UNIVERSITY OF SOUTHAMPTON

FACULTY OF SOCIAL SCIENCES

MATHEMATICAL SCIENCES

Earning while learning: Using Thompson Sampling to maximize rewards from online sales

by

Andria ELLINA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

April 2019

Declaration of Authorship

Andria ELLINA
"Earning while learning: Using Thompson Sampling to maximize rewards from online sales"
I declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research. I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University.
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
3. Where I have consulted the published work of others, this is always clearly attributed.
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
5. I have acknowledged all main sources of help.
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
7. None of this work has been published before submission

Signed:

Date:

University of Southampton

Abstract

Faculty of Social Sciences

Mathematical Sciences

Thesis for the degree of Doctor of Philosophy

Earning while learning: Using Thompson Sampling to maximize rewards from online sales

Andria ELLINA

The problem of finding the best option amongst a range of suboptimal candidates in an uncertain environment is a challenging task in a number of domains ranging from clinical trials to advertising, website optimization and dynamic pricing. Initially, very little is known about the performance of the different options and the decision maker needs to simultaneously learn about the performance of the different options and earn some reward from the decisions made. This introduces a trade-off between "exploration", the phase where new information is being acquired and "exploitation", where the goal is maximizing rewards or alternatively minimizing total regret. Regret is defined as the difference between the reward of an oracle strategy that selects the best option at each time step and the reward of the option we choose.

In this thesis, we develop new algorithms based on Thompson Sampling that improve the overall performance and minimize total regret. Numerical experiments are performed on simulated datasets in order to examine the effect of the algorithms' hyperparameters, to assess the robustness of the algorithms presented and compare the performance of our new algorithms with current algorithms. We use benchmarking experiments for a fair comparison of the different algorithms on the simulated datasets.

An additional complication, especially common in the area of revenue management, is seasonal changes that have an impact on the performance of the different options and consequently affect our decisions. In order to tackle the challenge of non-stationarity we deploy contextual Thompson Sampling to account for seasonality and develop a new algorithm that combines contextual Thompson Sampling with a standard statistical model selection method to solve the problem of unknown seasonality in the reward distribution of the candidate options.

Finally, we focus on an application of dynamic pricing in which we develop an algorithm that learns how to price a product in a competitive environment where the demand function is unknown. Using simulation we compare our algorithm in an oligopoly and duopoly setting with a set of other algorithms introduced elsewhere in the literature.

Acknowledgements

This journey was accompanied and made possible due to amazing people.
First and foremost, I would like to thank my PhD supervisors, Dr Christine Currie and Prof Christophe Mues. Their motivation, confidence and curiosity have encouraged me throughout these years. A special thanks to Christine for always making the time to discuss my work, regardless how confusing these discussions could get!
A very special gratitude goes to my family for their endless support and for always being there to inspire me, even from 4000 km away.
I would also like to thank all of my friends in Cyprus, Loughborough and Southampton for their continuous support.
Last but by no means least, a special thanks to Christos for being my co-rider in this 3.5-year adventurous journey. He has always been the best companion I could ever have!
Ευχαριστώ πολύ!!!!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis we focus on two main applications arising in earning while learning methodologies: website optimization and dynamic pricing. Website optimization is the process of finding the best version of a website in order to attract more customers. Dynamic pricing describes the process of changing the price of a product in real time in order to maximize revenue. The aim is to maximize total rewards by finding the best option amongst a range of candidate options, when we have limited initial information about their performance and the rewards are stochastic. In website optimization, the best option refers to the version of a website that will result in high sales and consequently high revenue. In dynamic pricing the best option is the price that will result in the highest possible revenue for the retailer. In both applications decisions are made in real-time with the effect of each decision becoming available during the process and not in advance.

An alternative term used to describe the idea of earning while learning, which has been receiving increased amount of interest by academics and practitioners (Sutton and Barto, 1998; Wang, Deng, and Ye, 2014), is exploration vs exploitation. This name captures the need to collect new information whilst operating the system and find the right trade-off between acquiring new information (exploration/ learning) and making the best decision, given the current information (exploitation/ earning). This approach is common in companies that seek for continuous improvements in their systems with information becoming available during the process. Thus, a good algorithm can avoid losses caused by wrong decisions in the exploitation phase.

Earning while learning methodologies are of interest in various other fields, ranging from hotel and flight reservations to personalized recommendation systems (Li et al., 2010a) and advertising (Agarwal, Chen, and Elango, 2009). What links these applications is the need to learn the optimal action on the fly so that in the long term the total revenue is maximized. These problems are particular examples of sequential ranking and selection methods but with the rewards being earned during the experimentation.

Since information about the performance of the different options is collected in real-time, the task of finding the best option is even more challenging when there are seasonal changes in the rewards of the candidate options. This implies that when collecting new information about the different options, as is the case in the website optimization problem we consider in Chapter 4, we always need to take into consideration the seasonality factor. Ignoring the impact of seasonality would result in comparisons between options tested at different times being meaningless.

In dynamic pricing, a particular problem in revenue management, the goal is to find the most profitable price for a product. Hence, learning refers to the phase where the retailer aims to obtain information about the demand of the candidate prices and earning is the phase where the most profitable price is chosen in order to maximize revenue. Here, we consider an example in which we have no prior information about the relationship between price and demand.

The methods developed in this research project are in the class of algorithms known as multi-armed bandits (MAB). In particular, our methods build on Thompson Sampling, which was originally applied to clinical trials (Thompson, 1933) but has since been applied to a number of different areas with very good empirical performance (Chapelle and Li, 2011). Initially, we study the problem in a stationary environment and then we proceed to the case where the reward of the different candidate options follows some seasonality. Finally, we consider an example of dynamic pricing, a particular application of MAB, in order to get a good understanding of what constitutes a good pricing strategy.

1.1 Contributions

Below we list the main contributions of this thesis before proceeding to the thesis outline describing what is included in each chapter.

- Development of the *learning and deployment buckets algorithm* that combines Thompson Sampling and ε-greedy. The algorithm uses the idea of splitting the traffic of a website into two buckets, introduced by Agarwal et al. (2009). Our empirical results on artificial datasets show that the algorithm has good performance in a stationary environment, in situations where there is high variability in rewards.

- A rigorous statistical comparison of Thompson Sampling, ε-greedy and our newly developed algorithm, based on benchmarking tests. We compared these algorithms with different parameter values on a large number of artificial datasets. The benchmarking tests we performed for a consistent comparison of the algorithms shed light upon the effect that the different characteristics of the datasets have on their performance and the benefits of our newly developed algorithm.

- Development of *AIC-TS*, a new algorithm that combines the idea of contextual bandits and statistical model selection for situations where the reward of the different candidate options follows some unknown seasonality. In this algorithm we incorporates the information about the season as contextual information, i.e. additional information that may affect the performance of the different options.

- Development of *TS-HR*, a novel pricing strategy that performs well in situations where little is known about market behaviour prior to the selling period. The pricing strategy builds on Thompson Sampling and incorporates a racing machine learning algorithm, Hoeffding races. We demonstrated its efficacy in dynamic pricing, by conducting an empirical study in monopoly and in a competitive market environment against other pricing strategies from the literature, in oligopoly and duopoly. The results demonstrate that it outperforms the other pricing algorithms in one-to-one competitions and provide further insights into the problem of dynamic pricing in a competitive environment.

1.2 Thesis outline

In Chapter 2, we present a literature review of important methodologies from the area of ranking and selection and machine learning that are related to our work. Algorithms from both areas have the same objective - compare a number of different options/“arms” in order to find the best one and maximize total rewards. Even though theoretical analysis of the algorithms is out of the scope of this thesis, some existing theoretical results are included along with important methodological details and significant empirical results.

In Chapter 3, we consider the problem of online decision making, in the framework of earning while learning, in a stationary environment. At the beginning, we provide a formal problem formulation and we introduce the terms that we will be using throughout the thesis. Then we present results from the practical implementation of two existing algorithms from the area of MAB, ε-greedy and Thompson

Sampling, and the new algorithm that we propose. The new algorithm, called learning and deployment buckets algorithm, aims to improve the performance of the standard formulation of the Bayesian learning algorithm. The three algorithms are empirically tested on two real-world datasets, in order to observe how they work and discuss their benefits and drawbacks. We are particularly interested in the practical implementation of these algorithms. Thus, in our results, we examine how the different values for the hyperparameters affect their performance when the reward of the different candidate options follows a normal distribution with unknown mean and variance. In Section 3.7, we test the performance of all algorithms described in the chapter on a large number of artificial datasets. The tests on the algorithms are followed by benchmarking tests that provide a consistent comparison of their performance on the different datasets.

Chapter 4 focuses on the implementation of Thompson Sampling in a seasonal environment. For this purpose, we use the idea of contextual bandits throughout the chapter and incorporate the information about the season as contextual information in the learning process. We first perform some testing on the hyperparameters to observe how they can affect the performance of contextual Thompson Sampling (Section 4.4). We then develop a new algorithm that combines contextual bandits with statistical model selection in order to handle the problem of unknown seasonality (Section 4.5). The algorithms are then tested on different artificial datasets and different seasonal environments in order to study the advantages and disadvantages of our method, AIC-TS, over different versions of the traditional Thompson Sampling algorithm.

In Chapter 5, motivated by the Dynamic Pricing Challenge [1] (Geer et al., 2018), we pay attention to a practical application from the area of revenue management, dynamic pricing. We propose a pricing strategy that builds on Thompson Sampling and incorporates an idea from Hoeffding's inequality. Then, we study the performance of our pricing algorithm by testing it on a simulated market environment with three different customer segments in two different environments: monopoly (Section 5.6.1) and in a competitive market environment (Section 5.6.2). In the competitive market environment the algorithm competes against other pricing strategies from the literature . Our findings demonstrate the complexity of the dynamic pricing problem and the benefits and drawbacks of our pricing algorithm, under the demand mechanism used in the experiments.

[1]Dynamic Pricing Challenge was a group challenge held on the occasion of the 17th INFORMS Revenue Management and Pricing Section Conference 2017

The thesis concludes in Chapter 6 which summarizes our contributions and presents future research directions.

Chapter 2

Literature review

This chapter provides an overview of existing methods used to solve the problem of finding the best option amongst a range of candidate options.

The problem we examine can be described as one of sequential ranking and selection. Thus, we start by explaining some existing ranking and selection procedures, mainly used in simulation, before proceeding to explain multi-armed-bandit methods, in order to examine whether there is a single method that is superior in a number of different settings. Along with the description of the main algorithms, different applications and examples of empirical work are also included in order to identify the research gaps in the area.

2.1 Ranking and selection algorithms

Sequential ranking and selection (R&S) procedures were first developed by statisticians but have also attracted attention within the computer simulation community (Branke, Chick, and Schmidt, 2007). The main objective is to either identify the best option (alternative) out of a set of candidate options based on some expected performance measure, compare all options against a standard, select the option with the highest probability of being the best or select the option with the highest probability of success, where success is a binary outcome. A general rule for sequential R&S procedures is that estimates of the mean and variance of each option are used to determine how samples should be collected sequentially until a stopping criterion is met. The three main steps followed by a typical R&S procedure are: initialization, screening, stopping rule.

What distinguishes a good R&S procedure is the requirement of fewer samples (replications) to achieve the same level of evidence for correct selection (Fu, 2015). In general, for the comparison of the performance of various alternatives, the number of samples for each option is gradually increased until the variance of the estimator becomes sufficiently small. Using the same number of samples for all options can be inefficient because if the variance of one option is very low then only a few samples can be enough for an accurate estimation of its performance (Fu, 2015). The same is true for the case where the performance of an option is much worse than that of the other options; continuing to sample this bad option is not very efficient.

R&S procedures have been used throughout simulation optimization (Boesel, Nelson, and Kim, 2003). Outputs in simulation are stochastic and algorithms are designed to be computationally efficient by choosing the appropriate set of input parameters to use in each iteration of the simulation optimization algorithm. Therefore, many of the ideas used in simulation optimization may be of benefit in online learning. Since simulations do not generate real output, the idea of earning while learning is not relevant and the focus is purely on efficient "learning". Instead of an earning/learning trade off, here we are dealing with a trade off between the estimated quality of a feasible solution and the uncertainty of the estimate in order to make an efficient allocation of the simulation budget and find the best option (Chau et al., 2014).

The three most popular approaches used to solve the problem of selection are the indifference zone (IZ) (Bechhofer, 1954), the optimal computing budget allocation (OCBA) (Chen, 1995) and the value of information procedure (VIP) (Branke, Chick, and Schmidt, 2007). All the approaches differ by the way sampling allocations are made and by the way the evidence of correct selection is described. Branke, Chick, and Schmidt (2007) give an overview of these three approaches in order to compare their performance.

Below we describe two of the most popular R&S procedures with the most relevance to our project.

2.1.1 Indifference-zone method

Indifference zone methods (IZ) determine an efficient sampling scheme that allows the best option to be selected with probability $1 - \alpha$ assuming that the difference between the objective values of the best and the second best is at least $\delta > 0$. The idea of IZ methods was firstly introduced by Bechhofer (1954) and it has been studied extensively in the literature since then (Rinott, 1978; Kim and Dieker, 2011). IZ methods aim to guarantee a lower bound for probability of correct selection (PCS), subject to the indifference-zone constraint, that the best option is at least $\delta > 0$ better than others. The methods provide a PCS guarantee and estimate the variance of the output of each option by using the sample variance from a first stage of sampling.

The state of the art for IZ methods is the KN++ procedure introduced by Kim and Kim and Nelson (2006a), extending their previous KN algorithm to allow for correlated samples. In the KN++ procedure there is one additional step, the update step, where the variance estimators are updated as more data are obtained and thus the efficiency increases since more samples are used for the calculation of the sample variance (Kim and Nelson, 2006b). Options that perform badly are screened out, a similar idea used in earning while learning algorithms as well, with the bad options stop getting chosen after sufficient information about their bad performance has been collected. The method provides only an asymptotic PCS guarantee.

A step by step description of KN++ is provided:

1. Specify:
   - Confidence level: $\frac{1}{N} < 1 - \alpha < 1$, where N is the number of options
   - Indifference-Zone Parameter: $\delta > 0$
   - First stage sample size: $n_0 \geq 2$
   - Initial batch size (i.e. number of samples to run per non-eliminated option, per subsequent stage): $m_0 < n_0$
2. **Initialization**: Let $I = \{1, 2, ..., N\}$, the set of the options we are comparing, $n = 0, z = n_0$
3. While the stopping rule is not satisfied:
   - Observe z samples from option i, $\forall i \in I$. Set $n = n + z$ and $z = m_0$
   - Update: $h^2 = 2\eta(n-1)$ and $\eta = \frac{1}{2}\left[\left(\frac{2\alpha}{N-1}\right)^{\frac{-2}{n-1}} - 1\right]$. Update the sample statistics $\bar{x}_i$ and σ_i^2 $\forall i \in I$.
     Screening:
   - For all $i, j \in I$ and $i > j$ set $d_{ij} = \bar{x}_i - \bar{x}_j$ and
     $$\epsilon_{ij} = \max\left\{0, \frac{\delta}{2n}\left(\frac{h^2(\sigma_i^2 + \sigma_j^2)}{\delta^2} - n\right)\right\}$$
     where n is the number of observations
   - If $d_{ij} < \epsilon_{ij}$ then stop sampling option i. If $d_{ij} < -\epsilon_{ij}$ then stop sampling option j, i.e. remove option i or option j from I respectively.

The **stopping rule** in KN++ is: $\mid I \mid = 1$ i.e. all options have been eliminated and there is only one option left

2.1.2 Optimal computing budget allocation

The main idea of optimal computing budget allocation (OCBA) (Chen, 1995) is to allocate a limited sampling budget in a way that maximises the probability of correct selection (PCS). Thus, the algorithm tries to estimate how PCS will be improved when additional samples are added to one of the options. Branke, Chick, and Schmidt (2007) summarize the OCBA algorithm as follows:

1. Specify:
    - First-stage sample size $n_0 > 2$
    - Number of options to simulate per stage: q
    - $z > 0$: Sampling increment z to allocate per subsequent stage
    - Stopping rule parameters
2. **Initialization**: Run independent replications $x_{i1}, x_{i2}, ..., x_{in_0}$ and initialize the number of replications $n_i = n_0$ run so far $\forall i \in I$.
3. Determine the sample statistics $\bar{x}_i$ and σ_i^2 and put the samples means $\bar{x}_1$, $\bar{x}_2$,...$\bar{x}_N$ in an increasing order .
4. While the stopping rule is not satisfied:
    - Compute $EAPCS_i$ (Estimated approximate probability of correct selection) $\forall i \in I$
      Screening:
    - Calculate $EAPCS_i - PCS_{Slep}$ and set $z_i = \frac{z}{q}$ for the z options with largest $EAPCS_i - PCS_{Slep}$ and set $z_j = 0$ for the others.
    - Run z_i additional observations for option i.
    - For all i with $z_i > 0$, update $n_i = n_i + z_i$, the sample statistics $\bar{x}_i$ and $\widehat{\sigma^2}$ and order statistics so that $\bar{x}_{(1)} < .... < \bar{x}_{(N)}$.
5. Select the option with the best estimated mean.

For more details on the calculation of some of the statistics mentioned below we refer the readers to (Chen, 1995).

The algorithms described above are very popular methods used in sequential R&S, for simulation optimization. However, in order to investigate further how the problem of R&S can be solved in online fashion we choose to move towards methods used in machine learning to solve similar problems. A category of algorithms in machine learning used to solve the problem of finding the best option amongst a range of suboptimal candidates is known as *Multi-armed bandits*.

2.2 Multi-armed bandit algorithms

Multi-armed bandit (MAB) algorithms aim to balance the trade-off between collecting new information (exploration) and maximizing rewards (exploitation), when there is limited initial information about the performance of the different options we are comparing. The objective is to find the optimal/best option and maximize rewards over a given time horizon. Within the MAB framework, the options we are choosing between are the arms and we use the terms arm and option interchangeably in what follows.

Even though the short term reward is maximized by choosing the arm that has performed well in the past, long term reward is maximized by sufficient exploration of all arms in order to find the optimal. At each iteration of a MAB experiment, the algorithm chooses which arm $a \in \{1, 2, ..N\}$ to "pull", i.e. which option to test during the next period and it receives its reward $r_{a_t}(t)$, while the rewards that could have been obtained from the other arms remain unknown. The MAB algorithm will then use all the information collected so far about the performance of the options, to determine which option to choose in the next time period.

The main objective of a MAB algorithm is to maximize the total reward over a fixed time horizon T, $\sum_{t=1}^{T} r_{a_t}(t)$, where $r_{a_t}(t)$ is the reward of arm a at time t. This is equivalent to minimizing total regret, where regret is defined as the difference between the reward of an oracle strategy that selects the best arm at each time step and the reward of the chosen arm. It is given by the following equation:

$$E[R(T)] = E[\sum_{t=1}^{T}(r_{a_t^*} - r_{a_t}(t))], \tag{2.1}$$

where r_{a^*} is the expected reward of the best arm at time t, given by $a_t^* = \underset{a=1,2,..,N}{\text{argmax}} E[r_{a_t}]$. Due to the limited initial knowledge about the options, the uncertainty at the beginning of the experiment is expected to be very high. The time horizon T is very important because, as Li et al. (2010b) point out, the algorithm may have a huge regret in the short term since we "play" suboptimal arms more often, while in the long run regret stops increasing since our decisions are based on collected information about arms that have previously performed well.

The main characteristics of a typical MAB problem are the constant number of arms, the unknown but fixed reward distribution of each arm, the reduction in the uncertainty after each pull and finally the fact that, at each time step, only the reward of the selected arm is observed. The fact that we get only partial information for the problem is the main difference between MAB and other machine learning procedures (Burtini, Loeppky, and Lawrence, 2015).

The most commonly used MAB algorithms are ε-greedy, UCB (Auer, Cesa-Bianchi, and Fischer, 2002) and Thompson Sampling (Thompson, 1933). Boltzmann Exploration/SoftMax (Sutton and Barto, 1998), Pursuit Algorithm (Thathachar, 1984) and Gittins index (Gittins, 1979) are also popular heuristics used for the same purpose. A description of the most popular algorithms follows this brief introduction to the general problem.

2.2.1 ε-greedy

The algorithm makes a random selection of any of the options with probability ε and chooses the best option with probability $1 - \varepsilon$; therefore, on average we would expect it to spend εT of the time on exploration and $(1 - \varepsilon)T$ on exploitation (Vermorel and Mohri, 2005). The probability of exploration, specified by the parameter ε, is in the interval [0,1] and defined by the user at the outset. Consequently, if $\varepsilon = 1$ we have pure exploration while in the case where $\varepsilon = 0$ we have pure exploitation.

White (2012) showed that the performance of ε-greedy depends first on the value of ε and then on the time we look at our results. After running simulations for different values of ε, he showed that when ε is high the algorithm finds the best arm more quickly since there is much more exploration taking place but at the end of the experiment the probability of selecting the best arm is not high enough because the algorithm is exploring without sufficient exploitation. On the other hand, as ε gets smaller the algorithm finds the best arm much later but the probability of selecting the best arm at the end reaches a higher value. He also obtained similar results when he compared the average reward and the cumulative reward at each point in time for the different ε values. Thus, the choice of ε is very important and its optimal value depends on the length of the time horizon we intend to leave an ε-greedy algorithm to run (White, 2012).

The fact that the value of ε is chosen at the beginning means that in many cases the algorithm may keep exploring even after the optimal option has been found and computational time is wasted in the exploration of non-optimal options while at the same time we keep missing the opportunity to apply the best option. This is why a new modified algorithm, called ε-decreasing, which decreases the amount of exploration, is sometimes used (Vermorel and Mohri, 2005). In ε-decreasing, $\varepsilon_t = \min\left\{1, \frac{\varepsilon_0}{t}\right\}$, where $\varepsilon_0 > 0$ so that $\varepsilon \to 0$ as $t \to T$. The value of parameter ε_0 is chosen at the beginning and it is very important. Similar to ε-greedy, a small value for ε_0 will lead to insufficient exploration while a large ε_0 will lead to high cumulative regret due to a delayed convergence. However, the algorithm is still wasteful since even with a decreasing value of ε the exploration is still based on simple random sampling (Scott, 2010).

A particular example of ε-greedy strategy is known as ε-first. ε-first strategy is the simplest variant of ε-greedy. In comparison with standard ε-greedy, where exploration takes place at the same time as the exploitation phase, in ε-first the exploitation comes only after all the exploration is done. A major drawback of ε-first is that since all learning takes place upfront, in the case of a non-stationary reward distribution the algorithm does not inform the user about these new changes. On the other hand, ε-first may work well enough in the case of short-terms experiments because in this case even when inferior strategies are not explored, the consequences are not so important (White, 2012).

Moreover, A/B testing is a variant of the simple ε-first algorithm and it is used for the comparison of usually two options. It is also known as split test, and is the benchmark method used for controlled experiments, in website optimization, to measure clicks and conversion of website visitors. The idea behind A/B testing is that the traffic of a website is split into two, some users of the website are shown the existing version of the website, called the control version, and some users are shown the new version that needs to be evaluated, called treatment. This method is similar to ε-greedy in a way that exploration, defined by ε, is dictated by the proportion of traffic sent to the treatment option. For an overview of A/B controlled experiments go to Kohavi et al. (2009).

2.2.2 Upper confidence bound

One of the most popular algorithms with various practical applications, introduced by Lai and Robbins (1985), is Upper Confidence Bound (UCB) which is based on the idea of optimism in the face of uncertainty. UCB can be characterized as an optimistic exploration algorithm since it may continue to explore all arms, even though some of them do not perform very well at the beginning (Agrawal and Goyal, 2012). While other algorithms would under-explore these arms, UCB will keep exploring them. Thus, the main difference between UCB and other well known algorithms is that instead of playing the arm with the highest reward, each arm is assigned an UCB for its mean reward, and the arm with the largest bound is played (Scott, 2010).

One of the most commonly used versions of UCB is UCB1 for cases where the reward of each arm is in $[0, 1]$ (Auer, Cesa-Bianchi, and Fischer, 2002). Firstly, each arm is played once. Then the algorithm picks the arm j at time t such that

$$j(t) := \underset{a=1,2,..,N}{\operatorname{argmax}} \Big(\widehat{r_a} + \sqrt{\frac{2\ln(t)}{k_a}} \Big), \tag{2.2}$$

where $\widehat{r_a}$ is the average expected payoff of arm a, over the previous $t - 1$ iterations, and k_a is the number of times arm a has been played (Burtini, Loeppky, and Lawrence, 2015; Kuleshov and Precup, 2014). What makes UCB so interesting and easy to understand is the absence of parameters chosen by the user before the implementation of the algorithm (White, 2012). At time t, the expected regret of the UCB1 algorithm is bounded by

$$8 \sum_{a=1}^{N} \tfrac{\ln(t)}{\Delta_a} + (1 + \tfrac{\pi^2}{3}) \sum_{a=1}^{N} \Delta_a,$$

where $\Delta_a = r^* - r_a$ with r^* being the reward of the best arm (Auer, Cesa-Bianchi, and Fischer, 2002). So UCB1 achieves the optimal regret up to a multiplicative constant.

There are various UCB methods that use different functions to calculate the bounds for the regret and they all perform very well in cases where the number of arms is small and the reward variances are high but their performance gets worse as the number of arms gets larger (Kuleshov and Precup, 2014). They also return good results when the rewards are independent and have no shared parameters. One reported practical application of UCB is website optimization, where Yahoo! uses it to optimize the articles shown on the Yahoo! front page (Li et al., 2010a).

2.2.3 Thompson Sampling

Thompson Sampling, also known as randomized Bayesian heuristic, is one of the earliest heuristics developed to handle the exploration/exploitation problem (Thompson, 1933; Scott, 2010). The idea is to first assume a prior distribution on the parameters of the reward distribution of each arm. After each time step, the prior distribution for the reward of the arm that has been played is updated to give a posterior distribution. Which arm to play is chosen by random sampling of the posterior distribution of each of the arms, where the arm with the highest sampled reward is the one that is played. This is what distinguishes the algorithm from other MAB techniques such as ε-greedy where the decision made is based on the expectation of the reward (Russo et al., 2018). Another term used very often in the literature is the probability matching technique which describes the matching between the probability to choose an arm and the probability that this arm has the highest reward (Scott, 2015).

In this process conjugate priors are usually used for computational purposes. This means that for a certain likelihood, the posterior probability distribution and the prior probability distribution are in the same family. Arms are more likely to be played if their posterior distribution has either a high centre point, which implies a high average reward, or a high variance, which implies limited information about their performance. Therefore, the algorithm is more likely to choose arms that have performed well in the past or arms that have not been chosen enough times in the past to learn about their performance. Hence, the fact that the algorithm stops exploring inferior arms leads to greater sample sizes for the superior ones.

The main steps are listed in Algorithm 1. Note here that initially we take a random sample from the prior distribution, then based on the reward obtained from the chosen arm we calculate the posterior distribution which then becomes the new prior in the next iteration.

Algorithm 1 Thompson Sampling

```
for t=1,2,..,T do
    Set the initial hyperparameters
    Take a random sample $\tilde{r}_a$ from the prior of arm $a$ for all $a \in \{1, 2, ..N\}$
    Select the arm with the highest sample value $a_t = \underset{a=1,2,..,N}{\text{argmax}} \tilde{r}_a$ and observe its reward
    Update the hyperparameters of arm $a_t$ and calculate its posterior distribution. This posterior becomes the new prior used in the sampling process in the next iteration
end for
```

Thompson Sampling has been shown to work well in practical applications (Scott, 2010; Chapelle and Li, 2011) but theoretical results have only become available relatively recently. The first theoretical analysis of the algorithm comes from Kaufmann, Korda, and Munos (2012) who defined a problem-dependent regret bound for bernoulli bandits i.e. binary/binomial reward distribution. Their asymptotically optimal regret bound for N arms and time T is given by the following equation.

$$R(T) \leq (1+\epsilon) \sum_{a \in N: \mu_a \neq \mu^*} \frac{\Delta_a(\ln(T) + \ln\ln(T))}{D(\mu_a \parallel \mu^*)} + c(\epsilon, \mu_1, ..., \mu_N), \tag{2.3}$$

where μ^* is the mean reward of the best arm, $\epsilon > 0$, $c(\epsilon, \mu_1, ..., \mu_N)$ is a problem dependent constant and $D(\mu_a \parallel \mu^*)$ is the Kullback-Leibler divergence between μ_a and μ^* given by

$$D(\mu_a \parallel \mu^*) = \mu_a \ln\left(\tfrac{\mu_a}{\mu^*}\right) + (1-\mu_a)\ln\left(\tfrac{1-\mu_a}{1-\mu^*}\right),$$

They managed to reach the lower bound for regret, that Lai and Robbins (1985) have proven for any stochastic MAB problem. Until then, UCB was the only MAB algorithm satisfying this bound and thus their work played an important role in making Thompson Sampling comparable to the most commonly used algorithm at the time. Note here that the lower bound on regret for UCB is

$$E[R(T)] \geq \Big[\sum_{a=2}^{N} \frac{\Delta_a}{D(\mu_a \parallel \mu^*)} + o(1)\Big] \ln T. \tag{2.4}$$

Agrawal and Goyal (2012) continued the work on the theoretical analysis of the Bayesian algorithm and they obtained the following results for a 2-armed Bernoulli bandit problem

$$E[R(T)] = O\Big(\frac{\ln T}{\Delta} + \frac{1}{\Delta^3}\Big), \tag{2.5}$$

where $\Delta = \mu^* - \mu_2$ provided that $\mu^* = \mu_1$. For the case of N Bernoulli arms

$$E[R(T)] \leq O\Big((\sum_{a=2}^{N} \frac{1}{\Delta_a^2})^2 \ln T\Big). \tag{2.6}$$

A year later, Agrawal and Goyal (2013a) attempted an extension of this work on Bernoulli bandits to the case of Gaussian rewards. Following the general lower bound of $O(\sqrt{NT})$ for any MAB problem, they obtained a general near-optimal problem-independent bound of $O(\sqrt{NT \ln T})$ and a problem-independent bound of $O(\sqrt{NT \ln N})$ for Gaussian priors with rewards in the range $[0,1]$. At this point we should mention that even though in our algorithms cumulative regret stops increasing and convergences to a certain value, the regret bounds defined above for Bernoulli bandits do not hold since we concentrate on Normal reward distribution. Nonetheless, with the right set up there is a chance that the algorithms would follow the convergence rates given by Agrawal and Goyal (2013a) but this has not been tested in our work on stationary MAB problems. In the case of seasonality, developing convergence is likely to be much more difficult due to the additional complexity of seasonality.

Even though Thompson Sampling is a flexible heuristic that can be used with various reward distributions (Scott, 2015), the most common example used in the literature is the case of binomial rewards, where the prior assigned to the parameters is its conjugate beta prior. Examples of real-life situations that can be described by this model of binary rewards are the clicks/no clicks for a website advertisement or a buy/no buy model (Scott, 2010; Chapelle and Li, 2011; Agrawal and Goyal, 2012). Other authors have considered different reward distributions; e.g Honda and Takemura (2014) studied the case where the reward distribution is normal.

The choice of priors is very important and it represents the prior knowledge about the performance of the different arms. When we take knowledge into account, the algorithm spends less time in exploration and thus it has more time to exploit superior arms. Thus, in cases where prior knowledge from empirical data is ignored, we face the situation of large losses, due to too much exploration (Russo et al., 2018).

When empirically compared to UCB in particular, Thompson Sampling has shown more robustness in a situation with delayed feedback (Chapelle and Li, 2011). Delayed feedback is defined by Agrawal and Goyal (2012) as the situation where even though we want to make a quick decision about which arm to play next, the result of playing an arm becomes available only after some time delay. This property of Thompson Sampling makes it the most suitable algorithm in the area of display advertisement where there is a time delay between the time a user sees an advertisement and the time they click on it (Chapelle and Li, 2011). A possible explanation for this robustness of the algorithm, is the fact that the heuristic is randomized. This means that it splits the traffic into different arms in proportion to their performance while the system is waiting for new updates, when other non-randomized algorithms end up with greater variance in their rewards since they keep pulling the same arm during this delay (Scott, 2015). Other practical applications of this Bayesian online learning algorithm include Microsoft's Bing Search where the decision is made between different advertisements to be placed on their website (Graepel et al., 2010).

An important assumption made for any standard MAB algorithm is the stationarity of the reward distributions. However, this assumption is very often violated, since in many real-world applications assuming that all arms follow a stationary distribution over time is not feasible. Thus, in the following section we take a closer look to a number of algorithms from the literature that handle non- stationary problems.

2.3 Time-dependent bandits

Detecting changes in the reward distributions is more difficult for online learning algorithms than in offline situations where all of the data are available in advance. Thus, making suitable changes to the algorithms is crucial in order to first capture these changes and then adapt the procedure of finding the best alternative by using these information.

Whittle (1988) introduces the term "restless bandits" to describe the situation where the characteristics of the different arms change continuously, even for those arms that are not played. To describe the problem in terms of MAB, he introduces the terms active and passive arms to define whether an arm is played or not, respectively. Thus, the basic assumption for the case of restless bandits is that even in the passive phase, i.e. some of the arms are not played, their characteristics are changing. It has been shown that for a non-stationary problem, an algorithm based on stationarity assumptions performs almost as badly as the random choice of options (Bouneffouf and Feraud, 2016). In particular, in the case of Thompson Sampling with binary rewards, the standard formulation of the algorithm becomes less effective in time, while Thompson Sampling that tries to capture the underlying changes of the system performs better. However, it is worth mentioning that in non-stationary situations, regret may not vanish completely by the end of the experiment (Russo et al., 2018).

The proposed solutions for non-stationary problems fall within two main categories. First are the methods that forget some of the historical observations and use only the most recent ones in the process of decision making and second are the methods that aim to track the changes in the time-varying parameters in order to find the underlying model that describes the performance of the different options over time. We split the non-stationary problems into three main categories based on how these changes occur: Brownian motion, switching environment and drifting environment. Subsequently, we explain some of the existing methodologies used to solve problem within these three categories.

1. **Brownian motion:** This is the case where the changes in the reward distribution are completely random and can be described only by Brownian motion (Slivkins and Upfal, 2008).

2. **Switching behaviour:** Switching behaviour involves a sudden change in the reward distribution of the arms. This implies that the reward distributions of the different options remain constant for some periods of time and experience sudden changes at unknown times. Sudden changes are very common in various domains such as the financial sector where unexpected failures may happen.

   A further distinction in switching behaviour is made between situations where, when a switch happens, all arms are affected (global switching) and situations where the change happens independently for the different arms (Per Arm Switching). In the case of switching behaviour, either Global or Per Arm Switching, the optimal adaptation is for the algorithm to forget some of the rewards observed in the past in order to take into account only recent information that is probably more relevant (Hartland et al., 2007). Two algorithms that build on this idea are modifications of the UCB algorithms and are known as discounted UCB (Kocsis and Szepesvári, 2006) and Sliding UCB (Garivier and Moulines, 2011). In a simulation study, Garivier and Moulines (2011) study the problem of having three different Bernoulli variables and the probability of success for one of the arms changes at two time steps. From their results, it is clear that Sliding Window UCB performs slightly better that Discounted-UCB and much better than UCB1. Moreover, it is suggested that the algorithms that build on UCB1, which performs a logarithmic regret in the stationary case, can reach a regret of order $t/log(T)$ in a switching environment. In the example of two arms, where the one arm changes periodically, the two algorithms perform almost the same.

   Mellor and Shapiro (2013) combine Thompson Sampling with a change-point detection algorithm and test it in both environments; global and per arm. Since they do not know when the last switching event occurred in their Global Change-Point Thompson Sampling algorithm, they define the posterior distribution they need to sample from, $P(\theta \mid D_{t-1})$, as the product of the posterior of model, given the data and the probability of run length r_t since a switching event occurred. Hence,

   $$P(\theta \mid D_{t-1}) = P(\theta \mid D_{t-1}, r_t) P(r_t \mid D_{t-1}).$$

   Thus, in order to use Thompson Sampling, they first sample the run length since last change occurred, in order to understand how much data from the past they can use. Then they sample individually for each arm, from the posterior distribution of the arms, given the run length, and they pull the arm for which the sample from the posterior distribution is highest.

3. **Drifting behaviour:** These cases are characterized by continuous changes to the reward distributions. In this category of problems, Bouneffouf and Feraud (2016) study the problem of multi-armed bandits with known trend, which implies that the reward distribution of each arm is known but the environment is changing. They assume that the distributions of the arms change based on some known functions. For the formulation of their problem, they define the non-stationary reward function $z(t) = r_{a_t}(t)D(n_{a_t}(t))$, where $D(n_{a_t})$ is the known function for trend , $n_{a_t}(t)$ is the number of times arm a is played and $r_{a_t}(t)$ is the stationary reward of arm a.

2.4 Contextual bandits

The algorithms that use additional information in the process of decision making to improve their overall performance are known as contextual bandits, a subcategory of MAB algorithms. In many settings, contextual information can be treated as extra information that can reveal the current state of the environment. The algorithm will then take the context into account when recommending the next arm to pull. For example, seasonality in demand can be treated as contextual information in the situation where the price of a product must be frequently adjusted in order to maximize revenue.

In this project, our interest mainly revolves around stochastic contextual bandits. This class of algorithms is based on stochastic data generation models and the rewards of the arms are drawn from unknown reward distributions. Most of the existing work carried out in the framework of stochastic contextual bandit problems introduce a new vector c_t which includes the context up to time t. Then the user for every new observation uses the vectors $(c_t, a_t, r_{a_t}(t))$, form all the previous time steps until $t - 1$, to decide which arm to play next (Li et al., 2010a; Chapelle and Li, 2011).

Recent development in the area is driven by contextual advertising (Li et al., 2010b), personalized web search (Li et al., 2010a), personalized recommendation systems (Wang et al., 2014) and web page organization (Agarwal, Chen, and Elango, 2009). In those examples, contextual information may include information about the user, the different arms and perhaps some information about the environment.

In the particular case of contextual advertising, the aim is to find the most profitable advertisements to place on a website based on some side information about the content of the site, the user and his recent behaviour and the advertisements (arms) (Madani and DeCoste, 2005). In all of the above cases the most common measure of success is the Click Through Rate (CTR) which is defined as the number of users that click on a website divided by the total number of users that have visited the page.

A popular algorithm in website optimization, developed to handle the problem of personalized recommendations was developed by Yahoo!, one of the biggest portal sites. Their aim is to increase their users' satisfaction by presenting them with news articles that fit better with their interests and place the best article on the story position (i.e top position) on Yahoo! Front Page. The algorithm they use for this purpose is based on the idea of confidence bounds and is called LinUCB. Using this algorithm, which assumes a linear relationship between c_t and r_{a_t}, they rank the remaining stories based firstly on their users interests and also on the popularity of each article, by comparing the individual CTRs (Li et al., 2010a). Their methodology builds on the idea of Agarwal et al. (2009) which involves two separate buckets to which they send different amount of traffic. They use a "learning bucket", to which they send only a small amount of traffic in order to learn the CTR of each article and the "deployment bucket" that serves the users using the information obtained by the "learning bucket". In the "deployment bucket" the article with the highest CTR is chosen for the users but this CTR changes with time according to the information obtained in the "learning bucket". In their study they compared the performance of the random exploration algorithm, ε-greedy with their proposed algorithm and they found out that LinUCB outperforms the simple ε-greedy algorithm.

In contextual advertising, some modifications to the simple ε-greedy algorithm are made in order to handle additional complications arising in the area. A main complication is the fact that the arms are no longer independent, since some advertisement topics may be related, and getting information about one of the arms may give us information about others as well. Thus, in order to make the algorithm work in the contextual advertising setting, Li et al. (2010b) keep changing the value of parameter ε dynamically and then they try to change the randomness of the algorithm by choosing only possible superior arms.

In the framework of Thompson sampling with arbitrary contexts and stochastic rewards, the main algorithm uses the history set $H_{t-1} = \{(a_\tau, r_{a_\tau}(\tau), c_\tau) \mid 0 \leq \tau \leq t-1\}$. Thompson Sampling with contextual information has also been studied by

Porter et al. (2016) and Agrawal and Goyal (2013b) with important practical and theoretical results, respectively. The main steps of the algorithm are then adjusted and the formulation is given below(Chapelle and Li, 2011):

Algorithm 2 Contextual Thompson Sampling

```
$H = \emptyset$
for t=1,2,..,T do
    Receive context $c_t$
    Draw $\theta_t$ from $P(\theta \mid H)$
    Select arm $a_t = \underset{a=1,2,..,N}{\operatorname{argmax}} E[r \mid c_t, a, \theta_t]$
    Observe reward $r_{a_t}$
    $H = H \cup (c_t, a_t, r_{a_t})$
end for
```

Lloyd and Leslie (2013) solved the problem of animals adaptation in different environments by using contextual bandits, based on Thompson Sampling. In this case, their main model consists of the generative model, the inference model and the choice model. The generative model is used to identify whether there is a change in the context (environment) or not. There is a probability π that the context switches and it either switches to a previously seen context or to a new context. In the inference model the animal learns the characteristics of each context in order to identify which one is currently active. Thompson Sampling is then used in the choice model in order for the animal to decide which action to take. Thus, each context-action pair has its own hyperparameters and at each time step the algorithm calculates the posterior for each context-action pair after it has decided whether there was a change in the context. More details on their work, which is based on Normal likelihood with a Normal-Inverse Gamma prior are included in Chapter 3.

An important assumption for contextual bandits is that the reward distribution stays the same under the same context. Very recently, Zeng et al. (2016) described a new variation of the contextual bandits where even though the context stays the same, the reward distribution of the arms changes. Their work combines contextual bandits with non-stationarity for the reward distribution of the different arms. Using the standard formulation of Thompson Sampling for normally distributed arms with unknown variance, they incorporate an additional drift factor to capture the change in the reward distribution. So given the model where the estimated reward of arm a at time t is given as a linear combination of context X

and the coefficient vector $\boldsymbol{\theta}_{\alpha,t}$ they define

$$\boldsymbol{\theta}_{\alpha,t} = \mathbf{c}_{\theta_{\alpha,t}} + \delta_{\theta_{\alpha,t}}, \tag{2.7}$$

where $\mathbf{c}_{\theta_{\alpha,t}}$ is the stationary component of the coefficient vector and $\delta_{\theta_{\alpha,t}}$ the drift component. This work could be useful in a situation with both seasonality and trend in the reward distribution of the different arms.

2.5 Concluding remarks

In conclusion, the problem of decision making in an uncertain environment has been studied extensively in the literature, in different areas. Our literature review revolves around methodologies developed in two particular frameworks: ranking and selection and multi-armed bandits, with more attention given to the latter, which accounts for the impact of earning while earning. The algorithms explained build on different ideas and interestingly, even though there are settings and situations that justify the use of certain algorithms (Agrawal and Goyal, 2012), there is not a single algorithm that performs well in all settings.

In recent years, similar algorithms have also been developed in the popular area of Operational Research, dynamic programming. Two categories of algorithms within the area of dynamic programming that have been developed in order to handle real-time problems are the Adaptive Real-Time dynamic programming and the Real-time dynamic programming (Barto, Bradtke, and Singh, 1995; Barto, 2010; Sanner et al., 2009). Their difference is that the first one is a heuristic that learns a model from data collected during agent-environment interaction while the later one does not have the model learning component.These learning algorithms that are based on dynamic programming aim to improve the computational efficacy for conventional dynamic programming algorithms.

From the literature it became apparent that the biggest research gaps lie within the practical implementation of the algorithms. Firstly, there is only limited work on how the properties of the candidate options affect the performance of MAB algorithms (Kuleshov and Precup, 2014). Secondly, all comparisons between two or more algorithms in the literature are based on their performance on particular datasets, without any statistical significance (Chapelle and Li, 2011; May et al., 2012). Thus, with our work in the first two chapters we aim to shed some light upon the factors that have an impact on the performance of some MAB algorithms, by carrying out benchmarking tests in which we implement them with different parameter values on a number of artificial datasets, and perform a consistent statistical comparison between different algorithms.

