Online Learning Application Project

II Semester 2022

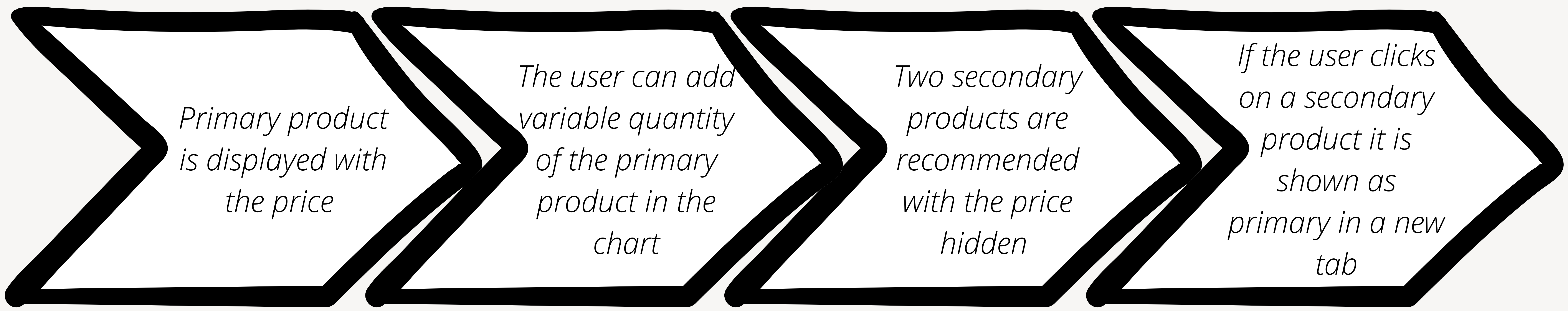
Group 11

Corbella Andrea - *10529703*
Martins Cesar - *10580039*
Siracusa Giovanni - *10530452*
Vivaldi Paolo - *10623984*
Pistone Santi Pier - *10867402*

Scenario

The scenario is an ecommerce that sells forniture and household products

These are the main steps of the process in the ecommerce:


Primary product is displayed with the price
The user can add variable quantity of the primary product in the chart
Two secondary products are recommended with the price hidden
If the user clicks on a secondary product it is shown as primary in a new tab

This is an iterative process

Scenario - Customer Classes

People < 30 years

Males > 30 years

Females > 30 years

We have clustered males and females < 30 years together due to the higher similarities in the purchasing behaviour and due to the limited purchasing capacity allocated for the household products

Scenario - Products

We have selected these 5 products as representative sample and every product has different conversion rates based on the preferences of all the customer classes

Alexa	
People < 30	▬
Male > 30	▬
Female >30	⬇

Wine cellar	
People < 30	⬇
Male > 30	⬆
Female >30	▬

Painting	
People < 30	▬
Male > 30	▬
Female >30	▬

Plant	
People < 30	▬
Male > 30	▬
Female >30	⬆

Pouf	
People < 30	⬆
Male > 30	⬇
Female >30	▬

Legenda

⬆	*Higher CR*
⬇	*Lower CR*
▬	*Average CR*

Scenario - Config

The configuration files include all the characteristics of the environment and the possible behaviours of the users. In these files is possible to find:

- The **conversion rate (CR) matrixes** that define the different conversion rates based on the customer classes and on the price of product displayed

- The **prices matrix**, where we defined the 4 possible prices for each product,and the **cost array**, where we defined the ecommerce acquisition cost for every product, and the **margin**, that is computed by the price displayed to the user minus the acquisition cost

Scenario - Config

- **α** is an array that defines the probability to land in a specific product page, **α_0** defines the possibility to not enter in the ecommerce

- **Daily users** are the upper and lower bound that we defined for the range of users that get access to the ecommerce for each customer class

- **Sold items arrays** define what's the average quantity of each products sold for every customer class

- **Graph probs matrixes** define the probability to select a specific secondary product starting from a specific primary one and **lambda matrix** sets the first and second secondary product displayed

Step 1

Goal: Creation of the environment

Input :

- Configuration Files:
  - Margin
  - Conversion rates
  - Upper and lower bound of daily users
  - Alpha
  - Number of items
  - Graph probabilities

Output:

- Unitary reward for every product
- How many people bought every product
- How many people visited the webpage of each single product
- Estimated Alphas
- Average number of products purchased by each client
- Sequence of webpages visited by the customer

Step 1 - Simulation

- There's a random number belonging to the range of the **daily users** for each customer class of users accessing the site
- For each user it uses the **alpha probabilities** to select the landing page
- Then with the **conversion rates** it checks if the user buys the primary product
- The user will buy 1 + sample from Poisson distribution with lambda equal to the **number of items sold**
- Checks if the user clicks on the secondary products using **graph probabilities** and knowing that it's not possible to visit the same page again
- This process is replicated on the new product
- Save the sequence of pages visited by that user

Step 1 - Bruteforce

- In order to computer the optimal price configuration we use a **bruteforce approach**
- It's feasible since there are not much price configurations $4^5 = 1024$

1.Compute the **theoretical reward** for each price configuration ***C***

$$theoretical\ reward = \sum_{i \in C} CR(p,i) * margin(p,i) * \alpha(p)$$

2.Select the configuration with highest theoretical reward

3.Compute the **actual reward** as the mean of 100 simulations with that configuration

Step 2

Goal: Select the best configuration with a greedy approach

Input :

- Configuration Files:
  - Margin
  - Conversion rates
  - Alpha

Output:

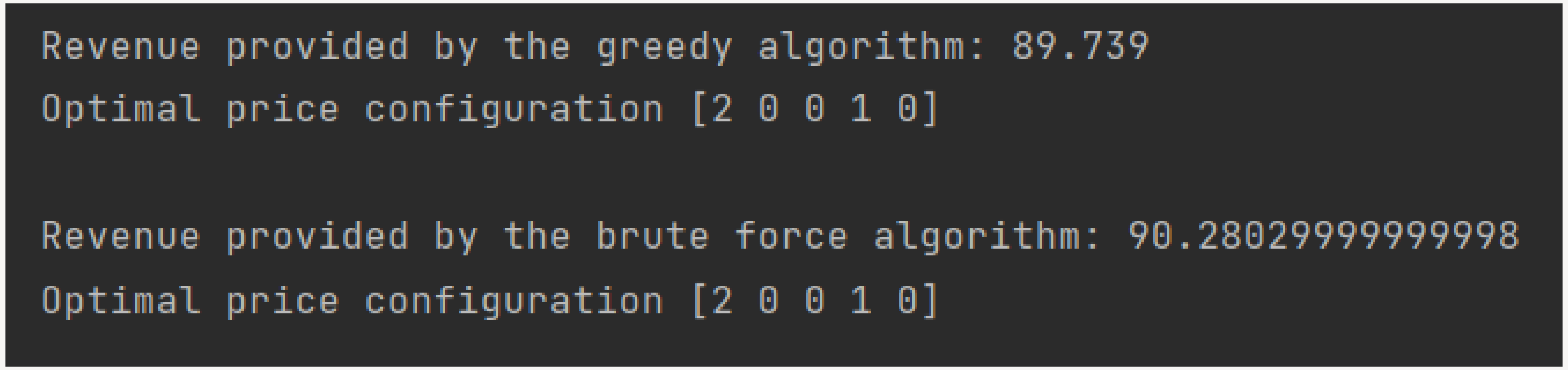
- Price configuration

Step 2 - Greedy Learner

Algorithm:

- Start with all the prices equal to zero
- Iteratively try to increase all product's prices one at a time
- Choose the price configuration that maximizes the reward between the five configurations computed
- When the reward stops to increase, the **theoretical best configuration** is found

Step 2 - Greedy Learner



```
Revenue provided by the greedy algorithm: 89.739
Optimal price configuration [2 0 0 1 0]

Revenue provided by the brute force algorithm: 90.28029999999998
Optimal price configuration [2 0 0 1 0]
```

Greedy algorithm and bruteforce algorithm return the **same price configuration**; this is due to the parameters set on configuration file.
Despite the same price configuration, the **revenues are different** as a result of randomness of the simulation.
With a different configuration file the Greedy algorithm could return a different configuration because the iteration stops when it finds a **relative maximum**.

