Data Intelligence Applications

Pricing and Matching

Authors: *Karim Saloma, Damian Vlaicu, Valentín Chávez, Luciano Mansonar and Gabriel Evensen*

July 15, 2021

Abstract

Pricing is the problem of choosing the price for a product that will maximize the returns of a store, while matching is the problem of assigning resources. In this report we are going to simulate a store selling two products and offers a series of discounts for the second. The aim is to solve the problem of finding the best price for both products and the optimum promotion structure to maximize the profit of the store.

The environment is built of four different classes of clients that arrive at random to the store. Each class has different predisposition to buy each product at different prices, and this predisposition can change during the year.

The report is divided in eight different tasks, the first two provides a mathematical formulation of the problem in deterministic and stochastic case. In the third one the UCB and Thompson algorithms are implemented to solve the problem of fixing the price for the first item. Then the algorithms are adapted to a sequential arrival of customers, to solve the problem of picking the price of both items. In task five the UCB matching algorithm is introduced, to help with the assignment of the promotions. Then, in task six, both methodologies are mixed to find the solution to the problem where both prices and the matching needs to be fixed. Finally in task seven and eight a non-stationary environment is introduced, and the sliding-window and change-detection algorithms are implemented.

The results are reported of each algorithm including the convergence of the implemented algorithm and a comparison with the clairvoyant algorithm.

Contents

1 Introduction

In this technical project report, consider the case in which a shop has a number of promotion codes to give to the customers who are buying an item in order to get the customer to buy an additional different item. The customers can belong to different classes and the promotion codes provide the customer with different levels of discounts.
The structure of this project consists of 8 points starting from a mathematical formulation of the problem in the case when all the parameters are known. Step by step we implement different algorithms, beginning with a stationary environment and ending with non-stationary environment. This is done in order to optimize the profit when selling two different products. Finding the optimal price for the two products is a pricing problem. The introduction of promotions, in this case discounts, on the second product if the customer buys the first one makes this also into a matching problem. The environments and the algorithms has to be built using python. Plots are made to visualize the regret and the reward of every algorithm. In order to make the algorithms and environment more intuitive the creation of scenario is made and presented in the beginning of the report.

2 Scenario

Consider the scenario in which there is a technology consumer product retailer selling two products. The first product is a smartphone produced by a well known manufacturer, named product 1. The second product is a smartwatch produced by a lesser known manufacturer, named product 2. In order to boost the sales of the smartwatch from the less known manufacturer, the retailer offers a promotion bundle to customers that purchase the smartphone. This means that all customers who buy a smartphone, will get a promotion offer on the smartwatch. The number of promotion bundles is preset by the retailer business unit and are offered to the customer based on which customer class they belong to. In total there are four classes of customers, each customer is classified into one of these four classes:

1. Young customers under 30 years of age
    (a) Willing to pay considerable amount of money for a smartphone but have a somewhat limited budget
    (b) Interested in buying a smartwatch and will pay a higher price if the product capabilities justifies it.
2. Middle aged customers over 30 years of age.
    (a) Willing to pay larger amount for the smartphone and have larger upper bound on their budget

(b) Not as interested in a smartwatch as class 1 however they would buy it in case of an attractive offer

3. Older customer over 60 years of age

   (a) Not as interested in buying tech products as the first two classes and are therefore not willing to spend as much money

   (b) Not as interested in a smartwatch as class 1 however they would buy it in case of an attractive offer

4. Recurring loyalty card holder customer

   (a) As a loyal customer they expect attractive prices however they have a trusting relationship with the retailer therefore they are motivated to buy a smartphone from the retailer

   (b) Motivated in buying smartwatch due to the bundle offered by the retailer and have increased interest as the bundle discount is increased

The cost of the smartphone is 550 per unit for the retailer. Table 1 presents the conversion rates for each class given seven price candidates.

Table 1: Product 1 conversion rates

	Price / Margin						
	600 / 50	**700 / 150**	**800 / 250**	**900 / 350**	**1000 / 450**	**1100 / 550**	**1200 / 650**
Class 1	0.45	0.60	0.57	0.52	0.37	0.15	0.08
Class 2	0.50	0.55	0.51	0.47	0.42	0.35	0.21
Class 3	0.45	0.42	0.35	0.27	0.14	0.10	0.05
Class 4	0.65	0.70	0.67	0.55	0.30	0.21	0.11

Consider a customer buying the smartphone, the costumer is then given a promotion on the smartwatch. The promotion consists of four different discount levels on the smartwatch. The four levels are P_0, P_1, P_2 and P_3. They give a discount of 0% 10% 20% and 25% respectively.
The cost of the smartwatch is 50 per unit for the retailer. The following table present the price and the margin for each original price and promotion.

Table 2: Product 2 discounted prices and margins

	Price / Margin						
	Price 1	**Price 2**	**Price 3**	**Price 4**	**Price 5**	**Price 6**	**Price 7**
P_0	70/20	75/25	80/30	85/35	90/40	95/45	100/50
P_1	63/13	67.5/17.5	72/22	76.5/26.5	81/31	85.5/35.5	90/40
P_2	56/6	60/10	64/14	68/18	72/22	76/26	80/30
P_3	52.5/2.5	56.25/6.25	60/10	63.75/13.75	67.5/17.5	71.25/21.25	75/25

Table 3 shows the conversion rate for all classes given the price and the promotion:

Table 3: Product 2 conversion rates

	Price 1	**Price 2**	**Price 3**	**Price 4**	**Price 5**	**Price 6**	**Price 7**
Class 1/P0	.57	.55	.53	.48	.46	.43	.32
Class 1/P1	.6	.58	.56	.55	.53	.48	.46
Class 1/P2	.67	.65	.6	.58	.56	.55	.53
Class 1/P3	.69	.67	.65	.6	.58	.56	.55
Class 2/P0	.38	.35	.31	.25	.22	.2	.18
Class 2/P1	.4	.39	.37	.35	.31	.25	.22
Class 2/P2	.49	.43	.4	.39	.37	.35	.31
Class 2/P3	.53	.49	.43	.4	.39	.37	.35
Class 3/P0	.17	.12	.08	.06	.06	.05	.04
Class 3/P1	.24	.2	.15	.12	.08	.06	.06
Class 3/P2	.37	.3	.23	.2	.15	.12	.08
Class 3P3	.39	.36	.3	.23	.2	.16	.12
Class 4/P0	.49	.35	.29	.25	.21	.18	.15
Class 4/P1	.53	.5	.46	.34	.29	.25	.21
Class 4/P2	.65	.55	.52	.5	.46	.34	.26
Class 4/P3	.68	.65	.55	.52	.5	.46	.35

