University of Piraeus
Department of Digital Systems
MSc Digital Systems and Services
Specialization Big Data & Analytics

Online Machine Learning for Bilateral Trading Problems

EIRINI ANGEOPOULOU ME 1917

SUPERVISING PROFESSOR ORESTIS TELELIS

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Assistant Professor Orestis Telelis of the Department of Digital Systems at University of Piraeus, for his constant guidance and support. I would also like to express my gratitude to my family and friends for their continuous encouragement. Finally, I owe my deepest gratitude to Nikolaos Sambatis. I am forever thankful for the unconditional love and support throughout the entire thesis process and every day.

Eirini Angelopoulou

ABSTRACT

In this work, we are studying Online Learning and its application in Online Auctions through intermediation. These days, all transactions on modern platforms are conducted exclusively through real-time online auctions. The design of such auctions and their properties have initiated a lot of research in the past years, focusing on designing the optimal auction mechanism to maximize the seller's revenue.

Specifically, throughout this study, we explore bibliographically the Online Learning and its key ideas and methods. We begin with the Full Feedback and the Expert Advice methods, followed by the Partial Feedback and the Bandit general setting. Afterwards we continue our investigation with the problem of Online Auctions and the Double Auction or Bilateral Trade setting, which we will also examine in the experimental section too.

For the experiment, we explore the algorithms we presented in the section on Online Learning. Initially, we selected three Full-feedback algorithms, the **Follow the Best Price** algorithm, a modified **Follow the Best Price with Greedy sampling**, and the **Exponential Weighted Average** algorithm. We then continued the analysis with an **ε-Greedy** algorithm from the Partial-feedback category of algorithms.

We compare the performance of the above algorithms in the Bilateral Trade setup, selecting four different sets for the sellers' and the buyers' prices. All the algorithms tested achieved small (logarithmic) regret and generally the findings verify the theoretical results. Comparing the results of the algorithms, the **Follow the best price** algorithm is the best performing algorithm for most of the sets while the **ε-Greedy** algorithm is the worst. Our modified **Follow the Best Price with Greedy sample** algorithm achieved the best results, actually outperforming the original **Follow the Best Price** algorithm, in one of the four experiments.

ΠΕΡΙΛΗΨΗ

Στα πλαίσια της παρούσας διπλωματικής εργασίας, θα ασχοληθούμε με την θεματική του «Online Learning» και τις εφαρμογές του στις «online» δημοπρασίες με διαμελάβηση. Στις μέρες μας όλες οι συναλλαγές, στις σύγχρονες ηλεκτρονικές πλατφόρμες, πραγματοποιούνται αποκλειστικά μέσω «online» δημοπρασιών σε πραγματικό χρόνο. Ο σχεδιασμός τέτοιων δημοπρασιών αλλά και η ανάλυση των ιδιοτήτων τους έχουν δώσει κίνητρο για διεξαγωγή μεγάλου όγκου ερευνών τα τελευταία χρόνια. Εστιάζοντας κυρίως στον σχεδιασμό της βέλτιστης δημοπρασίας με στόχο την μεγιστοποίηση των εσόδων του πωλητή.

Συγκεκριμένα, ερευνούμε βιβλιογραφικά το πρόβλημα του «Online Learning» και παρουσιάζουμε κάποιες βασικές ιδέες και μεθόδους. Ξεκινώντας με την κατηγορία προβλημάτων «Πλήρους Πληροφορίας» με «Συμβουλή ειδικού», και «Μερικής Πληροφορίας» προβλήματα με «Κουλοχέρηδες». Με σκοπό στην συνέχεια να εμβαθύνουμε στις «online» δημοπρασίες και τις δίπλες δημοπρασίες, τις οποίες θα χρησιμοποιήσουμε παρακάτω και στο πειραματικό κομμάτι της εργασίας.

Αναλυτικά, για το πειραματικό σκέλος εξετάζουμε τους αλγόριθμους που παρουσιάσαμε στην ενότητα του «Online Learning». Αρχικά επιλέγουμε τρεις «Πλήρους Πληροφορίας» αλγορίθμους, τον «Follow the Best Price» αλγόριθμο, μια τροποποιημένη εκδοχή του πρώτου την οποία αναφέρουμε ως «Follow the Best Price with Greedy sampling», και τον «Exponential Weighted Average» αλγόριθμο. Και στην συνέχεια επιλέγουμε τον «ε-Greedy» αλγόριθμο από τις «Μερικής Πληροφορίας» μεθόδους.

Πραγματοποιούμε πειράματα με την χρήση των προαναφερθέντων αλγορίθμων, σε τέσσερα διαφορετικά σύνολα τιμών αγοραστών και πωλητών. Παρατηρούμε ότι σχεδόν όλοι οι αλγόριθμοι που εξετάσαμε πέτυχαν μικρά (λογαριθμικά) «Regret». Συγκρίνοντας τους αλγορίθμους μεταξύ τους, ο «Follow the Best Price» αλγόριθμος είναι ο αλγόριθμος με την καλύτερη απόδοση για τα περισσότερα σύνολα και ο «ε-Greedy» ο αλγόριθμος με την χειρότερη. Και τέλος, ο τροποποιημένος αλγόριθμος «Follow the Best Price with Greedy sampling», ξεπερνάει σε αποτελέσματα τον αρχικό «Follow the Best Price» σε μόνο ένα από τα σύνολα που εξετάστηκαν.

TABLE OF CONTENTS

1 Introduction

In this thesis, we consider the problem of an Online Auction of a specific product by an intermediary. Meaning that we have sellers and buyers coming in pairs, in a random order, declaring their price. Every time a pair of sellers and buyers arrive, the intermediary must decide if they want to sell and buy, based on a threshold price they have set. The desired mechanism for this problem would optimize the gain, resulting in trading the item.

This problem falls under the category of Online Learning problems with intermediation. More specifically, double auction or bilateral trade is a model where the intermediary concentrates on the maximization of their efficiency.

In Online Learning, the samples arrive in a sequence or a “stream”, and the algorithm must process them in real-time and make a decision. In contrast, in Offline Learning, the algorithm receives the whole input of data, to output a decision.

In the following paragraphs, we give an overview of the Online Learning setting, some key algorithms, and some theoretical results. We introduce the Online Double Auction problem, and we examine the extent of the algorithms presented before, within this setup. Lastly, we conduct experiments of online double auctions with these algorithms and compare their performance and results.

One very popular application of double online auctions is the Ad Exchange method of selling advertisements. An intermediary (the Ad Exchange platform) interacts with both buyers (the advertisers) and sellers (the publishers) that arrive sequentially, one iteration at a time, and all the items are identical (ad slots). The intermediary must decide on each bid as they come.

These days all transactions are handled exclusively through real-time online auctions. As previously mentioned, the design of such auctions and their properties has been the subject of extensive research in the recent years, focusing on designing the optimal auction mechanism to maximize the seller’s profit. The most popular Ad Exchanges technologies now [14] are, DoubleClick (Google) having the 49.36% of the market share, AppNexus 11.92%, OpenX 10.77%, and many more.

1.1 Online Learning

In this chapter, a brief introduction of online learning methods and some popular algorithms will be presented.

Unlike the usual, batch processing, machine learning algorithms, in online learning, the samples arrive in sequential order and the algorithm processes them one at a time and updates them per interaction. That means that we do not have the usual training data set and that the algorithm updates instantly for all the newly arrived samples making these algorithms optimum for real-world applications where data are not only large but also arriving at a high velocity.

Online learning has become a promising technique for learning from continuous streams of data in many real-world applications. This comes in contrast with the usual machine learning algorithms, where they are trained with a specific data set, and then we use the model without adapting it to new data. Online learning methods appear to fit better in environments like auctions, trading stocks, search auctions, etc. where the values change frequently.

In the next sections, some cases of online learning algorithms both Full and Partial Feedback will be introduced and examined based on their value in the Online Auctions set up.

1.2 Full Feedback - Predicting from Expert Advice

The general Online Learning setting includes T rounds. An instance arrives, and the algorithm comes to a decision. Then it receives feedback and calculates the loss using some metric.

We will begin by describing the setting of Full Feedback - Online Learning with Expert Advice, which has been studied extensively in the literature as well as some metrics for judging the performance of the algorithms presented. The presentation is based on studies [8],[9].

A popular way to describe this relates to the weather example. Imagine that every day t we would like to be able to predict on a daily basis if it will rain or not, it being a binary result $\{0,1\}$. This kind of problem has been studied a lot and many machine learning algorithms can use historical data to predict or forecast the output. In this approach, we do not have historical data, but we can use the advice of N experts $E = \{E_1, \dots, E_N\}$. Each day the experts make their prediction $\{0,1\}$ and then the algorithms must use this to make their own decisions, assuming that no other input has been given to the algorithm. A few hours later we will have our feedback (rain or not) from nature.

We do not make any assumption on the experts, they could be arbitrarily correlated, or based on the quality of their information. So, the goal is to perform nearly as well as the *best expert*. *The best expert* is the one that has made the fewest mistakes so far.

When describing this example, we consider a game with a binary output $\{0,1\}$, but within another context we can have distinct or continuous value outputs. For instance, we could have the same problem, but the value of each expert's prediction would be the probability of whether it will rain or not. Based on this output we calculate the value loss for the algorithm and for all the experts. As a result, after T rounds we will have a general loss of L_T and a vector with N different losses for each of our experts $[L_{T,1}, \dots, L_{T,N}]$.

As discussed previously, in online learning it is not necessary to know machine learning metrics, such as accuracy. We could use the notion of regret that can be calculated as the difference between our algorithm's loss and the best expert's loss instead:

$$R_T = L_T - min_{i \in E} L_{T,i} .$$

The definition of the loss function for the general online setting with T rounds is presented below. The analysis is based on [9, Chapter 8].

Definition 1. [9, Chapter 8] At the t round we make a prediction $\widehat{y_t} \in \mathcal{Y}$ and receive feedback $y_t \in \mathcal{Y}$. The loss function is

$$L(\widehat{y_t}, y_t) \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+ .$$

In the literature we encounter a lot of different loss functions. Specifically, for our set up where $y \in \{0,1\}$ one common encounter is described below.

Definition 2. [9, Chapter 8] At the t round, we make a prediction $\widehat{y_t} \in \{0,1\}$, and receive feedback $y_t \in \{0,1\}$. The loss function is

$$L(\widehat{y_t}, y_t) = |\widehat{y_t} - y_t| .$$

So, in every round it can be zero if the prediction is correct and one if it is not. The objective of the online game is, after T rounds, to minimize the cumulative loss:

$$\sum_{t=1}^{T} L(\widehat{y_t}, y_t).$$

In a more generic setup with $L \in \mathbb{R}$, we present below another common loss function.

Definition 3. [9, Chapter 8] At the t round we make a prediction $\widehat{y_t} \in \mathbb{R}$, and receive feedback $y_t \in \mathbb{R}$. The loss function is

$$L(\widehat{y_t}, y_t) = (\widehat{y_t} - y_t)^2.$$

The algorithms that belong to the Online Learning with Expert Advice set-up are numerous in the literature. In the following sections, we will present the *Exponential Weighted Average* algorithm [9], and the *Follow the Leader* [13],[1] algorithm.

1.2.1 Exponential Weighted Average Algorithm

The Exponential Weighted Average algorithm, to be described in the current section, is based on [9, Chapter 8]. In the specific algorithm we assume that the loss function has continuous values $L \in [0,1]$, and will be used at the weight update rule.

The algorithm begins with a list of uniform weights, $[w_1(1), \ldots, w_N(1)]$. At every round t we receive a list with the expert's predictions $[x_1(t), \ldots, x_N(t)]$ and their weights $[w_1(t), \ldots, w_N(t)]$. And then, the algorithm's prediction is calculated by the following rule:

$$\widehat{y_t} = \frac{\sum_{i=1}^{N} w_i(t) x_i(t)}{\sum_{i=1}^{N} w_i(t)}$$

After calculating the prediction, we receive the feedback and update the weights accordingly, with the following rule:

$$w_i(t+1) = w_i(t) e^{-\eta L_i(t)},$$

where $L_i(t) = L\big(\widehat{y_i(t)}, y_i(t)\big)$, is the total loss of the expert i at the round t.