Step 3

Goal: Optimization with unknown conversion rates

Input :

- Configuration Files:
  - Margin
  - Alpha
  - Number of items
- Simulation output:
  - Buyers
  - Offers

Output:

- Estimated Conversion rates
- Price configuration

Step 3 - TS

- Every product is associated with a **TS learner**
- Every arm is associated to a **specified price**
- **Beta parameters:** they define a distribution which mean is the estimated conversion rate

1.Every day, for each arm draw a sample ***b*** from the beta distribution

2.Select the arm that **maximizes** this formula

$$b * margin * alpha * n^{\circ}\ of\ items$$

3.Run the simulation with the configuration of prices computed by the learners

4.**Update** the beta parameters of each arm:

$$\alpha_t \rightarrow \alpha_{t-1} + buyers$$
$$\beta_t \rightarrow \beta_{t-1} + offers - buyers$$

Step 3 - UCB

- Every product is associated with a **UCB learner**
- Every arm is associated to a **specified price**
- **Empirical mean:** estimation of conversion rate computed as the weighted average of ratio between buyers and offers
- **Confidence:** standard confidence of the UCB

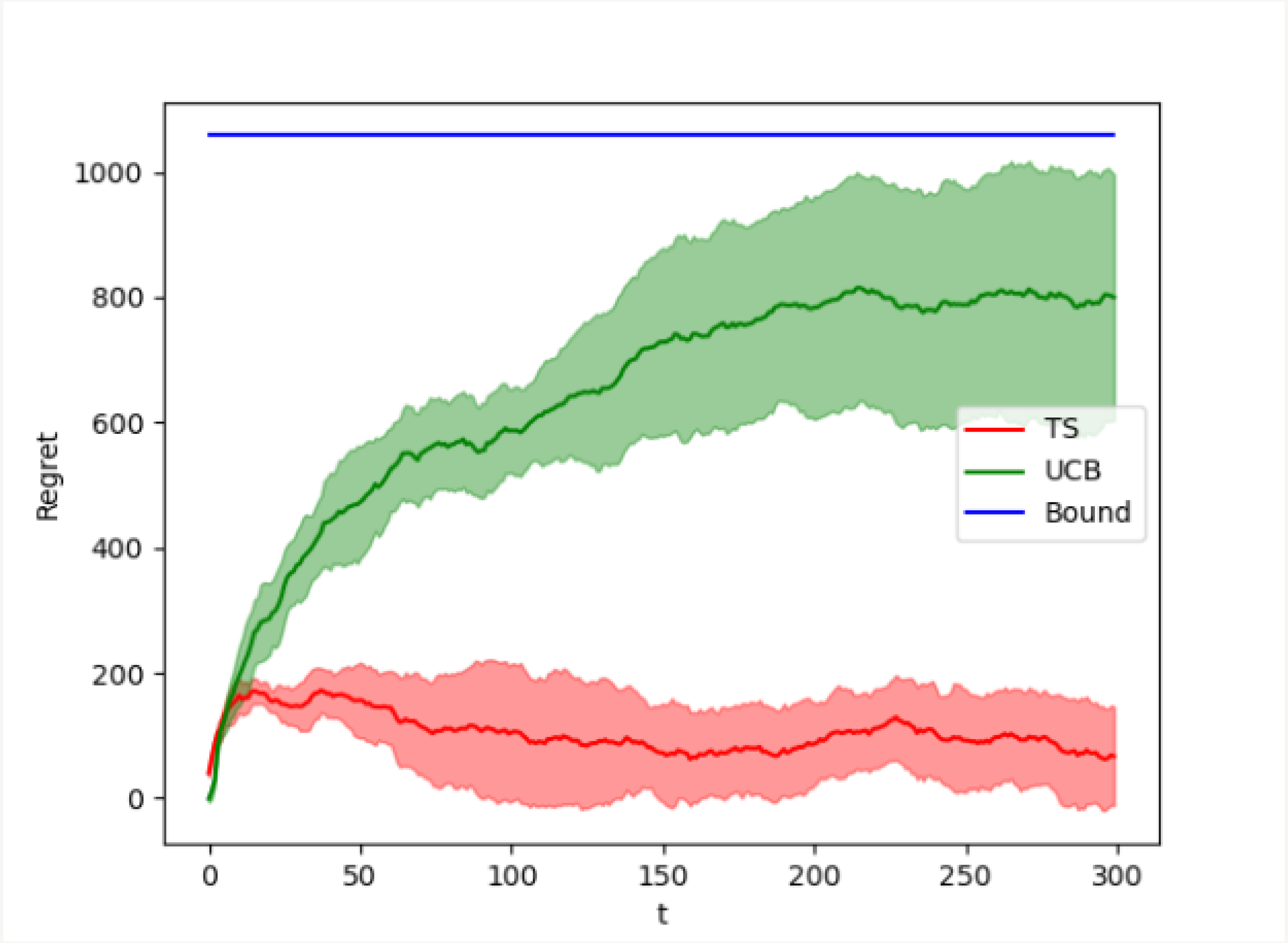
1.Every day, for each arm is computed the upper confidence as $empirical\ mean + confidence$

2.Select the arm that **maximizes** $upper\ confidence * margin * alpha * n°of\ items$

3.Run the simulation with the configuration of prices computed by the learners

4.**Update** the confidence according to the standard formula $confidence = \sqrt{\frac{2 * \log(t)}{\#samples}}$