The following assumptions are made in this scenario:

- On average there are 500 customers per day arriving to the store
- The average number of customers in each class has the following: distribution $[200, 150, 50, 100]$ for classes 1, 2, 3 and 4 respectively
- Promotion distribution settings:
  - Setting 1: $[0.3, 0.15, 0.25]$ for promo levels P_1, P_2, and P_3 respectively
  - Setting 2: $[0.25, 0.35, 0.2]$ for promo levels P_1, P_2, and P_3 respectively
  - P_0 is unlimited in all cases since it corresponds to no discount at all
- Conversion rates for the non-stationary second phase:
  - Conversion rates for first product decreased for classes 1 & 2 while staying the same for classes 3 & 4
  - Conversion rates for second product decreased for classes 3 & 4 while staying the same for classes 1 & 2

3 Point 1

3.1 Approach

In this first part we need to provide a mathematical formulation of the problem in the case we know all parameters. The optimization is based on the average

number of customers in each class. To provide the mathematical formulation we need to introduce the following notation:

- C_i: is the average number of customers of the class i that arrive each day at the store.
- P_t is the price that the store chooses for the product 1 at time t.
- Q_t is the price that the store chooses for the product 2 at time t.
- $M(P)$ is the margin of the store when it sells one unit of the product 1 at price P.
- $N_f(Q)$ is the margin of the store when it sells one unit of the product 2 at price Q and with the promotion f.
- $R_{i,t}(P)$ is the conversion rate of the class i, for the first product, at time t, given that the price is P.
- $S_{i,t,f}(Q)$ is the conversion rate of the class i, for the second product, at time t, given that the price is Q and the promotion f was offered.
- $F_{i,t,f}$ is the fraction of the promotion f that is offered to the class i ad time t.

For the first product, the total gain of the store in one day will correspond to the margin of the price that it chooses times the amount of clients that buy, which corresponds to the sum of the average number of clients of each class times the conversion rate of the class at the designed price. This means that the total margin can be expressed as:

$$TM_1^t = M(P_t) \cdot \sum_{i=0}^{3} C_i \cdot R_{i,t}(P_t)$$

The amount of customers of each class that can buy the second product corresponds only to the fraction that bought the first one, then the total amount of customers, of class i, that will buy the second item is:

$$\sum_{f=0}^{3} F_{i,t,f} \cdot C_i \cdot R_{i,t}(P_t) \cdot S_{i,t,f}(Q_t)$$

where we need to sum over all the promotions to obtain the total amount. If we sum over all the classes, and include the margin of the second product we are left with the second part of the total margin of the store for each day, which corresponds to:

$$TM_2^t = \sum_{i=0}^{3} \sum_{f=0}^{3} N_f(Q_t) \cdot F_{i,t,f} \cdot C_i \cdot R_{i,t}(P_t) \cdot S_{i,t,f}(Q_t)$$

By summing both Total Margin terms, over the interval of a year, we are left with the total gain of the store in one year:

$$TM = \sum_{t=0}^{365} \left(M(P_t) \cdot \sum_{i=0}^{3} C_i \cdot R_{i,t}(P_t) + \sum_{i=0}^{3} \sum_{f=0}^{3} N_f(Q_t) \cdot (F_{i,t,f} \cdot C_i \cdot R_{i,t}(P_t) \cdot S_{i,t,f}(Q_t) \right)$$

This is the amount that the store will try to maximize, and the variables that they need to fix are the set of prices P_t and Q_t of both products and the set of fractions $T_{i,t,f}$ of the promotions that are offered to each class. Then the store needs to solve the following optimization problem:

$$\max_{P_t, Q_t, F_{i,t,f}} \quad \sum_{t=0}^{365} \left(M(P_t) \cdot \sum_{i=0}^{3} C_i \cdot R_{i,t}(P_t) + \sum_{i=0}^{3} \sum_{f=0}^{3} N_f(Q_t) \cdot (F_{i,t,f} \cdot C_i \cdot R_{i,t}(P_t) \cdot S_{i,t,f}(Q_t) \right)$$

$$\text{subject to} \quad \sum_{f=0}^{3} F_{i,t,f} = 1 \qquad \forall i \in \{0,1,2,3\}, \forall t \in \{0,\ldots,365\}$$

where the only restriction that we need to add is that the sum fraction of the promotions offered are equal to one. The nature of this optimization problem depends exclusively on the functions M, N, R and S. If the functions are linear, then the entire problem becomes a linear optimization problem, and its solution can be found using the classical Simplex algorithm. If the functions are not linear, then we need to use non-linear solvers to find the solution. For example, if the functions are continuous, any method derived from the gradient-descent algorithm will provide a suitable solution. Finally, if the prices and fractions can only belong to a finite set, then the nature of the functions becomes irrelevant but the problem is transformed into a combinatorial problem, which needs to be solved with brute-force methods.
As a final remark, since the parameters and variables of one day don't affect the other days, the original problem can be separated into small problems that are focused into the optimization of each day.

4 Point 2

4.1 Approach

In this part we need to consider that there are three different sources of randomness. First, the amount of customers of each class that arrive at the store each day, then how many of these clients will buy the first product, and finally how many of the customers that bought the first product will buy the second one.
The amount of customers that arrive each day can be modelled with any distribution with support on the natural numbers. We will denote the distribution of the number of customers of class i at day t as $C_{i,t}(\theta_{i,t})$, where $\theta_{i,t}$ are the

correspondent parameters of the distribution.
The amount of customers that actually buy the first product can be modelled as the result of a Bernoulli experiment where the amount of trials is the number of customers that arrived and the probability of success is the conversion rate of the class at the correspondent price. Then the number of buyers of the first product can be modelled as a Binomial distribution with number of trials equal to the amount of customers and with success probability equal to the conversion rate.
Finally, the amount of sells of the second product can also be modelled as Bernoulli trial, but in this case the number of trials corresponds to the amount of buyers of the first item, and the probability of success is the conversion rate of the second item. Also, we need to consider that the customers are separated by the promotion that where offered.
Then, the deterministic optimization problem of Part 1 can be modified into the following stochastic optimization problem:

$$
\begin{aligned}
\max_{P_t, Q_t, F_{i,t,f}} \quad & \sum_{t=0}^{365} \left(M(P_t) \cdot \sum_{i=0}^{3} \chi^1_{i,t} + \sum_{i=0}^{3} \sum_{f=0}^{3} N_f(Q_t) \cdot \chi^2_{i,t,f} \right) \\
\text{subject to} \quad & \sum_{f=0}^{3} F_{i,t,f} = 1 \qquad \forall i \in \{0,1,2,3\}, \forall t \in \{0,\dots,365\} \\
& c_{i,t} \sim C_{i,t}(\theta_{i,t}) \\
& \chi^1_{i,t} \sim B\left(c_i, R_{i,t}(P_t)\right) \\
& \chi^2_{i,t,f} \sim B\left(F_{i,t,f} \cdot c_i \cdot R_{i,t}(P_t), S_{i,t,f}(Q_t)\right)
\end{aligned}
$$

where $C_{i,t}(\theta)$ is any probability distribution with support on the natural numbers and parameters (θ), and $B(c,p)$ is the Binomial distribution with n number of trials and p probability of success. In this case $c_{i,t}$, $\chi^1_{i,t}$ and $\chi^2_{i,t,f}$ are the random variables that models the number of customers of each class that arrive in one day to the store, the amount of people that buy the first item, and the amount of people that buy the second item, respectively.
Since this optimization problem is not deterministic we can't use the same methods as in the previous section and we need to use an online learning approach. For this kind of problem we can use a multi-armed bandit approach, where each arm is a combination of the prices for both products and a valid assignation of the promotions, and the parameters are updated after each day.

5 Point 3

5.1 Introduction

In this part the goal is to provide a solution of a pricing problem for product 1, the smartphone. In the solution of learning an optimal price of the first item,

the price of the second product, the smartwatch is fixed, meaning that both the promotion assignment and the price is constant during the learning process. The known problem parameters in this part are the conversion rate of product 2 and the number of customers arriving each day and their class distribution. The daily customers are drawn from a Gaussian distribution and the number of rounds per day are known at once since the number of customer is known.

5.2 Approach

To learn the optimal price of the first item both an upper-confidence bound approach (UCB) and a Thompson-sampling approach are adopted and their performance are compared. In this approach the number of arms are equal to the number of price candidates of product 1 which are 7 in total. The number of customers arriving to the store each day of class i are set to $class_i = \{200, 150, 50, 100\}$. The conversion rate of product 2 are the same for every arm. Using the stationary environment, the two learners are implemented and the learning process are simulated each day for 365 days. For each day, the arms of both separate learners are pulled , the reward is calculated for the chosen arm and the learners are updated.

5.3 Result

Both Thompson-sampling and UCB converged to 900. Studying the slopes of the two algorithms one can observe that the slope of the Thompson Algorithm is a bit steeper than the slope of UCB Algorithm, in Figure 1 presenting the cumulative rewards collected by the two algorithms. The cumulative reward collected by the Thompson sampling algorithm is a bit larger than the UCB algorithm.

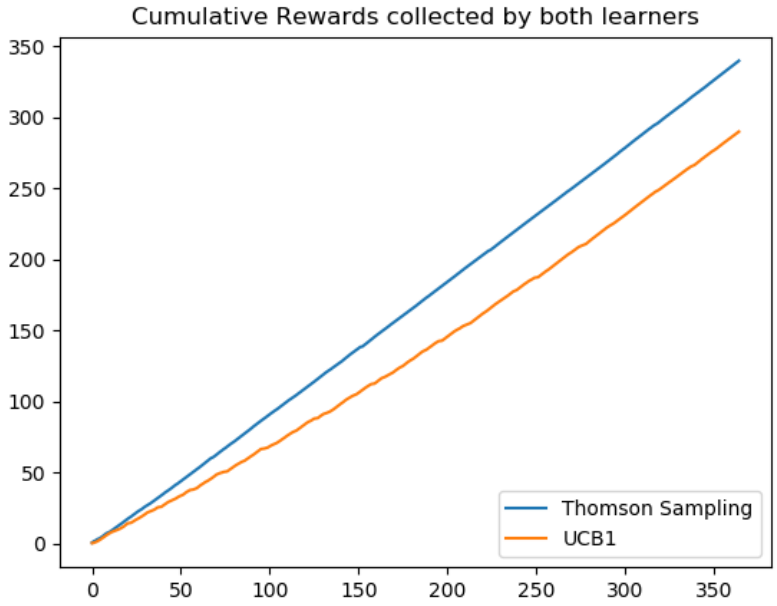


Figure 1

Figure 2 is showing the 10-day moving average of rewards collected by both learners. In this figure the observation can be made that the Thompson Sampling algorithm faster reaches a higher 10-day moving average of collected reward and the variation after reaching this higher 10-day moving average is smaller than the UCB. The 10-day moving average for the UCB algorithm takes longer time to reach the values of the Thompson algorithm and the variation in the UCB is larger than in the Thompson algorithm.

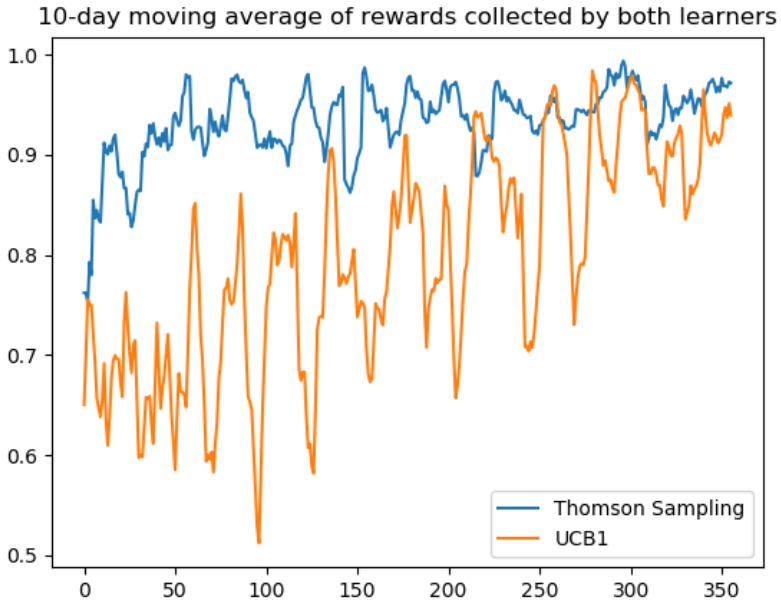


Figure 2

In Figure 3 the 10-day moving average of expected rewards for each algorithm is presented. The three algorithms are Thompson sampling-, UCB- and Clairvoyant algorithm. The clairvoyant algorithm is the optimal solution to the problem calculated when all parameters in the problem are known. Studying the graph in Figure 3 one can observe that the Thompson sampling algorithm yields a solution closer to the optimal one.

