Online Learning Applications

Option 1:

Pricing-Advertising

POLITECNICO
MILANO 1863

Pietro Gadaleta (10801034)
Adrian Perez (10913548)

0 Motivations and environment desing

MOTIVATION

- Our company is a Travel Agency, and we sell special travel packages to visit different parts of the world.

- We want to introduce to our products a new package to visit Milan, which includes flights, activities, hotels, etc.

- For this new package, we want to study the best price which will lead to the best benefit. Also, the best marketing strategy to maximize our sales.

TRAVELAGENCY

0 Motivations and environment desing

PRODUCT PRICE

- Our new product has the following characteristics:
  - Cost for the company: 210€
  - Price: our price is to be defined.
    - The values that we are studying are: [350, 400, 450, 500, 550].

ADVERTISING STRATEGY

- We can observe two binary features through the advertising platform:
  - **F1: Age of the customer** (under or below 45 years old).
  - **F2: Country of residence of the customer** (Italy residence or not).

TARGET COSTUMERS

- With the two binary features we can observe from the advertising platform, we can split our costumers into three classes:

Classes of Customers

C1

F1: young people

F2: foreign or italian residence

Behaviour: As young people, their travel style is more conservative and looks for the cheapeast options.

C2

F1: old people

F2: italian residence

Behaviour: Are willingness to pay more in order to have the best accomodation and activities. But, as they already live in italy, it is not so exclusive experience.

C3

F1: old people

F2: foreign residence

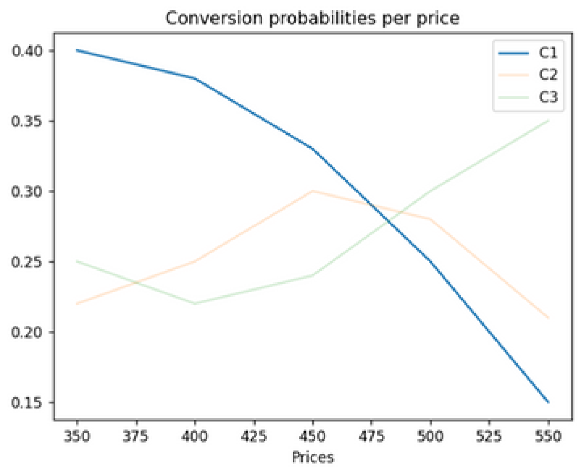
Behaviour: Are willingness to pay more in order to have the best accomodation and activities. Since Italy is a foreign contry, the travelpack is considered as a luxury experience.

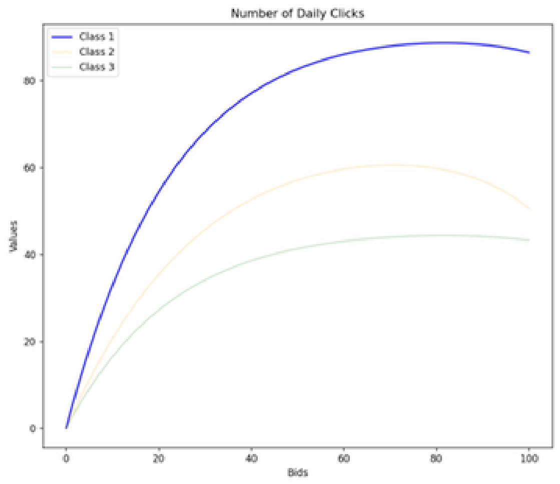
0 Motivations and environment desing

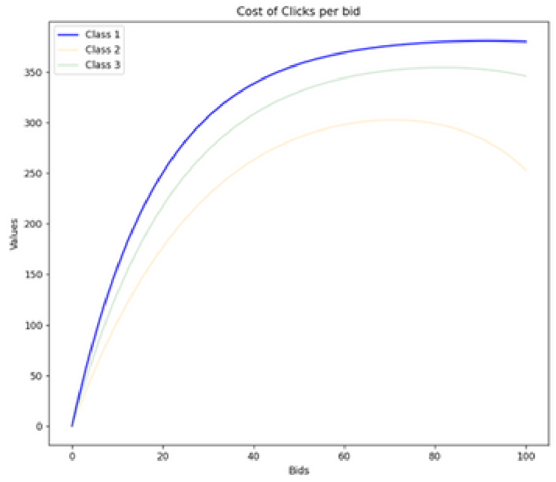
Classes of Customers

C1

This are our young potential customers. As young people, their travel style is more conservative and looks for the cheapest options. For this reason, the conversion probabilities decrease as the price increases, reaching a maximum probability on the lowest price. Also, they are very familiar with internet commerce, so they have the biggest daily clicks on our web side. The cost of the clicks is also high for this reason.





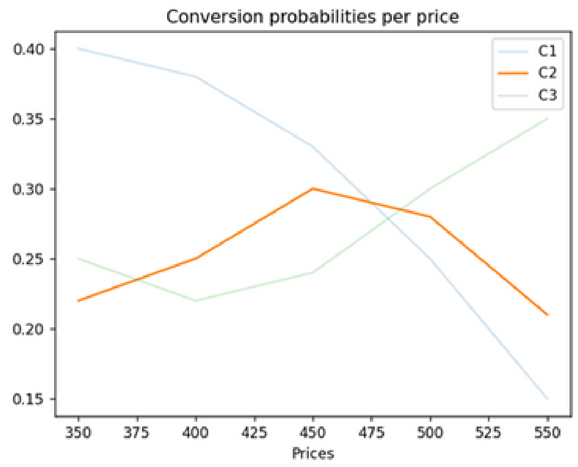


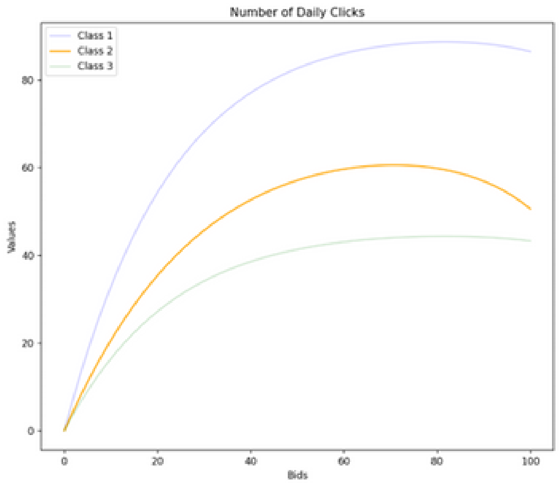
0 Motivations and environment desing

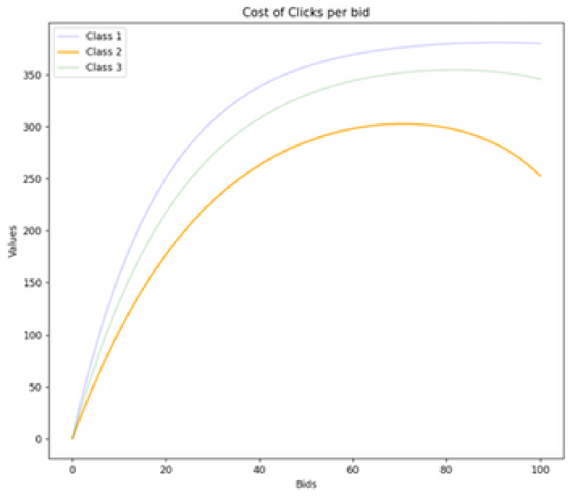
Classes of Customers

C2

·This class corresponds to old people that live in Italy: they are willingness to pay more to have the best accommodation and activities. But, as they already live in Italy, is not so exclusive experience, so they are looking for a good price while keeping a good quality of the product (the best conversion probability occurs in an average price).