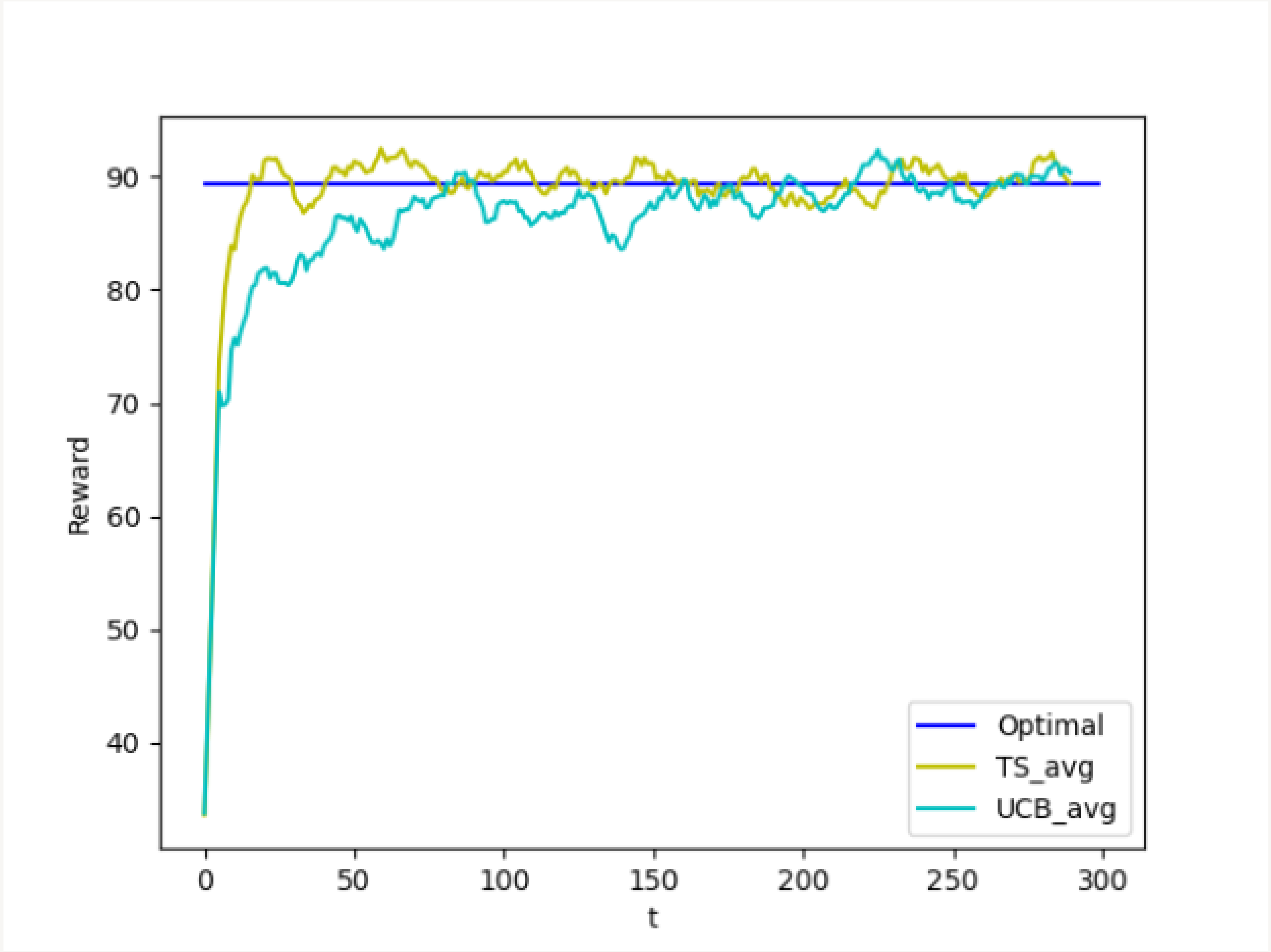
Step 3 - Graphs



The graph shows the cumulative regret of the algorithms:

- 5 experiments
- Time horizon: 300 days
- Both plots are **sublinear**, in particular the UCB has a logarithmic trend
- The regret of UCB is greater than TS
- The shown bound is the theoretical regret bound of the UCB computed over the time horizon

Step 3 - Graphs



The graph shows the daily reward of the algorithms:

- 5 experiments
- Time horizon: 300 days
- Both algorithms converge to optimal but **UCB takes more time**
- Optimal is the reward obtained by the bruteforce algorithm
- To reduce some noise we applied moving average with a window of 10 days

Step 4

Goal: Optimization with unknown conversion rates, alphas and number of sold items

Input :

- Configuration Files:
  - Margin
- Simulation output:
  - Buyers
  - Offers
  - Alphas
  - N° of items sold

Output:

- Estimated conversion rates
- Estimated Alphas
- Estimated Number of sold items
- Price configuration

Step 4 - TS

- Every product is associated with a **TS learner**
- Every arm is associated to a **specified price**
- **Beta parameters:** they define a distribution which mean is the estimated conversion rate

1.Every day, for each arm draw a sample ***b*** from the beta distribution

2.Select the arm that **maximizes** this formula $b * margin * estimated\ alphas * estimated\ n° \ of\ sold\ items$

3.Run the simulation with the configuration of prices computed by the learners

4.**Update** the beta parameters of each arm:
$$\alpha_t \rightarrow \alpha_{t-1} + buyers$$
$$\beta_t \rightarrow \beta_{t-1} + offers - buyers$$

5.Update the **estimated alphas** and **n° of sold items**

Step 4 - UCB

- Every product is associated with a **UCB learner**
- Every arm is associated to a **specified price**
- **Empirical mean:** estimation of conversion rate computed as the weighted average of ratio between buyers and offers
- **Confidence:** standard confidence of the UCB

1.Every day, for each arm compute the upper confidence as $empirical\ mean + confidence$

2.Select the arm that **maximizes**

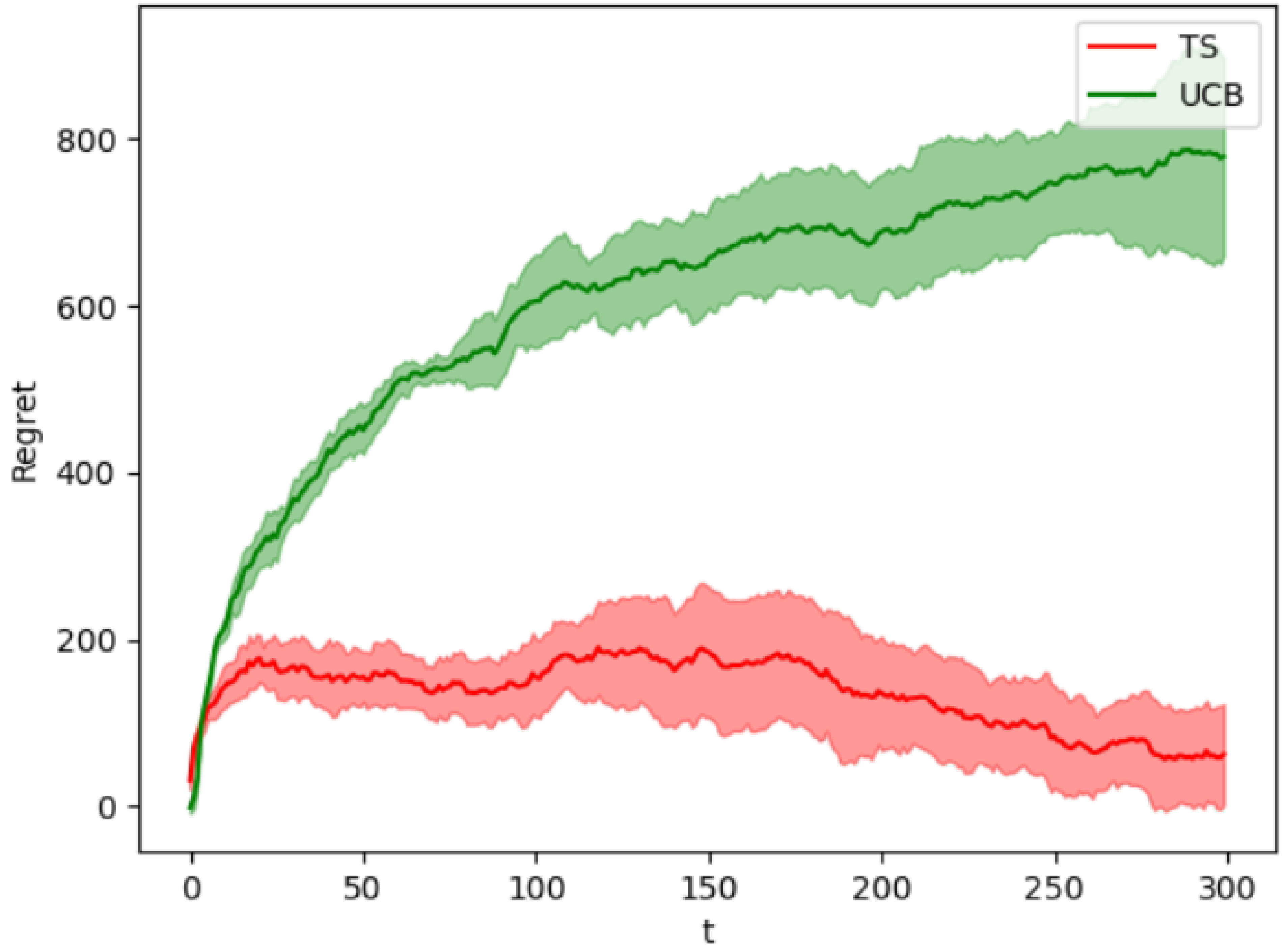
$$upper\ confidence * margin * estimated\ alphas * estimated\ n° \ of\ sold\ items$$

