AA. 2022-2023

Online Learning Applications

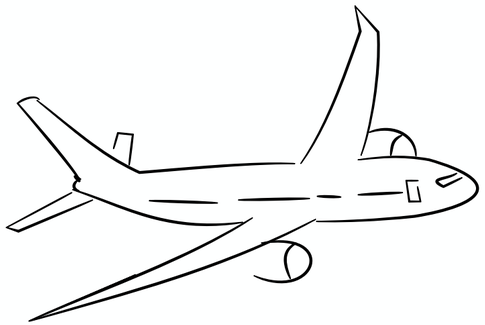
“Pricing and Advertising strategy for E-Commerce of Flight Booking”

Filippo Lazzarin, Alberto Pirillo,
Michele Simeone, Simone Tognocchi

POLITECNICO
MILANO 1863

Overview

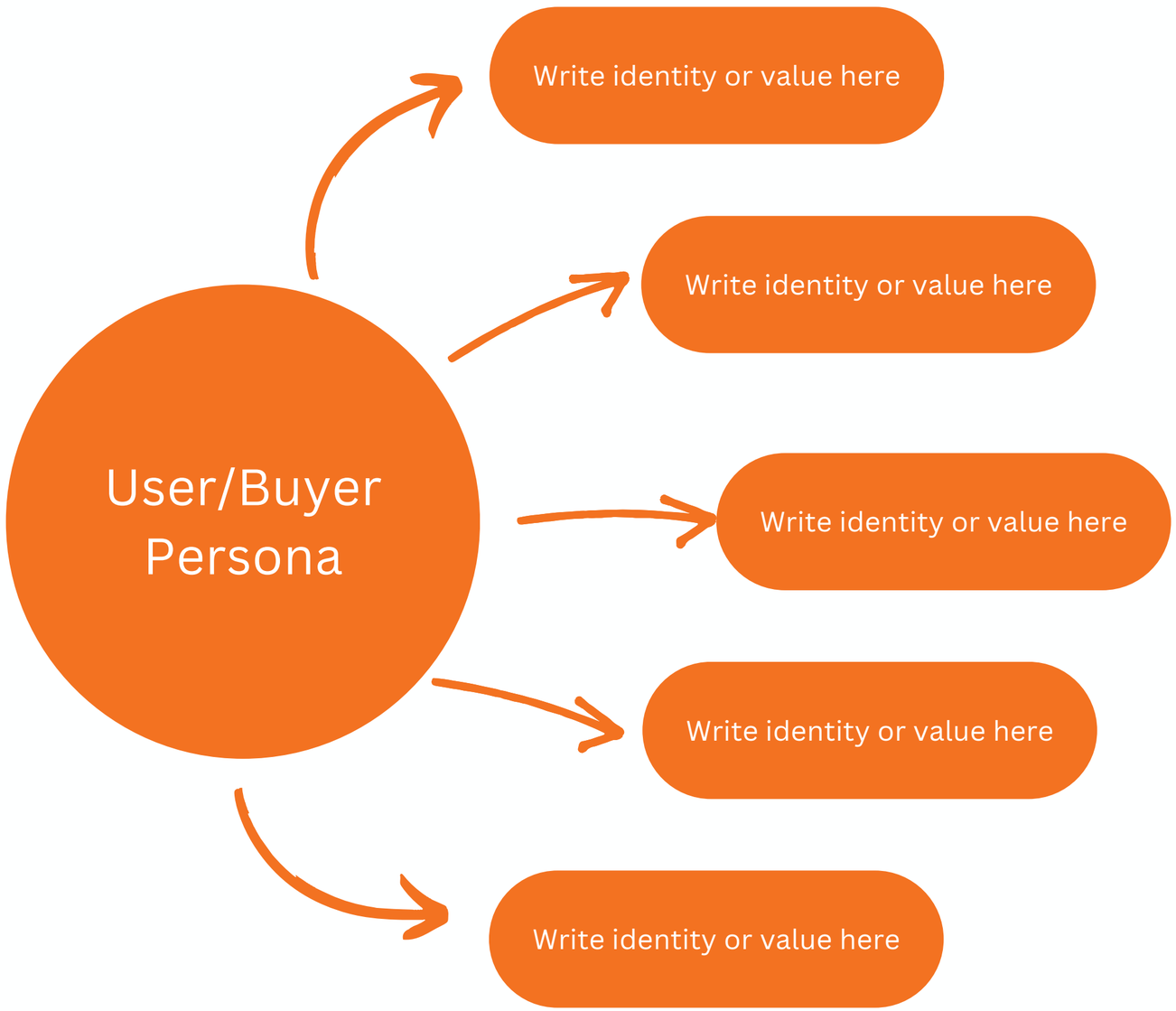
- Problem setting
- Environment design
- Clairvoyant algorithm



Steps

1. Learning for pricing
2. Learning for advertising
3. Learning for joint pricing and advertising
4. Contexts and their generation
5. Dealing with non-stationary environments with two abrupt changes
6. Dealing with non-stationary environments with many abrupt changes

Classes and Prices



Problem setting

Every day there are more than 26000 flights in the world. An **airline company** which sells plane tickets in its **e-commerce** platform wants to maximize its profit by dynamically learning an optimal pricing and advertising strategy.

Pricing: the same ticket could be sold at 5 different prices. The company wants to know the best one, i.e. the one that maximizes the **product of the conversion rate by the price**. The conversion rate represents whether a user who has seen the ticket will buy it.

Standard **Multi-Armed Bandits (MABs)** algorithms can be used to learn the conversion rates in an online fashion efficiently.

Advertising: the company wants to develop an **online advertising campaign** so that many users will discover and see the services it offers by visiting the webpage of a **publisher.** The company has to select a **bid** (the maximum amount that it is willing to pay to the publisher): selecting a higher bid will result in more users seeing the advertisement but also in more costs charged by the publisher to the company.

Gaussian Process (GP) Bandits algorithms can be used to dynamically learn the optimal bid while minimizing the loss in the meantime.

Environment Design - Classes

Based on the information that we retrieve from the purchase of the ticket we can divide customers into different segments.

The features that we have chosen to classify users with are:

- Whether a customer chooses to buy **checked luggage**
- Whether a customer chooses to buy a **first-class ticket**

Based on these features we created three different contexts:

- First class of users is represented by those **under 25**
- Second class by adult users with an age in the **range of 25-40**
- Third class by **over 40**

The **under-25** do not pick up any feature for their ticket.
The **adult** users pick up just one between checked luggage and a first-class ticket.
The **over-40** class is the one that chooses both features when purchasing the ticket.

Environment Design - Conversion Rates

The flight market is affected by **seasonality** which subsequently influences the conversion rates.

In general, we can split the time horizon (**365 days**) into **three periods**:

- **Winter period** goes from October to January
- **Spring period** from February to May
- **Summer period** from June to September

Usually, tickets tend to have higher conversion rates during summer, while conversion rates tend to lower during winter and are the lowest in the spring period.

When we decided on the conversion rates we tried to be as faithful as possible to real-world case scenarios while using values that allow for easily visualizable results.

Environment Design - Prices and Bids

We propose **5 different ticket price** ordered in ascending order based on the inclusion of the checked luggage and the first class ticket features. The possible ticket prices are the following: [15.5, 30.7, 60.2, 70.6, 90.8].
Every user class has a **Bernoulli distribution** associated with each price.

Moreover, the three proposed classes **differs** in terms of:

- the function that expresses the number of **daily clicks** as the **bid varies**
- the function that assigns the **cumulative daily cost** of the clicks as the **bid varies**

Both the functions are **concave curves** obtained by using an exponential function with different constants and then by adding **Gaussian noise**.

An example of these functions can be seen in the following plots:

Clairvoyant algorithm

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Experiment parallelization

When working with **stochastic processes**, it is required to run multiple simulations of the same scenario in order to **smooth out the stochasticity**. We refer to these simulations as *experiments*.

The **proper number of experiments** depends on the environments and on the algorithms used. We found that pricing algorithms (e.g. UCB1, TS) required much more experiments w.r.t. advertising algorithms (e.g. GPU-UCB, GP-TS).

Every experiment is **independent** from the others. Therefore, we can **run multiple experiments in parallel** on different CPU cores, using the *map* function of the *multiprocessing* Python module.