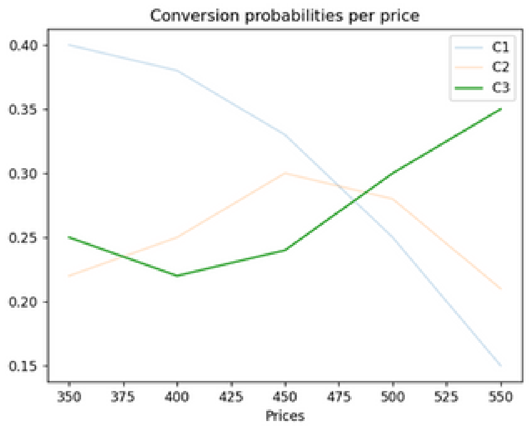


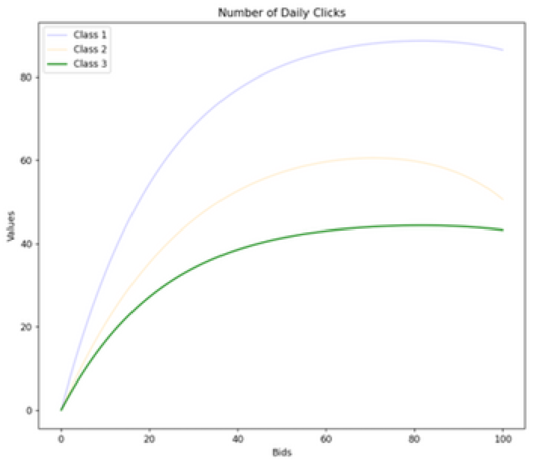
0 Motivations and environment desing

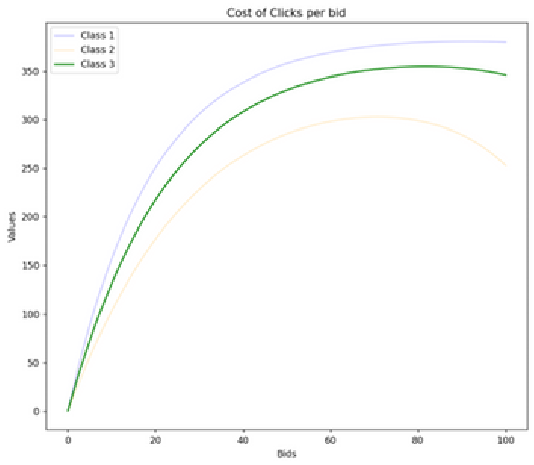
Classes of Customers

C3

·This class corresponds to old people that live in Italy: they are willingness to pay more to have the best accommodation and activities. Since Italy is a foreign country, the travel pack is considered as a luxury experience, so they perceive the high price as a good quality and experience. For this reason, the conversion probability is bigger when the price is increases).
They aren't familiar with internet commerce, so the daily clicks is lower than the young people. The cost of the clicks is a little higher than C2 due to foreign ad services are more expensive.







0 Motivations and environment desing

- The reward function of our problem is:

$$Reward = \sum_{i=C1}^{C3} NC(bids) \cdot CP(price) \cdot (price - cost) \cdot CC(bids)$$

$$\begin{gathered} CP \equiv Conversion\ Probability \\ NC \equiv Number\ of\ daily\ Clicks \\ CC \equiv Cost\ of\ click \end{gathered}$$

1 Learning for pricing

INITIAL INFORMATION

- We are only going to use class C1.
- The curves related to the advertising part are known.
- The curve related to the pricing problem is not.

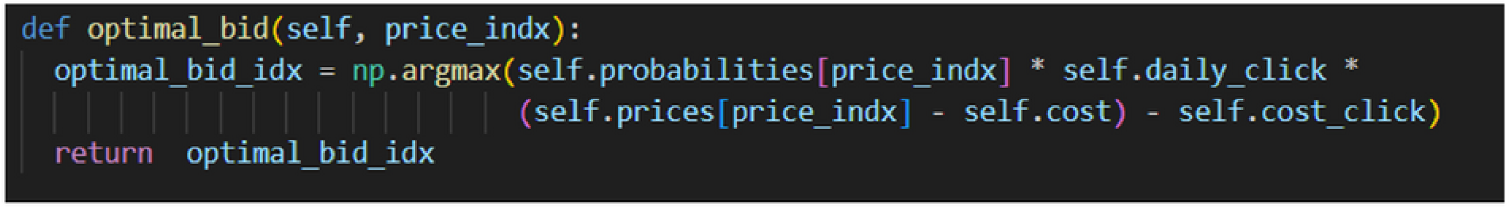
GOAL

- With this information, the goal of the first step is to apply UCB1 and TS algorithm to estimate the conversion probabilities and find the best price.

- Also, plot all the results in order to show: cumulative regret, cumulative reward, instantaneous regret, instantaneous reward.

Environment

- Due to the advertising part of the problem are known, we are going to select always the bid that maximize the reward. We are going to do this by implementing a function that finds this optimal value of bid:



```
def optimal_bid(self, price_indx):
    optimal_bid_idx = np.argmax(self.probabilities[price_indx] * self.daily_click *
                                (self.prices[price_indx] - self.cost) - self.cost_click)
    return  optimal_bid_idx
```

$$Reward = NC(bids) \cdot CP(price) \cdot (price - cost) \cdot CC(bids)$$

- This method simulates a round of interaction with the environment, given the pulled_arm (the chosen bid) by the agent. It calculates the reward, conversion rate, and number of clicks for the selected bid, for every round.

TS_Learner

- The way TS_learner works, is:
  - For every arm, we draw a sample according to the corresponding Beta distribution.
  - We choose the arm with the best sample. At every time t, we play the arm that:

$$a_t \leftarrow \arg max_{a \in A} \{\tilde{\theta}_a\}$$

- We update the Beta distribution of the chosen arm according the observed realization:

$$(\alpha_{a_t}, \beta_{a_t}) \leftarrow (\alpha_{a_t}, \beta_{a_t}) + (x_{a_t t}, 1 - x_{a_t t})$$

1 Learning for pricing

UCB1_Learner

- The way UCB1_Learner works is:
  - In our case, we initially apply a function (find_first_zero) in order to select every arm before applying the algorithm, so every one is considered:
  - Every arm is associated with an upper confidence bound.
  - At every round, the arm with the highest upper confidence bound is chosen.

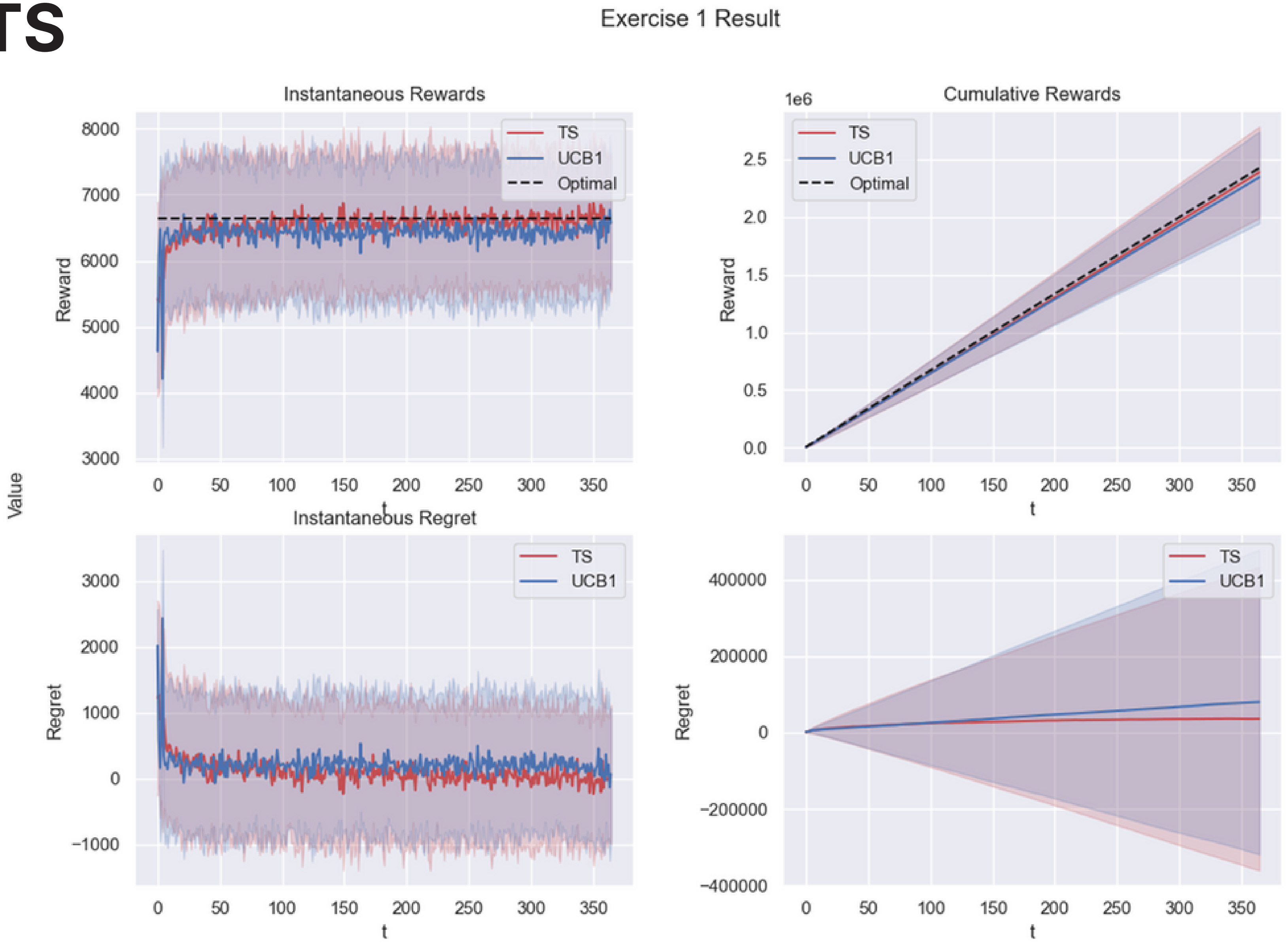
$$a_t \leftarrow \arg max_{a \in A} \left\{ \sqrt{\frac{2 \log(t)}{n_a(t-1)}} \right\}$$