We choose to build our studies, in both stationary and seasonal environments, on Thompson Sampling, the Bayesian exploration algorithm whose performance has been very good in many real-world applications (Scott, 2010; Chapelle and Li, 2011). In both settings, we choose to modify the standard formulation of the algorithm in order to examine possible advantages over the standard formulation. Additionally, since most of the practical work on Thompson Sampling is on binomial reward distributions we choose to extend its main formulation into cases where the reward follows a continuous distribution, specifically the normal distribution.

Chapter 3

Stationary case

In this chapter, we consider the problem of finding the best option amongst a range of suboptimal candidates in an uncertain but stationary environment. The decision at each time step is made on the fly which means that, in the process of decision making, the information collected so far about the performance of the different options is being used. Three important conditions of the problem studied in this chapter constitute the main characteristics of any standard multi-armed bandit problem, where the term arms refers to the different candidate options we are comparing: constant number of arms, reduction in the uncertainty after each pull and each arm being characterized by an unknown but fixed reward distribution.

Any of the algorithms described in Section 2.2 can be implemented for the solution. We present algorithms and results using Thompson Sampling and ε-greedy for the solution of this stationary problem. ε-greedy is one of the simplest algorithms in the family of MAB and thus it is very popular in industry despite its non-optimal results in many cases. The simple idea behind it and its lack of assumptions make it easy to use in a wide range of applications which is its main advantage over other MAB techniques. Hence, it serves as a useful benchmark against which to compare other more sophisticated methods. Secondly, Thompson Sampling is included due to its good empirical performance when compared to other popular bandit techniques (Chapelle and Li, 2011). Following the implementation of the two standard algorithms, we introduce a new methodology we developed that combines the two ideas and is based on a widely used idea in the area of website optimization.

To empirically validate the empirical performance of each of these methods, we consider the problem setting of a major online travel agency that aims to find the most profitable version for their website. For this practical application, we study how the choice of different parameters affects the performance of the algorithms in order to assess how the three algorithms perform. Then we proceed with a more comprehensive study of the algorithms in a range of different settings by applying them to a large number of artificial datasets and compare their performance using benchmarking tests, used in parts of the machine learning literature.

The structure of this chapter is as follows: In Sections 3.1 to 3.4, we give some important background information involved in the algorithms we implement, we describe the problem we are trying to solve and explain in more detail two standard techniques from MAB: ε-greedy and Thompson Sampling. In Section 3.5, we introduce a new methodology, based on bucketing, and then we proceed with some experimentation on the three algorithms using two real-world datasets in Section 3.6. This is followed by Section 3.7, where we explain the set-up we use to create additional artificial datasets and the results obtained by applying the three algorithms on these datasets. In the same section, we perform a fair comparison between the three algorithms using techniques from the area of benchmarking classification methods and we discuss the differences in their performance when tested on the large number of artificial datasets. Finally, Section 3.8 concludes the chapter and highlights the main outcomes and the most important results obtained for the stationary environment problem.

3.1 Background information

Some of the algorithms described throughout the thesis are based on important statistical theorems/rules in a Bayesian framework. In Bayesian updating, prior knowledge is treated as a random variable and expressed formally by a prior distribution, while sample data are expressed by the likelihood function. Bayes rule describes the relation between prior distribution and likelihood in order to obtain the posterior distribution. This relation is given by the following equation (West and Harrison, 2006):

$$p(\theta|y) = \frac{p(\theta,y)}{p(y)} = \frac{p(\theta)p(y|\theta)}{p(y)}, \tag{3.1}$$

where $p(y) = \sum_\theta p(\theta)p(y|\theta)$, $p(\theta|y)$ is the posterior distribution of some parameters θ given some data y and $p(y|\theta)$ is the likelihood. Since $p(y)$ is independent of θ we can omit it from Equation 3.1 and obtain Bayes rule in the following form

$$p(\theta|y) \propto p(y|\theta)p(\theta), \tag{3.2}$$

where $p(\theta)$ is the prior distribution. Moreover, the distribution of the unknown but observable y, also known as prior predictive distribution is

$$\int p(y,\theta)d\theta = \int p(\theta)p(y|\theta)d\theta.$$

Given a certain likelihood, if the posterior probability distribution and the prior probability distribution are in the same family then the two are called conjugate distributions and the prior probability distribution is called the conjugate prior of the given likelihood (Raiffa and Schlaifer, 1961). Some commonly used pairs of likelihood-prior probability distributions are given in Table 3.1. Note here that the pair of likelihood-prior probability distributions incorporated in the algorithms explained in the following chapters are not included in the table.

Likelihood	Parameters	Conjugate prior distribution	Prior hyper parameters
Normal with known variance σ^2	μ	Normal	μ, σ
Normal with known precision λ	μ	Normal	μ, λ
Normal with known mean μ	σ^2	Inverse Gamma	α, β(scale)
Normal with known mean μ	λ	Gamma	α, β(rate)
Bernoulli	p(probability)	Beta	α, β

TABLE 3.1: List of some likelihood functions with their conjugate prior distributions and prior hyperparameters

3.2 Problem formulation

We want to compare N different options, such that $a \in \{1, 2, ..., N\}$, when we have very limited prior information about their performance. A common industry problem that falls in this category is the problem of website optimization where the different options compared are different candidate versions of a website and each visitor decides either to click on a link or not. The rewards in this case come from a binomial reward distribution (click or no click) which is why most of the existing work on Thompson Sampling in this context assumes binomial reward distribution (Scott, 2015). However, in our work, we consider the case of normal likelihood $N(m_a, s_a^2)$ for each option with unknown mean m_a and standard deviation s_a ($s_a = \frac{1}{\lambda_a}$, where λ_a is precision), in order to account for a variety of real-world situations where the reward of the options comes from a continuous distribution. For example, in the same setting of website optimization, it might be possible that the aim is to increase the total revenue that results from the total number of clicks on a website rather than the number of clicks itself (click or no-click). Note here that the most common term used in the multi-armed bandits literature that refers to the options is "arms", thus we keep using this term when describing any methodology in this category of algorithms while we choose to use the term option when we are in a more general context.

At each time step $t \in [0, T]$, the algorithm decides which arm a_t to choose and it receives its reward $r_{a_t}(t)$ while the rewards of non-chosen arms remain unknown. Using this reward $r_{a_t}(t)$, the algorithm updates its beliefs about the chosen arm and the criteria involved in the process of selecting the best arm, repeating the same procedure until the end of the experiment. The two main measures of performance for the algorithms are cumulative regret and percentage of optimal selection. Regret indicates how much the algorithm loses due to exploration while the percentage of optimal selection describes the proportion of time spent in choosing the best option, the one with the highest expected reward.

3.3 Stationary Thompson Sampling

3.3.1 Bayesian analysis

The choice of normal reward distribution with unknown mean and variance is very limited in the literature with most of the existing work being on the binary reward distribution (Scott, 2010; Chapelle and Li, 2011; Agrawal and Goyal, 2012). The equations used in the Bayesian Updating part of the algorithm are analogous with those used by Lloyd and Leslie (2013), to model changes in animal behaviour based on some contextual information about the environment the animals live in, with their rewards following by normal distribution with unknown mean and variance. Note here that, for the stationary case, we do not use any contextual information.

In the case of normal likelihood with both mean μ and precision λ unknown, we use the following conjugate prior:

$$\begin{aligned} \lambda &\sim Ga(\alpha_0, \beta_0) \\ \mu|\lambda &\sim N(\mu_0, \frac{1}{\kappa_0 \lambda}) \end{aligned} \tag{3.3}$$

where μ_0 and κ_0 are the hyperparameters of the prior distribution of μ given λ while α_0 and β_0 are the hyperparameters of λ. Thus, the prior of μ and λ is the Normal-Gamma distribution, $NG(\mu, \lambda; \mu_0, \kappa_0, \alpha_0, \beta_0)$.

In order to obtain the expression for the Normal-Gamma prior we multiply the prior normal distribution for μ given λ with the hyperparameters μ_0 and $\frac{1}{\kappa_0 \lambda}$ by the gamma prior distribution for λ with parameters α_0 and β_0, where $\lambda = 1/\sigma^2$.

Thus,

$$\begin{aligned}
NG(\mu, \lambda; \mu_0, \frac{1}{\kappa_0 \lambda}, \alpha_0, \beta_0) &= N(\mu; \mu_0, \frac{1}{\kappa_0 \lambda}) Ga(\lambda; \alpha_0, \beta_0) \\
&= \frac{1}{\sqrt{\frac{2\pi}{\kappa_0 \lambda}}} \exp\left\{ \frac{-(\mu - \mu_0)^2}{\frac{2}{\kappa_0 \lambda}} \right\} \frac{\beta_0^{\alpha_0} \lambda^{\alpha_0 - 1} e^{-\lambda \beta_0}}{\Gamma(\alpha_0)} \\
&= \frac{\kappa_0^{1/2} \lambda^{1/2}}{\sqrt{2\pi}} \exp\left\{ \frac{\kappa_0 \lambda}{2} (\mu - \mu_0)^2 \right\} \frac{\beta_0^{\alpha_0} \lambda^{\alpha_0 - 1} e^{-\lambda \beta_0}}{\Gamma(\alpha_0)} \\
&= \frac{\kappa_0^{1/2} \lambda^{\alpha_0 - 1/2} \beta_0^{\alpha_0} \exp\left\{ \frac{-\kappa_0 \lambda}{2} (\mu - \mu_0)^2 - \lambda \beta_0 \right\}}{\sqrt{2\pi} \Gamma(\alpha_0)} \\
&= \frac{\sqrt{\kappa_0} \lambda^{\alpha_0 - 1/2} \beta_0^{\alpha_0}}{\sqrt{2\pi} \Gamma(\alpha_0)} \exp\left\{ \frac{-\lambda}{2} \left[\kappa_0 (\mu - \mu_0)^2 + 2\beta_0 \right] \right\}
\end{aligned}$$

Moreover, the posterior distribution can be written as

$$\begin{aligned}
p(\mu, \lambda \mid y) &= p(y | \mu, \lambda) p(\mu, \lambda) \\
&= p(y \mid \mu, \lambda) p(\lambda) p(\mu \mid \lambda) \\
&= N(y \mid \mu, \lambda) Ga(\lambda; \alpha_0, \beta_0) N(\mu; \mu_0, \frac{1}{\kappa_0 \lambda}) \\
&= N(y \mid \mu, \lambda) NG(\mu, \lambda; \mu_0, \kappa_0, \alpha_0, \beta_0)
\end{aligned}$$

where $p(y \mid \mu, \lambda)$ is the normal likelihood of data y. We can write the posterior distribution as

$$p(\mu, \lambda \mid y) = NG(\mu, \lambda; \mu_n, \kappa_n, \alpha_n, \beta_n),$$

where

$$\begin{aligned}
\mu_n &= \frac{\kappa_0 \mu_0 + n\bar{y}}{\kappa_0 + n} \\
\kappa_n &= \kappa_0 + n \\
\alpha_n &= \alpha_0 + \frac{n}{2} \\
\beta_n &= \beta_0 + \frac{1}{2} \sum_{i=1}^{n} (y_i - \bar{y})^2 + \frac{\kappa_0 n (\bar{y} - \mu_0)^2}{2(\kappa_0 + n)}
\end{aligned} \tag{3.4}$$

Note that n is the number of times an option has been tested, $\bar{y}$ the sample mean and $\sum_{i=1}^{n} (y_i - \bar{y})^2$ is the sample sum of squares. Thus, the following are true:

$$\begin{aligned}
\lambda \mid y &\sim Ga(\alpha_n, \beta_n) \\
\mu | \lambda, y &\sim N(\mu_n, \frac{1}{\lambda \kappa_n})
\end{aligned} \tag{3.5}$$

This means that in order to sample from the joint posterior we first draw λ from its marginal posterior distribution $Ga(\lambda; \alpha_n, \beta_n)$ and then draw μ from $N(\mu, 1/\lambda\kappa_n)$.

As we are most interested in the reward of each arm we integrate the prior $p(\mu, \lambda)$ with respect to λ to find the prior marginal on μ, $p(\mu)$, such that

$$\begin{aligned} p(\mu) &\propto \int_0^\infty p(\mu, \lambda) d\lambda \\ &= \int_0^\infty NG(\mu, \lambda; \mu_0, \kappa_0, \alpha_0, \beta_0) d\lambda \\ &= \int_0^\infty \lambda^{\alpha_0 + 1/2 - 1} \exp\Big\{ -\lambda \big[\beta_0 + \frac{\kappa_0(\mu - \mu_0)^2}{2}\big]\Big\} d\lambda \end{aligned}$$

This is the unnormalized gamma distribution with parameters $a = \alpha_0 + \frac{1}{2}$ and $b = \beta_0 + \frac{\kappa_0(\mu - \mu_0)^2}{2}$. So,

$$\begin{aligned} p(\mu) &\propto \frac{\Gamma(a)}{b^a} \\ &\propto b^{-a} \end{aligned} \tag{3.6}$$

Substituting in the values for a and b we get

$$\begin{aligned} p(\mu) &= \left(\beta_0 + \frac{\kappa_0(\mu - \mu_0)^2}{2}\right)^{-\alpha_0 - 1/2} \\ &= \left(1 + \frac{1}{2\alpha_0} \frac{\alpha_0 \kappa_0 (\mu - \mu_0)^2}{\beta_0}\right)^{-(2\alpha_0 + 1)/2} \end{aligned} \tag{3.7}$$

This is a student t-distribution with $2\alpha_0$ degrees of freedom, location μ_0 and scale $\frac{\beta_0}{\alpha_0 \kappa_0}$. So, the marginal prior on μ and its marginal posterior are

$$\begin{aligned} p(\mu) &\sim t_{2\alpha_0}(\mu; \mu_0, \frac{\beta_0}{\alpha_0 \kappa_0}) \\ p(\mu | y) &\sim t_{2\alpha_n}(\mu; \mu_n, \frac{\beta_n}{\alpha_n \kappa_n}) \end{aligned} \tag{3.8}$$

respectively. Moreover, the marginal prior and marginal posterior on λ are

$$\begin{aligned} p(\lambda) &\sim Ga(\lambda; \alpha_0, \beta_0) \\ p(\lambda | y) &\sim Ga(\lambda; \alpha_n, \beta_n) \end{aligned} \tag{3.9}$$

3.3.2 Algorithm

In this section we describe the main steps of Thompson Sampling when the rewards of the different arms follows a normal distribution. At the beginning, we have only limited information about the arms and we choose the same initial prior distribution, $NG(\mu_0, \kappa_0, \alpha_0, \beta_0)$, for all arms. The main steps are explained in Algorithm 3.

Algorithm 3 Thompson Sampling for normal likelihood with unknown mean and variance

```
for each $\tau \in \{1, 2, ..., T\}$ do
    for each arm $a \in \{1, ..., N\}$ do
        Sample reward $\tilde{r}_{a,\tau}$ from the t-distribution $t_{2\alpha_n}(\mu; \mu_n, \frac{\beta_n}{\alpha_n \kappa_n})$
    end for
    Play arm $a_\tau = \underset{a=1,2,...,N}{\text{argmax}} \tilde{r}_{a,\tau}$ and get its real reward $r_{a_\tau}(\tau)$
    Calculate regret $R_\tau = m_{a^*} - m_{a_\tau}$ where $m_{a^*}$ is the expected reward of the
    empirically best option $a^*$ and $m_{a_\tau}$ the expected reward of option $a_\tau$.
    Update the hyperparameters of arm $a_\tau$ using Equations 3.4 where $n$ is the
number of times arm $a_\tau$ has been played and $\bar{y}$ is the mean reward of arm $a_\tau$
calculated as $\bar{y} = \frac{1}{n}\sum_{i=1}^{\tau} r_{a_i}(i)$
end for
```

3.4 Stationary ε-greedy

In the ε-greedy algorithm, $\varepsilon \in [0, 1]$ defines the amount of time spent in exploration. The algorithm chooses at random whether it will explore or exploit, with probability of exploration ε and probability of exploitation $1 - \varepsilon$. Thus, for an experiment with total duration T, εT time is spent on exploration and $(1 - \varepsilon)T$ of time is spent on exploitation. Note that in the exploitation phase the algorithm chooses the arm with the highest observed mean reward, while in the exploration phase the choice of the arm is completely random. The choice of arm is also random at the first time step when the algorithm has no information about the performance of any of the candidate arms. The main steps of the algorithm are given in Algorithm 4.

Algorithm 4 ε-greedy algorithm

```
Define average reward $\bar{y}_a = 0$ and number of times an arm has been chosen $\tilde{c}_a = 0 \; \forall \; a \in \{1, 2, ..., N\}$
Set a value for $\varepsilon$ such that $\varepsilon < 1$
for each $\tau \in \{1, 2, ..., T\}$ do
    if $\tau = 1$ then
        Exploration Phase: Choose a random arm $a_\tau$
    else
        Generate a random number $\varphi_\tau \in [0, 1]$
        if $\varphi_\tau > \varepsilon$ then
            Exploitation Phase: Choose arm $a_\tau = \underset{a=1,2,..,N}{\text{argmax}} \bar{y}_a$
        else
            Exploration Phase: Choose a random arm $a_\tau$
        end if
    end if
    Receive reward $r_{a_\tau}(\tau)$ from the chosen arm
    Update the number of times arm $a_\tau$ has been chosen, $\tilde{c}_{a_\tau} = \tilde{c}_{a_\tau} + 1$
    Update the weighted average of the chosen arm, $\bar{y}_{a_\tau} = \frac{\tilde{c}_{a_\tau} - 1}{\tilde{c}_{a_\tau}} \bar{y}_{a_\tau} + \frac{1}{\tilde{c}_{a_\tau}} r_{a_\tau}(\tau)$
    Calculate regret, $R_\tau = m_{a^*} - m_{a_\tau}$, where $a^*$ is the empirically best arm
end for
```

3.5 Learning and deployment buckets

In this section, we introduce a new algorithm we developed that combines ideas from the two standard algorithms explained above. This new algorithm can be used in website optimization where a company aims to find the most profitable version for their website in terms of revenue earned. It can also be applied to solve problems with similar characteristics, such as stationary reward distribution and constant number of options. In the context of website optimization, the idea is to split the traffic of users that get any of the candidate options into two buckets, deployment and learning, with p and $1 - p$ proportion of traffic sent to each respectively.

The purpose of the learning bucket is to learn about the performance of the different options while in the deployment bucket the same option is being shown (the one with the highest mean observed reward) until a new one becomes the best. The option shown in the deployment bucket changes only when the mean observed reward of a different option gets higher than the mean observed reward of the current option. This may be considered as a modification of the simplistic ε-greedy algorithm, explained earlier, with the amount of exploration being heavily dependent on the proportion of traffic $1 - p$ sent to the learning bucket, instead of it being defined by probability of exploration ε as it happens in ε-greedy. However, in this method we apply a more sophisticated MAB algorithm, Thompson Sampling in the learning bucket in order to get enough information about the performance of the different options and at the same time minimize the regret caused in the learning phase of the algorithm. The main steps of the algorithm are shown in Figure 3.1 and explained in more detail in Algorithm 5. Note that m_{a^*} is the expected reward of the best arm a^* and $m_{a_{\tau,l}}$ and $m_{a_{\tau,d}}$ are the expected reward of the arm chosen at time τ, in the learning or deployment bucket respectively. The update equations used are different and explained in Section 3.5.1.

This new methodology mirrors common practice in industry where a company, while experimenting on different versions of a website, chooses a known good performer to display to a portion of traffic for safety and to avoid a sudden drop in revenues caused by the full experimentation. A similar idea of splitting the traffic into two buckets was considered by Agarwal et al. (2009) who named their learning and deployment buckets random and serving buckets respectively. However, the algorithm used in the random bucket was a simple algorithm very close to ε-greedy. Our aim is to observe whether this idea of splitting the traffic into two buckets can benefit from the implementation of Thompson Sampling.

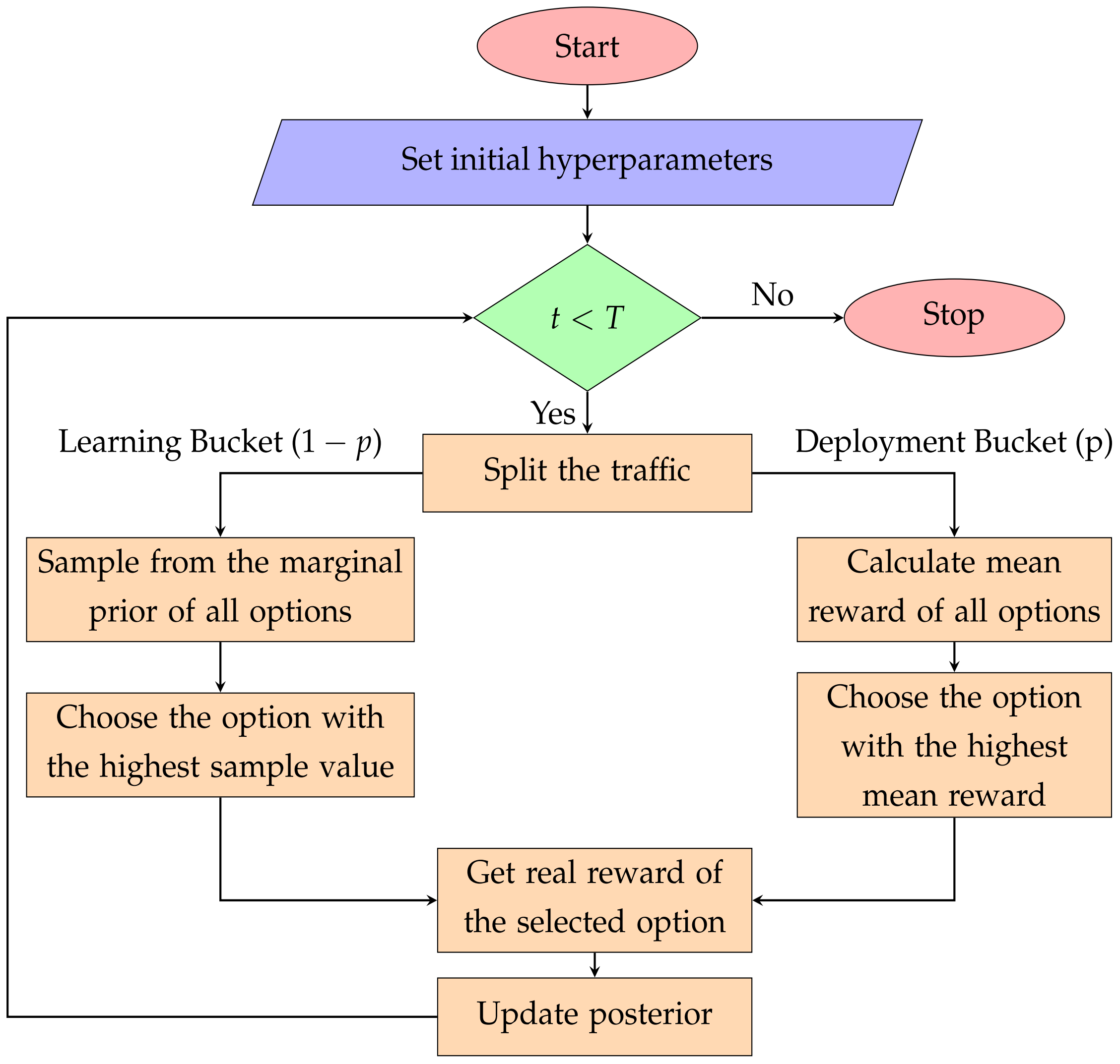

Start
Set initial hyperparameters
$t < T$
No
Stop
Yes
Learning Bucket $(1 - p)$
Split the traffic
Deployment Bucket (p)
Sample from the marginal prior of all options
Calculate mean reward of all options
Choose the option with the highest sample value
Choose the option with the highest mean reward
Get real reward of the selected option
Update posterior

FIGURE 3.1: Flowchart of the main steps of the learning and deployment buckets algorithm

3.5.1 Bayesian analysis

Since Thompson Sampling, which is employed in the learning bucket, uses the information collected from both buckets, it is important to adapt the update equations in order to take into account the reward obtained at each time step from each bucket accounting for the proportion of traffic sent to each bucket. For the implementation of the algorithm, we use the idea of pooled variances in order to make the necessary amendments to the equations for the update of the hyperparameters.

We collect data on the total revenue, X, obtained by a website each day. This follows a normal distribution with mean μ and variance σ^2. If we split the traffic into two buckets, a deployment bucket and a learning bucket, then we can define two new random variables x_d and x_l, respectively. Thus, the total revenue is $x_d +$

x_l. We assume that we send a proportion p of the traffic to the deployment bucket and $1-p$ to the learning bucket.

In order to simulate the output appropriately for x_d and x_l, we use the following:

$$x_d + x_l \sim N(\mu, \sigma^2).$$

If we assume that x_d and x_l are independent, then $x_d \sim N(p\mu, p\sigma^2)$ and $x_l \sim N((1-p)\mu, (1-p)\sigma^2)$. This follows the argument of pooled variances.

Moreover, using $\lambda = \frac{1}{\sigma^2}$ the likelihood is given by

$$L = \prod_{i=1}^{d} \sqrt{\frac{\lambda}{2\pi p}} \exp\left[\frac{-\lambda}{2p}(x_i - p\mu)^2\right] \prod_{j=1}^{l} \sqrt{\frac{\lambda}{2\pi(1-p)}} \exp\left[\frac{-\lambda}{2(1-p)}(y_j - (1-p)\mu)^2\right]$$

where x are the data obtained from the deployment bucket and y the data obtained from the learning bucket $\forall i \in 1,2,..d$ and $\forall j \in 1,2,..l$ respectively. The prior is:

$$\mathrm{p}(\mu, \lambda) = \sqrt{\frac{\kappa_0 \lambda}{2\pi}} \exp\left[-\frac{\kappa_0 \lambda}{2}(\mu - \mu_0)^2\right] \frac{\beta_0^{\alpha_0} \lambda^{\alpha_0 - 1} e^{-\lambda \beta_0}}{\Gamma(\alpha_0)}.$$

Multiplying these together and taking logs, we get the following expression for posterior, where [consts] is made up of terms that do not involve λ or μ:

$$\begin{aligned} \mathrm{Log}\ \mathrm{p}(\mu, \lambda | x, y) \quad = \quad & [\text{consts}] + \frac{1}{2}\ln(\lambda) - \frac{\kappa_0 \lambda}{2}(\mu - \mu_0)^2 + (\alpha_0 - 1)\ln\lambda - \lambda\beta_0 \\ & + \sum_{i=1}^{d}\left[\frac{1}{2}\ln\lambda - \frac{\lambda}{2p}(x_i - p\mu)^2\right] \\ & + \sum_{j=1}^{l}\left[\frac{1}{2}\ln\lambda - \frac{\lambda}{2(1-p)}(y_j - (1-p)\mu)^2\right]. \end{aligned}$$

In the next step we expand the sums and match similar terms before completing the square in order to write down the updated parameters based on the d observations of the deployment bucket to which a proportion p of the traffic is sent and the l observations of the learning bucket to which a proportion $1-p$ of the traffic is sent.

We find that

$$\begin{aligned} \tilde{\mu} = \quad & \frac{\kappa_0 \mu_0 + d\bar{x} + l\bar{y}}{\kappa_0 + dp + l(1-p)} \\ \tilde{\kappa} = \quad & \kappa_0 + dp + l(1-p) \\ \tilde{\alpha} = \quad & \alpha_0 + (d + l - 1)/2 \end{aligned} \tag{3.10}$$

where $\bar{x}$ is the mean reward observed in the deployment bucket and $\bar{y}$ the mean reward in the deployment bucket. In order to find $\tilde{\beta}$ we make use of the fact that

$$d^2\bar{x}^2 - d\sum_{i=1}^{d} x_i^2 = -d\sum_{i=1}^{d}(x_i - \bar{x})^2$$

(and similar for y). Then, we can simplify the expression to the following.

$$\begin{aligned}\tilde{\beta} &= \beta_0 + \frac{d\sum_{i=1}^{d}(x_i-\bar{x})^2 + l\sum_{j=1}^{l}(y_j-\bar{y})^2 + \frac{l(1-p)}{p}\sum_{i=1}^{n}x_i^2 + \frac{dp}{1-p}\sum_{j=1}^{l}y_j^2 - 2dl\bar{x}\bar{y}}{2(dp+l(1-p))} \\ &\quad + \frac{\kappa_0(dp+l(1-p))}{2(\kappa_0+dp+l(1-p))}\left(\mu_0 - \frac{d\bar{x}+l\bar{y}}{dp+l(1-p)}\right)^2.\end{aligned}$$

We can take this one step further by focusing on the top line of this equation. We expand the brackets again using

$$\sum_{i=1}^{d}(x_i-\bar{x})^2 = \sum_{i=1}^{d}x_i^2 - d\bar{x}^2.$$

This allows us, after some manipulation to write

$$\begin{aligned}&\frac{d\sum_{i=1}^{n}(x_i-\bar{x})^2 + l\sum_{j=1}^{l}(y_j-\bar{y})^2 + \frac{l(1-p)}{p}\sum_{i=1}^{n}x_i^2 + \frac{dp}{1-p}\sum_{j=1}^{l}y_j^2 - 2dl\bar{x}\bar{y}}{2(dp+l(1-p))} = \\ &\qquad (1/2) * \left[\frac{1}{p}\sum_{i=1}^{d}x_i^2 + \frac{1}{1-p}\sum_{j=1}^{l}y_j^2 - \frac{(d\bar{x}-l\bar{y})^2}{dp+l(1-p)}\right].\end{aligned}$$

Let's call the expression above A for convenience. Then,

$$\begin{aligned}2A &= \frac{1}{p}\sum_{i=1}^{d}x_i^2 + \frac{1}{1-p}\sum_{j=1}^{l}y_j^2 + \frac{(dp+l(1-p))(d\bar{x}+l\bar{y})^2}{(dp+l(1-p))^2} - 2\frac{(d\bar{x}+l\bar{y})^2}{(dp+l(1-p)} \\ &= p\sum_{i=1}^{d}\left(\frac{x_i}{p} - \frac{d\bar{x}+l\bar{y}}{dp+l(1-p)}\right)^2 + (1-p)\sum_{j=1}^{l}\left(\frac{y_j}{1-p} - \frac{d\bar{x}+l\bar{y}}{dp+l(1-p)}\right)^2.\end{aligned}$$

This final expression gives us the weighted sum of the squared differences between the observations in each of the buckets and the weighted sum of the means from the two buckets. An alternative representation takes out a factor of $1/p^2$ or $1/(1-p)^2$ respectively, to give

$$2A = \frac{1}{p}\sum_{i=1}^{d}\left(x_i - \frac{p(d\bar{x}+l\bar{y})}{dp+l(1-p)}\right)^2 + \frac{1}{(1-p)}\sum_{j=1}^{l}\left(y_j - \frac{(1-p)(d\bar{x}+l\bar{y})}{dp+l(1-p)}\right)^2.$$

So,

$$\tilde{\beta} = \beta_0 + \frac{1}{2}\left[\frac{1}{p}\sum_{i=1}^{d}\left(x_i - \frac{p(d\bar{x}+l\bar{y})}{dp+l(1-p)}\right)^2 + \frac{1}{(1-p)}\sum_{j=1}^{l}\left(y_j - \frac{(1-p)(d\bar{x}+l\bar{y})}{dp+l(1-p)}\right)^2\right] + \frac{\kappa_0(dp+l(1-p))}{2(\kappa_0+dp+l(1-p))}\left(\mu_0 - \frac{d\bar{x}+l\bar{y}}{dp+l(1-p)}\right)^2. \tag{3.11}$$

3.5.2 Algorithm

Below we explain the main step of the learning and deployment buckets method:

Algorithm 5 Learning and deployment buckets

```
Set initial values of hyperparameters $\mu_0$, $\kappa_0$, $\alpha_0$ and $\beta_0$
Set $1-p$ the amount of traffic sent to the learning bucket where we get $l$ observations and $p$ the amount of traffic sent to the deployment bucket where we get $d$ observations.
for each $\tau \in \{1, 2, ..., T\}$ do
    Learning bucket:
    Sample reward $\tilde{r}_{a,\tau}$ from the t-distribution $t_{2\tilde{\alpha}}(\mu; \tilde{\mu}, \frac{\tilde{\beta}}{\tilde{\alpha}\tilde{\kappa}})$ for all arms
    Play arm $a_{\tau,l} = \underset{a=1,2,..,N}{\operatorname{argmax}}(\tilde{r}_{a,\tau})$ and get its real reward $r_{a_{\tau,l}}(\tau)$
    Calculate regret $R_{\tau,l} = (m_{a^*} - m_{a_{\tau,l}})(1-p)$
    Deployment Bucket:
    Calculate mean reward of all arms $\bar{r}_{a,\tau}$ $\forall a \in 1, ...N$ .
    Choose arm with the highest mean reward $a_{\tau,d} = \underset{a=1,2,3,...N}{\operatorname{argmax}} \bar{r}_{a_\tau}$
    Receive real reward $r_{a_{\tau,d}}(\tau)$ of the chosen arm
    Calculate regret $R_{\tau,d} = (m_{a^*} - m_{a_{\tau,d}})p$
    Calculate total regret $R_\tau = R_{\tau,l} + R_{\tau,d}$
    Update the hyperparameters of arm $a_{\tau,l}$ using the new update Equations 3.10, 3.11
end for
```

3.6 Numerical experiments on real-world datasets

The case study uses simulated data generated by an online travel agency to mimic real data for the case of seven candidate options that correspond to different versions of their website. In the first dataset the mean performance of the seven options is very similar, while in the second dataset there is a "clear winner" between the options. The real mean value m and standard deviation s for the options in the two different datasets are presented in the tables below. In Table 3.2, Option 3 appears as the best option, since it has the highest mean value, but the differences between the means of all options are relatively small. However, in Table 3.3, it is clear that the best option is Option 2 with m_2 being much greater than the mean value of the other options. Note here that even though m_2 is the largest mean value, s_2 is also very large which means that in practice the reward varies more from one time step to the next, making it even harder for the algorithm to decide which option is best.

Testing the algorithms on two different datasets will give us enough information in order to make some first initial conclusions about their performance and how this is affected by several parameters. For example, it has been suggested that in Thompson Sampling, the choice of priors is very important and can cause significant changes to the performance of the algorithm (Honda and Takemura, 2014; Russo et al., 2018; Griffin and Brown, 2010). For example, when Chapelle and Li (2011) studied the problem of binary likelihood they used the conjugate Beta distribution as prior and they examined posterior reshaping by changing the parameters of the posterior Beta(a, b) distribution into Beta$(\frac{a}{\alpha}, \frac{b}{\alpha})$ in order to favour either exploration or exploitation. In particular, they discuss that depending on the value of α, a wider posterior results in more exploration while a tighter posterior causes more exploitation. In the case of ε-greedy, it has also been noted that its performance depends a lot on the value of ε which defines the amount of exploration (White, 2012). In our case, our focus is on determining the impact that the values of the prior hyperparameters μ_0, κ_0, α_0, β_0 have on the performance of the Thompson Sampling algorithm and comparing these results with those obtained using an ε-greedy algorithm. In order to examine this dependence, we conducted similar experiments on the two different real-world datasets.

First, we present the results obtained from the first dataset where the seven options have similar performance and then the algorithms are tested in the second dataset where there is a clear winner. For each example different parameters are being tested in order to get the optimal possible results from each algorithm and a comparison between the two methods is made. As measures of performance we use the percentage of optimal selection and cumulative regret. The results are based on 100 independent runs for each algorithm.

Dataset 1

a	1	2	3	4	5	6	7
m_a	1040	1946	2176	1752	1957	1494	1922
s_a	328	1129	239	553	720	586	501

TABLE 3.2: Mean m_a and standard deviation s_a for the "real system" when $N = 7$ in Dataset 1

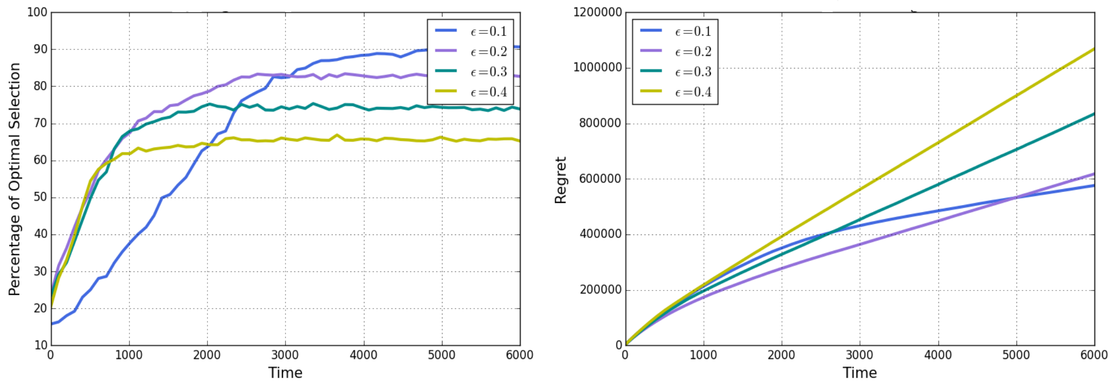
Dataset 2

a	1	2	3	4	5	6	7
m_a	2111	6024	1049	1381	1841	1470	1704
s_a	1230	3781	832	743	1023	1380	1194

TABLE 3.3: Mean m_a and standard deviation s_a of the "real system" when $N = 7$ in Dataset 2

3.6.1 Results for Dataset 1

First, we apply ε-greedy and we test its performance against parameter ε, which is basically the percentage of exploration, when the performance of the different options in the "real system" are very similar. Figure 3.2a and Figure 3.2b illustrate how the two measures of performance are affected by ε. Note here that even though the optimal selection reaches a higher final percentage when $\varepsilon = 0.1$, it takes more time for the algorithm to reach a constant amount of optimal selection when compared to higher values of ε. On the other hand, when $\varepsilon = 0.4$, even though the percentage of optimal selection reaches 65% relatively early, it stays at the same level for the rest of the experiment. Consequently, its cumulative regret keeps increasing linearly for the whole time of the experiment. These results indicate that with a high percentage of exploration the algorithm finds the best option quickly but afterwards it keeps loosing rewards due to unnecessary exploration.



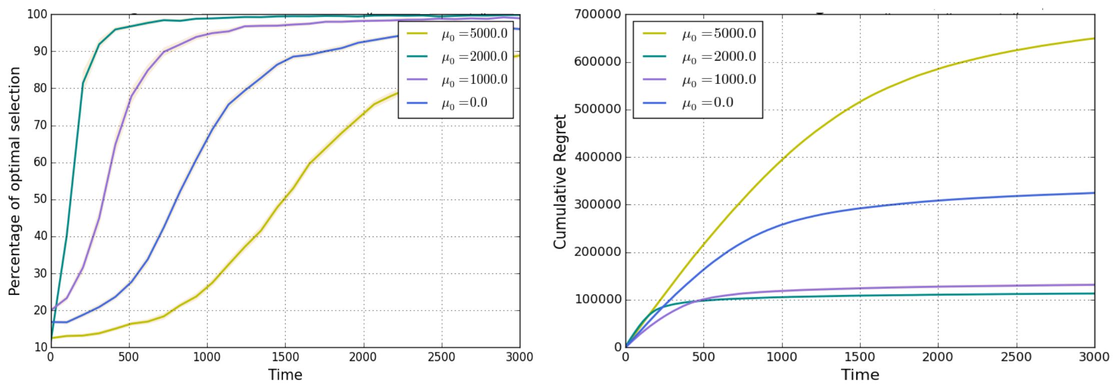
(A) (B)

FIGURE 3.2: Impact of varying ε on percentage of optimal selection and cumulative regret in the stationary case when $N = 7$ i Dataset 1. The curves are averages over 100 runs.

Percentage of optimal selection				
	$t = 1000$	$t = 2000$	$t = 3000$	$t = 4000$
$\varepsilon = 0.1$	40%	60%	82%	90%
$\varepsilon = 0.2$	70%	80%	82%	82%
$\varepsilon = 0.3$	70%	75%	75%	75%
$\varepsilon = 0.4$	60%	65%	65%	65%

TABLE 3.4: Percentage of optimal selection at different time steps t using ε-greedy with different ε

In Figures 3.3, we show the performance of Thompson Sampling under different values of μ_0, i.e.the initial estimate of the mean reward, when κ_0, α_0, β_0 are equal to 1. Both measures of performance are heavily dependent on μ_0 with the percentage of optimal selection reaching almost 100% when $\mu_0 = 2000$ at $t = 400$ while in the case of $\mu_0 = 5000$ it increases with a much smaller rate. This is also supported by Figure 3.3b which shows that $\mu_0 = 2000$ and $\mu_0 = 1000$ cause very similar rate of increase in the cumulative regret which reaches a constant value at $t = 200$ and $t = 500$ respectively, while $\mu_0 = 0$ and $\mu_0 = 5000$ have much higher cumulative regret. Then we proceed with varying parameter α_0, i.e. the degrees of freedom for the variance, with $\kappa_0 = 1$ and $\beta_0 = 1$ when μ_0 has the two best values as shown earlier.

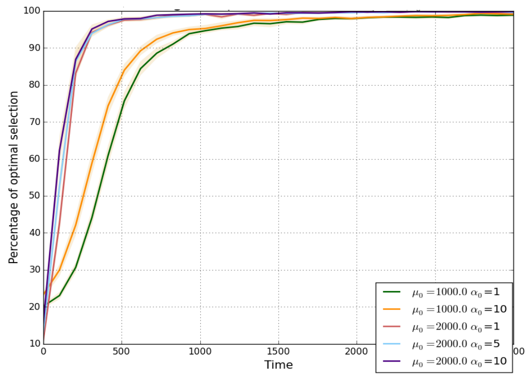


(A) (B)

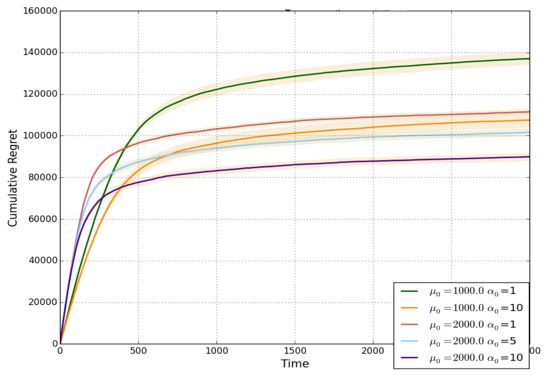
FIGURE 3.3: Impact of varying μ_0 when κ_0, α_0, β_0 are equal to 1 on percentage of optimal selection and cumulative regret in the stationary case when $N = 7$ in Dataset 1. The curves are averages over 100 runs.

Figure 3.4a shows that percentage of optimal selection is not very sensitive to the value of α_0. However, a smaller value of α_0 when $\mu_0 = 1000$ causes greater regret, while in the case of $\mu_0 = 2000$ cumulative regret is not affected that much by the value of α_0. Note that in all cases percentage of optimal selection reaches almost 100% before $t = 1000$, while in the case where $\mu_0 = 2000$ it reaches 100% at around $t = 500$.

As a final step of the sensitivity analysis, we tried different κ_0 and β_0 to test the dependence of the performance on the parameters. From Figure 3.5b, it is clear that a greater value of κ_0 causes a large increase in the regret while a change in β_0 does not affect the results. Figure 3.6 shows the decisions made at each time step and illustrates how exploration and exploitation are distributed over the total time of the experiment.