The pseudocode for the Exponential Weighted Average algorithm is shown in figure 1.1.

```
Algorithm Exponential Weighted Average [9]
Parameters: A constant $\eta > 0$
Initialization: For each expert $k$, $w_k(1) = 1$.
for t $= 1,2, \dots, T$ do:
        Make a prediction: $\hat{y}_t \leftarrow \frac{\sum_{i=1}^{N} w_i(t) x_i(t)}{\sum_{i=1}^{N} w_i(t)}$;
        Receive feedback, $y_t$;
        for every expert k do:
                Calculate the loss: $L_i(t)$;
                Calculate the weight: $w_k(t+1) \leftarrow w_i(t) e^{-\eta L_i(t)}$ ;
        end for
end for
```

Figure 1.1 **Exponential Weighted Average**

The next step is to present a bound for the Regret of the Exponential Weighted Average algorithm, as presented in [9, Chapter 8].

If we assume that the loss function will have continuous values $L \in [0,1]$ then for any value of the constant $\eta > 0$ and any series of events $y_1, \dots, y_T \in \mathcal{Y}$, the regret of the Exponential Weighted Average algorithm after T rounds will satisfy:

$$R_T \leq \frac{\log(N)}{\eta} + \frac{\eta T}{8} \ .$$

The optimum choice for the number η, expects from us to know the horizon T. However, a general method treating this, consists of interpreting η as a function of time, $\eta_t = \sqrt{(8 \log N)/t}$, which leads to a $O(\sqrt{T})$ dependency on T, that cannot be improved for general loss function.

Apart from the algorithm mentioned above, many others, belonging to the Weighted Majority family, have been presented in the literature. Some differ in the update rule, others in the loss function and bounds that they have used.

1.2.2 Follow the Leader Algorithm

The Follow the Leader algorithm, as described in [13] and [1], is a simple and intuitive strategy for Online Learning problems. We will present this method in the same general setting we described in the previous algorithm. In this framework, the leader's predictions $[x_1(t), \dots, x_N(t)]$ are a convex set, and the loss function $L_i(t)$ is a convex function with respect to its first argument.

This method, just like the previous one of the Exponential Weighted Average, is initiated by the assumption that few leaders are performing conceivably better than others as time passes. Therefore, we want to increase the probability of these leaders being selected in the next rounds.

The algorithm begins with a list of leaders' predictions $[x_1(1), \dots, x_N(1)]$ and at every round it calculates the prediction according to the following rule:

$$\widehat{y_t} = argmin_{x \in [x_1(t), \dots, x_N(t)]} \sum_{i=1}^{t} L_i(x) \quad ,$$

where $L_i(t) = L\big(\widehat{y_\iota(t)}, y_i(t)\big)$, is the total loss of the leader i at the round t. As we mentioned above, we assume that the loss is a convex function.

The pseudocode for the Follow the Leader algorithm is shown in figure 1.2.

```
**Algorithm** Follow the Leader [13]
**Initialization:** The expert's predictions $[x_1(t), \dots, x_N(t)]$.
                    For each expert $k$, $L_k(1) = 0$.
**for** t $= 1,2, \dots, n$ **do**:
    Make a prediction $\widehat{y_t} = argmin_{x \in [x_1(t), \dots, x_N(t)]} \sum_{i=1}^{t} L_i(x)$;
    Receive feedback, $y_t$;
**end for**
```

Figure 1.2 **Follow the Leader**

Evidently, this algorithm is a greedy algorithm. Similarly with the previous one, the goal is to design one that achieves a total expected loss not greater than the loss of the best leader.

As presented in [1], the Regret of Follow-the-Leader algorithm is bounded by the number of times the leader is overtaken by another expert. As a result, the value of Regret remains small as long as the best predictions are made by a single expert on average.

Accordingly, FTL works very well in scenarios where the losses are independent and identically distributed (i.i.d.), when due to the uniform law of large numbers, the leader is overtaken by another expert only a finite number of times. Yet, the algorithm has bad performance for worst-case data.

In the next paragraph, we will introduce the Partial Feedback and specifically a simple version of the k-armed Bandit problem, as it has been described in [10].

1.3 Partial Feedback – Bandit Problem

We have discussed until now the Full Information setting, where we were able to observe the loss of all the experts. In some problems this assumption is valid. Yet, in other cases this is not possible, and we can only observe the loss of the expert we actually choose. We will present in this section a very popular example of Partial Feedback problems, as presented in [10].

The Multiarmed Bandit problem is one where a gambler must decide among K non-identical slot machines, which to play in a sequence of trials so as to maximize his reward. In this example, at the end of each round, the gambler only knows the output of the slot he played and has no other feedback about the other slots that weren't selected.

A more general description of Partial Information games is the following: in every round, we have a choice among k different actions. After each round, we receive a reward based on the action we selected, but no other feedback about the rest of the available options. The objective is to maximize the expected total reward over a period of time.

In this set up you can see that we face a dilemma, since we receive no information about the accuracy of the actions unless we choose them. We need to balance our actions between exploiting the information we have received until now and exploring other actions that may be better. In the literature, this is referred to as the exploration versus exploitation dilemma.

So, in each round, each of the experts has an expected reward. This is the value of each action. Let's say that the action selected on round t is A_t and the equivalent reward R_t, so the expected reward for selecting the actions a is the conditional expectation:

$$\mathbb{E}[R_t|\, A_t = a]\,.$$

If we knew the value of each of the available actions, then our problem would be trivial as we would be able to always choose the action with the highest value. We assume that this is not the case. Instead, we have some estimations. Therefore, we denote that the estimated value of action a at the t^{th} round is $Q_t(a)$. The goal here is, that the $Q_t(a)$ would be as close as possible to the expected reward $\mathbb{E}[R_t|\, A_t = a]$.

During the game, we estimate these action values, and we do that by exploring. Consequently, at any given moment we know that at least one action excels the others, by having the greatest estimated value. These are greedy actions, and by selecting them we are exploiting the information we have until now. If instead, we select one of the nongreedy actions, say with a small probability, then we are exploring the other options. While exploiting may seem the right choice on each round, by exploration you can get a greater total reward in the end.

A straightforward way to estimate these action values is by average the rewards that were received until now, as presented in [10, Chapter 2].

Definition 4. [10, Chapter 2] At the t^{th} round, the estimated value of the action a is the sum of the rewards when a is taken prior to t, over the number of times a is taken prior to t.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \cdot 1_{A_i=a}}{\sum_{i=1}^{t-1} 1_{A_i=a}} .$$

When the denominator is zero, we set $Q_t(a)$ equal to a default value, like zero. And as it goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $\mathbb{E}[R_t|\, A_t = a]$. This is called the sample-average method of estimating the action values.

Obviously, this is just one way to estimate action values and a quite simple one. Consequently, based on the above definition, a greedy action rule will be:

$$A_t = argmax_a\big(Q_t(a)\big) .$$

Where the function $argmax_a$ results in the action a which we need to choose to get the max estimated value.

In the following paragraphs, we will describe one of the most popular and simple in implementation algorithm of Partial Information Bandit problems. The ε-Greedy algorithm, as presented in [10].

1.3.1 ε-Greedy Algorithm

As described above, Greedy actions always exploit the prior knowledge in order to maximize immediate reward and do not spend any time sampling other actions that may seem inferior at the time, but may be better in the long run.

A reasonable alternative to this is to behave greedily most of the time, but occasionally select randomly with equal probability among all the actions, independently of the estimated action value. These are the ε-Greedy methods. Below we present an ε-Greedy method for the Bandit problem, as analysed in [10, Chapter 2].

If we assume that we are using an ε-Greedy for action selection, as the rounds pass by, every action will sample an infinite number of times. As a result, by law of large numbers, the estimated value (**Definition 4**) $Q_t(a)$ will converge to the expected reward $\mathbb{E}[R_t | A_t = a]$. This is a big advantage of this method, since it directly indicates that the probability of selecting the optimal action converges to near certainty $(1-\varepsilon)$.

In order to be able to create a computational efficient ε-Greedy algorithm, we need to express the estimated value $Q_t(a)$ to an incremental formula that depends only on the previous step.

Defining first that Q_n is the estimate value of action a after it has been selected n times and R_n is the reward received after the action a has been selected n times, the new average of all n rewards can be computed by:

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i = Q_n + \frac{1}{n}[R_n - Q_n]\,.$$

The expression $[R_n - Q_n]$ is the error between the estimated value and the reward.

This formula requires memory only for Q_n and n, and only the small computation for each new reward.

The pseudocode for the bandit problem using ε-greedy action selection algorithm [10, Chapter 2] is shown in figure 1.3.

```
**Algorithm** ε-Greedy [10]
**Initialization:** For each action $k$, $Q(k) = 0$ and $N(k) = 0$.
**for** t = 1,2, … **do**:
    $A \leftarrow \begin{cases} argmax_a Q_t(a); & with\ probability\ 1-\varepsilon \\ a\ random\ action; & with\ probability\ \varepsilon \end{cases}$
    $R \leftarrow bandit(A);$
    $N(A) \leftarrow N(A) + 1;$
    $Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)];$
**end for**
```

Figure 1.3 **ε-Greedy**

One popular implementation of Online Learning is Online Auctions. In the next paragraph, we are going to introduce the Online Auctions and specifically the Double Auction or Bilateral Trade setup, presenting in detail the theory and the methods we are going to use in our experimentation.

1.4 Online Double Auctions

Online Auctions have been studied extensively in the literature. We will present the Double Auction problem in this section, as described in [11]. The algorithms, methods, and some theoretical results we are about to see here, are the ones that we will also use in our experiment. The results of our experimentation will be presented in section 3.

We consider the double auction of a specific product by an intermediary. The sellers and buyers are coming in pairs, in a random order, and they declare their price. Whenever a seller and a buyer arrive, we must decide whether we want to allow the transactions based on a threshold price.

Online Auction problems can be divided into Full Feedback or Partial Feedback based on the information we are receive after the end of every round. In the first case, we have available all the information about the seller's and buyer's values as well as the output of the interaction. In the second and more realistic case, we know the output of the interaction and the value we set up, but we do not have any information about the second party price. We will present below some methods that belong to the Full Information Expert Advice setup, followed by a Partial Information Bandit method. All of them will be used later in the experimentation phase.

Two metrics commonly used to assess the performance of a mechanism are Social Welfare and Gain from Trade. Both are presented below, as given in [11], [7].

For the intermediation setting, we consider that a set of binaries of buyer's and seller's bits arrives online in a random order at every round: $M(B,S) = \{(b_1, s_1), (b_2, s_2), \dots, (b_n, s_n)\}$, with $|B| = |S| = n$.

The intermediary has set a price $p \in B \cup S$. In every round we buy and sell one item from the sellers to buyers who accept our price, over n steps. So, the transactions that were successfully made are:

$$T(B,S) = \left\{(b,s) \in \left\{\{B\} \cup \{S\}\right\} \,\middle|\, \exists t \;\; b_t \geq p_t \;\; \wedge \;\; s_t \leq p_t \right\}.$$

Below we present the definition of Social Welfare as presented in [7], [11].

Definition 5. [7], [11] The Social Welfare of the online auction we describe, is the sum of the valuations of all agents with items. More specifically, all the sellers that did not sell their items and all the buyers that did buy an item.

$$W(B,S) = \mathbb{E}\left[\sum\nolimits_{s\in S\setminus T(B,S)} s + \sum\nolimits_{b\in T(B,S)} b\right],$$

or differently expressed: $W(p_t, s_t, b_t) = W_t(p_t) = s_t + (b_t - s_t)\mathbb{I}\{s_t \le p \le b_t\}$.

Below we present the second function we have for evaluating the performance: the Gain from Trade as presented in [7], [11].

Definition 6. [7], [11] The Gain from Trade of the online auction we described, is the difference between the final and the starting Welfare. In other words, the difference of the values of buyers and sellers for the transactions that were successfully made during the execution of our algorithm.

$$GFT(B,S) = \mathbb{E}\left[\sum\nolimits_{b\in T(B,S)} b - \sum\nolimits_{s\in T(B,S)} s\right],$$

or differently expressed: $GFT(p_t, s_t, b_t) = GFT_t(p_t) = (b_t - s_t)\mathbb{I}\{s_t \le p \le b_t\}$.

Note that, from the definitions given above, the $GFT(B,S) = W(B,S) - s_t$.

Lastly, we will also examine the performance of our algorithms based on Regret notion we present below.

Definition 7. [11] The Regret of the online auction we describe, is the difference between the expected total performance of our algorithm, and the best fixed price strategy assuming full knowledge of the distribution.

$$R_T = max_{p\in[0,1]} E\left[\sum_{t=1}^{T} GFT_t(p) - \sum_{t=1}^{T} GFT_t(p_t)\right].$$

Therefore, the goal is to design a mechanism that achieves asymptotically vanishing time-averaged Regret with respect to the best fixed-price strategy, or equivalently Regret sublinear in the time horizon T.