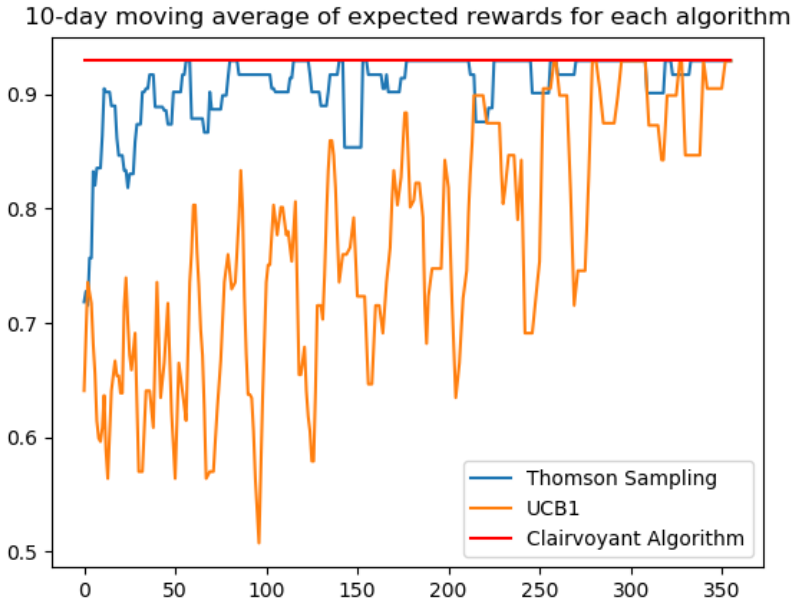


Figure 3

6 Point 4

6.1 Introduction

Similar to Part 3 but in this case the conversion rate associated with product 2 and also the amount of daily customers is not known. The goal is to provide a solution of a pricing problem for both product 1 and 2. The unknown daily number of customers are drawn from a Gaussian distribution. The learning process is applied for every arriving customer each day were the customer class of the arriving customer is drawn randomly according to the predefined class distribution.

6.2 Approach

In order to learn the optimal price of both the first product and the second product an upper-confidence bound approach (UCB) is used. Different from Part 3 is the introduction of 7 price candidates for the second product. As in task 3 the promotion assignment for the product is constant. The number of arms in the learners for product 1 and 2 is going to be seven each, equal the number of price candidates. In the learning process the number of customers for each class is sampled from a normal distribution. In the simulation, the

customer arrival is a random choice of a class that has customers remaining for the day. The arm of product 1 is pulled and the reward is observed. If the reward of product 1 is positive, meaning that the customer bought this first product, the arm of the second product is pulled and the reward can be calculated. Then both learners are updated with the corresponding arm and the cumulative reward is computed.

6.3 Result

The learner for product 1 converge to 900 and the learner for product 2 converge to 80 for constant matching {(class 1, P0),(class 2, P1),(class 3, P3),(class 4, P2)}.
Figure 4 is showing the cumulative rewards from each product in the pricing problem of the two products. In these two graphs one can observe two linear slopes and the reason for this linear behaviour is due to the assumption of a stationary environment in this problem.

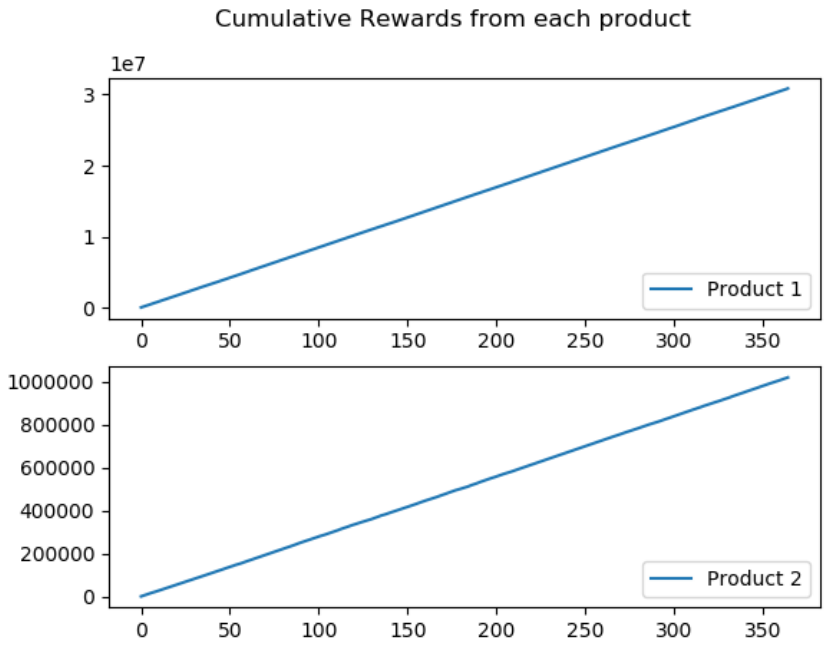


Figure 4

In Figure 5 the 10-day moving average of collected reward for the UCB algorithm is presented.

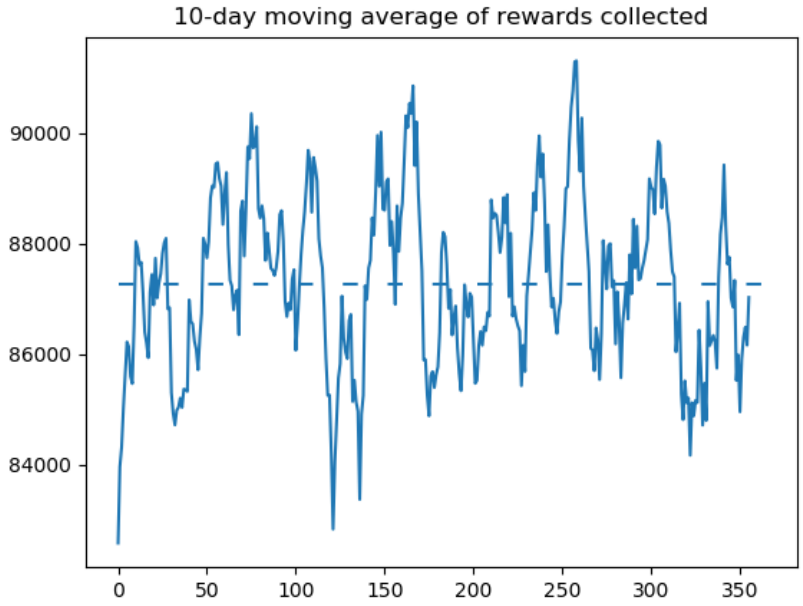


Figure 5

Figure 6 is presenting the 10-day moving average of expected rewards of both the UCB- and Clairvoyant algorithm. The UCB algorithm never reaches as large average of expected rewards as the Clairvoyant algorithm. One can see in the figure that the difference between the average over the 365 days is about 86 000 for the UCB and about 92 000 for the Clairvoyant meaning that the UCB reach about 89 % of the 10-day moving average of expected rewards of the Clairvoyant algorithm.