- After having observed the realization of the reward of the arm, the upper confidence bound is uplated.
- The pull_arm method returns the index of the selected arm.

1 Learning for pricing

RESULTS

- n_exp = 100
- T = 365



1 Learning for pricing

RESULTS

- As we can see in the Instantaneous Reward and Instantaneous regret graph, the TS algorithm achieves optimal solution.
- In the other hand, the UCB1 has a lower perform, but also achieves a great solution.
- ·These results were expected, since UCB1 algorithm works with upper confidence bounds, which leads to a lowest accuracy to find the best arm, since it's more sensible to the number of samples. Using TS algorithm, we obtain an accurate result, thanks to a small variance of the Beta distribution.
- We can attribute the similar results in UCB1 and TS due to the predominance of one hand against the others.
- The two algorithms converge, but the TS goes faster.

2 Learning for advertising

INITIAL INFORMATION

- We are only going to use class C1.
- The curves related to the advertising part are NOT known.
- The curve related to the pricing problem known.

GOAL

- With this information, the goal of the first step is to apply GP-UCB and GP-TS to find the advertising curves.

- Also, plot all the results in order to show: cumulative regret, cumulative reward, instantaneous regret, instantaneous reward.

2 Learning for advertising

Bidding environment

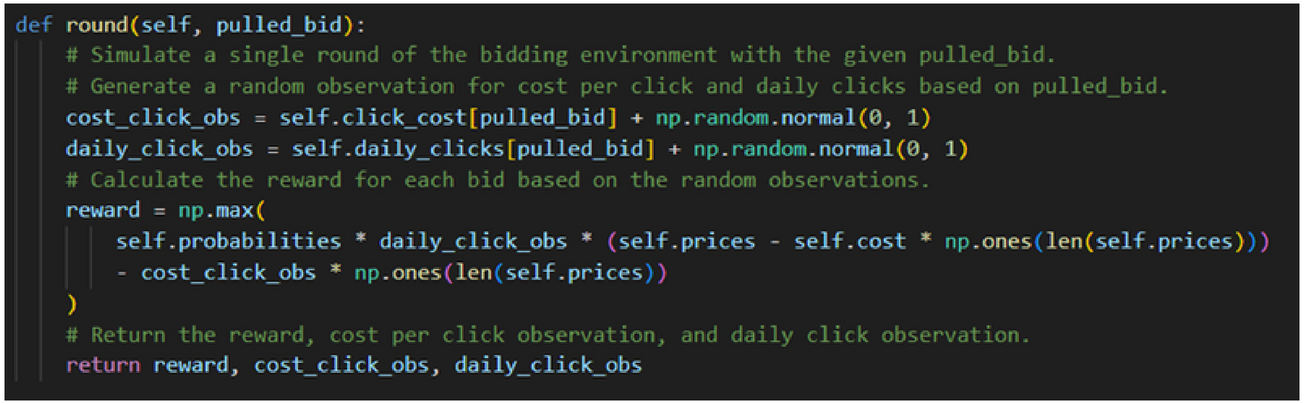
- We are only going to use class C1. The curves related to the advertising part are NOT known. The curve related to the pricing problem known.

Advertising curve

- To calculate the advertising curves, we use Gaussian processes:
    - We use them to store the information we obtain about number of clicks and cost of the clicks.
    - Arms have correlation among them (smooth curves).
    - We predict mean and standard deviation of click cost. The standard deviation is used to calculate the uncertainty.

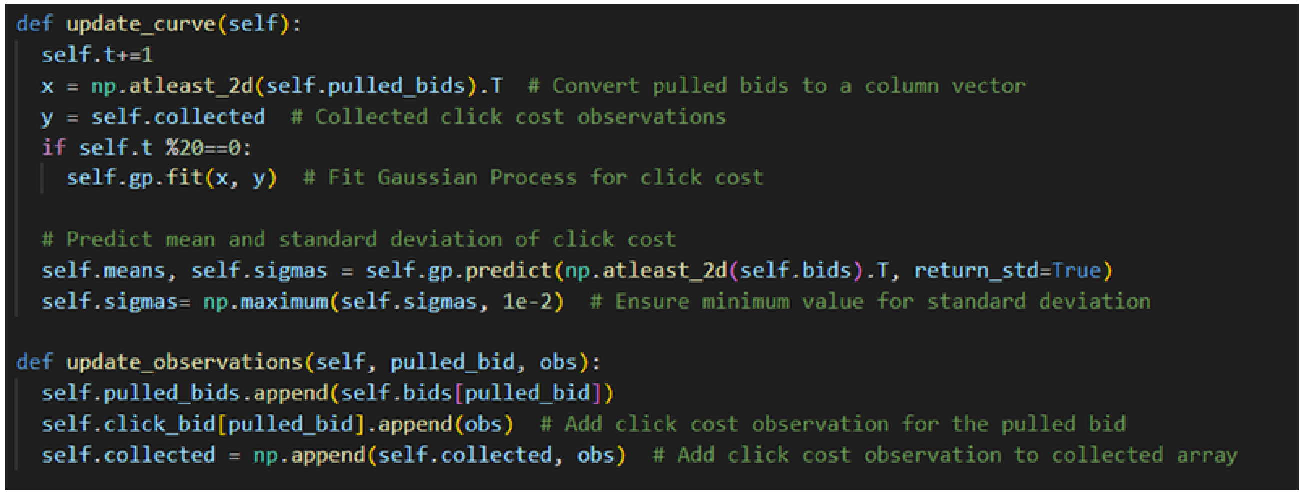
2 Learning for advertising

Bidding environment



```python
def round(self, pulled_bid):
    # Simulate a single round of the bidding environment with the given pulled_bid.
    # Generate a random observation for cost per click and daily clicks based on pulled_bid.
    cost_click_obs = self.click_cost[pulled_bid] + np.random.normal(0, 1)
    daily_click_obs = self.daily_clicks[pulled_bid] + np.random.normal(0, 1)
    # Calculate the reward for each bid based on the random observations.
    reward = np.max(
        self.probabilities * daily_click_obs * (self.prices - self.cost * np.ones(len(self.prices)))
        - cost_click_obs * np.ones(len(self.prices))
    )
    # Return the reward, cost per click observation, and daily click observation.
    return reward, cost_click_obs, daily_click_obs
```