The first algorithm we will present belongs to the Full Feedback Experts' Advice setup and is the "Follow the Best Price" algorithm as presented in [11].

1.4.1 Follow the Best Price Algorithm

In [11], Nicolò Cesa-Bianchi, Tommaso R. Cesari, Roberto Colomboni, Federico Fusco, Stefano Leonardi (2021), present the Follow the Leader approach we saw in a previous section, which they call Follow the Best Price. As we mentioned this algorithm belongs to the Full Feedback methods. The "Follow the Best Price" algorithm [11] is shown below, in figure 1.4.

The seller/buyer pairs arriving are $(b_t, s_t) \in [0,1]^2$ independent and identically distributed (i.i.d.) random variables, without any further assumptions on their common distribution (B, S). They could even be arbitrarily correlated.

So, the algorithm begins by selecting a starting value for the price, arbitrarily. At step t, the algorithm presents the price p, then the pair (b_t, s_t) arrives and receives the result of the transaction. Lastly, it recalculates the value price, selecting the value $p \in \{S_1, B_1, \dots, S_t, B_t\}$ with the maximum Gain from Trade value until now. The update rule is presented below:

$$argmax_{p \in \{S_1, B_1, \dots, S_t, B_t\}} \sum_{i=1}^{t} GFT(x, s_i, b_i) .$$

```
Algorithm Follow the Best Price [11]
Initialization: The price value $\mathrm{p} = p_1$.
for $\mathrm{t} = 1,2, \dots, n$ do:
        Post the price value p;
        Receive feedback $(s_t, b_t)$ ;
        Compute $p \leftarrow argmax_{p \in \{S_1, B_1, \dots, S_t, B_t\}} \sum_{i=1}^{t} GFT(x, s_i, b_i)$ ;
end for
```

Figure 1.4 **Follow the Best Price**

Finally, a bound for Regret is presented below as a function of the number of mistakes of the best expert, as stated in [11].

In a Full-Feedback stochastic (iid) setting, the Regret of the Follow the Best Price algorithm, is bound of the following regardless of $T \in \mathbb{N}$:

$$R_T \leq c\sqrt{Tlog(T)}\ ,\ \text{where } c \leq 90.$$

The next algorithm we will present is the greedy sample or direct implementation of the dominant strategy action, for double auctions, as presented in [12].

1.4.2 McAfee's Double Auction Direct Implementation

In this scenario, we have an Offline Double Auction mechanism. So instead of pairs of sellers and buyers, we have a list of m buyers and n sellers. Each buyer i has a privately observed value b_{i} and each seller j has a privately observed value s_{j}.

Let $\{b_1, b_2, \dots, b_{\mathrm{m}}\}$ be the list of buyers' values and $\{s_1, s_2, \dots, s_{\mathrm{n}}\}$ be the list of sellers' values, we have available. We are ordering the buyers' values in descending order and the sellers' values in ascending order, as shown below:

$$b_{(1)} \geq b_{(2)} \geq \cdots \geq b_{(\mathrm{m})} \text{ and } s_{(1)} \leq s_{(2)} \leq \cdots \leq s_{(\mathrm{n})}\ .$$

Where the notation (i) is used for the ith highest valuation buyer or the ith lowest valuation seller.

Additionally, we are assuming that:

$$b_{(\mathrm{m}+1)} = \sup\{b : F(b) = 0\}\ , \quad s_{(\mathrm{n}+1)} = \inf\{s : F(s) = 1\}\ .$$

The efficient number of trades is the number $k \leq \min\{m, n\}$ where:

$$b_{(\mathrm{k})} \geq s_{(\mathrm{k})}\ \text{, and } b_{(\mathrm{k}+1)} < s_{(\mathrm{k}+1)}\ .$$

Finally, we define the trading price:

$$p_0 = \frac{1}{2}\left(b_{(\mathrm{k}+1)} + s_{(\mathrm{k}+1)}\right).$$

This method, as defined in [12], is called the Direct Implementation of the Dominant Strategy action and it always produces full information first best price. The pseudocode of the algorithm is shown in figure 1.5.

```
Algorithm McAfee's Direct Implementation [12]
Initialization: The buyers $[b_1, b_2, \dots, b_{\mathrm{m}}]$ and the sellers $[s_1, s_2, \dots, s_{\mathrm{n}}]$.
Order by descending the list of buyers $[b_1, b_2, \dots, b_{\mathrm{m}}]$
Order by ascending the list of sellers $[s_1, s_2, \dots, s_{\mathrm{n}}]$
for $\mathrm{t} = 2, \dots, \max(m, n)$ do:
        if $b_{t-1} \geq s_{t-1}$ $and$ $b_t < s_t$ do:
                Compute $p \leftarrow \frac{s_{t+1} - b_{t+1}}{2}$;
                break;
        end if
end for
```

Figure 1.5 **McAfee's Direct Implementation**

Later, we will use the algorithms and metrics described above in our experimentation section.

1.4.3 ε-Greedy Algorithm

In this paragraph we will present the ε-Greedy algorithm that has been introduced in paragraph 1.3.1. This time however, it will be studied in the Double Auction setup as we will use it in the experimental section of this work.

The seller/buyer pairs arriving $\{(b_1, s_1), (b_2, s_2), \dots, (b_n, s_n)\}$, are independent and identically distributed (i.i.d.) random variables, without any further assumptions on their common distribution (B, S).

As we saw in paragraph 1.3 in definition 5, the estimated value $Q_n(a)$ of the action a at the nth round, will be given by:

$$Q_{n+1}(a) = Q_n(a) + \frac{1}{n(a)} [R_n - Q_n(a)] .$$

Where $n(a)$ is the times the action a has been selected and the reward at the nth round is the Gain from Trade value $(b_n - s_n)\mathbb{I}\{s_n \leq p \leq b_n\}$.

Hence, in each round the algorithm decides if we will follow the greedy action, with probability $1 - \varepsilon$, receives the seller/buyer pair and calculates the reward value and lastly, updates the estimated value for the action selected, based on the updated rule described above. The pseudocode for ε-Greedy algorithm is shown in figure 1.6.

```
Algorithm ε-Greedy
Initialization: For each action $k$, $Q(k) = 0$ and $N(k) = 0$.
for t = 1,2, … do:
        $A \leftarrow \begin{cases} argmax_a Q_t(a); & with\ probability\ 1-\varepsilon \\ a\ random\ action; & with\ probability\ \varepsilon \end{cases}$

        Receive feedback $(s_t, b_t)$ ;
        if $b_t \geq p_A$ and $p_A \leq s_t$ do:
                $R \leftarrow b_t - s_t$;
        end if
        $N(A) \leftarrow N(A) + 1$;
        $Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$;
end for
```
Figure 1.6 **ε-Greedy**

1.4.4 Exponential Weighted Average Algorithm

In this paragraph we will present the Exponential Weighted Average method, which has been introduced in paragraph 1.2.2, but in the Double Auction setup as we will use it in the experimental section of this work.

In this algorithm we assume that the loss function is the Gain from Trade and will be used at the weight update rule.

The algorithm starts again with a list of uniform weights, $[w_1(1), \dots, w_N(1)]$. At every round t we receive a list with the expert's predictions $[x_1(t), \dots, x_N(t)]$ and their weights $[w_1(t), \dots, w_N(t)]$. And then, the algorithm's prediction is calculated by the following rule:

$$\widehat{y_t} = \frac{\sum_{i=1}^{N} w_i(t) x_i(t)}{\sum_{i=1}^{N} w_i(t)}$$

After calculating the prediction, we receive the feedback and update the weights accordingly with the following rule:

$$w_i(t+1) = w_i(t) e^{-\eta L_i(t)} ,$$

where $\eta_t = \sqrt{(8 \log N)/t}$ and $L \in [0,1]$.

For calculating the loss function, we follow the steps shown below:

1. Calculate loss for each expert $L_i(t) = \max(GFT(t)) - GFT_i(t)$
2. Normalize loss $L(t)$, since $L \in [0,1]$.

The seller/buyer pairs arriving $\{(b_1, s_1), (b_2, s_2), \dots, (b_n, s_n)\}$, are independent and identically distributed (i.i.d.) random variables, without any further assumptions on their common distribution (B, S).

The pseudocode for the Exponential Weighted Average algorithm, for Online Double Auction setting, is shown in figure 1.7.

```
**Algorithm** Exponential Weighted Average
**Parameters:** A constant $\eta > 0$
**Initialization:** A list with experts $X = [x_1, \dots, x_N] \in [0,1]^N$ and for each expert $k$, $w_k(1) = 1$.
**for** t = $1,2, \dots, T$ **do**:
    Calculate $\eta_t = \sqrt{(8 \log N)/t}$;
    Make a prediction for the price: $p_t \leftarrow \frac{\sum_{i=1}^{N} w_i(t) x_i(t)}{\sum_{i=1}^{N} w_i(t)}$;
    Receive feedback $(s_t, b_t) \in [0,1]$ ;
    **for** every expert k **do**:
        Calculate: $GFT_k(t) \leftarrow GFT_k(t) + b_t - s_t$ ;
    **end for**
    Calculate the loss: $L_k(t) = \max(GFT(t)) - GFT_k(t)$;
    Normalize loss $L(t)$;
    **for** every expert k **do**:
        Calculate the weight: $w_k(t+1) \leftarrow w_k(t) e^{-\eta L_k(t)}$ ;
    **end for**
**end for**
```

Figure 1.7 **Exponential Weighted Average**

2 Related work

Numerous papers have already investigated the subject of online auctions. Below we will see some papers that study the classic online auction for a single product setup, some that consider Full Feedback methods, others with Partial Feedback, others investigate the problem from the seller's perspective, others from the buyer's, and some with instrumentation. Lastly, some that are dealing with the Bilateral Trade Auctions that we are studying in this thesis and we will describe their findings and their individual approaches.

Some of these papers have investigated the subject of Online Auctions for a single product in unlimited resources. For instance, in their paper Blum, Kumar, Rudra, and Wu [3] investigate a Full Information Online Auction and a post price mechanism, where an auctioneer declares a price ahead and a bidder decides if he is going to accept it or not. In comparison to the standard online auction, the authors' mechanism offers much less information to the auctioneer about the bidder's valuation. The authors present, somewhat surprisingly, an asymptotically constant-competitive algorithm for both problems, using a weighted majority (WM) algorithm for their online auction and an Exp3 algorithm for the posted price set up.

In their research paper of Feng, Podimata, and Syrgkanis [4], the authors were also interested in the online auction of digital goods, but focusing on the fact that in many cases the bidder does not have a clear picture of the product's worth, or its worth changes over time. Their primary application is that of value-per-click sponsored search auctions. In this paper, the authors address the problem of "learning how to bid" in a Partial Information setup, in order to minimize the Regret with respect to the best-fixed price bid. Their algorithm, WIN-EXP (a variant of EXP3), outperforms the classical Bandit approach, and it is proven to hold even when they introduce noise in the bidder's feedback. The WIN-EXP algorithm achieves Regret $\tilde{O}\left(\sqrt{T|O|\log|B|}\right)$, better than the generic Multi-armed Bandit Regret of $\tilde{O}\left(\sqrt{T|B|}\right)$, where B is the set of possible action values (bids) and O the set of potential outcomes, since $|O|$ in most applications it will be a small constant.

One of the most recent studies on the Bilateral Trade problem is the one of Nicolò Cesa-Bianchi, Tommaso R. Cesari, Roberto Colomboni, Federico Fusco and Stefano Leonardi [11]. In their work they study a mechanism design problem to perform bilateral trades, in a Regret minimization setting. Their goal is to bound the total loss in a long period, by learning the important features of the prior distributions. In their analysis, they experiment with a stochastic and an adversarial setting with the objective to investigate how the bounds of the Regret change, depending on the quality of the feedback received. Their results show that under the stochastic setting (iid), for the Full-Feedback model, their algorithm achieves Regret of $\tilde{O}\left(T^{1/2}\right)$. For the Realistic-Feedback, assuming also that the valuations of the seller and the buyer are independent of each other and have bounded densities, their algorithm achieves Regret of $\tilde{O}\left(T^{2/3}\right)$. But if either they are not independent or they do not have bounded densities, no strategy can achieve sublinear worst-case regret. They also showed that these rates cannot be improved more than a $\log T$ factor. Lastly, they demonstrate that in an adversarial setting, no strategy can achieve sublinear worst-case regret, regardless of the feedback.

