UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE
MASTER'S THESIS

Contextual Bandits: An application to mobile games

Author:
Enric AZUARA OLIVERA

Supervisor:
Arnau ESCAPA
Dr. Jordi VITRIÀ

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 30, 2022

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Contextual Bandits: An application to mobile games

by Enric AZUARA OLIVERA

Mobile game designers have the challenging duty of producing games where users spend their time and money. There is a wide range of possible game features that can affect users' game experience, and finding which is the most suitable is not trivial. In this thesis, we face this as an exploitation-exploration problem. We propose a contextual bandit methodology, Lin-UCB, to acknowledge how to allocate different game variants to maximize the revenue or the user experience. Diverse experiments were carried out with a historical A/B tests dataset provided by Socialpoint, a mobile game company. After an offline evaluation of the algorithm, overall results were great, especially when the game variants were related to the evaluated metric. However, the algorithm does not perform well if the metric has a low percentage of non-zero cases like in-game purchases. For the game-experience case, although game variants provided were not impactful enough to make the retention of players change, the algorithm performed decent enough. We observed how having more context of the users and more waiting time to get the reward improves the algorithm's results. Thus, the algorithm has the potential to be deployed in production as long as users are trained by batches and some prior considerations of the game variants are made. Code is available in GitHub

Keywords: Contextual bandits, Lin-UCB, offline evaluation, game experience, mobile games.

Acknowledgements

First, I want to thank Social Point for making this collaboration possible and my project advisor from there, Arnau. I really appreciated his insight, willingness to help, and experience to develop my final master thesis.

A special thanks to Jordi Vitrià and the Universitat of Barcelona for allowing me to do a collaboration with external parties and provide assistance when needed.

Lastly, thanks to those who supported and encouraged me through this rough year.

Contents

Abstract ii

Acknowledgements iii

Contents iv

1 Introduction 1
1.1 Road map . . . 2

2 Background 3
2.1 Multi-armed bandits . . . 3
2.1.1 Trade-off dilemma and A/B Testing . . . 4
2.2 Algorithm framework and variants . . . 5
2.2.1 UCB . . . 6

3 Methods 9
3.1 Lin-UCB . . . 9
3.1.1 Algorithm . . . 10
3.2 Mobile-games application . . . 11
3.3 Data . . . 12
3.3.1 Dataset exploration . . . 14
3.4 Baseline methods . . . 16
3.5 Model evaluation . . . 17

4 Results 19
4.1 Revenue maximization . . . 19
4.1.1 Ad revenue . . . 19
4.1.2 In-game purchases . . . 22
4.2 Game experience maximization . . . 25
4.2.1 Time played . . . 25
4.2.2 Number of sessions . . . 26
4.2.3 Returned to the game . . . 27

5 Conclusions 29

Bibliography 31

Chapter 1

Introduction

The starting point of this thesis is the genuine interest in video games, especially mobile ones. The mobile games industry has changed significantly over recent years, mainly in its business model. The business model from this industry is principally sustained in two ways: ad revenue and in-game purchases. Both earning methods and their interaction with the game influence the players' experience. For instance, if you download a game, and at first glance, you get three ads, or you are continuously getting asked for in-game purchases as boosts[1], skins[2] or game-pass[3], you might get overwhelmed and straight uninstall the game. On the other hand, you could have a smooth run on a game without any add or in-game purchase suggerence, but then the game's business model might not be sustainable. Game designers need to acknowledge how to find the best way to allocate the game features and the business ones to maximize the time a user plays the game and the revenue obtained. This is the main problem we try to solve in this project: how to allocate different game features to maximize the income or time played by a given user while minimizing the total costs of the process.

In this thesis, we approach this problem as an exploitation-exploration one. Understanding which is the best game-feature option for a given set of alternatives is crucial. The time we spend until we discover the best one results in high costs, especially when there are slight differences between those alternatives. This problem is one of the most studied ones in Reinforcement Learning, especially by multi-armed bandit approaches. These techniques are a classic reinforcement learning example where we have an agent that aims to maximize our reward by learning sequentially which are the best arms (game experiences) and updating the selection criteria given the reward (revenue or time played) obtained for that specific game variant. However, in the mobile-games case, we have some context about each user, so we need to use contextual bandits, a generalization of the multi-armed bandits where the agent also has a d-dimensional feature vector. Specifically, we will work with Lin-UCB, an algorithm that uses the upper confidence bound to decide

[1] Boosts are features proposed by the game to have faster progress in exchange of money.

[2] Skins are game features that allow you to modify visual aspects of the game.

[3] Game Pass is a progression method were the user pays an amount of cash to obtain rewards which unlock while game progress is being made.

which game variant is the best and considers the rewards linear with respect to its linear feature vector.

To solve this problem, we use Lin-UCB with a dual approach: the first will be the business, where we will focus on maximizing the revenue gained by ads or in-game purchases. The second viewpoint is related to the user since we try to maximize the number of hours played or the number of sessions made. To carry out these points of view, we work with real data provided by Socialpoint, a mobile-games developer based in Barcelona. They have provided an extensive database with A/B tests performed on three different games. For each game, a set number of game experiences was tested for concurrent and new game users, collecting metrics such as time played, ad revenue, etc., after one, three, and seven days. We will evaluate this dataset from an offline perspective, which will allow us to get an idea of how the algorithm performs.

Results show that applying this method to live data is viable as long as some considerations are made. First, we need to train the model with batches of users to avoid high delays in the reception of the reward. The second one is that game variants need to have a relationship with the metric we are evaluating, i.e., if we want to maximize ad revenue, game variants should have some relationship with it. Besides, the expected impact of the features will determine the α parameter that will be used to control the algorithm's sensitivity when exploring or exploiting game variants. The third condition is that the metric is not zero in most cases. We observed how we obtained poor results with the in-game purchases metric since we only get the reward 99% of the cases. Lastly, when treating with a game-experience-related metric, we need more context about the users. We can obtain it by adding short-term metrics as features, such as day one day number of sessions, and considering the second or following days as a reward.

1.1 Road map

This thesis is structured as follows:

- **Backgrund:** In this chapter, we introduce the exploitation-exploration dilemma, an overview of multi-armed bandit problems, and some algorithms to tackle the problem.
- **Methods:** In this chapter, we expose Lin-UCB algorithm and how we connect it with mobile games. After that, we describe the dataset and outline some things to consider. Finally, we explain the implementation details and the evaluation method.
- **Results:** This chapter explains the results obtained from revenue and game experience approaches when considering different metrics and datasets containing diverse game variants.
- **Conclusions:** This final chapter covers the conclusions of the thesis and future work that could be implemented to enhance the project.

Chapter 2

Background

In this section, we will provide some background needed to understand what is a multi-armed bandit problem. A short introduction about A/B tests will be given to acknowledge the problem's origin. Later, we will talk about the framework of the algorithm to solve this problem, and we will cover some fundamental aspects of algorithms that pave the way to contextual bandits.

2.1 Multi-armed bandits

Imagine you are with your mobile phone at a big building and the signal is not great where you are currently standing. The first thing you would do is start moving up until you consider the internet connection is good enough. Once you find some spots where the connection improves, you might think it worth trying a new place or returning to where you had the best connection. This problem that, as a human, we would have solved with trial and error is what is called an exploitation-exploration problem. With data usage, Machine Learning[1] tries to solve these problems, more specifically Reinforced Learning approaches. These algorithms make a sequence of decisions, where the agent learns to achieve a goal in an uncertain environment. The algorithm will get a reward or a penalty depending on the decisions made, aiming to maximize the total reward. In the Wi-Fi example, we have an agent, the person looking for better internet, the decisions to be made, that would refer to the different spots where you tried to get a connection, and a reward, which is the quality of Wi-Fi connection at the tried locations.

One way to approach the Wi-Fi example is with bandit problems. The classical bandit problem is used to show how exploration can be examined by isolation. In the standard multi-armed bandit problem, we aim to win as much money as possible from playing a set of one-armed bandits (known as slot machines) where each can give a different payout. This is a classic reinforcement learning problem that has been deeply studied previously by Even-Dar, Mannor, and Mansour, 2006 and by Lai and Robbins, 1985. This problem shows different ways to explore and solve the exploration-exploitation trade-off dilemma. We will cover this dilemma and how to face it next.

[1] Part of artificial intelligence that aims to solve tasks using data.

2.1.1 Trade-off dilemma and A/B Testing

The previous example opens up the trade-off dilemma between exploitation and exploration. It was first introduced by March., 1991. In this paper, he defined exploration as "includes things captured by terms such as search, variation, risk-taking, experimentation, play, flexibility, discovery, innovation" and exploitation as "includes such things as refinement, choice, production, efficiency, selection, implementation, execution." His definition of the dilemma covers a wide range of areas where it holds, such as personal decisions and actions or business planning in different sectors.

From the business perspective, this dilemma is more than present since it is related to resource allocation optimization, maximizing benefits from decisions, etc. One of the methods used for this purpose is A/B testing. This method consists of having a certain number of variants and randomly sending part of your target to each so you can compare them afterward. Historically, A/B testing has been used to recollect data from different groups or variants of products and make decisions after the recollecting phase is over. Some cases where A/B testing is used are websites to drive traffic, marketing campaigns, etc.

In statistical terms, we could consider the A/B testing period as an exploration time. We are only capturing how a given variant affects our traffic. After this period, we obtain the final results, and decisions are made accordingly. This has two main problems:

1. We are directly jumping from an exploration phase into an exploitation one when the ideal would be to transition between them smoothly.
2. During the exploration phase, we are wasting resources looking for bargain-basement options. The number of potentially sub-optimally resources invested will depend on how much data we want to gather, which could cost us more time.