Essentially, we **encapsulated the entirety of an experiment inside of a single function**. Then, the function is executed in parallel on different cores. In the end, the results are collected in a common data structure.

Observed speedups are in the range of **5x to 7x** depending on the number of available CPU cores.
Parallelization was the reason why we were able to perform this many experiments.

JSON Environments

In order to duplicate the least amount of code possible and ensure that all experiments are run in the same environment (when required), we **store the parameters** of the **environment** inside of a **JSON file.**

As a consequence, when we instantiate an object the **Environment** class, the parameters are **read** from a JSON file.

```
{
    "num_classes": 3,
    "bids": [0, 1, 100],
    "prices": [15.5, 30.7, 60.2, 70.6, 90.8],
    "noise_mean": 0.0,
    "noise_std": 5.0,
    "conv_rates": [
        [0.36, 0.77, 0.51, 0.28, 0.12],
        [0.22, 0.27, 0.61, 0.59, 0.35],
        [0.12, 0.16, 0.31, 0.62, 0.59]
    ],
    "class_probabilities": [1.0, 0.0, 0.0]
}
```

```
{
    "num_classes": 3,
    "bids": [0, 1, 100],
    "prices": [15.5, 30.7, 60.2, 70.6, 90.8],
    "noise_mean": 0.0,
    "noise_std": 5.0,
    "conv_rates": [
        [0.36, 0.77, 0.51, 0.28, 0.12],
        [0.22, 0.27, 0.61, 0.59, 0.35],
        [0.12, 0.16, 0.31, 0.62, 0.59]
    ],
    "class_probabilities": [0.34, 0.33, 0.33]
}
```

```
{
    "num_classes": 3,
    "bids": [0, 1, 100],
    "prices": [15.5, 30.7, 60.2, 70.6, 90.8],
    "noise_mean": 0.0,
    "noise_std": 5.0,
    "phase_length": 120,
    "class_probabilities": [1.0, 0.0, 0.0],
    "conv_rates": [
        [[0.36, 0.77, 0.51, 0.28, 0.12],
        [0.22, 0.27, 0.61, 0.59, 0.35],
        [0.12, 0.16, 0.31, 0.62, 0.59]],

        [[0.29, 0.68, 0.41, 0.6, 0.4],
        [0.17, 0.21, 0.53, 0.50, 0.28],
        [0.8, 0.11, 0.25, 0.53, 0.50]],

        [[0.40, 0.81, 0.42, 0.32, 0.14],
        [0.26, 0.35, 0.69, 0.67, 0.40],
        [0.20, 0.18, 0.35, 0.65, 0.62]]
    ],
    "horizon": 365
}
```

Struttura delle slide sugli step

- Descrizione dello step (preso dalla specifica e tradotto/decifrato)
- Introduzione ad algoritmi non ancora usati nella presentazione (e.g. UCB1)
- Risultati (plot)
- Commenti sui risultati **(FONDAMENTALE COMMENTARE I VALORI DEGLI IPERPARAMETRI UTILIZZATI FORNENDO UNA SPIEGAZIONE PRATICA)**

Step 1: Learning for pricing

In a specific scenario where all users belong to a single class, let's call it "C1," we have some information about the advertising aspect of a problem, but we lack information about the pricing aspect.

To tackle this, we will apply two algorithms: **UCB1** and **TS** (Thompson Sampling).

Multi-Armed Bandit Problem

Imagine a scenario where you have multiple slot machines (arms) to choose from, each with an unknown probability of giving a reward.

The **goal** is to minimize your total regret over a series of trials by deciding which arms to pull.

In our scenario the arms are the given price.

UCB1

Key Idea:
UCB1 balances the exploration-exploitation trade-off, trying to both explore the arms with uncertain rewards and exploit arms that seem promising.

Upper Confidence Bound (UCB):

- For each arm, UCB1 calculates an upper confidence bound representing the uncertainty in the estimated reward.
- This upper bound is based on the arm's historical performance and a confidence level parameter.

Benefits:

- UCB1 is simple, efficient, and widely used in various applications, such as online advertising, clinical trials, and recommendation systems.
- It strikes a balance between exploration and exploitation, leading to good overall performance.

UCB1

Considerations:

- The choice of the confidence level parameter impacts the algorithm's behavior. A smaller value encourages exploration, while a larger value favors exploitation.
- UCB1 assumes that the underlying reward distribution remains stationary, which may not hold in some dynamic environments.

Performance:

UCB1 has been theoretically analyzed and shown to achieve near-optimal regret bounds, making it a strong choice for solving the multi-armed bandit problem.

Thompson Sampling (TS)

Key Idea:

Thompson Sampling takes a Bayesian approach to the problem, modeling the uncertainty about the reward probabilities for each arm.

Probability Distributions:

- For each arm, Thompson Sampling maintains a probability distribution (usually a Beta distribution) over possible reward probabilities.
- These distributions represent our beliefs about the arms' performance.

Benefits:

- Thompson Sampling is effective in situations where the reward probabilities can change over time or exhibit complex patterns.
- It adapts quickly to emerging trends and uncertainties in the environment.

Thompson Sampling (TS)

Considerations:

- The choice of probability distribution to model arm rewards can affect the algorithm's performance.
- Thompson Sampling requires maintaining and updating probability distributions, which may be computationally intensive for a large number of arms.

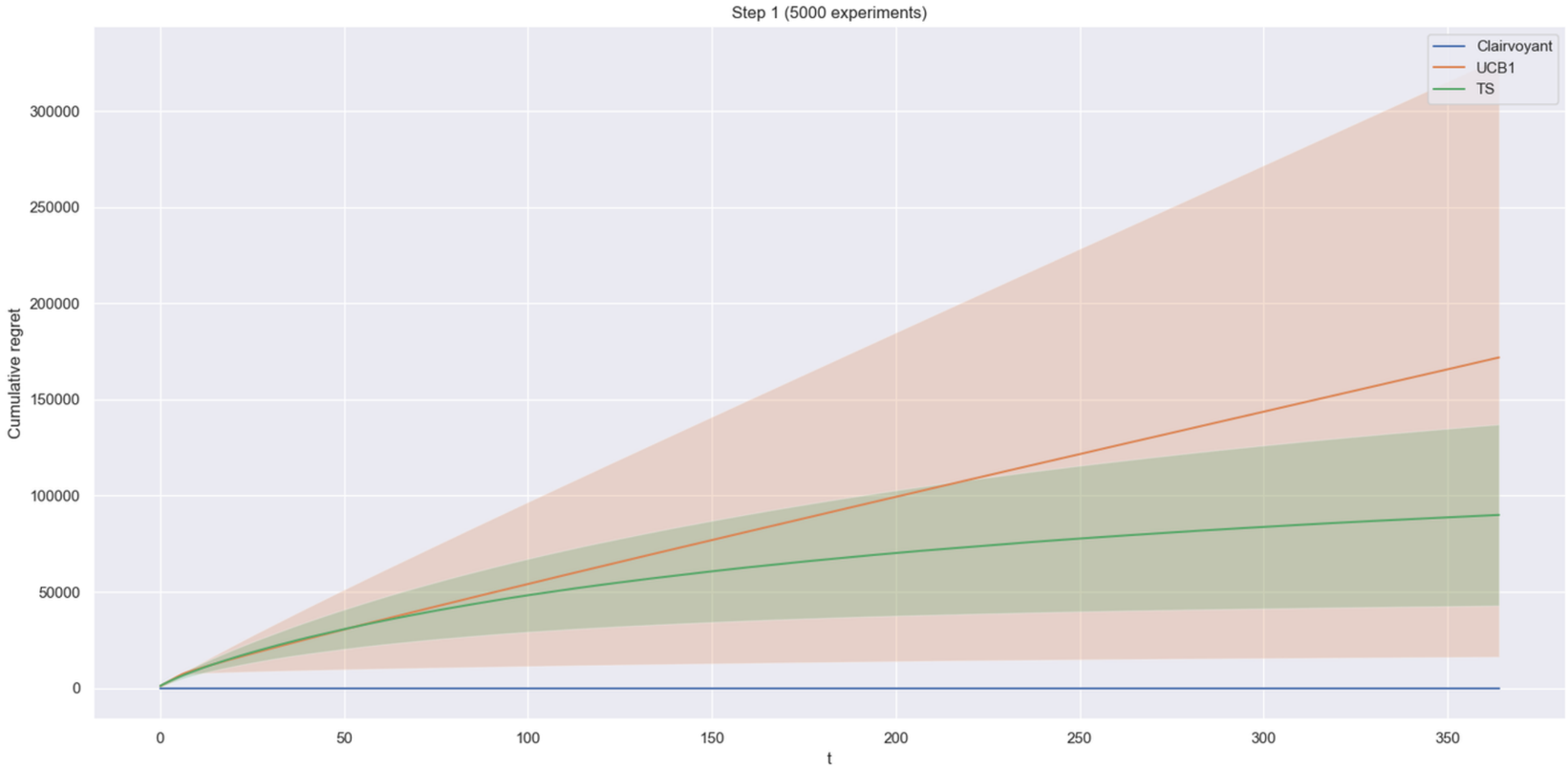
Performance:

Thompson Sampling has been proven to achieve near-optimal regret bounds in various scenarios, making it a robust choice for solving the multi-armed bandit problem.