In their work, Jonathan Weed, Vianney Perchet, and Philippe Rigollet in [5], also consider the online advertising setting as a well-fitted problem to approach with the Online Learning algorithms. In this setup, the good that is in sale is the advertising space, the seller is the respected platform's owner, and the advertisers are the bidders. They consider a market where repeated Vickrey auctions take place and study this case from the bidder's perspective with the objective of maximizing the expected revenue generated by the ads. This setup can be expressed as a Bandit problem, or a Learning problem with Partial Feedback, with the goal to construct the optimal bidding strategy. They research the problem from two perspectives, from the Stochastic setting with a UCB-type of algorithm and from the Adversarial setting discretizing the possible scenarios to convert this to a classic bandit set up. In the stochastic approach, logarithmic regret $\tilde{O}(\log T)$ is possible and in the Adversarial approach, comparing the performance against that of the best fixed bid, a sublinear regret is also achievable.

Mehryar Mohri and Andres Munoz Medina in [6], also consider the online advertising setting of a Vickrey auction with a reserve, but in their study they focus on determining the optimal reserve price to maximize the revenue, from the publisher's perspective. The revenue depends on the reserve value that the publisher is setting. If this value is too low the winner may end up paying a very small price whereas if it is too high, it is possible that no bidder will bid high enough and the slot will be empty. Their approach is to use historical data from past auctions to predict the optimal reserve price, assuming full information with the objective of minimizing the expected value of the loss function. They do not make an assumption on the distribution of the bids but only that the outcome of each auction is independent and identically distributed allowing the bidders to be correlated. Finally, they present an algorithm that achieves $\tilde{O}(\mathrm{T}\log T)$.

There are more interesting studies and approaches than the ones we have described so far. We reference the reader to Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, Pavel Kolev, (2020) [2], research, where they consider Secretary and Online Bipartite Matching problems. The high-level idea is to incorporate some form of predictions in an existing online algorithm to get the best of two worlds.

3 Experiment of Online Double Auction with Intermediation

In the previous chapters, we talked about some algorithms of two different groups, Full Feedback and Partial Feedback. From the algorithms described so far, we will be using the algorithms in section 1.4 Online double auctions, for our experiments. Therefore, the algorithms used are defined in the table below.

Algorithms	Algorithm Names	Category
Algorithm 1	Follow the Best Price (FBP)	Full Feedback - Predicting from Expert Advice
Algorithm 2	Follow the Best Price algorithm with Greedy sample	Full Feedback - Predicting from Expert Advice
Algorithm 3	Exponential Weighted Average (EWA)	Full Feedback - Predicting from Expert Advice
Algorithm 4	ε-Greedy (ε=0.01)	Partial Feedback – Bandit Problem

Table 3.1 **Algorithms used in our experiments.**

For the ε-Greedy algorithm, we examine different possible values for the constant value ε and we choose 0.01 based on the results.

In the paragraph below we will present the datasets created for the experiments and the methodology we are going to use.

3.1 Experimentation Framework

The setting of our experiments is similar to the one described in section 1.4 Online Double Auctions.

We consider the double auction of a specific product by an intermediary. At every round, a set of binaries of buyer's and seller's bits, $M(B,S) = \{(b_1,s_1),(b_2,s_2),\dots,(b_n,s_n)\}$, with $|B| = |S| = n$,. The intermediary has set a price $p \in B \cup S$ and we buy and sell one item from the sellers to buyers who accept our price, over n steps. So, the transactions that were successfully made are:

$$T(B,S) = \{(b,s) \in \{\{B\} \cup \{S\}\} \mid \exists t \;\; b_t \geq p_t \;\; \wedge \;\; s_t \leq p_t \} .$$

In the following sections, we will make a comparison of the performance of the algorithms presented in table 3.1 and examine the differences. Our aim is to experiment with differentiated datasets and see how the above algorithms perform in each case. We will present the datasets used in the next paragraph.

3.1.1 Datasets used

As we have already mentioned, the seller/buyer pairs are independent and identically distributed (i.i.d.) random variables.

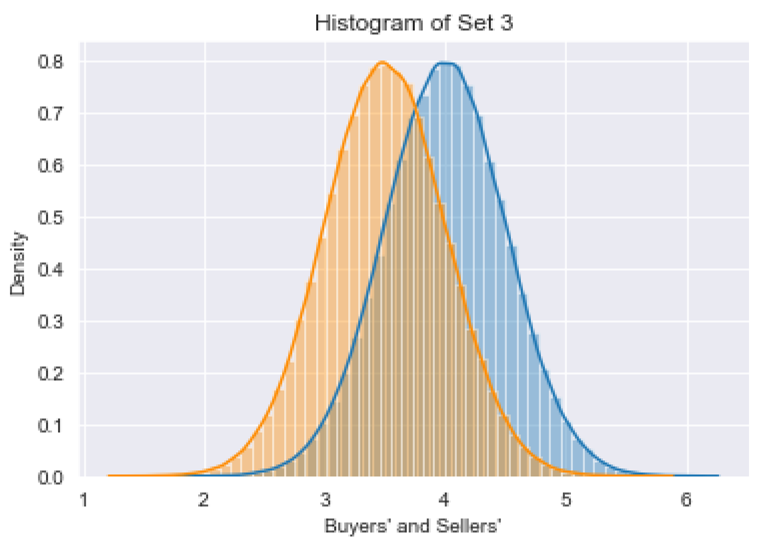
In table 3.2 we show the different datasets, for the values of the sellers, buyers, and the prices of the experts/bandits, used in our experiments. We also give the number of rounds T and the number of the experts or bandits, N.

We defined the price dataset (the experts/bandits) by taking the range between the $min(B \cup S)$ and the $max(B \cup S)$, with step k.

Set	Description	Distribution	T	N
Set 1	Sellers’ dataset	Uniform [0,1]	250000	-
	Buyers’ dataset	Uniform [0,1]	250000	-
	Price dataset	range(0, 1, 0.05)	-	20
Set 2	Sellers’ dataset	Normal (μ=3,σ=1/2)	250000	-
	Buyers’ dataset	Normal (μ=3,σ=1/2)	250000	-
	Price dataset	range(1, 4, 0.1)	-	30
Set 3	Sellers’ dataset	Normal (μ=4,σ=1/2)	250000	-
	Buyers’ dataset	Normal (μ=3.5,σ=1/2	250000	-
	Price dataset	range(2, 5, 0.1)	-	30
Set 4	Sellers’ dataset	Poisson (5)	250000	-
	Buyers’ dataset	Poisson (3)	250000	-
	Price dataset	range(0, 17, 1)	-	17

Table 3.2 **Types of datasets we use in our experiment.**

For the two last sets, we didn’t use the same distribution for both the sellers and the buyers. Instead, the sellers’ distribution is shifted to the right compared to the buyers’, based on the assumption that sellers desire to sell at a higher price and buyers to buy at a lower. In figure 3.1 we can observe the histograms, of set 3 and set 4.



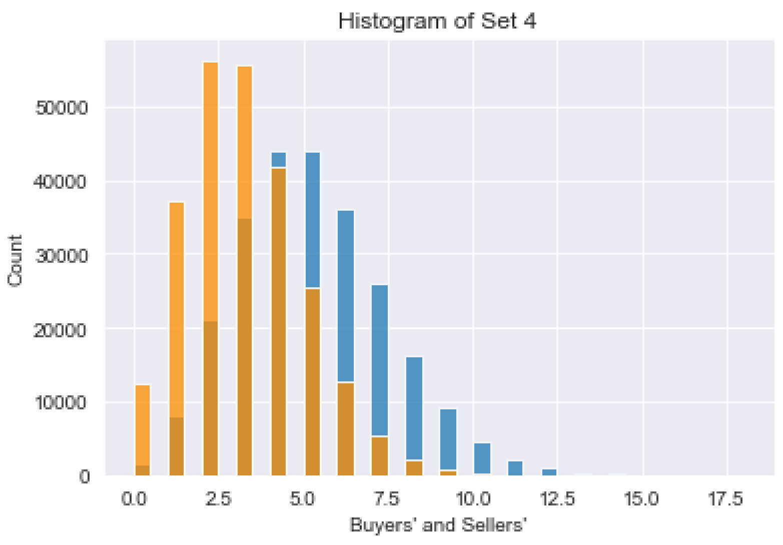


Figure 3.1 **Buyers' and Sellers' distribution for Set 3 and Set 4.**

3.1.2 Experimental phase

In this section, we present the aspects of the experimental phase. All the experiments were implemented in Python. As shown above, in Table 3.2, we run the experimentations for mutable iteration (250000). For each set of datasets, we examine all the algorithms shown in Table 3.1. and we observe the performance of the methods based on the following metrics:

- Gain from Trade
- Social Welfare
- Regrate

Particularly, for the **Follow the Best Price with Greedy sample** algorithm which we will see below, we need to specify the experimental phases, since it is a combination of two algorithms we described in the previous chapters. The idea is to use **McAfee's Double Auction Implementation** algorithm, presented in paragraph 1.4.2, to sample the first n rounds and use the "optimum" value it returns as the initial value of the price for the **Follow the Best Price** algorithm, presented in paragraph 1.4.1.

We call this the **Follow the Best Price with Greedy sample** algorithm and experimental phases are the following:

1. We sample the 0.01 of the whole number of rounds 250000. Hence, we run the **McAfee's Double Auction Implementation** algorithm for 2500 sets of binaries of buyers and sellers.
2. Then we use the output of the above algorithm, to determine the initial value of the price for the **Follow the Best Price**.
3. Combined with the above, we run the **Follow the Best Price**, for the rest, 247500 rounds.

Continuing with our experiment, in the next paragraphs we present the results of each dataset for all the algorithms and finally we summarize by comparing the performance of the algorithms.

3.2 Set 1: Uniform [0,1]

For the first experiment, we will be using the Uniform [0,1] distribution to create the sets of buyers' and sellers' values. We will be running the experiment for 250000 rounds; and we will compare the performance of the algorithms presented above, with each other and with the best expert. The results are shown in Table 3.3 below.

Compared with the best expert, we see that the Gain from Trade value of our algorithm is always less yet very close to that of the best offline game.

The value of the Price of the best offline game is ~ 0.5 , and that is also the mean value of the Price for all our algorithms. Even so, the **EWA** and **ε-Greedy** algorithms have the worst value of Regret while the **FBP** has the best. We can also point out that the **FBP with Greedy sample** algorithm has not improved the results from the classic **FBP** algorithm.

Set	Metrix	FBP	FBP with Greedy	EWA	ε-Greedy
Set 1	**Best expert**	**0,500**	**0,500**	**0,500**	**0,500**
	Best expert GFT	**31187,106**	**31187,106**	**31187,106**	**31187,106**
	Mean Price	0,499	0,499	0,498	0,595
	GFT	31185,288	31184,740	31158,294	30551,956
	Social Welfare	31185,104	31184,556	31158,110	30565,240
	Regret	1,818	2,366	28,812	590,623

Table 3.3 **Algorithms' results for Set 1: Uniform [0,1] where T=250000**

In the plot below we can observe the correlation between the value of the Price and the Gain from Trade value on the offline game. It is obvious that the best expert has Price ~ 0.5.

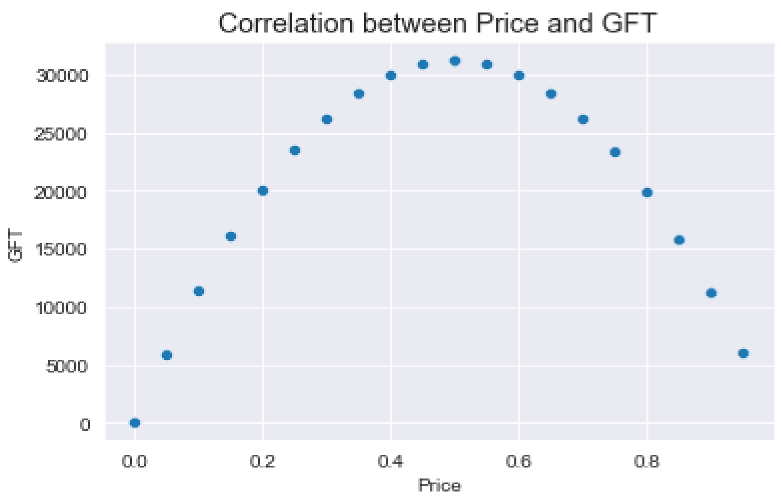


Figure 3.2 **Correlation between GFT and the Price for Set 1: Uniform [0,1], on the offline game.**

Below we present some graphs with the Regret and Gain from Trade metric during time (T), along with the Price variation, filter for the first n rounds. For the Price value, it is noticeable that the **ε-Greedy** algorithm explores more, since all the other algorithms after the first rounds land on the 0.5 and do not change for the rest of the rounds. Additional to that, the **ε-Greedy** algorithm needed more time to approach the optimal value (0.5). These findings explain the bad results the algorithm had to the metrics above.