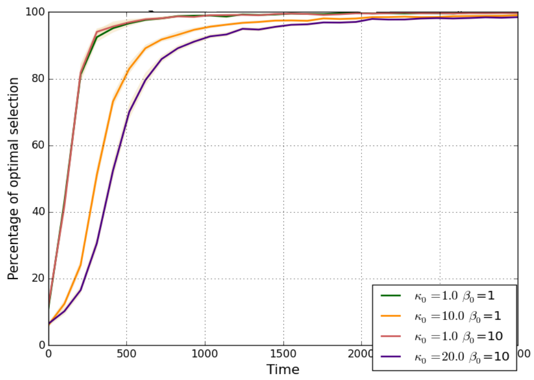


(A)

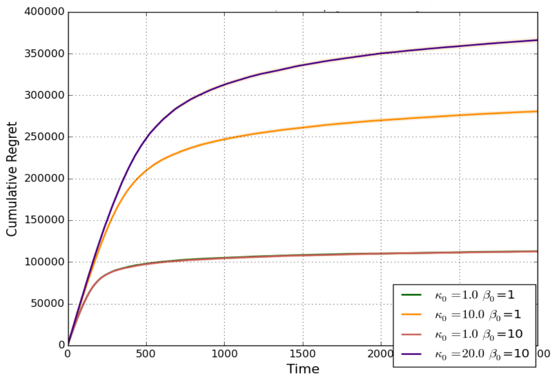


(B)

FIGURE 3.4: Impact of varying μ_0 and α_0 when κ_0 and β_0 are equal to 1 on the percentage of optimal selection and cumulative regret in the stationary case when $N = 7$ in Dataset 1. The curves are averages over 100 runs.

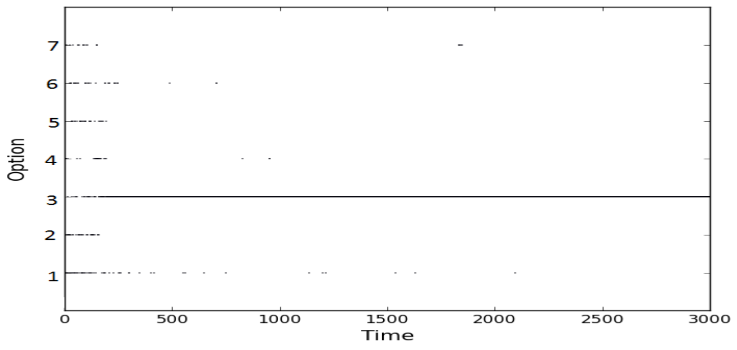


(A)

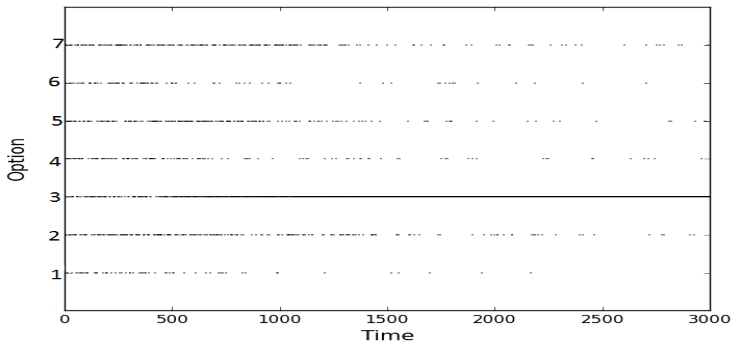


(B)

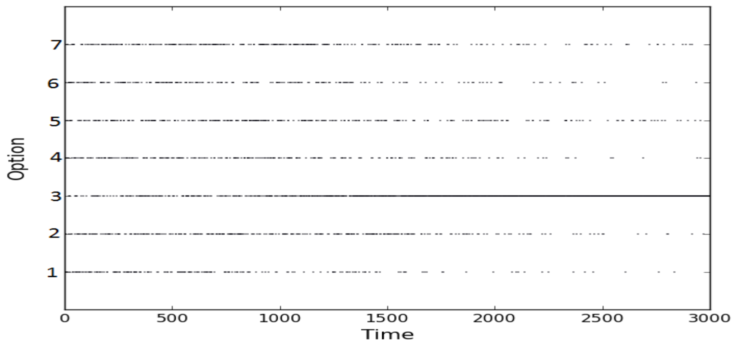
FIGURE 3.5: Impact of varying κ_0 and β_0 when $\mu_0 = 2000$ and $\alpha_0 = 1$ on the percentage of optimal selection and cumulative regret in the stationary case when $N = 7$ in Dataset 1. The curves are averages over 100 runs.



(A)



(B)

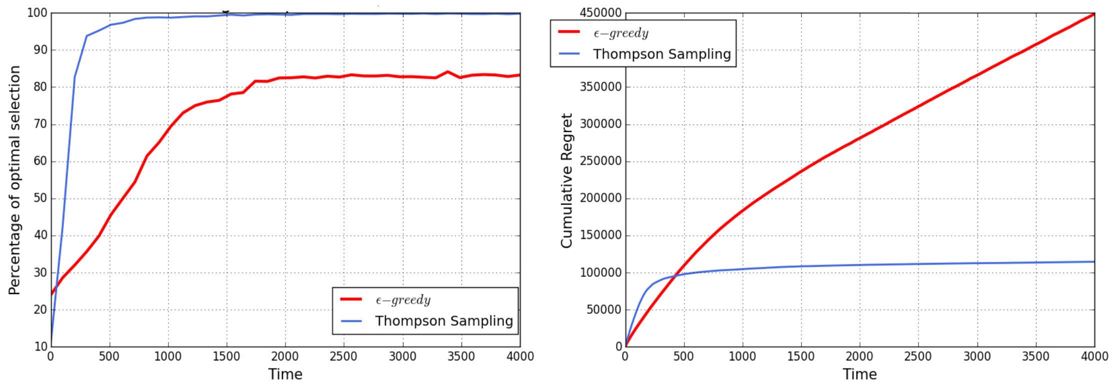


(C)

FIGURE 3.6: The decisions made by the algorithm on a single run when $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$. The dots indicate the option chosen at each time step t when (a) $\mu_0 = 2000$: The algorithm spends just 7% of the time in exploration and 93% in exploitation. (b) $\mu_0 = 0$: The algorithm spends 31% in exploration and 69% in exploitation. (c) $\mu_0 = 5000$: The algorithm spends 51% in exploration and 49% in exploitation.

Figure 3.7 shows the performance of ε-greedy against Thompson Sampling on the same dataset. In order to make a fair comparison we tested the algorithms using the best parameters found in the experiments above; ε-greedy with $\varepsilon = 0.2$

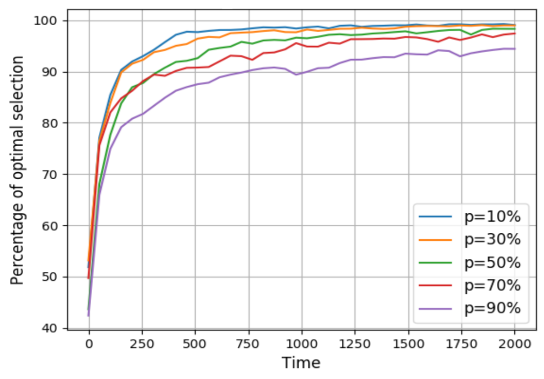
is compared with Thompson Sampling with $\mu_0 = 2000$, $\kappa_0 = 1$, $\alpha_0 = 1$ and $\beta_0 = 1$. In terms of both measures of performance it is clear that Thompson Sampling performs much better than ε-greedy.



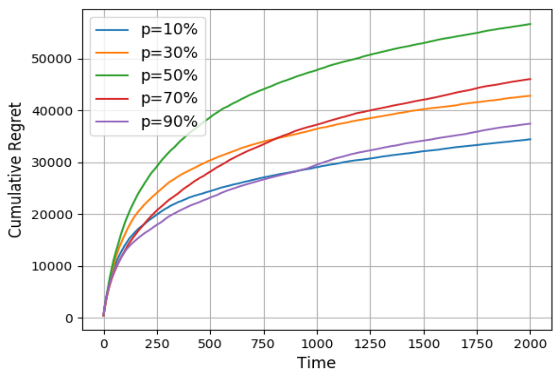
(A) (B)

FIGURE 3.7: Comparison between ε-greedy with $\varepsilon = 0.2$ and Thompson Sampling with $\mu_0 = 2000$, $\kappa_0 = 1$, $\alpha_0 = 1$ and $\beta_0 = 1$ in the stationary case when $N = 7$ in Dataset 1. The curves are averages over 100 runs.

Finally, in order to observe the effect that the proportion of traffic sent to each bucket has on the performance of the learning and deployment buckets algorithm, in Figure 3.8 we study its performance for different values of p. Looking at the percentage of optimal selection we observe that as p increases, the final percentage of optimal selection decreases with $p = 10\%$ reaching the highest percentage of optimal selection. Its cumulative regret is also the lowest but very close to the cumulative regret when $p = 90\%$ that never reaches a 100% of optimal selection.



(A)

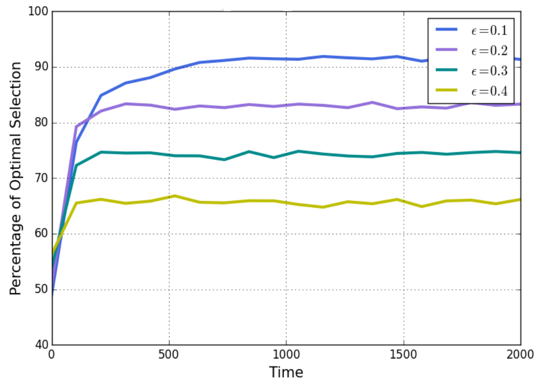


(B)

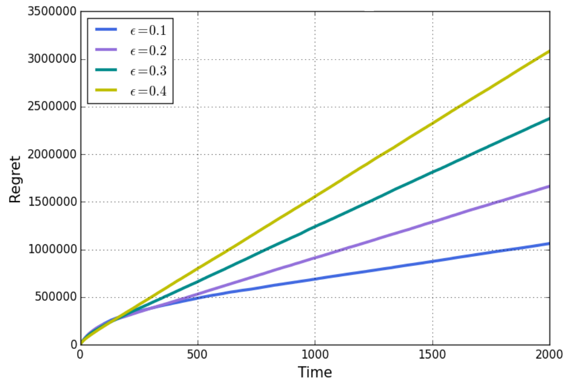
FIGURE 3.8: Performance of learning and deployment buckets algorithm when $\mu_0 = 2000$, $\kappa_0 = 1$, $\alpha_0 = 1$ and $\beta_0 = 1$ for different percentages of p, when $N = 7$ in Dataset 1. The curves are averages over 100 runs.

3.6.2 Results for Dataset 2

In this section, we perform similar experiments on a situation where there is a clear winner between the candidate options but greatest variance. Figure 3.9 shows that, for all values of ε, the algorithm reaches a constant amount for optimal selection almost at the same time, with $\varepsilon = 0.4$ and $\varepsilon = 0.1$ reaching a final rate of 65% and 91%, respectively. For all values of ε, cumulative regret starts increasing linearly very early in the experiment. This is because it never converges to a 100% of optimal selection and settles at a lower percentage very quickly.



(A)

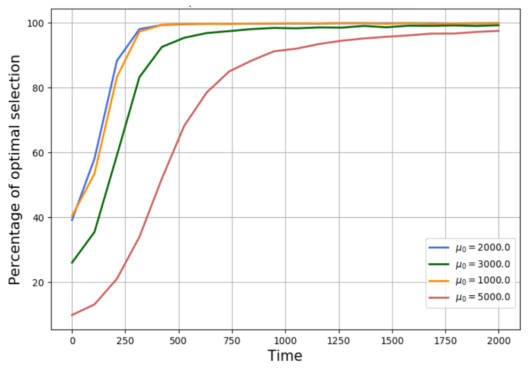


(B)

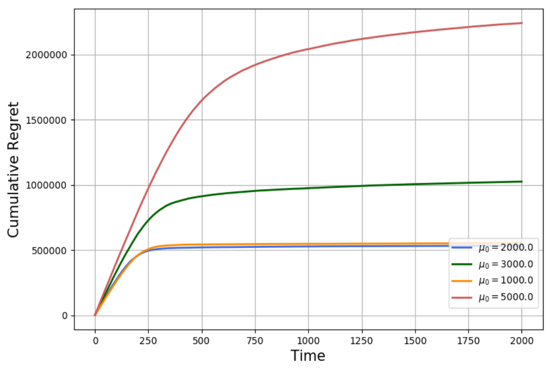
FIGURE 3.9: Impact of varying ε on the percentage of optimal selection and cumulative regret in the stationary case when $N = 7$ in Dataset 2. The curves are averages over 100 runs.

Then, similarly to Section 3.6.1, we perform some experimentation on the initial hyperparameters of Thompson Sampling in Dataset 2. In Figure 3.10 we try different μ_0 to test its impact on the results. We can see that the results agree with those for Dataset 1. The best performance for both percentage of optimal selection and regret is when $\mu_0 = 2000$, i.e.closer to the true mean. The algorithm converges towards 100% of optimal selection at $t = 500$ with the cumulative regret reaching stability.

Based on the results obtained for Dataset 1, we know that α_0 does not affect the performance of the algorithm. So, in the next test, we try different values for β_0 and κ_0 when $\mu_0 = 2000$ and $\alpha_0 = 1$. From Figures 3.11, we see once again that β_0 does not affect the results. At the same time, a larger value for κ_0 makes the algorithm settle on the optimal option later, resulting in greater cumulative regret at the end of the experiment. Thus, a good choice of initial hyperparameters is the same as in Example 1: $\mu_0 = 2000$, $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$. Then, we perform a comparison between ε-greedy with $\varepsilon = 0.1$ and Thompson Sampling with$\mu_0 = 2000$, $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$. In Figures 3.12, it is clear the Thompson Sampling again performs much better that ε-greedy over the course of the full experiment. However, we note that for $t < 150$ ε-greedy chooses the best option more often than Thompson Sampling and as a consequence, until $t = 500$ ε greedy has smaller cumulative regret.

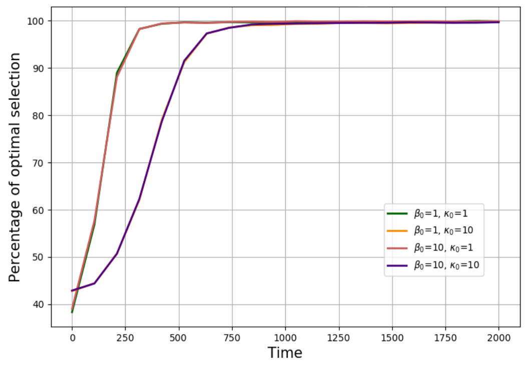


(A)

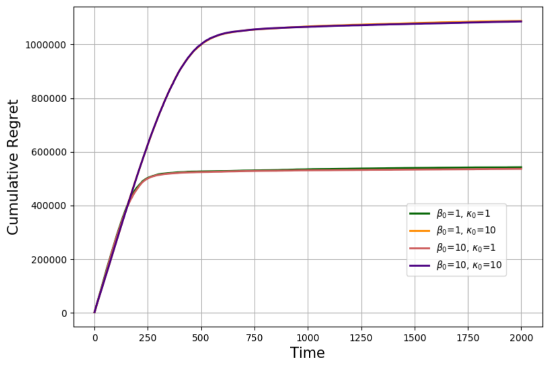


(B)

FIGURE 3.10: Impact of varying μ_0 on percentage of optimal selection and cumulative regret when $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$ in the stationary case when $N = 7$ in Dataset 2. The curves are averages over 100 runs.

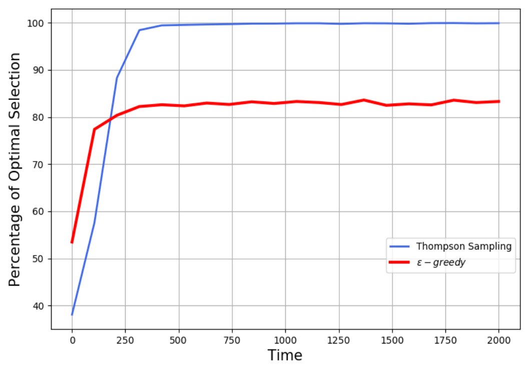


(A)

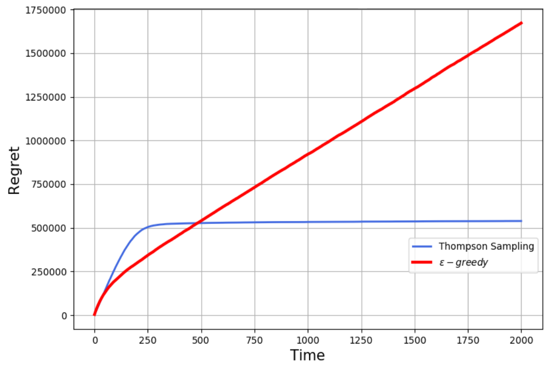


(B)

FIGURE 3.11: Impact of varying β_0 and κ_0 when $\mu_0 = 2000$ and $\alpha_0 = 1$ on percentage of optimal selection and cumulative regret in the stationary case when $N = 7$ in Dataset 2. The curves are averages over 100 runs.



(A)



(B)

FIGURE 3.12: Comparison between ε-greedy with $\varepsilon = 0.1$ and Thompson Sampling with $\mu_0 = 2000$, $\kappa_0 = 1$, $\alpha_0 = 1$ and $\beta_0 = 1$ in the stationary case when $N = 7$ in Dataset 2. The curves are averages over 100 runs.

Finally, in Figure 3.13 we show the performance of learning and deployment buckets algorithm for different values of p. Once again, we observe that the algorithm does not manage to reach a 100% optimal selection for any value of p% greater than 10%.

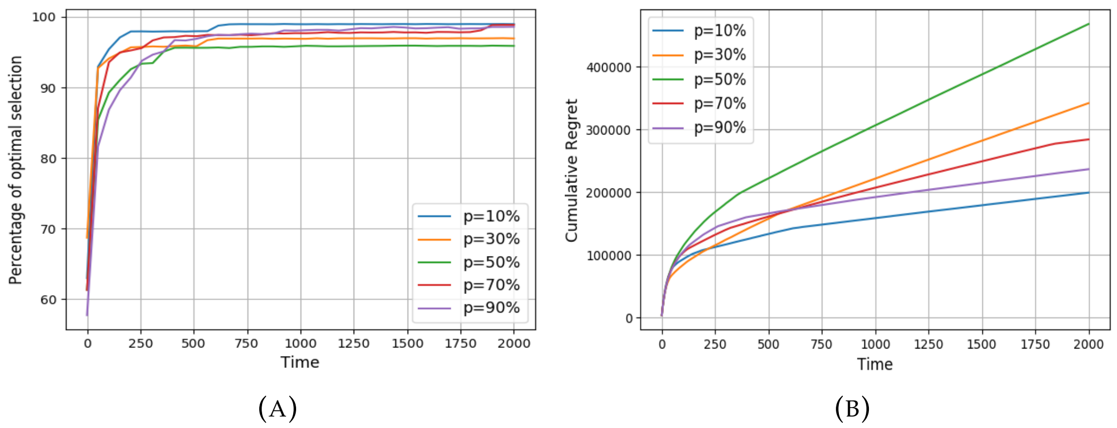


FIGURE 3.13: Performance of learning and deployment buckets algorithm when $\mu_0 = 2000$, $\kappa_0 = 1$, $\alpha_0 = 1$ and $\beta_0 = 1$ for different percentages of p, when $N = 7$ in Dataset 2. The curves are averages over 100 runs.

3.6.3 Discussion

In the results presented above, it is clear that the performance of ε-greedy is heavily dependent on the value of ε. This is one of the major drawbacks of the method since defining ε at the beginning of the experiment means that we set the amount of exploration before we have enough information about the options we are comparing. For this reason ε- decreasing, with $\varepsilon_t = \min\left\{1, \frac{\varepsilon_0}{t}\right\}$, where $\varepsilon_0 > 0$, so that $\varepsilon \to 0$ as $t \to T$, is sometimes used instead (Vermorel and Mohri, 2005). This is meant to avoid wasting time in exploring options that have performed badly in the past. However, even with this algorithm, we still need to set an initial ε_0.

One important conclusion we can draw about the performance of the ε-greedy algorithm, is that, in the case where the rewards of the different options are very similar, the time T over which we can run the experiment is also very important. If ε is high, this means that we allow the algorithm to do more exploration which leads to smaller percentage of optimal selection and consequently to a greater increase in the cumulative regret. For the first dataset, $\varepsilon = 0.1$ makes the algorithm choose randomly for around 10% of time T and it reaches a final 90% of optimal selection after 3000 time steps. However, if ε is set to be 0.2 the algorithm reaches a 70% of optimal selection much earlier (at $t = 1000$) leading to less cumulative regret. Note here that, due to its design, the algorithm will never reach a 100% optimal selection. Consequently, even though the rate of increase in cumulative regret gets smaller, it never stops increasing. This is also true for the case where there is a clear winner in the system (i.e. Dataset 2). Since the mean value of the best of option is much higher than the rest, with the same amount of exploration (i.e same ε), the algorithm reaches a higher percentage of optimal selection much earlier, when compared to the first example, but it again never reaches 100%.

In the case of Thompson Sampling, μ_0 is an estimation of the true mean value of the options, κ_0 is the uncertainty about μ_0, α_0 is the degrees of freedom for the variance and β_0 is the scale of variance. After the sensitivity analysis performed in the last section on Dataset 1 and Dataset 2, we conclude that a good choice of initial hyperparameters for both situations is $\mu_0 = 2000$, $\kappa_0 = 1$, $\alpha_0 = 1$ and $\beta_0 = 1$. With μ_0 very close to the true mean of the options as given in Table 3.2 and Table 3.3, the algorithm needs less time to estimate the true mean of the options. This is also supported by the calculation of β in equations 3.4. From the third term, we can see that the difference between the mean value of our samples and μ_0, $(\bar{y} - \mu_0)$ plays an important role in the whole Bayesian updating procedure. A large difference implies that our initial estimation of the mean is far away from the real one and this results in larger β and consequently larger variance for σ^2.

Similar tests on the other hyperparameters suggest that changes caused by different α_0 and β_0 were insignificant. However, a higher value of κ_0 implies greater uncertainty about the initial estimate of the mean performance and consequently more time is needed for the algorithm to find the best option. This is again explained by the update equations since κ_0 is the factor which is multiplied by $(\mu - \mu_0)$ in the expression for the marginal prior of μ (Equation 3.6). So a combination of bad choices for μ_0 and κ_0 can severely reduce the efficiency of the algorithm. A choice of hyperparameters close to the true values implies more knowledge about the performance of the different options which reduces the learning part of the algorithm and consequently its overall performance (Russo et al., 2018). In the results presented we observed that a μ_0 greater than the mean of the observed rewards caused a greater increase in the amount of exploration in both examples and this affected its performance negatively.

Finally, a comparison between ε-greedy and Thompson Sampling shows clearly that even with a good choice for ε, Thompson Sampling outperforms the simple ε-greedy whose decisions made in the exploration phase are made completely at random. Motivated by the results obtained when testing the algorithms on two real-world datasets, we will attempt to test the impact that the characteristics of the data have on the performance of the algorithm further, by creating artificial datasets and performing controlled experiments. For the learning and deployment buckets algorithm we choose to test values for p that are less than 50% because in the experiments presented in this chapter, it is clear that as p increases the final percentage of optimal selection decreases and consequently cumulative regret keeps increasing.

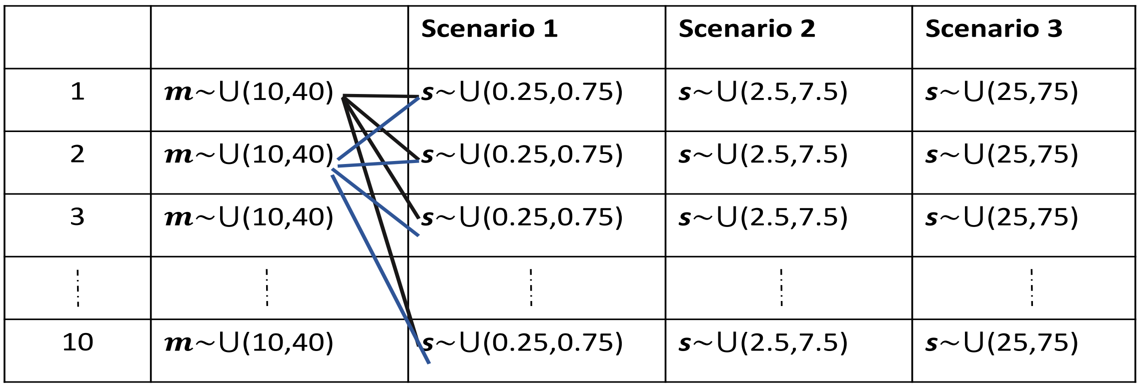
3.7 Numerical experiments on artificial datasets

3.7.1 Setting up the artificial datasets

This section aims to investigate further the performance of the algorithms explained earlier, compare them and examine their dependence on the environment they are tested on. For this reason, we create a number of artificial datasets in order to test the algorithms on a wide range of different situations that satisfy our basic underlying assumptions, i.e. stationary and normal reward distributions with unknown mean and variance. To our knowledge, most of the practical implementations of both ε-greedy and Thompson Sampling present results obtained using specific examples and parameter values for the representation of the real performance of the different options (Koulouriotis and Xanthopoulos, 2008; Chapelle and Li, 2011) and there is not enough research on the effect that the real system has on the performance of the MAB algorithms.

What we are most interested in is how the variability of rewards for the different options affects the overall performance of the algorithms. This is because we noticed that this was the most important factor that affected the results of the experiments we did using the real-world datasets. As we did in the previous section, we continue to consider an example in which we compare the rewards obtained from seven arms. We admit that the choice of seven is somewhat arbitrary but is within the likely range of options being considered and is not too large nor too small.

To get the parameters, m and s, that represent the real mean and the real standard deviation for the seven options, respectively, we sample from a Uniform distribution. For the real mean of the seven options, we sample seven values from $U(10, 40)$, while for their real standard deviation, we create three different scenarios represented by three different sets of parameters for the Uniform distribution,representing low, moderate and high variability respectively. First, for the case of low variability in the rewards, we sample seven values for the standard deviation from $U(0.25, 0.75)$; we call this Scenario 1. For moderate variability, we sample seven values from $U(2.5, 7.5)$ (Scenario 2) and finally, for a high variability in the rewards, we sample seven values from $U(25,75)$ (Scenario 3). This procedure is repeated ten times so that we get ten different sets for the m parameters of the seven arms and ten different sets that represent the s parameters of the seven arms for each scenario. An overview of how we create the artificial datasets is presented in Table 3.14. Each **m** is paired with each **s** from each Scenario to create a total of 300 different datasets. Each algorithm is tested on 100 different datasets for each scenario, resulting in a total of 300 different examples.



		Scenario 1	**Scenario 2**	**Scenario 3**
1	$\boldsymbol{m}$~U(10,40)	$\boldsymbol{s}$~U(0.25,0.75)	$\boldsymbol{s}$~U(2.5,7.5)	$\boldsymbol{s}$~U(25,75)
2	$\boldsymbol{m}$~U(10,40)	$\boldsymbol{s}$~U(0.25,0.75)	$\boldsymbol{s}$~U(2.5,7.5)	$\boldsymbol{s}$~U(25,75)
3	$\boldsymbol{m}$~U(10,40)	$\boldsymbol{s}$~U(0.25,0.75)	$\boldsymbol{s}$~U(2.5,7.5)	$\boldsymbol{s}$~U(25,75)
⋮	⋮	⋮	⋮	⋮
10	$\boldsymbol{m}$~U(10,40)	$\boldsymbol{s}$~U(0.25,0.75)	$\boldsymbol{s}$~U(2.5,7.5)	$\boldsymbol{s}$~U(25,75)

FIGURE 3.14: Overview of setting up the artificial datasets. $\boldsymbol{m}$ and **s** are sets with seven values representing the real mean and the standard deviation respectively for each of the seven options. Each **m** is paired with each **s** from each Scenario to create a total of 300 different datasets.

3.7.2 Experimental framework

Firstly, we present the results obtained from the implementation of Thompson Sampling, ε-greedy and the learning and deployment buckets algorithm at the end of the experiment, $t = 2000$. The parameters to be tested for the learning and deployment buckets algorithm are the value of μ_0 and the proportion of traffic sent to the deployment bucket, p. For both Thompson Sampling and learning and deployment buckets algorithm we do not experiment on the other three hyperparameters since μ_0 had the greatest impact on the performance of the algorithm when tested on the two real-world datasets. We keep the rest of the hyperparameters the same; $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$. The choice of μ_0 values was based on the parameters used when creating the artificial datasets as described above. The three values tested are the min, mean and max values for $U(10,40)$ used when defining the real mean reward of the different options, representing pessimistic, neutral and optimistic initial guesses. Additionally, the values that represent the percentage of traffic in the deployment bucket, p, we test vary from 10% to 50%. Values higher than 50% were rejected after some initial experimentation that showed that a learning percentage, $1 - p$, less than 50% has a very bad impact on the final percentage of optimal selection of the algorithm which contributes toward a continuous increase of cumulative regret (See Figures 3.8 3.13). For the different sets of parameters we capture the value of the two measures of performance, percentage of optimal selection and cumulative regret, for each scenario, at different time steps in the experiment. We do 200 runs for each dataset and the values presented are averaged over all runs for the 100 datasets form each scenario.

Then, we compare the results obtained when applying the three algorithms to the 300 artificial datasets. We compare the results of the algorithms in different scenarios using cumulative regret as the measure of performance when conducting the different tests, based on the assumption that most companies are more interested in minimizing their losses. We capture the results at four different time steps, $t = 100$, $t = 500$, $t = 1500$ and $t = 2000$ and use the cumulative regret obtained for the 100 different datasets in the three different scenarios separately. We treat each algorithm that uses different parameter values as a different one which results in 23 different algorithms.

3.7.3 Comparison of algorithms over multiple datasets

In order to evaluate the performance of the algorithms, a method recommended by Demšar (Demšar, 2006) for the comparison of multiple classifiers over multiple datasets is implemented. The proposed non-parametric method, the Friedman test (Friedman, 1937), was recommended by Demšar after a thorough review of benchmarking classification, a common problem in the area of machine learning. The aim was to avoid the well-known ANOVA test, which makes various assumptions, such as normality, an assumption that, in cases where the sample size is not big enough, may be violated. The goal is to test whether the differences in the two measures of performance for the different algorithms are significant. If they are significant, it means that the differences in performance are not random and we can proceed with a post hoc test. Below are the main steps of the Friedman test, which relies on the ranked performance of the different algorithms instead of their real performance, when there are L algorithms and K data sets such that $l = 1, 2, 3, ..., L$ and $k = 1, 2, 3, ..., K$, where $L = 23$ and $K = 100$.

1. Rank all algorithms in ascending order according to their performance for each data set. In our case this is averaged over the 100 runs. Let R_k^l be the rank of algorithm l in the data set k

2. Calculate the mean rank of algorithm l, MR_l across all data sets such that $MR_l = \frac{1}{K}\sum_{k=1}^{K} R_k^l$

3. Calculate the test statistic of the Friedman test

$$\chi_F^2 = \frac{12K}{L(L+1)}\left[\sum_{l=1}^{L} MR_l^2 - \frac{L(L+1)^2}{4}\right] \tag{3.12}$$

   which is distributed according to the Chi-squared distribution with $L-1$ degrees of freedom. Additionally, Iman and Davenport (1980) propose a more powerful different statistic defined as

$$F_F = \frac{(K-1)\chi_F^2}{k(K-1)-\chi_F^2} \tag{3.13}$$

   which is distributed according to the F-distribution with $L-1$ and $(L-1)(K-1)$ degrees of freedom. The null hypothesis is that the algorithms are all equivalent and their mean ranks are equal. Check the value of the test statistics and if they are larger than the critical values, reject the null hypothesis

4. If the null hypothesis is rejected, proceed to a post hoc test which tests the null hypothesis for each pair of algorithms. According to the Nemenyi test (Nemenyi, 1962), the null hypothesis is rejected if the difference between their mean ranks exceeds the critical difference D which is defined as:

$$D = q_{\alpha,\infty,L}\sqrt{\frac{L(L+1)}{12K}} \tag{3.14}$$

   where the value of $q_{\alpha,\infty,L}$ is based on the studentized range statistic.

Some other post hoc tests that can be used are Shaffer static procedure, Bergmann-Hommel's procedure and Holm procedure. According to Garcia and Herrera (2008) the most powerful post hoc test is Bergmann-Hommel's procedure but it is the most difficult one and computationally expensive. Moreover, the authors emphasize the advantage of Holm procedure when there is a large number of data sets but, based on their experimental results, they encourage the use of Shaffer static procedure. Even though in their work they prove that Nemenyi's test is very conservative, we choose to use this test since it is very simple and we are more interested in the relative performance of all algorithms and not in the comparison of all algorithms against a control.

3.7.4 Results

In Tables 3.5, 3.6, 3.7, we observe that the best parameter setting for the learning and deployment buckets algorithm, in all scenarios, is when $\mu_0 = 40$ and $p = 10\%$ since it reaches the highest final percentage of optimal selection . This algorithm is the closest to the standard version of Thompson Sampling since most of the traffic (90%) is sent to the learning bucket where the Bayesian exploration algorithm is implemented. When the variability of rewards is small and moderate (i.e. Scenarios 1 and 2), the parameters that optimize either percentage of optimal selection and cumulative regret are the same. However, in Scenario 3, the highest percentage of optimal selection is reached when $\mu_0 = 40$ and $p = 10\%$ but the lowest regret is obtained when $\mu_0 = 40$ and $p = 50\%$. This indicates that the algorithm that maximizes percentage of optimal selection does not necessarily minimize regret. In cases with small and moderate variability in rewards, an algorithm similar to standard Thompson Sampling, i.e. learning and deployment buckets algorithm with small p, and standard Thompson Sampling itself have a very good performance . The performance of standard Thompson Sampling is better compared to the learning and deployment buckets algorithms and the introduction of the deployment buckets does not improve the performance of standard Thompson Sampling. However, in the case of high variability in rewards, an algorithm that sends a significant amount of traffic to the option with the best observed performance manages to minimize regret. When $\mu_0 = 25$ and $\mu_0 = 40$, the learning and deployment buckets methodology performs better than Thompson Sampling in term of the final cumulative regret.

Algorithm-Parameters	**Optimal selection(%)**	**Cumulative regret**
ε-greedy($\varepsilon = 0.1$)	$\underline{77.12}$	$\underline{4846.93}$
ε-greedy($\varepsilon = 0.2$)	69.84	5508.48
ε-greedy($\varepsilon = 0.3$)	71.44	8291.14
ε-greedy($\varepsilon = 0.4$)	63.56	8976.16
ε-greedy($\varepsilon = 0.5$)	54.43	12776.19
Thompson Samp. ($\mu_0 = 10$)	90.40	2464.91
Thompson Samp. ($\mu_0 = 25$)	$\underline{99.83}$	$\underline{384.66}$
Thompson Samp. ($\mu_0 = 40$)	99.70	480.30
$\mu_0 = 10, p = 50\%$	77.23	3319
$\mu_0 = 10, p = 40\%$	82.72	4303
$\mu_0 = 10, p = 30\%$	83.93	3020
$\mu_0 = 10, p = 20\%$	87.19	2848
$\mu_0 = 10, p = 10\%$	87.37	2737
$\mu_0 = 25, p = 50\%$	93.27	2284
$\mu_0 = 25, p = 40\%$	93.71	2227
$\mu_0 = 25, p = 30\%$	95.36	1785
$\mu_0 = 25, p = 20\%$	96.97	1168
$\mu_0 = 25, p = 10\%$	98.68	792
$\mu_0 = 40, p = 50\%$	93.26	2380
$\mu_0 = 40, p = 40\%$	96.78	2070
$\mu_0 = 40, p = 30\%$	98.16	1666
$\mu_0 = 40, p = 20\%$	98.94	1208
$\mu_0 = 40, p = 10\%$	$\underline{99.66}$	$\underline{747}$

TABLE 3.5: Cumulative regret and percentage of optimal selection for the learning and deployment buckets algorithm, standard Thompson Sampling and ε-greedy with different parameter values in Scenario 1 at $t = 2000$. The values are averaged over 100×200 runs. We underline the best parameter setting for each algorithm.

Algorithm-Parameters	**Optimal selection(%)**	**Cumulative regret**
ε-greedy($\varepsilon = 0.1$)	$\underline{76.50}$	$\underline{4953.07}$
ε-greedy($\varepsilon = 0.2$)	68.58	5627.92
ε-greedy($\varepsilon = 0.3$)	69.51	8405.04
ε-greedy($\varepsilon = 0.4$)	63.12	9072.22
ε-greedy($\varepsilon = 0.5$)	52.97	12825
Thompson Samp. ($\mu_0 = 10$)	88.62	2518.00
Thompson Samp. ($\mu_0 = 25$)	$\underline{98.48}$	$\underline{556.57}$
Thompson Samp. ($\mu_0 = 40$)	95.90	3764.08
$\mu_0 = 10, p = 50\%$	75.40	3152
$\mu_0 = 10, p = 40\%$	83.43	4176
$\mu_0 = 10, p = 30\%$	81.23	2954
$\mu_0 = 10, p = 20\%$	82.08	2794
$\mu_0 = 10, p = 10\%$	84.50	2749
$\mu_0 = 25, p = 50\%$	85.47	2036
$\mu_0 = 25, p = 40\%$	89.59	2081
$\mu_0 = 25, p = 30\%$	91.71	1675
$\mu_0 = 25, p = 20\%$	92.27	1164
$\mu_0 = 25, p = 10\%$	97.06	866
$\mu_0 = 40, p = 50\%$	89.84	2193
$\mu_0 = 40, p = 40\%$	91.92	1884
$\mu_0 = 40, p = 30\%$	96.81	1498
$\mu_0 = 40, p = 20\%$	96.80	1146
$\mu_0 = 40, p = 10\%$	$\underline{98.25}$	$\underline{785}$

TABLE 3.6: Cumulative regret and percentage of optimal selection for the learning and deployment buckets algorithm, standard Thompson Sampling and ε-greedy with different parameter values in Scenario 2 at $t = 2000$. The values are averaged over 100×200 runs. We underline the best parameter setting for each algorithm.

Algorithm-Parameters	Optimal selection(%)	Cumulative regret
ε-greedy($\varepsilon = 0.1$)	<u>57.92</u>	<u>8278.79</u>
ε-greedy($\varepsilon = 0.2$)	52.84	8469.57
ε-greedy($\varepsilon = 0.3$)	50.18	10887.64
ε-greedy($\varepsilon = 0.4$)	45.57	11316.49
ε-greedy($\varepsilon = 0.5$)	42.43	14480.87
Thompson Samp. ($\mu_0 = 10$)	58.79	6136.35
Thompson Samp. ($\mu_0 = 25$)	<u>68.22</u>	<u>5722.30</u>
Thompson Samp. ($\mu_0 = 40$)	57.98	5783.08
$\mu_0 = 10, p = 50\%$	53.22	4404
$\mu_0 = 10, p = 40\%$	65.71	5259
$\mu_0 = 10, p = 30\%$	56.50	4675
$\mu_0 = 10, p = 20\%$	63.73	4560
$\mu_0 = 10, p = 10\%$	57.70	5235
$\mu_0 = 25, p = 50\%$	67.08	3982
$\mu_0 = 25, p = 40\%$	66.22	4374
$\mu_0 = 25, p = 30\%$	62.40	3858
$\mu_0 = 25, p = 20\%$	57.01	3887
$\mu_0 = 25, p = 10\%$	57.63	4569
$\mu_0 = 40, p = 50\%$	63.60	<u>3654</u>
$\mu_0 = 40, p = 40\%$	71.50	4286
$\mu_0 = 40, p = 30\%$	70.56	4787
$\mu_0 = 40, p = 20\%$	74.27	5251
$\mu_0 = 40, p = 10\%$	<u>76.34</u>	4921

TABLE 3.7: Cumulative regret and percentage of optimal selection for the learning and deployment buckets algorithm, standard Thompson Sampling and ε-greedy with different parameter values in Scenario 3 at $t = 2000$. The values are averaged over 100×200 runs. We underline the best parameter setting for each algorithm.

In order to investigate further the performance of the different algorithms with different parameter values, we are going to look at just the cumulative regret as a measure of performance, at different times, and use benchmarking tests to assess the results. For brevity we apply the tests at four different time steps: $t = 100$, $t = 500$, $t = 1500$ and $t = 2000$. In Tables 3.9, 3.10, 3.11 and 3.12 we present the average ranking of the 23 different algorithms for Scenario 1, Scenario 2 and Scenario 3 at $t = 100$, $t = 500$, $t = 1500$ and $t = 2000$ respectively. For each scenario the average ranking is calculated using the the ranking of each algorithm in all the 100 datasets in the scenario.

In our case, the critical values from the chi-squared distribution and from the F-distribution are 33.92 and 1.57 respectively. Based on Friedman test (Friedman, 1937), as explained by Demšar (Demšar, 2006), we calculate the test statistics χ_F^2 and F_F in order to assess the null hypothesis that all algorithms with the different parameter values are equivalent and their mean ranks are equal. Table 3.8 shows the test statistics, when $\alpha = 0.05$, for all the the 12 tests performed for the different scenarios at different times. We observe that for all tests the test statistics are greater than the critical values. Thus, we can reject the null hypotheses and proceed with a post hoc technique for pairwise comparisons.

	Scenario 1		Scenario 2		Scenario 3	
	χ_F^2	F_F	χ_F^2	F_F	χ_F^2	F_F
$t = 100$	1336.65	153.27	1362.72	161.13	529.03	31.34
$t = 500$	1727.57	362.02	1750.97	386.05	941.65	74.08
$t = 1500$	1907.87	646.58	1919.56	677.66	1255.54	131.61
$t = 2000$	1943.97	751.71	1955.84	793.04	1307.04	144.90

TABLE 3.8: Test statistics for the different algorithms at different time steps and in different scenarios

The family of hypotheses for all pairwise comparisons, in all 12 tests, are tested using Nemeneyi test. In Appendix A, we present the 253 pairwise comparisons with their p-value when $\alpha = 0.05$, in Scenario 1, at $t = 100$. For brevity we present only one of the tables and explain the results. The 253 pairwise comparisons are split into two groups; the pairs whose p-value is less than the p-value of the Nemeneyi test and the pairs whose p-values is higher than the p-value used in the Nemeneyi test. We also highlight the pairs in which the best performing algorithm, underlined in Table 3.9, performs significantly better. We observe that the best performing algorithm at this time step and in this scenario, standard Thompson Sampling with $\mu_0 = 25$ is significantly different from all the versions of ε-greedy and most of the versions of the newly developed algorithm, learning and deployment buckets algorithm. However, the p-value does not allow us to reject the null hypothesis, that 2 algorithms have equal performance, when the algorithm is compared against the learning and deployment buckets algorithm with $\mu_0 = 25, p = 10\%$ and $\mu_0 = 25, p = 20\%$ and against standard Thompson Sampling with $\mu_0 = 40$. As noted by Lessmann et al. (2008) by failing to reject the null hypothesis does not mean that it is true, since Nemeneyi test is not very powerful.

