Year 2008-2009.

Research Internship Report

Machine Learning tools for Online Advertisement

Victor Gabillon
victorgabillon@inria.fr

Supervisors INRIA:
Philippe Preux
Jérémie Mary

Supervisors Orange Labs:
Fabrice Clérot
Stéphane Sénécal

Mission carried out in INRIA Lille - Nord Europe.
6 April 2009 - 6 September 2009.
Abstract

On-line advertisement is the main income source on the Internet. In the last years, this recent market began to switch from pay per impression to pay per click contracts. The ability to present to the web-visitors advertisements they are likely to click on has now become momentous. Ad servers, such has Orange, anticipate this change by an R&D effort. In this context, we will present a review of solutions born in the field of machine learning to tackle those issues. Moreover, trying to put the existing ideas all together, we finally present and analyse a new model.

Keywords : Clustering, Multi-armed bandits, Linear programming, On-line optimization.
Acknowledgements

Thanks to Jérémie Mary and Philippe Preux for their constant support during this stimulant introduction to the world of R&D.

Thanks to Fabrice Clérot and Stéphane Sénécal, our colleagues from Orange Labs for having launch this collaboration and their precious guidances.

Thanks to Jean-Pierre Delmas who has accepted to be my ”Study Advisor” for this internship, but who has been one long before that.

Thanks to Rémi Munos and Sébastien Bubeck for their fruitful help.

Thanks to Daniil Ryabko, Manuel Loth, Alessandro Lazaric, Rémi Coulom, Mohammad Ghavamzadeh, Emmanuel Daucet, Odalric Maillard and Emmanuel Duflos for their advices and support.
Contents

Introduction 8

Definition of the problem 10

2.1 Context ......................................................... 10

2.2 Modeling ...................................................... 11

2.3 Tools .......................................................... 13

Review of existing models 15

3.4 Taxonomy ....................................................... 15

3.4.1 Collaborative Filtering ...................................... 15

3.4.2 Clustering ................................................... 15

3.5 Multi-armed bandit .......................................... 18

3.5.1 Bayesian Bandit ........................................... 19

3.5.2 Multi-armed bandit with side information .............. 20

3.5.3 On-line linear optimization ............................... 22

3.6 Linear Programming .......................................... 22

3.6.1 Deterministic Linear Programming ...................... 23
3.6.2 Stochastic Linear Programming ........................................... 24
3.6.3 Model driven optimization ................................................. 25

Building our own model ......................................................... 26

4.7 Aggregating ideas ............................................................. 26
4.7.1 The relevance of linear programming ................................. 26
4.7.2 Mixing bandits and linear programming ............................. 27
4.7.3 Methods ................................................................. 29

4.8 Playing sequentially ......................................................... 34
4.8.1 What horizon for our planning? ....................................... 34

4.9 Mortality ................................................................. 37

Conclusion and prospects ....................................................... 41
List of Figures

2.1 Historical revenue of Internet advertising in the US. 

2.2 Interactions between the actors of on-line advertisement

2.3 A web banner

3.4 Clustering with Dirichlet Processes. The role of the $\alpha$ parameter

3.5 Multi-armed bandits. The case of very low income probabilities

3.6 Linear programming array representation

4.7 Sensitivity to the frequency of the schedule

4.8 Contextualization of our Model

4.9 Comparison of policies on the Bandit Linear programming problem

4.10 Role of the Horizon in a dynamic ad environment.

4.11 Scheme of ads arrival and lifetime.

4.12 Role of the Horizon in a dynamic mortal ad environment
France Télécom is the third-largest telecommunication company in Europe whose ad serving activity reaches 25 million Internet users. Yet, this activity is still managed in a semi-automatic way with handmade rules. In 2009, a research contract was signed with INRIA (French National Institute for Research Computer Science and Control) to carry out an analysis on machine learning methods to tackle this problem.

My work as an intern, was to realize a bibliographical survey on existing works in this field of applications. Still, on-line advertising being one of the most lucrative business on the web, advertising companies are not willing to share the golden goose. For instance, Google has a very strict secrecy policy and Yahoo! who published theoretical papers on the subject (some of them will be reviewed on this report), is far from revealing all its tricks. Consequently we were brought to design our own ideas and model.