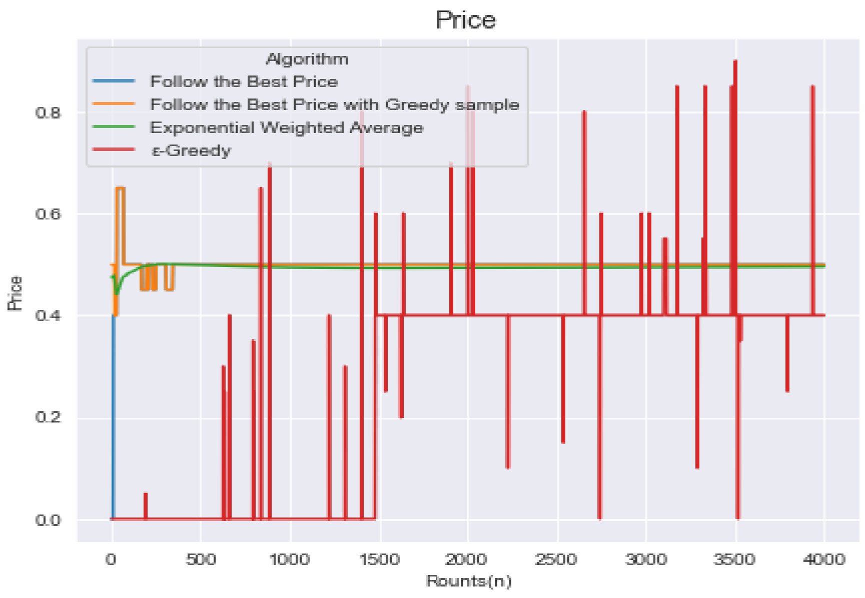


Figure 3.3 **Price variation for Set 1 for all algorithms (4000 rounds)**

In the following plot, we can see how Gain from Trade changes over time. We notice that the GFT for all the algorithms, has a linear relationship to time, approximately $\frac{1}{8}\,T$. Also, the Welfare graph has a similar picture that we choose not to present here for this reason.

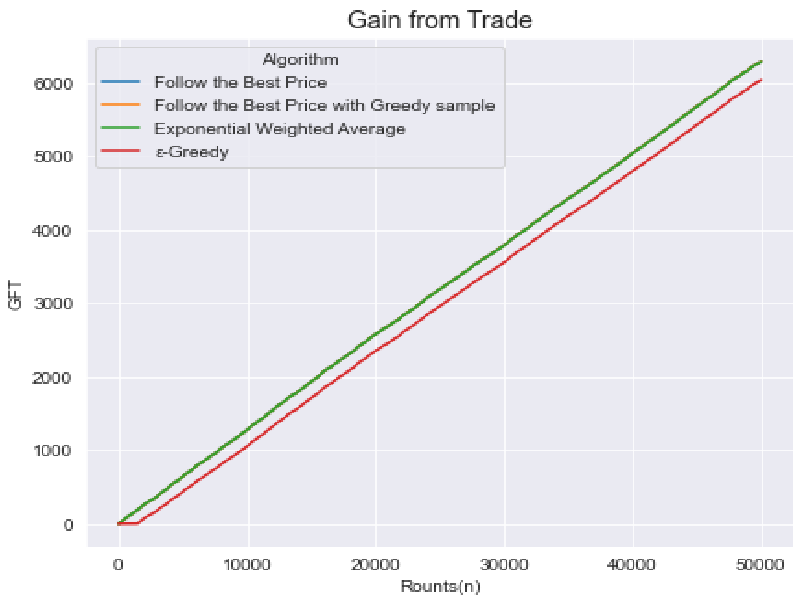


Figure 3.4 **Gain from Trade for Set 1 for all algorithms (50000 rounds)**

Lastly in the following plot, we can see the Regret over time and we notice that it gets asymptotical to the x-axis after the first couple of rounds for all the algorithms except the **ε-Greedy** algorithm which has an upward trend.

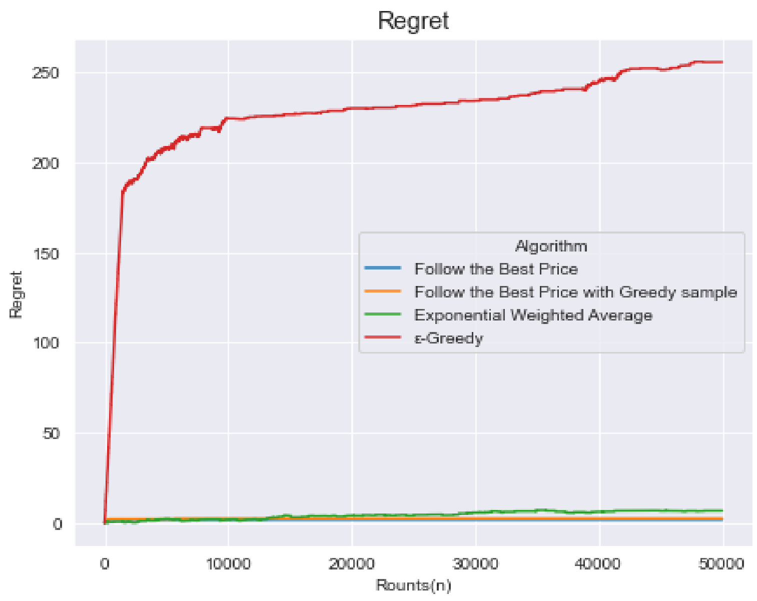


Figure 3.5 **Regret for Set 1 for all algorithms (50000 rounds)**

Uniform distribution was our first attempt, but it is an unrealistic distribution with a large variance in the values, we will use below the normal distribution and observe our algorithms' performance. Figure A.1 in the appendix shows the first hundred rounds of Price and Regret.

3.3 Set 2: Normal (μ=3, σ=1/2)

In this section we continue with set 2, using the Normal (μ=3, σ=1/2) distribution to create the buyers' and sellers' values. Again, we run the experiment for 250000 rounds; and we will present the performance metrics for each algorithm in table 3.4 below.

Compared to the best expert, we see again that the Gain from Trade value of our algorithm is always less than that of the best offline game.

The value of the Price of the best offline game is ∼3 , and the mean value of the Price for all our algorithms is also very close. Even so, the **EWA** and **ε-Greedy** algorithms again have the worst values of Regret, and the **FBP** has the best. We can also point out that the **FBP with Greedy sample** algorithm does not have improved results, compared to the classic **FBP** algorithm.

Set	Metrix	FBP	FBP with Greedy	EWA	ε-Greedy
Set 2	**Best expert**	**3,0**	**3,0**	**3,0**	**3,0**
	Best expert GFT	**50017,261**	**50017,261**	**50017,261**	**50017,261**
	Mean Price	2,999	2,999	3,007	2,792
	GFT	50013,098	50017,261	49986,760	45754,578
	Social Welfare	50010,378	50009,547	49984,039	48116,370
	Regret	4,162	4,993	30,501	4262,683

Table 3.4 **Algorithms' results for Set 2: Normal (μ=3, σ=1/2) where T=250000**

In the plot below we can observe, the correlation between the value of Price and the Gain from Trade value on the offline game. It is obvious that the best expert has Price ~3 and that the line is shifted to the right.

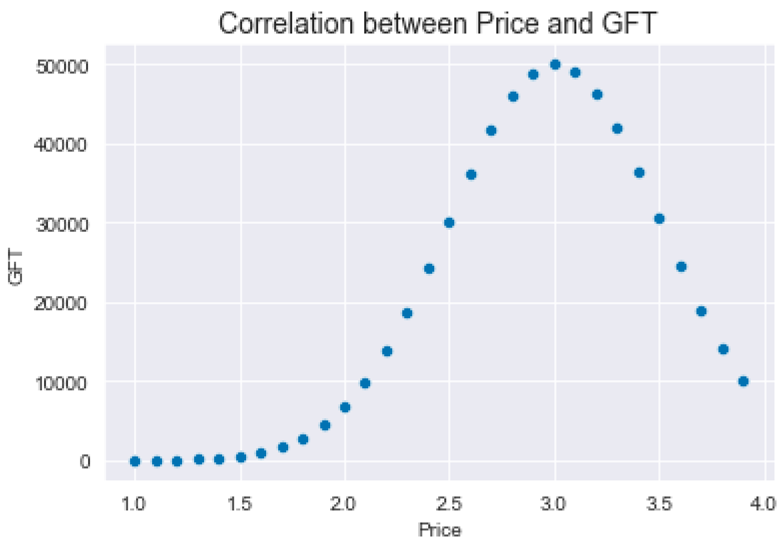


Figure 3.6 **Correlation between GFT and the Price for Set 2: Normal (μ=3, σ=1/2), on the offline game.**

For the Price value, we can see again that the **ε-Greedy** algorithm explores more and needs more time to approach the optimal value. The other algorithms after the first rounds land on the value 3 and do not explore further.

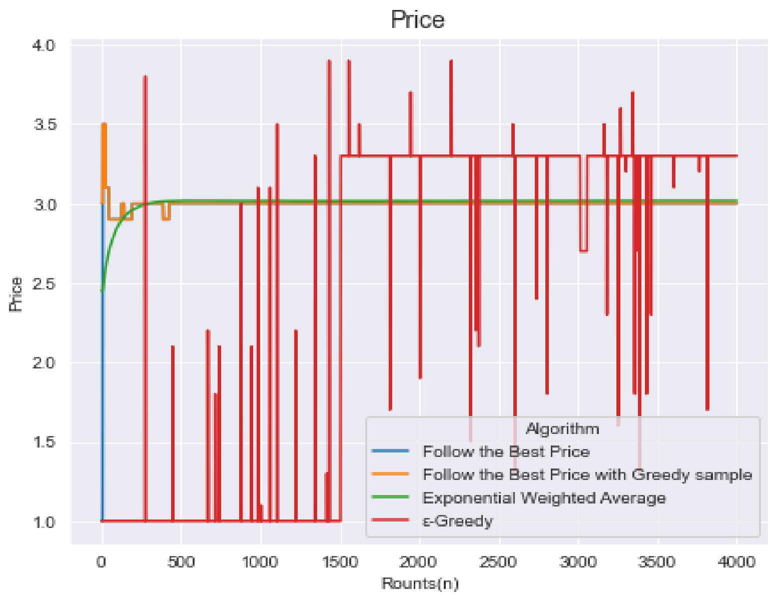


Figure 3.7 **Price variation for Set 2 for all algorithms (4000 rounds)**

In the following plot, we can see how Gain from Trade changes over time. We notice that the GFT for all the algorithms, has a linear relationship to time, approximately $\frac{1}{5}T$. We notice that the GFT of the **ε-Greedy** algorithm has the late start we observe also on the Price plot. For the first hundred rounds its stays at zero and then increases at a smaller rate than the rest of the algorithms.

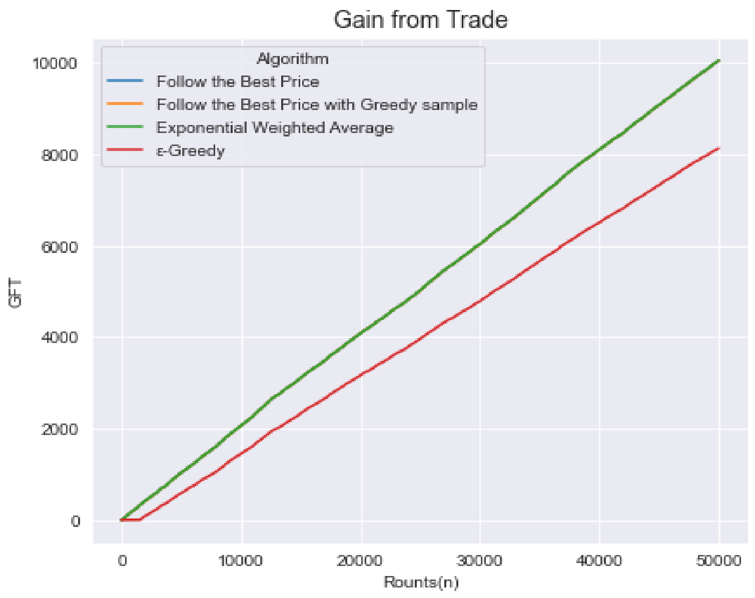


Figure 3.8 **Gain from Trade for Set 2 for all algorithms (50000 rounds)**

As expected from the Gain from Trade results, the Regret for most of the algorithms gets asymptotical to the x-axis over time, except for the ε-Greedy algorithm which has a clear upward trend.