3.Run the simulation with the configuration of prices computed by the learners

4.**Update** the confidence according to the standard formula

5.Update the **estimated alphas** and **n° of sold items**

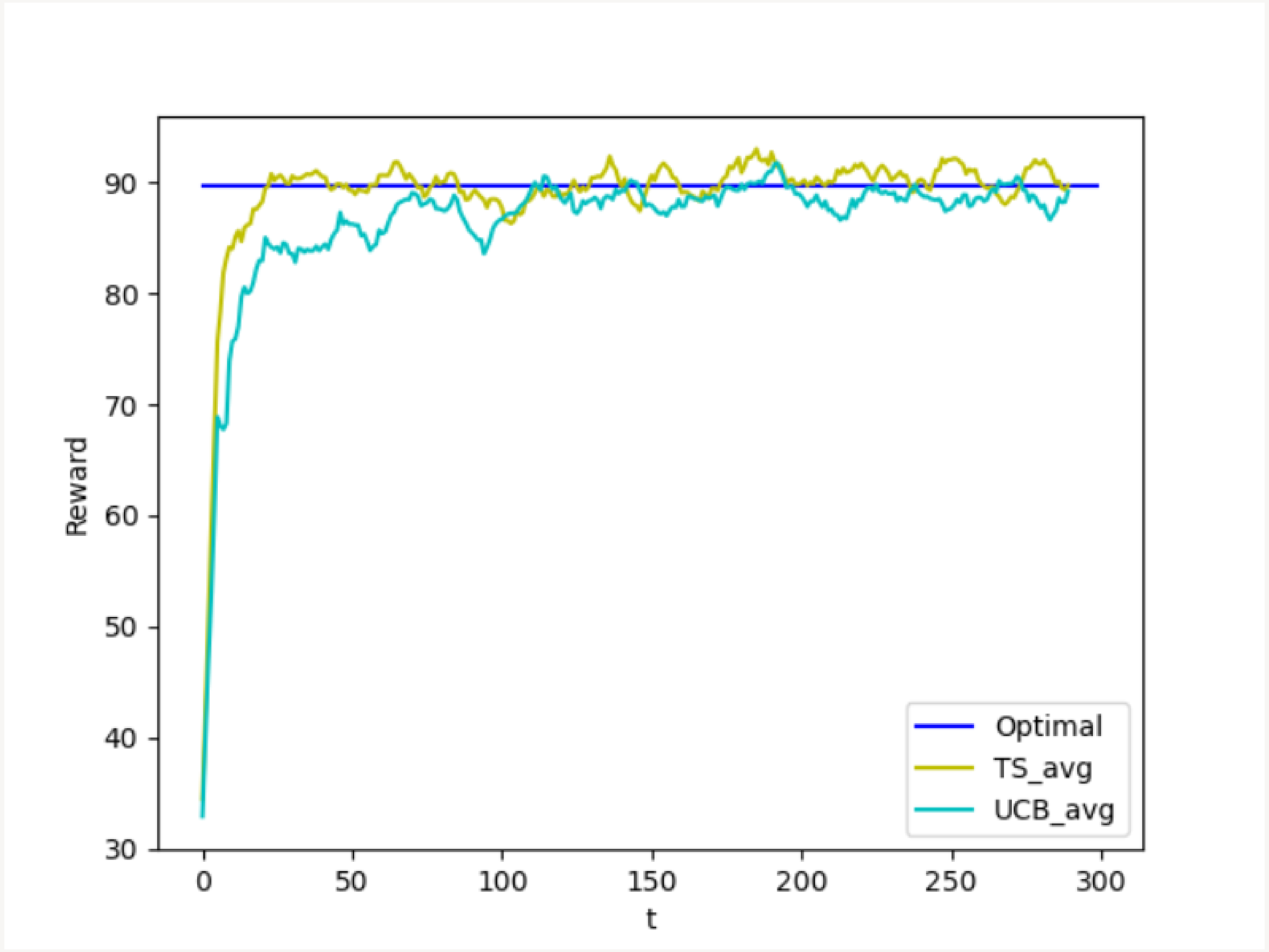
Step 4 - Graphs



The graph shows the cumulative regret of the algorithms:

- 5 experiments
- Time horizon: 300 days
- It behaves exactly as step 3 because **alphas and the number of sold items do not depend on the price configuration**

Step 4 - Graphs



The graph shows the daily rewards of the algorithms:

- 5 experiments
- Time horizon: 300 days
- As expected the reward graph behaves exactly as step 3 because **alphas and n° of sold items do not depend on the price configuration**

Step 5

Goal: Optimization with unknown graph weights

Input :

- Configuration Files:
  - Margin
- Simulation output:
  - Buyers
  - Offers
  - Sampled graph probabilities through Montecarlo Sampling

Output:

- Estimated graph weights
- Price configuration

Step 5 - Montecarlo Sampling

Montecarlo Sampling is used to estimate the graph probabilities

At the end of each day for each product it computes the probability that a user clicks on a secondary product ***b*** when visiting the page of product ***a***

$$graph(a,b) = \frac{credits(a,b)}{active(a)}$$

$$credits(a,b): \#times\ a\ user\ visited\ page\ \boldsymbol{b}\ from\ page\ \boldsymbol{a}$$

$$active: \#times\ a\ user\ visited\ page\ \boldsymbol{a}$$

Step 5 - Learner

- Every product ***p*** is associated with a learner
- Every arm is associated to a specified price