The idea was not to construct a complete software solution, but rather to identify the main features of the problem, and perform an analysis on the sensibility to the parameters in action. Thus our model was complicated little by little to capture more and more intuition about the realistic implementation issues.

This modelling is based on the information France Télécom possesses about its web-visitors and advertisement campaigns as well as on the insight of Orange advertisers. From those, we noticed the main obstacles we had to overcome. One is the lack of precise information about the customers and the ads. Another is the critically low click rate of the ads on the Internet. In such a context, this problem seems hard to solve. How can we predict one customer behavior on a given ad? Should we infer it from previous gestures observed from other Internet users? Should we cluster the visitors into groups of similar persons? But then the response of the visitors to the different commercials have to be learned while playing. Should we stick to what we think is best or should we try to explore and perhaps find better elsewhere? Those questions are very classical issues in the machine learning domain. And thus we will rely on strong techniques to tackle those problems. But still, our problem contains some particularities which makes it challenging. Indeed, in our formulation of the problem the resources are at the same time limited and infinite. Limited because each ad comes with its own budget (the contracts stipulate an expected number of clicks) and that they are temporary (the contracts stipulate a display period). And infinite because new ads will arrive constantly. The budget constraint creates dependencies which are solved through scheduling. But to be really optimal this planning has to take into account the uncertain arrival of
new commercials. So our policy, to be as efficient as possible, has to manage at the same time the present limited resources according to its prediction of the upcoming resources.

Hence, our study will muster subjects like clustering (to infer customers clicks), multi-armed bandits (for the exploration versus exploitation dilemma), linear programming (to plan the allocation of the budgeted ads) and on-line optimization (to face the new ads). And our final goal will be to exploit and combine all of them.

The framework of this report tries to lead smoothly the reader to the construction of our model. First, the problem will be more precisely formulated. The main difficulties raised in this introduction will be addressed referring to different existing solutions and analysing their ability to tackle our problem. Finally we will construct our own model and study its performances.

This report will also be an occasion to discuss how this internship has fit my desire to perform applied research. Indeed, in my curriculum I tried to acquire both engineering skills and more theoretical knowledge. In TELECOM SudParis, I was introduced to the information technology, developing my abilities in programming and networks. During my master in ENS Cachan, I discovered applied mathematics through the scope of machine learning. Thus this internship permitted me to merge those two facets trying to bring new mathematical solutions to a challenging company issue.

Still, even though those facets are complementary, i realised how difficult it could be to try to re-enter thr academic world when comming from the an engineering school. Indeed, one has to learn to become autonomous.
Definition of the problem

2.1 Context

Advertising revenue on the Internet are growing bigger and bigger reaching $23 billion in 2008 according to the 2008 IAB (Internet Advertising Bureau) & PricewaterhouseCoopers survey. As advertisers are getting convinced of the efficiency of on-line campaigns, a large amount of publicity effort is moved from classical media (TV, newspapers...) to Internet marketing.

Let us first describe how on-line advertisement works. A company owns a website and wants to earn money by adding ad banners on it. Most of the time this company will not manage this operation on its own and will call an ad serving company which takes charge of collecting ad campaigns, connecting its server to the client websites and supply them with ads. The advertiser pay the ad server company to distribute their ads and the ad serving company rent space in the client website. So the goal of the ad servers is to collaborate with as many websites as possible in order to propose a large scope of audience to the advertisers and then to fulfill as many contracts as possible. However, there exist different revenue models for the ad serving company. The two main ones are the following:

Figure 2.1: Historical revenue of Internet advertising in the US from 1997 to 2008. This graphic is an extract of the 2008 IAB Internet Advertising Revenue Report.
2. DEFINITION OF THE PROBLEM

2.2. MODELING

Cost per Mile (CPM): The advertiser pays for the exposure of its message to a specific audience. Per “mile” means that a certain price is fixed for a thousand of impressions.

Cost per Click (CPC): The advertiser pays each time an Internet user click on the message and is directed to the advertiser website.

