is omniscient, thus it knows the pricing curves and the environment parameters).

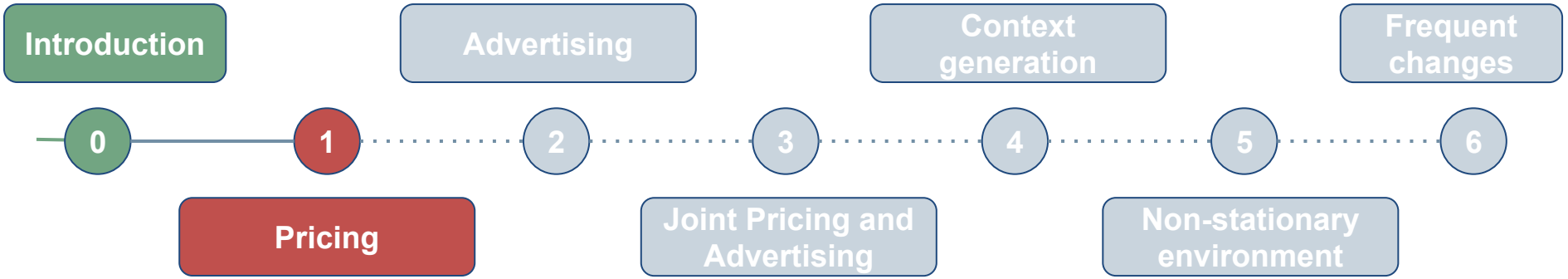
**2.**

**Exhaustive search over** the list of **bids** that **returns** the **bid** which **maximize** the

$\#_{clicks}$ **** conversion_times_margin - cumulative daily costs***

quantity (note that the $\#_{clicks}$ and the cumulative daily costs are known since the clairvoyant is omniscient, thus it knows the advertising curves).

# Roadmap


Introduction
Advertising
Context generation
Frequent changes
0
1
2
3
4
5
6
Pricing
Joint Pricing and Advertising
Non-stationary environment

# Step 1: Learning for pricing

## Setting

In this step the **pricing setting** is **unknown**, instead the **advertising** one is **known**. The **objective** is to **maximize the reward** while estimating the purchase conversation rates.

**Scenario:**

- All the users belong to **class C1**
- Curves related to the **advertising part** are **known**
- Curve related to the **pricing part** is **unknown**

**Task:**

- Apply **UCB1** and **TS** to tackle the problem
- Plot the average value and standard deviation of the **cumulative regret**, **cumulative reward**, **instantaneous regret** and **instantaneous reward**

# Step 1: Learning for pricing

## Approach

In this scenario we use **TS** and **UCB1** to minimize the regret while estimating the **conversion probabilities** associated to each **price**, so that it is possible to understand which is the best one.

Since in the pricing setting the functions of the number of clicks and cumulative daily costs are known, it is possible to use the **optimal bid** given the price. At each round, for each possible price, the best bid is computed exploiting the clairvoyant and then the associated number of clicks and daily cost are calculated.

Given these data, the **arm** is **pulled** by choosing the one that **maximizes** the **reward** formula where the **conversion probability** is substituted by:

- a **sample drawn** from the **Beta** distribution for **TS**
- the **upper confidence bound** for **UCB1**

Successively, each learner is updated using the data collected, namely the total reward for the day and for each user the interaction with the website, that is a 1 if there is a purchase, otherwise 0.

# Step 1: Learning for pricing

## Approach

The **updates** for the learners are:

- **TS**: $(\alpha_{a_{t+1}}, \beta_{a_{t+1}}) \leftarrow (\alpha_{a_t}, \beta_{a_t}) + (c_{1,t}, c_{0,t})$
  where:
  - $(\alpha_{a_t}, \beta_{a_t})$ are the beta parameters for arm *a* at time *t;*
  - $c_{1,t}$ is the number of conversions at time *t;*
  - $c_{0,t}$ is the number of people who click but don't purchase at time *t*.

- **UCB**: $UCB \leftarrow \bar{x}_a + \sqrt{\frac{2 \log t}{n_a(t)}}$
  where:
  - $\bar{x}_a$ is the estimate of the conversion probability and it is given by the total number of conversions for arm *a* until this point divided by the total number of observations for the arm *a;*
  - $n_a(t) \leftarrow n_a(t-1) + c_{1,t} + c_{0,t}$ is the total number of clicks collected by arm *a* until time *t.*

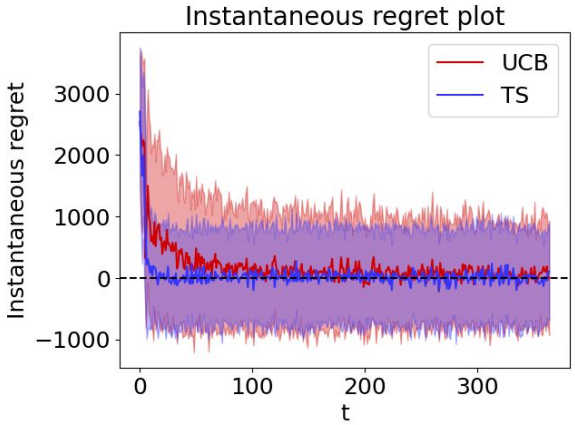
# Step 1: Learning for pricing

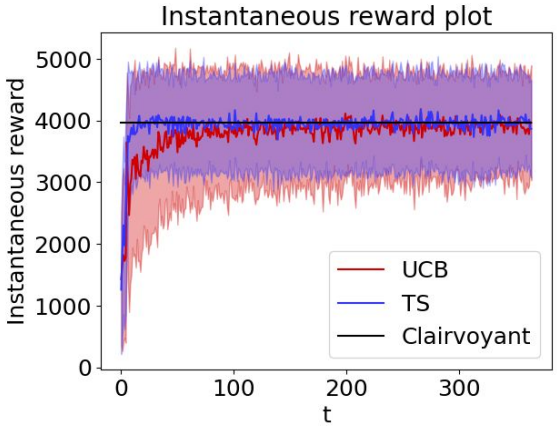
## Results

The following plots are obtained by launching **100 experiments**.

In this case **TS** performs much better since in few days understands which one is the best arm.

It **pulls** the **best arm** the majority of the time because it tries all the arms in the initial days and can estimate the conversion probability well as it receives roughly 100 samples for the pulled arm in a single day. The conversion probability is also well estimated by **UCB**, but it more frequently plays sub-optimal arms during the first rounds when the upper bounds are larger.

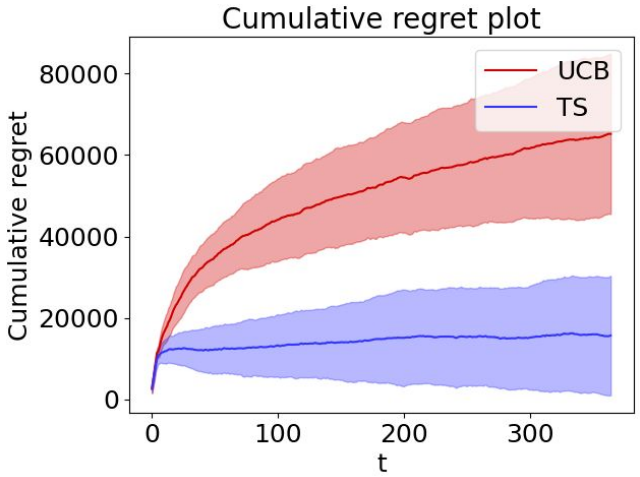


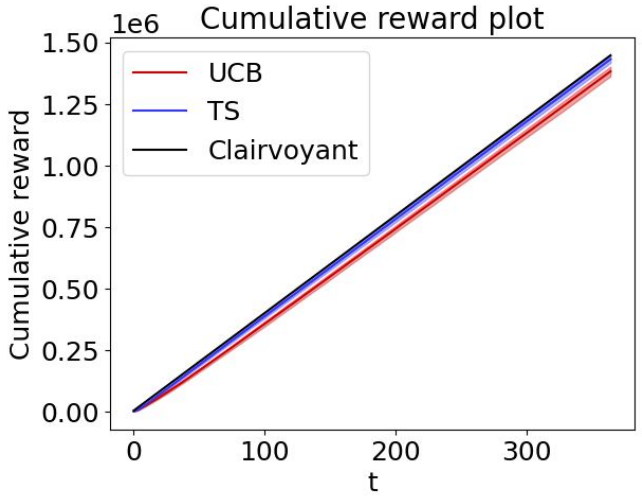



# Step 1: Learning for pricing

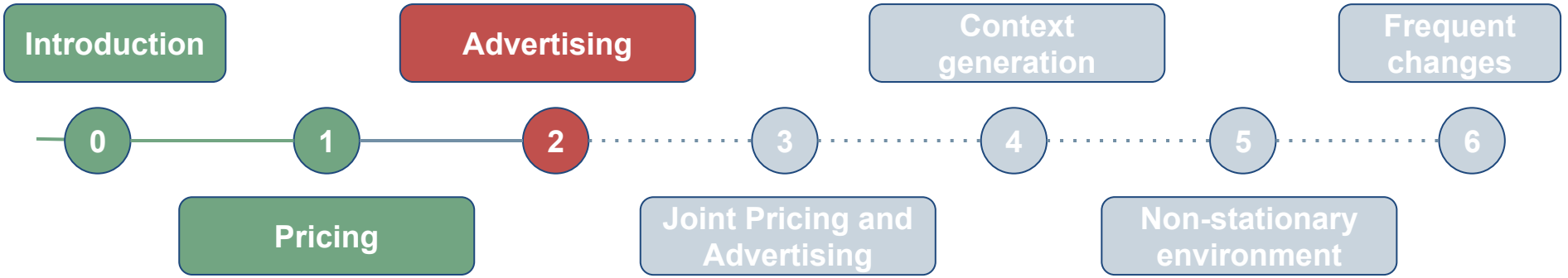
## Results

It is clear that both the learners achieve a **sub-linear cumulative regret** as expected from their theoretic bounds. In this case **TS** performs much better since in few days understands which one is the best arm.
It is possible to observe that the cumulative regret of **UCB** is around **4 times higher** than the one obtained by **TS**.



# Roadmap


Introduction
Advertising
Context generation
Frequent changes
0
1
2
3
4
5
6
Pricing
Joint Pricing and Advertising
Non-stationary environment

# Step 2: Learning for advertising

## Setting

In this step the **advertising setting** is **unknown**, instead the **pricing** one is **known**. The **objective** is to **maximize** the **reward** while estimating the number of clicks and cumulative daily costs.

**Scenario:**

- all the users belong to the **class C1**
- the **curve** related to the **pricing problem** is **known**
- the **curves** related to the **advertising problem** are **not known**


**Task:**

- application of **GP-UCB1 algorithm** and **GP-TS algorithm** to maximize the reward while estimating the clicks and cumulative cost curves of the advertising problem
- plot the average value and standard deviation, over a sufficiently large number of runs, of the **cumulative regret**, **cumulative reward**, **instantaneous regret** and **instantaneous reward**

# Step 2: Learning for advertising
## Assumptions

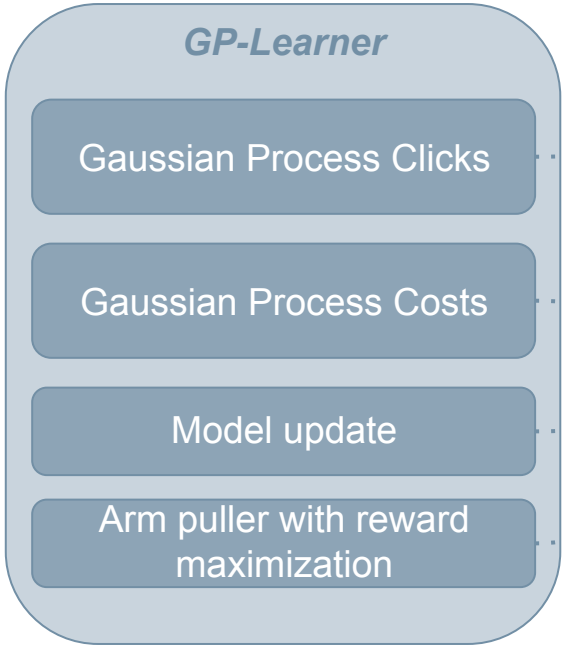
In the advertising setting, the pricing curve is fully known, thus the **price** is considered to be **fixed** and **set to be the best one** in terms of **conversion rate times margin**.

In this context, the aim is maximize the reward while giving an estimate of two curves which respectively describe the number of clicks and the cumulative costs as the bid varies.

To give these estimates we used **Gaussian Processes** (GP), because they allow to tackle the regression problem with mild assumptions on the function to estimate and, most importantly, because they also **return**, together with the average value learnt from observations, a **measure of the uncertainty** of the result of the regression, i.e. the output is a probability distribution over the outcome.
This fact is crucial to assure the **convergence** of bandit algorithms.

Since estimating the clicks and cumulative costs curves for every possible bid might take too long time, we assume that there is **correlation between arms** (i.e. values of the bids) **that are close** each other. This requires the two curves to be sufficiently **smooth**.

# Step 2: Learning for advertising
## Component structure

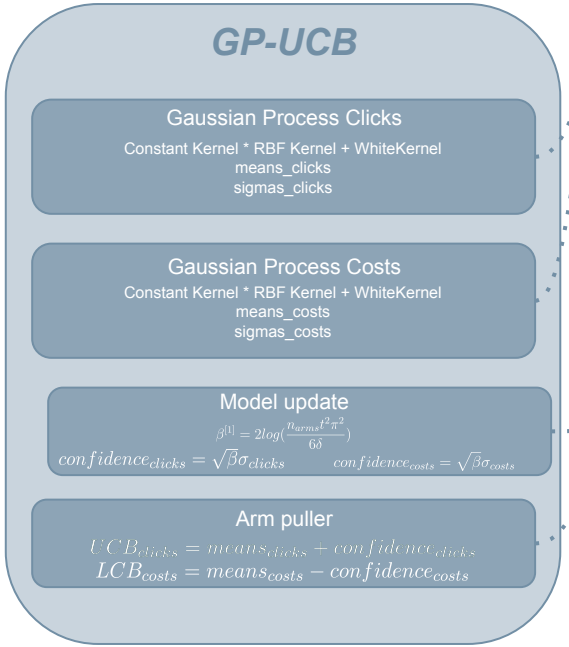



The **general structure** of our learners based on Gaussian processes is the following:

- a gaussian process is used to model the number of clicks given the bids, so provided the observations it gives the mean value of the function describing the clicks and a measure of the uncertainty of its estimate
- another gaussian process is deployed to model the cumulative daily costs curve given the bids
- a module is dedicated to the update of the GP parameters whenever a new observation is available.
- to pull an arm (i.e. choose a bid), the reward $r_{bid} = \#_{clicks_{bid}} * conversion\_prob * (price - other\_costs) - cum\_daily\_costs_{bid}$ is maximized and the corresponding bid is taken.

$$pulled\_bid = argmax_{bid}\{r_{bid}\}$$

# Step 2: Learning for advertising
# Component structure - GP-UCB1




A **Gaussian Process** is used, given the observations, to estimate the **mean value** of the **clicks curve** together with a **measure of the uncertainty** it has in doing so. Another GP, on the other hand, does the same for the **cumulative daily cost curve**.