Algorithm	Av.Ranking(Scenario 1)	Av. Ranking(Scenario 2)	Av. Ranking (Scenario 3)
ε-greedy($\varepsilon = 0.1$)	19.23	19.30	15.12
ε-greedy($\varepsilon = 0.2$)	15.44	15.74 (16)	13.37
ε-greedy($\varepsilon = 0.3$)	17.20	17.37 (20)	16.48
ε-greedy($\varepsilon = 0.4$)	16.98	17.25	15.73
ε-greedy($\varepsilon = 0.5$)	19.65	20.02	19.59
Thompson Samp. ($\mu_0 = 10$)	11.54	11.12	9.88
Thompson Samp. ($\mu_0 = 25$)	$\underline{2.10}$	$\underline{2.21}$	12.29
Thompson Samp. ($\mu_0 = 40$)	4.19	5.74	13.27
Buckets ($\mu_0 = 10$, p=50%)	10.97	10.82	7.34
Buckets ($\mu_0 = 10$, p=40%)	16.57	16.22	11.89
Buckets ($\mu_0 = 10$, p=30%)	11.30	11.07	7.42
Buckets ($\mu_0 = 10$, p=20%)	13.42	13.19	10.46
Buckets ($\mu_0 = 10$, p=10%)	12.86	12.57	11.43
Buckets ($\mu_0 = 25$, p=50%)	8.78	8.14	9.92
Buckets ($\mu_0 = 25$, p=40%)	8.62	8.54	11.89
Buckets ($\mu_0 = 25$, p=30%)	6.43	5.52	9.04
Buckets ($\mu_0 = 25$, p=20%)	4.61	4.35	$\underline{6.38}$
Buckets ($\mu_0 = 25$, p=10%)	2.97	2.93	7.00
Buckets ($\mu_0 = 40$, p=50%)	15.04	15.96	12.31
Buckets ($\mu_0 = 40$, p=40%)	17.54	17.60	14.90
Buckets ($\mu_0 = 40$, p=30%)	15.40	15.47	11.63
Buckets ($\mu_0 = 40$, p=20%)	15.31	14.90	14.28
Buckets ($\mu_0 = 40$, p=10%)	9.75	9.91	14.31

TABLE 3.9: Average Rankings for the three algorithms, with different parameter settings, in Scenario 1, Scenario 2, Scenario 3 based on cumulative regret at $t = 100$. We underline the best parameter setting for each algorithm. The results are averaged over 100×200 runs in each scenario.

Moreover, with a closer look at the rankings in Table 3.9,3.10, 3.11 and 3.12 we observe that the performance rankings are very similar at the different time steps. One important result is that at any time step the best algorithm for Scenario 1 and Scenario 2 is standard Thompson Sampling with $\mu_0 = 25$, followed by some of the learning and deployment buckets algorithms with high proportion of traffic sent to the learning bucket. However, when there is high variability in rewards (Scenario 3), the performance of Thompson Sampling is worse than the learning and deployment buckets algorithm with $\mu_0 = 25$ and $p = 20\%$ being the best at the beginning of the experiment, and learning and deployment buckets algorithm with $\mu_0 = 40$ and $p = 50\%$ being the best at the latest steps of the experiment. Additionally, it is clear once again that the value of μ_0 has a big impact on the performance of standard Thompson Sampling, with $\mu_0 = 10$ being a very bad choice.

Algorithm	Av.Ranking (Scenario 1)	Av.Ranking (Scenario 2)	Av.Ranking (Scenario 3)
ε-greedy($\varepsilon = 0.1$)	18.17	18.54	16.78
ε-greedy($\varepsilon = 0.2$)	17.24	17.77	15.95
ε-greedy($\varepsilon = 0.3$)	20.13	20.17	18.56
ε-greedy($\varepsilon = 0.4$)	20.57	20.72	19.93
ε-greedy($\varepsilon = 0.5$)	22.63	22.62	22.15
Thompson Samp. ($\mu_0 = 10$)	11.82	12.08	12.03
Thompson Samp. ($\mu_0 = 25$)	$\underline{1.48}$	$\underline{1.44}$	13.50
Thompson Samp. ($\mu_0 = 40$)	1.90	2.32	14.36
Buckets ($\mu_0 = 10$, p=50%)	14.53	14.51	7.52
Buckets ($\mu_0 = 10$, p=40%)	17.37	17.26	11.70
Buckets ($\mu_0 = 10$, p=30%)	13.32	13.28	7.82
Buckets ($\mu_0 = 10$, p=20%)	13.98	14.03	10.01
Buckets ($\mu_0 = 10$, p=10%)	12.50	12.64	10.38
Buckets ($\mu_0 = 25$, p=50%)	10.81	10.30	8.13
Buckets ($\mu_0 = 25$, p=40%)	11.52	11.80	9.57
Buckets ($\mu_0 = 25$, p=30%)	8.41	8.66	7.09
Buckets ($\mu_0 = 25$, p=20%)	5.38	5.29	$\underline{5.44}$
Buckets ($\mu_0 = 25$, p=10%)	3.74	3.65	7.69
Buckets ($\mu_0 = 40$, p=50%)	14.23	14.47	7.52
Buckets ($\mu_0 = 40$, p=40%)	14.17	13.22	10.80
Buckets ($\mu_0 = 40$, p=30%)	10.53	10.03	11.60
Buckets ($\mu_0 = 40$, p=20%)	7.47	7.14	14.09
Buckets ($\mu_0 = 40$, p=10%)	4.00	3.94	13.26

TABLE 3.10: Average Rankings for the three algorithms, with different parameter settings, in Scenario 1, Scenario 2, Scenario 3 based on cumulative regret at $t = 500$. We underline the best parameter setting for each algorithm. The results are averaged over 100×200 runs in each scenario.

Algorithm	**Av.Ranking(Scenario 1)**	**Av. Ranking(Scenario 2)**	**Av. Ranking (Scenario 3)**
ε-greedy($\varepsilon = 0.1$)	18.50	18.67	18.54
ε-greedy($\varepsilon = 0.2$)	19.0	19.09	18.02
ε-greedy($\varepsilon = 0.3$)	20.71	20.79	20.68
ε-greedy($\varepsilon = 0.4$)	21.79	21.78	21.37
ε-greedy($\varepsilon = 0.5$)	22.80	22.80	22.56
Thompson Samp. ($\mu_0 = 10$)	11.53	11.82	14.10
Thompson Samp. ($\mu_0 = 25$)	$\underline{1.45}$	$\underline{1.47}$	13.75
Thompson Samp. ($\mu_0 = 40$)	1.74	2.15	13.70
Buckets ($\mu_0 = 10$, p=50%)	15.43	15.57	8.21
Buckets ($\mu_0 = 10$, p=40%)	17.45	17.30	11.75
Buckets ($\mu_0 = 10$, p=30%)	14.10	14.11	9.37
Buckets ($\mu_0 = 10$, p=20%)	13.48	13.69	9.15
Buckets ($\mu_0 = 10$, p=10%)	12.47	12.93	10.52
Buckets ($\mu_0 = 25$, p=50%)	12.29	11.60	6.71
Buckets ($\mu_0 = 25$, p=40%)	12.29	12.11	8.30
Buckets ($\mu_0 = 25$, p=30%)	8.79	9.08	6.12
Buckets ($\mu_0 = 25$, p=20%)	5.84	6.08	5.98
Buckets ($\mu_0 = 25$, p=10%)	3.77	3.63	8.84
Buckets ($\mu_0 = 40$, p=50%)	12.90	13.01	$\underline{5.55}$
Buckets ($\mu_0 = 40$, p=40%)	11.19	10.66	8.17
Buckets ($\mu_0 = 40$, p=30%)	8.58	8.32	10.27
Buckets ($\mu_0 = 40$, p=20%)	6.21	5.88	12.68
Buckets ($\mu_0 = 40$, p=10%)	3.60	3.37	11.51

TABLE 3.11: Average Rankings for the three algorithms, with different parameter settings, in Scenario 1, Scenario 2, Scenario 3 based on cumulative regret at $t = 1500$. We underline the best parameter setting for each algorithm. The results are averaged over 100×200 runs in each scenario.

Using the average rankings of all algorithms presented in Tables 3.9, 3.10, 3.11, 3.12, we see that, at any time step, the best algorithm in terms of cumulative regret when there is small and moderate variability in the rewards of the different options, is standard Thompson Sampling with $\mu_0 = 25$. However, the performance of this algorithm deteriorates when there is high variability in the rewards (Scenario 3). In this case, the best algorithm becomes one of the learning and deployment buckets algorithms. At $t = 100$ and $t = 500$, the best one is learning and deployment buckets algorithm with $\mu = 25$ and 80% of traffic sent to the learning bucket, while at $t = 1500$ and $t = 2000$ this algorithm gets second and third respectively with the optimal one being the learning and deployment buckets algorithm with $\mu_0 = 40$ and 50% of traffic sent to the learning bucket. Thus, a further investigation on the performance of these three algorithms, in terms of cumulative regret is performed. In particular, we are interested in the variability in cumulative regret for the different datasets in each scenario.

Algorithm	Av. Ranking (Scenario 1)	Av. Ranking (Scenario 2)	Av. Ranking (Scenario 3)
ε-greedy($\varepsilon = 0.1$)	18.68	18.88	18.79
ε-greedy($\varepsilon = 0.2$)	19.50	19.49	18.59
ε-greedy($\varepsilon = 0.3$)	21.00	21.04	20.88
ε-greedy($\varepsilon = 0.4$)	21.80	21.80	21.53
ε-greedy($\varepsilon = 0.5$)	22.80	22.80	22.60
Thompson Samp. ($\mu_0 = 10$)	11.33	11.72	14.44
Thompson Samp. ($\mu_0 = 25$)	$\underline{1.45}$	$\underline{1.48}$	13.72
Thompson Samp. ($\mu_0 = 40$)	1.74	2.18	13.55
Buckets ($\mu_0 = 10$, p=50%)	15.60	15.69	8.32
Buckets ($\mu_0 = 10$, p=40%)	17.29	17.22	11.69
Buckets ($\mu_0 = 10$, p=30%)	14.11	14.17	9.91
Buckets ($\mu_0 = 10$, p=20%)	13.35	13.53	8.87
Buckets ($\mu_0 = 10$, p=10%)	12.41	12.97	10.85
Buckets ($\mu_0 = 25$, p=50%)	12.24	11.65	6.56
Buckets ($\mu_0 = 25$, p=40%)	12.31	12.06	8.32
Buckets ($\mu_0 = 25$, p=30%)	8.81	9.08	5.97
Buckets ($\mu_0 = 25$, p=20%)	5.90	6.12	6.08
Buckets ($\mu_0 = 25$, p=10%)	3.77	3.64	9.07
Buckets ($\mu_0 = 40$, p=50%)	12.77	12.77	$\underline{5.26}$
Buckets ($\mu_0 = 40$, p=40%)	10.92	10.47	8.03
Buckets ($\mu_0 = 40$, p=30%)	8.45	8.12	9.94
Buckets ($\mu_0 = 40$, p=20%)	6.05	5.76	11.99
Buckets ($\mu_0 = 40$, p=10%)	3.60	3.28	10.87

TABLE 3.12: Average Rankings for the three algorithms, with different parameter settings, in Scenario 1, Scenario 2, Scenario 3 based on cumulative regret at $t = 2000$. We underline the best parameter setting for each algorithm. The results are averaged over 100×200 runs in each scenario.

Figures 3.15, 3.16, 3.17 show the cumulative regret for each of the 100 different datasets, in the three different scenarios, for Thompson Sampling with $\mu_0 = 25$, learning and deployment buckets algorithm with $\mu_0 = 25$ and $p = 20\%$ and learning and deployment buckets algorithm with $\mu_0 = 40$ and $p = 50\%$ respectively. Each line is the cumulative regret for each individual dataset, averaged over the 200 runs and the red line shows the mean cumulative regret over all the 100 different datasets, in each scenario.

In Scenario 1, standard Thompson Sampling converges very quickly while at the same time it also has a very small variability in the results. In contrast, the learning and deployment buckets algorithm that starts with an initial estimate of 40 for the mean reward and just 50% of traffic sent to the learning bucket gets much higher cumulative regret. Even though standard Thompson Sampling has the best performance during the whole time of the experiment in Scenarios 1 and 2, the results in Scenario 2 have a high variability while the variability for the same scenario in the two settings of the learning and deployment buckets algorithms examined is much smaller. However, both learning and deployment buckets algorithms with the two sets of parameters converge at much higher values of cumulative regret and even the upper bound for standard Thompson Sampling is less than the mean and in some cases the lower bound of cumulative regret for the other two algorithms. Finally, in Scenario 3 cumulative regret for standard Thompson Sampling increases almost linearly while the two settings of the learning and deployment buckets algorithms that get much less regret perform very similarly.

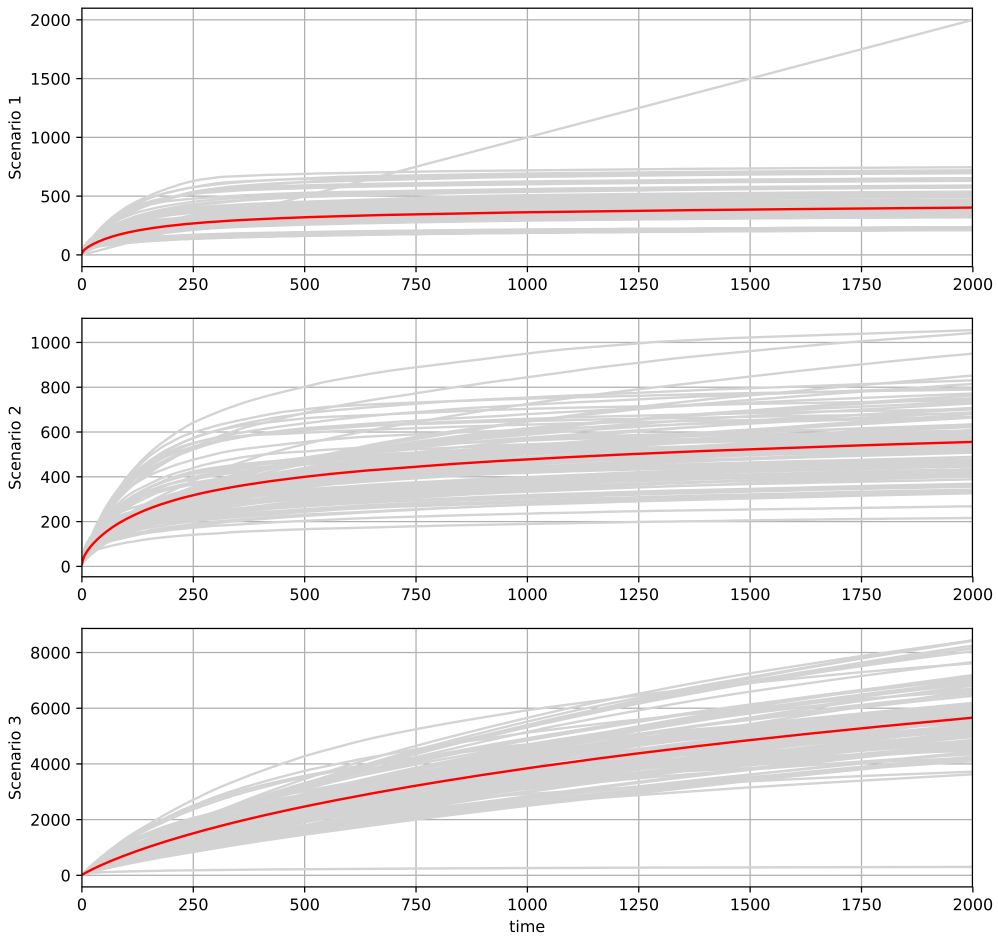


FIGURE 3.15: Cumulative regret over time for 100 different datasets in each scenario for standard Thompson Sampling with $\mu_0 = 25$. The red curve shows the mean cumulative regret over all datasets in each scenario. Each curved is averaged over 200 runs.

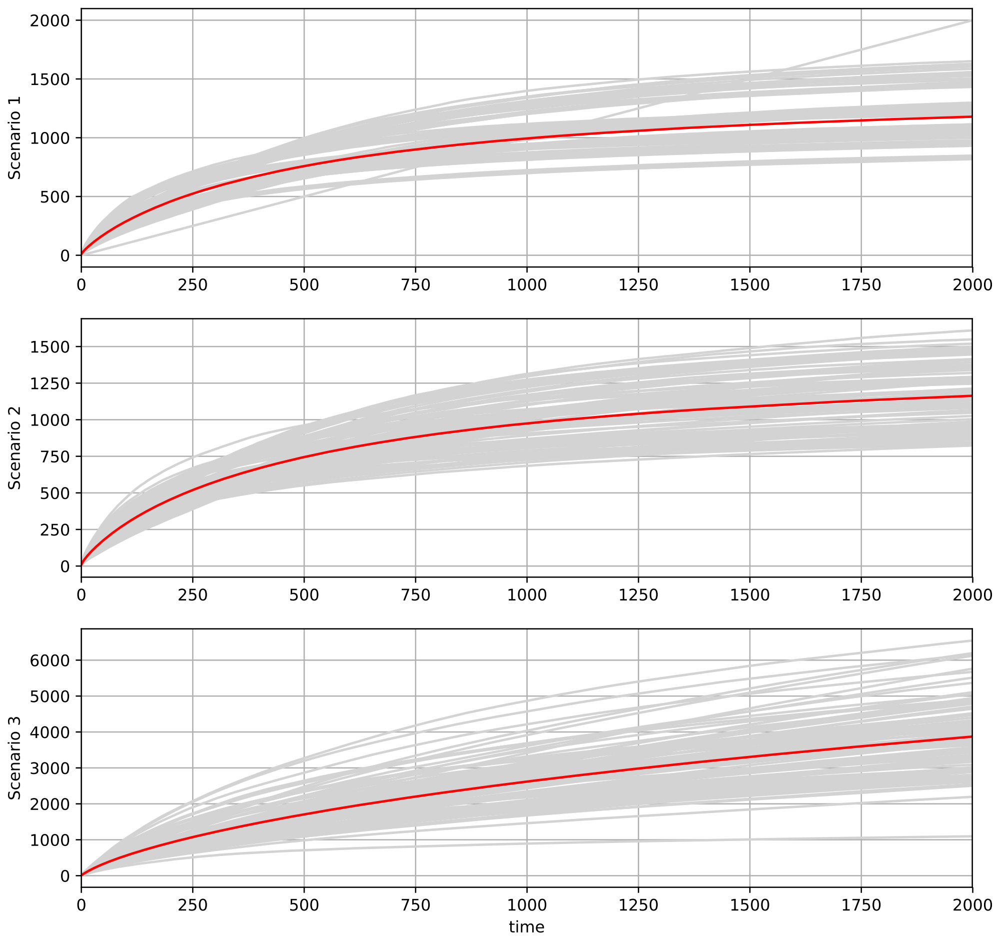


FIGURE 3.16: Cumulative regret over time for 100 different datasets in each scenario for learning and deployment buckets with $\mu_0 = 25$ and 80% of learning. The red curve shows the mean cumulative regret over all datasets in each scenario. Each curved is averaged over 200 runs.

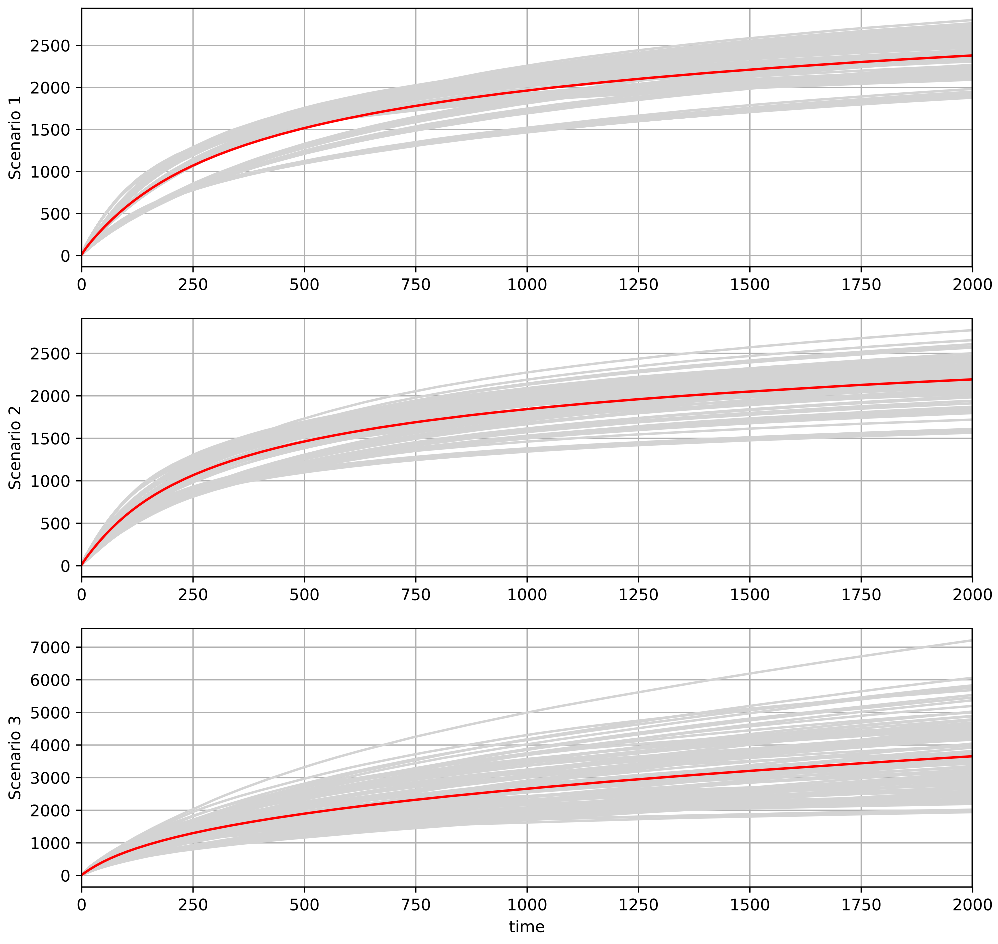


FIGURE 3.17: Cumulative regret over time for 100 different datasets in each scenario for learning and deployment buckets with $\mu_0 = 40$ and 50% of learning. The red curve shows the mean cumulative regret over all datasets in each scenario. Each curved is averaged over 200 runs.

3.7.5 Discussion

To sum up, testing the MAB algorithms on different datasets that represent the real system shows that the variability in the reward distribution of the different options has a big impact on the performance of all algorithms. Various existing experimental results emphasize the superiority of Thompson Sampling with not enough attention given to the environment that was used to test its performance. Motivated by the results obtained when testing the algorithms on two real-world datasets we attempted to test the impact of this by creating various artificial datasets and perform controlled experiments. Firstly, the well known algorithms from the area of MAB, ε-greedy and Thompson Sampling were tested on these different datasets followed by some experimental results on a new proposed methodology that builds on the idea of different buckets by Agarwal et al. (2009).

The learning and deployment buckets algorithm is developed to test whether showing the best performing option so far to some amount of traffic improves the performance of standard Thompson Sampling. Even though the definition of the algorithm is tailored to the needs of a website optimization problem, the idea of splitting the traffic can be applied in any company that uses online experimentation. With our experimental results we show how important the choice of initial hyperparameters for standard Thompson Sampling is, while at the same time we are able to detect changes in its performance in different environments. Even though its performance is in general very good, when there is high variability in the rewards of the different options, the algorithm is not able to cope well enough with this high variability. In this situation, it is clear that one will benefit from an algorithm that sends some amount of traffic to the option that has been the best in the past. Thus, it is very important to use wisely any prior knowledge about the environment and about the options we are comparing in order to make the right decision about the most appropriate values for the initial hyperparameters and the most suitable algorithm that we can use.

3.8 Concluding remarks

This chapter considers an example from website optimisation in which rewards follow a normal distribution and are stationary with respect to time. We described three different algorithms within the chapter, two being well-known (epsilon-greedy and Thompson sampling) and the third, learning and deployment buckets, an algorithm that we developed. Comparisons were made initially on two real-world datasets but a full benchmarking test was also carried out to provide a more rigorous test of the different algorithms.

We find throughout that Thompson Sampling performs better than epsilon-greedy and that the learning and deployment buckets algorithm can improve upon Thompson Sampling where there is high variability in the rewards of the different options. When a small proportion of traffic is sent to the deployment bucket, the learning and deployment buckets algorithm performs very well too. Thus, in cases where there is some initial information that the variability in rewards of the different options is high, it might be beneficial to implement the learning and deployment buckets algorithm instead of the standard Thompson Sampling.

The benchmarking test also provides some interesting insights into how prior parameters should be set within Thompson sampling and how some characteristics of the datasets, like the variability in the rewards of the candidate options, affect its performance. To the best of our knowledge there have been no benchmarking tests of different MAB algorithms made previous to this study.

Chapter 4

Seasonal case

In this chapter, we study the problem of choosing the best option out of a number of candidate options when there is seasonality in their reward distribution, a common problem in many real-world applications. For example, in revenue management the demand for certain products varies for different seasons and hence it depends heavily on seasonality. Motivated by the problem of an online travel agency, where the reward obtained by the selected website on a certain day of the week depends both on its real mean performance and on the day that website was shown to the users, we aim to develop a methodology that manages to separate the option and the day effect.

The best option remains the same during the experiment. But in order for it to be correctly identified, the methodology needs to identify the effect that seasonality has on the reward obtained by each chosen option at each time step. What we are interested in is how quickly the algorithm can adapt to the changing environment, learn the underlying seasonality and be able to distinguish the seasonal and the option effect. This formulation allows us to separate the impact of seasonality from the impact of the option being played.

Initially, we implement Thompson Sampling in the framework of contextual bandits in order to handle the case of seasonality in the reward distribution of the different options. The algorithm combines Thompson Sampling with linear regression and is the same method used by Scott (2010) and more recently by Porter et al. (2016). After some experimentation on the hyperparameters involved in the standard contextual Thompson Sampling methodology, we proceed with the description of a new methodology that combines contextual Thompson Sampling with statistical model selection. In our knowledge, this is the first algorithm that solves the problem of MAB in an environment with seasonality by combining the Bayesian exploration algorithm with a statistical model selection method.

We perform experiments on different variations of contextual Thompson Sampling against standard Thompson Sampling and the new proposed method on a number of artificial datasets. Contextual Thompson Sampling is implemented on the two real-world datasets introduced in Chapter 3 to which we add daily additive seasonality that has the same impact on all options. Then, the artificial datasets we create combine different variabilities in the reward distribution of the different options with different levels and periods of seasonality that may affect the performance of the different options.

We organize the rest of this chapter as follows. Initially, in Section 4.1, 4.2 and 4.3 we explain the problem formulation and describe the main steps of contextual Thompson Sampling where seasonality is treated as contextual information. In Section 4.4 we present results from the experimentation on the hyperparameters involved when the algorithm is applied to the two real-world datasets introduced in the previous chapter with added daily seasonality affecting the performance of the different options. Section 4.5 describes the general idea of the new methodology we developed along with details about its practical implementation. Section 4.6 describes properties of the artificial datasets we created and presents the results obtained when testing the developed methodology on the artificial datasets. We finally conclude this chapter with Section 4.7 that highlights our main findings.

4.1 Problem formulation

Suppose we have N different options that we want to compare, such that $a \in \{1, 2, ..., N\}$, and at each time step t, we choose option a_t and receive its reward $r_{a_t}(t)$. We still assume normal likelihood with unknown mean and variance but in addition, we assume that rewards are affected by one level of seasonality and that the seasonality effect is additive. This means that for each time step, we choose an option and an additive factor $z_t \in \mathbf{Z}$ is added to the expected reward of each option, depending on the current season, where $\mathbf{Z}$ is a vector with all the seasonal factors. Thus, the reward of a_t, $r_{a_t}(t)$ satisfies $r_{a_t}(t) \sim N(m_{a_t} + z_t, s^2_{a_t})$ and the expected reward of arm a_t is $E[r_{a_t}(t)] = m_{a_t} + z_t$. The seasonal factors are assumed the same for all options so even though there is seasonality in our rewards, the best option remains the same for all "seasons". We keep using the term seasons in the remaining chapter, when referring to the level of seasonality. For the seasonal case with F seasons we introduce $F - 1$ dummy variables such that matrix X has t rows and $K = (F - 1) + N$ columns where the first $(F - 1)$ correspond to the season and the remaining N to the options. Thus,

$\mathbf{x}_{t,a} = \begin{pmatrix} x_1 & x_2 & \dots & x_{F-1} & x_{(F-1)+1} & x_{F+1} & \dots & x_{K-1} & x_K \end{pmatrix}$ is the t^{th} row of matrix X, filled with binary values that represent the season and the option chosen at time t.

4.2 Bayesian linear regression models

We assume the following linear regression model

$$\mathbf{Y} = X\boldsymbol{\theta} + \boldsymbol{\varepsilon}, \tag{4.1}$$

where $\mathbf{Y}$ is the n-dimensional vector of observations i.e. the rewards earned over the different time periods, X is the $n \times K$ matrix where $\mathbf{x}_i$ is the i^{th} row of the matrix with k elements and assumed fixed, showing the option chosen at each time step, $\boldsymbol{\theta}$ is the k-dimensional coefficient vector and $\boldsymbol{\varepsilon}$ is the n-dimensional normally distributed error vector. Using Bayes theorem we know that

$$P(\boldsymbol{\theta}, \sigma^2 \mid \mathbf{Y}) \propto P(\mathbf{Y} \mid \boldsymbol{\theta}, \sigma^2) P(\boldsymbol{\theta}, \sigma^2), \tag{4.2}$$

where $P(\mathbf{Y} \mid \boldsymbol{\theta}, \sigma^2)$ is the normal likelihood and $P(\boldsymbol{\theta}, \sigma^2)$ the joint prior. We assume a Normal-Inverse Gamma prior distribution as this is a conjugate prior distribution for the regression model, allowing a closed form expression to be obtained for the posterior distribution of the parameters. The Inverse Gamma (IG) prior density function is for σ^2 and the normal prior density function is for $\boldsymbol{\theta}$ given σ^2. Thus,

$$\begin{aligned} P(\boldsymbol{\theta}, \sigma^2) &= P(\boldsymbol{\theta} \mid \sigma^2) P(\sigma^2) \\ &= N(\boldsymbol{\theta}; \boldsymbol{\mu}, \sigma^2 V) IG(\sigma^2; \alpha, \beta) \\ &\propto (\frac{1}{\sigma^2})^{\alpha + k/2 + 1} \times exp\left[\frac{-1}{\sigma^2}\left\{\beta + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^T V^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu})\right\}\right] \\ &= NIG(\boldsymbol{\mu}, V, \alpha, \beta) \end{aligned} \tag{4.3}$$

where $\boldsymbol{\mu}$ is the k-dimensional mean vector, V is the $k \times k$ dimensional covariance matrix and α, β are the scalar hyper parameters of σ^2.

The likelihood at the beginning is given as

$$\begin{aligned} P(\mathbf{Y} \mid \boldsymbol{\theta}, \sigma^2) &= N(X\boldsymbol{\theta}, \sigma^2 \mathbf{I}) \\ &= (\frac{1}{2\pi\sigma^2})^{n/2} exp\left\{\frac{-1}{2\sigma^2}(\mathbf{Y} - X\boldsymbol{\theta})^T (\mathbf{Y} - X\boldsymbol{\theta})\right\} \end{aligned} \tag{4.4}$$

Since we use conjugate priors, after the update the posterior has the same distribution as the prior but with different parameter values. Using equation 4.2 we calculate the posterior

$$\begin{aligned} P(\boldsymbol{\theta},\sigma^2 \mid \mathbf{Y}) &\propto (\frac{1}{\sigma^2})^{\alpha+(n+\kappa)/2+1} \times exp\left\{\frac{-1}{\sigma^2}\left[\beta^* + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu}^*)^T {V^*}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu}^*)\right]\right\} \\ &= NIG(\boldsymbol{\mu}^*, V^*, \alpha^*, \beta^*) \end{aligned} \tag{4.5}$$

where

$$\begin{aligned} \boldsymbol{\mu}^* &= (V^{-1} + X^T X)^{-1}(V^{-1}\boldsymbol{\mu} + X^T\mathbf{Y}) \\ V^* &= (V^{-1} + X^T X)^{-1} \\ \alpha^* &= \alpha + n/2 \\ \beta^* &= \beta + \frac{1}{2}\left[\boldsymbol{\mu}^T V^{-1}\boldsymbol{\mu} + \mathbf{Y}^T\mathbf{Y} - {\boldsymbol{\mu}^*}^T {V^*}^{-1}\boldsymbol{\mu}^*\right] \end{aligned} \tag{4.6}$$

The same set of equations are used in Porter et al. (2016)

The marginal posterior of $\mathbf{Y}$ is calculated by integrating out θ and σ^2 from $P(\boldsymbol{\theta},\sigma^2|\mathbf{Y})$. So

$$\begin{aligned} p(\mathbf{Y}) &= \int_0^\infty P(\mathbf{Y} \mid \boldsymbol{\theta},\sigma^2)P(\boldsymbol{\theta},\sigma^2)d\boldsymbol{\theta}d\sigma^2 \\ &= \int_0^\infty N(X\boldsymbol{\theta},\sigma^2 I_m)NIG(\boldsymbol{\mu},V,\alpha,\beta)d\boldsymbol{\theta}d\sigma^2 \\ &= MVSt_{2\alpha}\left(X\boldsymbol{\mu}, \frac{\beta}{\alpha}(I + XVX^T)\right) \end{aligned}$$

Following the last result, the posterior predictive distribution is

$$P(\tilde{\mathbf{Y}}|\mathbf{Y}) = MVSt_{2\alpha^*}\left(\tilde{X}\boldsymbol{\mu}^*, \frac{\beta^*}{\alpha^*}(I + \tilde{X}V\tilde{X}^T)\right) \tag{4.7}$$

where $\tilde{X}$ is a new $m \times K$ matrix and $\tilde{\mathbf{Y}}$ is the vector with m predicted values. In this posterior predictive distribution, there exist two sources of uncertainty; one is due to σ^2, the fundamental source of variability which is unaccounted for by $\tilde{X}\theta$. The second comes from the uncertainty in $\boldsymbol{\theta}$ and σ^2 caused by their estimation using a finite sample $\mathbf{Y}$. To calculate the marginal posterior of $\boldsymbol{\theta}$, we integrate out σ^2 from the NIG joint posterior as follows:

$$\begin{aligned} p(\boldsymbol{\theta}|\mathbf{Y}) &= \int_0^\infty p(\boldsymbol{\theta},\sigma^2|\mathbf{Y})d\sigma^2 \\ &= NIG(\boldsymbol{\mu^*},V^*,\alpha^*,\beta^*)d\boldsymbol{\theta}d\sigma^2 \\ &\propto \left(\frac{1}{\sigma^2}\right)^{\alpha^*+1} exp\left\{\frac{-1}{\sigma^2}\left[\beta^* + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu}^*)^T V^{*-1}(\boldsymbol{\theta}-\boldsymbol{\mu}^*)\right]\right\} \\ &\propto \left[1+\frac{(\boldsymbol{\theta}-\boldsymbol{\mu}^*)^T V^{*-1}(\boldsymbol{\theta}-\boldsymbol{\mu}^*)}{2\beta^*}\right]^{\alpha^*+k/2} \end{aligned}$$

This is the multivariate t-distribution with $2\alpha^*$ degrees of freedom, location μ^* and scale $(\frac{\beta^*}{\alpha^*})V^*$.

4.3 Algorithm

At each time step $t \in [0,T]$, the algorithm decides which option a_t to choose based on the estimated rewards for all of the options. This estimated reward is sampled from the linear regression model $\tilde{y}_{t,a} = \mathbf{x}_{t,a}^T\boldsymbol{\theta}_t + \varepsilon$, where $\tilde{y}_{t,a}$ is the estimated reward of arm a at time t, $\mathbf{x}_{t,a}^T$ is the transpose of the K-dimensional vector containing information about the current "season" and the option chosen, $\boldsymbol{\theta}_t$ is the K-dimensional coefficient vector. The option with the highest sampled reward $a_t = \underset{a=1,2,..,N}{\text{argmax}}(\mathbf{x}_{t,a}^T\boldsymbol{\theta}_t)$ is chosen and its real reward $r_{a_t}(t)$ is revealed, while the rewards of the non chosen options remain unknown. A history set $H_{t-1} = \{(\mathbf{x}_{\tau,a}, r_{a_\tau}(\tau), a_\tau) : 1 \leq \tau \leq t-1\}$ is incorporated into the likelihood function and once an option is chosen we update the history set by adding a new triplet $(\mathbf{x}_{t,a}, r_{a_t}(t), a_t)$. The history set is introduced in cases where Thompson Sampling is used for the solution of contextual bandits (Li et al., 2010a; Chapelle and Li, 2011)

We assume normal likelihood with unknown mean $\boldsymbol{\mu}$ and variance σ^2. In order to estimate the coefficients $\boldsymbol{\theta}_t = \begin{pmatrix}\theta_1, & \theta_2, & \dots & \theta_{K-1}, & \theta_K\end{pmatrix}$, we assign a Normal-Inverse Gamma prior distribution with the initial hyperparameters $\boldsymbol{\mu}$, V, α, β so that $\sigma^2 \mid H_{t-1} \sim IG(\alpha^*,\beta^*)$ and $\boldsymbol{\theta}_t|\sigma^2, H_{t-1} \sim MVN(\boldsymbol{\mu}^*, \sigma^{2^{(s)}}V^*)$ where $\boldsymbol{\mu}^*$, V^*, α^*, β^* are the updated hyperparameters computed with Bayes Rule at each iteration as described in Section 4.2. We describe the steps of the algorithm in Algorithm 6.

Algorithm 6 Contextual Thompson Sampling for the seasonal case

```
Set $\boldsymbol{\mu_0}$, $V_0$, $\alpha_0$, $\beta_0$
Define a vector $\mathbf{S}$ with the seasonal factors such that $\mathbf{S} \in \mathbb{R}^{(F-1)}$
for each $\tau \in \{1, 2, ..., T\}$ do
    if $\tau = 1$ then
        $\boldsymbol{\mu}^* = \boldsymbol{\mu}_0$, $V^* = V_0$, $\alpha^* = \alpha_0$, $\beta^* = \beta_0$
    end if
    Find the "season" $d$, $d \in \{1, ...F\}$
    Sample $\boldsymbol{\theta}_\tau \sim MVT_{2\alpha^*}\left(\boldsymbol{\mu}^*, \frac{\alpha^*}{\beta^*}V^*\right)$, where $\boldsymbol{\theta}_\tau \in \mathbb{R}^K$
    Choose option $a_\tau = \underset{k=F+1,..,K}{\operatorname{argmax}}(\mathbf{x}_{t,a}^T\boldsymbol{\theta}_t)$ and receive its real reward $r_{a_\tau}(\tau)$
    Update $\boldsymbol{\mu}^*$, $V^*$, $\alpha^*$ and $\beta^*$ using Equations 4.6
end for
```

4.4 Numerical experiments on real-world datasets

After performing time series analysis on data from a major online travel agency, we observed that there is daily seasonality in the data. Table 4.1 shows the daily additive factors with Monday being the base day. The seasonal effect is assumed the same for all options so even though there is seasonality in our rewards, the best option remains the same for all days.

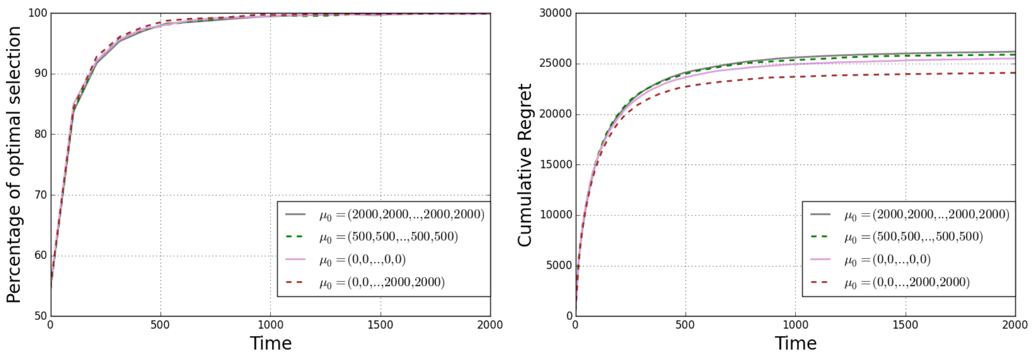
Daily Factors **Z**							
Day (d)	Mon.	Tues.	Wed.	Thur.	Fri.	Sat.	Sun.
Daily Factor (z_d)	0	74	714	-305	-879	-1205	-674

TABLE 4.1: Additive factors for each day of the week

In the following sections, we present the results obtained by using Algorithm 6. We test the algorithm on the two different datasets we introduced for the stationary case, using the same reward data for the seven candidate options, but with the seven daily factors now also added. First, we present the results obtained in an environment where the performance of the different options is very similar, described by the parameters in Table 3.2, and test the performance of the algorithm for different values of the four hyperparameters. Then we proceed with tests on the performance of the algorithm in an environment where there is a clear winner between the different options and the real performance of the seven options is characterized by the parameters in Table 3.3. For the latter dataset, we note again that even though the mean value of the best option is much higher than its competitors, its variance is also very high. For both datasets, we then add the same daily factors from Table 4.1. In the experiments, we study the dependence on the choice of prior hyperparameters and compare the effectiveness of Algorithm 6 which accounts for seasonality, with results obtained using Algorithm 3 which ignores any seasonal effects. This allows us to test whether accounting for seasonality indeed improves the convergence.

4.4.1 Results for Dataset 1

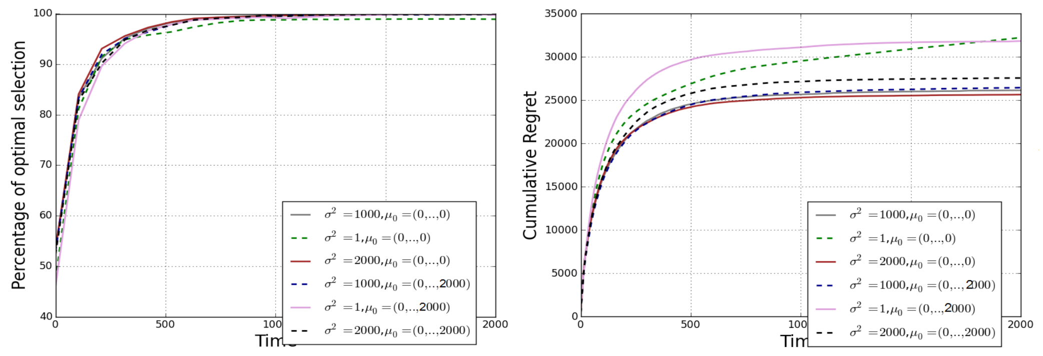
An assumption made is that days and options are independent. We therefore define V_0, the initial covariance matrix, to be a diagonal matrix and initially we choose the value of 1000 in the diagonal, representing the variance of the days and the options. First, we examine the dependence on the prior estimate of $\boldsymbol{\mu}_0$. At the beginning we assume that a reasonable value for the the first parameter of the normal prior distribution is 0 for the days and 2000 for the different options. Then, as we did in the stationary case, we set $\alpha_0 = 1$ and $\beta_0 = 1$.



(A) (B)

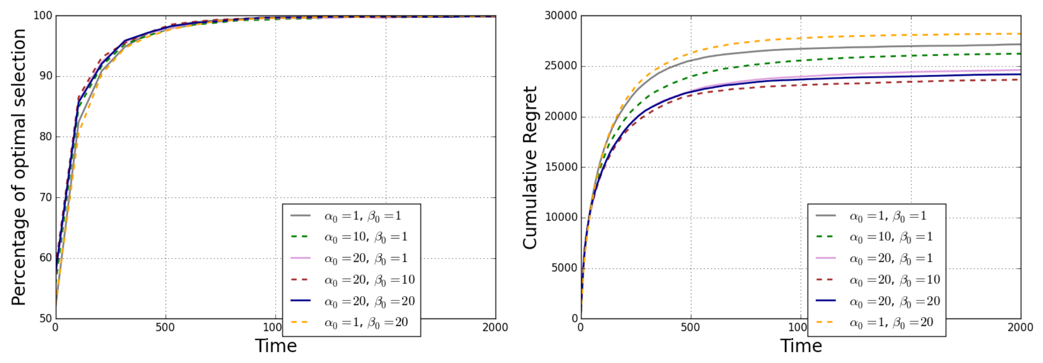
FIGURE 4.1: Percentage of optimal selection and cumulative regret for different $\boldsymbol{\mu}_0$ when $\sigma^2 = 1000$, $\alpha_0 = 1$, $\beta_0 = 1$. The curves are averages over 100 runs.