Multi-armed bandits try to solve the exploration-exploitation problem by not having two distinct periods of exploration and exploitation. They are adaptive through time, and they include both phases simultaneously. The algorithm learns how to allocate the traffic, avoiding sending cases on lower-performing variants. This can be seen as an efficient way of not loosing too much while learning[2]. We can observe the difference between their approaches in Figure 2.1

[2]Learning is considered as collecting data.

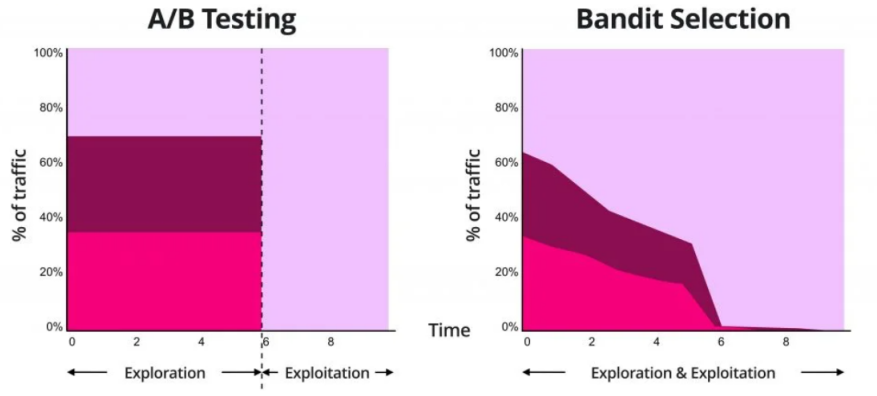

FIGURE 2.1: A/B Testing vs MAB dealing with exploration-exploitation tradeoff. Source: *Minimize A/B Tests losses* 2022

Different multi-armed bandits algorithms and approaches exist to solve the proposed problem. Next, we give a more formal formulation about multi-armed bandits and the different types.

2.2 Algorithm framework and variants

The general multi-armed bandit model has a set number of arms to choose from when deciding. This is a known parameter that we denote as $K > 2$. We also have the number of trials/rounds that the algorithm will have, which we represent as $t = 1, 2, ..., N$ and denotes the length of our dataset. Formally, a contextual-bandit algorithm proceeds in discrete t-trials as follows:

1. The algorithm observes a user u_t and a set of K arms.
2. From that previously observed rewards information, the algorithm chooses an arm and receives its payoff. The algorithm does not get any rewards for unobserved arms.
3. Lastly, given the reward obtained from the user at time t, the algorithm updates its decision-making.

This framework is the most general one, where the main goal is to minimize the regret between the chosen arm and the optimal arm in each trial. However, the approach to the problem of maximizing the payoff of each arm rather than reducing the regret ends up being the same, although the assumptions for the quality of the methods are usually made by analyzing the regret. We define the regret as:

$$R_T = \sum_{t=1}^{T} [\mu^* - \mu_{a,t}] \tag{2.1}$$

Where T is the total trials made, μ^* refers to the expected reward from the optimal arm, and $\mu_{a,t}$ stands for the expected reward obtained from arm a at trial t.

To minimize the regret from Equation 2.1, the algorithm will exploit, given the experience of previous users, the arm that seems to be the best. However, this possible best arm might be suboptimal if the algorithm lacks enough knowledge. This is solved by exploring which might be suboptimal arms with a focus on gathering more information. This is the previously mentioned trade-off dilemma; exploring new components can result in a more significant regret short-term while exploiting a suboptimal arm might reduce the regret short-term but not long-term.

Multi-armed bandits all follow the same idea described before. Still, there are many ways to classify their variations depending on how the bandit problem generates the reward, how the action space is defined, or which is the problem structure. One of the possible ways to classify multi-armed bandit problem is by stochastic or adversarial:

- Adversarial multi-armed bandits is a variant first introduced by Auer et al., 1998. In this type of algorithm, the agent chooses one arm and adversary simultaneously chooses the payoff structure for each of them. One of the most known is Exp3[3], which works by setting a list of weights for each action and using them to decide randomly which action to take next, modifying those weights given the payoff.
- Stochastic multi-armed bandits assume that each action corresponds to an IDD[4] reward. Each step has an underlying distribution from where we sample the reward when the action is selected. This distribution never changes, so the algorithm must explore the arm enough times to bound the reward distribution's shape properly.

Although there are several algorithms to solve the bandit problem, we will focus on the stochastic ones. Concretely, we will study Upper Confidence Bound (UCB) algorithm since this thesis focuses on an algorithm highly related to it.

2.2.1 UCB

Upper Confidence Bound[5] algorithm optimistically tackles the exploration-exploitation dilemma since it considers uncertainty an opportunity. UCB keeps track of the mean reward for all the possible arms up to the present trial. For that given trial, it calculates the upper confidence bound for each component, which indicates how uncertain the algorithm is of the potential of each arm. If a given arm has a very high confidence bound, the algorithm considers it to have a significant exploration opportunity, and therefore, that

[3] Exponential-weight algorithm for Exploration and Exploitation.
[4] IDD means Independent and Identically Distributed.
[5] UCB is often phrased as "optimism in the face of uncertainty".

given arm is selected. We can observe an example from a given trial t with four arms in Figure 2.2.

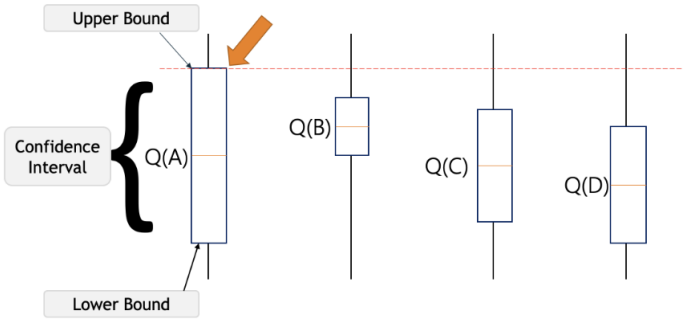

FIGURE 2.2: UCB algorithms best arm selection criteria. Source: *UCB Algorithm in Reinforcement Learning* 2020

In Figure 2.2, we observe a situation where the component "A" has a lower mean reward. Still, it has higher upper confidence bound and would be selected in the UCB algorithm because of this exploration emphasis. The selected upper bound of each arm at each time step A_t is given by Equation 2.2.

$$A_t = \arg\max_a [\mu_a + c\sqrt{\frac{2ln(n)}{n_a}}] \tag{2.2}$$

In Equation 2.2 we observe two main parts. The first one, situated on the left, is μ_a, which refers to the estimated value of arm a at a given time step t. This term represents the exploitation part of the equation. The second half of the equation refers to the exploration part. The first element is the hyperparameter c and controls the degree of exploration the algorithm will have. Then, inside the square root, we have the element that controls how many times a given arm is selected. If an arm has not been tried that much, n_t will be small, increasing the uncertainty and making the arm more likely to be chosen. Whenever an arm is selected, we become more confident about its real estimate μ_a. The logarithm function in the numerator assures that if an arm is not selected, the confidence bound will grow slowly, whereas if it is selected, it will shrink fast due to n_a being linear. The exploration term gradually decreases as time progresses because n goes to infinity. In that case, arms would be selected based only on the exploitation term.

For each of the T trials, we are computing the confidence interval for each arm $c_{t,a}$, so that $|\hat{\mu_{t,a}} - \mu_\alpha| < c_{t,a}$ holds with high probability. When confidence intervals are properly defined, it has been shown that UCB algorithms have a small total T-trial regret, which is the logarithm of the number of trials. This result proves to be favorable by Auer, Cesa-Bianchi, and Fischer, 2002 and by Chu et al., 2011.

So far, we have covered the basics of multi-armed bandits and given an algorithm that solves this problem. However, until now, we considered bandits where we do not have any context from the trial. This means we do not have any information about the user we are passing up to the algorithm. In a real case scenario, this is not always like this since sometimes we have access to some information about the trial before entering the algorithm process. In those cases, we need to use what is called Contextual Bandits. These bandits are more challenging than the context-free ones, and some algorithms have been studied, like Exp4 by Auer et al., 2002, or the epoch-greedy by Langford and Zhang, 2007. However, in this thesis, we will focus on Lin-UCB, which has been studied by Li et al., 2010.

We will deeply analyze Lin-UCB in the next section and apply it to a real-life application: acknowledging which game features/variants to include on a game for a particular user to maximize the revenue obtained or time played on a mobile game.

Chapter 3

Methods

This chapter will introduce the methods used to solve the problem raised at the beginning of the thesis. One of the goals is to acknowledge, from a mobile games company perspective, which game features include in a mobile game to maximize the time users spend playing and the revenue obtained from the game. Since we have a certain amount of game feature variants in a game, this type of problem can be viewed as a multi-armed bandit one. As we introduced before, there are different types of multi-armed algorithms. However, because we have features from users who install the game, we will work with Contextual bandits. Specifically with Lin-UCB, which we will explain next deeply. After that, we introduce the data we will use to experiment with the algorithm. Finally, we set up the experiments and how we will evaluate Lin-UCB.