Whenever a **new sample is obtained**, the **gaussian processes are updated** (i.e. it is trained with the past samples and the new one), the $\beta$ [1] parameter is calculated and finally the **confidence interval**, which is given by

$$confidence = \sqrt{\beta}\sigma$$

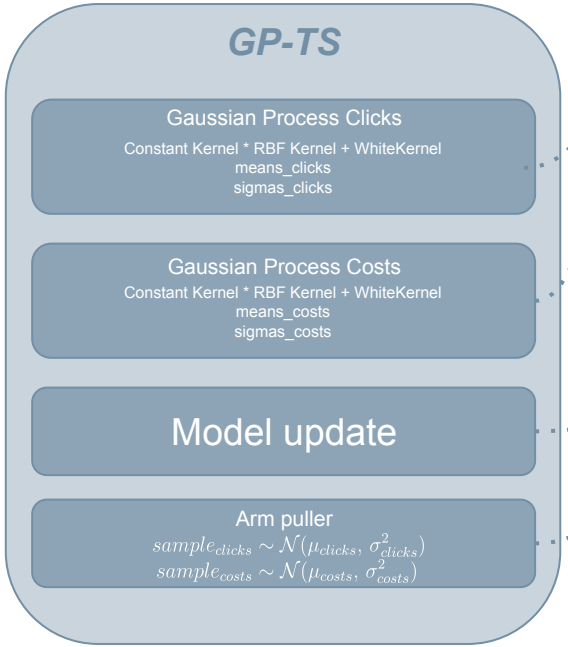
Once-per-day an arm (i.e. a bid) is pulled. The **upper confidence bound of the clicks** and the **lower confidence bound for the costs** is calculated, in order to finally **calculate and maximize the reward**, and play the corresponding bid.

$$UCB_{clicks} = means_{clicks} + confidence_{clicks} \qquad LCB_{costs} = means_{costs} - confidence_{costs}$$

$$r_{bid} = UCB_{clicks} * prob\_times\_margin - LCB_{costs} \qquad pulled\_bid = argmax_{bid}\{r_{bid}\}$$

[1] Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design, Srinivas et al. (2010), https://arxiv.org/pdf/0912.3995.pdf

# Step 2: Learning for advertising
## Component structure - GP-TS




A **Gaussian Process** is used, given the observations, to estimate the **mean value** of the **clicks curve** together with a **measure of the uncertainty** it has in doing so. Another GP, on the other hand, does the same for the **cumulative daily cost curve**.

Whenever a **new sample is obtained**, the **gaussian processes are updated** (i.e. they are trained with the past samples and the new one).

Once-per-day an arm (i.e. a bid) is pulled. In the case of **Thompson Sampling** the **number of clicks** and the **cumulative daily cost** to use to calculate the reward **are drawn from a Gaussian Distribution** whose **mean and standard deviation** parameters are the ones **estimated by the GPs**.

$$sample_{clicks} \sim \mathcal{N}(\mu_{clicks}, \sigma^2_{clicks}) \qquad sample_{costs} \sim \mathcal{N}(\mu_{costs}, \sigma^2_{costs})$$

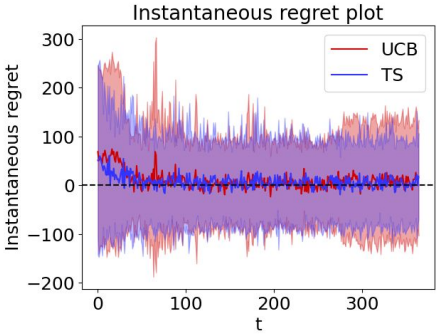
$$r_{bid} = sample_{clicks} * prob\_times\_margin - sample_{costs} \qquad pulled\_bid = argmax_{bid}\{r_{bid}\}$$

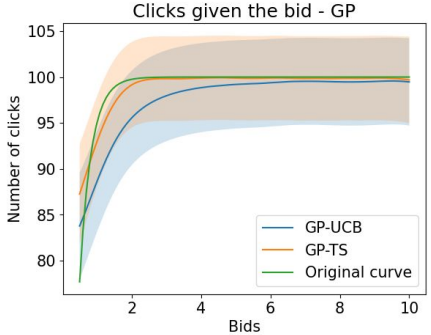
# Step 2: Learning for advertising
# Results

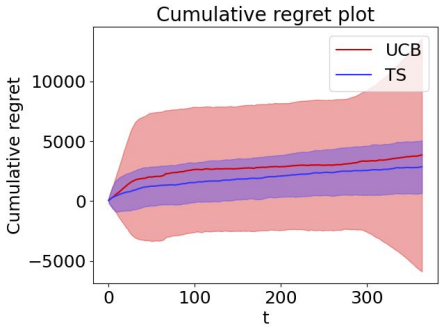
We ran **100 experiments** with a time interval of an year each.

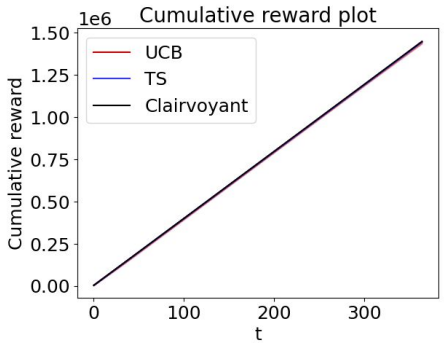
As we can see both the **clicks** and the **cumulative daily costs curves** have been **approximated quite well** by the Gaussian Processes of the learners.

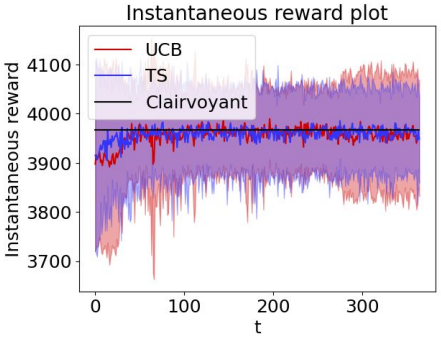
Here the plots of the **instantaneous reward/regret** and the **cumulative reward/regret** of both the algorithms used.

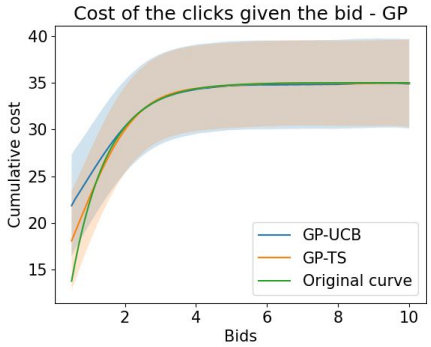


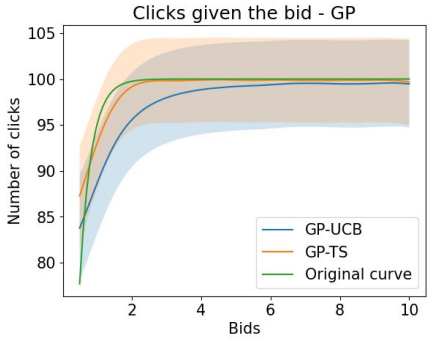



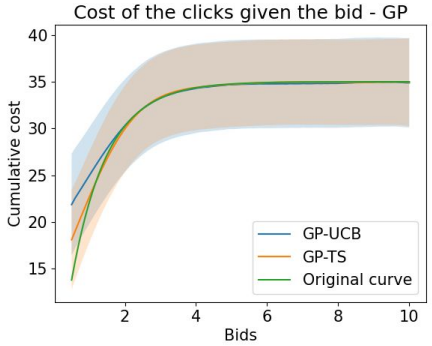



# Step 2: Learning for advertising
## Results

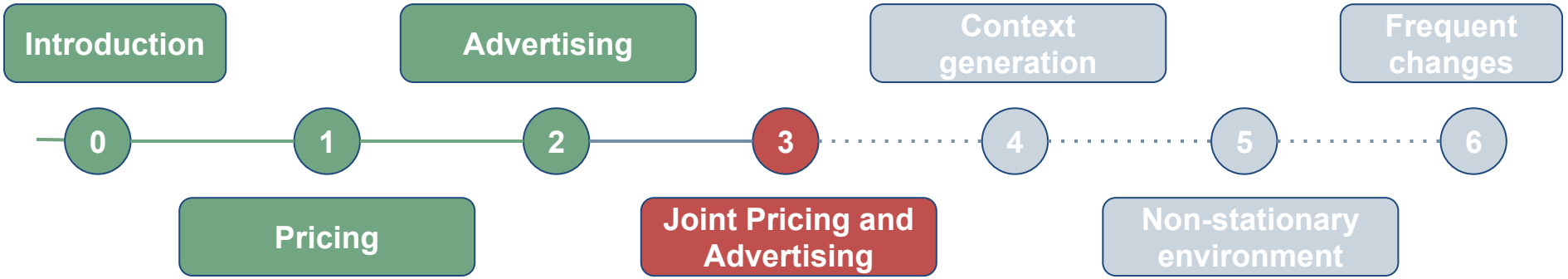
As we can see from the previous plots, **both algorithms exhibit sub-linear regret**, in alignment with their theoretical guarantees, and they achieve this with a **notable speed of convergence**.
Also in this step, Thompson Sampling is outperforming UCB1 in terms of regret minimization; nevertheless they both approach the best solution for the problem.

It is worth noticing that both algorithms achieve great results quite fastly. This can be **attributed** to the **deliberate design** of the **clicks** and **cumulative daily costs curves**, where a **significant portion** of the **bids** have **similar values** in terms of these quantities. As a result, with a **fixed** and **predetermined price**, both algorithms converge to **arms in the neighbourhood of the optimal one** after a relatively **small number of learning iterations**.



# Roadmap


Introduction
Advertising
Context generation
Frequent changes
0
1
2
3
4
5
6
Pricing
Joint Pricing and Advertising
Non-stationary environment

# Step 3: Learning for joint pricing and advertising

## Setting

The **pricing** and **advertising** settings are both **unknown**. The objective is to **maximize** the **reward** while estimating the purchase conversion rates, the number of clicks and cost. It is possible to obtain an optimal solution of the problem since we solve it in joint way.

**Scenario:**

- Users belong to **a single class** C1
- Curves related to the **advertising** problem are **unknown**
- Curve related to the **pricing** problem is **unknown**


**Task:**

- Apply **UCB1** and **TS** to maximize the reward while estimating the conversion probabilities
- Apply **GP-UCB1** and **GP-TS** to maximize the reward while estimating the advertising curves
- Plot the average value and standard deviation of the **cumulative regret**, **cumulative reward**, **instantaneous regret** and **instantaneous reward**

# Step 3: Learning for joint pricing and advertising

## Approach

The approach adopted in this point exploits the methods used in the previous two steps.
The **learners**, in this setting, are **composed by two parts**: **one for** solving the **pricing problem** and **one for** solving the **advertising one**. Both learners are used to interact with the environment trying to maximize the collected reward.
The **TS learner** is **composed by** a **TS learner** for the **pricing** problem and a **GP-TS learner** for the **advertising** problem.
The **UCB1 learner** is composed by a **UCB1 learner** for the **pricing** problem and a **GP-UCB1 learner** for the **advertising** problem.

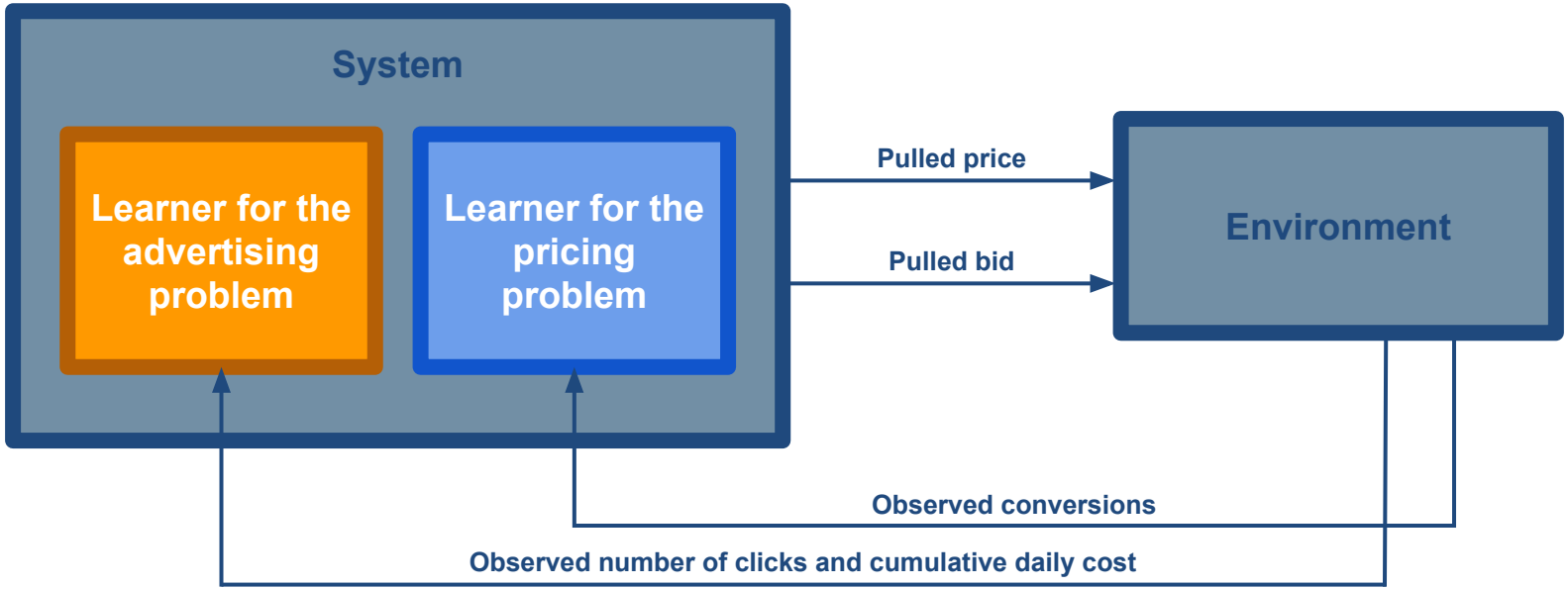
**Approach:**

- the **learner chooses the best price and the best bid** that maximize the reward;
- the l**earner plays the chosen price and bid** in the environment;
- the l**earner updates** its **models** associated to the pricing and to the advertising setting using respectively the obtained conversions and number of clicks and cumulative daily cost.

# Step 3: Learning for joint pricing and advertising

## Approach

**Schema of the system**



# Step 3: Learning for joint pricing and advertising

## Approach

The learner performs the **maximization of the reward by jointly optimizing prices and bids**.