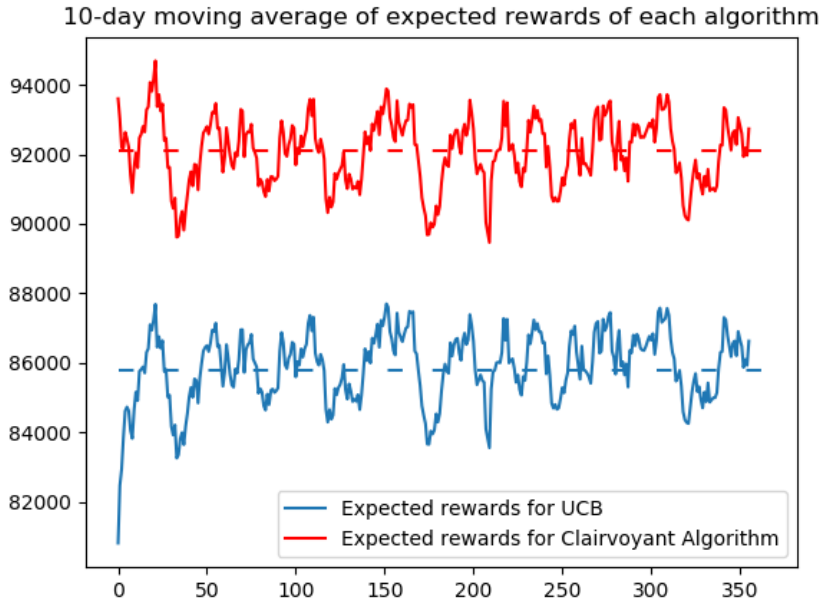


Figure 6

7 Point 5

7.1 Introduction

This part focuses on the matching problem. More precisely, the goal is to find the optimal assignment of promos to every customer class. To simplify matters, the prices are fixed. In this point there are two optional settings for these fractions are available as mentioned in the Scenarios section.

7.2 Approach

The approach is very similar to the previous part. The main difference is that the arms are no longer price candidates but rather optimal promotion assignments for every customer class. This comes down to using an assignment algorithm on a matrix where the rows represent the customer classes and the columns the promotions. Since every class can be assigned to the first promotion, three more promotion classes are introduced.

For every day we determine the number of promotions together with the number of customers. Once the arm of product 1 gets pulled and the observed reward is positive, we solve the matching problem. If there are no more promotions of a

certain type, the cost of that column in the matrix will be as high as possible so it will never be chosen. After the matching problem has been solved, we determine which promotion has been chosen for the arrived customer. The learner gets updated with the observed reward and the daily amount of promotions of the matched type gets decremented.

7.3 Result

For promo setting 1, the number of times a class was offered each promo level is shown in Table 4 below:

Table 4: Shows the number of times a class was assigned each promo level for promotion setting $[0.3, 0.3, 0.15, 0.25]$

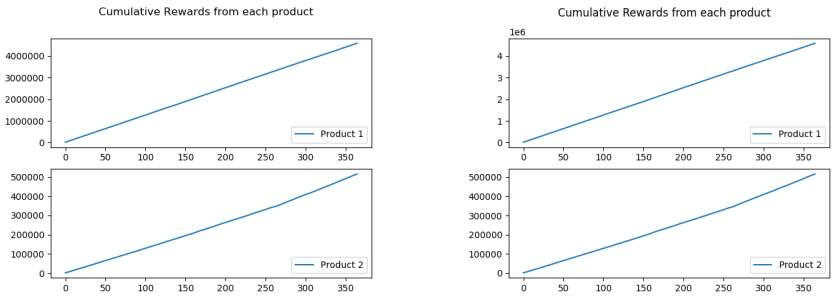
Promotion level:	P_0	P_1	P_2	P_3
Class 1	9 600	22 909	0	0
Class 2	1 243	10	26 215	19
Class 3	822	0	0	7 182
Class 4	5 857	8 043	1 594	8 212

For promo setting 2, the number of times a class was offered each promo level is shown in Table 5 below:

Table 5: Shows the number of times a class was assigned each promo level for promotion setting $[0.2, 0.25, 0.35, 0.2]$

Promotion level:	P_0	P_1	P_2	P_3
Class 1	7 091	25 433	15	0
Class 2	1 352	0	26 135	0
Class 3	847	0	0	7 157
Class 4	7 187	6 331	1 723	8 465

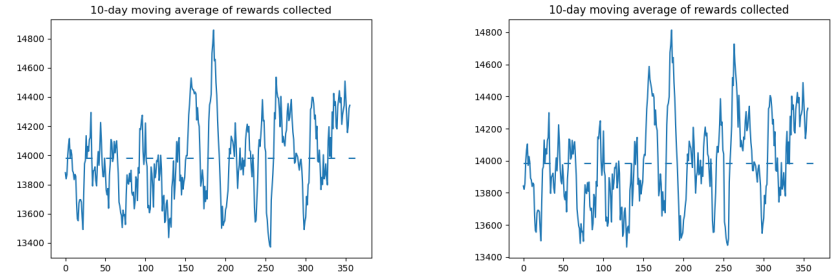
Similar to the previous point 4, one can observe the stationary environment observing the linear behaviour in the cumulative rewards in Figure 7.



(a) First setting (b) Second setting

Figure 7

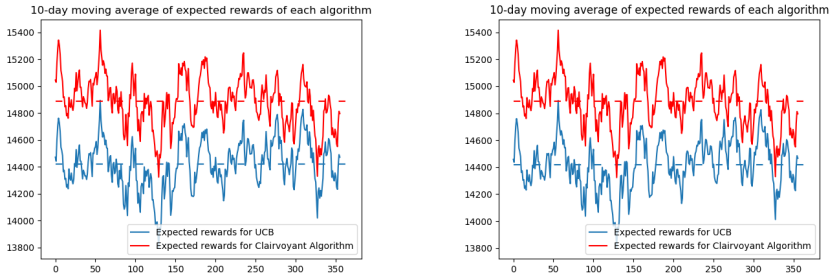
In Figure 8 the 10-day moving average of collected reward for the UCB algorithm is presented.