3.1 Lin-UCB

UCB method provides a strong regret bound and asymptotic optimality, which may invite us to use this type of algorithm for bandits with context. Though, estimating from data the confidence interval of the parameters given by each trial is expensive in general. Lin-UCB provides a way to compute this confidence interval efficiently in closed form, assuming that the expected reward of an arm a is linear with respect to its feature vector. Concretely, we will focus on Lin-UCB with Disjoint Linear Models since the parameters are not shared among different arms. We can define the expected payoff as:

$$E[r_{t,a}|x_{t,a}] = x_{t,a}^T \theta_a^* \tag{3.1}$$

where $x_{t,a}$ refers to a d-dimensional feature of an arm in time t and θ_a^* is an unknown coefficient vector.

Let D_a be a design matrix of dimension m x d at trial t. Its rows correspond to m training inputs (e.g features from the users who install the game), and b_a be a response vector (e.g if the user has paid or not). If samples $(x_{t,a}, r_{t,a})$ are independent, then we can get a closed-form estimator of θ_a^* applying ridge regression:

$$\theta_a^* = (D_a^T D_a + I_d)^- 1 D_a^T c_a, \tag{3.2}$$

where I_d is an identity matrix of dimension d. The accuracy of the estimator will depend on the amount of data we provide. As long as c_a components are independent conditioned on corresponding rows in D_a, it has been shown by Walsh et al., 2012 that with probability at least $1 - \delta$,

$$|x_{t,a}^T \hat{\theta}_a - E[r_{t,a}|x_{t,a}]| \leq \alpha \sqrt{x_{t,a}^T (D_a^T D_a + I_d)^{-1} x_{t,a}} \tag{3.3}$$

for any $\delta > 0$ and $x_{t,a}$ where $\alpha = 1 + \sqrt{ln(2/\delta)/2}$ is a constant. The arm selection strategy from Lin-UCB can be derived from Equation 3.3, since it gives a reasonably tight UCB for the expected payoff of arm a. Thus, at each trial t, if we define $A_a = D_a^T D_a + I_d$, we are going to select an specific a_t arm as follows:

$$a_t = \arg\max_a (x_{t,a}^T \hat{\theta}_a + \alpha \sqrt{x_{t,a}^T A_a^{-1} x_{t,a}} \tag{3.4}$$

As represented in Equation 2.2, we have the left term, which refers to the mean reward estimate of each arm, and the right term, which corresponds to the upper confidence bound where α is a hyperparameter that controls the exploration sensitivity of the algorithm. However, now we have a set of context features for each trial, and we assume that the expected reward of an arm is linear with this feature vector.

3.1.1 Algorithm

Algorithm 1 Lin-UCB with Disjoint Linear Models Algorithm.

Inputs: $\alpha \in \mathbb{R}_+$
for `t = 1, 2, 3, ..., T` **do**
  Observe features of all arms $a \in A_t$: $x_{t,a} \in \mathbb{R}^d$
  for all $a \in A_t$ **do**
    if a is new **then**
      $A_{a_t} \leftarrow I_d$ ▷ d-dimensional identity matrix
      $b_{a_t} \leftarrow 0_{dx1}$ ▷ d-dimensional zero vector
    end if
    $\hat{\theta}_a \leftarrow A_a^{-1} b_a$
    $p_{t,a} \leftarrow \hat{\theta}_a^T + \alpha \sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$
  end for
  Choose arm $a_t = \arg\max_{a \in A_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
  $A_{a_t} \leftarrow A_{a_t} + x_{t,a_t} x_{t,a_t}^T$
  $b_{a_t} \leftarrow b_{a_t} + r_t x_{t,a_t}$
end for

Algorithm 1 gives a detailed follow of the Lin-UCB algorithm. There are two considerations about the algorithm: the first one is its input. We only have α as a hyperparameter, which in some applications its value given by Equation 3.3 might be too large. Therefore, optimizing this parameter will be critical if we want higher payoffs when we transfer the algorithm into our real-case scenario and put it into production. The second thing to mention is how the algorithm selects the highest UCB arm for each trial when having ties between different arms, which sometimes occurs at the beginning. In that case, we choose one arm arbitrarily.

This algorithm has some properties that give it an edge over others. Those properties are the following:

- The first suitable property is its computational complexity, which is linear in the number of arms and at most cubic in the number of features. There are other methods to reduce the complexity, such as feature engineering or updating the weights periodically, although we will not use them in this thesis.
- Second good property of the algorithm is how it is not affected when the number of arms set is dynamic. Although we do not have this case scenario in our example, it is not strange to see how mobile game developers add or remove some game experiences (arms). Thus the content pool might change. As long as the pool size remains more or less constant, this is not a problem.
- Another good property is how given a fixed number of arms, the confidence interval decreases fast enough when we increment the data size. This proves a strong regret bound (Auer, 2002) satisfying Equation 3.1. We will not jump into details, but these theoretical results exalt the efficiency of the algorithm and its overall success.

Next, we will discuss the real-case problem we want to solve using this contextual bandit methodology.

3.2 Mobile-games application

In the game industry, acknowledging how to design a game is key to creating an enjoyable experience for the user. The business model plays a critical factor in the designing department since the game usually needs some revenue mechanics to guarantee its sustainability. Specifically, in mobile games, there are a lot of small features that can be added or not to maximize the quality and the rentability of the game. The progress you get, the number of rewards and periodicity of when you obtain them while playing, the number of ads being shown, or the in-game boosts in the form of purchases are vital factors when determining the user's opinion of a game. These small features might not have the same effect on distinct persons, so it is vital to acknowledge which, how, and when to allocate them into the player's game.

Understanding how, when, and to whom provide a game experience or not can be seen as an exploration-exploitation problem. One of the project's core objectives is to maximize the amount of time a user spends playing the game and maximize the revenue obtained from the user by acknowledging which is the best game experience for him. We consider revenue and game experience approaches because of how related they are when designing a game, although their perspective might be slightly different.

As we have covered in Chapter 2, one way to tackle this type of problem is by considering them as a contextual bandit problem; since we have some context about the users when they download the game, a set of arms in the form of game experiences variants, and a reward to maximize, which can be revenue or time played.

Next, we will introduce the dataset structure and how it will be used to experiment with Lin-UCB and solve the proposed problem.

3.3 Data

The provider of the dataset is Socialpoint, a mobile-games developer company based in Barcelona. Thanks to the collaboration made with them, we have obtained a comprehensive database with different A/B tests performed on mainly three games: "Monster Legends[1]", "Dragon city[2]", and "World Life"[3]. For each of the different A/B Tests "IDs," a set number of game experiences was tested for concurrent and new game users. After some time, metrics were collected, such as time played, in-game purchases, number of sessions, etc.

Some of the different A/B tests from the dataset had more than 1 million users, so the dataset is big enough for our purposes. Besides, we have cases where the A/B test had two, three, or four variants of game experiences, which will allow us to explore how the algorithm behaves when having more or few variants. We can observe all the dimensional features from the dataset in Table 3.1.

[1]Online strategy role-playing game. More than 50 million downloads.
[2]Collectionable videogame with more than 100 million downloads.
[3]Puzzle game with more than 10 million downloads.

Feature name	Type	Description
Game alias	Categorical	Describes the game where A/B test was performed.
AB test ID	Categorical	Expresses the A/B tests ID.
AB test experience	Categorical	Describes which game variant is the user testing.
Country alias	Categorical	Represents country from the user.
Platform alias	Categorical	Platform where the user downloaded the game (Android/iOS).
Source alias	Categorical	Source the user comes from (mobile or tablet).
AB test datetime start	Date	Describes when the user joined the A/B test.
AB test datetime start	Date	Describes the end date of user's A/B test.

TABLE 3.1: Dataset features.

Besides features from Table 3.1, we extracted the "NGU" feature to keep track of the new game users since we considered it essential to differentiate the A/B tests done to new users or already existing ones. Aside from previous features, we had a battery of metrics for all users. For each of those metrics, we had the result after day one, day three, and day seven of starting the A/B test. We can observe those metrics in Table 3.2.

Feature name	Type	Description
Return	Categorical	Describes if the user has returned to play after day 1, 3 or 7.
Time played	Numerical	Expresses the amount of time played up until day 1, 3, or 7.
Ad revenue	Numerical	Refers to the amount of ad revenue obtained up until day 1, 3 or 7.
IAP revenue	Numerical	Describes the revenue obtained from in-game purchases up until day 1, 3, or 7.
Num sessions	Categorical	Expresses the number of sessions made by the user up until day 1, 3 or 7.

TABLE 3.2: Dataset metrics.

Metrics shown in Table 3.2 are available for each of the three different timestamps commented on before. However, day one metrics can be used as features when we evaluate day three or day seven, and day three metrics can be treated as features when we consider day seven. We will explain how to approach this consideration next.

Socialpoint has modified the values of the features and metrics to preserve the privacy of users' data. Specifically, all categorical features' actual labels

have been replaced by numbers, and for the numerical ones, a multiplicative or additive factor was added depending on the feature. This fact has 0 implication for our results, although it might have a less inspiring analysis of the experiments.

3.3.1 Dataset exploration

The extension of the dataset impulsed us only to take some of the A/B test IDs to experiment with. The ones selected are 3582 and 3301 for three main reasons: their extension, they are new-game users oriented, and they have different game-experiences variants.

We can observe in Table 3.3 the distribution of the features for each of the IDs on both datasets.