**Steps of the optimization:**

1. the system uses the **learner** of the **pricing** problem **to compute** an estimate of the **conversion probabilities for all the prices**. The way the estimate is done depends on the algorithm used. In the case of **UCB1**, the value is the **upper confidence bound**; in the case of **TS**, the value is given by **sampling the Beta distributions**;
2. the system then uses the **learner** of the **advertising** problem **to compute** an estimate of the **number of clicks** and of the **cumulative daily costs for all the bids**. Also in this case the estimate depends on the algorithm used. In the case of **UCB1**, the value is the **upper confidence bound of the number of clicks** and **lower confidence bound of the costs** and, in the case of **TS**, the value is given by **sampling the distributions of the clicks and of the costs**;
3. using the obtained values, the learner computes the **estimates of the reward for all the couples price-bid** using the formula explained in the step 0;
4. the **pulled price** and **bid are the ones that maximize the reward**. Then they are played in the environment.

$$(price_{pulled}, bid_{pulled}) \leftarrow \arg\max_{(price,\, bid)} \{\#clicks_{est} \cdot conversion\ probability_{est} \cdot (price - other\ costs) - daily\ costs_{est}\}$$

# Step 3: Learning for joint pricing and advertising

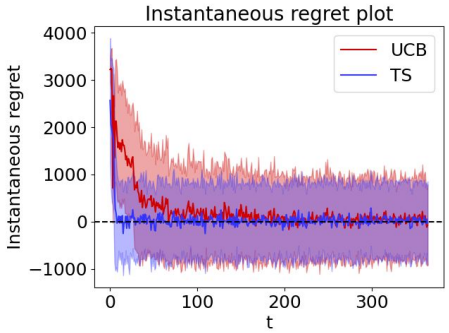
## Results

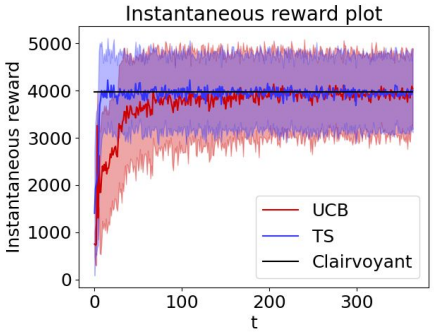
The following plots are obtained by averaging the results of **100 experiments**.

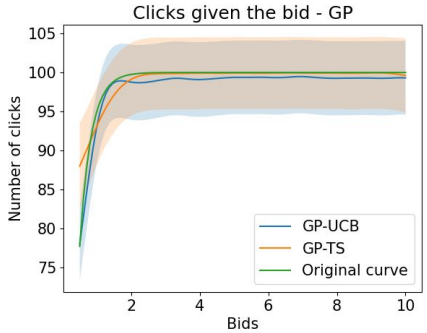
The **instantaneous regret tends to zero** for both the algorithm as the time grows, **they are learning** the best prices and bids to use in the environment. The **instantaneous regret** is a little **higher than the ones of the first two steps** of the project since the learner has to deal with more uncertainty, it doesn't know both the optimal price and the optimal bid.

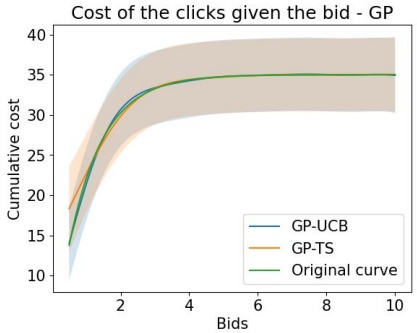
The **TS** algorithm is **faster in learning** the best arms with respect to UCB1, in a very small amount of days it achieves almost zero instantaneous regret and so the optimal instantaneous reward.

Also in this case the **clicks** and the **cumulative daily costs curves** have been **approximated well** by the Gaussian Processes of the learners.



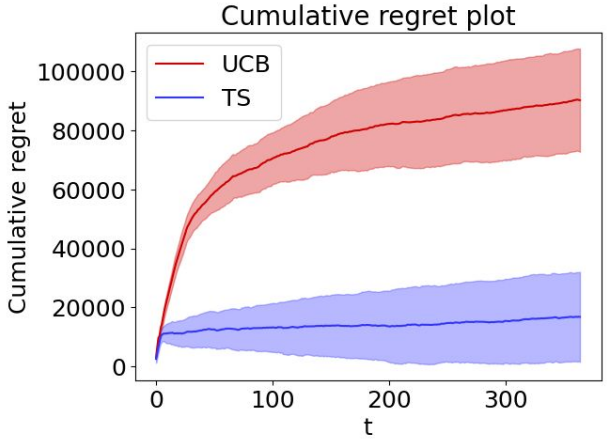



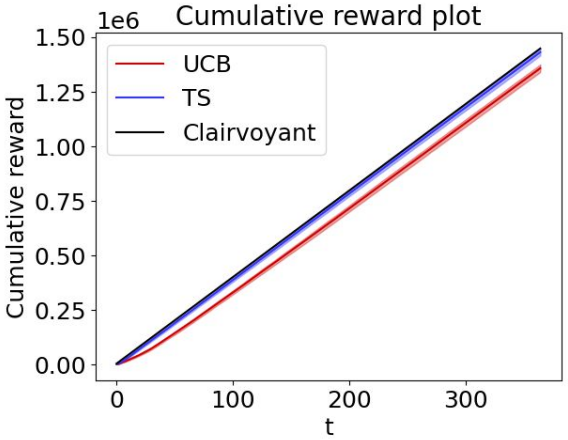
# Step 3: Learning for joint pricing and advertising

## Results

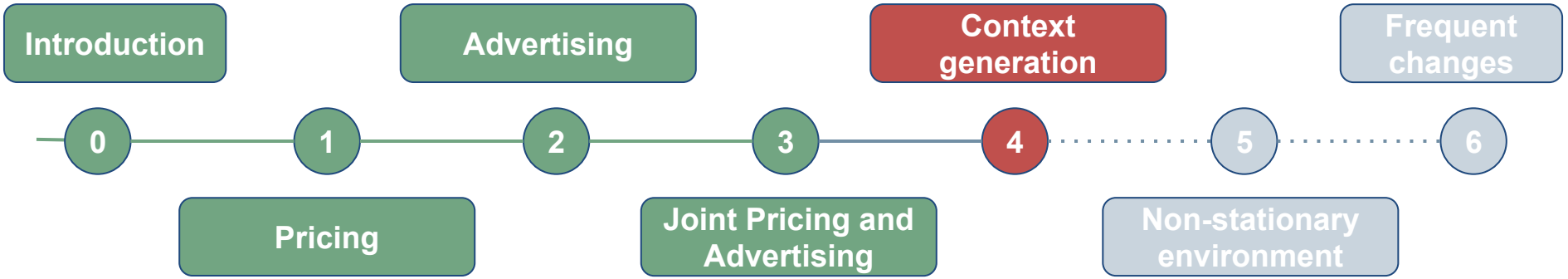
The cumulative regret plot shows that the algorithms meet their theoretical bounds since they have a **sub-linear cumulative regret**. It is possible to see that the cumulative regret of the algorithm is higher than the ones of the first two steps because the problem is more complex that the ones addressed by them. Still the plots are similar to the ones of the step 1 since the variable that makes the most of the difference in determining the value of the reward is the price.

The **quicker learning of TS** is even more visible in the cumulative regret plot. The performance of **TS** are higher, it **achieves a regret that is more than 4 times lower than** the one of the **UCB1** algorithm.



# Roadmap


Introduction
Advertising
Context generation
Frequent changes
0
1
2
3
4
5
6
Pricing
Joint Pricing and Advertising
Non-stationary environment

# Step 4: Context generation

## Setting

In the setting of this step, and very often in real scenarios, users have different characteristics.

Users interact with the e-commerce website in a variety of ways, which includes being more inclined to purchase the product at a different price and being more or less influenced by advertising campaigns.

Considering this scenario and based on the characteristic of the users it is usually possible to identify a set of classes within which users have the same features.

From a mathematical point of view each class can have a different demand curve and different functions of the clicks and the costs.

There are two way to tackle the problem:

1. considering an **aggregated model**: the e-commerce optimizes prices and bids considering the data as generated by a single class reaching sub-optimal results, in general;
2. considering a **disaggregated model**: the e-commerce optimizes prices and bids considering the data as generated by a many classes reaching better results with respect to the previous case.

In this step the performance of different settings and approaches will be evaluated and compared.

# Step 4: Context generation
# Setting - Case 1

In this part of the step the **context structure** is **known**. The **pricing** and **advertising** settings are **unknown**. The objective is to **maximize** the **reward** while estimating the purchase conversation rates, the number of clicks and cost. It is possible to obtain an optimal solution of the problem since we solve it in a disaggregate way.

**Scenario:**

- Users belong to **three classes**: C1, C2, C3
- Users are characterized in terms of **2 binary features**
- The **context structure** is **known** beforehand
- Curves related to the **advertising** problem are **unknown**
- Curve related to the **pricing** problem is **unknown**


**Task:**

- Apply **UCB1** and **TS** to maximize the reward while estimating the conversion probabilities for each class separately
- Apply **GP-UCB1** and **GP-TS** to maximize the reward while estimating the conversion probabilities for each class separately
- Plot the average value and standard deviation of the **cumulative regret, cumulative reward**, **instantaneous regret** and **instantaneous reward**

# Step 4: Context generation

## Approach - Case 1

In the case the actual context structure is known, it is possible to create a learner for each context in order to separately learn the curves of the different classes present in the environment.

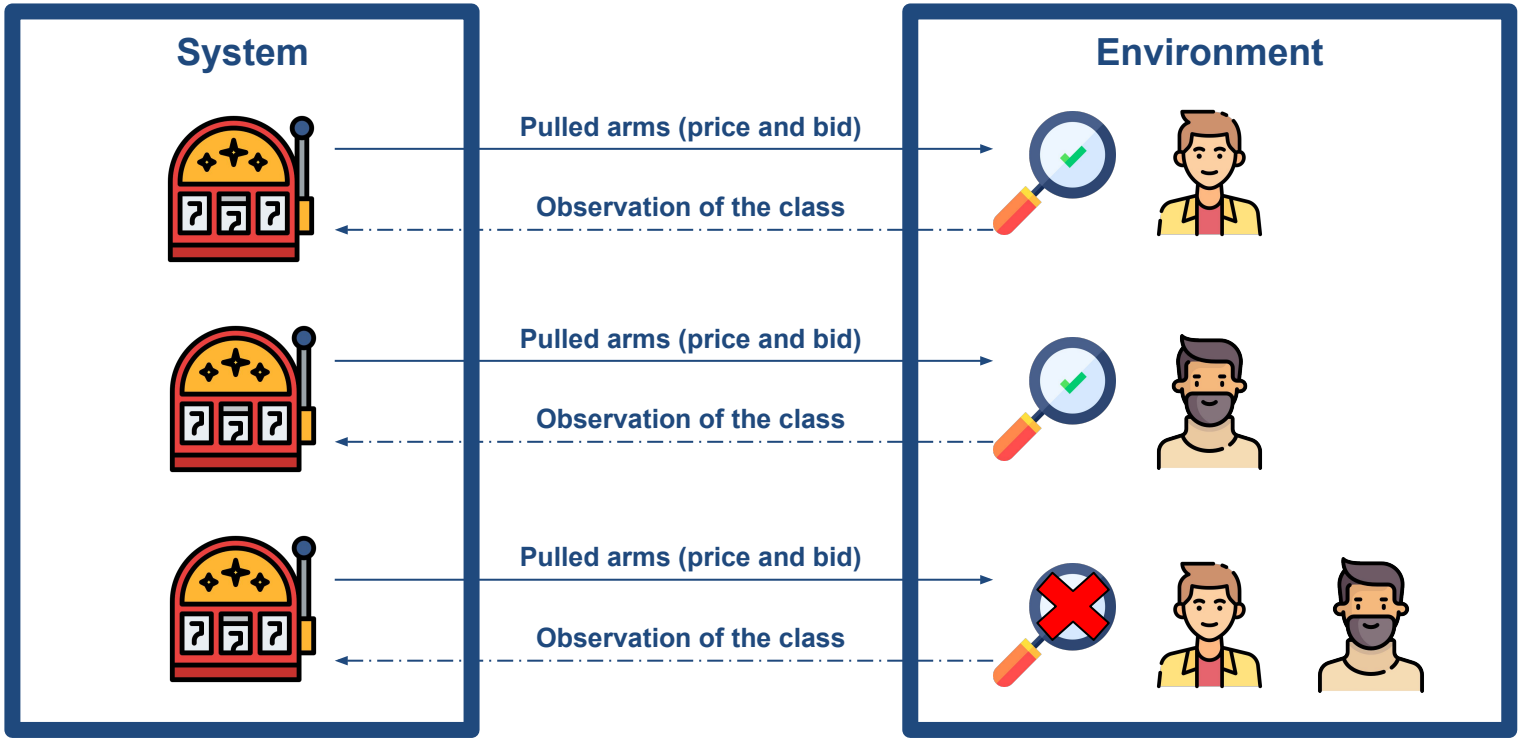
Each learner interacts with the environment trying to maximize its reward. The approach adopted by each regret minimizer is the same explained in the step 3.

**Approach:**

- the **learner chooses the best price and the best bid** that maximizes the reward;
- the l**earner plays the chosen price and bid** in the environment;
- the l**earner updates** its **models** associated to the pricing and to the advertising setting using respectively the obtained conversions and number of clicks and cumulative daily cost.

# Step 4: Context generation

## Approach - Case 1



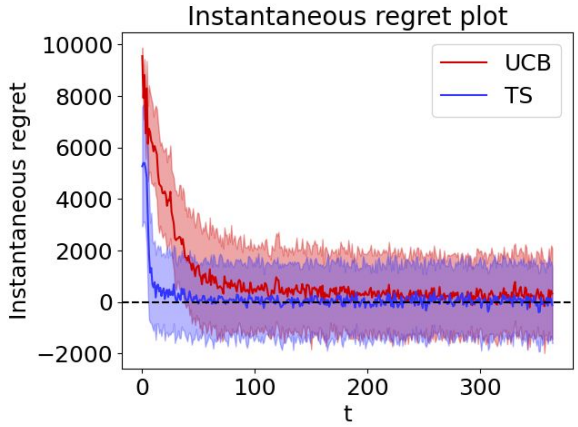
# Step 4: Context generation

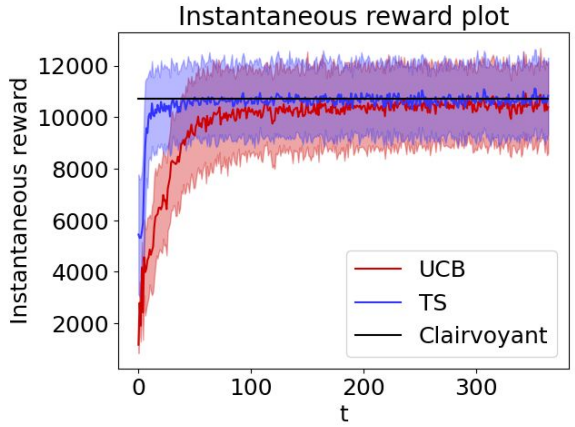
## Results - Case 1

The following plots are obtained by averaging the results of **100 experiments**.