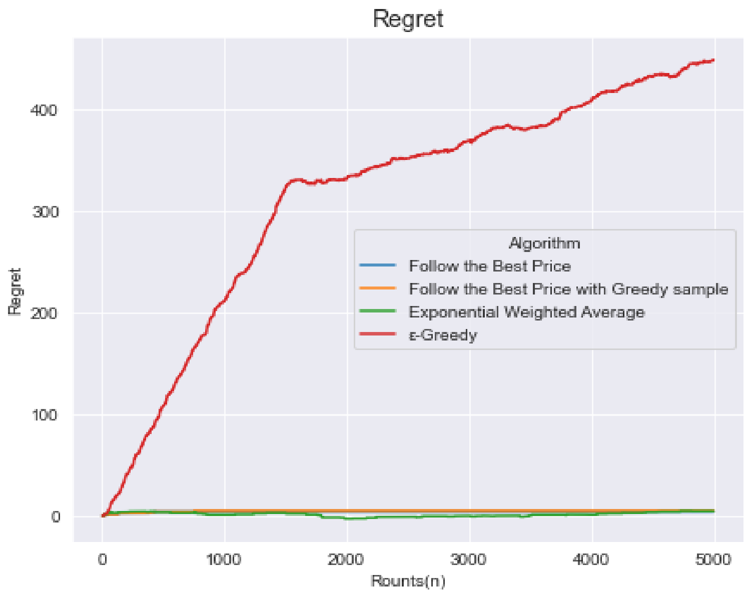


Figure 3.9 **Regret for Set 2 for all algorithms (5000 rounds)**

In the next set, we choose to have two different normal distributions for the sellers' and buyers' values. As we explain in an earlier paragraph, this scenario seems more realistic based on the assumption that the sellers aim is to sell at a higher price and the buyers to buy at a lower. In the appendix, Figure A.2 shows the first hundred rounds of Price and Regret.

3.4 Set 3: Normal sellers (μ=4, σ=1/2) buyers (μ=3.5, σ=1/2)

For this section we choose to use the Normal (μ=4, σ=1/2) for the sellers' values and Normal (μ=3.5, σ=1/2) for the buyers. Again, we run the experiment for 250000 rounds; and we present the performance metrics for each algorithm in table 3.5 below.

Unlike previous experiments, comparing the Gain from Trade results of the best experts versus our algorithms, we see that **EWA** algorithm performs better achieving a negative Regret. Furthermore, this time **FBP with Greedy sample** algorithm outperforms the classic **FBP** algorithm succeeding smaller Regret value.

Set	Metrix	FBP	FBP with Greedy	EWA	ε-Greedy
Set 3	**Best expert**	**3,7**	**3,8**	**3,7**	**3,7**
	Best expert GFT	**15256,184**	**15335,418**	**15256,184**	**15256,184**
	Mean Price	3,744	3,758	3,741	3,671
	GFT	15236,448	15325,981	15350,34	13239,639
	Social Welfare	15232,727	15322,260	15346,62	15075,517
	Regret	19,736	9,437	-94,156	2016,545

Table 3.5 **Algorithms' results for Set 3: Normal sellers (μ=4, σ=1/2) buyers (μ=3.5, σ=1/2) where T=250000**

In the plot below we can observe, the correlation between the value of Price and the Gain from Trade value on the offline game. It is obvious that the best expert has Price ~3 and that the line is shifted to the right.

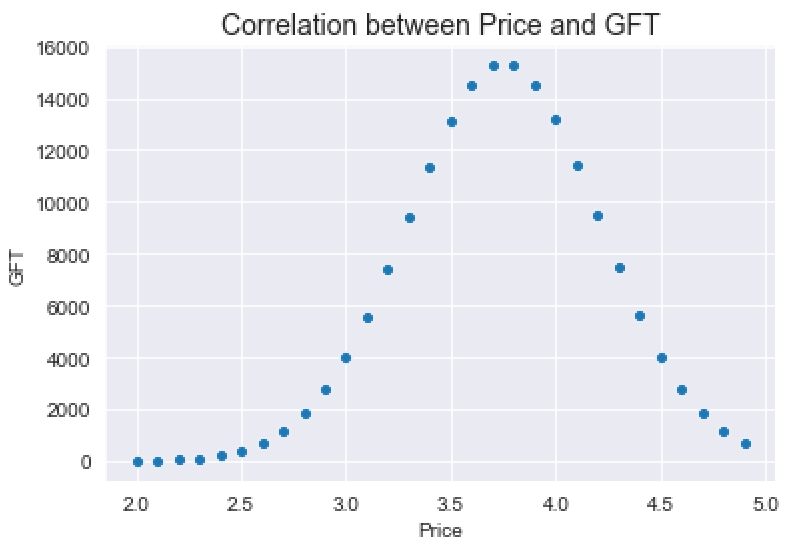


Figure 3.10 **Correlation between GFT and the Price for Set 3: Normal sellers (μ=4, σ=1/2) buyers (μ=3.5, σ=1/2), on the offline game.**

Looking at the Price graph we see that the **ε-Greedy** algorithm again does the most exploration. The other algorithms after the first rounds land on the optimal value ~3.7.

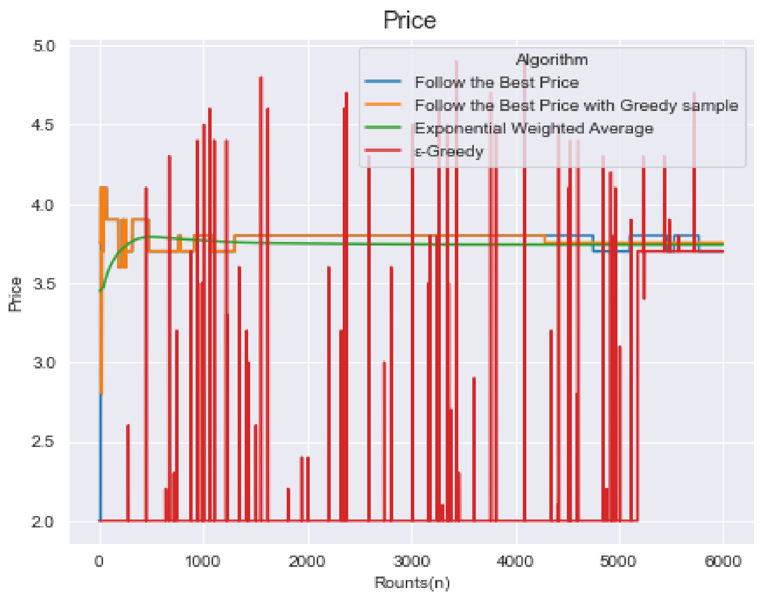


Figure 3.11 **Price variation for Set 3 for all algorithms (6000 rounds)**

From the Gain from Trade plot, we notice that the GFT, for all the algorithms, has a linear relationship to time, approximately $\sim \frac{1}{17} T$. We notice that the GFT of the **ε-Greedy** algorithm has the late start we observe also on the Price plot. The first thousand rounds stay at zero and then increase at a smaller rate than the rest of the algorithms.

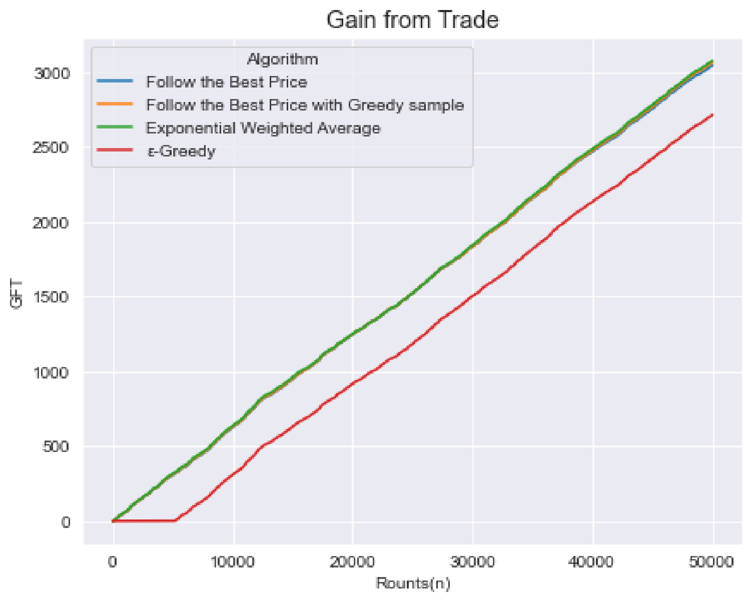


Figure 3.12 **Gain from Trade for Set 3 for all algorithms (50000 rounds)**

As expected from the Gain from Trade results, the Regret for most of the algorithms gets asymptotical to the x-axis over time, except for the **ε-Greedy** algorithm which has a clear upward trend before he also gets asymptotical.

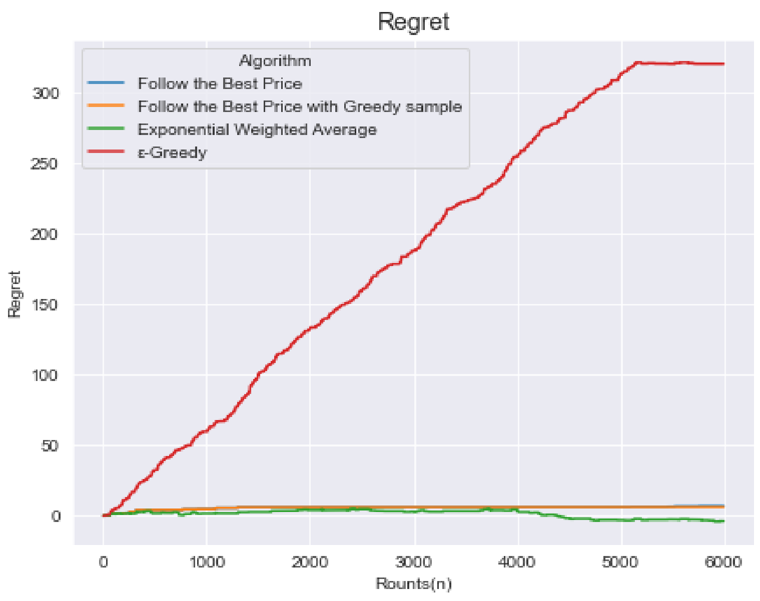


Figure 3.13 **Regret for Set 3 for all algorithms (6000 rounds)**

In the next set, we choose to have two different Poisson distributions for the sellers' and buyers' values to examine the performance of our algorithms in a discrete distribution. In the appendix, Figure A.3 shows the first hundred rounds of Price and Regret.

3.5 Set 4: Poisson sellers (5) buyers (3)

Unlike previous experiments, for this section we choose a discrete distribution, to be exact, a Poisson (5) for the sellers' values and Poisson (3) for the buyers. Again, we run the experiment for 250,000 rounds and we will present the performance metrics for each algorithm in Table 3.6 below.

In this setup, we examine the Regret results of our algorithms and see that the **FBP with Greedy sampling** algorithm has the best outcome, with a close second to be the **FBP** algorithm whereas **ε-Greedy** algorithm performs the purest. Compared with the optimum bandit, we see again that the Gain from Trade value of our algorithm is always less than that of the best offline game.

Set	Metrix	FBP	FBP with Greedy	EWA	ε-Greedy
Set 4	Best bandit	4,0	4,0	4,0	4,0
	Best bandit GFT	73372,0	73372,0	73372,0	73372,0
	Mean Price	3,9	3,9	4,0	3,9
	GFT	73370,0	73371,0	73357,0	71173,0
	Social Welfare	73369,0	73370,0	73356,0	71172,0
	Regret	2,0	1,0	15,0	2199,0

Table 3.6 **Algorithms' results for Set 3: Poisson sellers (5) buyers (3) where T=250000**

In the plot below we can observe the correlation between the value of Price and the Gain from Trade value on the offline game. It is obvious that the best expert has Price ~ 4 and that the curve is shifted to the left and is sharply peaked.