(a) First setting (b) Second setting

Figure 8

Observing in Figure 9a the differences between the average over the 365 days for the UCB and the Clairvoyant. UCB has an average of 14 400 while the Clairvoyant has the average of 14 900 meaning that the UCB reach about 97 % of the 10-day moving average of expected rewards of the Clairvoyant algorithm over the year. This is an improvement from the previous point 4.



(a) First setting

(b) Second setting

Figure 9

8 Point 6

8.1 Introduction

Similar to part 5 except that the pricing for product 1 and product two is not constant anymore. The problem is now a pricing and matching problem. In these part the following assumptions are made about the environment; it is stationary and the arrival of customers is sequential meaning that reward returned for each customer is based on class and price and not the reward of a whole day. The daily number of customers are drawn from a Gaussian distribution. A learning process is applied for each arriving customer each day. From what customer class the arriving customer belongs to is drawn at random according to a certain class distribution. Promotion level distribution is set as a fraction of the total customers arriving and are constant. Two optional settings for these fractions are available as mentioned in the Scenarios section.

8.2 Approach

The approach used in this part is a UCB approach for the pricing problem of the first product and a matching UCB approach for the matching problem of the second product. The number of arms for the pricing learner are equal to the number of price candidates which are 7 in total. For the second matching problem of product 2, the number of arms should be equal to 112, but in order use the linear sum assignment algorithm to solve the our optimization problem the creation of 84 more arms are made. These arms are additional copies of the P_0 promotion assignment. As before the number of customers for each class are sampled from a normal distribution and truncated at 0 to avoid negative numbers. The simulation of customer arrival is made by a random choice of class which has customers remaining for the day. As in the previous part we pull an arm for the first product and observe the reward. If the reward is positive we pull the second price arm, else the reward of the second product is set to

zero. Then the learners are updated. For every customer, the reward calculated for each customer is summarized into a cumulative reward and from this the expected reward is calculated for each day.

8.3 Result

For both promotion settings class 1 learner converged to price 900 for product 1 and 70 for product 2 at promotion level 1, class 2 converged to price 1100 for product 1 and price 70 for product 2 at promotion level 2, class 3 converged to price 800 for product 1 and price 70 for product 2 at promotion level 3 and class 4 converged to price 800 for product 1 and 75 for product 2 at promotion level 0. In the tow following tables, the number of times a class were assigned a each promotion level is presented.

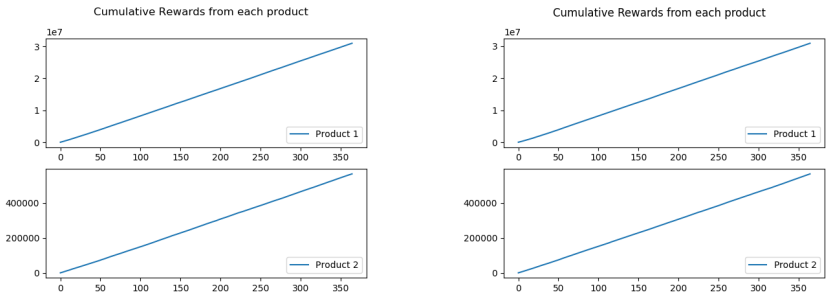
Table 6: Shows the number of times a class was assigned each promo level for promotion setting $[0.3, 0.3, 0.15, 0.25]$

Promotion level:	P_0	P_1	P_2	P_3
Class 1	4	37 614	0	35
Class 2	2	0	19 630	22
Class 3	1	0	0	5 556
Class 4	3	23 960	0	6

Table 7: Shows the number of times a class was assigned each promo level for promotion setting $[0.2, 0.25, 0.35, 0.2]$

Promotion level:	P_0	P_1	P_2	P_3
Class 1	1	37 504	43	0
Class 2	1	0	19 596	3
Class 3	1	0	0	5 583
Class 4	2	24 023	13	0

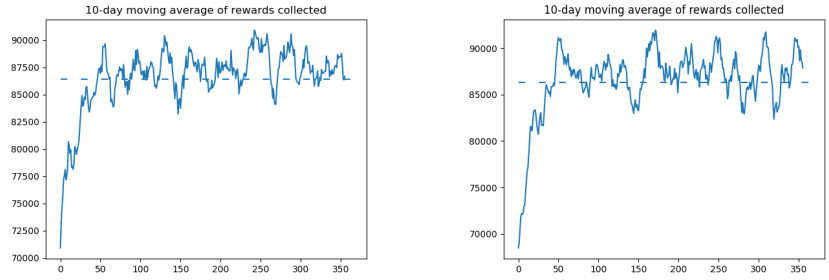
In Figure 10 the cumulative reward for each product is presented. Figure 10a and 10b presents the cumulative reward for the first and second promotion setting respectively.



(a) First setting (b) Second setting

Figure 10

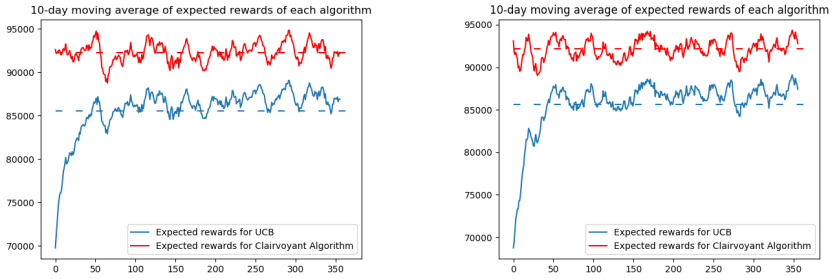
In Figure 11 the 10-day moving average of rewards collected is presented. Figure 11a and 11b presents the 10-day moving average for the first and second promotion setting respectively. Studying the slopes of Figure 10 it is possible to observe the stationary behaviour as the slopes are linear.



(a) First setting (b) Second setting

Figure 11

In Figure 12 the 10-day moving avarage of expected rewards of both the UCB- and Clairvoyant algorithm are presented. Figure 12a and 12b represent the first and second promotion setting respectively.