Advertising curve



```python
def update_curve(self):
    self.t+=1
    x = np.atleast_2d(self.pulled_bids).T  # Convert pulled bids to a column vector
    y = self.collected  # Collected click cost observations
    if self.t %20==0:
        self.gp.fit(x, y)  # Fit Gaussian Process for click cost

    # Predict mean and standard deviation of click cost
    self.means, self.sigmas = self.gp.predict(np.atleast_2d(self.bids).T, return_std=True)
    self.sigmas= np.maximum(self.sigmas, 1e-2)  # Ensure minimum value for standard deviation

def update_observations(self, pulled_bid, obs):
    self.pulled_bids.append(self.bids[pulled_bid])
    self.click_bid[pulled_bid].append(obs)  # Add click cost observation for the pulled bid
    self.collected = np.append(self.collected, obs)  # Add click cost observation to collected array
```

2 Learning for advertising

GPTS Learner

- The TS algorithm for this case, as said before, works calculating the Gaussian distribution instead of the Beta distribution. The rest of the algorithm remains the same:

GPUCB1 Learner

- Works as the previous UCB1 algorithm, but the upper bounds are calculated also with Gaussian distributions:

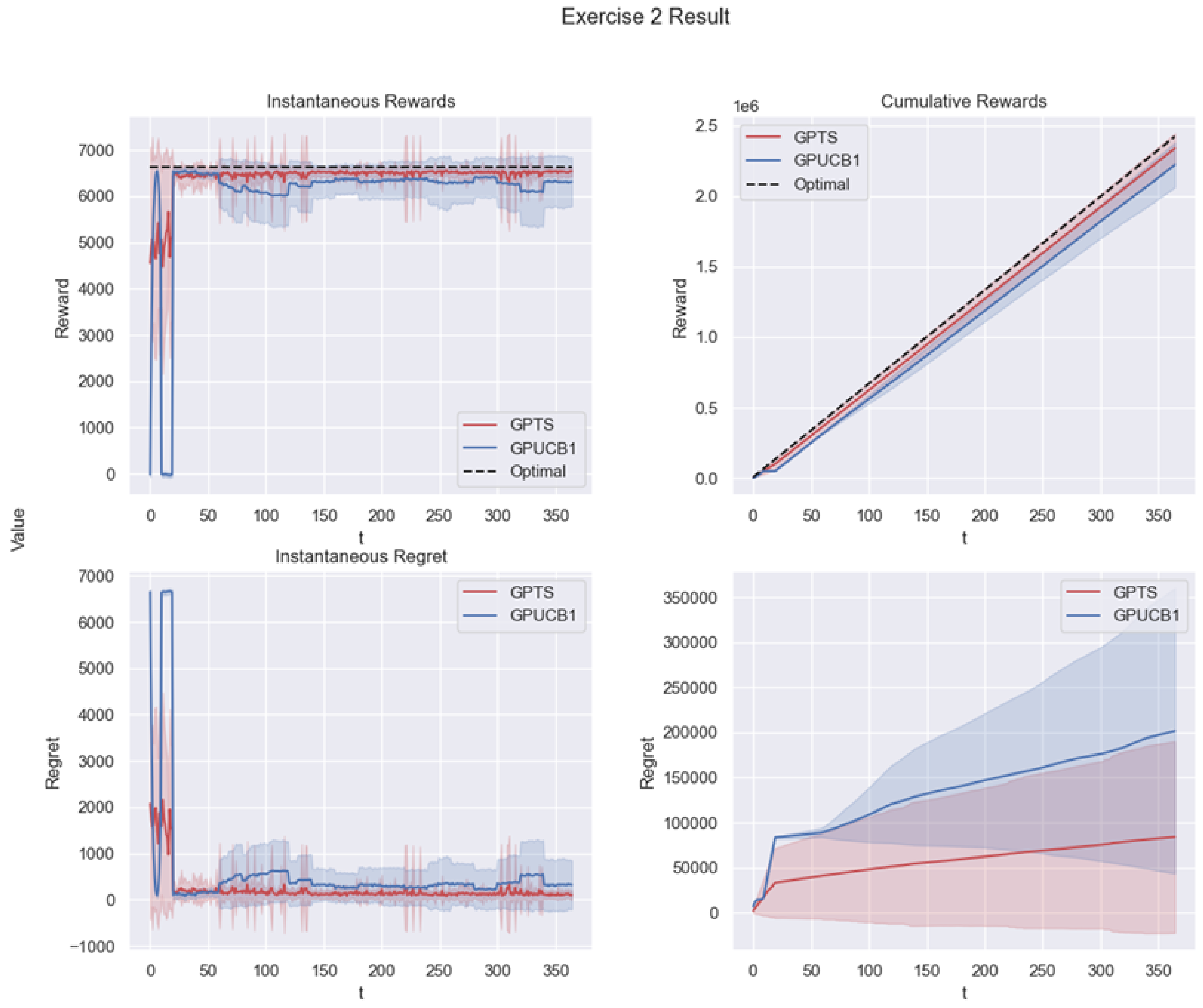
The outcome of both algorithms are expected reward based on the sampled click cost and daily clicks. This expected reward is compare with each bid and selected the maximum value.

$$Reward = NC(bids) \cdot CP(price) \cdot (price - cost) \cdot CC(bids)$$

Learning for advertising

RESULTS

- n_exp = 50
- T = 365



RESULTS

- As we can see in the Instantaneous Reward and Instantaneous regret graph, the TS algorithm achieves optimal solution.
- In the other hand, the UCB1 has a lower perform. In some round intervals the values decrease, making a “step” form.
- At the beginning, the GPUCB1 algorithm gives has a very bad reward values, since the upper confidence is more sensible at the beginning of the experiment.
- The two algorithms converge, but the TS goes faster.

3 Learning for joint pricing and advertising

INITIAL INFORMATION

- We are only going to use class C1.
- The curves related to the advertising and pricing part are NOT known.

GOAL

- With this information, the goal of the first step is to apply the previous algorithms to find the advertising curves.

- Also, plot all the results in order to show: cumulative regret, cumulative reward, instantaneous regret, instantaneous reward.