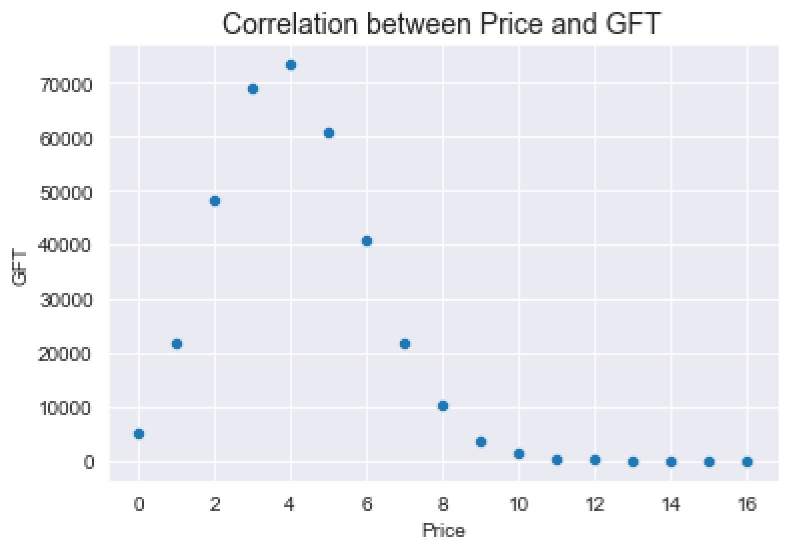


Figure 3.14 **Correlation between GFT and the Price for Set 4: Poisson sellers (5) buyers (3), on the offline game.**

Looking at the Price graph we see that the **ε-Greedy** algorithm again does more exploration. The other algorithms after the first rounds land on the optimal value 4 and do not vary from this for the rest of the rounds.

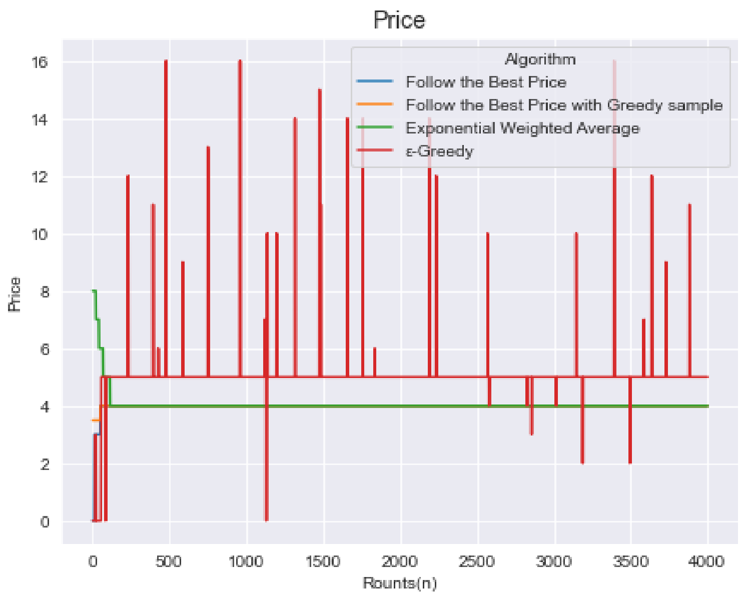


Figure 3.15 **Price variation for Set 4 for all algorithms (4000 rounds)**

From the Gain from Trade Figure 3.16, we notice that the GFT, for all the algorithms, has a linear relationship to time, approximately $\sim\frac{1}{3}\,T$, except for **ε-Greedy** which has $\sim\frac{1}{4}\,T$. We notice that the GFT of the **ε-Greedy** algorithm does not have the late start we observe in the previous sets.

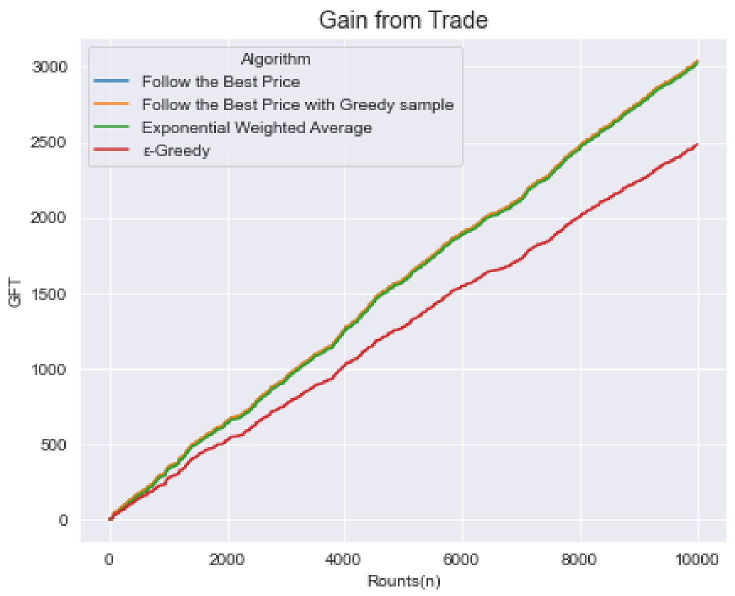


Figure 3.16 **Gain from Trade for Set 4 for all algorithms (10000 rounds)**

As expected the Regret -for most algorithms- gets asymptotical to the x-axis over time, except for the **ε-Greedy** algorithm which has a clear upward trend.

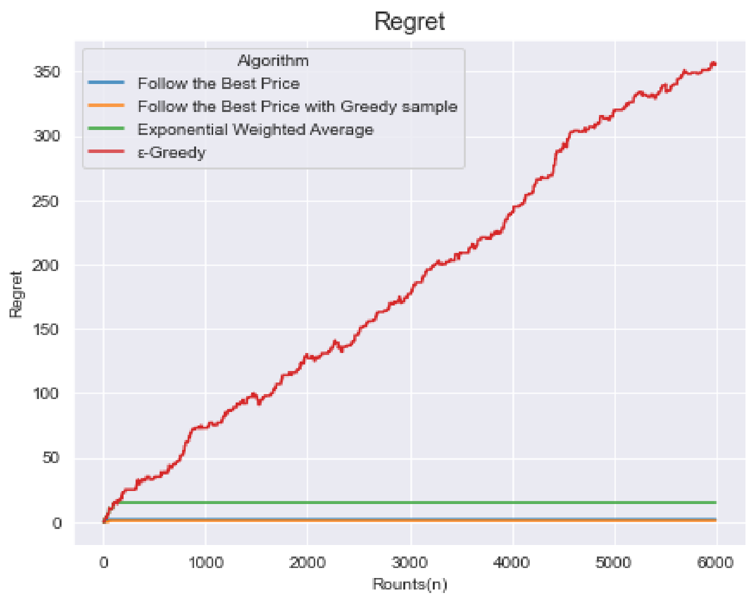


Figure 3.17 **Regret for Set 4 for all algorithms (6000 rounds)**

Figure A.4 in the appendix shows the first hundred rounds of Price and Regret.

3.6 Summary

In every case, looking at the experiments above, we manage to achieve Gain from Trade values close to the offline game most of the time, something that is supported by the theoretical results presented in previous chapters.

From the *"Correlation between GFT and the Price"* graphs presented in the previous paragraphs, it is safe to assume that the algorithm approaching the "optimal" value of the Price faster, is the one that will perform the best. Observing the Figures A.1-4, in the appendix paragraph at the end of our work, we see that even though the **Follow the Leader** algorithm and the modified **Follow the Leader with Greedy sample** have different begging values, they end up making the same prediction for the Price after a couple of rounds. It is also evident that all algorithms approach the "optimal value" quite fast, except for the **ε-Greedy** algorithm.

The "simple" **Follow the best price** algorithm is the best performing algorithm -Table 3.7- in the two first sets but not in the last two where the sellers' and buyers' distribution is not exactly the same. The **ε-Greedy** (with ε=0.01) algorithm has the worst results, since it is the only one from the Partial Information category.

Set	Best Algorithm
Set 1: Uniform [0,1]	Follow the Best Price
Set 2: Normal (μ=3, σ=1/2)	Follow the Best Price
Set 3: Normal sellers (μ=4, σ=1/2) buyers (μ=3.5, σ=1/2)	Exponential Weighted Average
Set 4: Poisson sellers (5) buyers (3)	Follow the Best Price with Greedy sample

Table 3.7 **Summarize best performing algorithms for each set based on Regret.**

Also, considering the results of the previous paragraphs, the modified **Follow the Best Price with Greedy sample** algorithm outperformed the original **Follow the Best Price** algorithm, only in the last two sets.

Looking at more detailed results of Regret and GFT, in the Table 3.8, most of the algorithms perform very well compared to the optimal offline result. We have Regret near zero and asymptotical to the time for all the algorithms except the **ε-Greedy**, which for most of the sets was getting asymptotical to time with an upward trend.

More precisely, the two cases that do not follow the above behavior for the Regret, are the **Set 4: Poisson sellers (5) buyers (3)**, where **ε-Greedy** seems to have a clear upward trend and **Set 3: Normal sellers (μ=4, σ=1/2) buyers (μ=3.5, σ=1/2)**, where the **Exponential Weighted Average** algorithm manages to achieve a negative Regret close to zero and outperform the offline optimal results.

Set	Metrix	FBP	FBP with Greedy sample	EWA	ε-Greedy
Set 1	**GFT results**	$\sim 1/8 \times T$			
	Regret	$\sim 0^+$			asymptotical to time
Set 2	**GFT results**	$\sim 1/5 \times T$			
	Regret	$\sim 0^+$			asymptotical to time
Set 3	**GFT results**	$\sim 1/17 \times T$			
	Regret	$\sim 0^+$		$\sim 0^-$	asymptotical to time
Set 4	**GFT results**	$\sim 1/3 \times T$			$\sim 1/4 \times T$
	Regret	$\sim 0^+$			upward trend

Table 3.8 **Summarize algorithms' results for all algorithms and sets, GFT and Regret.**

Lastly examining the results of the Gain from Trade value, from Table 3.8 and the plots presented in the previous paragraph, we see that all algorithms have a linear relation to time except the **ε-Greedy** algorithm which in most of the sets, has a late start in the first hundred rounds but after that it also gets linear to the time.

4 Conclusions

To summarize, this work explores the Online Learning Problems with Intermediation, giving a particular emphasis on the Double Auction, or Bilateral Trade. We start with the Full Feedback and the Expert Advice methods, followed by the Partial Feedback and the Bandit general setting. We studied bibliographically the algorithms and some theoretical results, that treat the above problems. Next, we present the problem of Online Auctions and more particular the Double Auction setting, we selected for our experimentation section.

We examined the algorithms that were presented on the Online Learning problems. Initially, we selected two Full-Feedback algorithms to analyze, the **Follow the Best Price** algorithm, a modified **Follow the Best Price with Greedy sampling**, and the **Exponential Weighted Average** algorithm. The analysis continued with an **ε-Greedy** algorithm from the Partial-Feedback category of algorithms.

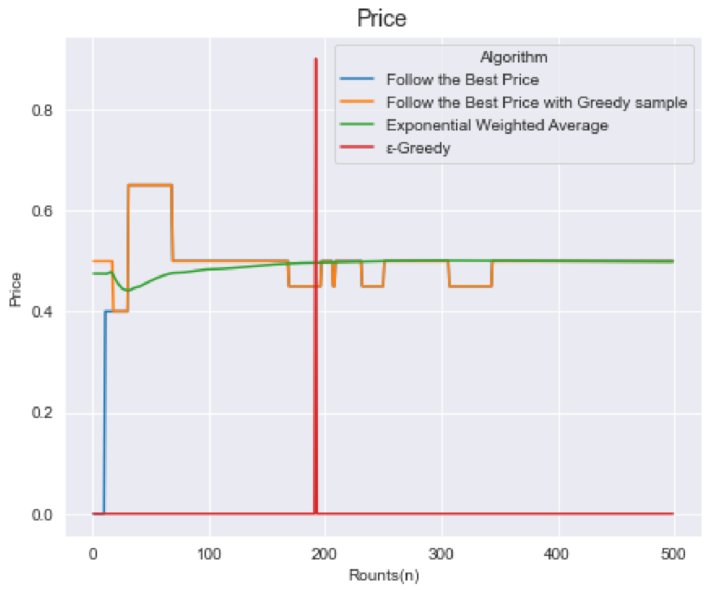
We compared the performance of the above algorithms in the Bilateral Trade set up, selecting four different sets for the sellers' and the buyers' prices. Our findings verified the theoretical results. All the algorithms tested achieved small (logarithmic) Regrets except for the **ε-Greedy**, in Set 4: Poisson sellers (5) buyers (3), that suffered a linear Regret.

More specifically, the **Follow the best price** algorithm is the best performing algorithm in the two first sets but not in the last two where the sellers' and buyers' distribution is not the same. The **ε-Greedy** algorithm has the worst results in the experiments. Our modified **Follow the Best Price with Greedy sample** algorithm achieved the bests results, only in Set 4: Poisson sellers (5) buyers (3).