Cumulative Regret

The objective function to maximize is defined as the reward.

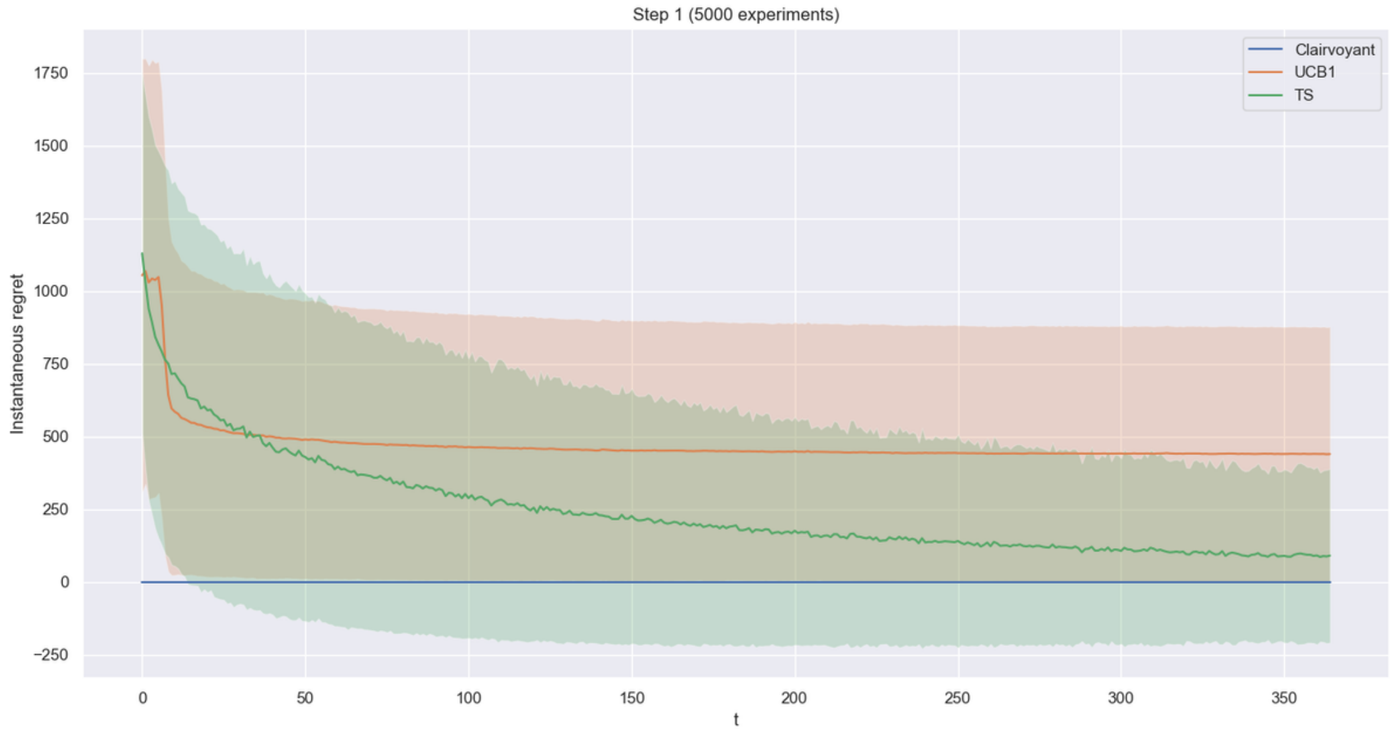
- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.



Instaneous Regret

The objective function to maximize is defined as the reward.

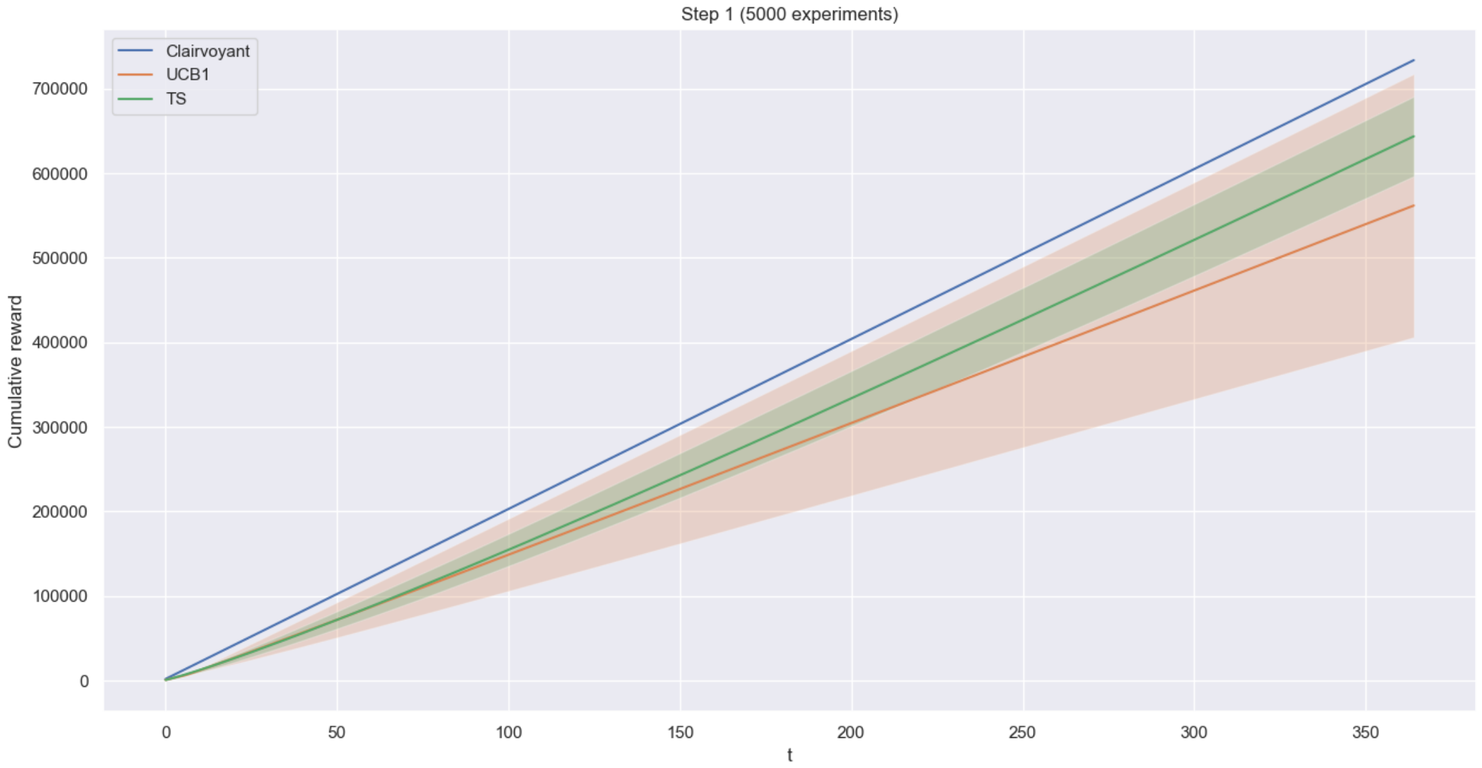
- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.