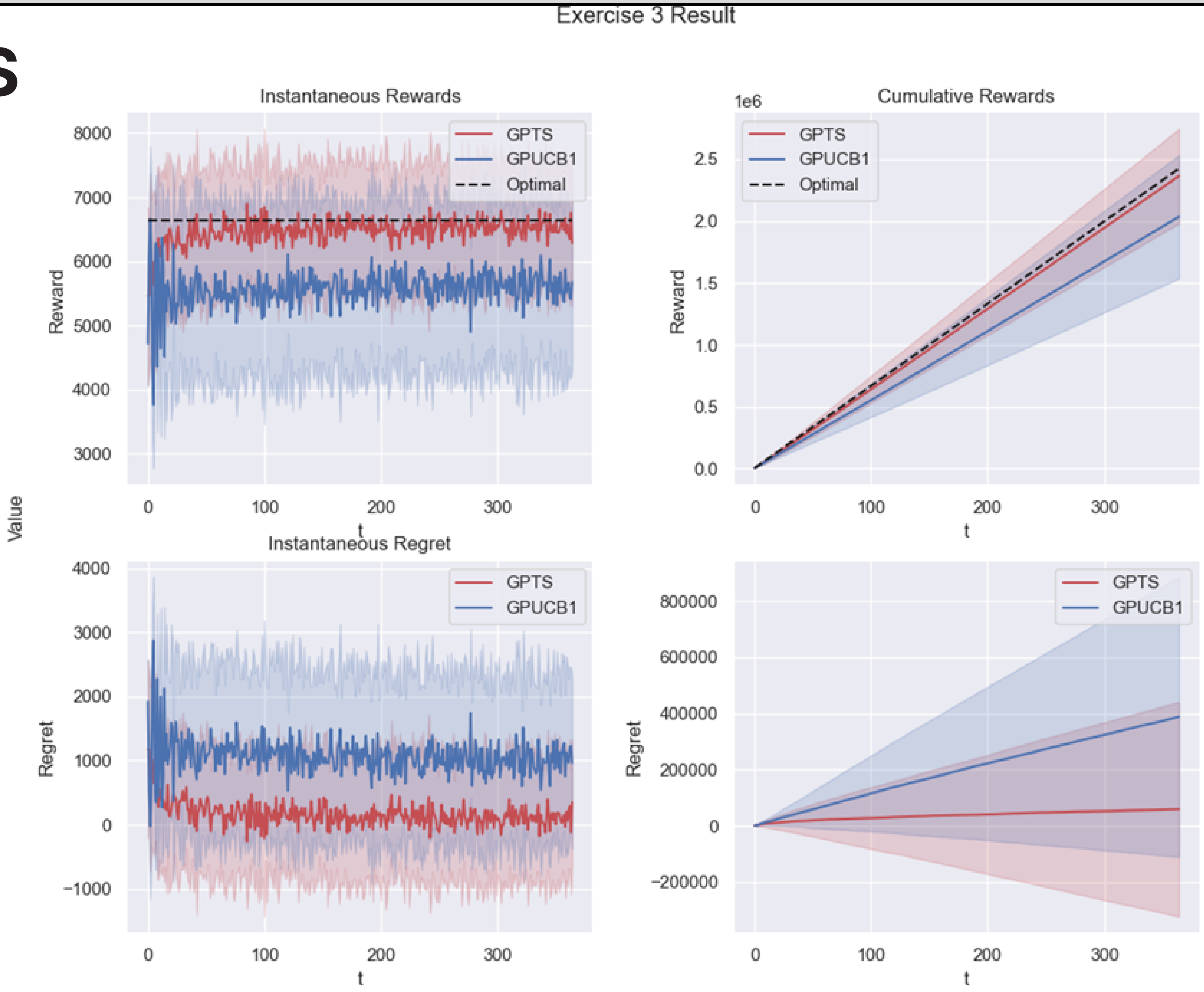
PROCEDURE

- As the project proposal suggest us, we are going to apply the following procedure:
    - First, we are going to optimize the pricing problem to find the best price.
    - Given this price, we optimize the advertising problem to find the suggested bid.
- The procedure of the single pricing and advertising problems are the same as the previous steps.

Learning for joint pricing and advertising

RESULTS

- n_exp = 50
- T = 365



3 Learning for joint pricing and advertising

RESULTS

- As we can see in the Instantaneous Reward and Instantaneous regret graph, the TS algorithm achieves optimal solution.
- In the other hand, the UCB1 has a lower perform.
- Like the other steps, this result was expected. But in this case, now we can distinguish between the results of the two algorithms. This is because in this case the difference between the rewards of the arms is more competitive.
- The two algorithms converge, but the TS goes faster and with a better result.

4 Contexts and their generation

INITIAL INFORMATION

- We consider the THREE classes: C1, C2 and C3.
- The curves related to the advertising and pricing are not known.

GOAL

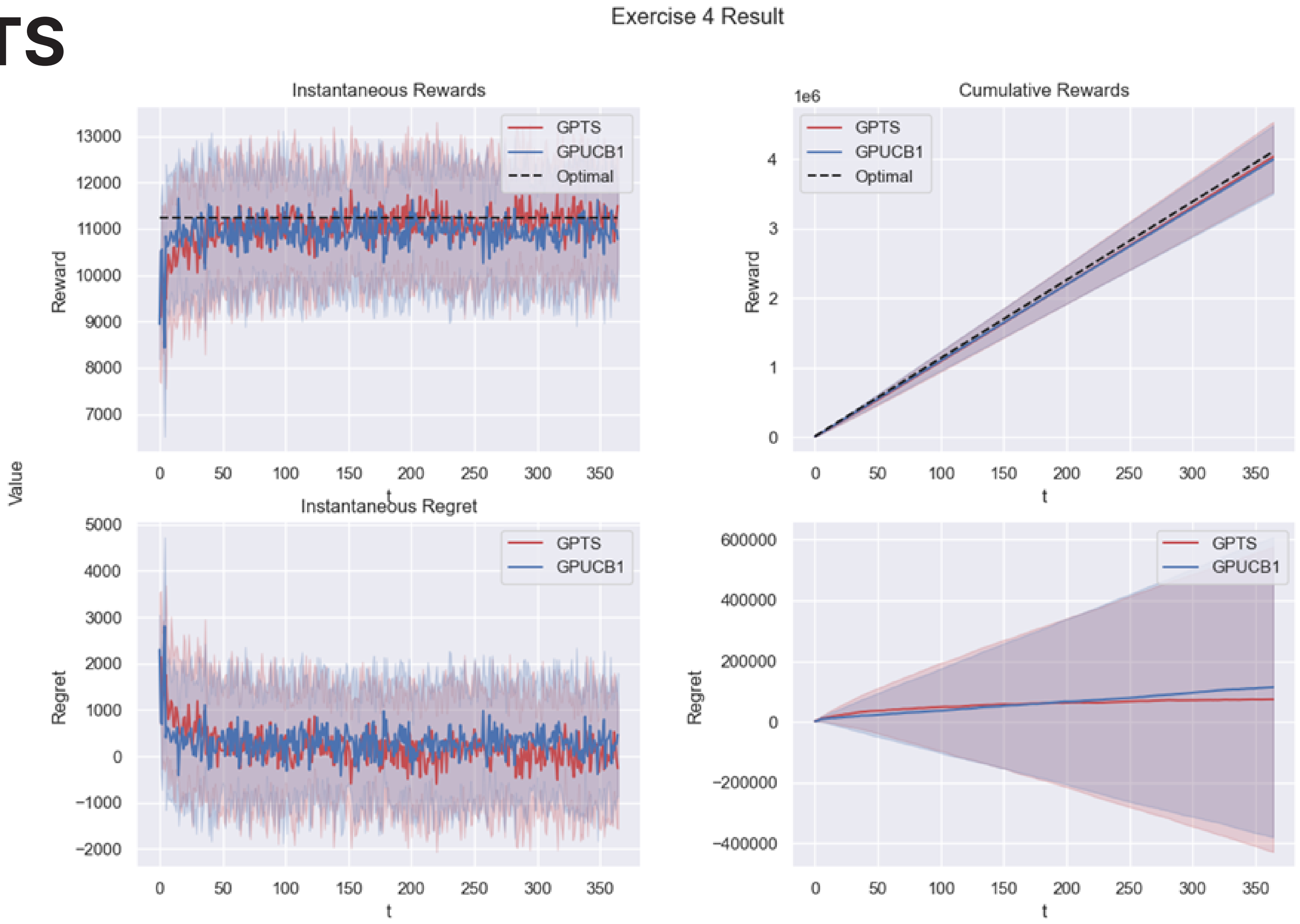
- With this information, the goal of the first step is to apply the previous algorithms to estimate the conversion probabilities and find the best price.

- Also, plot all the results in order to show: cumulative regret, cumulative reward, instantaneous regret, instantaneous reward.

4

Contexts and their generation

RESULTS



5 Non-stationary environments with two abrupt changes

INITIAL INFORMATION

- We are only going to use class C1.
- The curves related to the advertising problem are known.
- The curves related to the pricing problem are unknown, and they are NON-STARIONARY with three different phases.

GOAL

- With this information, the goal of the first step is to apply UCB1 and TS algorithm to estimate the conversion probabilities and find the best price.

- Also, plot all the results in order to show: cumulative regret, cumulative reward, instantaneous regret, instantaneous reward.

5 Non-stationary environments with two abrupt changes

UCB1 Algorithm

The UCB1 algorithm is the same as the one used in step1, for this reason we skip the description of the algorithm in this section. Please refer to the section 1 for more information.

SW UCB1 Algorithm

The "Sliding-Window UCB1" (SW UCB1) method is a version of the original UCB1 algorithm designed to handle non-stationary situations with changing rewards of arms. It is widely employed in multi-armed bandit issues where the reward distributions of the arms may vary and the algorithm must adapt to these changes. The main idea underlying SW UCB1 is to maintain track of recent awards using a sliding time frame and update the UCB estimations depending on the rewards inside that window. This sliding window method enables the algorithm to focus on recent data while discarding earlier, perhaps obsolete data. Utilizing a sliding window of recent data enables it to avoid reliance on outdated information, resulting in improved performance in non-stationary environments. Nonetheless, choosing the optimal window size W is crucial as it strikes a balance between adaptability and the sufficiency of data required for dependable estimates. For this reason in the further section we will present a sensitivity test on the parameter W.