5 Bibliography

1. S. de Rooij, T. van Erven, P. D. Grünwald, W. M. Koolen: Follow the leader if you can, hedge if you must. Journal of Machine Learning Research 15(1): 1281-1316 (2014)
2. A. Antoniadis, T. Gouleakis, P. Kleer, P. Kolev: Secretary and Online Matching Problems with Machine Learned Advice. Annual Conference on Neural Information Processing Systems 2020, NeurIPS (2020)
3. A. Blum, V. Kumar, A. Rudra, F. Wu: Online learning in online auctions. Theoretical Computer Science 324(2-3): 137-146 (2004)
4. Z. Feng, C. Podimata, V. Syrgkanis: Learning to Bid Without Knowing your Value. Proceedings of the 2018 ACM Conference on Economics and Computation (2018): 505-522
5. J. Weed, V. Perchet, P. Rigollet: Online learning in repeated auctions. Proceedings of the 29th Conference on Learning Theory (2015): 1562-1583
6. M. Mohri, A. M. Medina: Learning Algorithms for Second-Price Auctions with Reserve. Journal of Machine Learning Research 17: 74:1-74:25 (2016)
7. E. Koutsoupias, P. Lazos: Online Trading as a Secretary Problem. Algorithmic Game Theory - 11th International Symposium 2018: 201-212
8. S. Arora, E. Hazan, S. Kale: The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. Theory of Computing 8(1): 121-164 (2012)
9. M. Mohri, A. Rostamizadeh, A. Talwalkar: Foundations of Machine Learning. Adaptive computation and machine learning, MIT Press 2012
10. J. D. Johnson, J. Li, Z. Chen: Reinforcement Learning: An Introduction: R.S. Sutton, A.G. Barto, MIT Press, Cambridge. (2000)
11. N. Cesa-Bianchi, T. R. Cesari, R. Colomboni, F. Fusco, S. Leonardi: A Regret Analysis of Bilateral Trade. EC '21: The 22nd ACM Conference on Economics and Computation 2021: 289-309
12. R. P. McAfee: A Dominant Strategy Double Auction. Journal of Economic Theory, vol. 56, no. 2, 1992, pp. 434–450
13. S. Shalev-Shwartz: Online Learning and Online Convex Optimization. Foundations and Trends in Machine Learning 4(2): 107-194 (2012)
14. https://www.datanyze.com/market-share/ad-exchanges--399

6 Appendix – Additional results



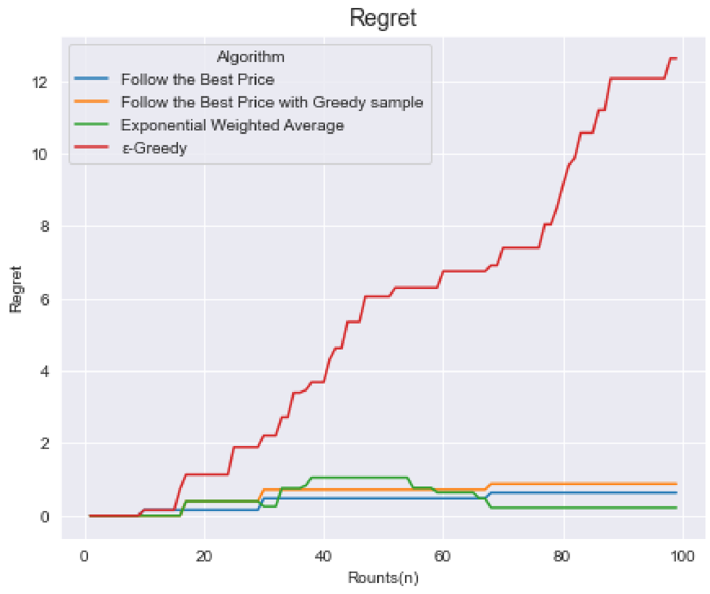
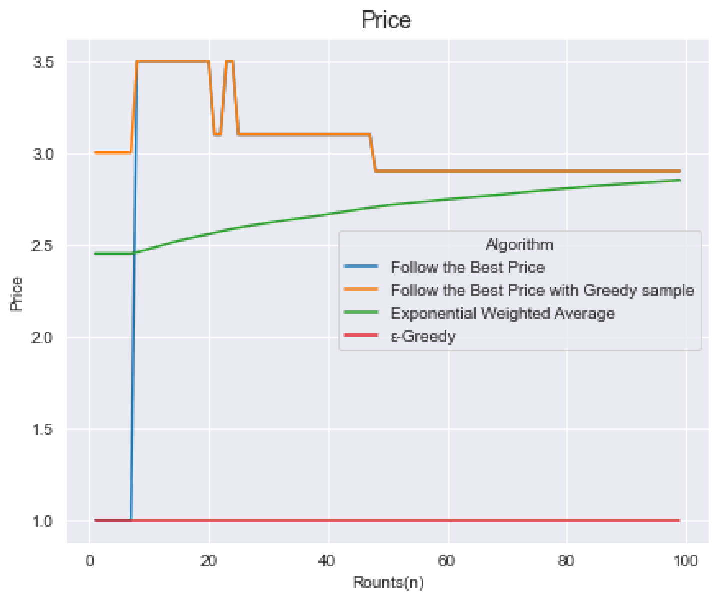


Figure A.1 **Price and Regret variation for Set 1 for all algorithms (100 rounds)**



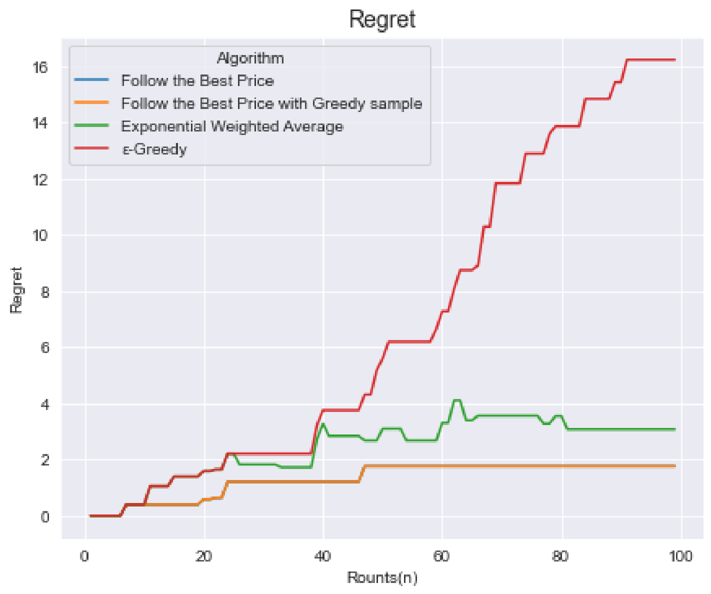
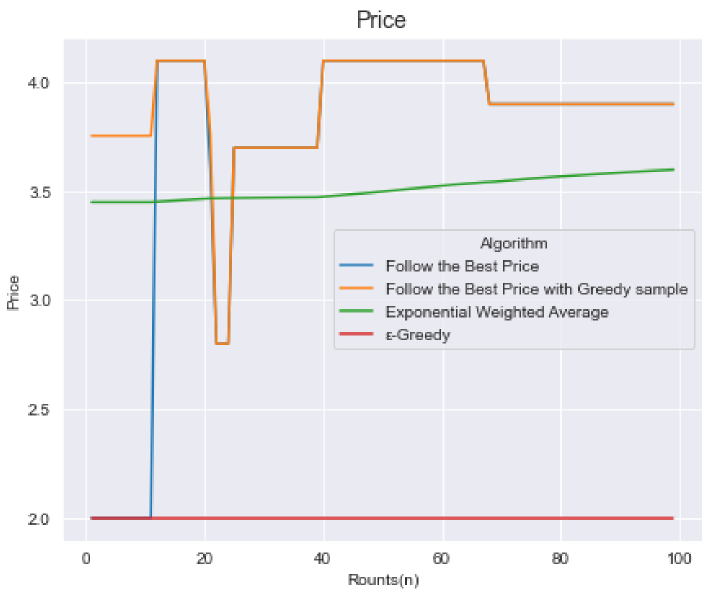


Figure A.2 **Price and Regret variation for Set 2 for all algorithms (100 rounds)**



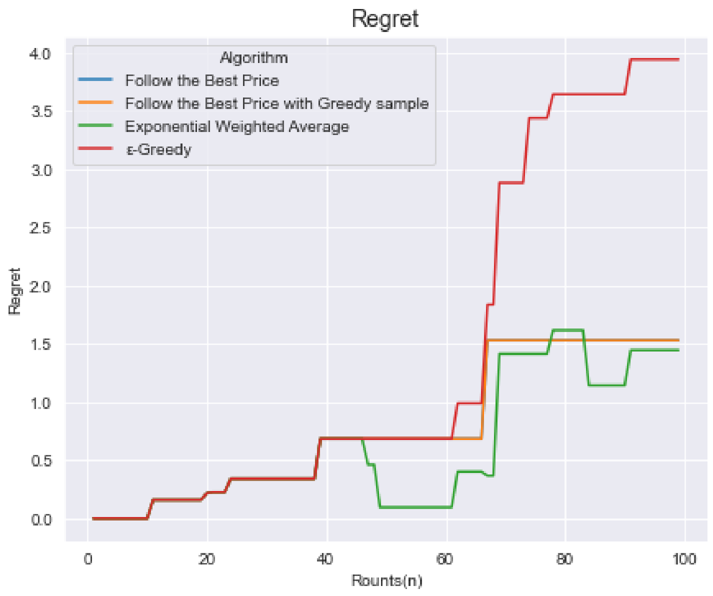
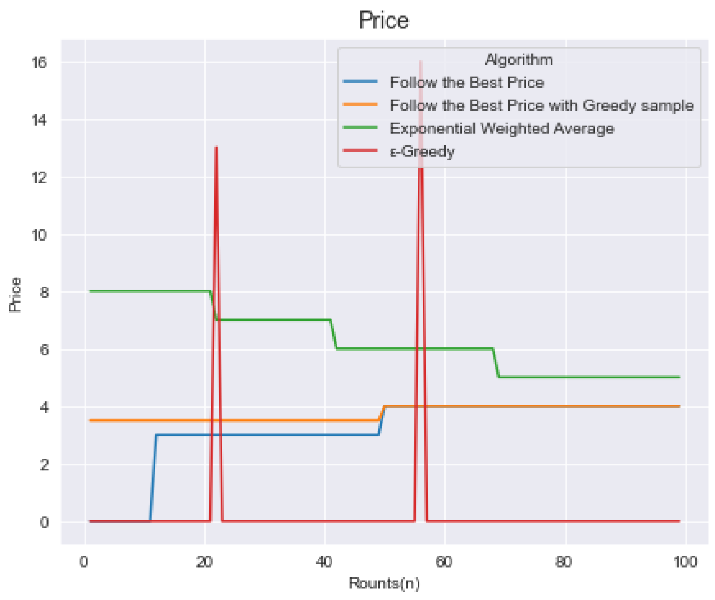


Figure A.3 **Price and Regret variation for Set 3 for all algorithms (100 rounds)**



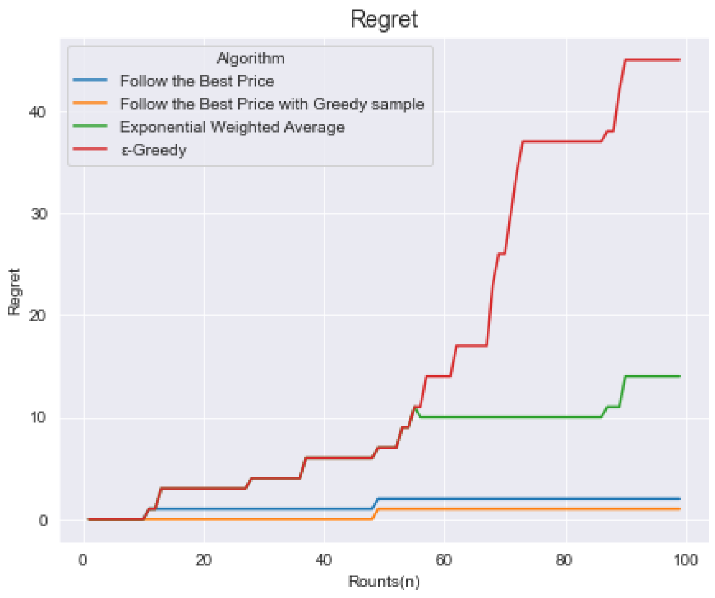


Figure A.4 **Price and Regret variation for Set 4 for all algorithms (100 rounds)**