Cumulative Reward

The objective function to maximize is defined as the reward.

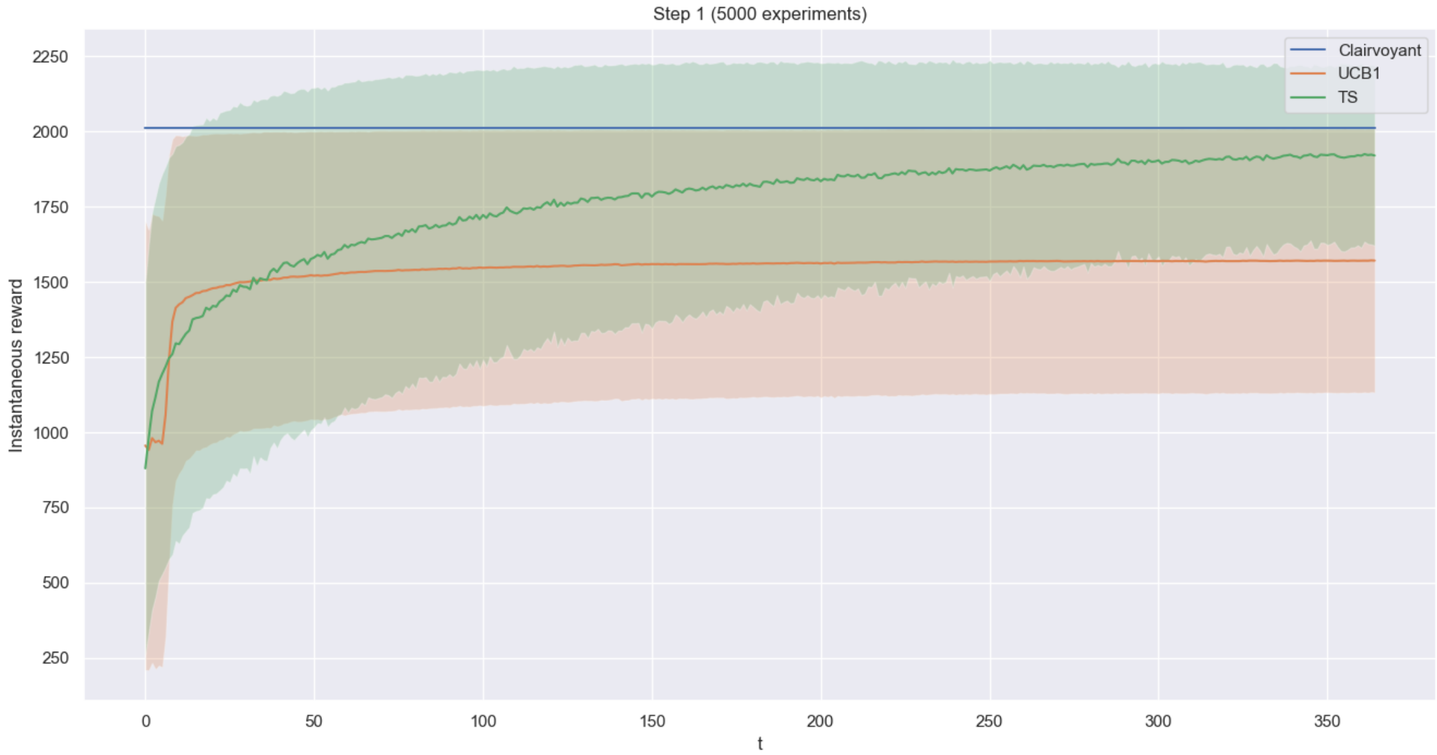
- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.



Instaneous Reward

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.



Comments and Conclusion

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Step 2: Learning for advertising

In the same scenario of Step 1, now we know the pricing aspect of a problem, but we lack information about the advertising aspects.

To tackle this, we will apply two algorithms: **GP-UCB** and **GP-TS**, both of which rely on Gaussian Processes (GPs) to model the uncertainty in the advertising curves.

Gaussian Process

GP-UCB and GP-TS utilizes Gaussian Processes, a **probabilistic modeling technique**, to represent our uncertainty about the arms' reward distributions.

GPs provide a flexible framework for **modeling complex, unknown functions.**

GP-UCB & GP-TS

Considerations:

- The choice of GP hyperparameters and the kernel function significantly impact algorithm performance.
- GP-MAB assumes that the reward distribution remains stationary, which may not hold in dynamic environments.

Performance:

GP-MAB has been widely applied in various domains and has demonstrated strong empirical performance.

Cumulative Regret

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Instaneous Regret

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Cumulative Reward

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Instaneous Reward

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Comments and Conclusion

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Step 3: Learning for joint pricing and advertising

In the same scenario of Step 1, we have no prior knowledge about the advertising and pricing patterns.

To tackle this, we combine the algorithms of the previous 2 steps.

SHORTLY RESOLUTIVE METHOD

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Cumulative Regret

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Instaneous Regret

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Cumulative Reward

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Instaneous Reward

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Comments and Conclusion

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Step 4: Contexts and their generation

In a complex situation, we have three different groups of users **(C1, C2, and C3)**, and we **lack** any prior information about how advertising and pricing strategies should be applied to them. We are exploring two scenarios:

1. **Scenario 1:** we already know the structure of the user contexts so we understand how different users are grouped or categorized beforehand.
2. **Scenario 2:** In the second scenario, we do not have prior knowledge of the context structure. We need to **learn** this structure from the available data. It's important to note that in this scenario, **we don't even know how many user contexts exist.**

To address both scenarios, we will use two algorithms: GP-UCB and GP-TS. Additionally in Scenario 2, we will pair these algorithms with a context generation algorithm.

At the end, we will also run GP-UCB and GP-TS algorithms without context generation. We'll treat all users as if they belong to a single context for the entire duration of the analysis.

Classes

Use the graph to present the expenses associated with the campaign.

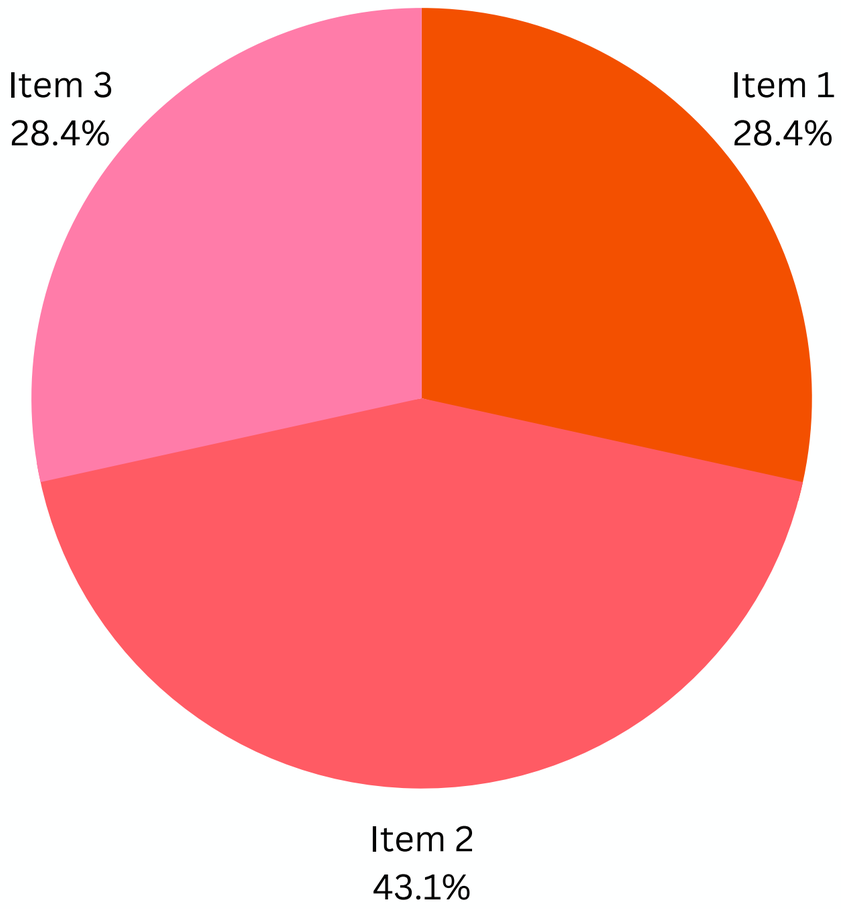
01 List the item and briefly explain it.