	2 Arms data		**4 Arms data**			
Features	**ID = 0**	**ID = 1**	**ID = 0**	**ID = 1**	**ID = 2**	**ID = 3**
Country						
Type: 1	57.05%	57.93%	11.72%	11.63%	11.69%	11.74%
Type: 2	27.50%	27.56%	12.11%	12.13%	12.06%	12.02%
Type: 3	15.45%	15.51%	76.17%	76.24%	76.25%	76.24 %
Source						
Type: 1	22.19%	22.13 %	74.87%	74.97%	74.98 %	75.10%
Type: 2	77.81 %	77.87%	25.13%	25.03%	25.02 %	24.90%
Platform						
Type: 1	24.32%	24.48%	80.65%	80.81%	80.73%	80.99%
Type: 2	75.68%	75.52%	19.35%	19.19%	19.27%	19.01%

TABLE 3.3: Datasets features analysis.

As shown in Table 3.3, there are no significant differences in the distribution of the features between the different IDs on both datasets. This means that when the data was collected, they assured to have identical distributions for the different variants, which is suitable for our purposes. Next, we will analyze the metrics in the same way. We can observe the average and the percentage of cases a is 0 (or 1 if it's the minimum value) in Table 3.4.

Socialpoint masked the values, so we will not analyze the actual numbers, just the difference between the IDs. The metrics that refer to the overall user game experience are the first three in Table 3.4. We can observe how for the dataset with two variants, the return rate is 55% while for the four variants dataset is 72% after one day. This tells us how fast users drop a game during the first 24 hours; although they might return after that, we can observe how this percentage keeps going up as the days progresses. From the number of sessions metric, we can watch how it has more presence since you need to have a session to create a user, so the minimum number of sessions is one. We observe 38% and 53% cases of one session in the datasets, which decays as

days go on. Lastly, time played grows as the days go on as well, and here we see differences between the actual values from different IDs on the two-arms dataset.

	2 Arms data		4 Arms data			
Metrics	**ID = 0**	**ID = 1**	**ID = 0**	**ID = 1**	**ID = 2**	**ID = 3**
Return						
Day: 1	0.44 (55%)	0.44 (55%)	0.28 (72%)	0.28 (72%)	0.28 (72%)	0.28 (72%)
Day: 3	0.26 (74%)	0.26 (74%)	0.14 (85%)	0.13 (85%)	0.14 (85%)	0.14 (85%)
Day: 7	0.17 (83%)	0.17 (83%)	0.09 (91%)	0.09 (90%)	0.09 (91%)	0.09 (90%)
Num sessions						
Day: 1	1.35 (38%)	1.34 (38%)	0.68 (53%)	0.68 (53%)	0.69 (52%)	0.68 (53%)
Day: 3	2.10 (34%)	2.09 (35%)	0.90 (47%)	0.90 (47%)	0.91 (47%)	0.90 (47%)
Day: 7	3.07 (33%)	3.04 (33%)	1.18 (43%)	1.19 (43%)	1.19 (43%)	1.18 (43%)
Time played						
Day: 1	102.85 (11%)	104.96 (11%)	54.42 (6%)	54.15 (7%)	54.96 (7%)	53.91 (7%)
Day: 3	145.85 (10%)	152.01 (10%)	73.85 (6%)	74.05 (6%)	74.95 (6%)	73.45 (6%)
Day: 7	196.08 (10%)	203.70 (11%)	99.30 (6%)	100.13 (6%)	101.51 (6%)	98.53 (6%)
Ad revenue						
Day: 1	0.01 (93%)	0.05 (84%)	0.34 (34%)	0.35 (34%)	0.35 (34%)	0.35 (34%)
Day: 3	0.02 (85%)	0.15(73%)	0.42 (32%)	0.44 (32%)	0.44 (32%)	0.44 (32%)
Day: 7	0.05 (79%)	0.25 (70%)	0.52 (31%)	0.55 (31%)	0.54 (31%)	0.54 (31%)
IAP revenue						
Day: 1	0.45 (99%)	0.44 (99%)	0.02 (99%)	0.01 (99%)	0.01 (99%)	0.01 (99%)
Day: 3	0.72 (98%)	0.61 (98%)	0.03 (99%)	0.02 (99%)	0.02 (99%)	0.02 (99%)
Day: 7	0.99 (98%)	0.81 (98%)	0.04 (99%)	0.04 (99%)	0.04 (99%)	0.03 (99%)

TABLE 3.4: Dataset metrics analysis. First value shows the average of a given metric for each correspondent ID and the value within parenthesis indicates the percentage of cases where the metric is 0.

The metrics related to revenue are the last two from Table 3.4. In the ad revenue metric, we observe how most users do not watch an ad on the two-arms dataset. Also, there is a vast difference in the mean value of the two IDs. On the other hand, we observe no difference in the four-arms dataset and a relatively low percentage of 0 ad revenue. Both datasets refer to different games, which might have a diverse ad-showing policy. Lastly, for the IAP revenue feature, we observe how only 1% of the cases spend money after day one for both datasets. However, we see on the two-arms dataset some disparity in the mean values between different IDs past day 3.

From this analysis, we observed the characteristics of the two datasets and the distribution of the features and metrics we considered. We will explain next the baseline methods we are going to use for revenue and user experience maximization.

3.4 Baseline methods

Different rewards will be tested depending on which of the two problems we are considering. We will evaluate day one, three, and seven metrics considering them as rewards for each dataset. However, when considering days three and seven, we will use the previously collected metrics as features to add more context about the user. We believe it's appropriate to compare the differences when dealing with a problem with only two game variants or four to test the algorithm's behavior.

From now on, we will divide the problem into revenue maximization and user experience maximization. Next, we define how we will maximize each of them:

- **Revenue maximization:** There are two types of revenue, ads income and in-game purchases, and we will consider both of them. However, as we have seen in Table 3.4, the number of people purchasing something in-game is very low. For this case, we will need to use more data than in other cases to get a decent result when evaluating the algorithm since it needs to see non-zero rewards. This case is more accentuated when dealing with the four-arm dataset since the complexity grows linearly as we increment the number of arms. On the other hand, since the ad revenue reward is more present through the cases, fewer data will be required.
- **User experience maximization:** From the user experience side, we have three metrics: "Return," "Time played," and "Num sessions." Each feature has some perks and drawbacks when we consider them as rewards. The first one indicates if the user has logged in again to the game, which is a good indicator in mobile videogames[4]. However, this feature is more strict than the others since it has a lower percentage of non-zero rewards. The second feature, time played, provides different information, although it does not tell if the user has played for several days. The last one might be the richest since it tells you how many times the user has logged in. Still, it does not explain to you the amount of time the user spends playing the game[5] and if after the first two days he keeps playing or not. Since these three metrics capture slightly different information, we will experiment with all of them within the two datasets, using more data when dealing with the return metric.

For each of the previously stated combinations of datasets and rewards, we need to adjust the α hyperparameter from Equation 3.4, which will determine how open the algorithm is to explore other arms or to exploit the one that seems to have a bigger mean reward. To evaluate the impact of this parameter on the different combinations of rewards/datasets, we carried out experiments with distinct values of α, which goes from 0.5 to 3.5. In practice,

[4] In the mobile game industry, most of the time, the user never returns to the game again.

[5] There are some players that play casual and just log in a few minutes per day to complete daily or time-gated content.

it has been shown by Burtini, Loeppky, and Lawrence, 2015 that the optimal choice of this value depends on the arm distributions, so trying it out with different values seems the best approach. Thus, we will run the algorithm several times for each combination and look at their mean performance to select the best α value for a given reward and dataset.

Once the best α parameter has been selected for that specific algorithm, we will keep track of different indicators. Those indicators include the times an arm is selected, distribution from all the features involved in the algorithm for each arm, evolution of the arm selection, performance of each arm, and overall performance of the algorithm. We will store these indicators for each time we have run the algorithm when selecting the best α, so we can analyze its behavior when we run it multiple times.

The actual algorithm performance evaluation is not as trivial as it seems since we are using an offline A/B test dataset. This has been studied by Li et al., 2011, and we will cover it next.

3.5 Model evaluation

The main goal is to acknowledge the rule for selecting an arm each time a user enters the system taking into users' context and previous rewards. Although it seems that the only way of evaluating the algorithm unbiasedly is to run it with "live"[6] data, in practice, is not that easy. There are logistical challenges such as extensive engineering resources to implement it with live data and problems with reward collection. If we look up rewards after days three or seven, we need to delay each trial until the prize is collected, making the hypothetical live procedure slow. One way to solve this problem could be using sub batches of users and collecting their reward after the next set of users enters the algorithm. For experimental purposes, we will evaluate it with an offline[7] dataset, which we already described.

The usage of offline data brings assumptions that need to be made. The first is that the A/B test dataset has a defined arm selection for each user. This arm selection policy will likely differ from the one the algorithm would choose, complicating the evaluation. This is a particular case of the "off-policy evaluation problem" in the reinforcement-learning literature covered by Precup, Sutton, and Singh, 2000. However, in the multi-armed problem, there is no need for "temporal credit assignment"[8], which allows us to have more efficient solutions.

The dataset presented meets all the assumptions made to have an unbiased evaluation metric: we have access to a long sequence of logged events where each consists of the context vector (information of the game user), a selected arm (A/B game experience feature), and the reward. We suppose

[6] Information that is delivered immediately after collection.
[7] Obtained from a historical testing data.
[8] Refers to measuring an action's influence on future rewards.

there is an unknown D distribution in which tuples are drawn IID of the form $(x, r_1, ...r_k)$, each consisting of observed context and non-observed payoffs from all possible arms. This approach also considers that the number of arms is fixed, which is the case.