1.Every day select the arm that **maximizes** this formula:

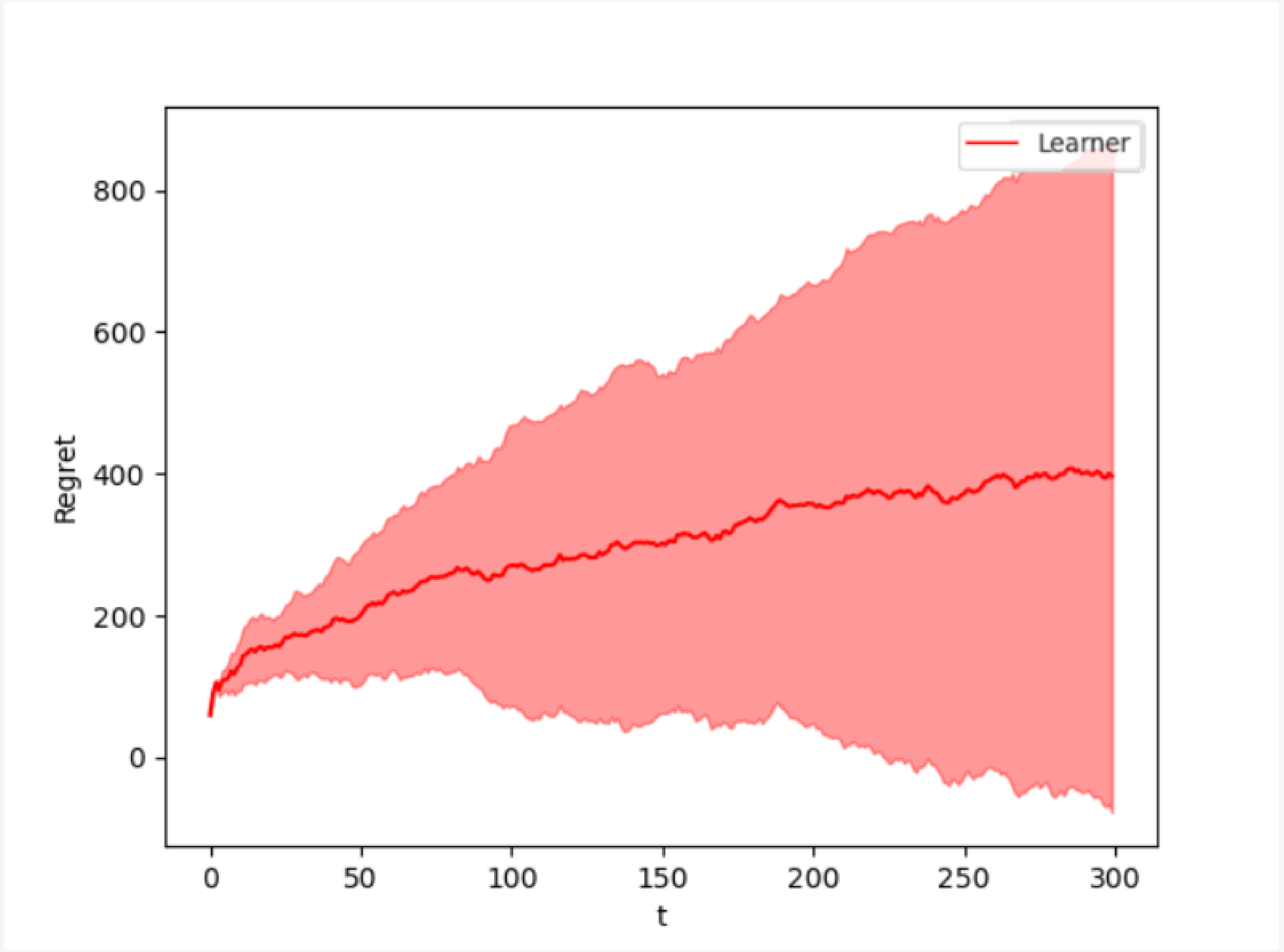
$$margin * \sum_{i} g_{p,i}\,, \qquad \forall i \neq p\,,$$

$$where\ g_{p,i}\ is\ the\ probability\ to\ pass\ from\ product\ p\ to\ product\ i$$

2.Run the simulation with the configuration of prices computed by the **learners**

3.**Update** the estimated graph probabilities

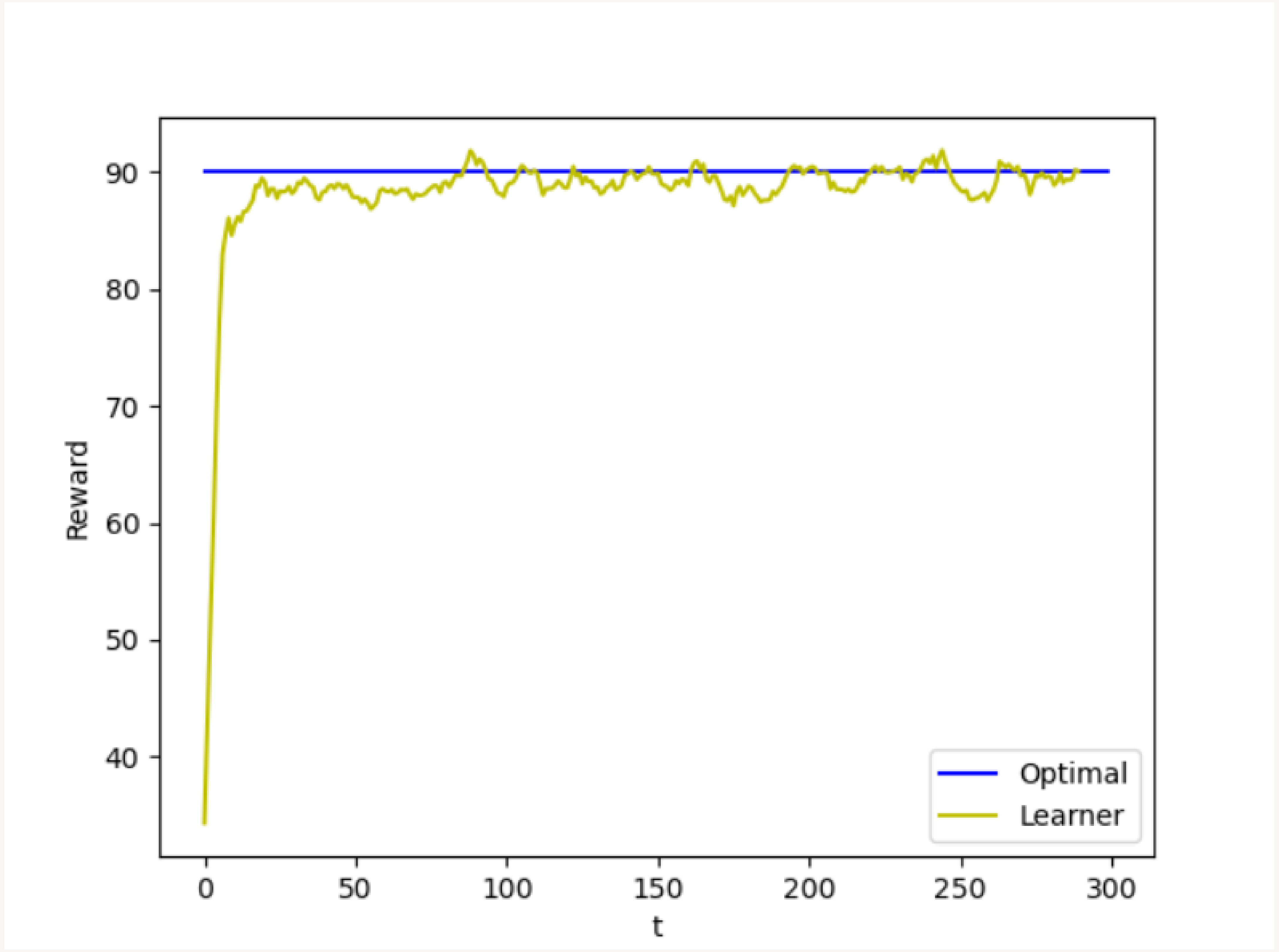
Step 5 - Graphs



The graph shows the cumulative regret of the algorithms:

- 5 experiments
- Time horizon: 300 days
- The graph has a **sublinear shape**
- There's a large confidence interval because the **estimated probability** derives from the simulation

Step 5 - Graphs



The graph shows the daily reward of the algorithm:

- 5 experiments
- Time horizon: 300 days
- The graph **converges to optimal reward**

Step 6

Goal: Optimization with non-stationary demand curves

Input :

- Configuration Files:
  - Margin
  - Alphas
  - N° of items
- Simulation output:
  - Buyers
  - Offers

Output:

- Estimated conversion rates
- Price configuration

Step 6

- The **demand curve** is subjected to changes
- There are three *phases* :
    - **Phase 1**: from day 1 to day 150, the used configuration file is the same of previous steps
    - **Phase 2**: from day 151 to day 300, the conversion rates drop by 30% due to a general increase of prices charged by the ecommerce
    - **Phase 3**: from day 301 to day 450, some products costs increase, specifically the raw materials and so the acquisition cost of the painting, the wine cellar and the pouf increase, but the prices return to the prices of phase 1
- The optimal is computed for each phase, in order to get coherent plots

Step 6 - SW UCB

- Every product is associated with a **sliding window UCB learner**
- Every arm is associated to a **specified price**
- **Window size:** number of samples in the window
- **Empirical mean**: estimation of conversion rate computed as the weighted average of ratio between buyers and offers
- **Confidence**: standard confidence of the UCB
- It works like UCB but uses only the samples in the window

Step 6 - CD UCB

- Every product is associated with a **change detection learner**
- Every arm is associated to a **specified price**
- The changes are detected using the **cumulative sum** test
- **Empirical mean**: estimation of conversion rate computed as the weighted average of ratio between buyers and offers
- **Confidence**: standard confidence of the UCB
- It works like UCB but when it detects a significant difference in *empirical mean*margin*, it resets everything