The **instantaneous regret tends to zero** for both the algorithm as the time grows, they are learning the best arms. The initial instantaneous regret is higher than the ones of the previous steps since it is the sum of the instantaneous regret of three learners, one for each class of users. It is possible to see that also the instantaneous reward is higher than the ones of the previous cases because it is the sum of the rewards given by the three classes.

The **TS** algorithm is **faster in learning** the best arms with respect to UCB1, in a very small amount of days it achieves almost zero regret and so the optimal instantaneous reward.



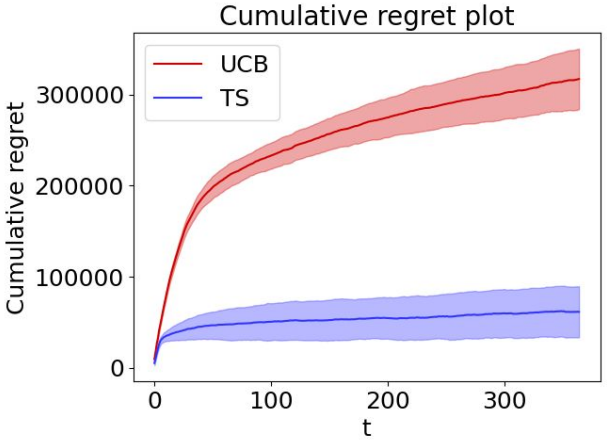
# Step 4: Context generation

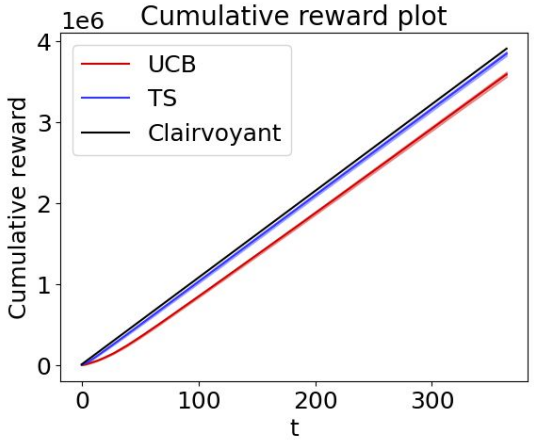
## Results - Case 1

The cumulative regret plot shows that the algorithms meet their theoretical bounds since they have a **sub-linear cumulative regret**.

The **TS** algorithm achieves a cumulative regret that is much lower than the one of UCB1 thanks to its **quicker learning**. TS has very good performance, after a year, it allows to have a reward of about 4 millions € and a regret of about 50 thousand €, the reward is much lower than the regret.

The **cumulative reward** is **higher** than the one of the previous cases since it comes from all the classes of users.



# Step 4: Context generation

## Setting - Case 2

In this part of the step the **context structure** is **unknown**. The **pricing** and **advertising** settings are **unknown**. The objective is to **maximize** the **reward** while estimating the purchase conversation rates, the number of clicks and cost. It is required to solve the problem in a disaggregate way, estimating the context structure of the environment from data.

**Scenario:**

- Users belong to **three classes**: C1, C2, C3
- Users are characterized in terms of **2 binary features**
- The **context structure** is **unknown**
- Curves related to the **advertising** problem are **unknown**
- Curve related to the **pricing** problem is **unknown**


**Task:**

- Apply **UCB1** and **TS** to maximize the reward while estimating the conversion probabilities for each class separately
- Apply **GP-UCB1** and **GP-TS** to maximize the reward while estimating the advertising curves for each class separately
- Apply a **context generation algorithm** in order to estimate the context (every two weeks)
- Plot the average value and standard deviation of the **cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward**

# Step 4: Context generation

## Approach - Case 2

In the case the actual context structure is not known, the number of learners to instantiate and the type of users with which the learners have to interact are not known.

**The context structure has to be estimated from the observations** using a context generation algorithm while the learning algorithms are executing.

The underlying idea is to find groups of people, identified by specific values of the two binary features considered by the problem, that have similar pricing and advertising models and then split the groups and create a learner for each one of them.

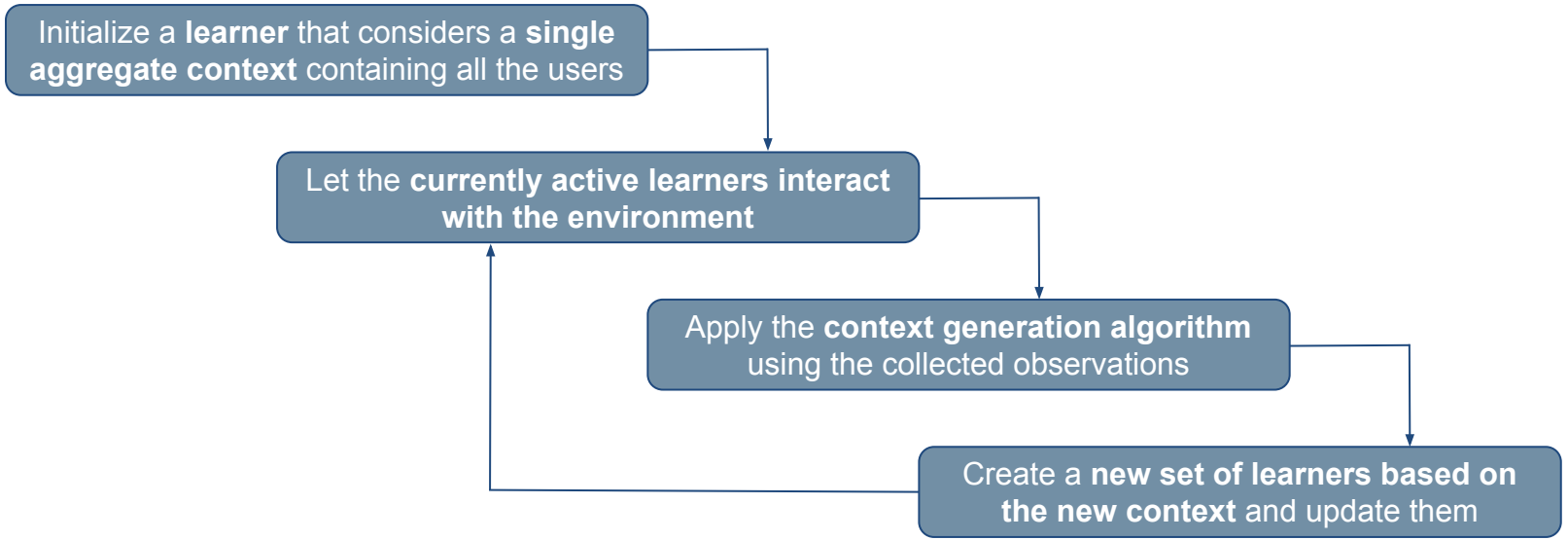
**At the beginning**, there is no information about the actual context structure so the only way to go is to use a **single learner** that interacts with the **aggregate model** of the environment where users of all the types are contained.

In the setting under consideration, there are **4 possible user types** based on the binary features **“age”** and **“has searched technology terms”.** The systems interact with the environment obtaining observations for each type of user and store them.

# Step 4: Context generation

## Approach - Case 2

**Steps of the approach from an high level point of view**



# Step 4: Context generation

## Learners and observations - Case 2

Each learner interacts with the environment trying to maximize its reward. The approach adopted by each regret minimizer is the same explained in the step 3.

**Approach:**

- the **learner chooses the best price and the best bid** that maximizes the reward;
- the **learner plays the chosen price and bid** in the environment;
- the **learner updates** its **models** associated to the pricing and to the advertising setting using respectively the obtained conversions and number of clicks and cumulative daily cost;
- the **observations** are stored **grouped by feature tuple**.

**Structure of an observation:**

1. **price pulled** by the algorithm;
2. observations of the **conversions** obtained from the environment;
3. **bid pulled** by the algorithm;
4. **number of clicks** obtained from the environment;
5. **cumulative daily cost** obtained from the environment;
6. **collected reward**.

# Step 4: Context generation

## Greedy algorithm - Case 2

The **context generation algorithm** applied is a **greedy algorithm** that estimates the reward of a context and the sum of the rewards coming from the possible sub-context and chooses whether to split or not based on them.

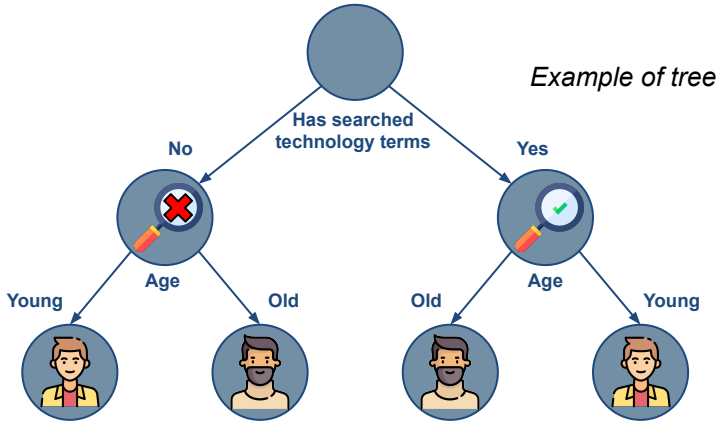
The **greedy algorithm builds a tree** where **each node represent a context**. The algorithm starts from a single node, namely the **root**, that **consists in the completely aggregate context**.

The split of a node is done choosing a feature of the users and creating a node for each value that the feature can assume.

The arcs of the tree constrain the value assumed by the split feature inside the child node.

The path that brings from the root node to another node defines the values of the features that a user must have to belong to the considered node (context).

Based on the data the algorithm can choose to expand some contexts and to not expand other contexts.



# Step 4: Context generation

## Implementation of the greedy algorithm - Case 2

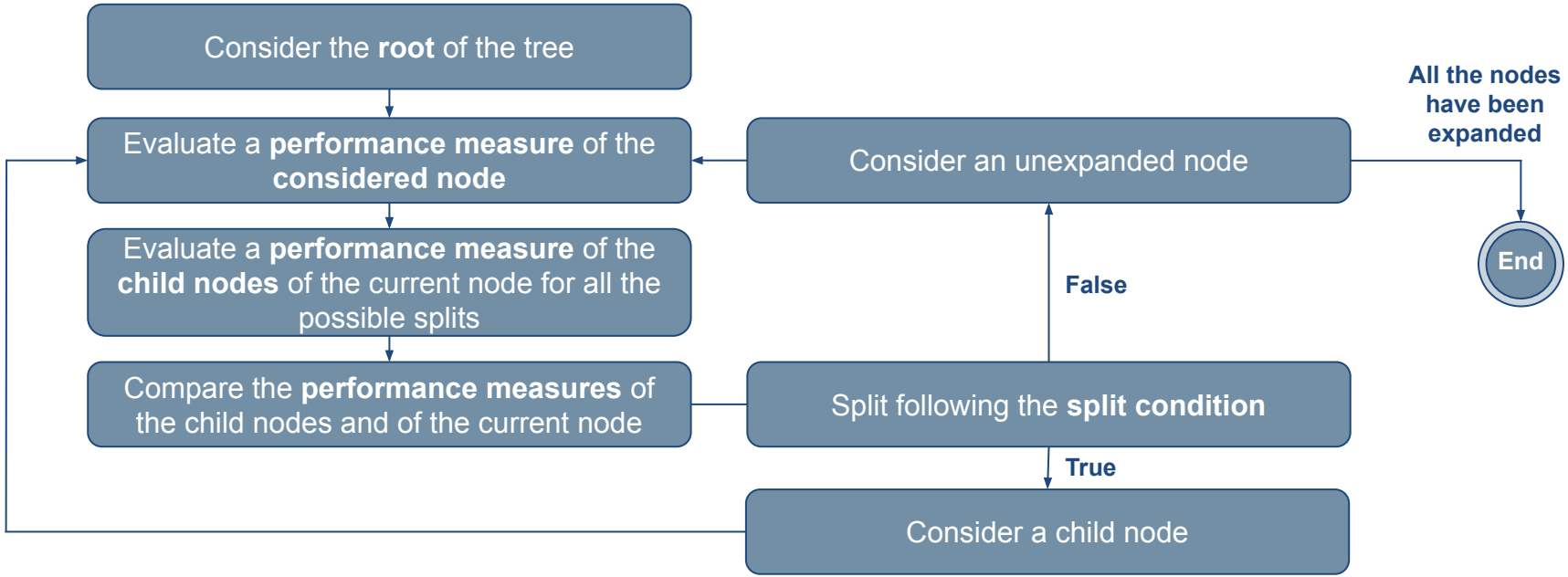
**Implementation choices:**

- context generation algorithm runs **every 14 days** (from specification);
- at each run of the **context generation** algorithm, it **starts considering a single aggregated context** and applies to it the procedure just described. In this way even if the algorithm chose in the previous 14 days a wrong split, it can correct the context structure, on the basis of the observations, starting again from the beginning;
- all the **observations from the beginning to the current step are used** by the context generation algorithm to evaluate the new context structure.

# Step 4: Context generation

## Greedy algorithm - Case 2

**Steps of the greedy algorithm for context generation from a high level point of view**



# Step 4: Context generation

## Split condition - Case 2

The **split of a context** in two sub contexts has to be done **only if it is valuable in terms of reward**, namely it allows to gain a bigger reward with a certain confidence.