5

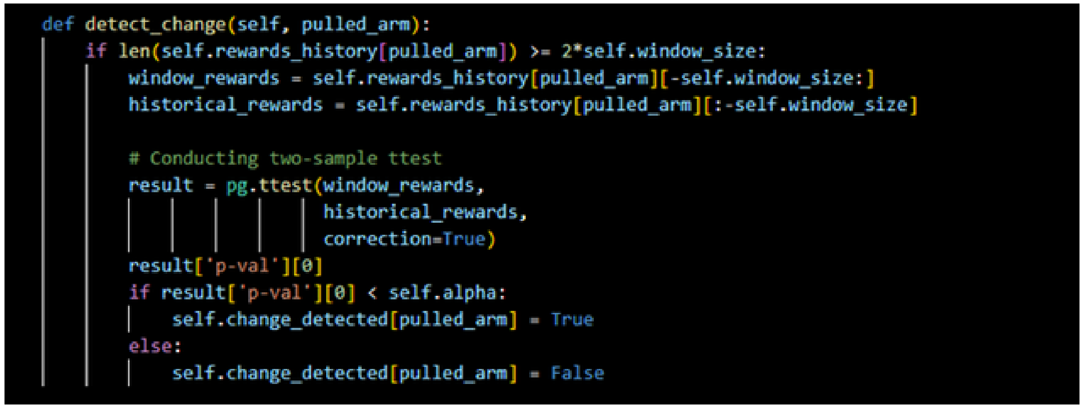
Non-stationary environments with two abrupt changes

CD UCB1 Algorithm

The change detector UCB1 algorithm is a version of the simplest UCB1 algorithm, where a change detector test is applied to verify if the means of the rewards are significantly different in the recent output compared to the historical one.

In our case, we are interested in identifying changes in the underlying reward distribution of the bandit arms while utilizing the t-Student test as a change detector in the UCB1 algorithm. The t-Student test is a statistical hypothesis test used to compare the means of two samples and determine if they vary substantially. We will implement this test on a the most recent portion of the reward vs the historical one.

For this reason, the CD UCB1 algorithm has 2 inputs, the length of the window W and the parameter alpha, i.e., the parameter of the t student test. Please find below the part of the code in which the test is applied.



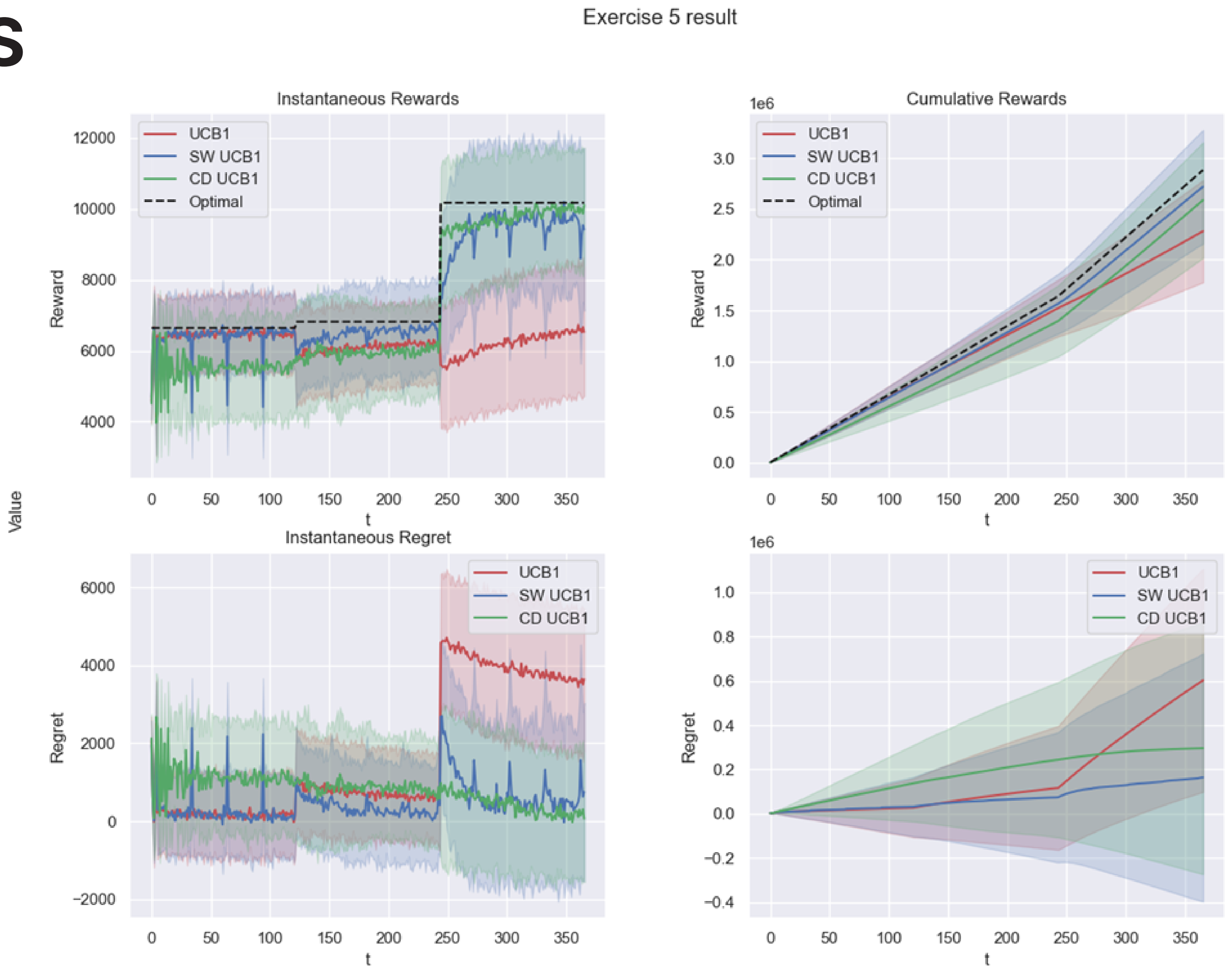
```
def detect_change(self, pulled_arm):
    if len(self.rewards_history[pulled_arm]) >= 2*self.window_size:
        window_rewards = self.rewards_history[pulled_arm][-self.window_size:]
        historical_rewards = self.rewards_history[pulled_arm][:-self.window_size]

        # Conducting two-sample ttest
        result = pg.ttest(window_rewards,
                          historical_rewards,
                          correction=True)
        result['p-val'][0]
        if result['p-val'][0] < self.alpha:
            self.change_detected[pulled_arm] = True
        else:
            self.change_detected[pulled_arm] = False
```