$$s_+ = (x_t - \hat{x}) - \varepsilon$$

$$s_- = -(x_t - \hat{x}) - \varepsilon$$

$$g_+ = \max(0, g_+ + s_+)$$

$$g_- = \max(0, g_- + s_-)$$

$$\textit{change detected if } g_+ > h \textit{ or } g_- > h$$

x_t: *sample at time t*

$\hat{x}$: *empirical mean*

s_+: *positive deviation from the mean*

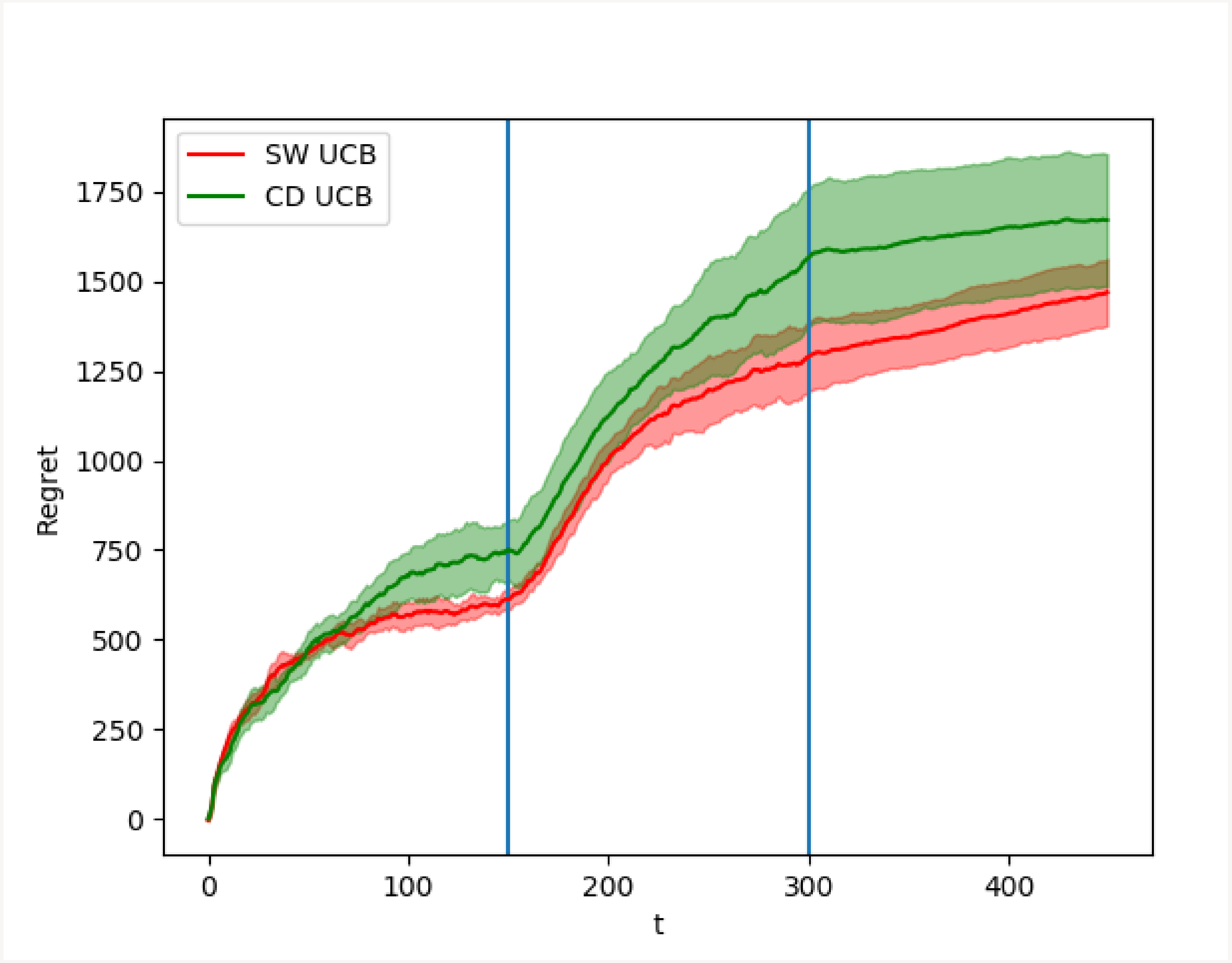
s_-: *negative deviation from the mean*

g_+: *cumulative positive deviation from the mean*

g_-: *cumulative negative deviation from the mean*

h: *threshold*

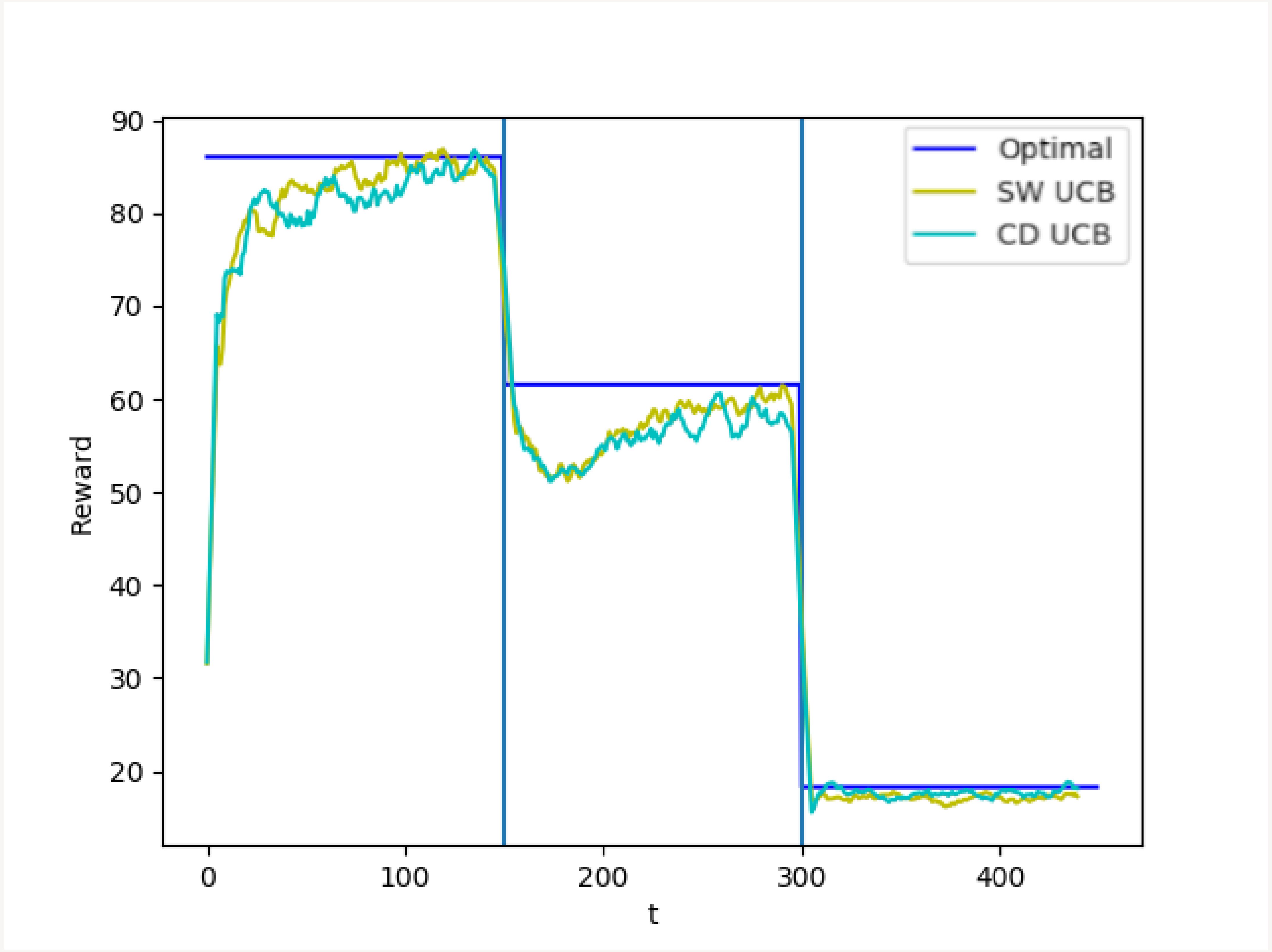
Step 6 - Graphs



The graph shows the cumulative regret of the algorithms:

- 5 experiments
- Time horizon: 450 days
- As expected after 150 days both plots have a **increase** in regret because learners have to estimate the new conversion rates
- Even if at day 300 the prices change, there is not a significant increase in regret
- CD performs worse than SW because:
    - **CD is unstable** and sometimes detect small changes and resets everything
    - **SW keeps updating its variables**

Step 6 - Graphs



The graph shows the daily rewards of the algorithms:

- 5 experiments
- Time horizon: 450 days
- At the end of every phase the rewards achieve to **converge to the computed optimal**
- In the second phase it takes more time to converge because there's a **major change** in conversion rates

Step 7

Goal: Context generation

Input :

- Configuration Files:
  - Margin
- Simulation output:
  - Buyers
  - Offers
  - Alphas
  - N° of items sold

Output:

- Estimated conversion rates
- Estimated Alphas
- Estimated Number of sold items
- Price configuration

Step 7 - TS

- Every product is associated with a **TS learner for each user class**
- Every arm is associated to a **specified price**
- **Beta parameters**: they define a distribution which mean is the estimated conversion rate

1. Every day, for each arm draw a sample ***b*** from the beta distribution
2. Select the arm that **maximizes** this formula $b * margin * estimated\ alphas * estimated\ n° \ of\ sold\ items$
3. Run the simulation with the configuration of prices computed by the learners
4. **Update** the beta parameters of each arm: $$\alpha_t \rightarrow \alpha_{t-1} + buyers$$ $$\beta_t \rightarrow \beta_{t-1} + offers - buyers$$
5. Update the **estimated alphas** and **n° of sold items**

Step 7 - UCB

- Every product is associated with a **UCB learner for each user class**
- Every arm is associated to a **specified price**
- **Empirical mean:** estimation of conversion rate computed as the weighted average of ratio between buyers and offers
- **Confidence:** standard confidence of the UCB

1.Every day, for each arm compute the upper confidence as $empirical\ mean + confidence$