In the case of **binary features** the **split condition** that is evaluated when splitting is:

$$\underline{p}_{C_1}\underline{\mu}_{a^*_{C_1},C_1} + \underline{p}_{C_2}\underline{\mu}_{a^*_{C_2},C_2} \geq \underline{\mu}_{a^*_{C_0},C_0}$$

where the quantities are:

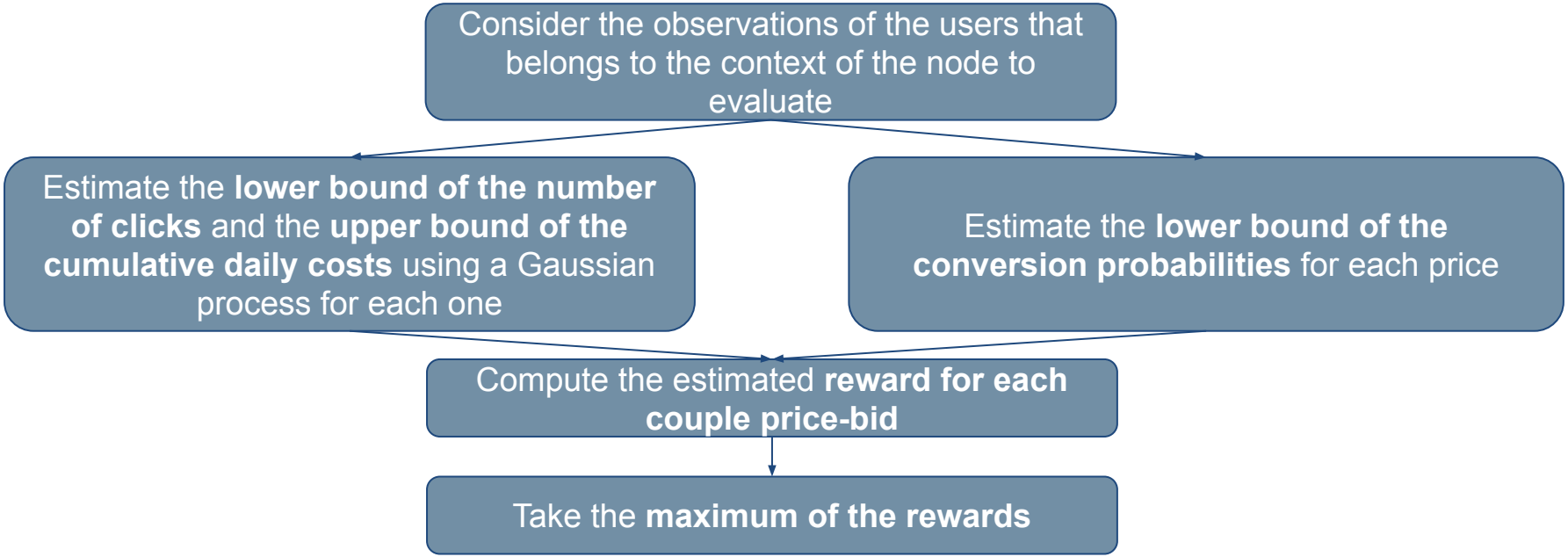
- $\underline{p}_{C_x}$ is the **lower bound of the probability of a user to belong to the context $C_x$**;
- $\underline{\mu}_{a^*_{C_x},C_x}$ is the **lower bound of the reward of the best arm in the context $C_x$**;
- $C_1$ and $C_2$ are two sub-contexts generated by $C_0$, in other words they are the child nodes of $C_0$ splitting on a feature.

The algorithm is greedy, so **when many features can be used to split** a context, the **one used provides the maximum estimated reward**, namely the one for which the **left-hand side of the previous inequality is bigger**.

# Step 4: Context generation

## Implementation of the context generation algorithm - Case 2

**Computation of the lower bound of the reward in each node**



# Step 4: Context generation

## Implementation of the context generation algorithm - Case 2

The usage of the **Gaussian processes** to estimate the **number of clicks** and the **cumulative daily costs** is driven by the fact that in general we could have many values of the bids for which the algorithm do not have measurements.

The Gaussian processes allow to exploit the shape of the advertising curves in order to have a good approximation of the number of clicks and costs for all the bids that can be used in the context generation algorithm. The reasoning is similar to the one adopted by the GP-TS and GP-UCB learners.

The **lower bound** of the **number of clicks** is computed as the **mean** of the Gaussian process of the number of clicks **minus the standard deviation** of the Gaussian process.

The **upper bound** of the **cumulative daily cost** is computed as the **mean** of the Gaussian process of the cumulative daily cost **plus the standard deviation** of the Gaussian process.

# Step 4: Context generation

## Lower bound - Case 2

The **lower bound** used for the **conversion probability** and for the **probability of a context** is the **Hoeffding bound**, the general formula is:

$$\bar{x} - \sqrt{-\frac{\log \delta}{2|Z|}}$$

where the quantities are:

- $\bar{x}$ is the **mean** of the considered **random variable**;
- $\delta$ is 1 minus the **confidence**, it can be considered as a metric used to control the level of certainty required for an algorithm to decide whether to split the context or not;
- $|Z|$ is the **size** of the considered **set of data**.

# Step 4: Context generation

## Lower bound - Case 2

The **lower bound** of the **conversion probability** is computed for **each** one **of the prices *p*** and it is:

$$\underline{conversion\ probability_p} = \frac{\#convesions_{p,t}}{\#observations_{p,t}} - \sqrt{-\frac{\log \delta}{2\,\#observations_{p,t}}}$$

- $\#convesions_{p,t}$ is the number of conversions until time *t* of the considered price *p*;
- $\#observations_{p,t}$ is the total number of observations until time *t* of the considered price *p*.

Using the **lower bound** for the **probability of a context**, it becomes:

$$\underline{p}_{C_1} = \frac{\#clicks_{C_1,t}}{\#clicks_{C_0,t}} - \sqrt{-\frac{\log \delta}{2\,\#clicks_{C_1,t}}}$$

- $\#clicks_{C_0,t}$ is the total number of clicks given by the users belonging to context $C_0$ until time *t*;
- $\#clicks_{C_1,t}$ is the total number of clicks given by the users belonging to context $C_1$ until time *t*;
- $C_1$ is a sub-context of the context $C_0$.

# Step 4: Context generation

## Training of the new learners - Case 2

After the new contexts have been produced by the greedy algorithm, the set of learners for the next 14 days has to be created:

- if **a new context** is **created**, different from all the ones of the previous context structure, **a new learner is deployed** for interacting with the users of that context;
- if **a context remains invariant** after the execution of the context generation algorithm, **its learner remains the one used in the previous days**. This is done for performance reasons.

**New learners**

Before starting the interaction with the environment, the **new learners are trained using all the past observations** of the users belonging to the related context.

Using all the observations allows to **exploit past data** and to have **better performances**.

In order to be efficient, the **Gaussian processes** of each new learner are **fit only once all the past observations have been assigned to them**, and **not** in an **online fashion.**

# Step 4: Context generation

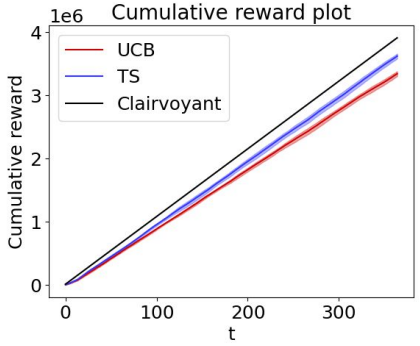
## Results - Case 2

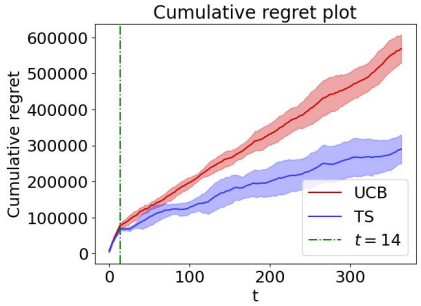
The following plots are obtained by averaging the results of **10 experiments**.

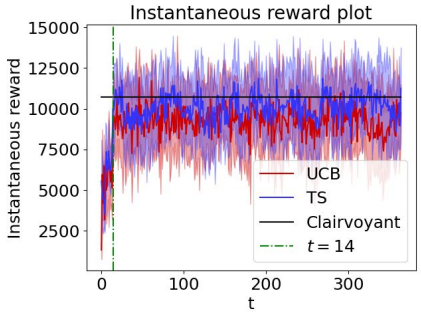
By looking at the instantaneous reward (regret) plot we can see, during the first days of execution, the actual effect of the **context generation algorithm**.

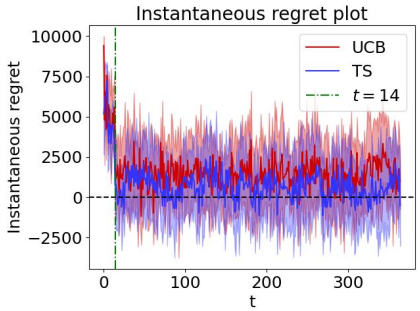
At ***t = 14** days* it starts the **first execution** of the **aforementioned algorithm**, thus leading to the **separation** of the **aggregated context** into 2 or more sub-contexts. This fact leads to a **rapid growth** (decrease) of the **instantaneous reward** (regret) which is then, more or less maintained for the remaining days of the year. However, the **behaviour is not stable**, but it presents a **fluctuating trend in some parts of the year**.

This is due to the fact that the **context generation algorithm may switch**, for some reasons that are due to the stochasticity of the observations and to our design of the environment, **from the optimal split** (3 contexts) **to sub-optimal ones** (e.g. 2 contexts).



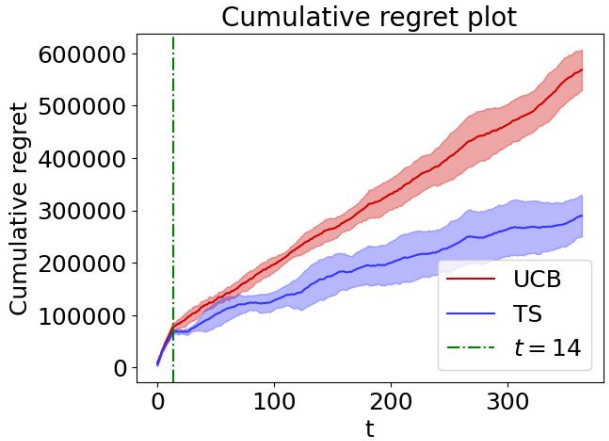
# Step 4: Context generation

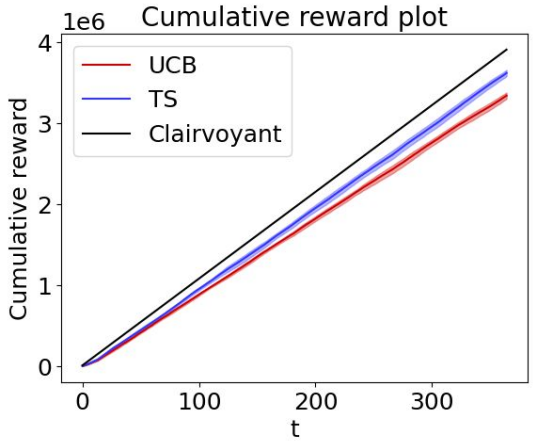
## Results - Case 2

It is possible to notice that **Thompson Sampling seems** to achieve **sub-linear** regret, while **UCB1** shows a **linear behaviour**, thus leading to sub-optimality. The curves are a little bit noisy due to the low number of experiments.

Even if the regret of the UCB1 algorithm looks linear, during the execution of the context generation algorithm, it was observed that **UCB1 was able to understand quite well the correct context structure**.

The behaviour of **UCB1** is **still** able to achieve a **not too high regret**.



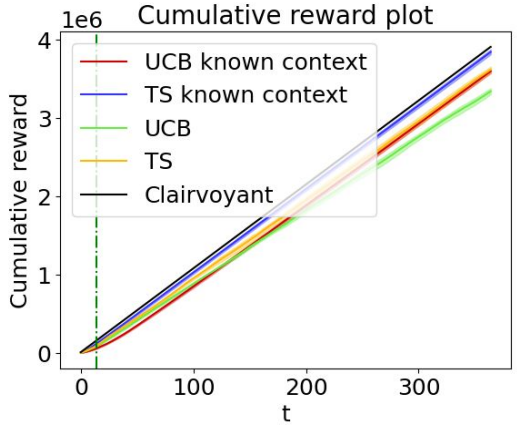
# Step 4: Context generation

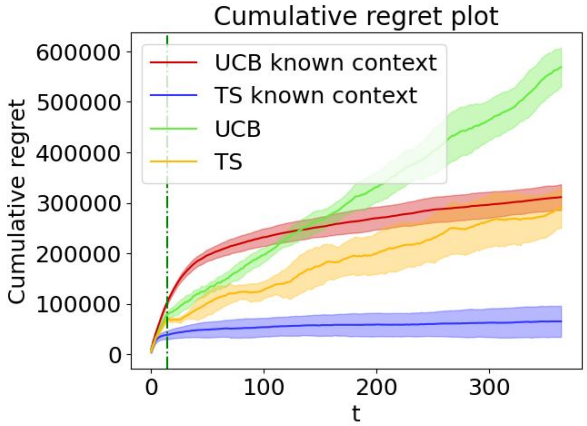
## Comparison cases 1 and 2

Here the **comparison** between the learners of the **scenario 1** and **2**.

The **learner** that implements the **context generation** algorithm achieves **good results,** but obviously the **learner that knows the context** in advance **outperforms it**. This is **valid for both TS and UCB**. Due to the big uncertainty related to the fact that the learner in the scenario 2 does not know the context, it cannot achieve the best performance.

TS of the scenario 2 is comparable to the curve of the TS algorithm in the case of known context since it is quite sub-linear, whereas the UCB1 algorithm in the unknown context setting obtains a much higher regret in comparison to UCB1 applied knowing the context (regret is double and it will increase even more).



# Step 4: Context generation

## Setting - Case 3

In this part of the step the **context structure** is **unknown**. The **pricing** and **advertising** settings are **unknown**. The objective is to **maximize** the **reward** while estimating the purchase conversation rates, the number of clicks and cost. It is required to solve the problem in a aggregate way, so without estimating the context structure of the environment.