Surprisingly, we observe that varying $\boldsymbol{\mu}_0$ values has less impact on the performance of the algorithm than in the stationary case. In Figure 4.1, we can see that there is just a small improvement in the final cumulative regret when $\boldsymbol{\mu}_0 = (0, 0, 0....2000, 2000)$ compared to when $\boldsymbol{\mu}_0 = (0, 0, ..., 0)$. Thus, we proceed with the test of these two mean values, combined with different values placed in the diagonal of V_0. We will denote the values in the diagonal by σ^2. Interestingly, Figure 4.2 highlights different points where regret converges for $\boldsymbol{\mu}_0 = (0, 0, ..., 0)$. Large variance ($\sigma^2 = 1000$ and $\sigma^2 = 2000$) causes cumulative regret to converge



(A) (B)

FIGURE 4.2: Percentage of optimal selection and cumulative regret for different $\boldsymbol{\mu_0}$ and σ^2 when $\alpha_0 = 1$, $\beta_0 = 1$. The curves are averages over 100 runs.



(A) (B)

FIGURE 4.3: Percentage of optimal selection and cumulative regret for different α_0 and β_0 when $\boldsymbol{\mu_0} = (0, 0, .., 00)$ and $\sigma^2 = 2000$. The curves are averages over 100 runs.

at around $t = 500$ whereas with $\sigma^2 = 1$ this is not the case and cumulative regret keeps increasing. From the obtained results, we see that even though the differences in percentage of optimal selection are relatively small, the best performance of the algorithm is obtained when $\sigma^2 = 2000$ and $\boldsymbol{\mu}_0 = (0, 0, ..., 0)$. A final test on α_0 and β_0 in Figures 4.3 shows that, as α_0 gets larger, the performance improves, while changes in β_0 do not affect the result. Thus, a good choice for the prior hyperparameters is $\boldsymbol{\mu}_0 = (0, 0, ..., 0)$, $\sigma^2 = 2000$, $\alpha_0 = 20$ and $\beta_0 = 1$.

Moreover, in order to study possible advantages of contextual Thompson Sampling in the same seasonal environment, we plot the two measures of performance with the optimal set of prior hyperparameters for both standard Thompson Sampling and contextual Thompson Sampling. Figure 4.8 shows the performance of the algorithm when $\boldsymbol{\mu}_0 = (0, 0, ..., 0, 0)$, $\sigma^2 = 2000$, $\alpha_0 = 20$ and $\beta_0 = 1$ against the performance of the standard Thompson Sampling when $\mu_0 = 2000$, $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$. From this, it is evident that contextual Thompson Sampling outperforms dramatically standard Thompson Sampling, both in terms of percentage of optimal selection and cumulative regret.

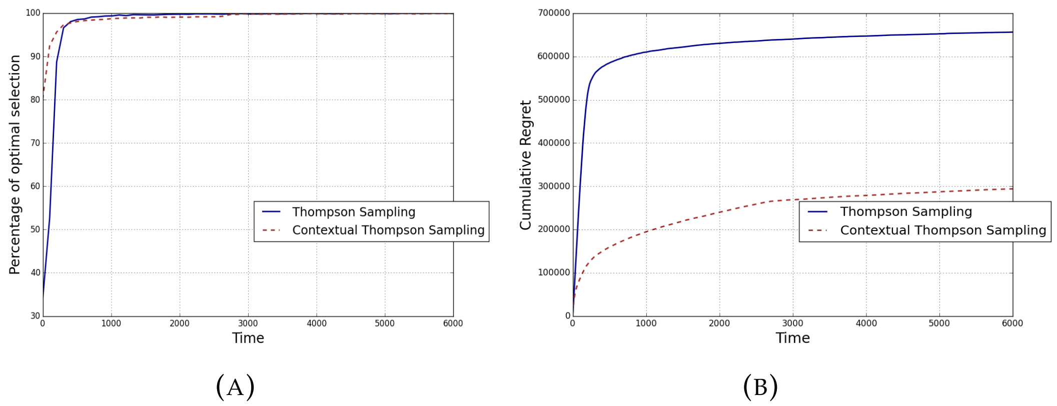
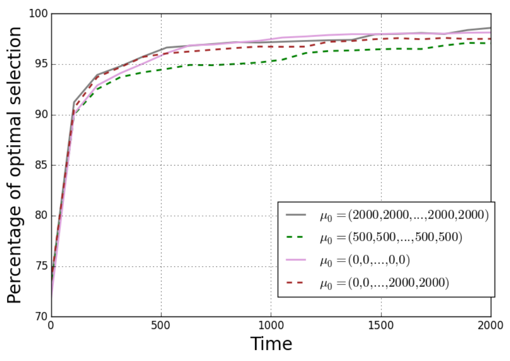


FIGURE 4.4: Comparison between the standard Thompson Sampling algorithm and contextual Thompson Sampling in the seasonal case for Dataset 1. The curves are averages over 100 runs.

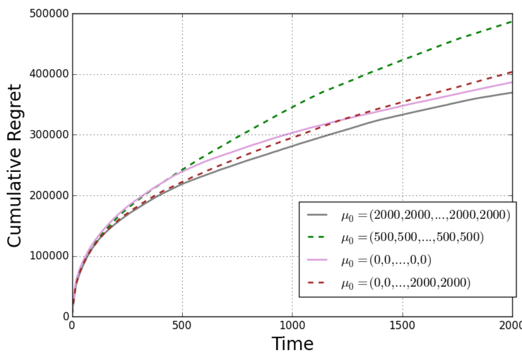
4.4.2 Results for Dataset 2

First, in Figure 4.5, we examine the dependence on the choice of $\boldsymbol{\mu}_0$. We can see that for all values of $\boldsymbol{\mu}_0$, the algorithm starts with 75% of optimal selection and at $t = 500$ all curves reach a percentage of optimal selection of $95 - 97\%$ which stays the same until the end of the experiment at $T = 2000$. Cumulative regret keeps increasing with a rate that gets smaller after $t = 500$. We observe that the best cumulative regret is reached when $\boldsymbol{\mu}_0 = (2000, 2000, ..., 2000, 2000)$ and $\boldsymbol{\mu}_0 = (0, 0, ..., 0, 0)$. Then we proceed with different values of σ^2 for these two starting vectors for μ_0. Figure 4.6 shows that, for most of the combinations tried, cumulative regret continues to increase until the end of the experiment. However, when $\boldsymbol{\mu}_0 = (2000, 2000, ..., 2000, 2000)$ and $\sigma^2 = 1$ and when $\boldsymbol{\mu}_0 = (0, 0, ..., 0, 0)$ and $\sigma^2 = 2000$, the rate of increase of cumulative regret gets much smaller and for both cases percentage of optimal selection reaches almost 99%, while for the rest the maximum percentage reached is 96%. This is what causes the continuous increase in cumulative regret. Finally, when testing different α_0 and β_0 values, the algorithm performs better when $\alpha_0 = 20$ and $\beta_0 = 10$.

In Figure 4.8, we make a comparison between contextual Thompson Sampling with hyperparameters $\sigma^2 = 1$, $\boldsymbol{\mu}_0 = (2000, 2000, ..., 2000, 2000)$, $\alpha_0 = 20$, $\beta_0 = 10$

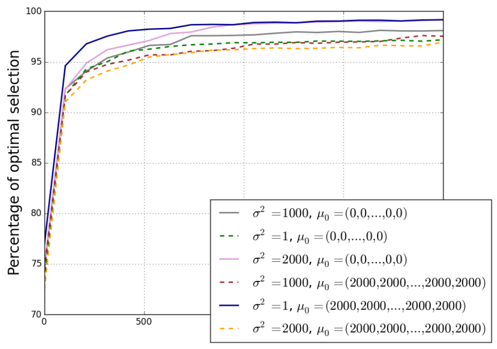


(A)

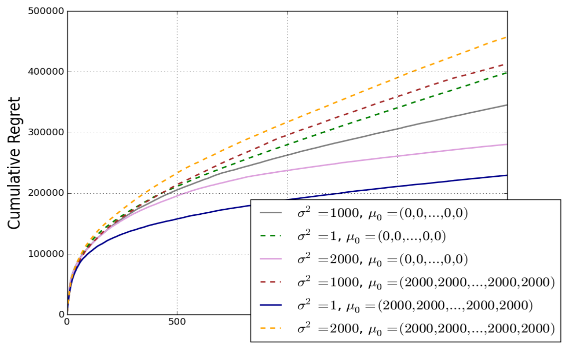


(B)

FIGURE 4.5: Percentage of optimal selection and cumulative regret for different μ_0 when $\sigma^2 = 1000$, $\alpha_0 = 1$, $\beta_0 = 1$. The curves are averages over 100 runs.



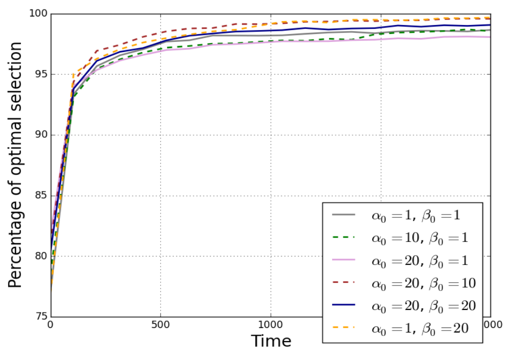
(A)



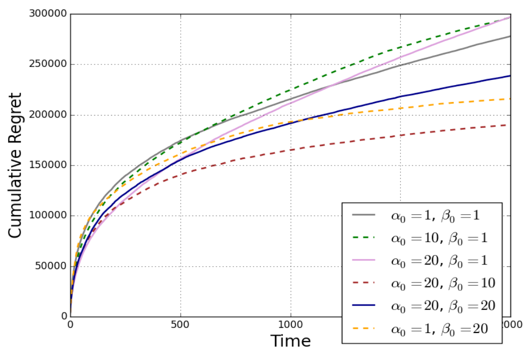
(B)

FIGURE 4.6: Percentage of optimal selection and cumulative regret for different μ_0 and σ^2 when $\alpha_0 = 1$, $\beta_0 = 1$. The curves are averages over 100 runs.

and standard Thompson Sampling with the optimal set of hyperparameters as they were stated in Section 3.6.2. We observe that contextual Thompson Sampling starts with a very high initial 80% of optimal selection while Thompson Sampling, which ignores seasonality, starts with just a 35% of optimal selection. This is what causes the large difference between their initial cumulative regret, which subsequently forces standard Thompson Sampling to make up for the huge regret it accumulates at the beginning. However, an interesting result is that, even though contextual Thompson Sampling starts with a much higher percentage of optimal selection, on average, it converges towards 100% much later compared to the standard Thompson Sampling which ignores seasonality. A further analysis of the results was carried out in order to investigate possible factors that caused this behaviour.

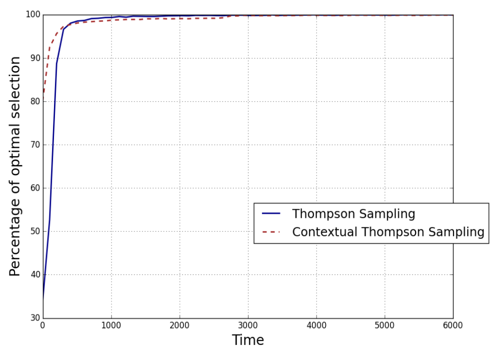


(A)

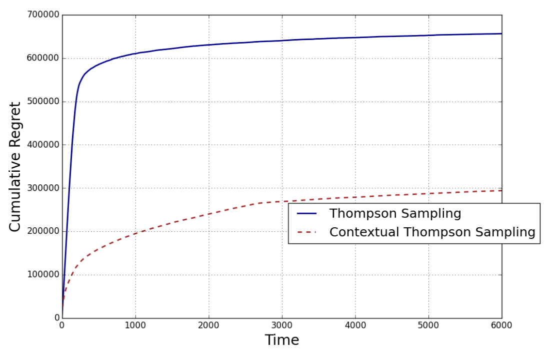


(B)

FIGURE 4.7: Percentage of optimal selection and cumulative regret for different α_0 and β_0 when $\mu_0 = (2000, 2000, .., 2000)$ and $\sigma^2 = 1$. The curves are averages over 100 runs.



(A)



(B)

FIGURE 4.8: Comparison between the standard Thompson Sampling algorithm and contextual Thompson Sampling in the seasonal case for Dataset 2. The curves are averages over 100 runs.

We investigate the results obtained from the percentage of optimal selection, at each single run, when $T = 2000$ and make a plot showing the fraction of runs in which each option was chosen. From Figure 4.9, we can see that even though in most of the simulation runs, the algorithm manages to reach either 99% or 100% of optimal selection by $t = 2000$, there are runs where the algorithm is choosing suboptimal arms throughout the experiment. This is what makes the average percentage over the 200 runs at the end of the experiment be less than the optimal 100%. A possible explanation for this could be the high variance of the best option. From Table 3.3, we can see that even though the true mean, m_2 of option 2 is much higher, its standard deviation, s_2 is also very high and this is probably what causes the delay in the convergence of contextual Thompson Sampling. Thus, a thorough exploration of their performance under high variance is recommended.

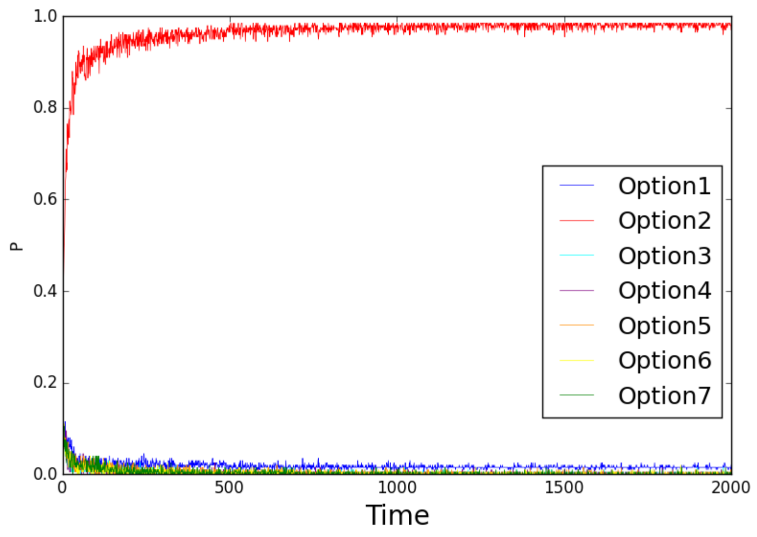


FIGURE 4.9: Percentage of runs in which contextual Thompson Sampling selects each option for Dataset 2 in the seasonal case (Option 2 is the best option).

4.4.3 Discussion

After a number of experiments performed on Thompson Sampling which uses the day of the week as contextual information, we observed that in the case where the performance of the different options are very similar, there is a very small dependence on the choice of prior hyperparameters. Moreover, the algorithm manages to find the best option and reach 100% of optimal selection at around the same time as it does for the stationary case. It is able to make a good estimate of how the mean value of the arms changes daily.

In Example 2, we observed that contextual Thompson Sampling needed a substantial number of iterations to reach 100% of optimal selection. Even though it exhibited with a high initial percentage of optimal selection, it reached a percentage of 95% quickly and stabilized for a significant amount of time before reaching the maximum of 100%. This is supported by the work of Russo et al. (2018) who suggest that in non stationary situations regret may not vanish completely by the end of the experiment, as we observe for $T = 2000$. Significantly higher values of T are needed to obtain 100% convergence, most likely due to the high variance of the best option. This increases significantly the cumulative regret obtained by the end of the experiment.

Only when we increased significantly the value of T we managed to reached a 100% and consequently minimize the regret. A possible explanation for this behaviour could be the high variance of the best option, which increases the uncertainty of the algorithm about whether the option that appears to be the best is indeed the best. So the increased variability results in the algorithm spending longer on exploration. This assumption is supported by an investigation of the fraction of simulations in which the optimal decision was made. Even though in most of the simulation runs the optimal 100% of optimal selection is reached, there are a few runs where there is only exploration and insufficient exploitation. This is in line with other previous empirical results (Amirizadeh and Mandava, 2015) which suggest that other MAB algorithms such as ε-greedy, softmax and UCB-Tuned are also dependent on the variance of the rewards.

Finally, in all of the examples described earlier, it is clear that when the algorithm uses the day of the week as contextual information in an environment with daily seasonality, the cumulative regret is less than the cumulative regret of the algorithm which ignores seasonality for the duration of the experiment. Bouneffouf and Feraud (2016) agree that for a non stationary problem an algorithm that assumes stationarity performs almost as badly as the random choice of options. Even in the case where the algorithm carries out more exploration in order to be certain about the best option, cumulative regret is always less than the cumulative regret of standard Thompson Sampling, mainly because contextual Thompson Sampling starts with a higher percentage of optimal selection.

4.5 Multi-armed bandits for unknown seasonality

We now consider the case where we have an initial belief that there is seasonality in the rewards but we know neither the period nor the level of the seasonality. As before, we are using the algorithm to identify the best option among a range of suboptimal candidates. The algorithm we developed fits different models to the data (rewards obtained from the chosen option), and at each time step, it firsts decides which model fits the data best, based on Akaike Information Criteria (AIC) (Akaike, 1974) and based on this model, it uses the respective version of Thompson Sampling in order to decide which option to choose. Thus, in addition to the option selection step that is fundamental for any MAB methodology, the algorithm has one additional step: the model selection step. Based on the method used for the model selection, we name the algorithm AIC-TS and we assume that, for each period, there are a finite number of options.

The two main methodologies we use for the online learning and option selection are the standard formulation of the Bayesian algorithm and the contextual Thompson Sampling explained earlier in Section 4.3, with the number of binary variables varying in order to represent daily, monthly and both daily and monthly seasonality. A similar idea was used in one of the pricing policies presented by Geer et al. (2018) when trying to solve a dynamic pricing problem with unknown demand. In this context in order to capture the right demand function and at the same time find a price that maximizes revenue at each time step, the different arms are four different demand models and, depending on the demand model that was chosen at each time step, a decision about the best price was made.

In our case, as an initialization stage, we choose each option once to get some initial information about their performance. When a decision is made about which option to choose at each time step τ, we update the four hyperparameters of the models of seasonality using the information obtained so far about the performance of the different options. In the case where the different options follow a stationary reward distribution, the update equations used are Equations 3.4 whereas in the remaining models where the reward distribution of the different options follow some seasonality, the update equations used are Equations 4.6 with $\mathbf{y}$ being a vector with τ entries representing the rewards obtained up to time τ. Algorithm 7 presents the mains steps of the methodology, which we call AIC-TS, in the case of four different models for seasonality, which are explained below:

4.5.1 Reward models

The four different models represent different initial assumptions about changes in the reward distribution of the different options.

- **Model 1**
  The reward of the seven different options follows a stationary normal distribution with unknown mean and variance and standard Thompson Sampling is used to decide which option to choose at each time step.

- **Model 2**
  The reward of the seven options exhibits seasonality with a period of one week and the methodology used is based on contextual Thompson Sampling with six variables representing the day of the week in which an option is chosen and seven variables representing the seven candidate options.

- **Model 3**
  The reward of the seven options exhibits seasonality with a period of one year and based on contextual Thompson Sampling, we introduce 11 variables to represent the month of the year that an option was chosen and seven variables representing the seven candidate options.

- **Model 4**
  The reward of the seven options exhibits two levels of seasonality with periods of one week and one year respectively and based again on contextual Thompson Sampling we introduce 17 variables that represent the day of the week (six variables) and the month of the year (11 variables) that an option was chosen and seven variables representing the seven candidate options.

Algorithm 7 AIC-TS

```
for each $\tau \in \{1, 2, ..., T\}$ do
    Find day and month
    if $1 \leq \tau \leq 7$ then
        Initialization:
        if $\tau = 1$ then
            Set the initial values of the prior hyperparameters of the different models
        end if
        Choose a random option and get its real reward $r_\tau$
    else if $\tau = 8$ then
        Select a random model $M$
    else
        Select the model such that $M = argmin_{M=1,2,3,4} AIC_M$
        if M=1 (no seasonality) then
            Select the best option based on Algorithm 3
        else if M=2 (daily seasonality) then
            Select the best option using the model with daily seasonality based on Algorithm 6
        else if M=3 (monthly seasonality) then
            Select the best option using the model with monthly seasonality based on Algorithm 6
        else if M=4 (daily and monthly seasonality) then
            Select the best option using the model with both daily and monthly seasonality based on Algorithm 6
        end if
    end if
    Get the real reward of the best option and calculate regret
    Update the hyperparameters of the 4 models
    Update the AICs values of the 4 models
end for
```

4.5.2 Model selection

For model selection, the algorithm uses the Akaike Information Criterion (Akaike, 1974), at each time step, which is calculated as $AIC = -2ln(\hat{L}) + 2k$ where $\hat{L}$ is the maximum value of the likelihood function for the model and k is the number of parameters. The AIC was chosen to account for the number of parameters when judging the quality of the model fit. When comparing different models, the best model is the one with the minimum AIC value.

In the stationary case, i.e. when $M = 1$, the equation of the maximum log likelihood is:

$$LL = \frac{-n}{2}\ln(2\pi) - \frac{1}{2}\sum_{a=1}^{N} n_a \ln(\hat{\sigma}_a^2) - \sum_{a=1}^{N}\sum_{i=1}^{n_a} \frac{(r_i - \hat{\mu}_a)^2}{2\hat{\sigma}_a^2}, \tag{4.8}$$

where r_i is the i^{th} reward of option a, n_a is the sample size of option a and n is the total sample size. We note that $\hat{\mu}_a$ and $\hat{\sigma}_a^2$ are the mean and variance of option a that maximize the log likelihood calculated as:

$$\hat{\mu}_a = \tfrac{1}{n_a}\textstyle\sum_{i=1}^{n_a} r_i$$

$$\hat{\sigma}_a^2 = \tfrac{1}{n_a}\textstyle\sum_{i=1}^{n_a}(r_i - \hat{\mu}_a)^2$$

Moreover, for linear regression, i.e. when $M = 2, 3$ or 4 the maximum log likelihood is:

$$LL = \frac{-n}{2}\ln(2\pi) - \frac{n}{2}\ln(\hat{\sigma}^2) - \frac{1}{2\hat{\sigma}^2}(\mathbf{r} - X\hat{\boldsymbol{\mu}})^T(\mathbf{r} - X\hat{\boldsymbol{\mu}}) \tag{4.9}$$

where X is the design matrix with binary variables representing the history of seasonal factors and options chosen so far and $\mathbf{r}$ is a vector with the rewards obtained. The mean and variance that maximize the log likelihood are:

$$\hat{\boldsymbol{\mu}} = (XX^T)^{-1}X^T\mathbf{r}$$

$$\hat{\sigma}^2 = \tfrac{1}{n}(\mathbf{r} - X\hat{\boldsymbol{\mu}})^T(\mathbf{r} - X\hat{\boldsymbol{\mu}})$$

The dimensions of $\mathbf{X}$ depend on the number of observations obtained so far and the time those observations were made. Since for the first seven time steps, which account for seven days of the week, all candidate options were chosen once, in Model 2, the number of columns for X is the same as the number of columns of X used in the update of the reward model (13 columns). However, in the case of Model 3 and Model 4, matrix X has a different number of columns for $\tau < 365$ since in this period of time there are months with no observations. For $\tau < 365$, X in Model 3 initially has seven columns for the seven options and a column is added to the matrix once an observation from a new month is obtained. Thus, when $\tau = 365$, X ends up having 18 columns which stay the same until the end of the experiment; 11 for the months with January being the base month and seven for the candidate options. The same idea holds for Model 4 where X has initially

13 columns (six for the days of the week and seven for the candidate options) and for $\tau > 365$ it has 24 columns (six for the days, 11 for the months and seven for the options). This mechanism is used to guarantee the right calculation of the maximum log likelihood used when calculating the AIC.

Below we describe the update of the parameters for each model:

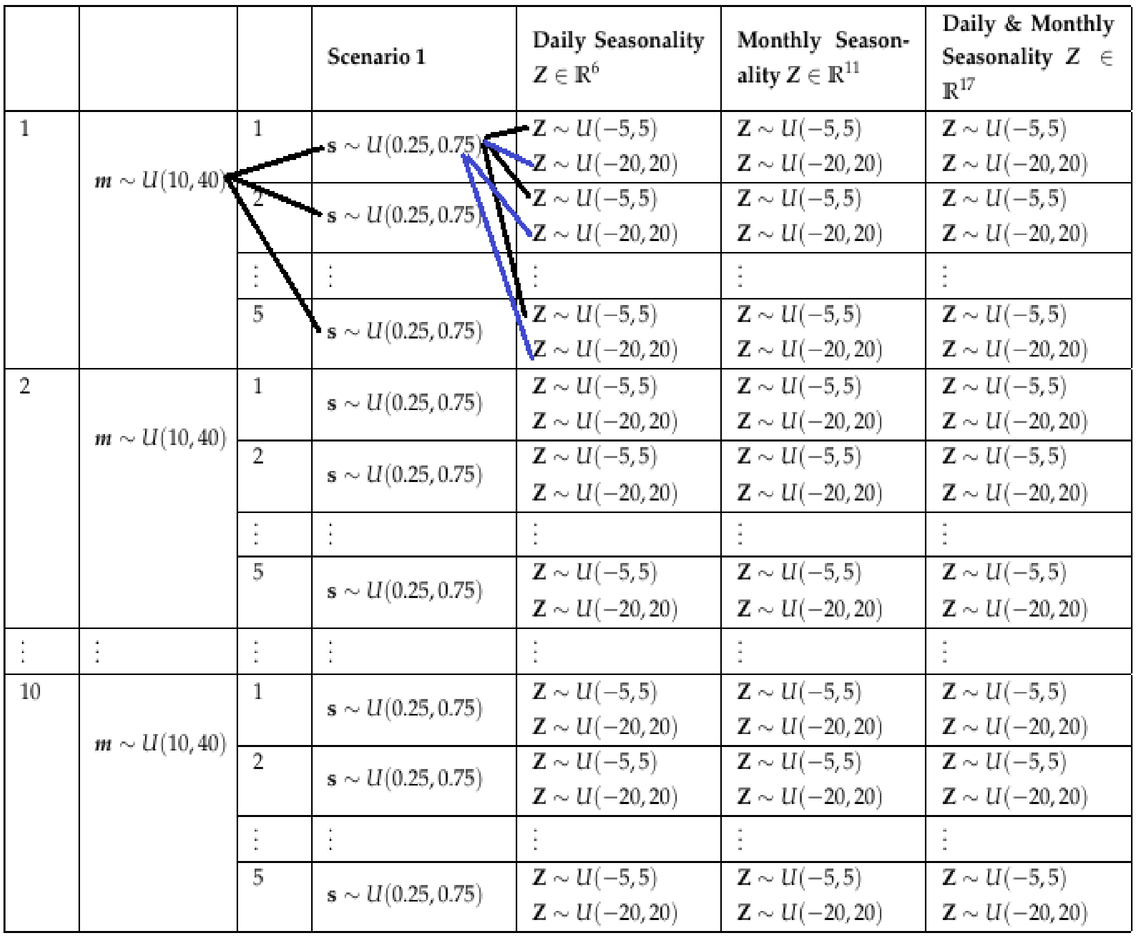
- Model 1 - Stationary reward distribution. In this model, $\bar{y}$ is the mean reward observed so far and n is the total sample size.
- Model 2 - Reward distribution with daily seasonality. In this case, $\boldsymbol{\mu}$ is a 13-element vector and V is a 13×13 matrix. We update the model with daily seasonality using Equations 4.6 where X is a $\tau \times 13$ matrix with binary entries such that the first six columns represent the day of the week and the remaining seven columns represent the seven candidate options.
- Model 3 - Reward distribution with monthly seasonality. In this case, $\boldsymbol{\mu}$ is a 18-element vector and V is 18×18 matrix. We update the model with monthly seasonality using Equations 4.6 where X is $\tau \times 18$ columns with binary entries such that the first 11 columns represent the month of the year and the remaining seven columns represent the seven candidate options.
- Model 4 - Reward distribution with monthly and daily seasonality. In this case, $\boldsymbol{\mu}$ is a 24 element vector and V is a 24×24 matrix. We update the model with both daily and monthly seasonality using Equations 4.6 where X is $\tau \times 24$ columns with binary entries such that the first six columns represent the day of the week, the following 11 columns represent the month of the year and the remaining seven columns represent the seven candidate options.

4.6 Numerical experiments on artificial datasets

4.6.1 Setting up the artificial datasets

Artificial datasets were created to test the performance of AIC-TS, the different versions of contextual Thompson Sampling and the standard formulation of Thompson Sampling. In order to do that, first we create a total of 150 different pairs of values that represent the real mean and standard deviation of the seven candidate options we are comparing. Following the idea we previously used for the creation of the artificial datasets in the stationary case, we sample 10 different sets for the expected reward of the seven options from the uniform distribution $U(10, 40)$. In order to account for different variability in the reward of the different options we create five different sets of standard deviation for each set of mean values. Thus, for Scenario 1 $\mathbf{s} \sim U(0.25, 0.75)$, for Scenario 2 $\mathbf{s} \sim U(2.5, 7.5)$ and for Scenario 3 $\mathbf{s} \sim U(25, 75)$. The aim of AIC-TS that combines ideas from MAB and statistical model selection is twofold. First, the algorithm needs to be able to find what the right seasonality is and at the same time minimize regret.

For each of the three scenarios of variability in rewards, we test the algorithms in three different seasonal environments: daily seasonality, monthly seasonality, both daily and monthly seasonality. For each environment, we consider two levels for seasonality, low and high. In the case of low seasonality, $\boldsymbol{Z} \sim U(-5, 5)$ and in the case of high seasonality, $\boldsymbol{Z} \sim U(-20, 20)$, where $\boldsymbol{Z}$ is a set of seven seasonal additive factors for each of the seven options. For each combination of $\mathbf{m}$ and $\mathbf{s}$, there exists a distinct set of seasonal factors, which is different for each period and level of seasonality. Thus, for each level-period of seasonality a total of 150 sets of seasonal factors are created in order to match every combination of m and s for all the three scenarios. Table 4.10 shows the mechanisms used to create the 50 different datasets for each level-period of seasonality for Scenario 1. The same mechanism is also used for Scenario 2 and Scenario 3 to create a total of 150 datasets. The results presented are averaged over 100 independent individual runs for each combination of parameters and seasonal additive factors.



			Scenario 1	Daily Seasonality $\mathbf{Z} \in \mathbb{R}^6$	Monthly Seasonality $\mathbf{Z} \in \mathbb{R}^{11}$	Daily & Monthly Seasonality $Z \in \mathbb{R}^{17}$
1	$m \sim U(10,40)$	1	$\mathbf{s} \sim U(0.25,0.75)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$
		2	$\mathbf{s} \sim U(0.25,0.75)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$
		⋮	⋮	⋮	⋮	⋮
		5	$\mathbf{s} \sim U(0.25,0.75)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$
2	$m \sim U(10,40)$	1	$\mathbf{s} \sim U(0.25,0.75)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$
		2	$\mathbf{s} \sim U(0.25,0.75)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$
		⋮	⋮	⋮	⋮	⋮
		5	$\mathbf{s} \sim U(0.25,0.75)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
10	$m \sim U(10,40)$	1	$\mathbf{s} \sim U(0.25,0.75)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$
		2	$\mathbf{s} \sim U(0.25,0.75)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$
		⋮	⋮	⋮	⋮	⋮
		5	$\mathbf{s} \sim U(0.25,0.75)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$	$\mathbf{Z} \sim U(-5,5)$ $\mathbf{Z} \sim U(-20,20)$

FIGURE 4.10: Overview of setting up the artificial datasets for Scenario 1. **m** and **s** are sets with seven values representing the real mean and the standard deviation respectively for each of the seven options. Each **m** is paired with each **s** to create a total of 50 different datasets and then matched with a different **Z**, a set with seven values representing the seasonal factors for each level-period of seasonality

4.6.2 Results

The initial hyperparameters used in the experiments are the following:
Model 1: $\mu_0 = 25$, $\kappa_0 = 1$, $\alpha_0 = 1$ and $\beta_0 = 1$
Model 2,3,4: $\boldsymbol{\mu}_0 = (0, 0, 0, .., 25, 25, 25)$, V_0 is a diagonal matrix with the value 25 in the main diagonal, $\alpha_0 = 1$ and $\beta_0 = 1$. Model 2 is the model that assumes daily seasonality, Model 3 is the model that assumes monthly seasonality and Model 4 is the model that assumes both daily and monthly seasonality.

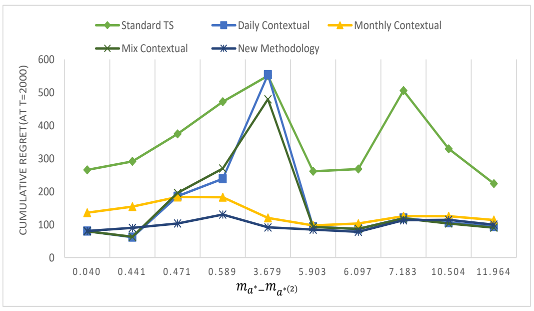
The first table summarizes the results of all algorithms for cumulative regret at the end of the experiment ($t = 2000$). We split the results into different categories based on the different seasonality and the variability in rewards between the seven candidate options we are comparing. We observe that the results are consistent for daily and monthly seasonality. In both cases, when the seasonality is either low or high, the new methodology achieves the lowest regret in most of the situations. However, this does not hold when the variability between the different options is moderate. In this case, when the seasonality is low, the contextual Thompson Sampling that assumes the right seasonality becomes the new best and when the seasonality is high, the contextual Thompson Sampling that assumes both daily and monthly seasonality performs the best. Similar results are obtained for environments in which both monthly and daily seasonality are present. The only exception is for the high seasonality and the high variability in the environment with monthly and daily seasonality.

In order to explore further the factors that affect the performance of the different algorithms, we make separate graphs for cumulative regret at t=2000 for the different seasonal environments and for the different scenarios. On the x-axis we have the difference in the expected reward between the first and the second best options as they were generated in the simulations ($m_{a^*} - m_{a^{*(2)}}$). From the graphs it is clear that the cumulative regret of the new methodology is not affected significantly by this difference and even when the difference between the expected reward of the two best options is very small, in most cases the new methodology maintains the lowest regret. This is because even though the percentage of optimal selection might be very low, the option that it chooses very often is the second best and this does not cause a big increase in the regret.

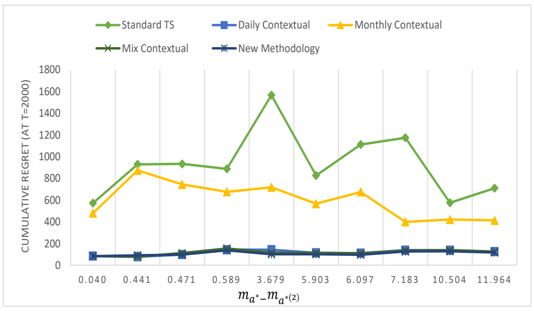
Seasonality	**Var. in rewards**	**Standard TS**	**Daily Cont.**	**Monthly Cont.**	**Both Cont.**	**AIC-TS**
Low Daily	**Low**	354.03	161.67	133.91	157.71	98.34
	Moderate	496.54	281.54	300.85	281.97	408.85
	High	5840.64	5798.55	5824.11	5832.97	4916.86
High Daily	**Low**	927.95	116.84	595.79	116.66	107.64
	Moderate	989.26	293.39	715.42	287.65	447.05
	High	5835.26	5824.89	5933.83	5833.44	5295.27
Low Monthly	**Low**	373.65	176.37	174.27	172.09	96.23
	Moderate	514.33	314.21	279.94	288.49	415.59
	High	5738.81	5836.58	5830.62	5907.18	4942.20
High Monthly	**Low**	1111.18	912.08	137.09	138.09	97.40
	Moderate	1326.91	1004.00	283.98	281.27	454.79
	High	5995.48	5928.24	5831.87	5875.40	5583.22
Low Mix	**Low**	397.02	166.21	128.94	167.37	99.59
	Moderate	537.86	317.14	309.66	288.86	437.94
	High	5777.24	5797.86	5829.51	5827.58	5076.26
High Mix	**Low**	1636.49	1039.77	603.84	124.24	109.17
	Moderate	1647.17	975.61	744.45	289.76	494.31
	High	6000.22	5887.10	5937.40	5856.14	6057.74

TABLE 4.2: Cumulative regret at $t = 2000$ for standard Thompson Sampling,for contextual Thompson Sampling that assumes daily, monthly, both daily and monthly seasonality and for the new methodology. We highlight the smallest value for each row.

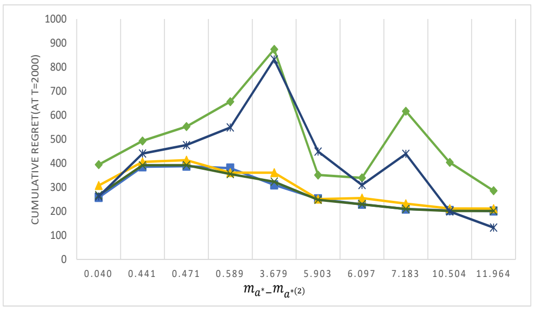
We note that in many cases there is a sudden peak when the difference is 3.67. With a further look at the artificial datasets we can see that for these datasets the difference between the expected reward of the best and the second best is in fact very close to the difference between the second and the third best option. Thus, it is possible that the algorithm spends a significant amount of time in the exploration of suboptimal options which in fact have very similar performance. This peak is even more obvious in situations with low seasonality. These results give us a very useful insight into how the difference between the observed performance of the different options affect each of the algorithms. It is clear that in any seasonal environment (daily, monthly, both daily and monthly), when there is high variability in the rewards of the different options, as $m_{a^*} - m_{a^{*(2)}}$ increases, the final cumulative regret increases for all algorithms. This is because the algorithm rarely reached 100% of optimal selection.