02 List the item and briefly explain it.

03 List the item and briefly explain it.

04 List the item and briefly explain it.

05 List the item and briefly explain it.



CONTEXT GENERATION ALGORITHM

Real-time context is the information available at the moment of decision-making.
In some cases, it **may not be readily** accessible or is **too costly** to obtain.

The Offline Context Generation algorithm **bridges the gap** by creating context data in advance.
It generates context features based on **historical data, patterns, or known factors relevant to the decision task**.

Considerations:

- The quality and relevance of the generated context features play a critical role in the algorithm's effectiveness.
- Regular updates to the offline-generated context may be necessary to adapt to changing conditions.

SPLIT CONDITION & HOEFFDING BOUND

After calculate the value of the best arms in each testing context for the choice pf the best one we select the maximum between the lower bound wighted sum of the classes belonging at each context.

INSERIRE IMMAGINE DISEQUAZIONE E HOEFDFDING BOUND

Michele Simeone

Hoeffding Bound is a useful tool for estimating lower bounds on probabilities when working with limited sample data. It provides a statistically sound method to calculate lower bounds with a specified level of confidence, making it valuable in various data-driven applications.

Summary Step 4.1 and 4.3

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Cumulative Regret

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Instaneous Regret

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Cumulative Reward

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Instaneous Reward

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Comments and Conclusion

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Step 5: Dealing with non-stationary environments with two abrupt changes

Now we have a single-user class called C1.
We know the curves related to advertising problems, but we have no prior information about the pricing curve. Additionally, the pricing curves are not constant; they change over time in a **non-stationary** manner, following **three distinct seasonal phases** throughout the time period.

To address this situation, we will apply the UCB1 algorithm along with two variations of the UCB1 algorithm designed specifically for handling non-stationary data.

1. **Passive Non-Stationary UCB1:** using **Sliding Window**
2. **Active Non-Stationary UCB1:** using **Change Detection**

NON STATIONARY ENVIRONMENT: ABRUPT CHANGES

Phases	Price1 - 15.5	Price2 - 30.7	Price3 - 60.2	Price4 - 70.6	Price5 - 90.8
Phase 1	0.36	0.65	0.51	0.28	0.12
Phase 2	0.29	0.32	0.41	0.60	0.40
Phase 3	0.40	0.81	0.42	0.32	0.14

SW-UCB1

Sliding window is a technique used to deal with **non-stationary environment.**
In this step we use the same UCB algorithm implemented for the previous steps but with the difference that now takes into account only the **last k samples**. This allows the algorithm to keep track of the **changes** within the environment.

In this case we used a window size of **100 days**, that is approximately every time the seasonality change, in order to obtain better performances. This algorithm doesn't work well with small windows size since the learner won't be able to learn in an efficient way.

CD-UCB1

In this case we implement a **Change Detection algorithm** in order to deal with the abrupt changes of the non-stationary environment. We take the same UCB algorithm used in the previous steps and we apply a **CUSUM algorithm**.
CUSUM consider every small positive and negative **variation w.r.t a mea**n computed on the first **k rewards.**

If the sum of all positive or negative variations exceeds a **defined bound** the algorithm **clears the mean**, and the algorithms starts the learning phase again. The defined bound is a hyperparameter of the problem and the optimal one was selected after performing some hyperparameter tuning.

Cumulative Regret

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Instaneous Regret

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Cumulative Reward

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Instaneous Reward

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

Comments and Conclusion

The objective function to maximize is defined as the reward.

- For 1 class the reward is defined as: number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising
- For multiple classes the reward is: the sum of the rewards provided by the single classes.

We have approximated the continuous set of bids with a finite set of 100 bids.

The optimization algorithm works as follow:

1. for every single class find the best price, independently from the other classes;
2. optimize the bid for each class independently from the other classes.

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

Now, we use the **EXP3 algorithm** in the previous step scenario.

Next, we'll consider a different non-stationary scenario characterized by a **higher degree of non-stationarity**. This increased degree of change can be modeled by introducing **five phases**, each associated with a different optimal price.
These phases will **rapidly** and **cyclically** change.

In this new scenario, we'll apply the EXP3 algorithm, the UCB1 algorithm, and the two non-stationary variations of UBC1.

EXP3

Key Idea:

The Exponential-weight algorithm for Exploration and Exploitation. It works by maintaining a list of weights for each arm, using these weights to decide randomly which action to take next, and increasing (decreasing) the relevant weights when a payoff is good (bad)

Algorithm:

1. Given $\gamma \in [0, 1]$, initialize the weights $w_i(1) = 1$ for $i = 1, \ldots, K$.
2. In each round t:
    1. Set $p_i(t) = (1 - \gamma)\dfrac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \dfrac{\gamma}{K}$ for each i.
    2. Draw the next action i_t randomly according to the distribution of $p_i(t)$.
    3. Observe reward $x_{i_t}(t)$.
    4. Define the estimated reward $\hat{x}_{i_t}(t)$ to be $x_{i_t}(t)/p_{i_t}(t)$.
    5. Set $w_{i_t}(t + 1) = w_{i_t}(t)e^{\gamma \hat{x}_{i_t}(t)/K}$
    6. Set all other $w_j(t + 1) = w_j(t)$.

EXP3

Exponential Weights:

- EXP3 assigns **exponentially decreasing weights** to arms based on their historical performance.
- This weighting scheme encourages **exploration** by giving **less-weighted arms a chance to be selected.**

Performance:

EXP3 has been widely studied and has theoretical guarantees for regret bounds, making it a strong choice for solving the multi-armed bandit problem.

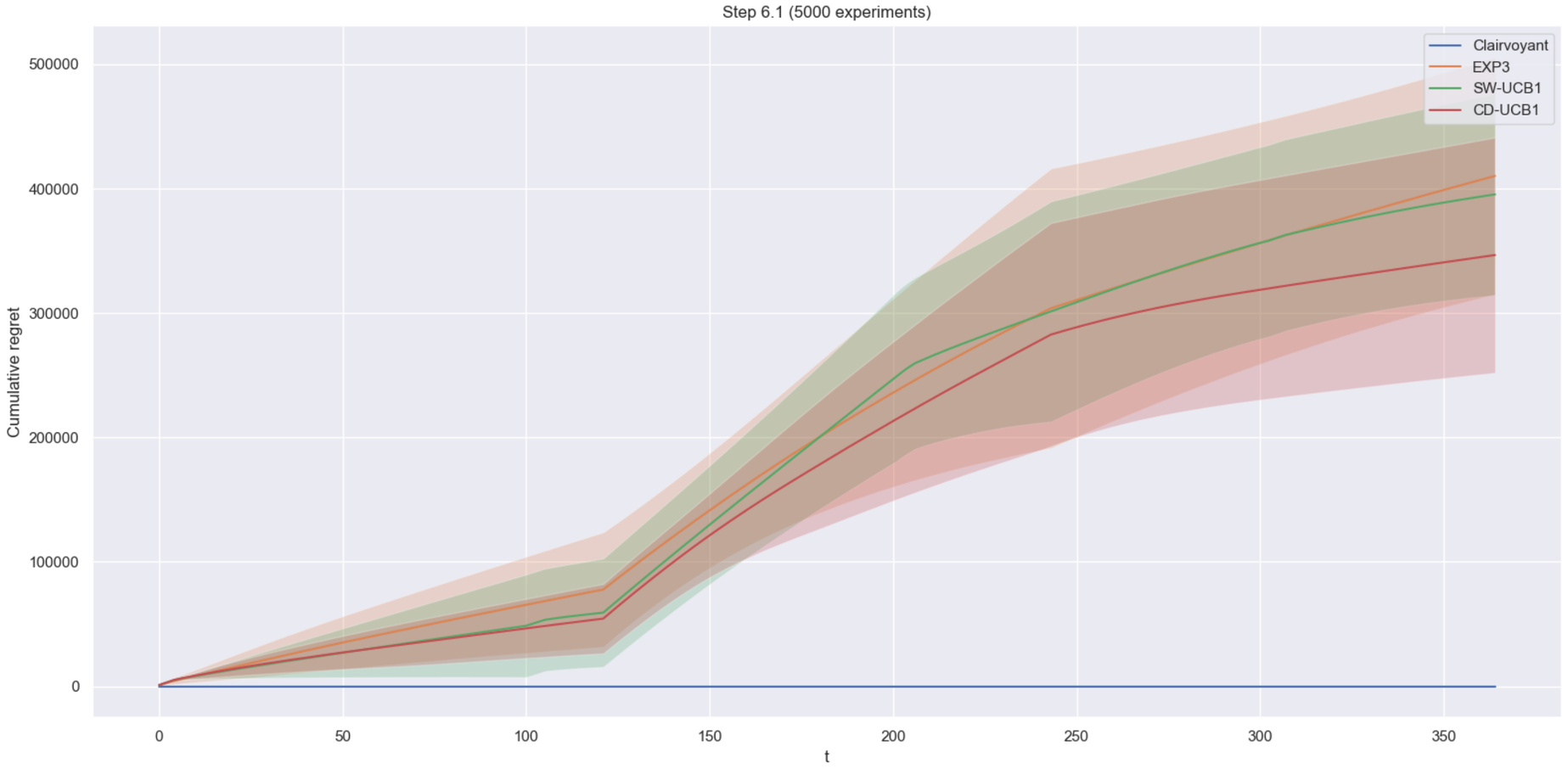
Goal and expectations:

Firstly we consider the scenario of Step 5 with the bid fixed and we expect that EXP3 performs worse than the two non-stationary versions of UCB1.

Then we consider a different non-stationary setting with higher non-stationary degree: 5 phases, associated with different optimal prices, that change more frequently. Now we expect that EXP3 outperforms the non-stationary version of UCB1.

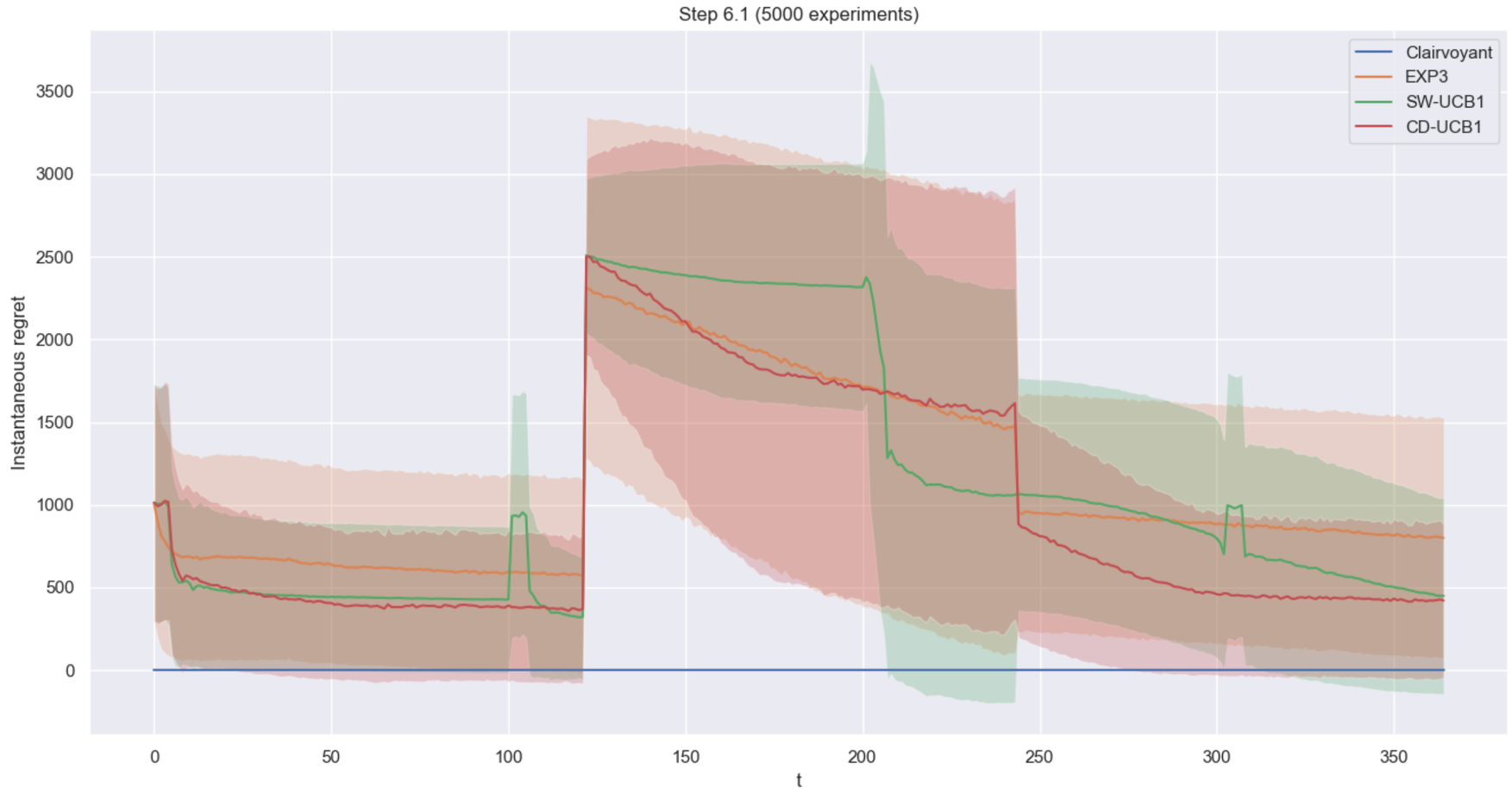
Cumulative Regret

In this case the bid is fixed and we consider 3 different phases.