5 Non-stationary environments with two abrupt changes

RESULTS

- n_exp = 200
- T = 365

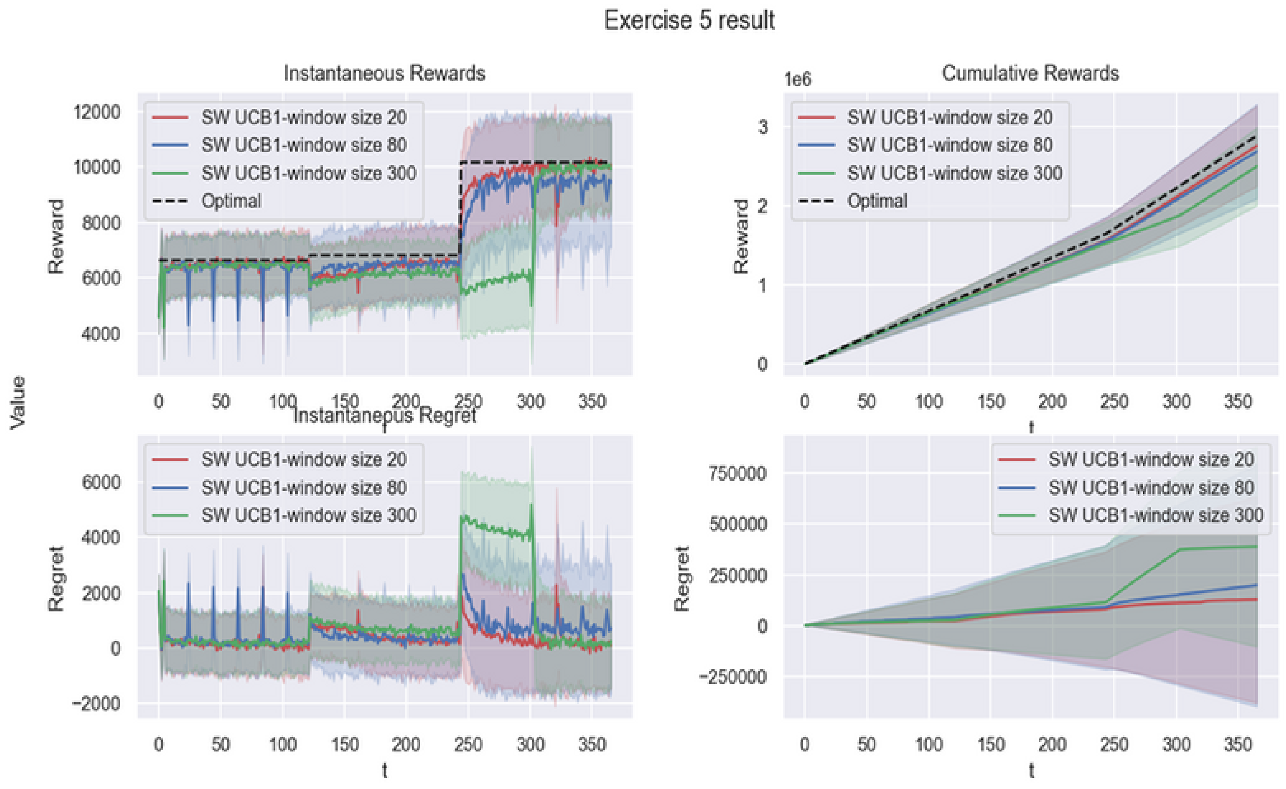


5 Non-stationary environments with two abrupt changes

Sensivity test

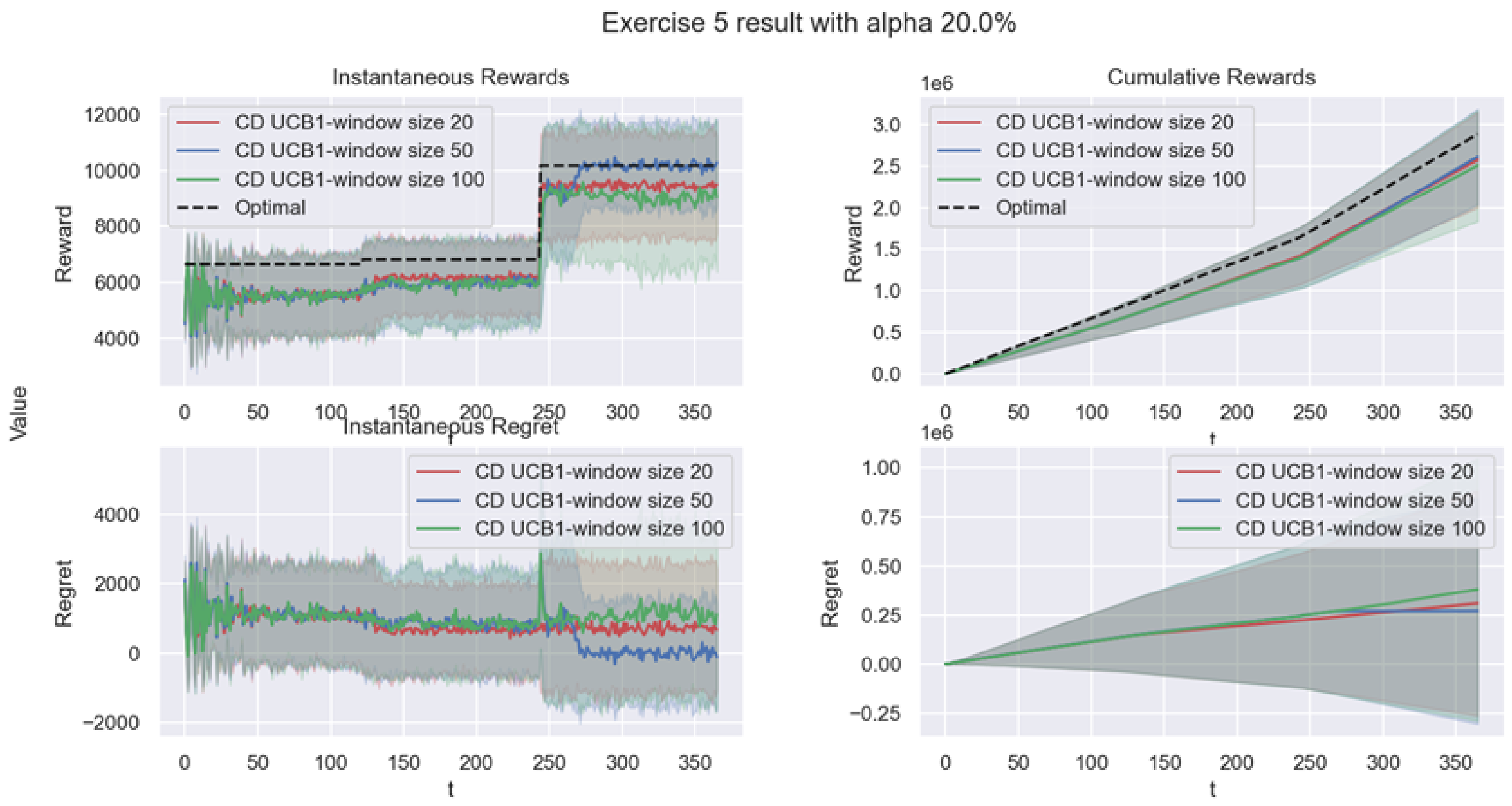
SW UCB1 considers W as the size of the sliding windows, whereas CD UCB1 contains two parameters, the test parameter alpha and the window size W.

We tried the SW UCB1 algorithm with three distinct W parameters: 20, 80, and 300. The approach with W=20 performs better than the one with W=300, as predicted, because the algorithm is too slow to detect changes in the ideal reward curve.