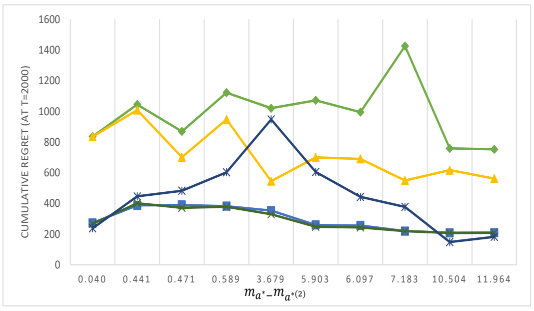
(A) Low daily seasonality-Scenario 1



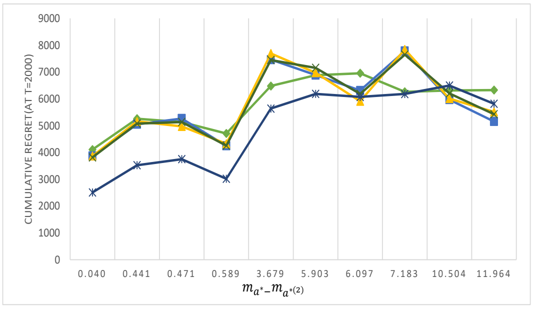
(B) High daily seasonality-Scenario 1



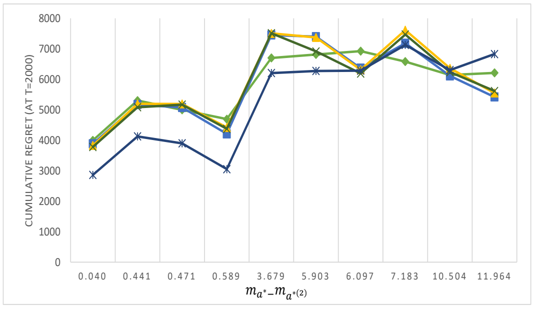
(C) Low daily seasonality-Scenario 2



(D) High daily seasonality-Scenario 2

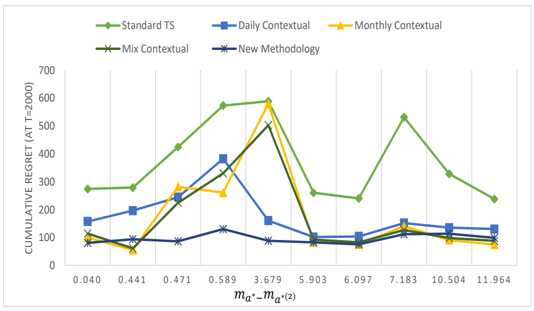


(E) Low daily seasonality-Scenario 3

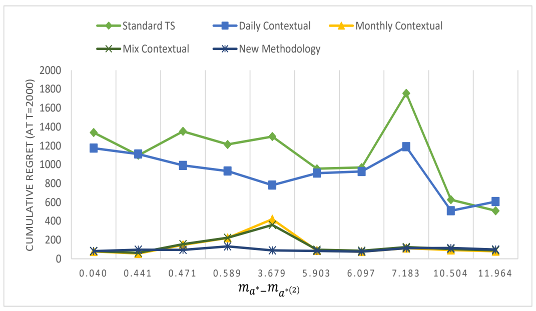


(F) High daily seasonality-Scenario 3

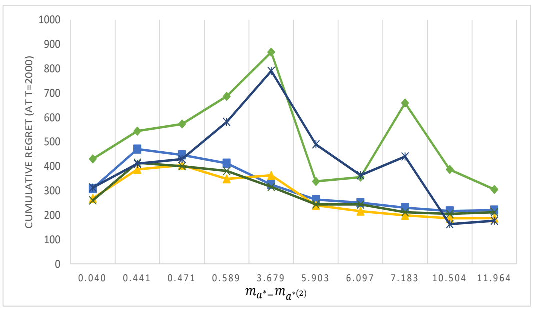
FIGURE 4.11: Cumulative regret for the different algorithms against the difference between the expected value of the best and the second best option when there is daily seasonality. The first column is for low seasonality and second column is the high seasonality. A-B, C-D, E-F are for Scenario 1, Scenario 2, and Scenario 3 respectively



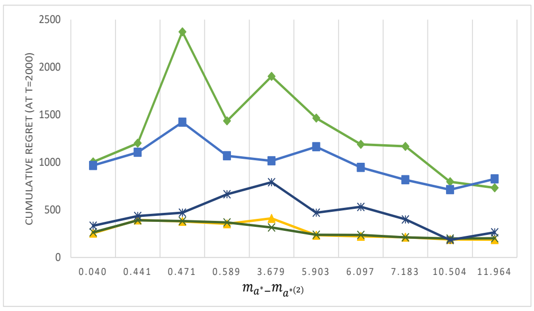
(A) Low monthly seasonality-Scenario 1



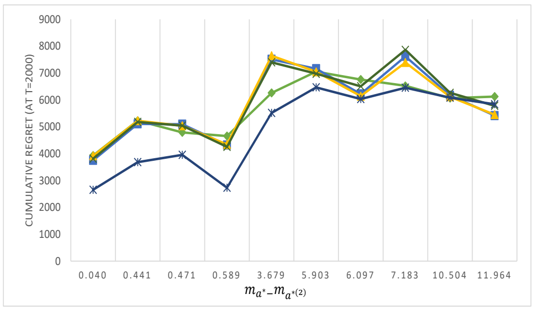
(B) High monthly seasonality-Scenario 1



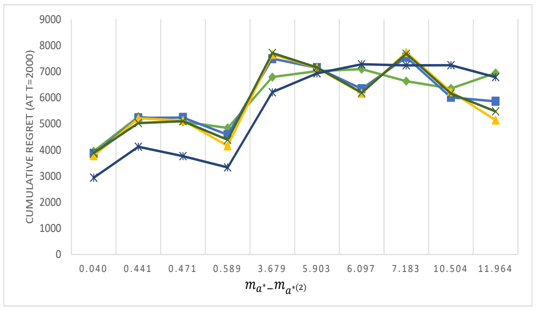
(C) Low monthly seasonality-Scenario 2



(D) High monthly seasonality-Scenario 2

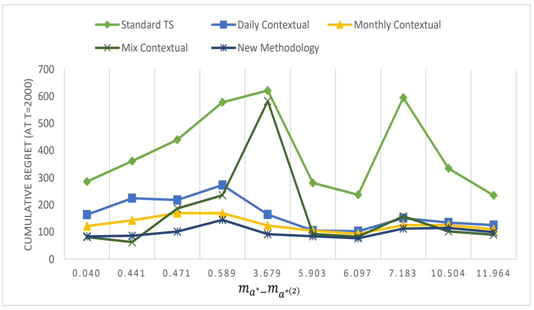


(E) Low monthly seasonality-Scenario 3

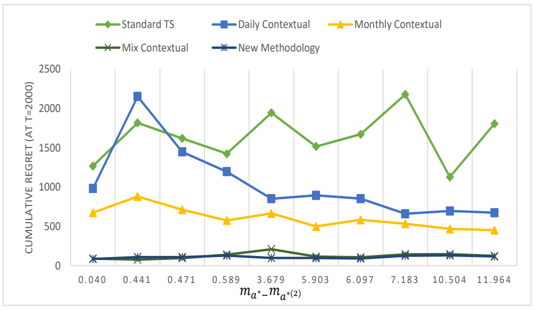


(F) High monthly seasonality-Scenario 3

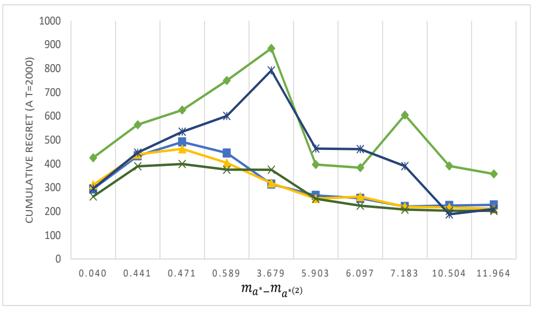
FIGURE 4.12: Cumulative regret for the different algorithms against the difference between the expected value of the best and the second best option when there is monthly seasonality. The first column is for low seasonality and second column is the high seasonality. A-B, C-D, E-F are for Scenario 1, Scenario 2, and Scenario 3 respectively



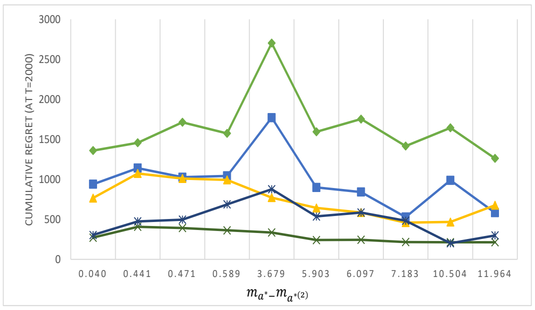
(A) Low daily and monthly seasonality-Scenario 1



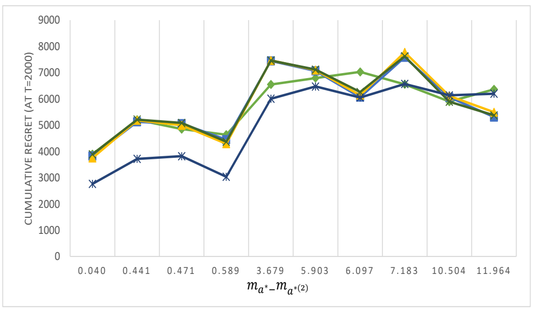
(B) High daily and monthly seasonality-Scenario 1



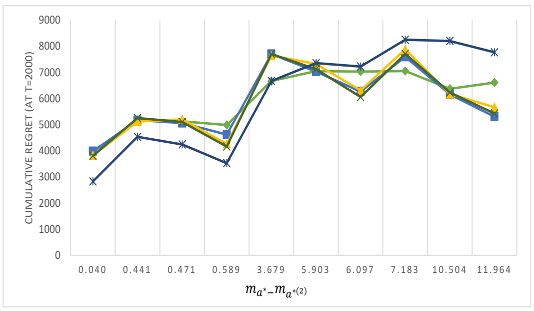
(C) Low daily and monthly seasonality-Scenario 2



(D) High daily and monthly seasonality-Scenario 2



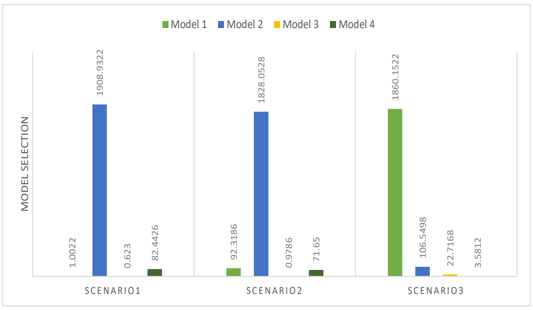
(E) Low daily and monthly seasonality-Scenario 3



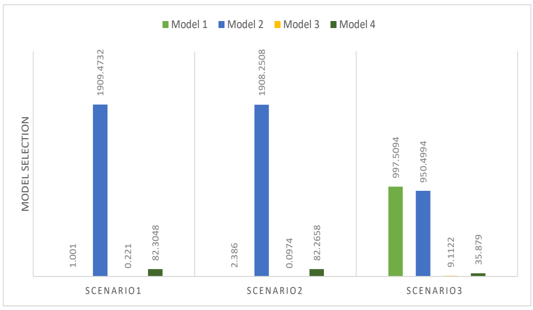
(F) High daily and monthly seasonality-Scenario 3

FIGURE 4.13: Cumulative regret for the different algorithms against the difference between the expected value of the best and the second best option when there is daily and monthly seasonality. The first column is for low seasonality and second column is the high seasonality. A-B, C-D, E-F are for Scenario 1, Scenario 2, and Scenario 3 respectively

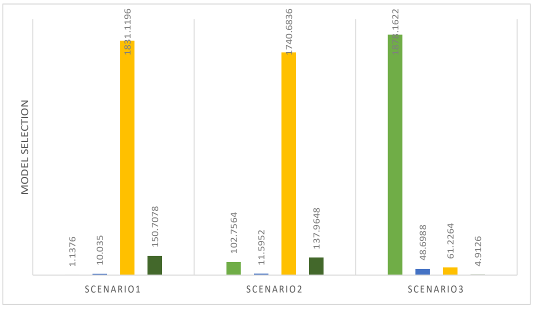
In Figure 4.14, we can have a closer look at the model selection of AIC-TS in the different seasonal environments. It shows on average over the 200 runs how many times in the 2000 time steps the algorithm chooses each of the different models. It is clear that for Scenarios 1 and 2 it usually chooses the right model. In the case of low seasonality, the algorithm manages to choose much more often the right model in Scenario 1 and Scenario 2. However, in Scenario 3 due to the high variability in the rewards obtained in an environment with high seasonality, the algorithm selects the stationary model more often. This suggests that when the variability is high, it masks the seasonal variation. Similar results are obtained for the case of high seasonality with the difference that, even in Scenario 3, the algorithm now manages to choose the correct model a large number of times - close to the number of times it chooses the stationary model. Combining the results from Figures 4.11, 4.12, 4.13 and Figure 4.14, we see that there is no obvious correlation between model selection and regret. Even in situations where the algorithm does not choose the right model more often, cumulative regret does not increase significantly.



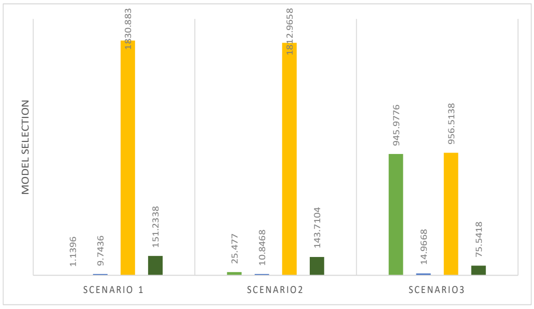
(A) Low daily seasonality
Correct model: Model 2



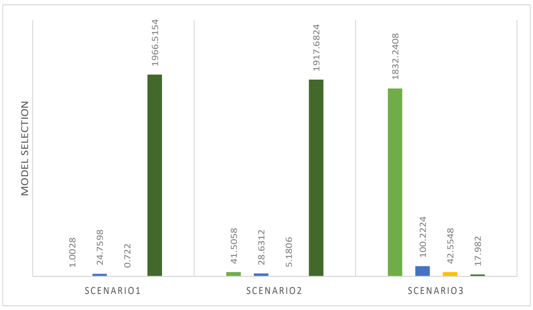
(B) High daily seasonality
Correct model: Model 2



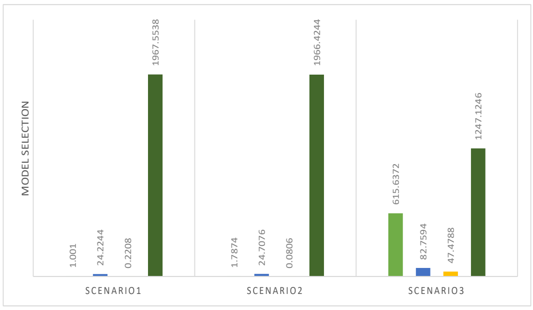
(C) Low monthly seasonality
Correct model: Model 3



(D) High monthly seasonality
Correct model: Model 3



(E) Low daily and monthly seasonality
Correct model: Model 4



(F) High daily and monthly seasonality
Correct model: Model 4

FIGURE 4.14: Model selection of the new methodology in 2000 time steps. The results are averaged over 200 individual runs

4.6.3 Discussion

The purpose of the above study is twofold. Firstly, we wanted, as we did in he previous chapter, to study how the real performance of the candidate options affects the performance of the algorithms when there is some seasonality in their reward distribution. For this reason, we tested standard TS, AIC-TS and contextual TS under different seasonality assumptions on our artificial datasets and on different seasonal environments. Secondly, we wanted to examine whether selecting the right model (the one closest to the true model) leads to better decisions and maximization of rewards (or minimization of regret).

Firstly, from the results it is clear that our algorithm (AIC-TS) performs very well, especially in cases with low and high variability in the rewards of the different options. In cases where its performance is not the best one, it is very often close to the version of the algorithm that knows the exact seasonality structure. Additionally, the results are in agreement with the main outcomes of Besbes and Zeevi (2015) who suggested that wrong models can lead to good and in some cases better decisions. Based on the results obtained for the cumulative regret combined with the results for the models selection for AIC-TS, it becomes apparent that the right model selection does not always lead to the lowest cumulative regret. Also, there are cases (low seasonality-Scenario 3) where even though AIC-TS selects a wrong model more often, its performance is not negatively affected .

4.7 Concluding remarks

This chapter considered examples in which rewards exhibit seasonality, a typical characteristic of reward data from website optimization applications. Two real-world datasets were considered initially before moving onto more rigorous tests on a large number of datasets. To account for seasonality, we implemented a contextual Thompson Sampling algorithm and a more advanced algorithm, AIC-TS, that aims to detect seasonality within the rewards.

Then based on the assumption that, when there is seasonality in the rewards of the different options, contextual Thompson Sampling performs better than standard Thompson Sampling that assumes stationarity, we wanted to observe whether the initial assumption about the period of seasonality affects the performance of contextual Thompson Sampling. Hence, we created a number of artificial datasets to test the standard formulation of contextual Thompson Sampling under different assumptions about the underlying seasonality. The artificial datasets created are for different period and level of seasonality. The results showed that the performance of the algorithms depends on both the period-level of seasonality and the variability in rewards of the different candidate options and the initial assumption about the period of seasonality does have an impact on the outcome.

In order to overcome the possibility of a wrong initial assumption on seasonality, we developed AIC-TS, an algorithm that combines contextual Thompson Sampling with statistical model selection. Its performance is definitely better than contextual Thompson Sampling with a wrong initial assumption and in some cases it is better than the one with the right assumption. This suggests that the correct model selection does not necessarily mean superiority in performance. Additionally, contextual Thompson Sampling with different initial assumptions was tested against standard Thompson Sampling and AIC-TS and it was clear that an additional factor that has an impact on their performance is the difference in the mean performance of the best and the second best option.

Chapter 5

Dynamic pricing application

This chapter introduces an application of multi-armed bandits to an area of particular interest, where the trade-off between exploration and exploitation is applied to maximize long-term revenue. We consider a situation where prices can be frequently adjusted throughout the selling season, making this a *dynamic pricing* or *optimal pricing* problem (Boer, 2015). Developing methods that simultaneously learn about the market and optimize prices is a growing area of research in revenue management. This has come about through the extensive development of online stores that give retailers the opportunity to change prices more frequently and the large range of products on sale, which means that stores need to automate price decisions. For more particular applications of pricing and revenue management, we refer the reader to Özer, Ozer, and Phillips (2012) and Talluri and Van Ryzin (2006).

It has been established that frequent price changes can be beneficial for any company that has the capacity to make changes to the price assigned to any product (Elmaghraby and Keskinocak, 2003; *Quantifying the benefits of dynamic pricing in the mass market*). Hence, online learning algorithms that aim to balance the trade-off between exploration and exploitation are used to estimate the underlying demand curve of a product, detect any seasonal changes that may affect the demand and change the price accordingly in order to maximize total revenue. Exploration phase can be defined as the period of time during which the retailers collect information about the demand of each candidate price while exploitation is the phase where they choose the most profitable price in order to maximize revenue.

The work in this chapter is motivated by the Dynamic Pricing Challenge, held on the occasion of the 17^{th} INFORMS Revenue Management and Pricing Section Conference on June 29-30, 2017 (Geer et al., 2018). The aim was to develop a dynamic pricing algorithm for a single product with no capacity constraints in order to evaluate the performance of different strategies in a competitive market environment with incomplete information. Based on the results and the managerial insights of three of the algorithms submitted, which are loosely based on MAB, we develop a new pricing strategy. In this new algorithm, we use Thompson Sampling and apply some of the recommended improvements in order to achieve high revenue and good performance against other competitors in the market.

In Section 5.1, we give an overview of the existing literature on revenue management and dynamic pricing, which is followed by the problem definition and the description of the demand mechanism we use to generate the demand in our experiments along with the experimental design, in Sections 5.2, 5.2.1 and 5.3 respectively. In Section 5.4, we describe the pricing strategy we have developed and then in Section 5.5, we give a brief overview of three other pricing strategies from the literature. Section 5.6 presents the results when running just our pricing strategy without any other competitors in the market, in a monopoly, and when running our algorithm against the other three competitors in two different settings: oligopoly where the four algorithms are run all together against each other and duopoly where the four algorithms compete in a round-robin setup. Lastly, Section 5.7 concludes the chapter.

5.1 Related literature

In a dynamic pricing problem without inventory constraints, the retailer has to choose real-time prices for different products in order to maximize revenue. A frequent assumption that reflects the complexity of the problem in many real-life situations, is that the demand function is unknown and stochastic; i.e we may have some prior information but we do not know the exact relationship between the price of a product and the demand earned and the realised demand is stochastic. This situation was first formulated as a problem of learning and earning in the area of economics by Rothschild (1974), who introduced the idea of price experimentation in order to learn the underlying demand function. Until then, the majority of firms were assuming a known demand function based on previous observations of the market. A vast amount of literature exists on such learning and earning problems (Besbes and Zeevi, 2009; Keskin and Zeevi, 2014; Ferreira, Simchi-Levi, and Wang, 2018; Besbes and Zeevi, 2015; Misra, Schwartz, and Abernethy, 2019a; Ban and Keskin, 2018; Boer and Keskin, 2017). A review paper by Boer (2015) provides a good overview of the area, along with additional limitations and modifications of the standard problem that exist in different environments.

Surprisingly, an important study by Besbes and Zeevi (2015) shows that in the case of an unknown demand function, even if the model assumed at the beginning of the learning procedure is incorrect, it may lead to correct pricing decisions under fairly general assumptions. In their study, they conclude that since misspecification of the demand model does not significantly harm the revenue earned by the pricing policy, using a simple linear model can be a good strategy that can perform well in a range of scenarios.

In the framework of MAB, this problem consists of two phases: the demand learning phase, which represents the exploration phase and the maximizing revenue phase, which represents the exploitation phase. The different arms are the different prices that the retailer can offer for each product and regret is defined as the difference between the optimal cumulative revenues when we know the demand distribution and the cumulative revenues of the firm's policy. (Besbes and Muharremoglu, 2013). Intuitively, this can be thought of as the lost profits due to experimentation (Misra, Schwartz, and Abernethy, 2019b). Thus, the decision maker can choose either exploitation by selecting the price that minimizes regret at the current time step, based on current information, or exploration, in order to earn more information that will be used in the future.

A MAB methodology is incorporated in the dynamic pricing policy of Misra, Schwartz, and Abernethy (2019b) in order to maximize revenue when the retailer has to decide between a number of different prices for a product and has limited information about the demand model. They solve the problem as a dynamic optimization problem and their suggested policy is based on UCB (Upper Confidence Bound). All of the above methods are designed to work in a monopoly market environment where the retailer aims to develop a pricing strategy that maximizes his own revenue when there are no other competitors in the market. A recent study (Geer et al., 2018) shows numerical results for a number of different algorithms developed to solve a dynamic pricing problem in an oligopoly setting where all competitors are compared against each other simultaneously and in a duopoly setting where the competitors compete in a round-robin setup. An important conclusion they draw is that taking into account competitors' decisions in the process of decision making significantly increases the revenue gained when compared to pricing strategies that ignore competitors' behaviour. This is even clearer in situations where there is a large number of competitors and/or the price sensitivity of the customers is high.

Many real life examples of dynamic pricing can be formulated as contextual bandit problems. The additional contextual information that the retailer is able to use can be competitor prices (Ferreira, Simchi-Levi, and Wang, 2018) or information about individual customers and their preferences. Ferreira, Simchi-Levi, and Wang (2018) present simulation results for the case where contextual information consists of competitors' selling prices. In their approach they use regression for every contextual information-purchase decision pair and they examine two different cases for the contextual information: the case where competitors' prices come from a uniform distribution and the case where they follow a Bernoulli distribution. Then, for each of the two cases, they compare the performance of the algorithm when they use contextual information and when they use only the Thompson Sampling- update algorithm. Between the two distributions, they concluded that Bernoulli context requires fewer samples and thus the performance in terms of percentage of correct selection is better compared to the Uniform distribution.

5.2 Problem definition

We study the same problem described by Geer et al. (2018). The aim is to develop a pricing strategy for a single product without inventory constraints under an unknown demand mechanism. The only information given about possible prices for the product is that "it seems unlikely that posting prices higher than 100 is optimal". The experiment consists of 1000 time steps and it is assumed that at each time step the algorithm can observe its own sales and the prices of the other competitors in the market and it is not able to see the sales figures for other competitors. The algorithm uses this information to set a price to charge in the next time period.

We begin by presenting results for a monopoly, where there are no other competitors in the market. This allows us to observe what impact some of the parameters involved in the methodology have on the sales and consequently on the total revenue earned. We then incorporate some of the competitors described in Geer et al. (2018) in the market, run our algorithm against these competitors and examine its performance in two different settings: oligopoly and duopoly. In the oligopoly setting, all competitors compete simultaneously with each other while in the duopoly setting, each competitor competes with all the other competitors in a round-robin setup. More details about the demand mechanism we use to simulate the customers' reaction to the different prices are included in Section 5.2.1.

5.2.1 Demand mechanism

The demand mechanism we use is the same demand mechanism explained in Geer et al. (2018). The idea behind the demand mechanism is that there is a competitive market environment with heterogeneous customer base. Its main characteristics are a Poisson arrival process of the customers with mean arrivals per time period equal to λ_i in simulation i, where $\lambda_i \sim U(50, 150)$ and time independence of demand within a single simulation. At this point we highlight once again that none of the participants in this competition were aware of any of the assumptions regarding the market structure and they could not see the revenue earned by the other competitors in the market.

There are three customer segments: shoppers, loyal customers and scientists whose shares are denoted as ζ_i^{sho}, ζ_i^{loy}, ζ_i^{sci} respectively, in simulation i. The scientists segment consists of PhDs and professors and their respective shares are γ_i^{phd}, γ_i^{prof}. Additionally, the number of shoppers, loyals and scientists are denoted as $m^{sho}, m^{loy}, m^{sci}$ respectively. Thus, the total number of arriving customers is $m^{sho} + m^{loy} + m^{sci}$ and the number of arriving scientists is $m^{phd} + m^{prof}$. Below we describe the arrival process for each simulation in more detail.

Algorithm 8 Arrival process

```
for $i \in \{1, ...1000\}$ do
    Sample arrival rate $\lambda_i \sim U(50, 150)$
    Sample segment shares $\zeta_i^{sho}, \zeta_i^{loy}, \zeta_i^{sci}$
    Sample subsegment shares $\gamma_i^{phd}, \gamma_i^{prof}$
    for $t \in 1, ...1000$ do
        Sample arrivals $m \sim Poisson(\lambda_i)$
        Sample segment arrivals $m^{sho}, m^{loy}, m^{sci} \sim Multinom(m, (\zeta_i^{sho}, \zeta_i^{loy}, \zeta_i^{sci}))$
        Sample subsegment arrivals $m^{phd}, m^{prof} \sim Multinom(m^{sci}, (\gamma_i^{phd}, \gamma_i^{prof}))$
    end for
end for
```

Below we describe the three customer segments in more detail:

- Shoppers: The willingness to pay (WTP) is exponentially distributed with different parameters. In each period in simulation i, we sample a WTP for each arriving shopper from the exponential distribution with mean β_i^{sho} and compare these WTPs to the lowest price offered in the market. In an environment with more than one competitor in the market, if the WTP of a shopper is higher than the lowest price offered then the shopper buys from the competitor that offers the lowest price. Otherwise, the shopper leaves without buying. Ties are broken randomly.

- Loyal customers: Each loyal customer is randomly assigned to a competitor. For each loyal customer we assume their willingness to pay is exponentially distributed with mean $\beta_i^{loyal} = u\beta_i^{sho}$, where $U \sim U(1.5, 2.0)$. If the WTP of the loyal customer is higher than the price offered by the competitor assigned to him then he buys. Otherwise the loyal customer leaves without making a purchase.
- Scientists: Scientists are split into two categories: professors and PhDs. Their demand follows a finite mixture logit model or latent class logit model, where the mixture comprises professors and PhDs. We make sure that the optimal price for the PhDs is within 50% of the optimal price for the shoppers.Additionally, the optimal price in a market that consists of only professors would be higher than in a market that consists solely of PhDs.

In summary, in each simulation $i \in \{1, ...1000\}$, in each period we see in expectation $\lambda_i \zeta_i^{sho}$ arriving shoppers, $\lambda_i \zeta_i^{loy}$ arriving loyals, $\lambda_i \zeta_i^{sci} \gamma_i^{phd}$ arriving PhDs and $\lambda_i \zeta_i^{sci} \gamma_i^{prof}$ arriving professors. Each of these customers chooses according to its own parameterized demand function as described above.

In a monopoly, we make the necessary adaptations of the above demand mechanism in order to account for a single competitor in the market.

5.3 Experimental design

We run f simulations, each consisting of 1000 time steps. At every time step t, the competitors choose what price to charge at $t + 1$ and, based on the undisclosed demand mechanism, the sales of each competitor are generated. Thus, the competitors learn how much revenue they earn from the price they have charged.

In a monopoly setting we use the total revenue earned as a performance measure while in the case of competitive market environment we choose to use revenue share as a measure of the performance of each individual strategy against the other competitors in the market. The competitive environment consists of two settings: an oligopoly including all z competitors and a duopoly made up of $\binom{z}{2}$ duopoly competitions. In each simulation we calculate the competitors' share of total revenue in the oligopoly and their share of total revenue in the duopoly competitions and then calculate the average share in the two environments in order to determine the overall revenue share of each competitor. Below we describe how we calculate the revenue share of z competitors in f simulations with $u_{i,j}$ being the revenue earned by competitor j in oligopoly in simulation i of the oligopoly and v_{ijk} being the revenue earned by competitor j against competitor k in simulation i of the duopoly.

Oligopoly:

$$\overline{u}_{ij} = \frac{u_{ij}}{\sum_{k=1}^{z} u_{ik}}$$

Duopoly:

$$\overline{v}_{ij} = \frac{\sum_{k=1}^{z} v_{ijk}}{\sum_{u=1}^{z} \sum_{k=1}^{z} v_{iuk}}$$

Thus, the overall score of competitor j is :

$$\frac{1}{f} \sum_{i=1}^{f} \frac{1}{2}(\overline{u}_{ij} + \overline{v}_{ij})$$

5.4 Algorithm

Our pricing strategy combines standard Thompson Sampling with a racing machine learning algorithm, Hoeffding races (Maron and Moore, 1994). Due to the very limited information about the possible prices and the continuous nature of the prices we are allowed to charge in the interval $(0, 100]$, we choose to split the interval into Z different buckets. For illustrative purposes in the rest of the chapter we have $Z = 5$. Each of the price buckets pertains to an arm and choosing a specific arm means posting a randomly sampled price from the corresponding price bucket. Even though a significant amount of literature on MAB with an infinite number of arms exists (Agrawal, 1995; Auer, Ortner, and Szepesvári, 2007), we choose to use traditional MAB techniques with a finite number of arms. This makes the algorithm applicable in many situations in industry where the choice of candidate prices is much more restricted than the choice in the current problem (Ferreira, Simchi-Levi, and Wang, 2018).

When the process starts, we have limited information about the buckets and as we proceed we use information we have collected about the performance of prices in the different buckets in order to decide which bucket to select at the next time step. Note here that we use the same formulation we used in Chapter 3 for

Thompson Sampling (see Equations 3.4). The normality assumption in this algorithm stands only for the distribution used in the random sampling, standard part of Thompson Sampling and it does not reflect any assumption regarding the demand distribution. Hoeffding races is used at each time step to decide whether there is a price bucket that may be eliminated in order to suppress the price intervals we are allowed to choose a price from. When enough observations from a poorly performing bucket have been collected to be confident it does not include the best price we stop charging a price from this bucket. The bucket with the highest mean revenue so far is then split into two new buckets in order to maintain a constant number of buckets and shrink the price interval of the best bucket in order to concentrate the "learning" in the best performing prices.

Figure 5.1 illustrates the mechanism used for the restructuring of price buckets when a price bucket is eliminated. In Figure 5.1, we see that at $t = 14$, Hoeffding test decides to eliminate price bucket $B_{2,14}$. Thus, the mean revenue earned by all price buckets is calculated and the price bucket with the highest mean revenue earned so far is $B_{1,14}$. So this bucket is then split into two new price buckets that will constitute, along with the remaining ones, the candidate price buckets that will be considered at $t = 15$. New price buckets are created only when there is a bucket rejection following the implementation of Hoeffding races. When new price buckets are created, we revisit the prices charged so far and allocate each of them to one of the new price buckets, in order to recalculate the hyperparameters of the newly formed price buckets.

The idea of dividing the price interval into a number of equally spaced intervals that are shrinking was implemented by Wang, Deng, and Ye (2014) when solving a dynamic pricing problem with inventory constraints. However, to the best of our knowledge this is the first work that combines Thompson Sampling methodology with the racing machine learning method. Hoeffding races is explained in more detail in a general context in Section 5.4.1, in order to get a better understanding of how certain price buckets are eliminated. Algorithm 10 summarises the main steps of the pricing algorithm.

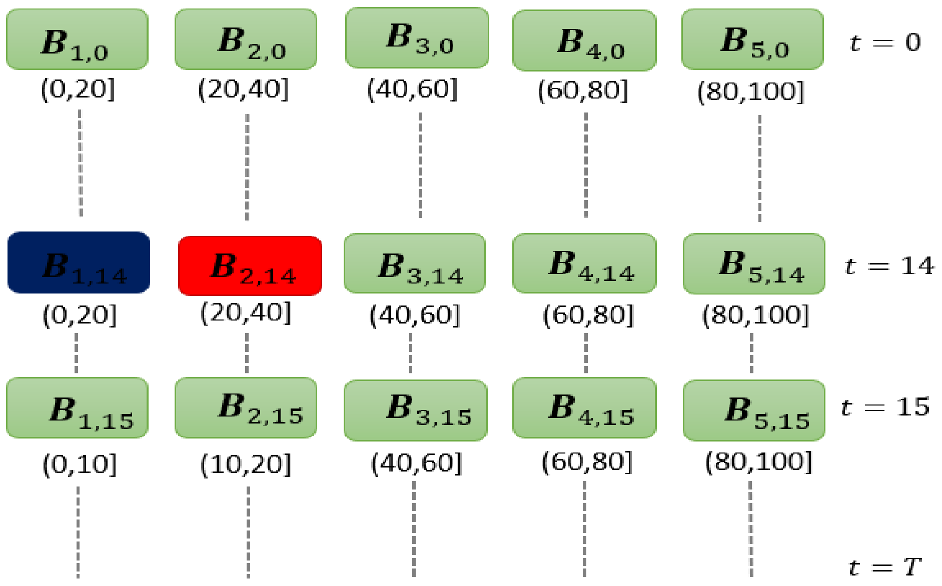


FIGURE 5.1: Illustration of the case where at $t = 14$ the worst bucket was $B_{2,14}$ and the best bucket was $B_{1,14}$

5.4.1 Hoeffding races

The idea of Hoeffding races (Maron and Moore, 1994) is the same as any other racing algorithm: discard bad systems in order to concentrate the computational effort on the differentiation between the better ones. In our problem each system corresponds to a different price bucket and we will be using the term price bucket for the description of this racing algorithm.

The algorithm uses the samples from all price buckets so far and calculates the error bounds. The estimated error bound for each price bucket is calculated using the following equation.

$$\epsilon(n) = \sqrt{\frac{R^2 \log(2/\delta)}{2n}}$$

where R bounds the greatest possible revenue that a price bucket can make, n is the number of points we have tested so far and δ is a confidence parameter. If E_{true} is the true error then the following condition follows from Hoeffding's inequality (Hoeffding, 1963).

$$P(\mid E_{true} - E_{est} \mid > \epsilon) < \delta, \delta = 2e^{\frac{-2n\epsilon^2}{R^2}}$$

When calculating the error bounds for the mean performance of the different buckets, the price bucket whose upper bound is lower than the lower bound of the best bucket is rejected. If this is the case for more than one price bucket we choose to reject the price bucket with the lowest mean revenue. As n gets larger, the ϵ bound becomes smaller and bad buckets are eliminated. The algorithm stops either when we are left with just one price bucket or when E_{est} has reached a certain threshold (Maron and Moore, 1994; Yeh and Gallagher, 2005). It is important to note that if the value of R is chosen to be too small, we end up discarding price buckets very quickly, but often the wrong ones, whereas if R is too big, it takes the algorithm a long time to identify price buckets that have a poor performance. Thus, a further experimentation on the R parameter is performed in the implementation of the proposed methodology. Algorithm 9 describes the steps of Hoeffding races applied at each time step of the main pricing algorithm which is described as Algorithm 10.

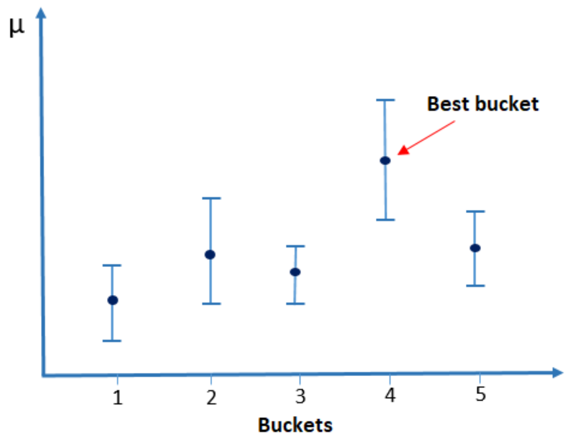


FIGURE 5.2: An example of case where there are 5 buckets in comparison. Bucket 1 is discarded because the upper bound of its mean performance is lower than the lower bound of the best bucket 4 and its mean is lower than the mean of Bucket 3

Algorithm 9 Hoeffding races

```
Define $\delta = 0.05$ and $R$. $P_\tau$ to be the set with the upper bound of each bucket at time $\tau$
for each arm $a \in P_\tau$ do
    Calculate error bound $\epsilon_a = \sqrt{\frac{R^2 \ln(2/\delta)}{2n}}$
    Calculate mean revenue $\overline{Rev}_a$
    Calculate upper and lower bound $LB_a = \overline{Rev}_a - \epsilon_a$ and $UB_a = \overline{Rev}_a + \epsilon_a$
end for
Find the best price bucket $a^* = \underset{a \in P_\tau}{\text{argmax}} \overline{Rev}_a$
for each arm $a \in P_\tau$ do
    if $UB_a < LB_{a^*}$ then
        Reject bucket $a_{rej} = a$
    end if
end for
```

Algorithm 10 Thompson Sampling with Hoeffding races (TS-HR)

```
Define $Z$ price buckets as the Z "arms" such that $B_{i,0} \forall i \in \{1, 2, .., Z\}$ represent the $Z$ price buckets
Define $P$ as the set with the upper bound for each bucket
Define initial prior hyperparameters $\mu_0, \kappa_0, \alpha_0, \beta_0$
for each $\tau \in \{1, 2, ..., T\}$ do
    for each arm $a \in P_\tau$ do
        Sample reward $\tilde{r}_{a,\tau}$ from the t-distribution $t_{2\alpha_n}(\mu; \mu_n, \frac{\beta_n}{\alpha_n \kappa_n})$
    end for
    Choose bucket $a_\tau = \underset{a_\tau \in P_\tau}{\text{argmax}} \tilde{r}_{a,\tau}$
    Charge a random price $p_\tau \in a_\tau$
    Get the demand $d(p_\tau)$ and calculate revenue $d(p_\tau) * p_\tau$
    Update the hyper parameters of price bucket $a_\tau$
    Use Hoeffding races to decide if there is any price bucket to eliminate and update $P_\tau$ and $B_{i,\tau} \forall i \in \{1, 2, .., Z\}$-Do Algorithm 9
    if a bucket is rejected then
        Update the hyperparameters of $a_{rej}$ and $a^*$
    end if
end for
```

5.5 Competitor algorithms

We choose three pricing strategies as competitors, to form the competitive market environment, as described in Geer et al. (2018): logit, WLS and greedy. Logit and WLS are chosen due to their superiority in oligopoly and duopoly respectively. Greedy is included in the competition in order to examine whether such a simple algorithm can win against other more complicated algorithms. Its overall performance in Geer et al. (2018) suggests that even though the methodology is simple, it still achieves a total revenue share that places it second out of seven competitors, with more complicated algorithms, in the original competition. Below is a brief description of these pricing strategies and their main features.

- **Logit:** The algorithm uses a finite mixture logit model where the mixture is taken over the number of possible customer arrivals. In the first 100 time periods it tries to estimate the maximum number of arriving customers in a single period. After that it uses an Expectation-Maximization algorithm to estimate a probability distribution over the number of arrivals and the parameters of the multinomial logit models. Regarding competitors' prices, the algorithm assumes that they follow a multivariate normal distribution and at each time step it predicts competitors' prices for the next time step.

- **WLS:** The algorithm aims to maximize its own revenue minus the revenue of the competitor that earns the maximum revenue. The algorithm assumes a demand function of the form $d(x, y) = a + bx + c\sum_{k=1}^{z-1} y_k$ and the parameters a, b, and c are estimated using weighted least squares (WLS). The price of competitor k in the next time step is predicted based on the median of the historical prices over some time window. This window's length is chosen so that it minimizes the Median Absolute Error of historical prices predictions.

- **Greedy:** This is a very simple strategy that does not try to estimate what price the competitors will charge in the next time period. The simple idea behind it is that it charges the minimum price observed in the previous time step. In order to maintain high enough prices the algorithm does the following check: if the minimum price observed in the last time step is lower than 10% percentile of the observed prices in the last 30 time steps then the price it chooses for the next time step is the maximum of this percentile and 5.

5.6 Results

5.6.1 Monopoly

Initially, we perform some experimentation on the parameters involved in the new methodology, denoted as TS-HR . We try different values for the initial estimate of the mean revenue in each bucket (μ_0) while keeping the rest of the hyperparameters equal to 1 and trying different values for R, which is the range parameter in Hoeffding races. Then, for the different values we report the total revenue earned along with the number of times there is a bucket elimination in the 1000 time steps of each individual simulation. The results presented are averaged over 1000 independent simulation runs.

Figure 5.3 shows the average total revenue over the 1000 runs. We observe that for values of R from 50 to 300 the total revenue is high with just small fluctuations for the different values while for values greater than 300 there is a continuous decrease in revenue with the least total revenue earned when $R = 500$. Additionally, the highest revenue for any value of R is earned when $\mu_0 = 1000$. In Figure 5.4 it is clear that for values of R from 50 to 300, the average number of bucket eliminations in a time period of 1000 time steps decreases, with a sharp decrease between the value of 50 and 100 and stays small and constant for any value of R greater than 300. This is in line with the results presented by Yeh and Gallagher (2005). They observe that an increase in the value of R results in an increase in the number of iterations needed before an option is discarded.

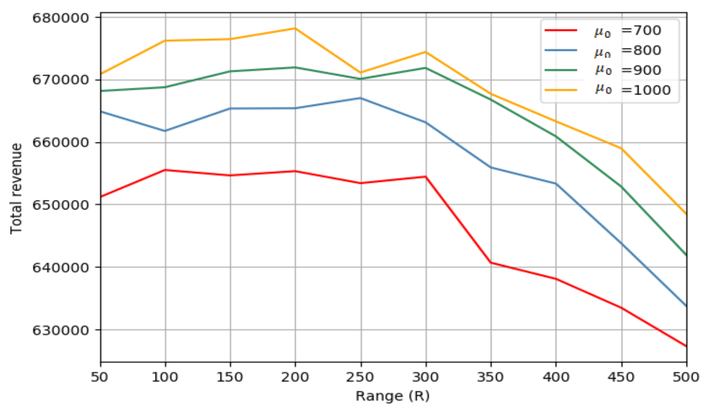


FIGURE 5.3: Average total revenue earned in 1000 time steps for 4 different values of μ_0 when $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$, for different values of Range (R). The results are averaged over 1000 independent simulation runs

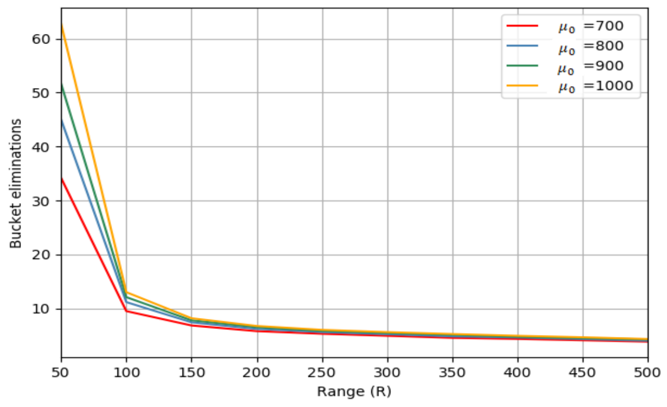


FIGURE 5.4: Average number of bucket eliminations in each run with 1000 time steps for 4 different values of μ_0 when $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$, for different values of Range (R). The results are averaged over 1000 independent simulation runs

In order to examine further the performance of the new proposed methodology we present a comparison between Thompson Sampling with 5 price buckets that remain the same throughout the experiment and Algorithm 10 that uses Hoeffding races in order to eliminate buckets that have not performed well in the past and shrink the price interval of the bucket that has been the best so far in the manner described in Figure 5.1. For consistency, we use the same values for the hyperparameters involved in the main steps of the Bayesian methodology, in both algorithms. In Figures 5.5 and 5.6 cumulative revenue and historical total demand of TS-HR are both higher at any time step. Even though the demand earned by standard Thompson Sampling increases over time, the new algorithm manages to earn high demand very early and keep it high for the rest of the experiment.

Additionally, the total revenue of standard Thompson Sampling averaged over the 1000 individual runs is 355,391.4 against TS-HR with $R = 100$ which is 653,231.4. Even when $R = 500$, the average total revenue of TS-HR is 627,311.2 which is still much higher than standard TS that does not use Hoeffding races at all.

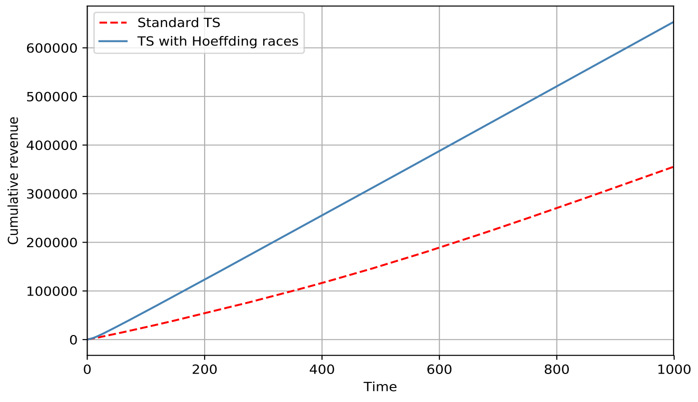


FIGURE 5.5: Comparison of cumulative revenue for standard Thompson Sampling and Thompson Sampling that uses Hoeffding races for bucket eliminations (TS-HR) with $R = 100$. In both algorithms $\mu_0 = 700$, $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$.

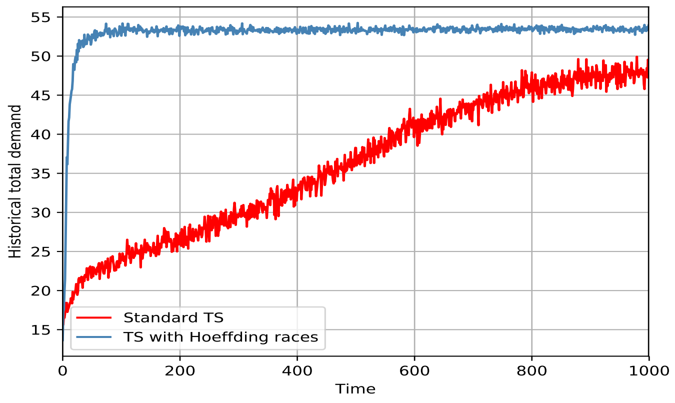
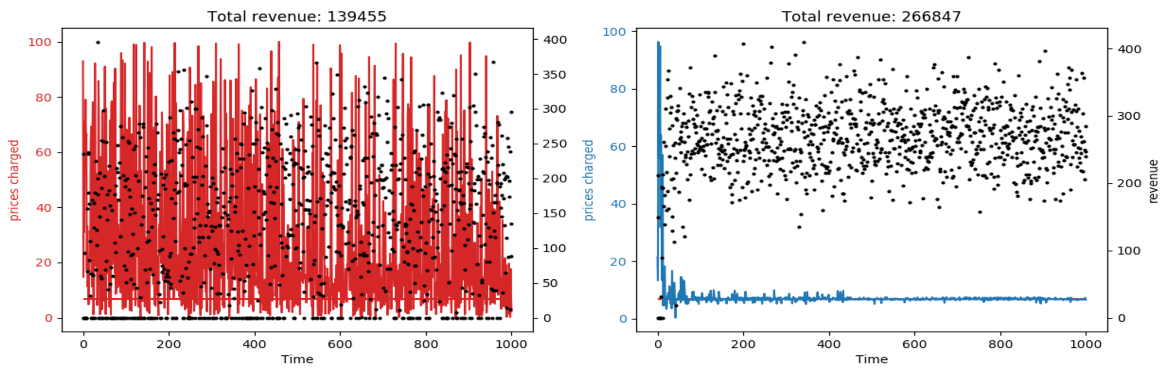


FIGURE 5.6: Comparison of demand for standard Thompson Sampling and Thompson Sampling that uses Hoeffding races for bucket eliminations (TS-HR) with $R = 100$. In both algorithms $\mu_0 = 700$, $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$.