2.Select the arm that **maximizes**

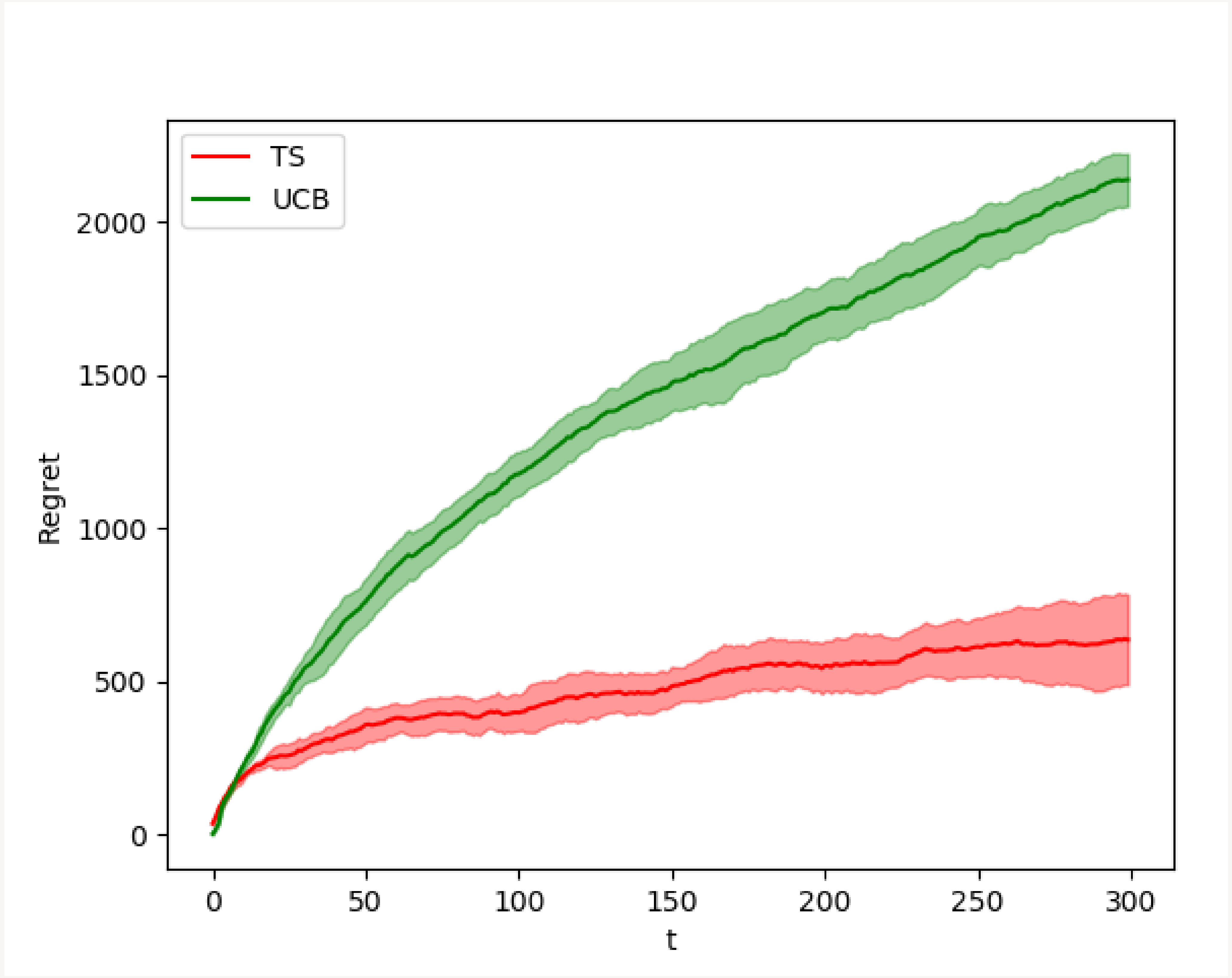
$$upper\ confidence * margin * estimated\ alphas * estimated\ n° \ of\ sold\ items$$

3.Run the simulation with the configuration of prices computed by the learners

4.**Update** the confidence according to the standard formula

5.Update the **estimated alphas** and **n° of sold items**

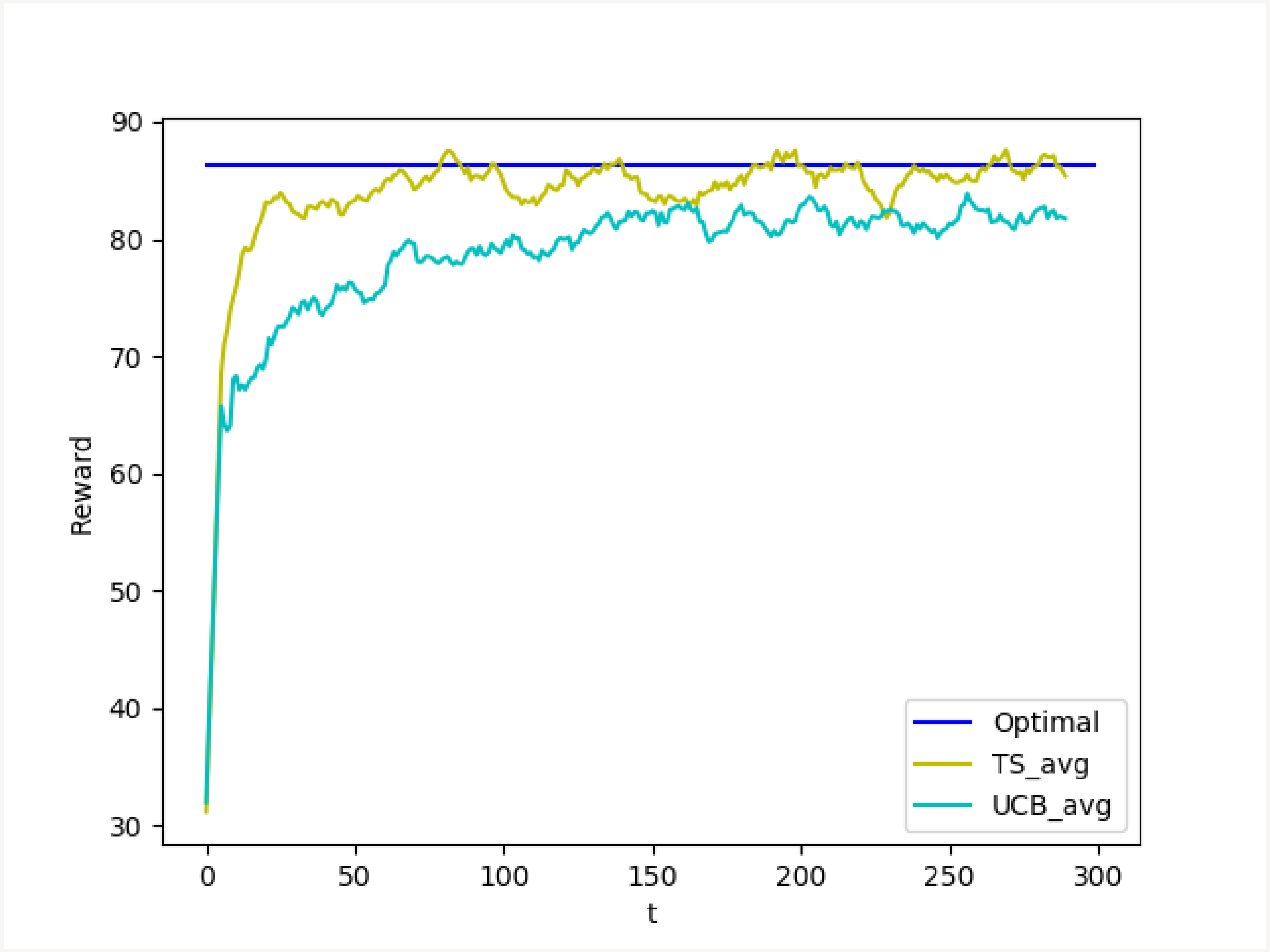
Step 7 - Graphs



The graph shows the cumulative regret of the algorithms:

- 5 experiments
- Time horizon: 300 days
- **UCB** performs **worse** with respect to the previous steps
- TS behaves similarly to the other steps

Step 7 - Graphs



The graph shows the daily rewards of the algorithms:

- 5 experiments
- Time horizon: 300 days
- Both algorithms **don't overcome** the **optimal reward** as it was expected
- Probably this result is due to the fact that we created a learner for each class when it was not necessary