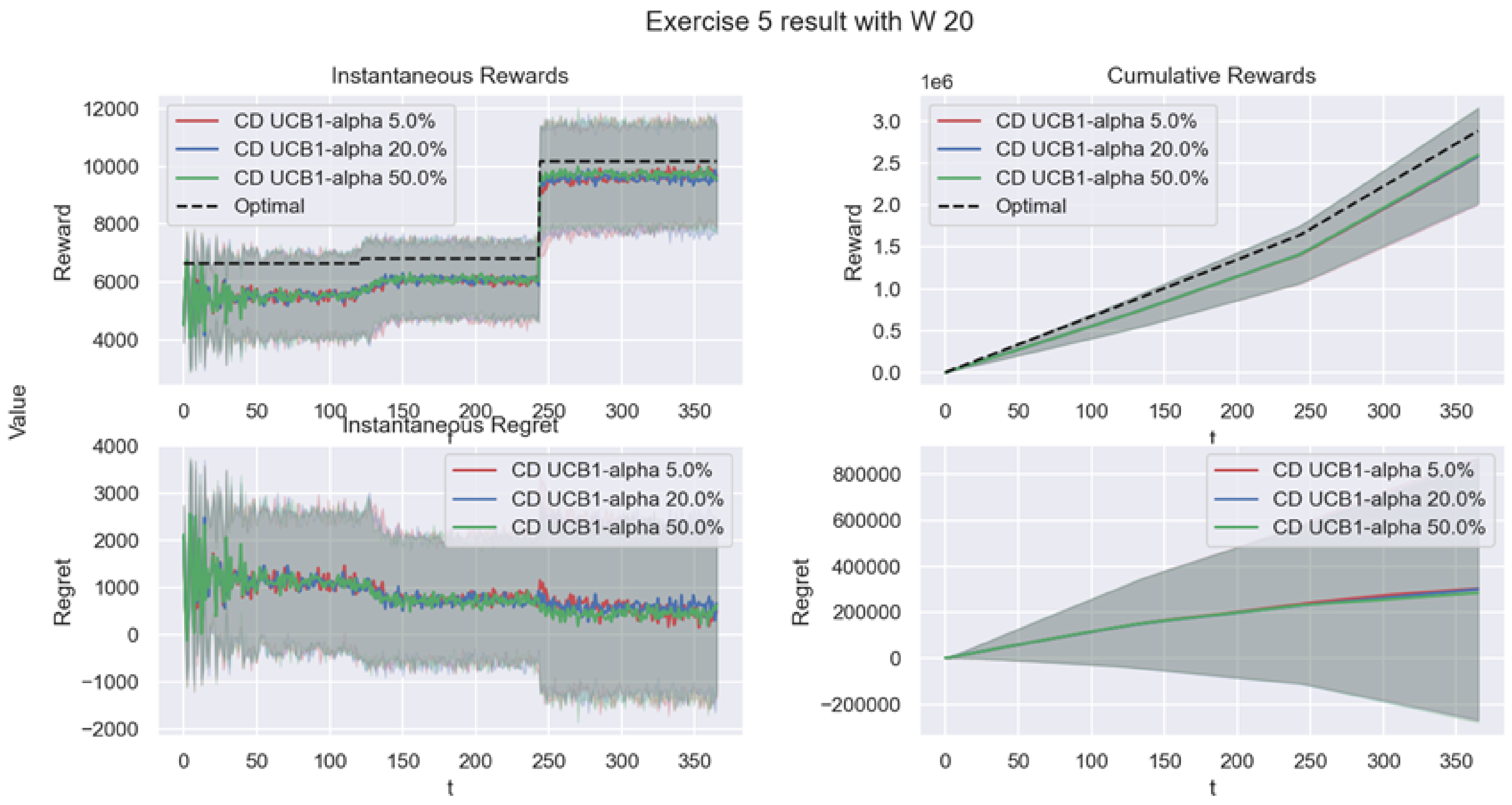
Sensivity test

In the same way, we simulated the CD UCB1 method with three alternative options for W, 20, 50, and 100, while leaving the alpha parameter fixed at 20%. Even in this scenario, the algorithm with W=20 and 50 performs best, whereas the algorithm with W=100 performs worst since the system is too slow to detect changes in the ideal reward curve.



Sensivity test

IFinally, we repeated the technique with the value W set to 20 and the parameters alpha set to 0,05, 0,2, and 0.5. The greater the alpha value, the more likely it is that a change is identified even if there is no change in the mean of the reward.



6 Non-stationary environments with many abrupt changes

INITIAL INFORMATION

- We are only going to use class C1.
- The curves related to the advertising problem are known.
- The curves related to the pricing problem are unknown, and they are NON-STARIONARY witht MANY different phases.

GOAL

- With this information, the goal of the first step is to apply UCB1 and TS algorithm to estimate the conversion probabilities and find the best price.

- Also, plot all the results in order to show: cumulative regret, cumulative reward, instantaneous regret, instantaneous reward.

6 Non-stationary environments with many abrupt changes

EXP3 is an algorithm tailored for the adversarial bandit setting. In each round, it chooses an arm based on a random draw from a probability distribution calculated in the previous step. EXP3's adversarial nature introduces learning tendencies. When an arm yields a high reward, its weight increases, leading to a higher probability of being selected in subsequent rounds.

The probability of selecting arm i at round t is governed by the formula:

$$pi(t) = (1 - \gamma) * \frac{w_i(t)}{\Sigma w_j(t)} + \frac{\gamma}{K}.$$

Here, K represents the number of arms, and w_i(t) denotes the weight associated with arm i at round t. The hyperparameter γ, ranging between 0 and 1, influences the probability distribution.
To update the weight of arm i at round t, the algorithm employs the formula:

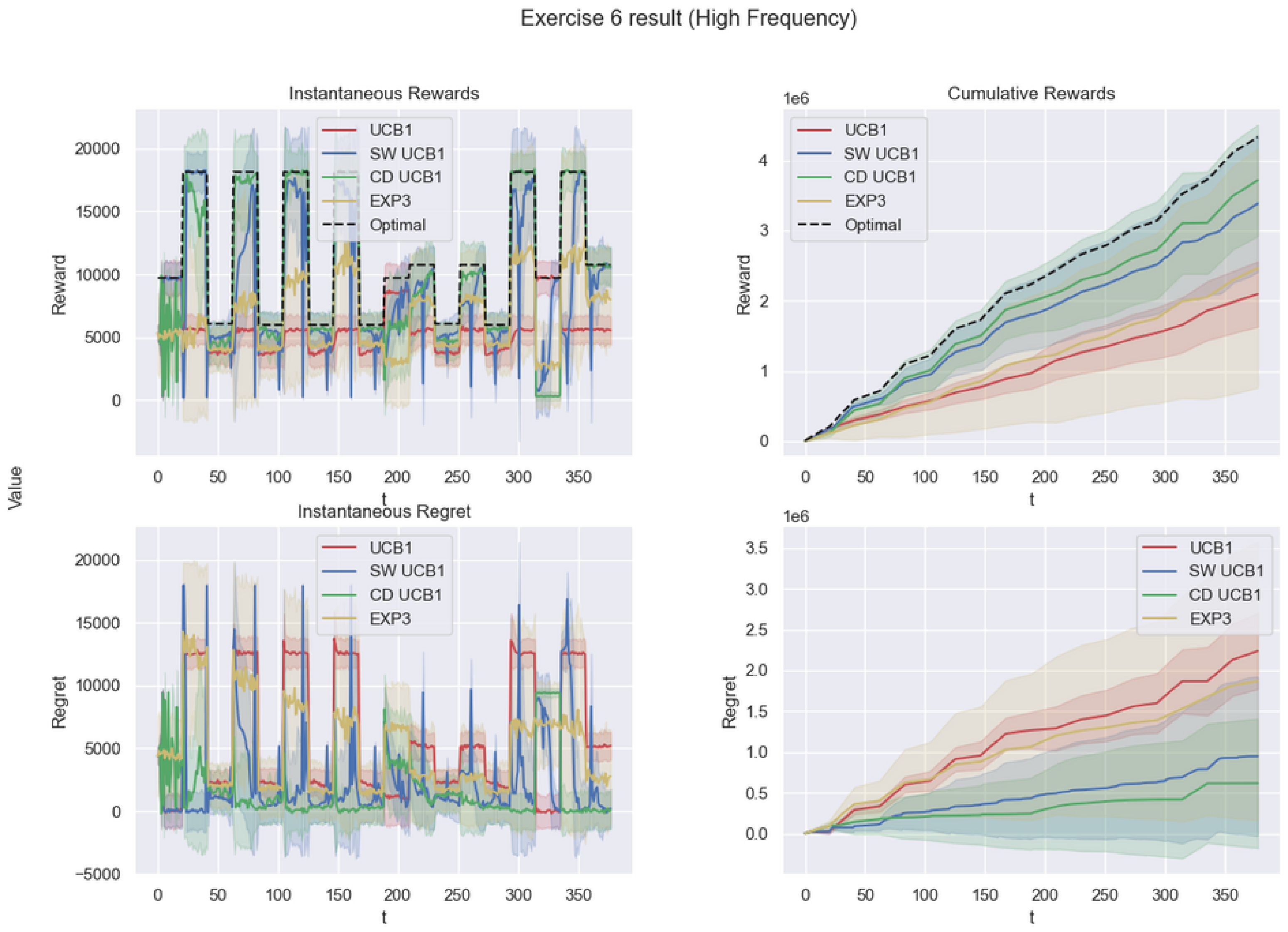
$$w_i(t+1) = w_i(t) * e^{\gamma/K * \frac{x_{it}}{p_{it}}},$$

where xit / pit represents the expected reward obtained from arm i at time t. It is essential to initialize the weights to 1 for each arm, and the rewards should be rescaled to fall within the interval [0, 1]. The value of γ dictates the balance between a more uniform probability distribution (closer to 1) and a distribution based on rewards obtained (closer to 0).

6 Non-stationary environments with many abrupt changes

RESULTS

- n_exp = 200
- T = 365



6 Non-stationary environments with many abrupt changes

RESULTS

- We examine EXP3's performance under the Step 5 scenario, where there are two transitions, each lasting roughly 120 rounds, resulting in three phases. The EXP3 method, as expected, does not perform well in a context with only two modifications over 365 cycles.

- If we consider instead in an high frequency scenario the model performs quite well, in our scenario we are considering a high number of changes in the pricing curve with the same length curve of 365 rounds. Unfortunately, we were expecting that algorithm was able to outperform the two precedent algorithms in an high frequency scenario, but as showed in the image below this was not the case in our implementation. For a lack of time we were not able to solve the problem.

Thank you