(a) First setting

(b) Second setting

Figure 12

9 Point 7

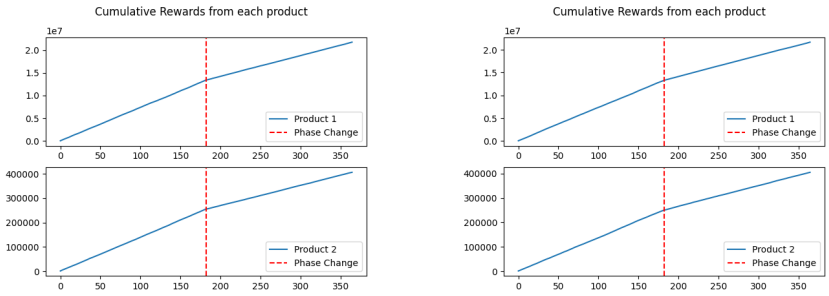
9.1 Introduction

The scope of this part is to test the algorithms in non-stationary environments, which means that the parameters will not be constant over time. In our case, we divided the year into two phases of equal time: in the first one, all the parameters are the same as in the previous tasks, and in the second one, the conversion rates are changed as described in the *Scenario* section.

9.2 Approach

Since the parameters are changing, the learners need to be able to adapt to these changes. One approach is to use the Sliding-Windows method. In this method, we select a time window of length t and train the learners only with the returns of the last t days. This technique allows the learner to update the parameters, ignoring the possible outdated information.
To adapt the algorithm to our approach and to be sure that all the clients are considered in a round we need to use a window of length equal to the square root of the period times the expected amount of clients of the day.

9.3 Result



(a) First setting

(b) Second setting

Figure 13

In graph 13 it is possible to observe the effects of the non-stationarity. Specifically, we can observe that the slope of cumulative rewards of product two is smaller in the second phase than in the first. This reduction is produced by the decrease in the conversion rate of this product, which is translated into fewer sales.

In the following tables is summarized the amount of promotions that are offered to each class in the two phases, for both settings of promotions.

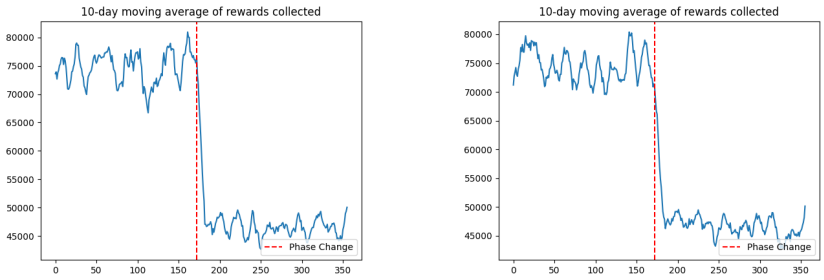
Table 8: Shows the number of times a class was assigned each promo level for promotion setting $[0.3, 0.3, 0.15, 0.25]$

	First phase				**Second phase**			
Promotion level:	P_0	P_1	P_2	P_3	P_0	P_1	P_2	P_3
Class 1	10	17130	0	0	10	10578	0	0
Class 2	10	3	11192	0	10	12	6559	0
Class 3	10	0	4	2299	10	0	3	2352
Class 4	20	9537	0	19	20	9337	1	72

Table 9: Shows the number of times a class was assigned each promo level for promotion setting $[0.20.250.350.2]$

	First phase				**Second phase**			
Promotion level:	P_0	P_1	P_2	P_3	P_0	P_1	P_2	P_3
Class 1	10	17146	32	0	10	10479	0	0
Class 2	10	6	11156	6	10	20	6504	0
Class 3	10	1	2	2321	9	0	16	2287
Class 4	20	9387	9	168	18	9557	1	83

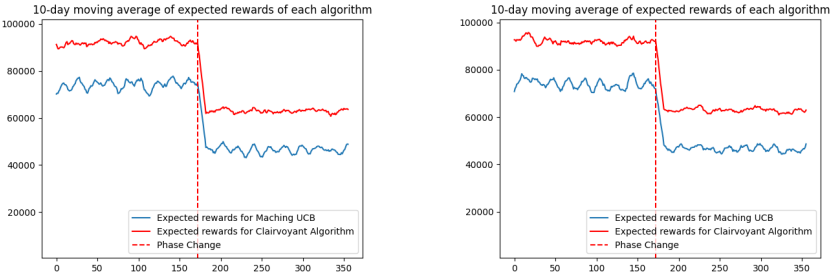
From both tables is easy to see that, even if we are constantly forgetting information from past days, the algorithm converges to a promotion offering structure. This phenomenon is also observed when we look at the price for the products that are chosen by the UCB algorithbm, which are 900 and 70 for classes one and three, 1000 and 70 for the second class and 900 and 75 for the fourth class.



(a) First setting (b) Second setting

Figure 14

In the previous sections, we could see that one of the weaknesses of the UCB approach is the amount of time and information that it needs to converge to the optimal configuration, especially in cases like our problem, where the amount of arms to test is high. Because we always forget information, this weakness is increased by the sliding window approach. This is shown in the figure 14, where we can see that the daily rewards does not increase, but oscillates between a mean that is constant during each phase.



(a) First setting (b) Second setting

Figure 15

This phenomenon is evidenced clearly when we compare the results of a clairvoyant algorithm with our results, as shown in figure 15, where we can see that the returns of the UCB algorithm never approach the optimal results.

10 Point 8

10.1 Introduction

The goal of this section is to, once again, test the algorithms in non-stationary environments, but now by adopting a change-detection approach rather than a sliding-window approach. Thus the parameters will not be constant over time. The environment setup is identical to the previous part.

10.2 Approach

This approach uses the cumulative sum (CUSUM) algorithm which has been proven to be optimal in detecting abrupt changes. The basic idea of the CUSUM algorithm is to take a function of the observed sample as the step of a random walk. This random walk is designed to have a positive mean drift after a change point and have a negative mean drift without a change. Hence, CUSUM signals a change if this random walk crosses some positive threshold.[1]

10.3 Result

For promotion setting 1 [0.3, 0.3, 0.15, 0.25] the following result was produced:

Phase 1 results:

- class 1 learners converged to price 900 for product 1 and price 70 for product 2
- class 2 learners converged to price 1100 for product 1 and price 70 for product 2
- class 3 learners converged to price 900 for product 1 and price 70 for product 2
- class 4 learners converged to price 900 for product 1 and price 75 for product 2

Table 10 are showing the number of times a class was assigned each promo level:

Table 10: Number of times a class was assigned each promotion level for promotion setting 1

Promotion level:	P_0	P_1	P_2	P_3
Class 1	20	18 405	1	27
Class 2	11	0	9 856	19
Class 3	1	0	0	2 509
Class 4	3	10 343	4	19