So, the CPM model is more attractive to the ad server which does not have to care about displaying the ads that might interest the visitors. On the contrary, the second model is preferable for the advertiser who has the guarantees that he pays only for successful impressions of its ads. The CPM is historically the first model and it is still prevailing, used for example in context where people almost never click (web-mail, home-pages...). But, even if the question of choosing the relevant model is still addressed, the situation is evolving and, in particular in France Telecom, CPC is constantly growing.

2.2 Modeling

The CPC context urges the ad server to develop accurate tools to predict the behavior of the website visitors in order to display to him ads he would be likely to click on. That’s how a collaboration was initiated between the Orange Labs and the Sequel Team form INRIA. The first part of this partnership will end in November with the writing of a special report on the present researches. During the first meetings, Fabrice Clérot and Stéphane Sénecal presented how the Orange ad server operates. Until now, those servers are directly managed by hand by people in charge of choosing the ads on the websites. Whereas, in the CPC model, a automatic software could perhaps improve the efficiency of the allocations. Such a software could perform different related tasks: place the ads, recommend items to the buyers, and optimize the price of the commercials. Then our work was to make a survey on existing methods to tackle such problems. The idea would be to compare them and to question their scalability and sensitivity to large data basis.
Still, as we went along the meetings we decided to restrict our study to the up and coming CPC model. Indeed, this would permit Orange to be ahead of this changing market and to profit from a fully automatic solution. The difficulties associated with this model are the following:

**Rarity of clicks:** The very first difficulty is that people rarely clicks on ads. The order of magnitude is $10^{-4}$ clicks per page viewed. Therefore extracting information becomes a tough goal especially if we try to keep a personalised client follow-up. But, in compensation, the number of visitors on a web pages is considerable (they are millions of them). So, one could try to predict the behavior of one visitor based on those of visitors having similar profiles. This ideas will be developed in section 3.4 which spots on clustering methods and collaborative filtering.

**The lack of information about ads and profiles:** The information available at Orange about the web-visitors are typically 3 binary variables about their age (young / senior), gender (man / woman), socioprofessional group (− / +). But usually this information is only available for 20% of the visitors. Moreover the taxonomy of the ads is very poor. Still, to try to work it out we have access to a keyword indicating the segment of the website the Internet user is visiting.

**The access to real data:** Unfortunately, during this internship, we did not have access to real data from Orange. Indeed, a lot of authorizations has to be collected in order to make such transfer. Consequently we played with an appetite model made by hand by our Orange supervisors.

Moreover a significant number of simplifications were made to devote ourselves to the main characteristics of the problem. Web-pages usually possess several spaces to display ads on a single web-page. Here, we will only study the case where only one single ad can be displayed at a time. Thus questions about the attention decay due to the position on the web-page are brushed aside. We did not take into account the possible constraints imposed by the advertisers on the time slot. Neither did we try to avoid displaying the same ad twice to the same visitor. Words used on the search engine to access the website are ignored... All those elements could be part of further works and extensions of our present study.

So let us clearly lay what are the basic settings of our toy problem defined with Orange.

The web-visitors are characterized by 3 binary variables about their age (young / senior), gender (man / woman), socioprofessional group (− / +). Each of these information is only available for 20% of the visitors. So we can meet a visitor with \{Man, Young, -\} profile or with \{Man, ?, -\}. Each of these combinations defines a different profile.
There are 6 categories of web-pages: Home page, Automobile, Health, News, Traveling, Run of Site. The number of people from each category of profile which visit each day each page are known.

There are $K$ Ads in a pool on which we have no taxonomy information. So it is by playing them that we will acquire knowledge. Still each ad comes with a predefined lifetime and a maximal number of clicks. For instance, an advertising campaign could ask for 1000 clicks before the $10^{th}$ of October. Plus, most of the time, advertisers would like their ads to be clicked uniformly during the length of the campaign. Typically the number of ads to choose from is, in the case of Orange, around 20.

To create this toy model we have a generative model of ads. Each new ad has its own hidden characteristics. For example an ad $A$ has \{Man, Health\} characteristics. We define the base click probability as $10^{-4}$. But, each time a characteristic of an ad is found in the web-page it is placed or in the visitor it faces, this base probability is multiplied by 4. For instance, if ad $A$ is displayed in Traveling to a \{Man, Young, -\} then the click probability is $4 \times 10^{-4}$. For a \{Man, Health, Young, -\}, the probability reaches $16 \times 10^{-4}$.