**Scenario:**

- Users belong to **three classes**: C1, C2, C3
- Users are characterized in terms of **2 binary features**
- The **context structure** is **unknown**
- Curves related to the **advertising** problem are **unknown**
- Curve related to the **pricing** problem is **unknown**

**Task:**

- Apply **UCB1** and **TS** to maximize the reward while estimating the conversion probabilities
- Apply **GP-UCB1** and **GP-TS** to maximize the reward while estimating the advertising curves
- Plot the average value and standard deviation of the **cumulative regret**, **cumulative reward, instantaneous regre**t and **instantaneous reward**

# Step 4: Context generation

## Approach - Case 3

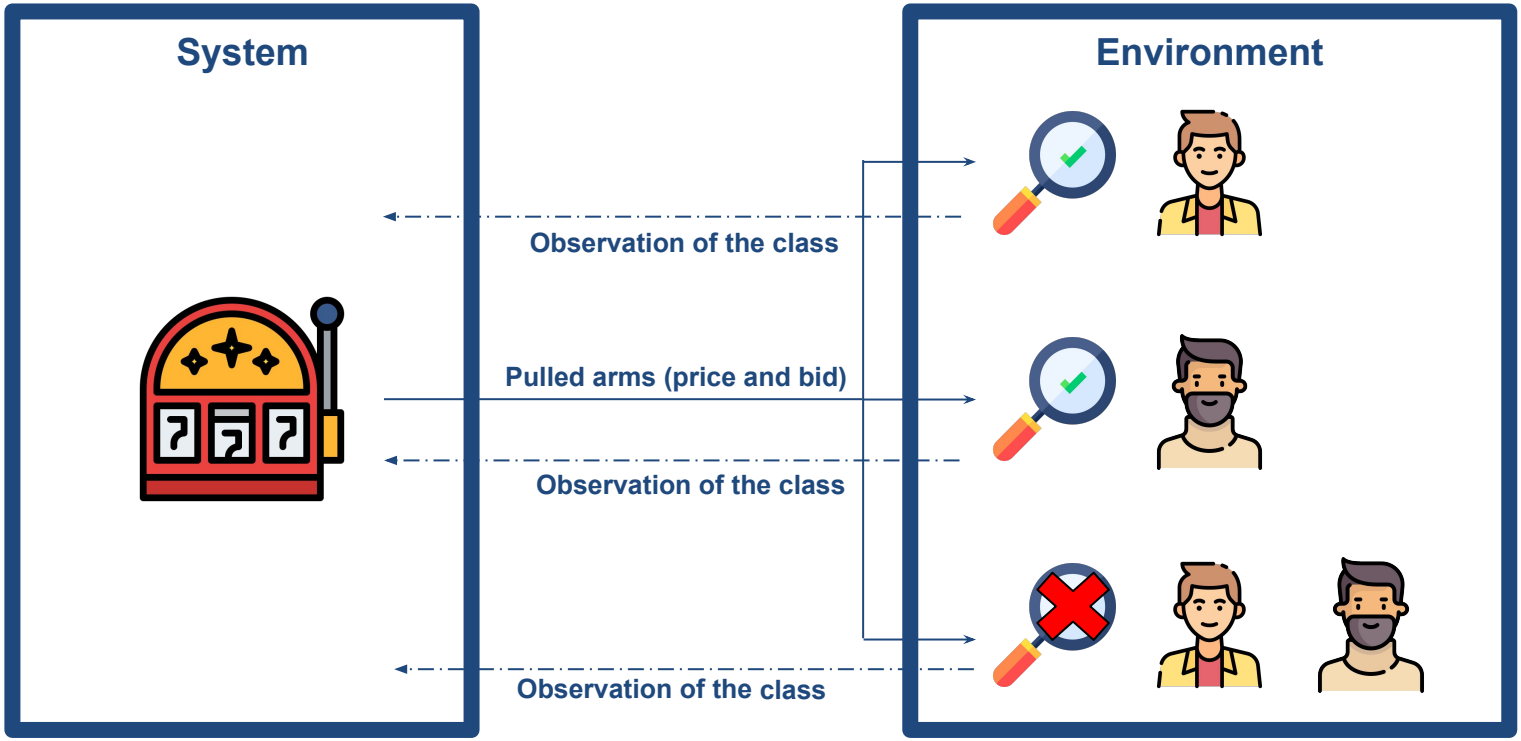
In this case the actual context structure is not known and the choice is to not use a context generation algorithm to learn the context structure of the environment. Instead a single learner is used to tackle the problem. The learner will choose a single price and a single bid per day and will use it for all the possible types of users.

The learner interacts with the environment trying to maximize its reward. The approach adopted by the regret minimizer is the same explained in the step 3:

- the **learner chooses the best price and the best bid** that maximizes the reward;
- the l**earner plays the chosen price and bid** in the environment;
- the l**earner updates** its **models** associated to the pricing and to the advertising setting using respectively the obtained conversions and number of clicks and cumulative daily cost.

# Step 4: Context generation

## Approach - Case 3



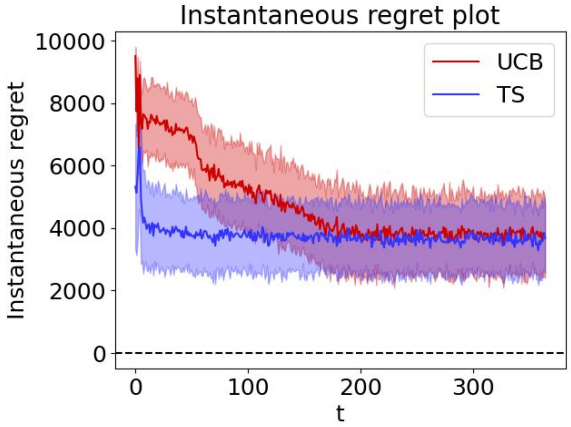
# Step 4: Context generation

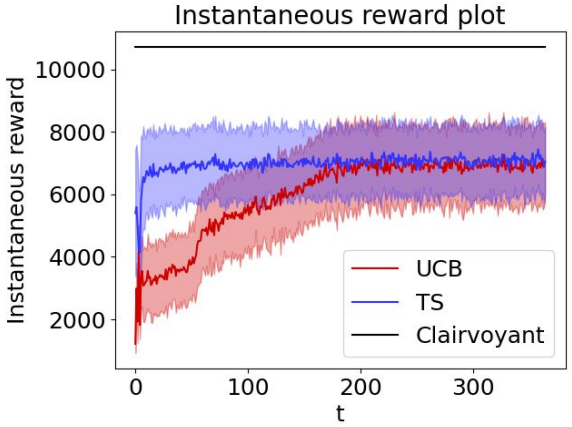
## Results - Case 3

The following plots are obtained by averaging the results of **100 experiments**.

The **instantaneous regret does not tend to zero** as the time grows, **neither for TS nor for UCB1**. The instantaneous reward and the instantaneous regret reaches a point where they are almost constant, this it the point in which the learner found the best price and bid for the setting. The single learner we are using is learning the best arms for the aggregate context that in general are sub-optimal. The reward cannot reach the instantaneous reward given by the clairvoyant algorithm in the case of disaggregate context since it doesn't distinguish between users belonging to different classes.

Also in this case **TS** algorithm is **faster in learning** with respect to UCB1.



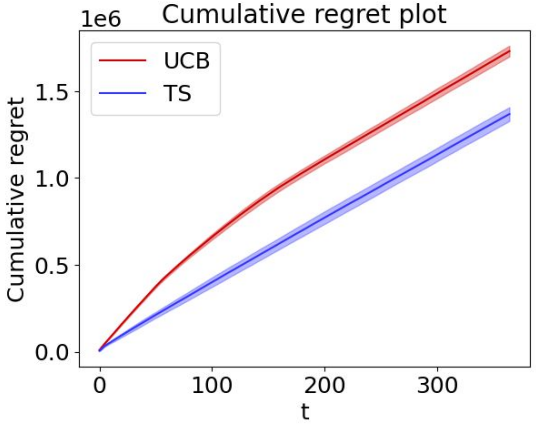



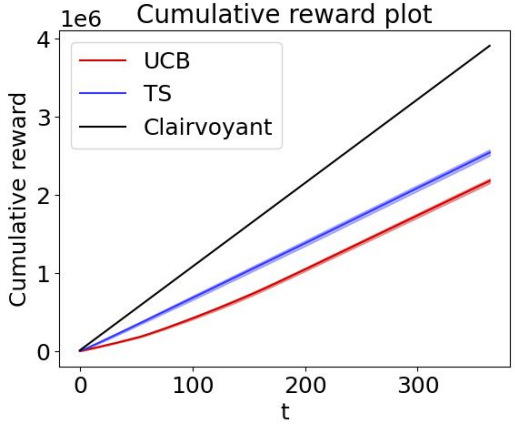
# Step 4: Context generation

## Results - Case 3

As expected the **cumulative regret is linear** for both the algorithms. This is due to the fact that the learners are pulling a single price and bid at each time and are using them for all the users of the environment. The learner, in this way, is estimating the best arms for the aggregate model that in general are not equal to all the ones of the user classes. The clairvoyant algorithm gives the optimal cumulative reward of the disaggregate model, that in general is higher than the one from the aggregate setting. It is possible to see indeed that there is a lot of difference in the two rewards. The fact that in the instantaneous regret does not tend to zero is translated in a linear shape of the cumulative regret.

Also in this case it is possible to see that **TS** algorithm **outperforms UCB1**.



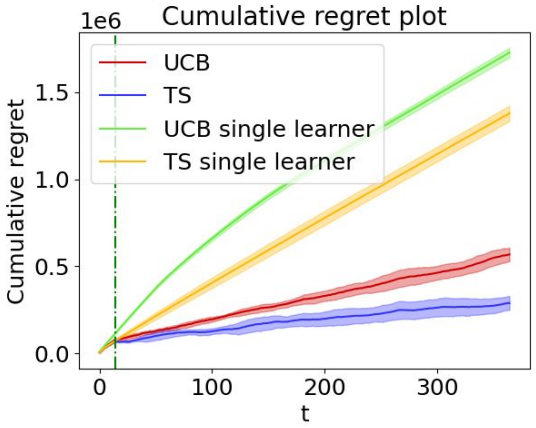
# Step 4: Context generation

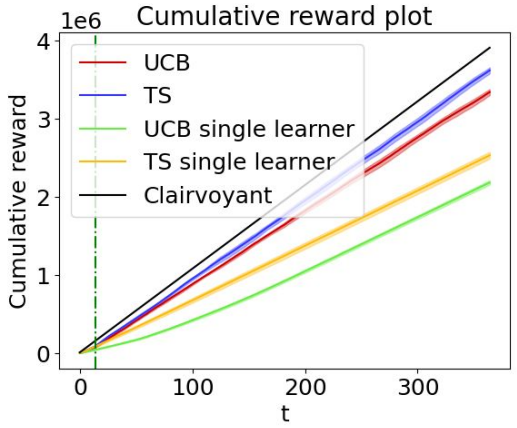
## Comparison cases 2 and 3

Here the **comparison** between the learners of the **scenario 2** and **3**.

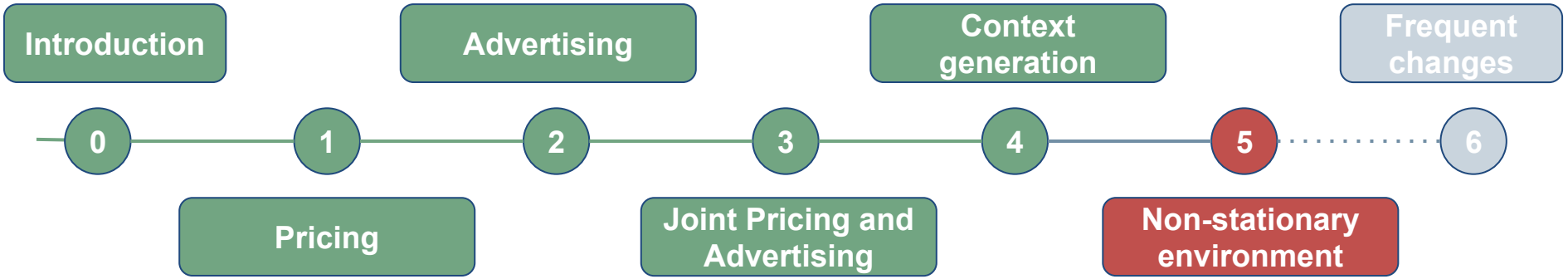
Thanks to price discrimination and bid discrimination strategies the **reward** collected by the **system that estimates the context structure** of the problem is **greater than** the one of the **system** that uses a **single learner**, which is using the fully aggregated context.

Even if the **context generation algorithm** does not always determine the actual context structure, it **estimates a better context structure** with respect to the one that considers **all the users in the same context**.



# Roadmap


Introduction
Advertising
Context generation
Frequent changes
0
1
2
3
4
5
6
Pricing
Joint Pricing and Advertising
Non-stationary environment

# Step 5: Non-stationary environments with two abrupt changes

## Setting

In this step the **pricing** part is **unknown**, instead the **advertising** one is **known**. In this case, the pricing curves are subject to **two abrupt changes**. The objective is to **maximize** the **reward** across all the phases while estimating the purchase conversation rates, using variations of UCB1 to deal with non-stationary settings.

**Scenario:**

- All the users belong to class **C1**
- Curves related to the advertising part are **known**
- Curves related to the pricing part are **unknown** and **non-stationary**, being subject to **three different seasonal phases**

**Task:**

- Apply **UCB1**, **UCB1 with sliding window** and **UCB1 with change detection test** to maximize the reward while estimating the conversion probabilities
- Provide a **sensitivity analysis** of the parameters employed in the algorithms
- Plot the average value and standard deviation of the **cumulative regret**, **cumulative reward**, **instantaneous regret** and **instantaneous reward**
- Compare the results of the three algorithms

# Step 5: Non-stationary environments with two abrupt changes

## Approach

In this scenario the advertising curves are fully known, thus the reward is computed using the real values of the number of clicks and cumulative cost given the bid related to the chosen price. Three different versions of UCB1 are employed to minimize the regret while learning the conversion probabilities.

The phases are spread over the year as follows:

1. the **first phase**, which lasts from February to May, is characterized by the **launch of the mobile phone** when the preferred price is the fourth one (650€) because customers are more eager to spend to obtain the new phone in their hands;
2. the **second phase**, that goes from June to around September, is characterized by **regular purchasing period** when the preferred price is the intermediate one (600€), thus there is a slight decrease in price;
3. the **third phase** lasts from October to January and it is characterized by **discounts sales** since it is a festive period and the launch of the next generation phone is upcoming. During this period the second price (550€) has a peak in its conversion rate**.**

[A3] Conversion probabilities for the three phases

# Step 5: Non-stationary environments with two abrupt changes

## Approach

The interaction between the learners and the environment is the same as step 1, namely for each price the best bid is computed and then the associated number of clicks and daily cost. Successively, the arm is pulled maximizing the reward using the learners' estimate of the conversion probabilities. Finally, the learners are updated using the collected data.

In this step, together with standard UCB1, it is required to use two different versions of UCB1 tailored for non-stationary environments. In particular UCB1 with sliding windows (**SW-UCB**) and UCB1 with change detection test (**CUSUM-UCB**).

We followed theoretical recommendations to determine the parameters of the two variants:

- in the case of **sliding window**, the value used for the window size is $4\sqrt{T}$ = 76
- in the case of **change detection** the algorithm used is **CUSUM**. The chosen parameters are:
  - M = 50
  - $\varepsilon$ = 0.1
  - h = $\frac{1}{2}\log T$ = 2.95
  - $\alpha$ = $\sqrt{\frac{\log T}{T}}$ = 0.13

# Step 5: Non-stationary environments with two abrupt changes

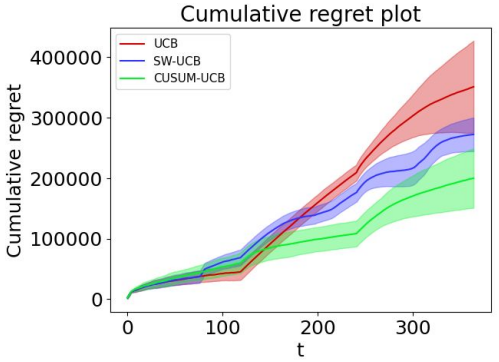
## Results

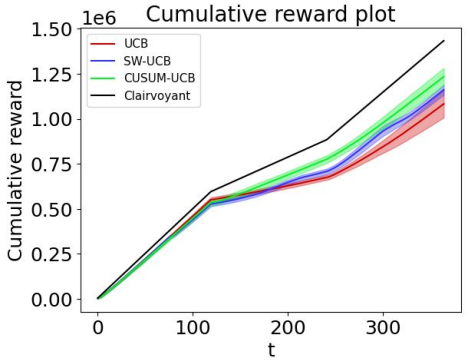
**100 experiments** were carried out to create the plots.