[1] https://arxiv.org/pdf/1711.03539.pdf

Phase 2 results:

- class 1 learners converged to price 900 for product 1 and price 70 for product 2
- class 2 learners converged to price 900 for product 1 and price 70 for product 2
- class 3 learners converged to price 900 for product 1 and price 70 for product 2
- class 4 learners converged to price 900 for product 1 and price 75 for product 2

Table 11 are showing the number of times a class was assigned each promo level:

Table 11: Number of times a class was assigned each promotion level for promotion setting 1

Promotion level:	P_0	P_1	P_2	P_3
Class 1	0	11 573	0	0
Class 2	0	0	6 834	0
Class 3	0	0	0	2602
Class 4	0	10 165	0	0

For promotion setting 2 [0.2, 0.25, 0.35, 0.2] the following result was produced:

Phase 1 results:

- class 1 learners converged to price 900 for product 1 and price 70 for product 2
- class 2 learners converged to price 1000 for product 1 and price 70 for product 2
- class 3 learners converged to price 900 for product 1 and price 70 for product 2
- class 4 learners converged to price 900 for product 1 and price 75 for product 2

Table 12 are showing the number of times a class was assigned each promo level:

Table 12: Number of times a class was assigned each promotion level for promotion setting 2

Promotion level:	P_0	P_1	P_2	P_3
Class 1	9	18 161	67	18
Class 2	1	0	10 687	32
Class 3	1	0	0	2 532
Class 4	2	10 464	31	0

Phase 2 results:

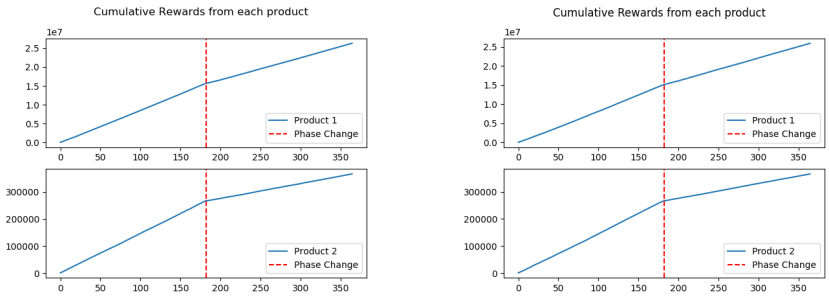
- class 1 learners converged to price 900 for product 1 and price 70 for product 2
- class 2 learners converged to price 900 for product 1 and price 70 for product 2
- class 3 learners converged to price 900 for product 1 and price 70 for product 2
- class 4 learners converged to price 900 for product 1 and price 75 for product 2

Table 13 are showing the number of times a class was assigned each promo level:

Table 13: Number of times a class was assigned each promotion level for promotion setting 2

Promotion level:	P_0	P_1	P_2	P_3
Class 1	0	11 645	0	0
Class 2	0	0	6768	0
Class 3	0	0	0	2 525
Class 4	0	10 219	0	0

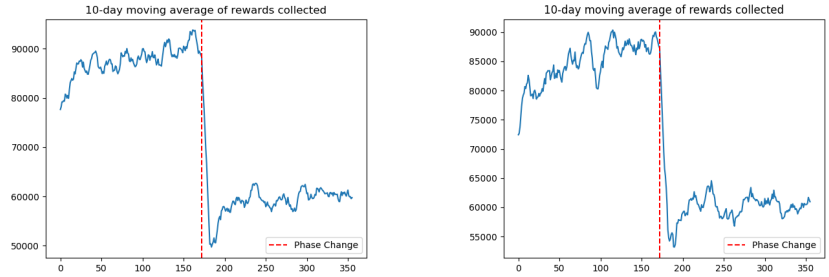
The observed behaviour in Figure 16 is similar to Figure 13. The slope of cumulative rewards of both products is smaller in the second phase due to a decrease in the conversion rate of these products.



(a) First setting (b) Second setting

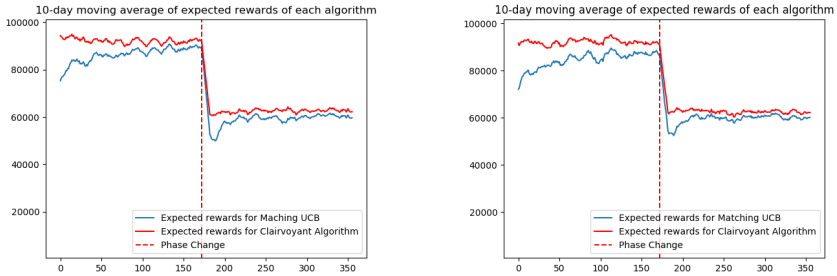
Figure 16

In figure 17 we can clearly see that the graph is slowly converging to a specific value. As soon as the second phase starts, the algorithm detects an abrupt change in environment and restarts the converging process.



(a) First setting (b) Second setting

Figure 17



(a) First setting (b) Second setting

Figure 18

11 Conclusion

In this report, we studied the problem of a store that needs to choose the price of two products and offers different levels of promotions for one of the items. The environment was composed of four different classes of clients, each one with a different predisposition to buy each product.
First, we described the optimization problem that the store was facing and gave the mathematical formulation in the deterministic case, where all the parameters were known. Then it was formulated the stochastic case, by modeling the arrival of the customers of each class as a truncated-Gaussian variable and the number of sales as a Bernoulli experiment. To solve this problem, the necessity of a multi-armed bandit approach was argued.
The whole problem was subdivided into different tasks, each one more complex than the previous. First, we started with the problem where everything was fixed except the price of the first product. The UCB and Thompson algorithms were tested, obtaining a greater reward for the later one mentioned. Then the task of choosing the price for both products was introduced, to solve this, we updated the UCB algorithm to one that was able to work under an environment with a sequential arrival.
The next task was to match the promotion levels with the classes under the assumption that both prices were fixed. To do this, two different promotion settings were tested using an UCB-matching algorithm, obtaining results that converged towards the same value as the clairvoyant algorithm. For the next task, both the pricing and the matching need to be done simultaneously. By mixing the results of the last parts, we were able to propose an algorithm that could solve this problem and approximate the results of the clairvoyant algorithm.
Finally, a non-stationary environment divided into two phases was introduced, where in each phase, the clients' conversion rates changed over time. Two algorithms were implemented to solve this new problem. The first one, the sliding-window, could not get closer to the optimum reward in neither of the phases. On the contrary, the change-detection approach approximated the optimal configuration and adapted rapidly to the phase change.