The offline evaluation method can be observed in Algorithm 2. This algorithm takes as input the contextual bandit and a given dataset S of length L. Then, when going through all the different users one by one, if the Lin-UCB algorithm selects the same arm a as the one that was chosen by the logging policy (dataset), we save that user, and the payoff is updated. If that is not the case, then the algorithm completely ignores that user and proceeds to the next one without any change. So, we will not use all the users from the dataset since the probability that an event is retained is linked to the number of arms. Thus, the events retained have the same distribution as if they were selected from D. This means that we are evaluating the algorithm against T real-world events from D and testing it using the evaluator on a stream of logged events. It has been proved by Li et al., 2011 that the output of Algorithm 2 is an accurate estimate of the true per-trial payoff with high probability when the fixed policy chooses an action independent of the history h_{t-1}.

Algorithm 2 Policy evaluator with finite data stream.

Inputs: bandit algorithm A; stream of events S of length L
$h_0 \leftarrow 0$ ▷ An initially empty history
$\hat{G_A} \leftarrow 0$ ▷ An initially zero total payof
$T \leftarrow 0$ ▷ An initially zero counter of valid events
for `t = 1, 2, 3, ..., L` **do**
  Get the t-th event (x, a, r_a) from S
  if $A(h_{t-1}, x) = a$ **then**
    $h_t \leftarrow$ `CONCATENATE`$(h_{t-1}, (x, a, r_a))$
    $\hat{G_A} \leftarrow \hat{G_A} + r_a$
    $T \leftarrow T + 1$
  else
    $h_t \leftarrow h_{t-1}$
  end if
end for
Output: $\hat{G_A}/T$

To obtain an accurate estimate of the total per-trail reward of any contextual bandit, we will provide a repetition of the Lin-UCB algorithm from all the combinations stated before and compute their average. This, added to the discarding users due to offline dataset, will add multiplicatively the computational resources needed, especially for the experiments where the rewards had a low appearance rate, such as in-game purchases.

Chapter 4

Results

In this chapter, we show the experiments' results with Lin-UCB when considering as reward revenue and user experience in two different datasets, one with two-game variants and the other with four. For each of them, we discuss model behavior after optimizing alpha, feature distribution on each arm, arm performance, and algorithm decision-making. The code can be reviewed in the following GitHub repository: Code.

4.1 Revenue maximization

Firstly, we will focus on the revenue-related metrics. We show the results when considering ad revenue or in-game purchases as a metric.

4.1.1 Ad revenue

Two game variants

The obtained results using as reward ad revenue day one are excellent. The mean revenue is 0.029, while the cumulative reward obtained is always higher than 0.036 independently of the α selected. Specifically, $\alpha = 1$ is the one who gets the best result (0.043). This means that the average cumulative ad revenue obtained after T users is higher than the actual reward average. In Figure 4.1, we can observe the mean cumulative reward and arm selection evolution. The algorithm has recognized that arm 1 provides better rewards and tends to select it since the α value is not high. Besides, the algorithm has stable results when running it multiple times, which indicates the result's robustness.

From the features side, we can observe in Figure 4.2 how is their distribution in each arm. We can perceive minor differences between arms on the source and platform features. However, the country element has a massive distribution disparity between arm 0 and arm 1. This means that when the algorithm views a user from "Type 3", it is more likely to be assigned to arm 0. As we can check in Table 3.3, "Type 3" only represented a 15% of the cases on both IDs, which denotes that we are getting valuable information.

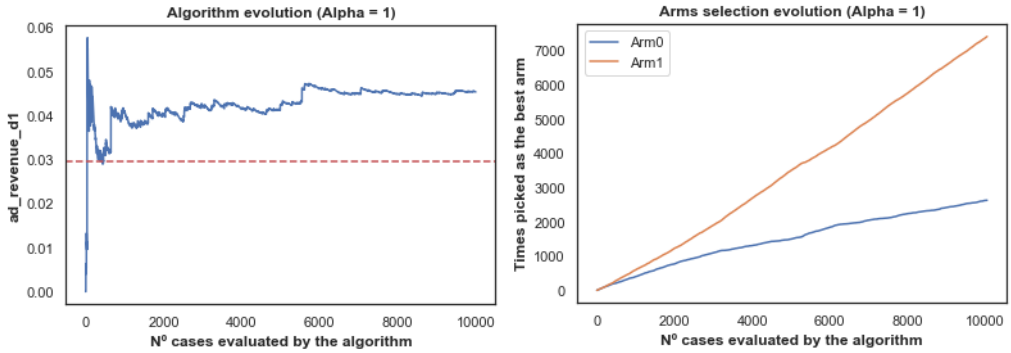

FIGURE 4.1: Algorithm evolution with ad revenue at day 1 as reward. Left plot shows mean cumulative reward evolution and right plot the arm selection evolution (2-arms dataset).

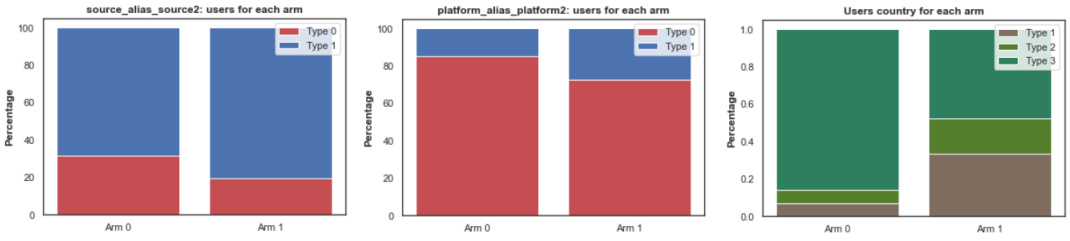

FIGURE 4.2: Features distribution for each arm with ad revenue day 1 as reward (2-arms dataset).

Following, we consider the reward on day three or seven. We also obtain excellent results in those cases where the α is optimized at 1.5 and 0.5, respectively. The cumulative mean reward is 34.03% and 13.68% higher after days three and seven, respectively, than their actual means. We get similar cumulative mean reward evolution as in Figure 4.1, and the results are stable when we run the algorithm multiple times. Features distribution are almost identical as in Figure 4.2. However, there is a crucial difference in the arm selection criteria. Though arm one still achieves a higher mean payoff, the algorithm selects more the other one. We can observe this in Figure 4.3. This is a consequence of including the day one metrics as context features. The algorithm acknowledges more about the user profile and ends up allocating arm one more precisely to maximize the average ad revenue of each variant.

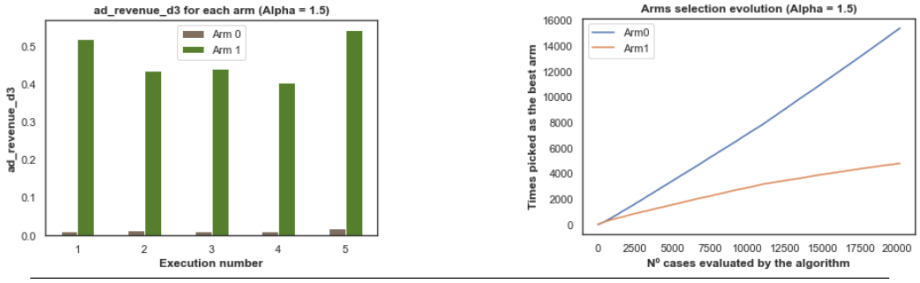

FIGURE 4.3: Algorithm arm payoff and arm selection evolution with iap revenue day 3 or 7 as reward (2-arms dataset).

Four game variants

This dataset had almost equal mean ad revenue among the four-game experience variants. This affected our results since the initial rewards were virtually identical for all the variants. However, the algorithm performed well when the alpha was on a neutral level. Specifically, with $\alpha = 1.5$, the best result was achieved, with a mean cumulative reward of 0.351, surpassing the actual mean of 0.34. This result is worse if we compared it with the previous dataset on day 1, mainly because it has less stable results. We can observe how the algorithm has behaved within three different runs in Figure 4.4. Each column in this figure shows a separate run with the same set α. Though most of the time it gets a result over 0.34, we can observe in the third column how sometimes it does not (plots are cut out at 2000 cases since it is stabilized after that). Specifically, in the second row, each algorithm run had a different arm selection criteria, determining diverse arms selection. The runs that obtain the best results are the ones that use almost a similar way to the four variants, while when it does prioritize one game experience, we get a worse outcome.

From the feature side, we can not analyze that much here since there are diverse interpretations of the arms features on each run. Two things cause this result's disparity: firstly, the dataset, which, as we mentioned before, has the same mean ad revenue for all the different game variants, concluding that the game features variants had nothing to do with ad revenue. The second one is the algorithm itself and analyzing it with offline data. We randomly select the users each time, discarding a 75% of the users (a consequence of having four arms), and the arm's UCB ties are chosen randomly.

When considering day three or seven, we get the same results, where each algorithm run results in slightly different feature distributions within arms. However, we obtain a mean cumulative ad revenue higher than the actual mean. Still, if we compare it with the previous dataset, the results are not that good, possibly because the feature variations have nothing to do with ad policy inside the game.