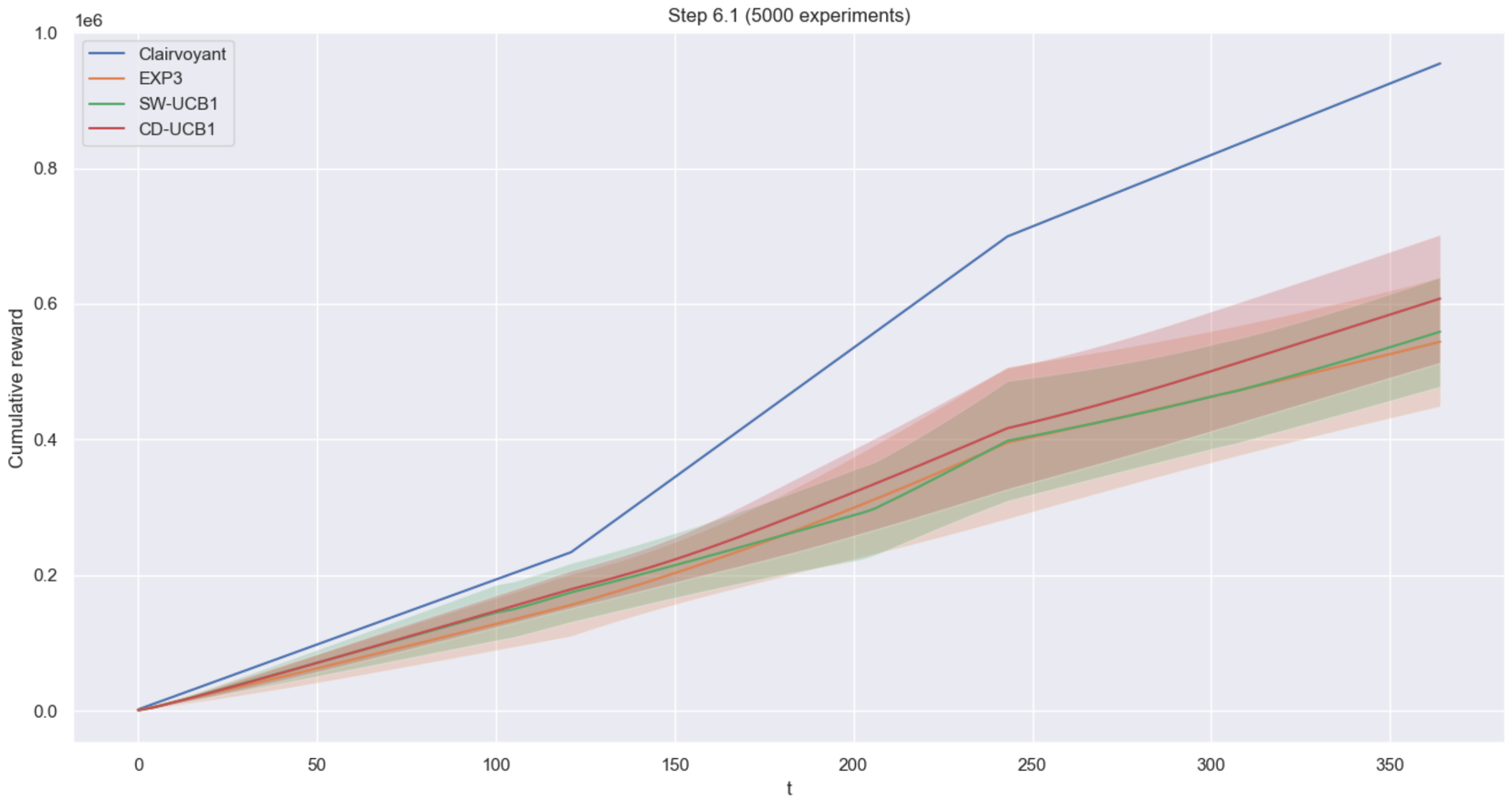
Instaneous Regret

In this case the bid is fixed and we consider 3 different phases.



Cumulative Reward

In this case the bid is fixed and we consider 3 different phases.

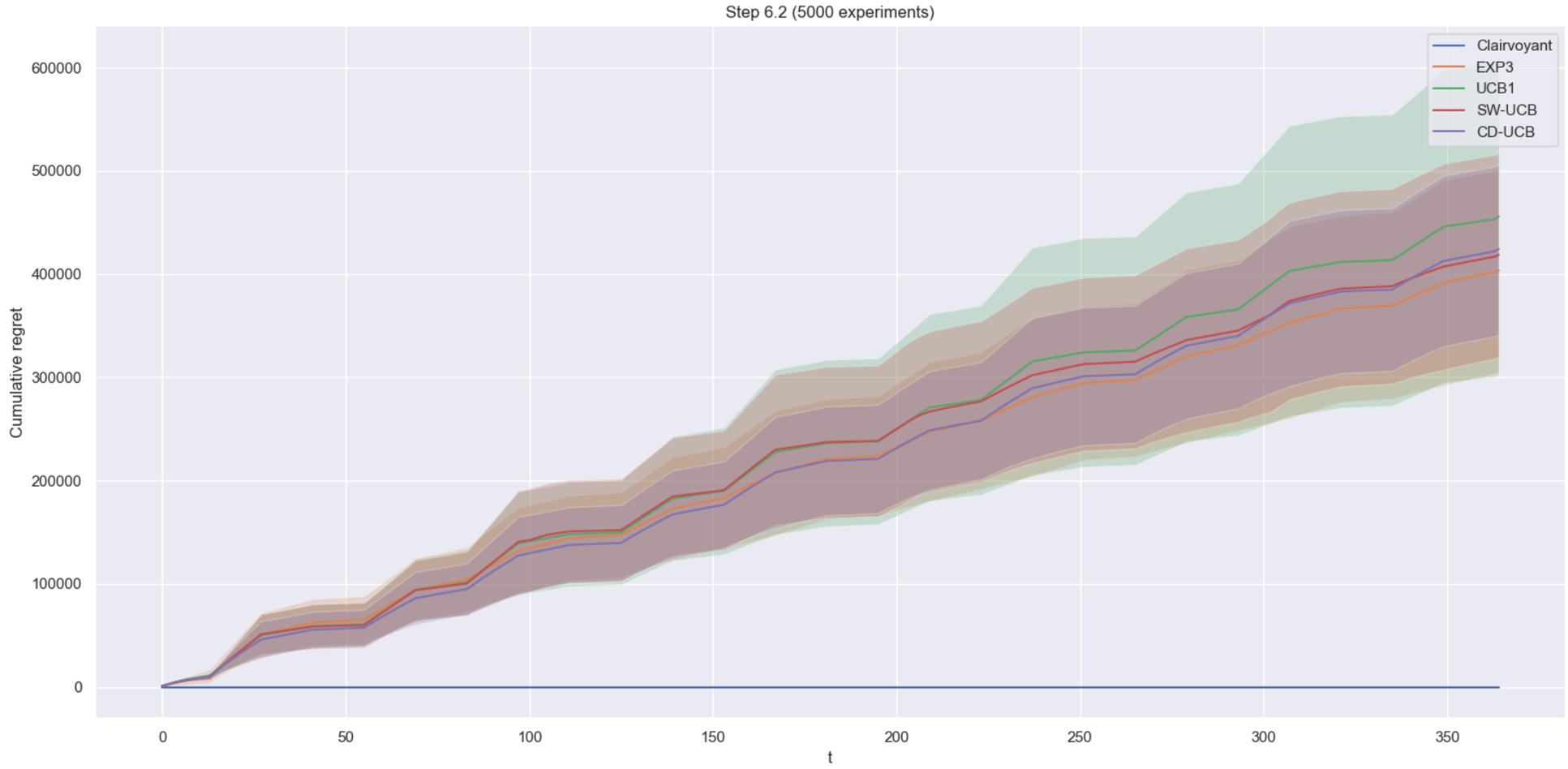


Instaneous Reward

In this case the bid is fixed and we consider 3 different phases.

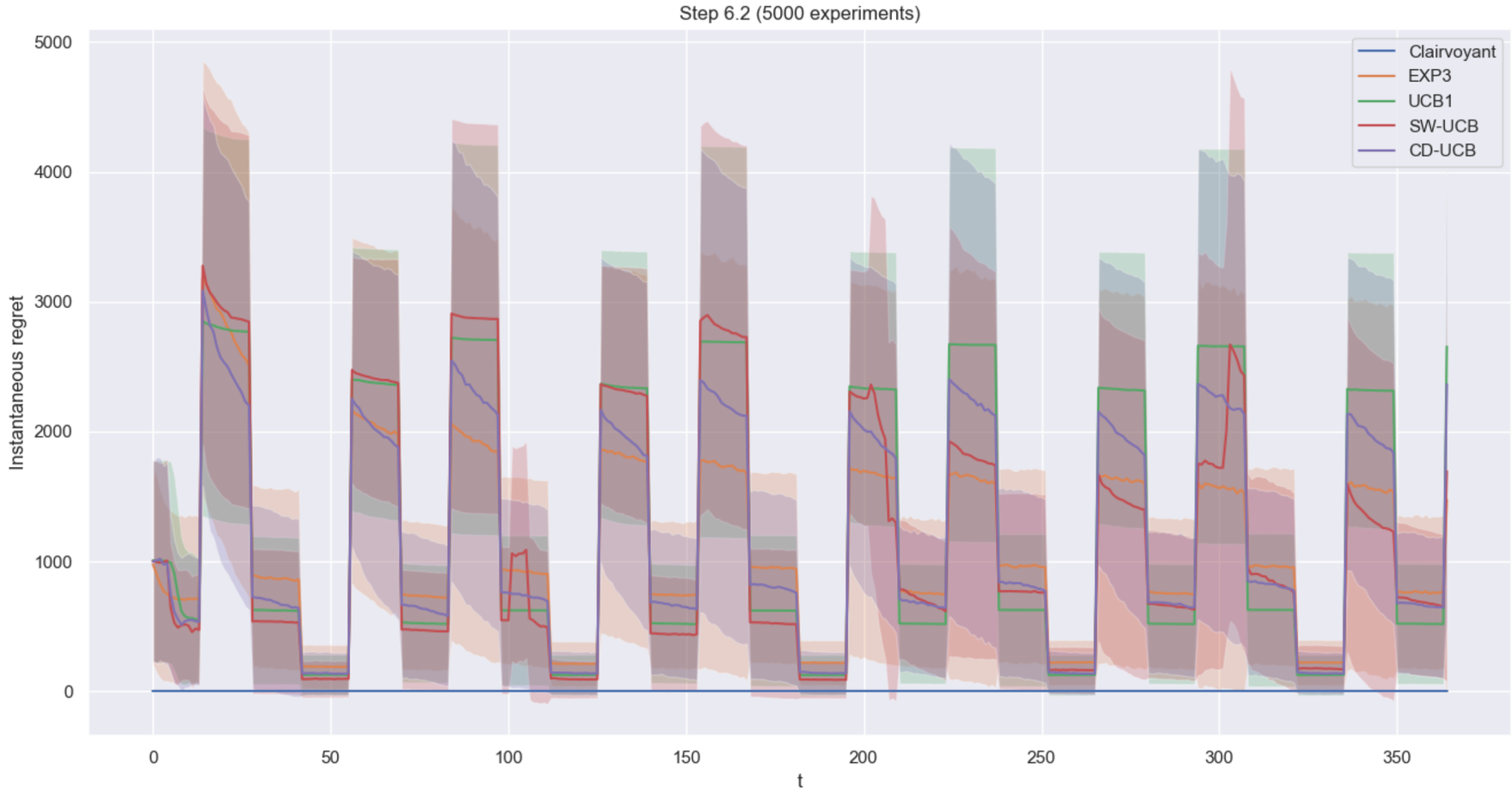
Cumulative Regret

In this case the bid is not fixed anymore and we are considering 5 phases that change with high frequency and each one has an optimal price.