As expected the **two non-stationary versions** of UCB perform **better** than the vanilla UCB, with **CUSUM-UCB** being the **best one**.

In particular, we can observe that **SW-UCB** and **CUSUM-UCB**, in each phase, obtain a **sub-linear regret**, being able to handle with non-stationary settings. UCB instead cannot, especially during the second phase where it doesn't detect the change obtaining a linear regret. The **cumulative regret** of **UCB increases significantly** during the second phase, outpacing the regret of the other algorithms.

In the average case employing SW-UCB in comparison to UCB provides for a loss reduction from around 350k€ to 275k€ (-22%), instead using CUSUM-UCB to 200k€ (-43%).



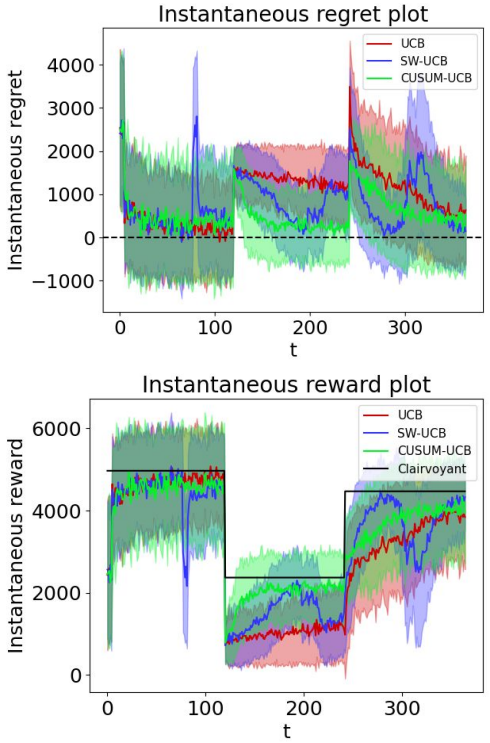



# Step 5: Non-stationary environments with two abrupt changes

## Results

These charts show that all three learners do well during the **first phase**, but when the **first abrupt change** occurs, the two non-stationary versions of UCB are able to handle it whereas UCB is unable to. In fact, after 120 days UCB has a good estimate of the conversion probabilities and the bounds are tight so it takes a lot of time to realise that the optimal price has changed. In the **third phase**, CUSUM-UCB and SW-UCB also quickly identify the best arm.

Periodically, with period of the window size, SW-UCB has negative **peaks** in the instantaneous reward (positive peaks in instantaneous regret). This is **due to the fact that suboptimal arms leave the window** since they are never played, causing the corresponding bounds to explode. Therefore the **algorithm is forced to play them at least once**, but by receiving a low reward, it is prone to play again the optimal price.



# Step 5: Non-stationary environments with two abrupt changes

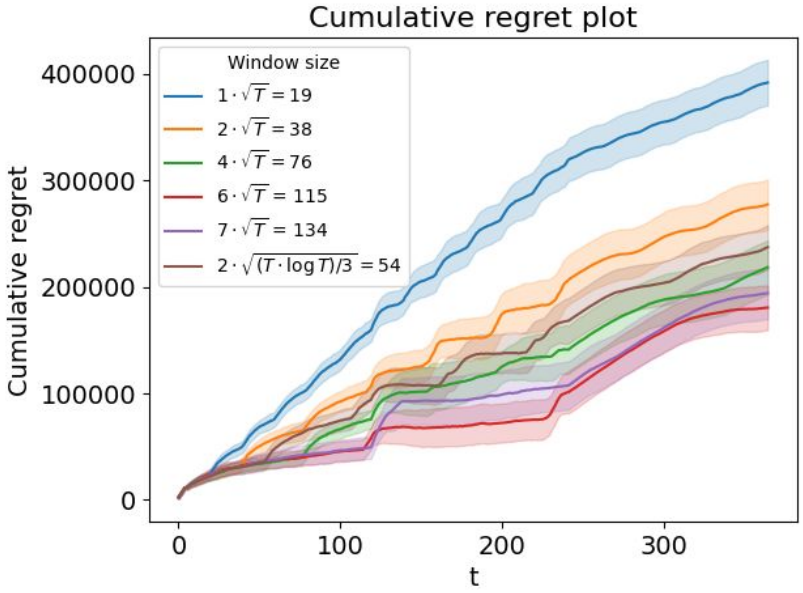
## Sensitivity Analysis - UCB-SW

We tried five different **window size** values for the UCB sliding window, all of them were proportional to $\sqrt{T}$.

Furthermore, we tested the following theoretical value (*Garivier et al.*[2]): if the horizon T and the growth rate of the number of breakpoints are known in advance we can set the window size to $2\sqrt{\frac{T \log T}{\#breakpoints}}$.

The lower is the window size, the higher is the number of times that the algorithm has to retry all the arms. In this case, by applying **small multiplicative factors**, like 1 or 2, the **performances decrease** because the window is too short with respect to the length of the phases and so the learner frequently needs to play suboptimal arms. While **the larger is the window size**, **the lower is the sensitivity of the algorithm**.

Values for the window in the range between 50 and 80, for instance using a factor of 4, are good. Using as multiplicative factor 6 or 7 would be **optimal** in this case because the window size would be similar to the length of the phases. Applying even **higher factors** would decrease the performance leading SW-UCB to don't explore enough.




[2] On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems.Garivier, A., & Moulines, E. (2008). https://arxiv.org/pdf/0805.3415.pdf

# Step 5: Non-stationary environments with two abrupt changes

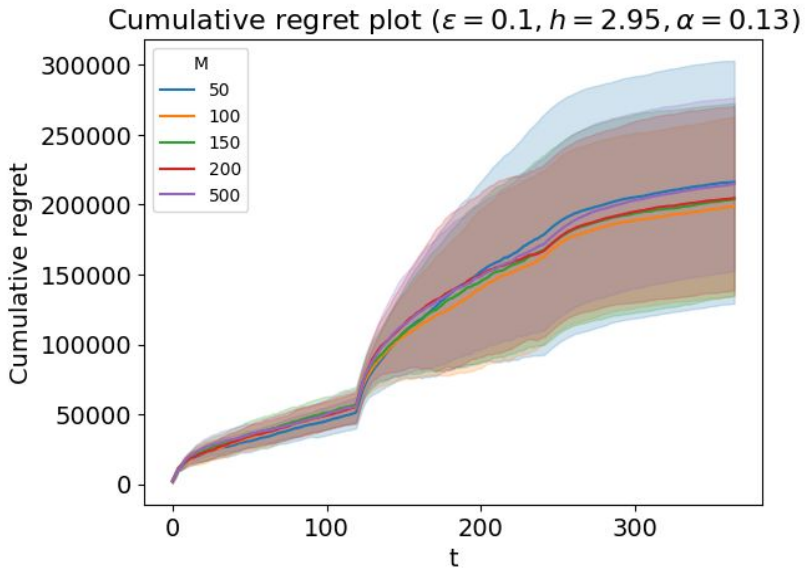
## Sensitivity Analysis - UCB-CUSUM

**CUSUM - M analysis**

We analyzed five different values for M, that is the number of samples used to compute the **reference point**, namely the **empirical mean of the conversions** for a given price, employed to detect changes.

Because the detection test is not run until M samples have been gathered, in general, if M is **too large**, the **majority of the observations** are **utilised** to **compute** the **reference point**, making the algorithm behave similarly to a regular UCB. Therefore, with big values of M, it's possible that the changes aren't detected or take a long time to be identified.

A **too small M** is still **suboptimal** since it leads to **inaccurate estimations of the reference point** and **more frequent change detections**, even when no actual change occured.
Each arm should be played at least M times between each **breakpoint**, and M should be large enough to allow for an accurate estimation of the reference point. In this instance, placing M between 50 and 100 is sufficient to get a decent estimate of the reference point because each time an arm is chosen, it receives approximately 100 clicks. All of the values taken into consideration perform similarly.



# Step 5: Non-stationary environments with two abrupt changes

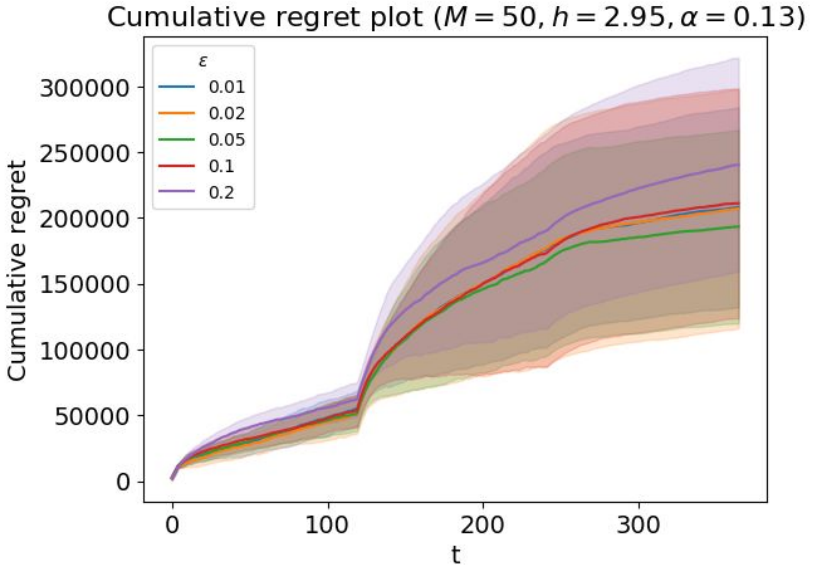
## Sensitivity Analysis - UCB-CUSUM

**CUSUM - $\varepsilon$ analysis**

We tested five different values for $\varepsilon$, which is used to have an **underestimation** of the **deviation** from the **reference point**.

In general, with **low values of $\varepsilon$**, **changes are detected even when there aren't**. Whereas as $\varepsilon$ **increases** the **sensitivity decreases**, so more samples of the new distribution are necessary to flag a change.

We note that values in the range between 0.01 and 0.1 produce similar performance, while increasing the value of $\varepsilon$ results in a rise of the cumulative regret, because $\varepsilon$ should be of the **order of the changes in the distributions**, which in our case are of the order of 0.1.



# Step 5: Non-stationary environments with two abrupt changes

## Sensitivity Analysis - UCB-CUSUM

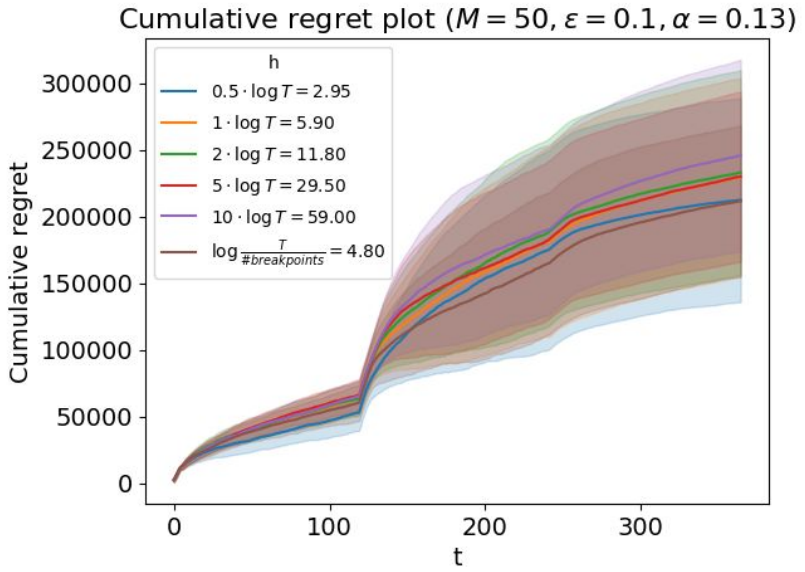
**CUSUM - h analysis**

We inspected five different values of h, that is the threshold over which a **detection is flagged**, namely a change in the conversion probabilities. As suggested in *Liu et al.*[3] we tried values proportional to $\log T$.

In general, **low values of h** makes the **algorithm more sensitive**, while **high values** makes it **less sensitive**.

With our parameters the **best performing h** is the one with **multiplicative factor 0.5**. As the **factor increases** the **cumulative regret gets worse** since the learner needs more evidence to detect a change in the distribution.

In case the number of breakpoints is known, *Liu et al.*[3] propose to use $\log \frac{T}{\#breakpoints}$, that would be optimal in this case as it is possible to see from the plot.




[3] A Change-Detection based Framework for Piecewise-stationary Multi-Armed Bandit Problem, Liu et al., 2017, https://arxiv.org/pdf/1711.03539.pdf

# Step 5: Non-stationary environments with two abrupt changes

## Sensitivity Analysis - UCB-CUSUM

**CUSUM - $\alpha$ analysis**

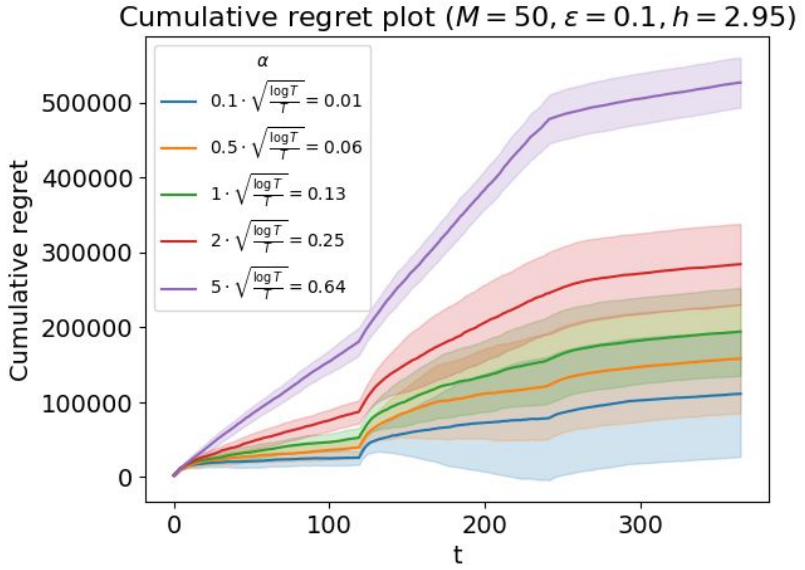
We inspected five different values of $\alpha$, that is the **pure exploration parameter**. As suggested in *Liu et al.*[3] we tried values proportional to $\sqrt{\frac{logT}{T}}$.

$\alpha$ is the probability with which a random arm is played rather than the one recommended by UCB.

If $\alpha$ is **too low**, the **algorithm** is **suboptimal** since it **exploits too much**, pulling almost always the suggested arm and rarely trying other arms that could became optimal in a non-stationary environment.