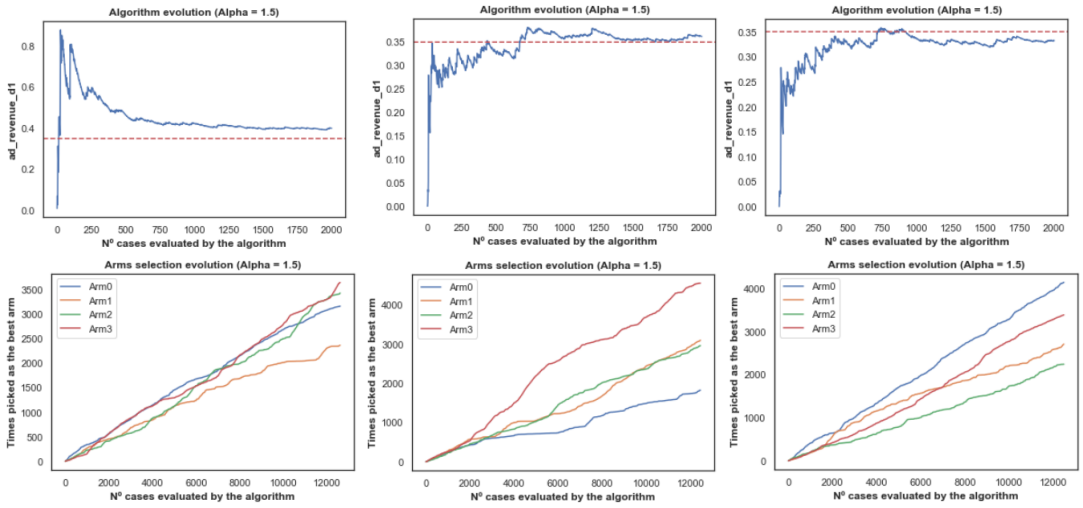

FIGURE 4.4: Algorithm evolution with ad revenue at day 1 as reward. Each column represents a different algorithm run (4-arms dataset).

4.1.2 In-game purchases

In-game purchases have 99% of cases where is 0. This holds for different days and both datasets. However, by incrementing data usage drastically, we have obtained some results.

Two game variants

We obtained good results when considering in-game purchases after day one since almost all the different α parameters surpassed the mean in-game purchases (0.489). Specifically, when selecting $\alpha = 0.5$, we get a mean accumulative reward of 0.56. However, we have a disparity of results when executing it multiple times. This is expected because, from the initial 70000 users used to train, we only end up having 350 users with a non-zero reward. Thus, the algorithm is highly influenced by if the arm selected is the same as the A/B test assigned initially. Still, we got decent results, which we can observe in Figure 4.5. We can spot in the top plots how high peaks refer to when the algorithm visits users with a reward associated.

There is not much to say from the feature analysis side because of the previously mentioned reasons. We have a disparity of profiles in the arms when running the algorithm. However, we do observe some differences between the platform distribution within arms. In Figure 4.6 we can observe how in most cases, one arm always has a percentage of "Type: 0" much higher than the other, although it changes on which arm depending on the algorithm's run.

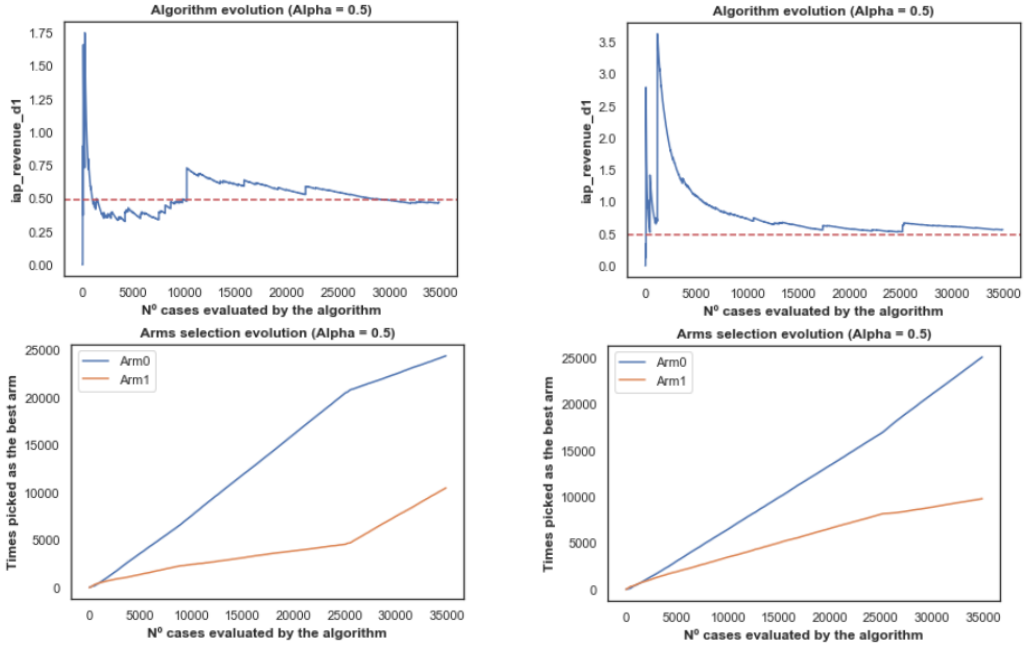

FIGURE 4.5: Algorithm evolution with iap revenue day 1 as reward (2-arms dataset).

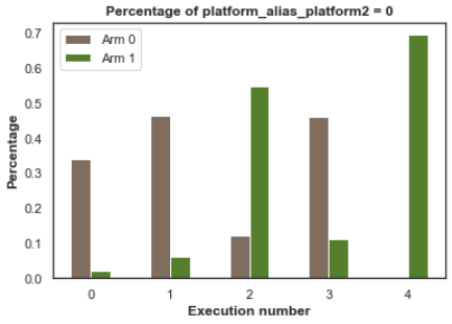

FIGURE 4.6: Platform distribution of users on each arm for different runs with iap revenue day 1 as reward (2-arms dataset).

Results obtained when considering day three or seven are still better than their respective average, but they do not add almost anything new from the previously mentioned. The only difference is that the algorithm determines that arm 0 has a higher mean iap and a higher percentage of the platform "Type: 0". We conclude that this metric is not the best to experiment with this dataset because of its low appearance rate and the no average difference among variants, which drive us to say that the variants had nothing to do with in-game purchases.

Four game variants

We obtained slightly better results with the 4-arms dataset, especially considering days one and seven. For day 1, we determined that the best α is 1.5.

The mean cumulative reward is 0.02, which is a good result since the average in-game purchase is 0.017. Although all arms have been selected almost the same amount of times, arm zero has been the one with better average payoffs. From the features side, there are differences among arms on all features, especially between arm zero and the other ones. All these considerations can be observed in Figure 4.7

The results obtained for day three are not as good as on day one. Although they are still over the average, their results are inconsistent over different runs, and the arm's feature distributions diverge significantly. However, day seven is not the case since they meet all the previous requisites. The cumulative mean reward is higher than expected, and the results are consistent when $\alpha = 0.5$. We obtained similar behavior as day one (Figure 4.7) since arm zero is the one with a higher payoff and just some minor differences in the arm's features distributions.

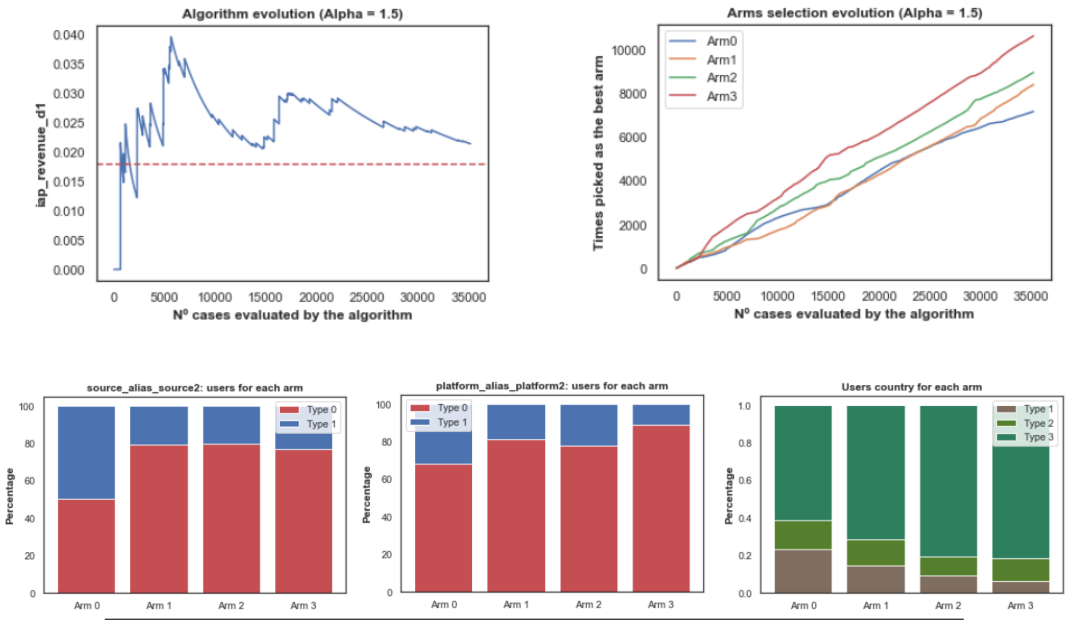

FIGURE 4.7: Algorithm and features analysis iap revenue day 1 as reward (4-arms dataset).

Overall, the results were great. However, we observed crucial differences when treating the datasets. The critical difference is the type of game variants included, while the number of arms is not a determining factor. We have observed that ad revenue is a better metric to consider as a reward since it is more present in the users and allows the algorithm to understand the arm's true expected payoff better.

4.2 Game experience maximization

Results obtained with this approach are not as great as the revenue one. Socialpoint already warned that the type of game variants are small details that might not have much impact on the game experience, especially when looking at short terms like one, three, or seven days. However, we will overview the most important facets of each of the metrics.