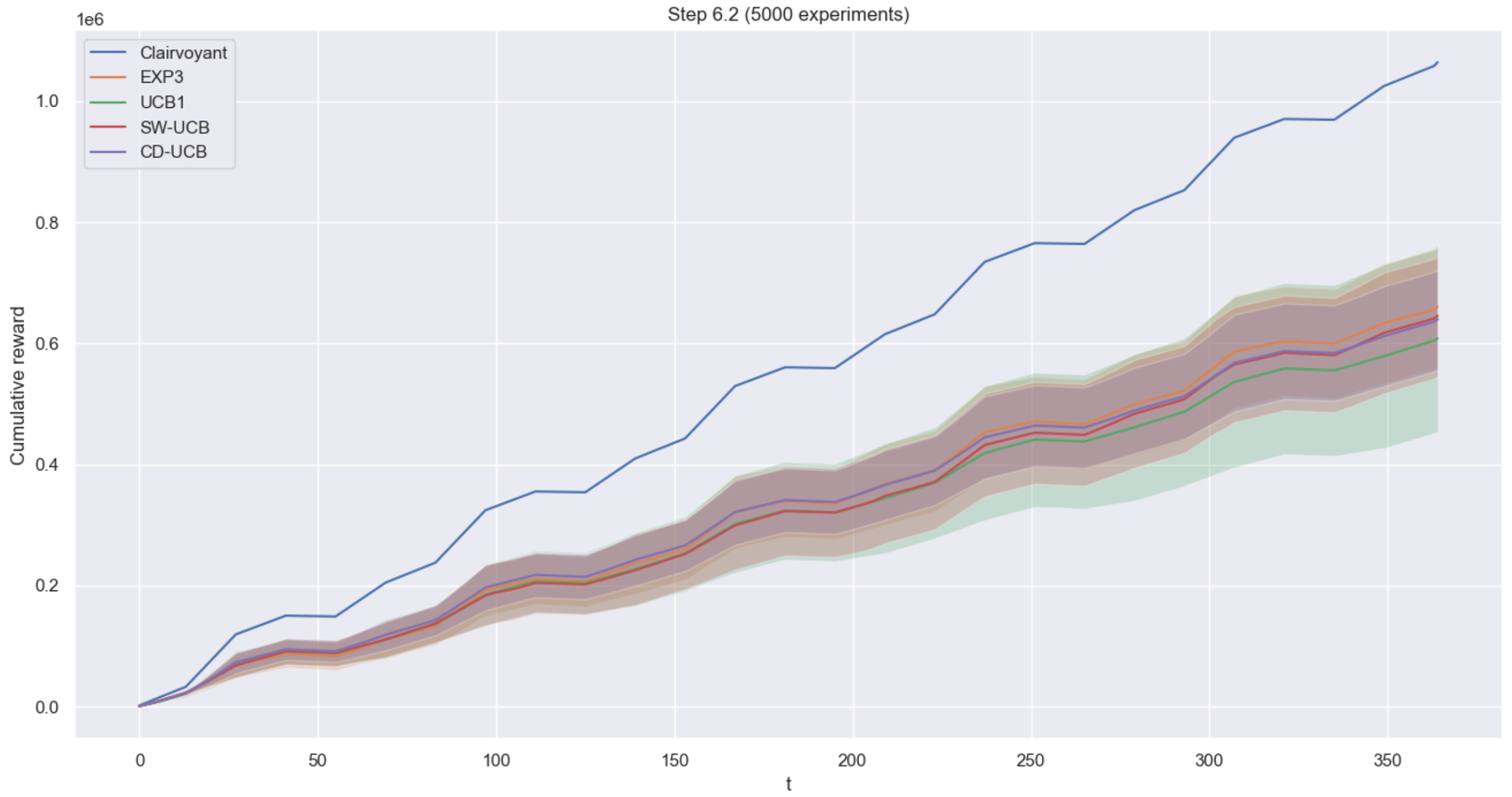
Instaneous Regret

In this case the bid is not fixed anymore and we are considering 5 phases that change with high frequency and each one has an optimal price.



Cumulative Reward

In this case the bid is not fixed anymore and we are considering 5 phases that change with high frequency and each one has an optimal price.



Instaneous Reward

In this case the bid is not fixed anymore and we are considering 5 phases that change with high frequency and each one has an optimal price.

Comments and Conclusion

As we expected in the first case, the environment like the step 5, but with fixed bid, EXP3 performs worse than the two non-stationary UCB1.

In the second case, with 5 phases, that change with high frequency, EXP3 performed better than the non-stationary UCB1.

Summary 6 steps

Lay out the timeline for the marketing activities and initiatives that will make the campaign succcesful.

01

Write another activity, deadline or milestone here.

02

Write another activity, deadline or milestone here.

03

Write another activity, deadline or milestone here.

04

Write another activity, deadline or milestone here.

05

Write another activity, deadline or milestone here.

06

Write another activity, deadline or milestone here.

Improvement

Explain how the following channels will help reach the campaign's target audience.

01

Marketing Channel 1

02

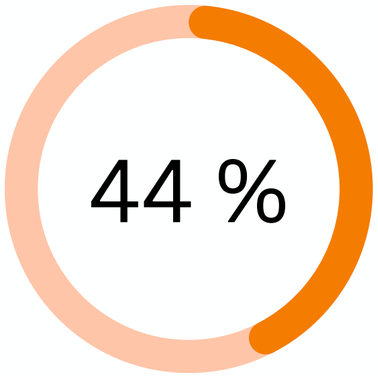
Marketing Channel 2

03

Marketing Channel 3

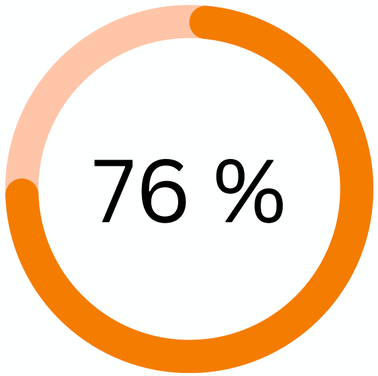
Key Performance Indicators

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis vulputate nulla at ante rhoncus, vel efficitur felis condimentum. Proin odio odio.



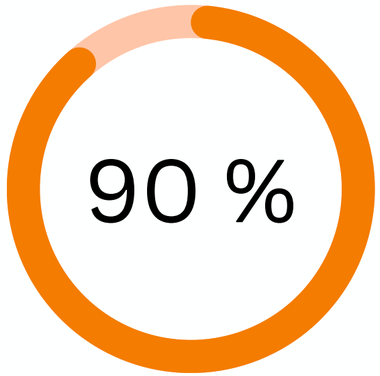
KPI 01

Briefly elaborate on the KPI.



KPI 02

Briefly elaborate on the KPI.



KPI 03

Briefly elaborate on the KPI.

Team

Michele Simeone

Add role here

Simone Tognocchi

Add role here

Alberto Pirillo

Add role here

Filippo Lazzarin

Add role here

GOT QUESTIONS?

Reach out.

123-456-7890

hello@reallygreatsite.com

LICERIA &
CO.

RESOURCE PAGE