The visitors of the day come sequentially and visit the pages according to known proportions. At each time step, a person $x$ arrives. His profile is chosen randomly by the environment with the probability of this person to belong to one profile group proportional to the number of persons in this profile group ($p(x \in profile_i) = \frac{Nbprofile_i}{NbT otal}$). In this context we choose which ad to display and note if a click has occurred or not. Our goal being to maximize the number of clicks by unit of time.

So this problem is an on-line problem where we have to constantly discover the characteristics of the new ads. On the contrary, if we have possessed a taxonomy on the ads and if each ad came with its taxonomy label, the problem would turn in an off-line mode. Indeed, a lot of information could be inferred from the data basis of historical logs. For the Orange problem such information is sometimes available but according to our Orange supervisors they are “50% false” so they could be included in our model as a prior knowledge associated with a confidence term.

2.3 Tools

Let’s recall our goal. Our goal is to maximize the number of clicks on the ads. In order to achieve that, we have to infer the relation between each ad and the profile features. Here we will quickly introduce the different existing tools that permit to tackle this problem.

- We cannot have a personalized treatment which would only look on one customer’s attribute to infer where he will click. Because, we don’t have to much information. Because we don’t see each visitor for a long time. Because they click so very rarely. Hence, data from a lot of similar profiles have to be gathered to extract global information about the group.
• This is an on-line problem where we simultaneously have to play according to our current knowledge and discover the features of the new ads. This dilemma is a well-known trade-off called “exploration versus exploitation” and which will be addressed with the multi-armed bandit setting.

• On last detail which could change it all is the limited budget of each ad. The optimal policy in some cases is no longer to choose the best ad for each customer. Indeed, now with the limited budget the customer are “fighting” for a restricted resource. In the worse case, one customer could waste the budget of one ad for himself alone urging other to make do with critically uninteresting commercials. And this could shrink the overall benefit. Then, a solution is to optimally schedule our display policy using linear programming.

• But linear program is limited to a fixed number of visitors and ads. Then how could it be used when it has to face a never ending traffic of visitors and new ads?
Review of existing models

3.4 A Taxonomy for advertisements and website visitors

We don’t see the web-visitors for a very long time so it can be hard to predict rapidly the behavior of one particular visitors on every ads. That’s why we can use information of siblings visitors.

3.4.1 Collaborative Filtering

Collaborative filtering (CF) is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Here it could be used to infer the click probabilities of one visitor based on its history and on other visitors having a similar history and characteristics. So such a method necessitates a large amount of computation because this process has to be made each time a visitor visits a new page. The Recommender Systems are based on this principles and used in Internet (like in Amazon.com) to recommend a article to a client.

Algorithms for collaborative recommendations can be grouped into two general classes: memory-based (or heuristic-based) and model-based. Memory-based algorithms essentially are heuristics that make click predictions based on the entire collection of previously clicked ads by the other visitors. Most of the time a metric as to be defined to measure the similarity of two visitors. In contrast to memory-based methods, model-based algorithms use the collection of clicks to learn a model, which is then used to make predictions. For a more detailed review see [3].

Most of the time those systems don’t perform any exploration of the other possibilities than the one given by the recommender. Moreover they do not consider the limited budget of the ads.

3.4.2 Clustering

Clustering our inputs, i.e. gathering inputs similar in some sense, reduces drastically the number of actors in the problem. We are no longer dealing with millions of visitors but with hundreds
(for the most) of groups of akin clients. Sometimes, this classification arises naturally from the
data available (this visitor is a man, young and “dynamic”), sometimes a model has to be built
artificially to face a lack of data or dubious information. But, in return, this simplification results
itself in a loss of information.

Then, we have to create a metric to compute the likeliness of two data. Here, one simple
approach is to compare the visitors regarding their behavior on the set of ads and conversely to
cluster the ads based on how the different types of visitors reacted to them. The asset, here, is
that it permits us to cluster our data with the real world correlations, a process that we intuitively
suppose more accurate than just relying our prediction on external features.