4.2.1 Time played

The results obtained after considering the time played on day one are not good enough. The algorithm is slightly higher than the metric average when considering $\alpha = 2.35$, but nor even 1% above it. Besides, multiple runs point out mixed results when considering which is the best arm or the feature distribution. We can observe this in Figure 4.8. Sometimes a game variant is never selected, which we do not want to happen.

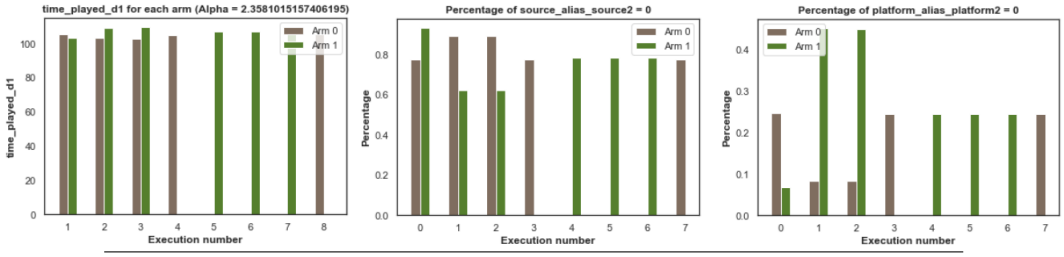

FIGURE 4.8: Arms average return and feature distribution with multiple runs when considering time played day 1 as reward (2-arms dataset).

Although the results for day one are not good, it slightly improves when looking out at day three, and it has good results on day seven, considering the day one metrics as features in both cases. This result explains why day one was not working at all. The user's features when downloading the game do not give enough context, given a game experience, and the day one metrics provide the information needed. So, the day seven cumulative reward is 3% higher than the average time played after day seven ($\alpha = 0.5$). In addition, results and arms features distribution are stable among different algorithm runs. We can observe the complete analysis in Figure 4.9.

Four game variants dataset ends up having similar results. Day one metrics do not work well, and the overall results and stability of the algorithm incrementally improve on days three and seven. These results point out how hard it is for game features to impact the time a user plays the game or not, especially when we analyze it with low delay. Besides, the initial information is insufficient, and we need to have some user metrics on the first day to obtain decent results.

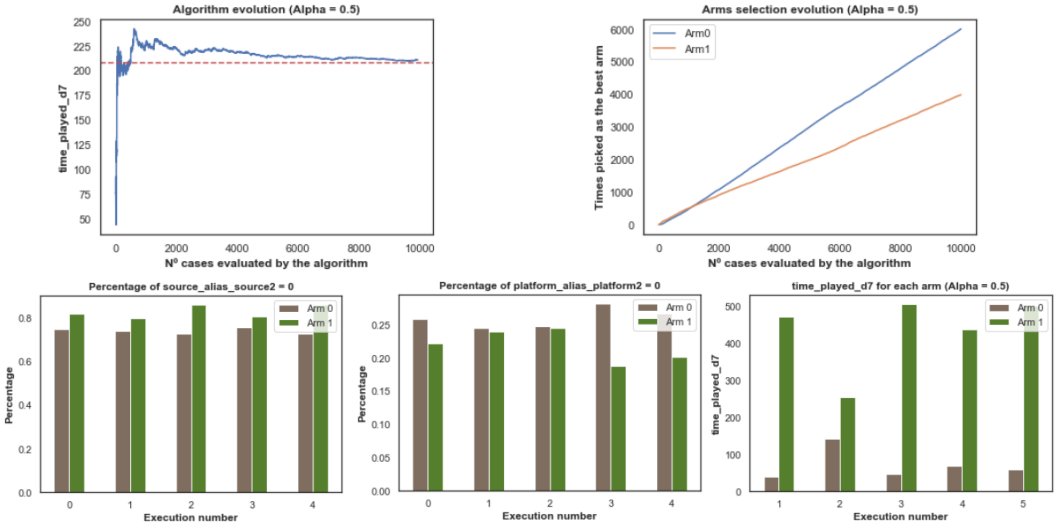

FIGURE 4.9: Algorithm and features analysis time played day 7 as reward (2-arms dataset).

4.2.2 Number of sessions

The overall results obtained with this metric as a reward are very similar to those obtained with the time played one. Day one does not work that well, and day seven experiments perform the best and have more stability among different runs. We have a relatively high $\alpha = 3$, and the arms have shifted their behavior compared with the previous metric. We observe it in Figure 4.10. Game variant zero is the one with the higher average number of sessions, although it is not the most selected. We already saw this effect when incorporating the metrics of day one as context features. Thus, the arm selection, the average payoff of each arm, and platform feature have a completely different distribution than in the time played (Figure 4.8). This result is interesting because it shows how time played and several sessions display different types of information even tho they try to achieve the same.

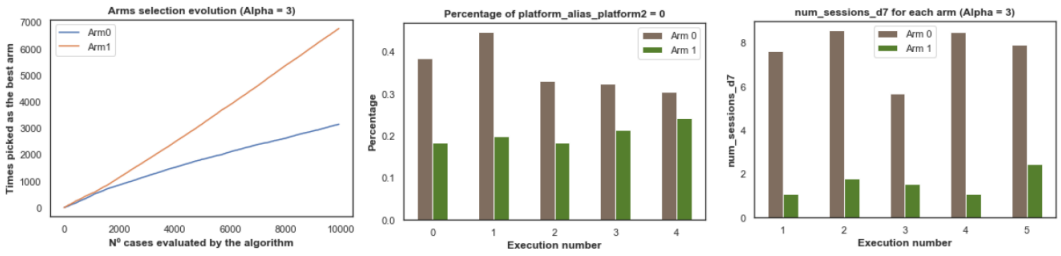

FIGURE 4.10: Algorithm and features analysis number sessions day 7 as reward (2-arms dataset).

Four variant game results show virtually the same described in the previous dataset. The overall results are slightly better, but the algorithm is a bit less consistent through different runs, probably because the number of

arms is higher. We suspect that the game variants are insufficient to affect the metric enough. However, day one results are slightly better than before, although the algorithm shows a lot of shifts within the arm selection evolution since we have a high $\alpha = 2.48$. In comparison, day seven has a lower alpha, and the arm selection criteria is more stable (Figure 4.11). When the α value is higher, the algorithm has more emphasis on exploring new arms.

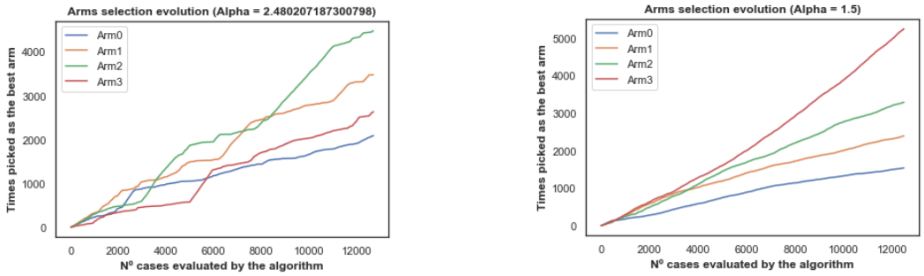

FIGURE 4.11: Alpha effect comparison with number of sessions as reward. Left plot shows day one and right one day seven (4-arms dataset)

4.2.3 Returned to the game

Results with this metric are the worse by far. The algorithm performed slightly above the mean return on day one with the best α. However, results are not consistent through different runs. For days three and seven, we are below the average. We can observe it in Figure 4.12

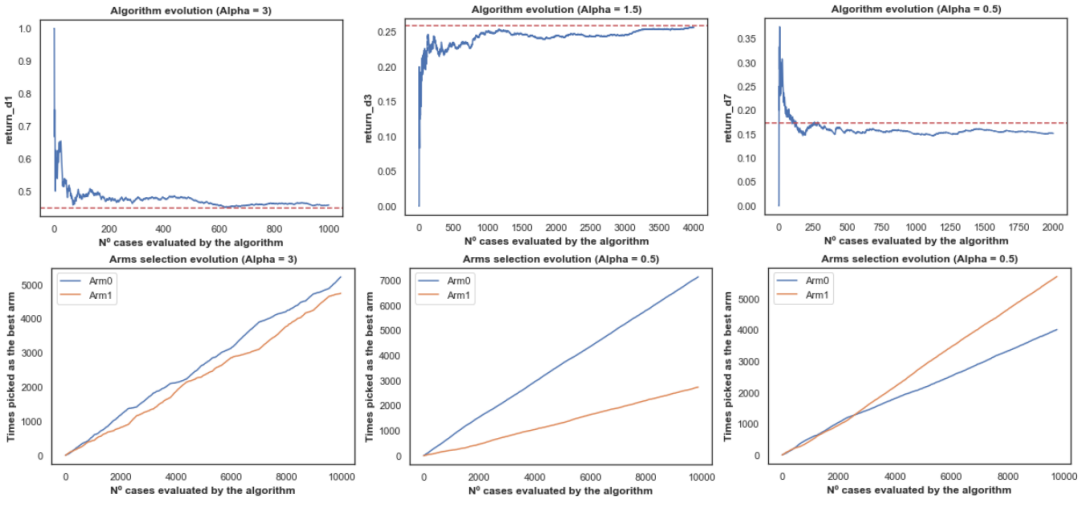

FIGURE 4.12: Algorithm performance and arm selection. First column shows day 1, second day 3, and third day 7 with return to game as reward (2-arms dataset).