Figure 5.7 shows the prices charged (line) and the revenue earned at each time step (dots) in a single individual run of standard Thompson Sampling (Figure 5.7a) and Algorithm 10 (Figure 5.7b). The red straight line shows the optimal price for each run as it is calculated using golden-section search. The golden-section search algorithm (Kiefer, 1953) is applied in order to understand whether in single runs, the algorithms manage to converge to what is estimated as the optimal price, in this particular run. For more details of the optimization method see Appendix B.



(A) Standard TS-run 900 (B) TS with Hoeffding races-run 900

FIGURE 5.7: Prices charged and revenue earned by standard TS with $\mu_0 = 700$, $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$ and Thompson Sampling that uses Hoeffding races for bucket eliminations with $R = 100$. The graphs are from simulation 900.

5.6.1.1 Discussion

Some initial analysis on the newly developed algorithm that combines the standard formulation of Thompson Sampling with Hoeffding races was performed in order to observe how the parameters involved in the methodology affect its performance. It is important to see how R affects the decision made about whether a bucket should be eliminated and whether the number of bucket eliminations has an impact on revenue. Our results show that from a certain value of R, as its value increases the total revenue decreases. Combining this with the results about the effect that the value of R has on the number of bucket eliminations it is clear that this reduction in total revenue is related to the reduction in the number of bucket eliminations that exist in each run. The values of R that result in a decrease in the number of bucket eliminations and consequently in the number of times the buckets get to shrink, reduce the total revenue over the time horizon. Thus, total revenue is inversely proportional to the total number of bucket eliminations.

A comparison between the algorithm that uses Hoeffding races and an algorithm that keeps the price intervals in the buckets the same throughout the experiment shows the clear advantage that the incorporation of Hoeffding races offers. When observing results from the individual runs we observe that the increased exploration of Thompson Sampling with no bucket eliminations is what causes big losses in both demand and total revenue.

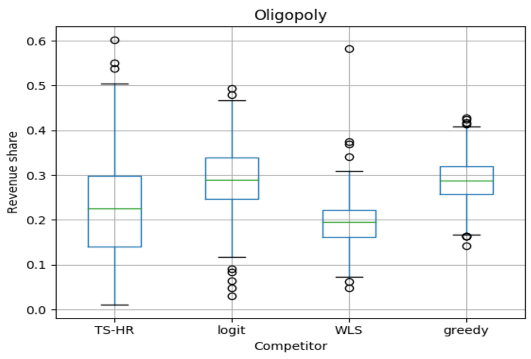
5.6.2 Competitive market environment

In this section we examine the performance of our algorithm in an environment with competition. The aim is to study the advantages and disadvantages of our algorithm against other pricing strategies in two different settings: oligopoly and duopoly.

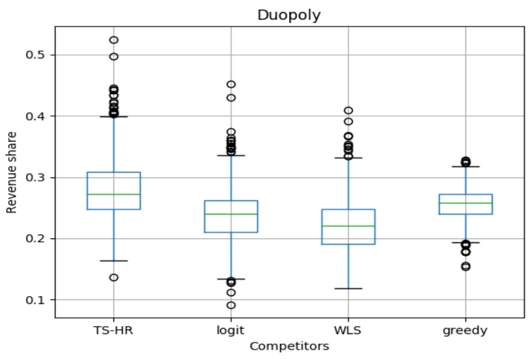
5.6.2.1 Oligopoly

Figure 5.8a shows the revenue share, calculated for 500 simulation runs, when in each simulation run the four strategies are competing all together. In these experiments the parameter values for our algorithm, TS-HR, are $\mu_0 = 800$, $\kappa_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$ and $R = 100$. It is clear that logit and greedy earn the highest revenue share with a median revenue share 0.2905 and 0.2891 respectively, making it hard to distinguish between the two. TS-HR comes third and WLS last. On the other hand, in Figure 5.8b we observe that in the duopoly competition TS-HR earns the highest revenue share against the other three competitor with greedy coming second, logit third and WLS last. The overall revenue share of the three competitors is presented in Figure 5.8c. Greedy comes first in the overall revenue share and logit follows with very small difference. Even though TS-HR comes first in the duopoly competition we observe that it still remains behind logit and greedy whose overall revenue share is dominated by the revenue share earned in the oligopoly stage. However, TS-HR's overall revenue share is very close to the first two competitors making it a good competitor compared to WLS who comes last in both oligopoly and duopoly.

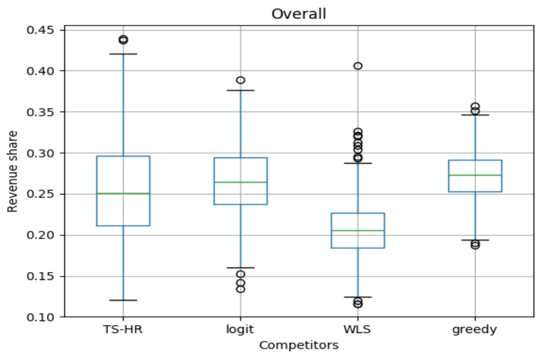
Some preliminary results from experimentation on the R and μ_0 parameters, when the algorithm is tested in oligopoly are included in Appendix C. The parameter values of TS-HR that we use throughout the chapter are the ones that optimize its performance in monopoly and oligopoly, based on the results presented in Section 5.6.1 and the indicative results in Appendix C respectively.



(A) Revenue share in oligopoly



(B) Revenue share in duopoly



(C) Revenue share in full competition

FIGURE 5.8: Revenue share for the four competitors in oligopoly, in duopoly and their overall revenue share. The results are the from 500 simulations.

In Figure 5.9 we show the prices each competitor charges. Out of a total of 500 simulation runs $\times$ 1000 time steps we take 50 simulation runs $\times$ 1000 time steps and observe that TS-HR charges significantly higher prices than the other three competitors that charge very low prices. Figure 5.10 shows the prices charged in 50 simulation runs $\times$ 500 time steps, where 500 refers to the last 500 time steps in a single simulation. This can give us an idea of whether the competitors converge to a certain price after some time of the simulation has passed. The absence of outliers for greedy and its small interquartile range indicate that from $t = 500$ to $t = 1000$ it converges to very low prices. On the contrary, TS-HR keeps charging a wide range of prices for the whole duration of the simulation.

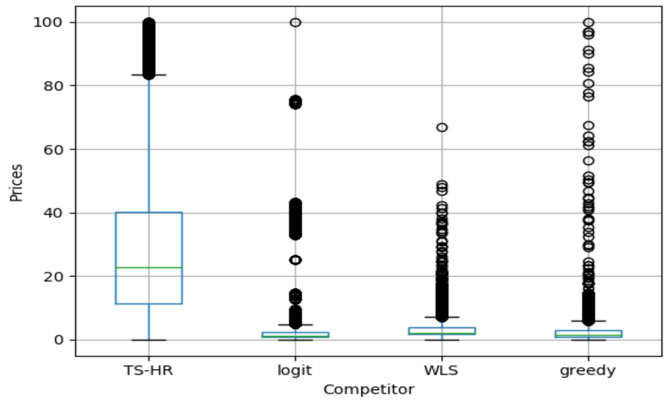


FIGURE 5.9: Boxplot based on the prices charged in 50 × 1000 randomly chosen time steps out of the total of 500(number of simulations) × 1000 (number of time steps) time steps.

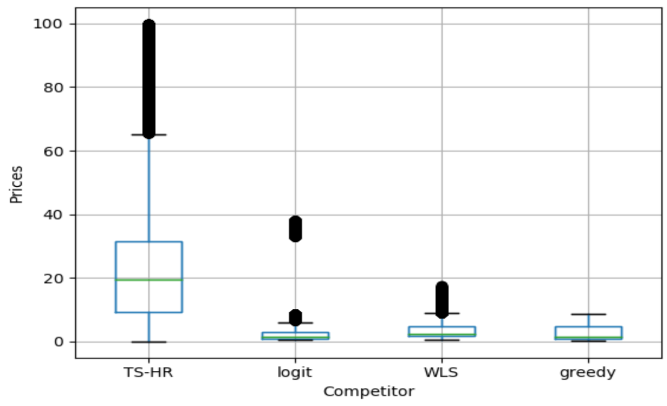


FIGURE 5.10: Boxplot based on the prices charged in 50 × 500 randomly chosen time steps out of the total of 500(number of simulations) × 1000 (number of time steps). 500 refers to the 500 last times steps in each simulation.

The mean demand for each time step for the four competitors split out for the three customers segments is shown in Figure 5.11. An interesting result is that WLS gets more total demand than TS-HR even though its revenue share in the oligopoly is the smallest, as illustrated in Figure 5.8a. This shows that low prices may attract more customers but this does not guarantee high revenue. Also, WLS does not attract almost any shoppers, which is in line with the results presented in Geer et al. (2018), where WLS does not get any shoppers even when there are more competitors in the competition.

Finally, in Figure 5.12 and Figure 5.13 we present the prices charged and the respective revenue for the four competitors in two different random simulations.

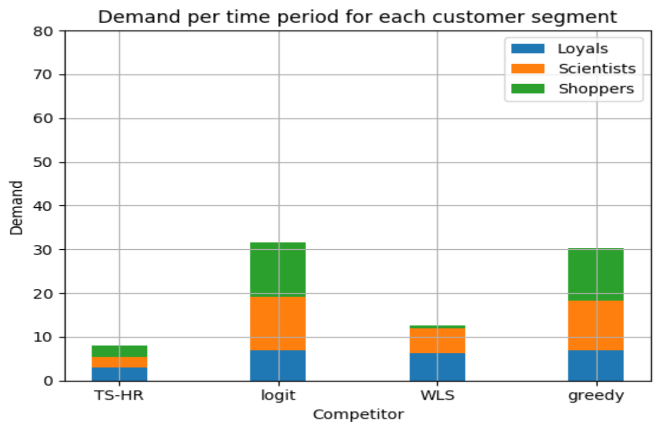
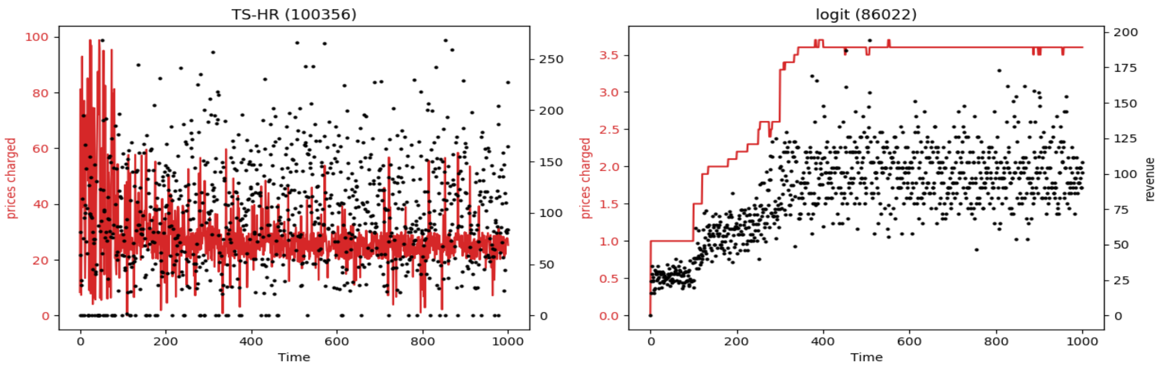
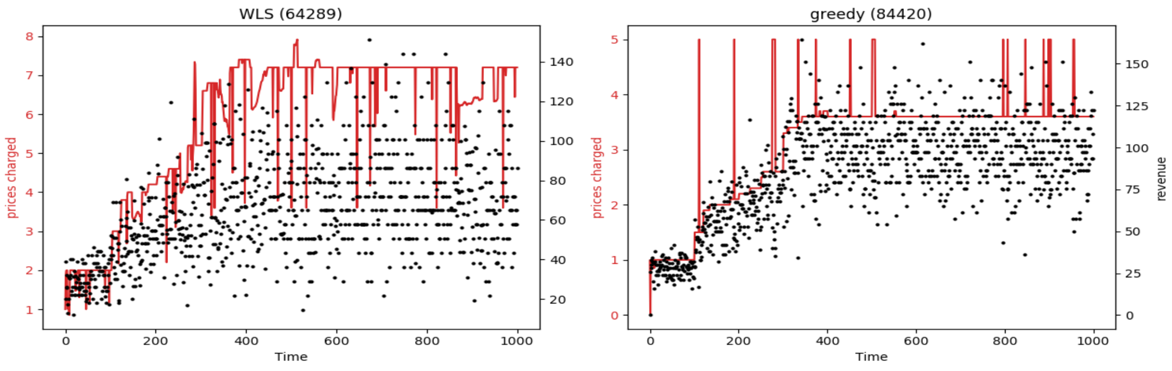


FIGURE 5.11: Mean demand for each time step split out over the three customer segments

The red line shows the prices charged and the black dots the revenue earned at each time step while the number in the brackets is the total revenue earned in this single simulation. These figures illustrate how the competitors behave in two different situations. In Figure 5.12a TS-HR converges to a small range of prices and stops charging prices from other buckets quite early. This limitation in the prices it charges leads to the highest total revenue against the other competitors in the same simulation. On the other side, in Figure 5.13a where TS-HR charges prices from all buckets, shown by the large fluctuations of the red line, until almost $t = 500$, it gets the lowest total revenue compared to the other competitors. The other three competitors behave similarly in both simulations and it is clear that they converge very early to very low prices.

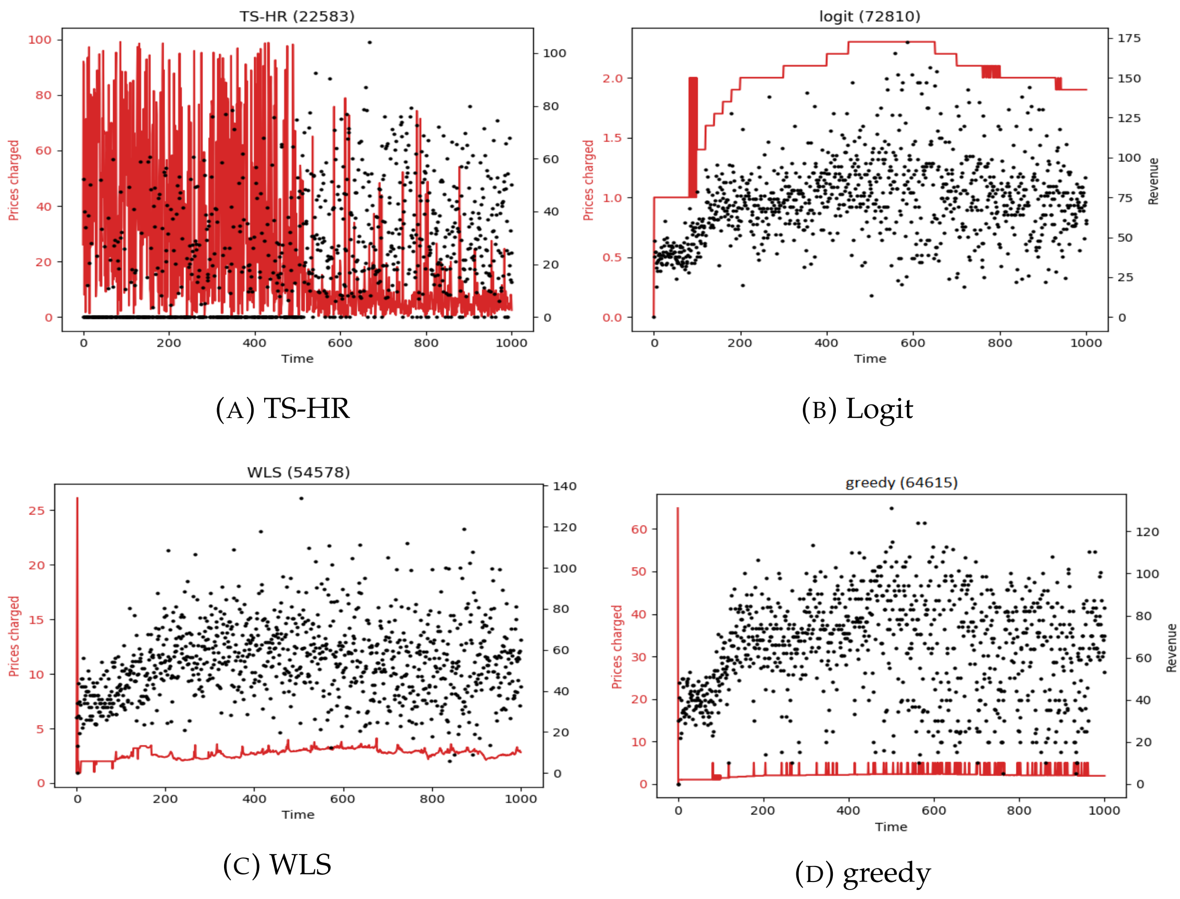


(A) TS-HR (B) Logit



(C) WLS (D) greedy

FIGURE 5.12: Revenue and prices charged for each competitor individually. These results are just for illustrative purposes from a random simulation, simulation 100. In the brackets it is the total revenue earned by the competitors in the same simulation. In this particular simulation TS-HR earns the highest total revenue.



(A) TS-HR (B) Logit (C) WLS (D) greedy

FIGURE 5.13: Revenue and prices charged for each competitor individually. These results are just for illustrative purposes from a random simulation, simulation 303. The number in brackets is the total revenue earned by the competitors in the same simulation. In this simulation logit earns the highest total revenue and TS-HR the lowest total revenue.

5.6.2.2 Duopoly

In Table 5.1, each row shows the mean revenue, at each time step, over 500 simulation runs, earned by the competitors, whose names are listed on the first column of the table, against the competitors in the other columns in a round-robin setup. For example, TS-HR gets a mean revenue 245.44, at each time step, against logit while logit gets a revenue of 255.44 against TS-HR. We underline the competitor that gets the highest mean revenue in each duopoly competition. The last column shows the average mean revenue each competitor gets in all duopoly competitions and the final row shows the average mean revenue the other competitors get when run in a duopoly with the competitor listed at the top of the column.

Even though WLS wins in all duopoly competitions, its average mean revenue in the pairwise competitions is smaller than the average mean revenue earned by

TS-HR and greedy from all duopoly competitions. Also, despite the fact that TS-HR wins only in the duopoly competition against greedy, it manages to get the highest average mean revenue.

Mean revenue per time step					
	TS-HR	logit	WLS	greedy	Average
TS-HR		245.44	186.83	266.56	232.94
logit	255.44		96.17	261.97	164.28
WLS	260.83	152.70		136.11	183.24
greedy	261.39	257.58	122.44		213.80
Average	259.22	218.57	135.15	221.55	

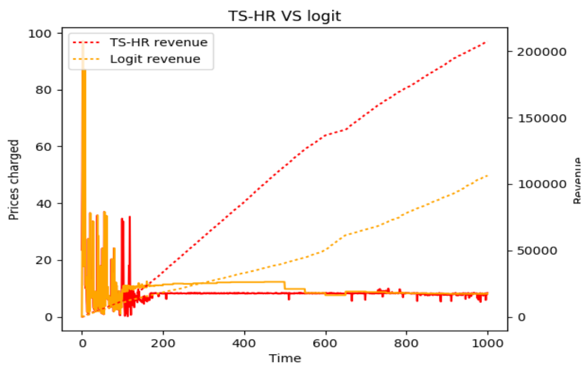
TABLE 5.1: Mean revenue at each time step for each competitor against each of the other competitors in all duopolies. Each row shows the mean revenue of the competitor in the first column against each of the other competitors in duopoly.

Some further analysis of the prices charged by the competitors in each duopoly competition, presented in Table 5.2, reveals the effect that prices charged have on the revenue earned in duopoly. In Table 5.2 each row represents the mean price the competitor in the first column charges against each competitor in the next four columns, for each time step. For example, TS-HR charges a mean price of 10.08 in the duopoly against logit, while the mean price logit charges in the duopoly against TS-HR is 10.04. Additionally, the average mean price TS-HR charges against all competitors in the duopoly stage is 9.81 while the average mean price all competitors charge against TS-HR is 9.53. It is clear that WLS forces the prices to stay quite low in all duopoly competitions (around 5) and this is what causes the small average mean revenue shown in Table 5.1 and consequently make its revenue share in duopoly the lowest, as shown in Figure 5.8b

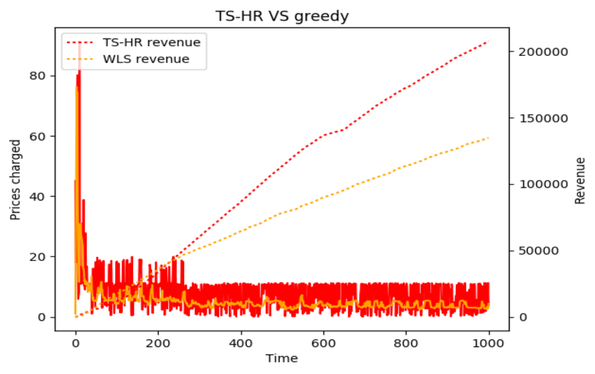
Mean price charged per time step					
	TS-HR	logit	WLS	greedy	Average
TS-HR		10.08	9.38	9.98	9.81
logit	10.04		4.56	10.91	8.50
WLS	8.39	3.53		4.15	5.36
greedy	10.16	10.96	4.57		8.56
Average	9.53	8.19	6.17	8.34	

TABLE 5.2: Mean price charged at each time step for each competitor against each of the other competitors in all duopoly competitions. Each row shows the mean price of the competitor in the first column against each of the other competitors in duopoly.

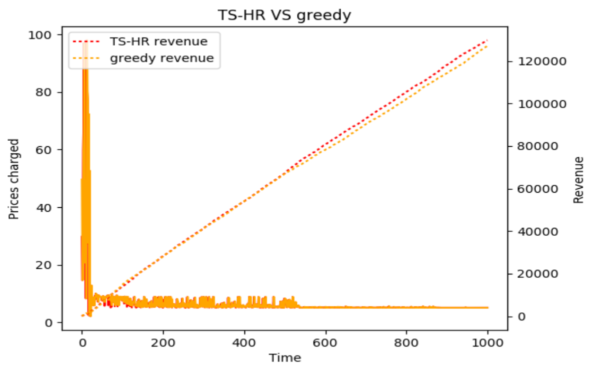
Finally, in Figure 5.14 we present the prices charged (line) and the cumulative revenue (dotted line) earned by TS-HR in all duopoly competitions for a random simulation, simulation 369. We can see that in this single simulation TS-HR wins against logit and WLS and get almost the same revenue as greedy.



(A) TS-HR VS logit



(B) TS-HR VS WLS



(C) TS-HR VS greedy

FIGURE 5.14: Revenue and prices charged by TS-HR in three different duopolies with the three competitors. These results are just for illustrative purposes from a random simulation, simulation 369.

5.6.2.3 Discussion

In the oligopoly setting, TS-HR obtains a lower revenue share than both logit and greedy, which appears to be due to the fact that it does not converge and continues to charge high prices. When observing the prices charged, we see that TS-HR charges much higher prices than its competitors during the whole selling period and does not converge. This suggests that the method of Hoeffding races does not contribute significantly to the convergence towards the most profitable prices. Some additional experimentation shows that varying the range parameter in the Hoeffding races section of the algorithm does not result in any significant increase in the revenue share in oligopoly (See Appendix C).

Individual simulations confirm that converging to either a single price or a small range of prices, contributes to a high revenue share. Since TS-HR does not take into account what competitors charge, it appears to carry out too much exploration in some of the simulations. In these simulations the algorithm loses a lot of customers due to the high prices it charges, compared to the other competitors in the market.

The results are different in the duopoly competition when there is only one competitor against TS-HR. In this case, TS-HR has more limited exploration, possibly due to the variability in the revenue obtained for different price buckets being lower for a duopoly than for an oligopoly. For all competitors the average mean prices they charge against each other are very close, indicating that in a duopoly both algorithms tend to follow each other's prices in order to be competitive in the market. Regardless of the algorithm TS-HR is competing with, it forces the prices charged to be relatively high and this contributes to its high revenue share in duopoly. Finally, if an algorithm beats a large number of competitors in the market in the duopoly stage, it does not imply that it will get a high total revenue share from all duopoly competitions. This observation is mentioned by Geer et al. (2018) as well and it reveals the complexity involved in evaluating learning algorithms.

5.7 Concluding remarks

To conclude, in this chapter we considered dynamic pricing, a particular application of learning versus earning methodologies, and how Thompson Sampling can be used for the development of a pricing strategy. Motivated by algorithms described in Geer et al. (2018), we used the concept of price buckets as a method of discretisation of a continuous range of prices that can be chosen, in order to apply the MAB technique with a finite number of "arms". Developing the idea further, we added a racing algorithm, Hoeffding races, in order to help Thompson Sampling eliminate bad performing prices (i.e. prices that return low revenues) and concentrate the sampling on the best-performing region of the price-space. Our experiments incorporated a monopoly, an oligopoly and a duopoly and we observed that a good performance in oligopoly does not guarantee good results in duopoly, as the winner in the two settings was different.

A pricing policy based on Thompson Sampling and price buckets benefits from the use of Hoeffding races for reducing the range of prices to experiment over. A possible modification we suggest as future work is instead of checking which price bucket should be eliminated, it may be better to have an algorithm that keeps all of the price buckets available but splits the best performing bucket into two new smaller price buckets. We understand though that this implies a different number of price buckets and consequently extra care must be taken when calculating the hyperparameters of the two buckets.

It is important to note that the results presented along with some additional tests, illustrate that the competitors in the market have a significant impact on the revenue earned by each of the competitors. Some tests we performed using different combinations of competitors in the market showed that there is not a single competitor that performs well in all situations. This, however, depends a lot on the demand mechanism used. We want to emphasize that all the tests were performed using the specific underlying demand mechanism introduced in Geer et al. (2018) and consequently we cannot guarantee the same results under a different demand mechanism. A further modification of the demand mechanism used in the simulations could involve costs, in order to avoid a race to the bottom, that has a big impact on the results, in this example.

Contextual bandits that use competitors prices as context were used in order to increase the revenue share of TS-HR but the results did not improve. In some preliminary tests we used a modification of the formulation of Algorithm 6, presented in Chapter 4 to check whether there are any improvements when competitors' prices are incorporated as context. For our experiments the context at t was the minimum price charged by all competitors at $t-1$ (See Appendix D for more details). Surprisingly, we did not observe any significant difference in the behaviour of the algorithm. We believe that a reason that the contextual algorithm did not work is because in these experiments all competitors were converging to a certain price and consequently using competitors prices as context did not offer more significant information to the algorithm. However, against different competitors, and under a different demand mechanism we believe that our pricing strategy will benefit from the incorporation of competitors' prices as context.

Chapter 6

Conclusions and future work

This chapter reviews the main contributions of this thesis and indicates directions for future research.

We presented several practical contributions to earning while learning methods, mainly based on the Bayesian exploration algorithm, Thompson Sampling. As motivated in Chapter 1, our goal was three-fold: the development of a new algorithm that finds the best option quickly in a stationary environment and the study of its performance on different scenarios; the development of a new algorithm that accounts for seasonality in the rewards of the different candidate options; and the study of dynamic pricing in the framework of earning while learning, through the development of a pricing strategy. To achieve the first two objectives the algorithms presented were tested on different examples arising in the problem of website optimisation and explained in this context.

6.1 Contributions

The contributions regarding learning algorithms in the stationary case, as presented in Chapter 1, include the development of the learning and deployment buckets algorithm and its implementation on a large number of artificial datasets. Benchmarking tests show that our algorithm benefits from the splitting of traffic into two buckets in the case where there is high variability in the rewards of the different options. However, our results show that continuously showing the same option to some proportion of traffic, a widely used approach in website optimisation, does not offer significant improvements when variability is moderate to low. Standard Thompson Sampling is superior in these situations.

To the best of our knowledge, there are no other examples in the literature that use benchmarking tests on Thompson Sampling. These tests helped us identify the impact that the prior hyperparameters have on the performance of the different algorithms and at the same time they were a key tool for the comparison of the different algorithms. Additionally, deploying the newly developed algorithm, Thompson Sampling and ε- greedy on a large number of artificial datasets gave us the opportunity to study their dependence on particular features of the dataset. The results obtained indicate that the variability in the rewards of the different options have a big impact on the performance of all algorithms and should be considered when examining the performance of any MAB algorithm.

The contributions to earning while learning algorithms for cases where there is some seasonality in the rewards of the different options were presented in Chapter 4. Namely, we presented a new algorithm that combines Thompson Sampling with a statistical model selection method, Akaike Information Criterion, which we compared against standard contextual Thompson Sampling under different seasonality assumptions. The methods were tested on a number of artificial datasets with different variability in the rewards of the different options and different period (daily, monthly etc) and level (low, high) of seasonality. The results show that our algorithm performs well in many cases, mostly when the variability of rewards is either low or high. Additionally, the results suggest that good performance does not require the selection of the correct model for seasonality. This is in line with some recent work suggesting that wrong models can lead to good and in some cases better decisions (Besbes and Zeevi, 2015).

Chapter 5 is devoted to the contribution on dynamic pricing. We developed a dynamic pricing strategy, TS-HR, that combines Thompson Sampling with Hoeffding races, to decide what price to charge from the range $(0, 100]$ for a single product. In order to formulate the problem in a MAB context we decided to split the price interval $(0, 100]$ into five different price buckets so that each of the price buckets pertains to an arm and choosing a specific arm means posting a randomly sampled price from the corresponding price bucket. Empirical results of TS-HR in a monopoly setting show that a pricing algorithm that discretises a big price range in this manner can benefit from further discretisation of the best performing bucket. In our pricing strategy, Thompson Sampling with a discrete number of price buckets is combined with Hoeffding races which contributes to the elimination of poorly performing buckets that fail a statistical test. Our algorithm is then tested in a competitive market environment, in oligopoly and in duopoly settings using the demand mechanism used by the organisers of the dynamic pricing challenge held on the occasion of INFORMS Revenue Management and Pricing Section Conference in 2017. When TS-HR competes in a competitive market environment against other pricing strategies from the literature in an oligopoly setting we observe the phenomenon of race to the bottom. Our algorithm is not able to converge to a single price while other strategies converge to very low prices very quickly. Even though this results in a low revenue share in oligopoly, in duopoly it contributes towards the highest revenue share against the other competitors in the market.

6.1.1 Contributions to practice

This thesis provides experimental results on the performance of MAB algorithms in common industrial problems where results are collected in finite time. This contrasts with theoretical studies which generally aim to develop proofs of convergence. The results provide insights to practitioners whose main objective is to find the best option amongst a range of candidate options. The first example we consider from website optimization shows that Thompson Sampling works well in this case and can be used to find the best version of a website. Our modification to Thompson Sampling, in which a small proportion of the traffic is always sent to the current best-performing option and the remainder is offered an option suggested by the Thompson Sampling algorithm performs well in situations where the variability in the rewards is high. Based on the benchmarking tests that we carried out our recommendation to practitioners is to always take into consideration the known characteristics of the options they compare in order to decide what method to use and decide on the values of the prior hyperparameters since these can represent the current knowledge about their performance. The same is also true for the case of seasonality in the rewards of the different options. Additionally our method, AIC-TS, can constitute a good algorithm for the case where there is unknown seasonality in their reward distribution.

An important area in both retail and revenue management applications is dynamics pricing. In this area, our algorithm determines a good dynamic pricing strategy when the seller has limited initial information about the performance of the different candidate prices. In particular, we showed that when there is a big price range that the retailer is allowed to charge, our method of discretizing the state space can be very effective in terms of revenue earned. Additionally, our tests in a competitive market environment demonstrated the complexity of the dynamic pricing problem and gave some insights on the factors that play an important role in this setting.

6.2 Future work

One direction for future work in the stationary environment could include the development of a new algorithm that builds on a different MAB algorithm, instead of Thompson Sampling. Kuleshov and Precup (2014) suggest that UCB methods perform well in cases with high variance in rewards, indicating that UCB could be a good base to build a new algorithm on. Hence, incorporating a new algorithm that builds on UCB into the benchmarking exercise would determine how it compares with the other algorithms we have tested, especially in the situation where there is high variability in the rewards of the different options. Additionally, in the learning and deployment buckets algorithm, it would be interesting to study whether replacing Thompson Sampling with IZ in the learning bucket, could improve its overall performance. As explained in Chapter 2, IZ is a ranking and selection algorithm that also builds on the idea of screening out options that have performed badly in the past.

An extension of our work in the seasonal environment could include further study of contextual Thompson Sampling on datasets with high variability in the rewards of the different options in order to observe what causes the percentage of optimal selection to fall short of 100%. From some initial experimentation, we observed that when there is high variability, the algorithm explores too much but it may be the case that the algorithm keeps choosing a non-optimal option. Hence, a further study on the effect that variance has on the percentage of optimal selection of contextual Thompson Sampling could lead to some useful insights on its performance under these circumstances. Additionally, it will be interesting to study what happens in an environment where there is both seasonality and trend. In this case restless bandits will be of great importance, combined again with contextual bandits.

The work presented in Chapter 5 suggests several future work directions. Firstly, the phenomenon of race to the bottom in oligopoly represents a good starting point for developing a new simulation for the demand mechanism. In a real-life situation where the objective is the maximization of profits, a strategy that charges very low prices, during the whole selling period, is unlikely to be optimal. Thus, possible modifications of the current simulation used for the demand mechanism can be the incorporation of costs, limited capacity and perhaps the addition of different customer segments whose decision whether to buy the product is guided by additional criteria. The study of MAB methods with varying number of arms is also an appealing subject for further research. This will allow us to keep shrinking the price intervals of the best performing buckets without eliminating any buckets, leaving the MAB algorithm to instead reduce the probability of selection more gradually. Finally, work presented in Chapter 5 demonstrates the complexity of dynamic pricing and the significant impact that other competitors in the market have on a pricing strategy. Thus, we suggest that a modification of TS-HR that takes into account competitors' prices in the framework of contextual bandits might improve its performance in an oligopoly.

Appendix A

Nemeneyi test for cumulative regret at $t = 100$ for Scenario1

p-value for Nemeneyi test: 1.976284584980237E-4

i	algorithms	$z = (R_0 - R_i)/SE$	p
253	**ε-greedy($\varepsilon = 0.5$) vs. Thompson Samp.($\mu_0 = 25$)**	**18.286714112805473**	**1.0559130610224399E-74**
252	**ε-greedy($\varepsilon = 0.1$) vs. Thompson Samp.($\mu_0 = 25$)**	**17.84883384328562**	**2.9513498164864983E-71**
251	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	17.39010213236005	9.805935596829731E-68
250	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	16.952221862840197	1.853160707073979E-64
249	ε-greedy($\varepsilon = 0.5$) vs. Thompson Samp.($\mu_0 = 40$)	16.107738485909046	2.2510729004371506E-58
248	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)**	**16.086887044503317**	**3.152969312877084E-58**
247	**ε-greedy($\varepsilon = 0.3$) vs. Thompson Samp.($\mu_0 = 25$)**	**15.742838261309139**	**7.691494043821277E-56**
246	ε-greedy($\varepsilon = 0.1$) vs. Thompson Samp.($\mu_0 = 40$)	15.669858216389196	2.4312111909384934E-55
245	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 25$,$p = 20\%$)	15.669858216389185	2.43121119093891E-55
244	**ε-greedy($\varepsilon = 0.4$) vs. Thompson Samp.($\mu_0 = 25$)**	**15.503046685143524**	**3.308254019269771E-54**
243	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 25$,$p = 20\%$)	15.231977946869334	2.1692809418027856E-52
242	Buckets($\mu_0 = 25$,$p = 10\%$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	15.190275064057893	4.1019479856438465E-52
241	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 10$,$p = 40\%$)**	**15.075592136326506**	**2.3440486665249816E-51**
240	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	14.846226280863714	7.359658628102465E-50
239	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	14.6064347046981	2.5554544390211622E-48
238	Buckets($\mu_0 = 10$,$p = 40\%$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	14.178980155881082	1.2363017239854643E-45
237	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	13.90791141760689	5.67114047112845E-44
236	**ε-greedy($\varepsilon = 0.2$) vs. Thompson Samp.($\mu_0 = 25$)**	**13.907911417606883**	**5.671140471129016E-44**
235	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)**	**13.855782814092633**	**1.1737028951435159E-43**
234	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 25$,$p = 30\%$)	13.772377048469806	3.737113801217601E-43
233	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)**	**13.772377048469783**	**3.737113801218829E-43**
232	ε-greedy($\varepsilon = 0.3$) vs. Thompson Samp.($\mu_0 = 40$)	13.563862634412711	6.558915072019568E-42
231	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 40$,$p = 50\%$)**	**13.480456868789885**	**2.0384069660859712E-41**
230	Buckets($\mu_0 = 25$,$p = 20\%$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	13.47003114808703	2.347669496651625E-41
229	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 25$,$p = 30\%$)	13.334496778949955	1.4583544605146428E-40
228	ε-greedy($\varepsilon = 0.4$) vs. Thompson Samp.($\mu_0 = 40$)	13.324071058247096	1.6770724925889579E-40
227	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 25$,$p = 20\%$)	13.12598236489285	2.3373197868840922E-39
226	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	13.011299437161458	1.0553214293208362E-38
225	Buckets($\mu_0 = 25$,$p = 10\%$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)	12.95917083364721	2.0848893008185924E-38

224	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 10, p = 40\%$)	12.896616509430078	4.702968417925444E-38
223	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 25, p = 20\%$)	12.886190788727236	5.383804840602658E-38
222	Buckets($\mu_0 = 25, p = 10\%$) vs. Buckets($\mu_0 = 40, p = 20\%$)	12.87576506802436	6.162538187277599E-38
221	Buckets($\mu_0 = 25, p = 10\%$) vs. Buckets($\mu_0 = 40, p = 50\%$)	12.583844888344462	2.591051540879812E-36
220	Buckets($\mu_0 = 10, p = 40\%$) vs. Buckets($\mu_0 = 25, p = 20\%$)	12.458736239910218	1.2534109757641041E-35
219	**Thompson Samp.($\mu_0 = 25$) vs. buckets($\mu_0 = 10$, $p = 20\%$)**	**11.79149011492758**	**4.3183211368847794E-32**
218	ε-greedy($\varepsilon = 0.2$) vs. Thompson Samp.($\mu_0 = 40$)	11.728935790710455	9.05904545623011E-32
217	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 40, p = 30\%$)	11.676807187196204	1.674684130134816E-31
216	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 40, p = 20\%$)	11.593401421573354	4.450986001045039E-31
215	Buckets($\mu_0 = 25, p = 30\%$) vs. Buckets($\mu_0 = 40, p = 40\%$)	11.57254998016765	5.676992924079856E-31
214	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 25, p = 40\%$)	11.499569935247685	1.3257433811801057E-30
213	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 25, p = 50\%$)	11.32233268329918	1.0172841716721872E-29
212	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 40, p = 50\%$)	11.301481241893455	1.2902314711974518E-29
211	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 25, p = 20\%$)	11.291055521190595	1.4528140244261846E-29
210	Buckets($\mu_0 = 25, p = 20\%$) vs. Buckets($\mu_0 = 40, p = 30\%$)	11.238926917676345	2.6255435051018E-29
209	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 25, p = 30\%$)	11.228501196973472	2.954476288529082E-29
208	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 10, p = 10\%$)**	**11.207649755567784**	**3.7399220002007805E-29**
207	Buckets($\mu_0 = 25, p = 20\%$) vs. Buckets($\mu_0 = 40, p = 20\%$)	11.155521152053495	6.72973283653746E-29
206	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 25, p = 40\%$)	11.061689665727833	1.9243772948781077E-28
205	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 25, p = 30\%$)	10.988709620807857	4.330732601543832E-28
204	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 25, p = 10\%$)	10.894878134482155	1.2192968657632054E-27
203	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 25, p = 50\%$)	10.88445241377933	1.3671781240241934E-27
202	Buckets($\mu_0 = 25, p = 20\%$) vs. Buckets($\mu_0 = 40, p = 50\%$)	10.863600972373597	1.718366941134078E-27
201	Buckets($\mu_0 = 10, p = 40\%$) vs. Buckets($\mu_0 = 25, p = 30\%$)	10.56125507199084	4.505963728419582E-26
200	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 40, p = 10\%$)	10.311037775122374	6.2819269707091065E-25
199	Buckets($\mu_0 = 10, p = 10\%$) vs. Buckets($\mu_0 = 25, p = 10\%$)	10.311037775122362	6.2819269707099165E-25
198	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 40, p = 10\%$)	9.873157505602522	5.442391548757642E-23
197	**Thompson Samp.($\mu_0 = 10$) vs. Thompson Samp.($\mu_0 = 25$)**	**9.831454622791075**	**8.241957933542039E-23**
196	Thompson Samp.($\mu_0 = 40$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	9.612514488031152	7.079853949666559E-22
195	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 10, p = 30\%$)**	**9.58123732592259**	**9.588919490552445E-22**
194	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 25, p = 30\%$)	9.393574353271216	5.799790399826588E-21
193	Buckets($\mu_0 = 25, p = 30\%$) vs. Buckets($\mu_0 = 40, p = 30\%$)	9.341445749756964	9.502754290456807E-21
192	Buckets($\mu_0 = 25, p = 40\%$) vs. Buckets($\mu_0 = 40, p = 40\%$)	9.299742866945529	1.407857417187936E-20
191	Buckets($\mu_0 = 25, p = 30\%$) vs. Buckets($\mu_0 = 40, p = 20\%$)	9.258039984134115	2.082191797762626E-20
190	**Thompson Samp.($\mu_0 = 25$) vs. buckets($\mu_0 = 10$, $p = 50\%$)**	**9.247614263431263**	**2.2955910203595312E-20**
189	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 25, p = 20\%$)	9.174634218511292	4.5311013891657136E-20
188	Buckets($\mu_0 = 25, p = 50\%$) vs. Buckets($\mu_0 = 40, p = 40\%$)	9.122505614997024	7.34071730452348E-20
187	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 10, p = 50\%$)	9.03909984937421	1.579672150996389E-19
186	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 10, p = 10\%$)	9.028674128671355	1.7376428076609652E-19
185	Buckets($\mu_0 = 25, p = 30\%$) vs. Buckets($\mu_0 = 40, p = 50\%$)	8.966119804454216	3.071438635265404E-19
184	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 25, p = 40\%$)	8.95569408375135	3.376049097147247E-19
183	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 25, p = 10\%$)	8.93484264234565	4.077581075387171E-19
182	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 25, p = 50\%$)	8.778456831802846	1.65732871990391E-18
181	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 25, p = 40\%$)	8.715902507585735	2.8845146165383647E-18
180	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 10, p = 30\%$)	8.705476786882883	3.1624248706035613E-18