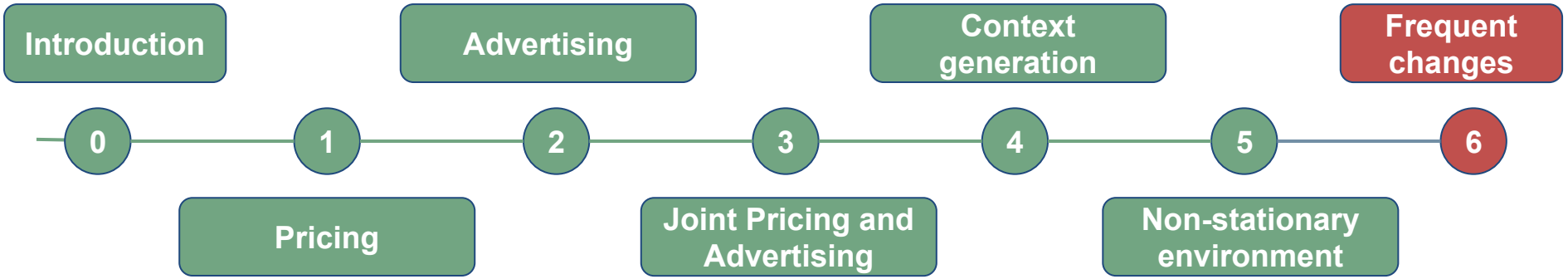
Instead, if $\alpha$ is **too large**, the **learner explores too much**, thus it frequently pulls random arms being able to detect changes, but **not exploiting what has learned**.

From the plot we observe that lower values of $\alpha$ give lower regret.




[3] A Change-Detection based Framework for Piecewise-stationary Multi-Armed Bandit Problem, Liu et al., 2017, https://arxiv.org/pdf/1711.03539.pdf

# Roadmap


Introduction
Advertising
Context generation
Frequent changes
0
1
2
3
4
5
6
Pricing
Joint Pricing and Advertising
Non-stationary environment

# Step 6: Non-stationary environments with many abrupt changes

## Setting 1

In this step the **pricing** part is **unknown**, instead the **advertising** one is **known**. In this case, the pricing curves are subject to **two abrupt changes**. The objective is to **maximize** the **reward** across all the phases while estimating the purchase conversation rates, using algorithms to deal with non-stationary settings.

**Scenario:**

- All the users belong to class **C1**
- Curves related to the advertising part are **known**, the **bid** is **fixed**
- Curves related to the pricing part are **unknown** and **non-stationary**, being subject to **three different seasonal phases**

**Task:**

- Apply **UCB1**, **UCB1 with sliding window**, **UCB1 with change detection test** and **EXP3** to maximize the reward while estimating the conversion probabilities
- Plot the average value and standard deviation of the **cumulative regret, cumulative reward**, **instantaneous regret** and **instantaneous reward**

[A3] Conversion probabilities for the three phases

# Step 6: Non-stationary environments with many abrupt changes

## Approach

This scenario is very similar to the one of step 5, the only differences are that also the algorithm **EXP3** is implemented to deal with the non-stationary setting and that the **bid** is **fixed** (value 2.90€). Therefore it is not necessary to optimize the advertising problem and so the number of clicks and cumulative daily costs are fixed too.

**EXP3** is an algorithm used for **adversarial bandit settings** or to deal with non-stationary settings when no information about the specific form of non-stationarity is known beforehand. It keeps one **weight** for each arm and selects the next arm to pull at random from the probability distribution provided by the weights. The weights are increased when the reward is good. EXP3 depends on the **hyperparameter $\gamma \in [0, 1]$** that **represents the probability of playing a random arm** rather than the one chosen according to the weights. The **closer $\gamma$ is to 1, the more the arms tend to be picked uniformly**. While **the closer $\gamma$ is to 0, the more the arms are pulled according to the weights**.

# Step 6: Non-stationary environments with many abrupt changes

## Approach

**EXP3** works as follows[4] :

1. at the beginning the **weights** associated to the arms are **all initialized to 1**;
2. in **each round** the **algorithm pulls** an arm $i_t$ **at random from the probability distribution provided by the weights** $w_i(t)$. The probability associated to each arm *i* is:

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

*K is the number of arms*

3. the **observed reward** associated to the played arm is **rescaled into the interval [0,1]** obtaining $x_{i_t}(t)$;
4. the estimated reward $\hat{x}_{i_t}(t)$ is defined as $\frac{x_{i_t}(t)}{p_{i_t}(t)}$;

5. **only the weight associated to the played arm is updated** according to:

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

[4] https://jeremykun.com/2013/11/08/adversarial-bandits-and-the-exp3-algorithm/

# Step 6: Non-stationary environments with many abrupt changes

## Results - Setting 1

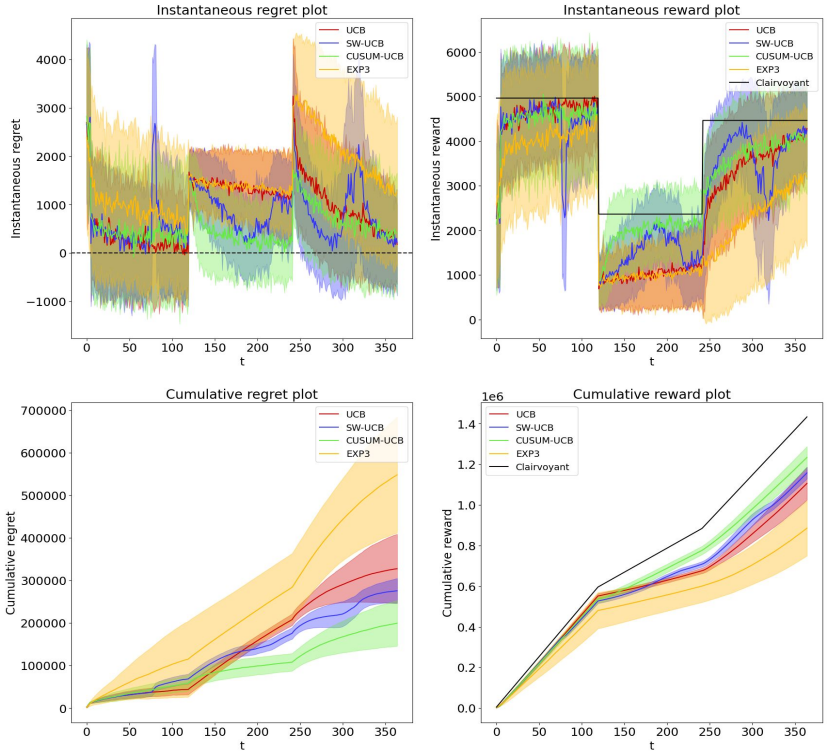
In this setting we want to compare the performances of EXP3 with UCB and its variations in the case of two abrupt changes, running **100 experiments**.

As expected, the **algorithm doesn't perform well** in a context where the **number** of **abrupt changes** is **low**.

The plot of cumulative regret shows that the **regret is sub-linear only in the first phase** and slightly during the third one, but in both phases the **learning occurs slowly** as we see from the plots of instantaneous regret and reward.

It's worth noticing that the cumulative regret of EXP3 is more than double than the regret of SW-UCB and CUSUM-UCB.

Overall, we can say that **EXP3 performs poorly** with respect the other algorithms tested.



# Step 6: Non-stationary environments with many abrupt changes

## Setting 2

In this step the **pricing** part is **unknown**, instead the **advertising** one is **known**. In this case, the pricing curves are subject to **many abrupt changes**. The objective is to **maximize** the **reward** across all the phases while estimating the purchase conversation rates, using algorithms to deal with non-stationary settings.

**Scenario:**

- All the users belong to class **C1**
- Curves related to the advertising part are **known**, the **bid** is **fixed**
- Curves related to the pricing part are **unknown** and **non-stationary**, being subject to **five different phases** that **cyclically change** with a **high frequency**

**Task:**

- Apply **UCB1**, **UCB1 with sliding window**, **UCB1 with change detection test** and **EXP3** to maximize the reward while estimating the conversion probabilities
- Plot the average value and standard deviation of the **cumulative regret**, **cumulative reward**, **instantaneous regret** and **instantaneous reward**

[A4] Conversion probabilities for the five phases

# Step 6: Non-stationary environments with many abrupt changes

## Approach

This setting is very similar to the previous one, with the only difference that now there are no more only three phases spread over the time horizon but instead there are **five different phases**, **each one associated** with a **different optimal price**, which **alternate cyclically** with a **high frequency**. The five phases have different lengths and last 16, 21, 23, 15 and 19 days, so between two or three weeks.

This **adversarial setting** could be **caused by** the presence of **competitor companies** that launch their smartphones or make important discounts during the year affecting the conversion probabilities of the customers of our product. In addition, during the year there are **discount periods** like *Back to School* and *Cyber Monday* that **can cause** a **change** in the **purchase habits of** the **customers** for just short periods.

The **high non-stationary degree** makes really difficult to address this scenario because the learners don't have enough rounds to learn the best arm so they tend to play suboptimally in all phases. This is expected for UCB which is not tailored for this kind of setting.
The interaction between the learners and the environment remains the same as it was in the previous step because there is only the aforementioned difference.

# Step 6: Non-stationary environments with many abrupt changes

## Results - setting 2

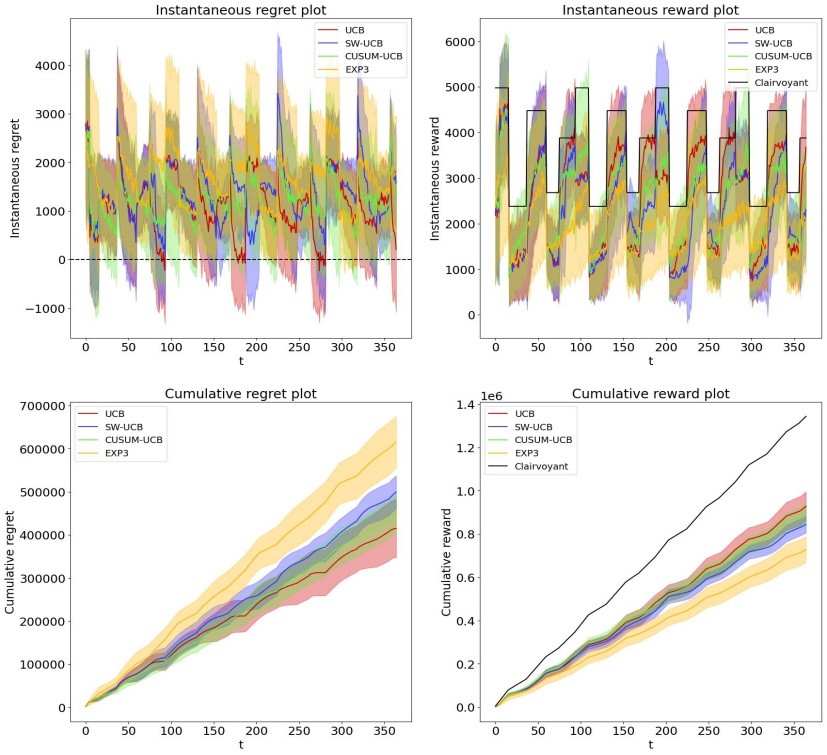
Also in this case we did **100 experiments** to obtain the plots.

It is clear that none of the algorithms can deal with this environment, since all achieve a **linear regret**. With a cumulative regret of over 100 thousand € compared to the others, EXP3 is by far the worst of the four.

The two non stationary variants of UCB perform worse than UCB itself, which results to be the best one.

We tried also different values for $\gamma$, but the outcome doesn't improve. Two **variations** between our implementation and the original algorithm may be the cause of **EXP3's poor performance**:

- we **rescale** the profit in [0, 1] for each observation, but as there are only five distinct prices and hence only five unique profits, then there are only five possible values as output of this normalization;
- at each round the learner receive around 100 observations and for each one the associated weight is updated, so there is the risk that this **weight** will **increase significantly** relative to the others.




POLITECNICO
MILANO 1863

# References


- [1] Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design, Srinivas et al. (2010), https://arxiv.org/pdf/0912.3995.pdf
- [2] On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems,Garivier, A., & Moulines, E. (2008), https://arxiv.org/pdf/0805.3415.pdf
- [3] A Change-Detection based Framework for Piecewise-stationary Multi-Armed Bandit Problem, Liu et al., 2017, https://arxiv.org/pdf/1711.03539.pdf
- [4] https://jeremykun.com/2013/11/08/adversarial-bandits-and-the-exp3-algorithm/

# Appendix - A1

⇦ [A1] Here it follows the precise numbers used for the conversion probabilities of the three classes for step 1 to 4:

| Conversion Probabilities per class | | | | | |
|---|---|---|---|---|---|
| Class \ Price (€) | 500 | 550 | 600 | 650 | 700 |
| C1 | 0.10 | 0.12 | 0.20 | 0.04 | 0.03 |
| C2 | 0.03 | 0.04 | 0.10 | 0.12 | 0.20 |
| C3 | 0.20 | 0.12 | 0.10 | 0.04 | 0.03 |

Steps 5 and 6 require a detailed analysis on non-stationary settings for a single class, thus these values change. View the relevant appendices.

# Appendix - A2

⇦ [A2] Here the equations used to plot the clicks and costs curves:

$$f(bid) = scale * (1 - e^{-slope*bid}),\ bid \in [0.5, 10]$$

| Clicks | | |
|---|---|---|
| Class \ Parameters | Scale | Slope |
| C1 | 100 | 3 |
| C2 | 90 | 1 |
| C3 | 70 | 0.5 |

| Cumulative daily costs | | |
|---|---|---|
| Class \ Parameters | Scale | Slope |
| C1 | 35 | 1 |
| C2 | 25 | 0.5 |
| C3 | 15 | 0.2 |

# Appendix - A3

⇦ [A3] Here it follows the precise numbers used for the conversion probabilities of the three phases for step 5 and step 6 case 1:

| Conversion Probabilities per phase | | | | | |
|---|---|---|---|---|---|
| Phase \ Price (€) | 500 | 550 | 600 | 650 | 700 |
| Phase1 | 0.07 | 0.10 | 0.10 | 0.20 | 0.10 |
| Phase2 | 0.08 | 0.06 | 0.12 | 0.03 | 0.03 |
| Phase3 | 0.18 | 0.30 | 0.15 | 0.02 | 0.02 |

# Appendix - A4

⇦ [A4] Here it follows the precise numbers used for the conversion probabilities of the five phases for step 6 case 2:

| Conversion Probabilities per phase | | | | | |
|---|---|---|---|---|---|
| Phase \ Price (€) | 500 | 550 | 600 | 650 | 700 |
| Phase1 | 0.07 | 0.10 | 0.10 | 0.20 | 0.10 |
| Phase2 | 0.08 | 0.06 | 0.12 | 0.03 | 0.03 |
| Phase3 | 0.18 | 0.30 | 0.15 | 0.02 | 0.02 |
| Phase4 | 0.27 | 0.12 | 0.06 | 0.02 | 0.01 |
| Phase5 | 0.09 | 0.06 | 0.12 | 0.10 | 0.13 |