On the other hand, results with the second dataset are slightly better. We are above the average for the three different timestamps. Day seven results

are the most consistent ones on the cumulative reward and feature distribution from the arm perspectives. There are no vast differences between features and metrics distributions across arms. Still, that extra information plus having six days delay[1] results in better performance. We can observe day seven algorithm performance in Figure 4.13.

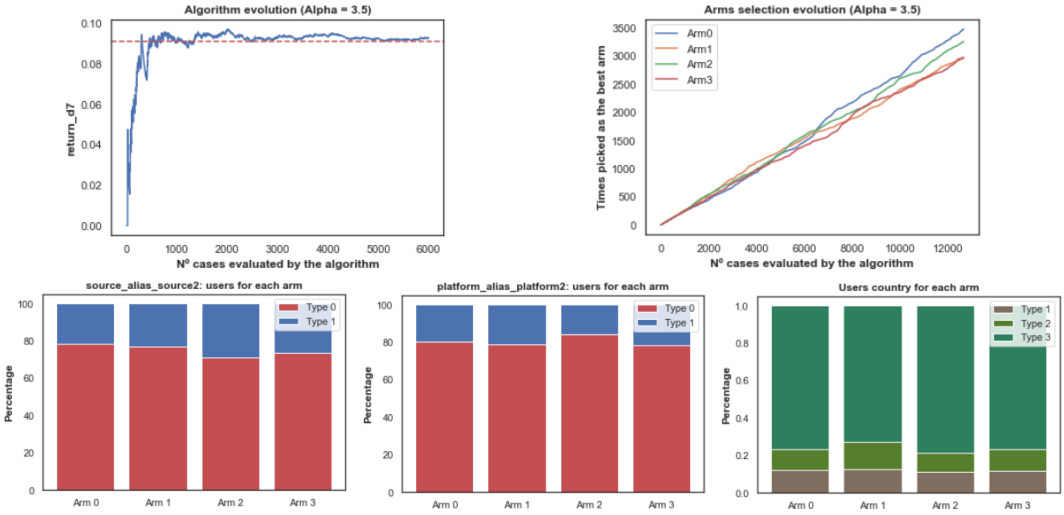

FIGURE 4.13: Algorithm and features analysis return day 7 as reward (4-arms dataset).

So far, we have seen that the return metric is the hardest one to consider for our purposes since, as already warned by Socialpoint, the game variants are minor, and it is hard to observe a significant change in in-game experiences. They told us as well that the return metric is usually the one that is harder to achieve since retaining the user after several days is the hardest to achieve. Thus, we consider this the reason why the method is not achieving good results with this metric.

[1] If we consider day one metric as a reward, technically the day seven analysis has delay=6.

Chapter 5

Conclusions

This thesis deals with the exploitation-exploration dilemma from a mobile games company's perspective. The main goal was to acknowledge how to allocate different game features to the users to maximize the revenue or the user game experience. To tackle this problem, we proposed a contextual bandit approach, specifically the Lin-UCB algorithm. This method consists of having a policy that sequentially learns how to allocate the game variants to the users. This policy selects the game variant with higher upper confidence bound and observes the corresponding reward to update its selection criteria to maximize the average arms payoff. Thus, Lin-UCB provided us a better way to face an experimental process of deciding which game feature to include for a particular user in a more optimally and smoothly way than the traditional A/B tests.

For the analysis of the Lin-UCB algorithm, we used a real-case dataset provided by Socialpoint, a mobile games company based in Barcelona. This dataset included several historical A/B tests where different game variants were tested on the users, and some metrics were captured after one, three, and seven days. These metrics, such as time played, number of sessions, ad revenue, or in-game purchases, allowed us to test the algorithm from an offline perspective. The volume of the data allowed us to perform this evaluation, considering the number of lost cases we dealt with when the algorithm's best game variant selection was not the one tested originally in the dataset.

The experiment obtained overall great results, although there is a battery of things to consider. First, the algorithm gets much better results if the game variants are associated with the metric we are evaluating. When we considered the ad revenue as a reward, we observed how the algorithm gave outstanding results if the game variant was related. However, when dealing with a dataset with no difference between the variant's metric distribution, the algorithm had just slightly better results than the actual average and, in some cases, disparity of results when running it multiple times. In the second place, the metric we evaluate needs to be present. For instance, 99% of the users do not make in-game purchases, making it hard for the algorithm to achieve great results since the percentage of cases where it observes a reward is too low. In the third place, we acknowledged that for the user experience metrics such as returned to play, number of sessions, or time played, it is hard

for the algorithm to perform great after only one day. Actually, it obtained much better results on days three and seven when considering the day one metrics as context features. This shows that the initial context features, combined with the rewards feedback, did not provide enough information to the algorithm. Besides, Socialpoint informed that the game variants usually are small details, making it hard to affect the actual user game experience, especially on the return to play metric. This external information matches with the results obtained on the user-experience approach.

The achieved results indicate the viability of the putting into production the algorithm. Although we would have a one, two, or six days delay in live data between each user, we can solve this problem by using batches of users. Thus, we would only wait to obtain the reward after each batch goes through the algorithm, drastically reducing time waits. To guarantee the success of the implementation, we need to meet the three requirements mentioned before: game variants related to the metric, not too high non-zero cases of the metric, and some extra context when dealing with the game-experience approach. Besides, some prior intuition of the effect of the variant needs to be made to select an appropriate α value, which will determine how open the algorithm is to explore the game variants or exploit the one that seems to be the best. If a game variant is expected to have a minor effect on the metric, a higher alpha value will be recommended; otherwise, a small one will work out superior.

Even though the main goal has been accomplished, some further work can be made to expand the project or tackle the problem differently. For instance, explore other algorithms and compare them when dealing with different rewards or types of game variants. Another roadmap is solving the mobile-game case with other Reinforcement Learning methods.

Bibliography

Auer, Peter (Jan. 2002). "Using Confidence Bounds for Exploitation-Exploration Trade-offs." In: *Journal of Machine Learning Research* 3, pp. 397–422. DOI: 10.1162/153244303321897663.

Auer, Peter, Nicolò Cesa-Bianchi, and Paul Fischer (May 2002). "Finite-time Analysis of the Multiarmed Bandit Problem". In: *Machine Learning* 47, pp. 235–256. DOI: 10.1023/A:1013689704352.

Auer, Peter et al. (July 1998). "Gambling in a rigged casino: The adversarial multi-armed bandit problem". In: *Foundations of Computer Science, 1975., 16th Annual Symposium on*. DOI: 10.1109/SFCS.1995.492488.

Auer, Peter et al. (2002). "The Nonstochastic Multiarmed Bandit Problem". In: *SIAM Journal on Computing* 32.1, pp. 48–77. DOI: 10.1137/S0097539701398375. eprint: https://doi.org/10.1137/S0097539701398375. URL: https://doi.org/10.1137/S0097539701398375.

Burtini, Giuseppe, Jason Loeppky, and Ramon Lawrence (2015). *A Survey of Online Experiment Design with the Stochastic Multi-Armed Bandit*. DOI: 10.48550/ARXIV.1510.00757. URL: https://arxiv.org/abs/1510.00757.

Chu, Wei et al. (2011). "Contextual Bandits with Linear Payoff Functions". In: Proceedings of Machine Learning Research 15. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík, pp. 208–214. URL: https://proceedings.mlr.press/v15/chu11a.html.

Even-Dar, Eyal, Shie Mannor, and Yishay Mansour (2006). "Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems". In: *Journal of Machine Learning Research* 7. URL: http://www.jmlr.org/papers/volume7/evendar06a/evendar06a.pdf.

Lai, T.L and Herbert Robbins (1985). "Asymptotically efficient adaptive allocation rules". In: *Advances in Applied Mathematics* 6.1, pp. 4–22. ISSN: 0196-8858. DOI: https://doi.org/10.1016/0196-8858(85)90002-8. URL: https://www.sciencedirect.com/science/article/pii/0196885885900028.

Langford, John and Tong Zhang (Jan. 2007). "The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information." In.

Li, Lihong et al. (Feb. 2010). "A Contextual-Bandit Approach to Personalized News Article Recommendation". In: *Computing Research Repository - CORR*. DOI: 10.1145/1772690.1772758.

Li, Lihong et al. (2011). "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms". In: *Proceedings of the fourth ACM international conference on Web search and data mining - WSDM '11*. ACM Press. DOI: 10.1145/1935826.1935878. URL: https://doi.org/10.1145%2F1935826.1935878.

March., James G (1991). "Exploration and exploitation in organizational learning". In: *Organization science*, pp. 71–87. URL: http://link.aip.org/link/?RSI/69/1236/1.

Minimize A/B Tests losses (2022). https://vwo.com/blog/multi-armed-bandit-algorithm/. Accessed: 2022-06-15.

Precup, Doina, Richard S. Sutton, and Satinder P. Singh (2000). "Eligibility Traces for Off-Policy Policy Evaluation". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*. San Francisco, CA, USA: Morgan Kauffman, pp. 759–766. ISBN: 1-55860-707-2. URL: http://www.cs.ualberta.ca/~papersdb/uploaded_files/467/paper_PSS-00.pdf.

UCB Algorithm in Reinforcement Learning (2020). https://www.geeksforgeeks.org/upper-confidence-bound-algorithm-in-reinforcement-learning/. Accessed: 2022-06-15.

Walsh, Thomas J. et al. (2012). "Exploring compact reinforcement-learning representations with linear regression". In: DOI: 10.48550/ARXIV.1205.2606. URL: https://arxiv.org/abs/1205.2606.