179	Buckets($\mu_0 = 10,p = 30\%$) vs. Buckets($\mu_0 = 25,p = 10\%$)	8.684625345477166	3.799927241414275E-18
178	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 10,p = 50\%$)	8.60121957985436	7.88734147563797E-18
177	Buckets($\mu_0 = 10,p = 10\%$) vs. Buckets($\mu_0 = 25,p = 20\%$)	8.590793859151496	8.63704704152427E-18
176	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 25,p = 50\%$)	8.53866525563723	1.3578109966101285E-17
175	ε-greedy($\varepsilon = 0.5$) vs. Thompson Samp.($\mu_0 = 10$)	8.455259490014399	2.784673740314272E-17
174	Buckets($\mu_0 = 10,p = 50\%$) vs. Buckets($\mu_0 = 25,p = 10\%$)	8.351002282985839	6.768611216058646E-17
173	Buckets($\mu_0 = 10,p = 40\%$) vs. Buckets($\mu_0 = 25,p = 40\%$)	8.288447958768717	1.1473574269365524E-16
172	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 10,p = 30\%$)	8.267596517363032	1.366865735239047E-16
171	Buckets($\mu_0 = 40,p = 40\%$) vs. Buckets($\mu_0 = 40,p = 10\%$)	8.111210706820216	5.011785994340329E-16
170	Buckets($\mu_0 = 10,p = 40\%$) vs. Buckets($\mu_0 = 25,p = 50\%$)	8.111210706820213	5.011785994340474E-16
169	ε-greedy($\varepsilon = 0.1$) vs. Thompson Samp.($\mu_0 = 10$)	8.017379220494547	1.0802521666742243E-15
168	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 40,p = 10\%$)**	**7.9756763376831**	**1.5154861029881357E-15**
167	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 40,p = 10\%$)	7.767161923626038	8.02642092596593E-15
166	Thompson Samp.($\mu_0 = 10$) vs. Thompson Samp.($\mu_0 = 40$)	7.652478995894646	1.9714131576164583E-14
165	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 40,p = 10\%$)	7.527370347460423	5.1772372619834606E-14
164	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 10,p = 30\%$)	7.402261699026162	1.3388431393765108E-13
163	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 25,p = 30\%$)	7.277153050591912	3.409383908180852E-13
162	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 25,p = 20\%$)	7.214598726374787	5.409324982950799E-13
161	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 25,p = 40\%$)	7.120767240049093	1.0732791164702322E-12
160	Buckets($\mu_0 = 10,p = 40\%$) vs. Buckets($\mu_0 = 40,p = 10\%$)	7.099915798643405	1.2483292077176807E-12
159	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 10,p = 10\%$)	7.079064357237689	1.4513097828808012E-12
158	Buckets($\mu_0 = 25,p = 10\%$) vs. Buckets($\mu_0 = 40,p = 10\%$)	7.079064357237677	1.4513097828809325E-12
157	Buckets($\mu_0 = 25,p = 40\%$) vs. Buckets($\mu_0 = 40,p = 30\%$)	7.068638636534844	1.564609915248057E-12
156	Thompson Samp.($\mu_0 = 40$) vs. buckets($\mu_0 = 10$, $p = 50\%$)	7.068638636534835	1.5646099152481705E-12
155	Buckets($\mu_0 = 25,p = 40\%$) vs. Buckets($\mu_0 = 40,p = 20\%$)	6.985232870911994	2.843838765122819E-12
154	Buckets($\mu_0 = 10,p = 30\%$) vs. Buckets($\mu_0 = 25,p = 20\%$)	6.964381429506302	3.2984981215509108E-12
153	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 25,p = 50\%$)**	**6.964381429506293**	**3.2984981215511135E-12**
152	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 25,p = 50\%$)	6.94352998810059	3.82421383769864E-12
151	Buckets($\mu_0 = 25,p = 50\%$) vs. Buckets($\mu_0 = 40,p = 30\%$)	6.891401384586341	5.524541402050541E-12
150	Buckets($\mu_0 = 10,p = 50\%$) vs. Buckets($\mu_0 = 40,p = 40\%$)	6.839272781072054	7.959619624016424E-12
149	Buckets($\mu_0 = 25,p = 50\%$) vs. Buckets($\mu_0 = 40,p = 20\%$)	6.80799561896349	9.896787772324926E-12
148	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 25,p = 40\%$)**	**6.7871441775577885**	**1.143748971198273E-11**
147	Buckets($\mu_0 = 10,p = 10\%$) vs. Buckets($\mu_0 = 25,p = 30\%$)	6.6933126912321175	2.1817424466660878E-11
146	Buckets($\mu_0 = 25,p = 40\%$) vs. Buckets($\mu_0 = 40,p = 50\%$)	6.693312691232096	2.1817424466664048E-11
145	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 10,p = 10\%$)	6.641184087717837	3.11173101278676E-11
144	Buckets($\mu_0 = 10,p = 50\%$) vs. Buckets($\mu_0 = 25,p = 20\%$)	6.630758367014975	3.3396655351355166E-11
143	Buckets($\mu_0 = 25,p = 50\%$) vs. Buckets($\mu_0 = 40,p = 50\%$)	6.516075439283592	7.217068890206595E-11
142	Buckets($\mu_0 = 10,p = 30\%$) vs. Buckets($\mu_0 = 40,p = 40\%$)	6.505649718580727	7.735829363362602E-11
141	ε-greedy($\varepsilon = 0.5$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	6.495223997877894	8.290995677060107E-11
140	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 10,p = 50\%$)	6.495223997877876	8.290995677061131E-11
139	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 10,p = 50\%$)	6.255432421712261	3.9641597112652544E-10
138	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 40,p = 40\%$)	6.255432421712243	3.964159711265731E-10
137	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 10,p = 30\%$)	6.161600935386548	7.201314563831752E-10
136	Buckets($\mu_0 = 25,p = 50\%$) vs. Buckets($\mu_0 = 25,p = 10\%$)	6.06776944906087	1.2969890615906653E-9
135	ε-greedy($\varepsilon = 0.1$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	6.057343728358043	1.3838767296312335E-9

134	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 40, p = 10\%$)	5.932235079923782	2.9883833247771147E-9
133	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 10, p = 30\%$)	5.9218093592209335	3.1841862663277764E-9
132	ε-greedy($\varepsilon = 0.3$) vs. Thompson Samp.($\mu_0 = 10$)	5.911383638518064	3.3924587675842374E-9
131	Buckets($\mu_0 = 25, p = 40\%$) vs. Buckets($\mu_0 = 25, p = 10\%$)	5.890532197112365	3.849538573161694E-9
130	Buckets($\mu_0 = 40, p = 30\%$) vs. Buckets($\mu_0 = 40, p = 10\%$)	5.880106476409533	4.100026536832547E-9
129	Buckets($\mu_0 = 10, p = 50\%$) vs. Buckets($\mu_0 = 10, p = 40\%$)	5.827977872895243	5.610301507547337E-9
128	Buckets($\mu_0 = 40, p = 20\%$) vs. Buckets($\mu_0 = 40, p = 10\%$)	5.796700710786682	6.7632230840113315E-9
127	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 40, p = 10\%$)	5.796700710786673	6.763223084011725E-9
126	ε-greedy($\varepsilon = 0.4$) vs. Thompson Samp.($\mu_0 = 10$)	5.67159206235245	1.4147653943436176E-8
125	Buckets($\mu_0 = 40, p = 50\%$) vs. Buckets($\mu_0 = 40, p = 10\%$)	5.504780531106784	3.6962844415833006E-8
124	Buckets($\mu_0 = 10, p = 40\%$) vs. Buckets($\mu_0 = 10, p = 30\%$)	5.494354810403916	3.9214138652927804E-8
123	Buckets($\mu_0 = 25, p = 20\%$) vs. Buckets($\mu_0 = 40, p = 10\%$)	5.358820441266813	8.376705642635159E-8
122	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 25, p = 30\%$)	5.317117558455407	1.0542396021530812E-7
121	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 10, p = 40\%$)	5.244137513535431	1.570150656362451E-7
120	Buckets($\mu_0 = 10, p = 30\%$) vs. Buckets($\mu_0 = 25, p = 30\%$)	5.066900261586922	4.04346005929267E-7
119	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 25, p = 40\%$)	5.004345937369791	5.6052019219704E-7
118	Buckets($\mu_0 = 10, p = 10\%$) vs. Buckets($\mu_0 = 40, p = 40\%$)	4.879237288935532	1.0649687416562254E-6
117	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 25, p = 50\%$)	4.827108685421287	1.385295285295501E-6
116	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 40, p = 50\%$)	4.8062572440155895	1.5378200588743428E-6
115	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 25, p = 50\%$)	4.785405802609865	1.7064220395791745E-6
114	buckets($\mu_0 = 10$, $p = 50\%$) vs. Buckets($\mu_0 = 25, p = 30\%$)	4.733277199095595	2.2092344592582423E-6
113	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 10, p = 50\%$)	4.66029715417562	3.157532013103586E-6
112	Buckets($\mu_0 = 10, p = 50\%$) vs. Buckets($\mu_0 = 40, p = 30\%$)	4.608168550661371	4.062313039212939E-6
111	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 25, p = 40\%$)	4.60816855066136	4.0623130392131495E-6
110	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 10, p = 10\%$)	4.5351885057413535	5.755205974989636E-6
109	Buckets($\mu_0 = 10, p = 50\%$) vs. Buckets($\mu_0 = 40, p = 20\%$)	4.52476278503852	6.046328386166638E-6
108	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 40, p = 20\%$)	4.514337064335691	6.351513133273166E-6
107	**Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 25, p = 30\%$)**	**4.514337064335669**	**6.351513133273825E-6**
106	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 40, p = 30\%$)	4.43093129871284	9.382698117206424E-6
105	Buckets($\mu_0 = 10, p = 10\%$) vs. Buckets($\mu_0 = 25, p = 40\%$)	4.420505578009996	9.847024368645032E-6
104	ε-greedy($\varepsilon = 0.2$) vs. ε-greedy($\varepsilon = 0.5$)	4.378802695198591	1.193331081545521E-5
103	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 40, p = 50\%$)	4.368376974495738	1.2517329620068423E-5
102	Buckets($\mu_0 = 25, p = 50\%$) vs. Buckets($\mu_0 = 25, p = 20\%$)	4.347525533090005	1.3768204671970711E-5
101	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 10, p = 30\%$)	4.326674091684293	1.5137768393700672E-5
100	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 10, p = 10\%$)	4.295396929575739	1.74381043212871E-5
99	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 40, p = 40\%$)	4.2953969295757375	1.7438104321287226E-5
98	Buckets($\mu_0 = 10, p = 30\%$) vs. Buckets($\mu_0 = 40, p = 30\%$)	4.274545488170043	1.9152760115834308E-5
97	Buckets($\mu_0 = 10, p = 10\%$) vs. Buckets($\mu_0 = 25, p = 50\%$)	4.243268326061492	2.202877762977228E-5
96	Buckets($\mu_0 = 10, p = 50\%$) vs. Buckets($\mu_0 = 40, p = 50\%$)	4.232842605358622	2.3075600326673513E-5
95	Buckets($\mu_0 = 10, p = 30\%$) vs. Buckets($\mu_0 = 40, p = 20\%$)	4.191139722547193	2.7755665829004007E-5
94	Buckets($\mu_0 = 25, p = 40\%$) vs. Buckets($\mu_0 = 25, p = 20\%$)	4.1702882811415005	3.042145654735582E-5
93	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 40, p = 20\%$)	4.076456794815839	4.572714721363635E-5
92	ε-greedy($\varepsilon = 0.2$) vs. Thompson Samp.($\mu_0 = 10$)	4.076456794815808	4.572714721364244E-5
91	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 40, p = 30\%$)	4.024328191301559	5.713818954618785E-5
90	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 40, p = 30\%$)	3.9930510291929893	6.522852390441114E-5

89	ε-greedy($\varepsilon = 0.3$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	3.9513481463815583	7.771217621451557E-5
88	ε-greedy($\varepsilon = 0.1$) vs. ε-greedy($\varepsilon = 0.2$)	3.94092242567874	8.116887481214492E-5
87	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	3.9409224256787083	8.116887481215561E-5
86	Buckets($\mu_0 = 10$,$p = 30\%$) vs. Buckets($\mu_0 = 40$,$p = 50\%$)	3.8992195428672947	9.650324455641392E-5
85	Buckets($\mu_0 = 10$,$p = 40\%$) vs. Buckets($\mu_0 = 10$,$p = 10\%$)	3.8679423807587208	1.0975759026689523E-4
84	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 40$,$p = 10\%$)	3.8158137772444793	1.3573484712640343E-4
83	ε-greedy($\varepsilon = 0.4$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	3.711556570215944	2.0598861803955984E-4
82	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 40$,$p = 50\%$)	3.64900224599881	2.632608027034708E-4
81	Buckets($\mu_0 = 25$,$p = 30\%$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	3.617725083890244	2.9720383551039357E-4
80	Buckets($\mu_0 = 25$,$p = 30\%$) vs. Buckets($\mu_0 = 40$,$p = 10\%$)	3.461339273347433	5.374949645234997E-4
79	Buckets($\mu_0 = 10$,$p = 40\%$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	3.284102021398926	0.0010230787988017604
78	Buckets($\mu_0 = 10$,$p = 10\%$) vs. Buckets($\mu_0 = 40$,$p = 10\%$)	3.231973417884684	0.0012293849189447686
77	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 10$,$p = 40\%$)	3.211121976478968	0.0013221780028460843
76	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 25$,$p = 40\%$)	3.0443104452332856	0.0023321434546417167
75	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 25$,$p = 50\%$)	2.8670731932847815	0.0041428718667235855
74	Buckets($\mu_0 = 10$,$p = 30\%$) vs. Buckets($\mu_0 = 25$,$p = 40\%$)	2.7940931483648015	0.005204548483616971
73	ε-greedy($\varepsilon = 0.4$) vs. ε-greedy($\varepsilon = 0.5$)	2.7836674276619497	0.005374811413704696
72	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 10$,$p = 40\%$)	2.7732417069591166	0.00555008802570923
71	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 10$,$p = 10\%$)	2.7002616620390976	0.006928496006008277
70	Buckets($\mu_0 = 10$,$p = 10\%$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)	2.6481330585248486	0.008093766191962181
69	Buckets($\mu_0 = 10$,$p = 30\%$) vs. Buckets($\mu_0 = 25$,$p = 50\%$)	2.6168558964162973	0.008874379742771499
68	Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 25$,$p = 20\%$)	2.6168558964162885	0.008874379742771735
67	Buckets($\mu_0 = 40$,$p = 50\%$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	2.6064301757134327	0.009149147579593791
66	Buckets($\mu_0 = 10$,$p = 10\%$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	2.564727292901998	0.010325692499780142
65	ε-greedy($\varepsilon = 0.3$) vs. ε-greedy($\varepsilon = 0.5$)	2.5438758514963355	0.010963004173647326
64	buckets($\mu_0 = 10$, $p = 50\%$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	2.543875851496317	0.01096300417364791
63	Buckets($\mu_0 = 10$,$p = 50\%$) vs. Buckets($\mu_0 = 25$,$p = 40\%$)	2.460470085873474	0.01387551387166986
62	Buckets($\mu_0 = 25$,$p = 50\%$) vs. Buckets($\mu_0 = 25$,$p = 30\%$)	2.4500443651706254	0.014283861390976047
61	ε-greedy($\varepsilon = 0.1$) vs. ε-greedy($\varepsilon = 0.4$)	2.3457871581420986	0.018986946459597885
60	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 25$,$p = 30\%$)	2.3353614374392393	0.01952454715565609
59	Buckets($\mu_0 = 40$,$p = 40\%$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	2.314509996033534	0.020639763576319466
58	Buckets($\mu_0 = 10$,$p = 50\%$) vs. Buckets($\mu_0 = 25$,$p = 50\%$)	2.28323283392497	0.022416657012725626
57	Buckets($\mu_0 = 25$,$p = 40\%$) vs. Buckets($\mu_0 = 25$,$p = 30\%$)	2.2728071132221213	0.02303780620434418
56	Buckets($\mu_0 = 10$,$p = 10\%$) vs. Buckets($\mu_0 = 40$,$p = 50\%$)	2.2728071132221	0.02303780620434547
55	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 40$,$p = 50\%$)	2.2623813925192535	0.023673849562855113
54	Buckets($\mu_0 = 40$,$p = 40\%$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)	2.2311042304106836	0.025674225602204108
53	Buckets($\mu_0 = 10$,$p = 30\%$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	2.21025278900499	0.027087623161060368
52	ε-greedy($\varepsilon = 0.5$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	2.199827068302157	0.027819166724312457
51	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	2.1789756268964346	0.029333478882314287
50	Thompson Samp.($\mu_0 = 25$) vs. Thompson Samp.($\mu_0 = 40$)	2.178975626896429	0.029333478882314717
49	ε-greedy($\varepsilon = 0.2$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	2.1164213026793024	0.03430899186806513
48	ε-greedy($\varepsilon = 0.1$) vs. ε-greedy($\varepsilon = 0.3$)	2.1059955819764844	0.03520473611880302
47	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)	2.0642926991650534	0.03898997282365532
46	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 40$,$p = 50\%$)	2.0225898163536393	0.04311545500316793
45	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	1.9808869335422032	0.047603954727872336

44	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	1.9704612128393553	0.04878553608198774
43	Buckets($\mu_0 = 10$,$p = 50\%$) vs. Buckets($\mu_0 = 10$,$p = 10\%$)	1.960035492136522	0.04999164205277938
42	Thompson Samp.($\mu_0 = 10$) vs. buckets($\mu_0 = 10$, $p = 20\%$)	1.9600354921365053	0.049991642052781324
41	Buckets($\mu_0 = 25$,$p = 30\%$) vs. Buckets($\mu_0 = 25$,$p = 20\%$)	1.8974811679193797	0.05776446195402968
40	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)	1.8870554472165049	0.05915286827634361
39	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 40$,$p = 10\%$)	1.8557782851079738	0.06348516695506146
38	ε-greedy($\varepsilon = 0.2$) vs. ε-greedy($\varepsilon = 0.3$)	1.834926843702256	0.06651652965727775
37	ε-greedy($\varepsilon = 0.1$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	1.7619467987823052	0.07807828659974222
36	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	1.730669636673741	0.08351070370789908
35	Buckets($\mu_0 = 25$,$p = 20\%$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	1.7202439159708642	0.08538811331093744
34	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 40$,$p = 50\%$)	1.6889667538623048	0.09122580132655143
33	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)	1.6472638710508907	0.09950382054487006
32	Buckets($\mu_0 = 10$,$p = 30\%$) vs. Buckets($\mu_0 = 10$,$p = 10\%$)	1.6264124296451947	0.10386194842612831
31	Buckets($\mu_0 = 10$,$p = 30\%$) vs. Buckets($\mu_0 = 40$,$p = 10\%$)	1.6055609882394895	0.10837040284968448
30	ε-greedy($\varepsilon = 0.2$) vs. ε-greedy($\varepsilon = 0.4$)	1.5951352675366417	0.11068199042334662
29	Buckets($\mu_0 = 10$,$p = 40\%$) vs. Buckets($\mu_0 = 40$,$p = 50\%$)	1.5951352675366213	0.1106819904233512
28	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 10$,$p = 10\%$)	1.3761951327767104	0.16876122858514425
27	Buckets($\mu_0 = 10$,$p = 40\%$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	1.3032150878567228	0.1925013405215716
26	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	1.2823636464510046	0.19971511061364636
25	Buckets($\mu_0 = 10$,$p = 50\%$) vs. Buckets($\mu_0 = 40$,$p = 10\%$)	1.2719379257481622	0.20339517532957962
24	Buckets($\mu_0 = 10$,$p = 40\%$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)	1.2198093222338724	0.22253716663534534
23	Buckets($\mu_0 = 25$,$p = 40\%$) vs. Buckets($\mu_0 = 40$,$p = 10\%$)	1.1885321601253118	0.2346238172437917
22	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 10$,$p = 40\%$)	1.1676807187196234	0.24293557709463853
21	Buckets($\mu_0 = 10$,$p = 40\%$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	1.0112949081768114	0.311875302381461
20	Buckets($\mu_0 = 25$,$p = 50\%$) vs. Buckets($\mu_0 = 40$,$p = 10\%$)	1.0112949081768077	0.31187530238146277
19	Thompson Samp.($\mu_0 = 25$) vs. Buckets($\mu_0 = 25$,$p = 10\%$)	0.8966119804454241	0.369926002967919
18	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 10$,$p = 40\%$)	0.6672461249826324	0.504614933006498
17	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 10$,$p = 50\%$)	0.5838403593598117	0.5593277219593547
16	buckets($\mu_0 = 10$, $p = 20\%$) vs. Buckets($\mu_0 = 10$,$p = 10\%$)	0.5838403593597951	0.5593277219593659
15	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	0.5838403593597932	0.559327721959367
14	Thompson Samp.($\mu_0 = 40$) vs. Buckets($\mu_0 = 25$,$p = 20\%$)	0.4378802695198597	0.6614730790423407
13	ε-greedy($\varepsilon = 0.1$) vs. ε-greedy($\varepsilon = 0.5$)	0.4378802695198514	0.6614730790423468
12	ε-greedy($\varepsilon = 0.4$) vs. Buckets($\mu_0 = 10$,$p = 40\%$)	0.4274545488170182	0.6690482833337136
11	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 40$,$p = 50\%$)	0.42745454881699785	0.6690482833337283
10	Buckets($\mu_0 = 40$,$p = 50\%$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)	0.37532594530274876	0.707418072630695
9	ε-greedy($\varepsilon = 0.3$) vs. Buckets($\mu_0 = 40$,$p = 40\%$)	0.344048783194179	0.7308095992509926
8	Buckets($\mu_0 = 10$,$p = 50\%$) vs. Buckets($\mu_0 = 10$,$p = 30\%$)	0.33362306249132734	0.7386640130762037
7	Buckets($\mu_0 = 40$,$p = 50\%$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	0.2919201796798984	0.7703476540230126
6	Thompson Samp.($\mu_0 = 10$) vs. Buckets($\mu_0 = 10$,$p = 30\%$)	0.2502172968684844	0.8024193096579612
5	ε-greedy($\varepsilon = 0.3$) vs. ε-greedy($\varepsilon = 0.4$)	0.2397915761656142	0.8104918377268797
4	Buckets($\mu_0 = 25$,$p = 50\%$) vs. Buckets($\mu_0 = 25$,$p = 40\%$)	0.1772372519485042	0.8593220360066377
3	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	0.1355343691370994	0.8921893923076426
2	Buckets($\mu_0 = 40$,$p = 30\%$) vs. Buckets($\mu_0 = 40$,$p = 20\%$)	0.08340576562285035	0.933528904183899
1	ε-greedy($\varepsilon = 0.2$) vs. Buckets($\mu_0 = 40$,$p = 30\%$)	0.052128603514249056	0.9584262216358782

TABLE A.1: Nemeneyi test for $\alpha = 0.05$. The horizontal line shows the pairs whose p-value allows as to reject the Null hypothesis. We highlight the pairs in which the performance of standard Thompson Sampling is significantly different than the algorithm is compared against

Appendix B

Golden-section search algorithm

The golden-section search method is used to find the maximum or minimum of a unimodal function f. i.e. a function that contains only one minimum or maximum in the interval $[a, b]$. Let say we have three points x_u, x_v, x_w such that $x_u < x_v < x_w$ with corresponding values $f(x_u)$, $f(x_v)$, $f(x_w)$. If $f(x_v) > f(x_u)$ and $f(x_v) > f(x_w)$ the maximum lies between x_v and x_w. A fourth point x' is chosen to be between the larger of the two intervals of $[x_u, x_v]$ and $[x_v, x_w]$. If $[x_u, x_v]$ is larger than $[x_v, x_w]$ we would choose $[x_u, x_v]$ as the interval in which x' is chosen. Otherwise we would choose $[x_v, x_w]$ as the interval in which x' is chosen.

If $f(x') > f(x_v)$ then the new three points would be $x_u < x' <$. If $f(x') < f(x_v)$ then the three new points would be $x' < x_v < x_w$. The process is continued until the distance between the outer points is sufficiently small. In other words, in order to find the maximum of a point we follow the steps below:

1. Define the points x_u and x_w that contain the maximum of the function $f(x)$.

2. Determine two intermediate points x_v and x' such that $x_v = x_u + d$ and $x' = x_w - d$, where $d = \frac{\sqrt{5}-1}{2}(x_w - x_u)$

3. if $f(x_v) > f(x')$, then determine new x_u, x_v, x' and x_w using: $x_u = x'$, $x' = x_v$, $x_w = x_w$, $x_v = x_u + \frac{\sqrt{5}-1}{2}(x_w - x_u)$

   if $f(x_v) < f(x')$, then determine new x_u, x_v, x' and x_w using: $x_u = x_u$, $x_w = x_v$, $x_v = x'$, $x' = x_u + \frac{\sqrt{5}-1}{2}(x_w - x_u)$

4. If $x_w - x_u, \varepsilon$, then the max point occurs at $\frac{x_w + x_u}{2}$ and stop iterating.

Appendix C

Revenue share in oligopoly for different parameter values

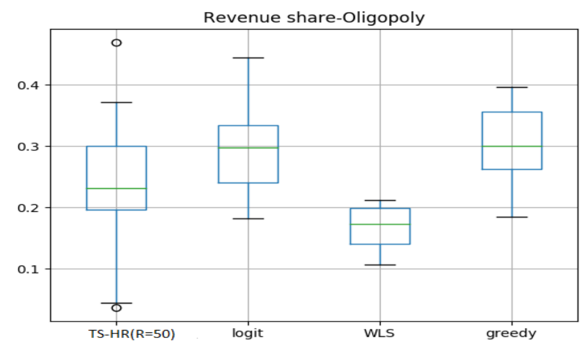


FIGURE C.1: Revenue share for the four competitors in oligopoly when $R = 50$, in TS-HR. The results are averaged over 10 independent simulation runs

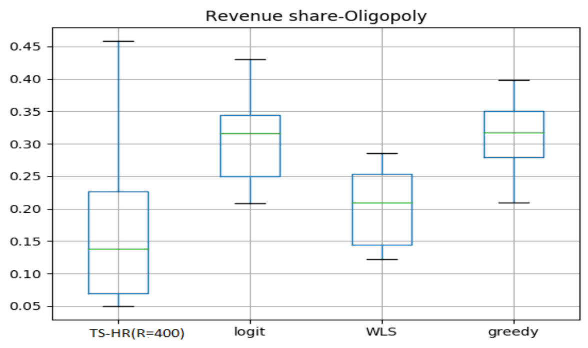


FIGURE C.2: Revenue share for the four competitors in oligopoly when $R = 400$, in TS-HR. The results are averaged over 10 independent simulation runs

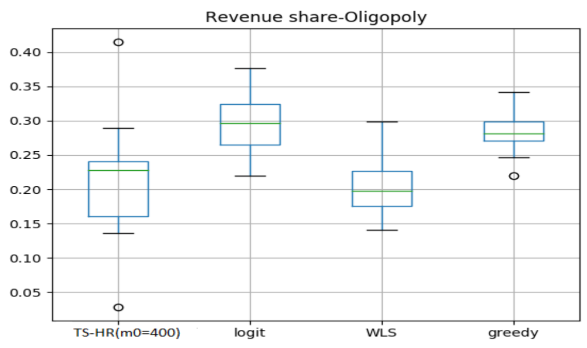


FIGURE C.3: Revenue share for the four competitors in oligopoly when $\mu_0 = 400$, in TS-HR. The results are averaged over 10 independent simulation runs

Appendix D

Contextual TS-HR

We developed a new pricing strategy based on contextual Thompson Sampling. The assumed context in our algorithm is dictated by the competitors' prices in the previous time step. Based on some preliminary results, we consider only the minimum price offered by the competitors at $t-1$ as context.

We split the price interval we are allowed to charge into u buckets, where in the current example $u=5$. Competitors' bucket chosen at time t is defined as $B'_{i,t}$ while the price bucket we choose at time t is defined as $B_{i,t}$, where $i=1,2,..,u$. In the case where $u=5$ there are 25 context-action pairs. Each of these context-action pairs has an associated Normal reward distribution with unknown mean and variance. At each time step t, we update the reward distribution corresponding to the context-action pair $(B'_{i,t}, B_{i,t})$. However, our buckets and competitors' buckets change and this make the task of updating the reward distribution of each pair even more challenging. Below we describe the main parts of the pricing strategy.

Update of competitors' buckets

We define n equally spaced buckets. Every j time steps we do the following:

- Order $p'(t)$ in non-decreasing order of the values in $p'(t)$.
- Set the bucket limits of $B'_{i,t}, \forall i = 1,2,..,5$ so that each bucket contains the same number of observations.

Update of our buckets

For the update of our buckets we use the same mechanism which is based on Hoeffding races, as we did in TS-HR. We use only the revenue obtained by our own price buckets we have chosen so far and ignore context.

- Apply Hoeffding races to decide if there is sufficient information to eliminate any of the u buckets.

- If a bucket is eliminated we find the bucket with the highest mean revenue obtained so far and split it into two new equally spaced buckets.

Recalculate bucket distributions

Every time either the competitors' buckets or our buckets change, we recalculate the reward distribution for each (B'_i, B_i) pair.

- Identify which observations correspond to $(B'_{i,t}, B_{i,t})$ and update $b'(t)$ and $b(t)$ where $b'(t)$ is the set with competitors' buckets up to $t-1$ and $b(t)$ is the set with our buckets up to $t-1$.

Bibliography

Agarwal, Deepak, Bee-Chung Chen, and Pradheep Elango (2009). "Explore/exploit schemes for web content optimization". In: *2009 Ninth IEEE International Conference on Data Mining (ICDM)*. IEEE, pp. 1–10.

Agarwal, Deepak, Bee-Chung Chen, Pradheep Elango, Nitin Motgi, Seung-Taek Park, Raghu Ramakrishnan, Scott Roy, and Joe Zachariah (2009). "Online models for content optimization". In: *Advances in Neural Information Processing Systems*, pp. 17–24.

Agrawal, Rajeev (1995). "The continuum-armed bandit problem". In: *SIAM journal on control and optimization* 33.6, pp. 1926–1951.

Agrawal, Shipra and Navin Goyal (2012). "Analysis of Thompson Sampling for the Multi-armed Bandit Problem". In: *Conference On Learning Theory (COLT)*, pp. 39–1.

Agrawal, Shipra and Navin Goyal (2013a). "Further Optimal Regret Bounds for Thompson Sampling". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 99–107.

Agrawal, Shipra and Navin Goyal (2013b). "Thompson Sampling for Contextual Bandits with Linear Payoffs". In: *International Conference on Machine Learning (ICML)*, pp. 127–135.

Akaike, Hirotugu (1974). "A new look at the statistical model identification". In: *IEEE transactions on automatic control* 19.6, pp. 716–723.

Amirizadeh, Khosrow and Rajeswari Mandava (2015). "Accelerated-Greedy Multi Armed Bandit Algorithm for Online Sequential-Selections Applications". In: *Journal of Software* 10.3, pp. 239–249.

Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer (2002). "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* 47.2-3, pp. 235–256.

Auer, Peter, Ronald Ortner, and Csaba Szepesvári (2007). "Improved rates for the stochastic continuum-armed bandit problem". In: *International Conference on Computational Learning Theory (COLT)*. Springer, pp. 454–468.

Ban, Gah-Yi and N Bora Keskin (2018). "Personalized dynamic pricing with machine learning". In: *Available at SSRN 2972985*.

Barto, Andrew G (2010). "Adaptive Real-Time Dynamic Programming". In: *Encyclopedia of Machine Learning*. Springer US, pp. 19–22.

Barto, Andrew G, Steven J Bradtke, and Satinder P Singh (1995). "Learning to act using real-time dynamic programming". In: *Artificial intelligence* 72.1-2, pp. 81–138.

Bechhofer, Robert E (1954). "A single-sample multiple decision procedure for ranking means of normal populations with known variances". In: *The Annals of Mathematical Statistics*, pp. 16–39.

Besbes, Omar and Alp Muharremoglu (2013). "On implications of demand censoring in the newsvendor problem". In: *Management Science* 59.6, pp. 1407–1424.

Besbes, Omar and Assaf Zeevi (2009). "Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms". In: *Operations Research* 57.6, pp. 1407–1420.

Besbes, Omar and Assaf Zeevi (2015). "On the (surprising) sufficiency of linear models for dynamic pricing with demand learning". In: *Management Science* 61.4, pp. 723–739.

Boer, Arnoud den and N Bora Keskin (2017). "Dynamic Pricing with Demand Learning and Reference Effects". In: *Available at SSRN 3092745*.

Boer, Arnoud V den (2015). "Dynamic pricing and learning: historical origins, current research, and new directions". In: *Surveys in operations research and management science* 20.1, pp. 1–18.

Boesel, Justin, Barry L Nelson, and Seong-Hee Kim (2003). "Using ranking and selection to "clean up" after simulation optimization". In: *Operations Research* 51.5, pp. 814–825.

Bouneffouf, Djallel and Raphael Feraud (2016). "Multi-armed bandit problem with known trend". In: *Neurocomputing* 205, pp. 16–21.

Branke, Jürgen, Stephen E Chick, and Christian Schmidt (2007). "Selecting a selection procedure". In: *Management Science* 53.12, pp. 1916–1932.

Burtini, G., J. Loeppky, and R. Lawrence (2015). "A Survey of Online Experiment Design with the Stochastic Multi-Armed Bandit". In: *arXiv preprint arXiv:1510.00757*.

Chapelle, Olivier and Lihong Li (2011). "An empirical evaluation of thompson sampling". In: *Advances in neural information processing systems*, pp. 2249–2257.

Chau, Marie, Michael C Fu, Huashuai Qu, and Ilya O Ryzhov (2014). "Simulation optimization: a tutorial overview and recent developments in gradient-based methods". In: *Proceedings of the 2014 Winter Simulation Conference (WSC)*. IEEE Press, pp. 21–35.

Chen, Chun-Hung (1995). “An effective approach to smartly allocate computing budget for discrete event simulation”. In: *Proceedings of the 34th IEEE Conference on Decision and Control (CDC)*. Vol. 3. IEEE, pp. 2598–2603.

Demšar, Janez (2006). “Statistical comparisons of classifiers over multiple data sets”. In: *Journal of Machine learning research*, pp. 1–30.

Elmaghraby, Wedad and Pınar Keskinocak (2003). “Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions”. In: *Management science* 49.10, pp. 1287–1309.

Faruqui, Ahmad and Lisa Wood. *Quantifying the benefits of dynamic pricing in the mass market*. Tech. rep.

Ferreira, Kris Johnson, David Simchi-Levi, and He Wang (2018). “Online network revenue management using thompson sampling”. In: *Operations research* 66.6, pp. 1586–1602.

Friedman, Milton (1937). “The use of ranks to avoid the assumption of normality implicit in the analysis of variance”. In: *Journal of the american statistical association* 32.200, pp. 675–701.

Fu, Michael C et al. (2015). *Handbook of Simulation Optimization*. Vol. 216. Springer.

Garcia, Salvador and Francisco Herrera (2008). “An extension on”statistical comparisons of classifiers over multiple data sets”for all pairwise comparisons”. In: *Journal of Machine Learning Research* 9, pp. 2677–2694.

Garivier, Aurélien and Eric Moulines (2011). “On upper-confidence bound policies for switching bandit problems”. In: *International Conference on Algorithmic Learning Theory (ALT)*. Springer, pp. 174–188.

Geer, Ruben van de et al. (2018). “Dynamic pricing and learning with competition: Insights from the dynamic pricing challenge at the 2017 informs rm & pricing conference”. In: *Journal of Revenue and Pricing Management*, pp. 1–19.

Gittins, John C (1979). “Bandit processes and dynamic allocation indices”. In: *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 148–177.

Graepel, Thore, Joaquin Q Candela, Thomas Borchert, and Ralf Herbrich (2010). “Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft’s bing search engine”. In: *Proceedings of the 27th international conference on machine learning (ICML)*, pp. 13–20.

Griffin, Jim E, Philip J Brown, et al. (2010). “Inference with normal-gamma prior distributions in regression problems”. In: *Bayesian Analysis* 5.1, pp. 171–188.

Hartland, Cédric, Nicolas Baskiotis, Sylvain Gelly, Michele Sebag, and Olivier Teytaud (2007). “Change point detection and meta-bandits for online learning in dynamic environments". In: *Conférence Francophone sur l'apprentissage automatique, Cepadues (CAp)*, pp. 237–250.

Hoeffding, Wassily (1963). "Probability inequalities for sums of bounded random variables". In: *Journal of the American statistical association* 58.301, pp. 13–30.

Honda, Junya and Akimichi Takemura (2014). "Optimality of Thompson Sampling for Gaussian Bandits Depends on Priors". In: *International Conference on Artificial Intelligence and Statistics(AISTATS)*, pp. 375–383.

Iman, Ronald L and James M Davenport (1980). "Approximations of the critical region of the Friedman statistic". In: *Communications in Statistics-Theory and Methods* 9.6, pp. 571–595.

Kaufmann, Emilie, Nathaniel Korda, and Rémi Munos (2012). "Thompson sampling: An asymptotically optimal finite-time analysis". In: *International Conference on Algorithmic Learning Theory (ALT)*. Springer, pp. 199–213.

Keskin, N Bora and Assaf Zeevi (2014). "Dynamic pricing with an unknown demand model: Asymptotically optimal semi-myopic policies". In: *Operations Research* 62.5, pp. 1142–1167.

Kiefer, Jack (1953). "Sequential minimax search for a maximum". In: *Proceedings of the American mathematical society* 4.3, pp. 502–506.

Kim, Seong-Hee and AB Dieker (2011). "Selecting the best by comparing simulated systems in a group of three". In: *Proceedings of the 2011 Winter Simulation Conference (WSC)*. IEEE, pp. 3987–3997.

Kim, Seong-Hee and Barry L Nelson (2006a). "On the asymptotic validity of fully sequential selection procedures for steady-state simulation". In: *Operations Research* 54.3, pp. 475–488.

Kim, Seong-Hee and Barry L Nelson (2006b). "Selecting the best system". In: *Handbooks in operations research and management science* 13, pp. 501–534.

Kocsis, Levente and Csaba Szepesvári (2006). "Discounted UCB". In: *2nd PASCAL Challenges Workshop*, pp. 784–791.

Kohavi, Ron, Roger Longbotham, Dan Sommerfield, and Randal M Henne (2009). "Controlled experiments on the web: survey and practical guide". In: *Data mining and knowledge discovery* 18.1, pp. 140–181.

Koulouriotis, Dimitris E and A Xanthopoulos (2008). "Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems". In: *Applied Mathematics and Computation* 196.2, pp. 913–922.

Kuleshov, Volodymyr and Doina Precup (2014). "Algorithms for multi-armed bandit problems". In: *Journal of Machine Learning Research*.

Lai, Tze Leung and Herbert Robbins (1985). “Asymptotically efficient adaptive allocation rules”. In: *Advances in applied mathematics* 6.1, pp. 4–22.

Lessmann, Stefan, Bart Baesens, Christophe Mues, and Swantje Pietsch (2008). “Benchmarking classification models for software defect prediction: A proposed framework and novel findings”. In: *IEEE Transactions on Software Engineering* 34.4, pp. 485–496.

Li, Lihong, Wei Chu, John Langford, and Robert E Schapire (2010a). “A contextual-bandit approach to personalized news article recommendation”. In: *Proceedings of the 19th international conference on World wide web*. ACM, pp. 661–670.

Li, Wei, Xuerui Wang, Ruofei Zhang, Ying Cui, Jianchang Mao, and Rong Jin (2010b). “Exploitation and exploration in a performance based contextual advertising system”. In: *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, pp. 27–36.

Lloyd, Kevin and David S Leslie (2013). “Context-dependent decision-making: a simple Bayesian model”. In: *Journal of The Royal Society Interface* 10.82, p. 20130069.

Madani, Omid and Dennis DeCoste (2005). “Contextual recommender problems”. In: *Proceedings of the 1st international workshop on Utility-based data mining*. ACM, pp. 86–89.

Maron, Oded and Andrew W Moore (1994). “Hoeffding races: Accelerating model selection search for classification and function approximation”. In: *Advances in neural information processing systems*, pp. 59–66.

May, Benedict C, Nathan Korda, Anthony Lee, and David S Leslie (2012). “Optimistic Bayesian sampling in contextual-bandit problems”. In: *Journal of Machine Learning Research* 13, pp. 2069–2106.

Mellor, Joseph and Jonathan Shapiro (2013). “Thompson sampling in switching environments with Bayesian online change detection”. In: *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 442–450.

Misra, Kanishka, Eric M Schwartz, and Jacob Abernethy (2019a). “Dynamic Online Pricing with Incomplete Information Using Multi-Armed Bandit Experiments”. In: *Marketing Science* 38.2, pp. 226–252.

Misra, Kanishka, Eric M Schwartz, and Jacob Abernethy (2019b). “Dynamic Online Pricing with Incomplete Information Using Multi-Armed Bandit Experiments”. In: *Marketing Science* 38.2, pp. 226–252.

Nemenyi, Peter (1962). “Distribution-free multiple comparisons”. In: *Biometrics*. Vol. 18. 2, p. 263.

Özer, Özalp, Ozalp Ozer, and Robert Phillips (2012). *The Oxford handbook of pricing management*. Oxford University Press.

Porter, Barry, Matthew Grieves, Roberto Rodrigues Filho, and David Leslie (2016). "{REX}: A Development Platform and Online Learning Approach for Runtime Emergent Software Systems". In: *12th Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 333–348.

Raiffa, Howard and Robert Schlaifer (1961). *Applied statistical decision theory*. Division of Research, Graduate School of Business Adminitration, Harvard ...

Rinott, Yosef (1978). "On two-stage selection procedures and related probability-inequalities". In: *Communications in Statistics-Theory and methods* 7.8, pp. 799–811.

Rothschild, Michael (1974). "A two-armed bandit theory of market pricing". In: *Journal of Economic Theory* 9.2, pp. 185–202.

Russo, Daniel J, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. (2018). "A Tutorial on Thompson Sampling". In: *Foundations and Trends in Machine Learning* 11.1, pp. 1–96.

Sanner, Scott, Robby Goetschalckx, Kurt Driessens, and Guy Shani (2009). "Bayesian real-time dynamic programming". In: *Twenty-First International Joint Conference on Artificial Intelligence*.

Scott, Steven L (2010). "A modern Bayesian look at the multi-armed bandit". In: *Applied Stochastic Models in Business and Industry* 26.6, pp. 639–658.

Scott, Steven L (2015). "Multi-armed bandit experiments in the online service economy". In: *Applied Stochastic Models in Business and Industry* 31.1, pp. 37–45.

Slivkins, Aleksandrs and Eli Upfal (2008). "Adapting to a Changing Environment: the Brownian Restless Bandits". In: *Computational Learning Theory (COLT)*, pp. 343–354.

Sutton, Richard S and Andrew G Barto (1998). *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge.

Talluri, Kalyan T and Garrett J Van Ryzin (2006). *The theory and practice of revenue management*. Vol. 68. Springer Science & Business Media.

Thathachar, MAL (1984). "A class of rapidly converging algorithms for learning automata". In: *IEEE International Conference on Cybernetics and Society*, pp. 602–606.

Thompson, William R (1933). "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples". In: *Biometrika* 25.3/4, pp. 285–294.

Vermorel, Joannes and Mehryar Mohri (2005). "Multi-armed bandit algorithms and empirical evaluation". In: *European Conference on Machine Learning (ECML)*, pp. 437–448.

Wang, Xinxi, Yi Wang, David Hsu, and Ye Wang (2014). "Exploration in interactive personalized music recommendation: a reinforcement learning approach". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 11.1, p. 7.

Wang, Zizhuo, Shiming Deng, and Yinyu Ye (2014). "Close the gaps: A learning-while-doing algorithm for single-product revenue management problems". In: *Operations Research* 62.2, pp. 318–331.

West, Mike and Jeff Harrison (2006). *Bayesian forecasting and dynamic models*. Springer Science & Business Media.

White, Tohn Myles (2012). *Bandit Algorithms for Website Optimization Developing, Deploying, and Debugging*. O'Reilly Media.

Whittle, Peter (1988). "Restless bandits: Activity allocation in a changing world". In: *Journal of applied probability* 25.A, pp. 287–298.

Yeh, Flora Yu-Hui and Marcus Gallagher (2005). "An empirical study of hoeffding racing for model selection in k-nearest neighbor classification". In: *International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*. Springer, pp. 220–227.

Zeng, Chunqiu, Qing Wang, Shekoofeh Mokhtari, and Tao Li (2016). "Online Context-Aware Recommendation with Time Varying Multi-Armed Bandit". In: *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, pp. 2025–2034.