This is an old and rummaged problem which has led to flurries of methods. Among them,
the most well-known are the k-means, spectral clustering, Dirichlet processes mixtures... During
our study, we focused on methods to be applied on our on-line problem. Thus Dirichlet process
mixtures were tested out. Indeed, they propose a non parametric method which viewing clustering
as a countably infinite mixture. So, contrary to k-means, it does not require foreknowledge of the
number of clusters. And their recourse to Markov chain sampling makes them shaped for on-line
acquiring data problem whereas spectral clustering would need to perform successive eigenvalue
decomposition on a huge and growing matrix.

Dirichlet Process Mixture Model

Let us first get back to the Dirichlet distribution. Defined on an open $K-1$ dimensional simplex, it
can be seen as the conjugate prior\footnote{See the box on Conjugate prior for more...} of the categorical distribution and the multinomial distribution
thus giving the posterior probability distribution of K discrete rival events (So a distribution over
K discrete distributions). Intuitively, we jump to the Dirichlet processes\footnote{\cite{29} for a good introduction to Dirichlet Processes} by taking K to infinity
resulting in a distribution over distributions. Moreover, realizations of the Dirichlet process are
countably infinite mixtures. This discrete form which comes with some really interesting properties
will permit us to model our clusters. Here comes the basic model :

$$
y_i \mid \theta_i \sim F(\theta_i) \quad (3.1)
$$

$$
\theta_i \mid G \sim G \quad (3.2)
$$

$$
G \sim DP(G_0, \alpha) \quad (3.3)
$$

Taking it from the bottom to the top : In 3.3, $G$ is sampled from the Dirichlet Process. $G_0$ is
the base distribution, basically the mean of the process. And the scalar parameter $\alpha$ controls the
variance which increases conversely to $\alpha$. $G$, the “countably infinite mixture”, then serves in 3.2 to
sample the parameters of the clusters. And finally, in 3.1, the data $y_i$ are sampled from their own
cluster.

So in our model, $F$, $G_0$ and $\alpha$ have to be carefully chosen so that the model fits the data as
well as possible. Our goal is to infer the parameters $\theta_i$ from the data $y_i$. But trying to compute the
posterior is infeasible when there are more than few data. One efficient solution is to use the Markov
3. REVIEW OF EXISTING MODEL

3.4. TAXONOMY

Figure 3.4: Clustering with Dirichlet Processes. The role of the $\alpha$ parameter. $\alpha = 0.1$ (1), $\alpha = 5$ (2) and $\alpha = 50$ (3). The number of clusters increases with $\alpha$.

chain Monte Carlo methods. In [23], Radford M. Neal reviews different variants which handle both where $F$ and $G_0$ are conjugate priors (using Gibbs sampling) or not (with Metropolis-Hastings sampling). See [15] for one detailed implementation.

All rely on the key property of the process, the expression of the conditional prior:

$$\theta_i | \theta_1, \ldots, \theta_{i-1} \sim \frac{1}{i - 1 + \alpha} \sum_{j=1}^{i-1} \delta(\theta_j) + \frac{\alpha}{i - 1 + \alpha} G_0$$

Where $\delta(\theta)$ is the distribution concentrated at the single point $\theta$. The probability of a cluster to join another one is proportional to the number of members of the “attractive cluster”. So one data is more likely to join an important cluster. On the other side, the $\alpha$ parameter controls the ability to create a new cluster following the $G_0$ prior.

Figure 3.4 presents some experiments I’ve made, clustering 1D data taking normal distribution for $F$ and Normal-gamma for $G_0$. Those experiments where first inspired by Escobar and West work [9]. But their work focused on the use of particular conjugate priors and my attempts of simulations were fruitless. The experiments displayed here were made following the general work of Radford M. Neal.

The convergence of Markov chain Monte Carlo involves very complex issues which are not yet completely mastered. Still, in practice, they is a strong tool used in numerous applications of physics, chemistry, biology, statistics, and computer science. And our cluster experiments in the Dirichlet Process framework showed rapid and accurate convergence. Moreover, the sampling method permits to deal with new data very rapidly. Indeed, the current “n observations” state of clustering can be taken as a near-optimal solution for the $n + 1$ problem. Finally, this solution still applies in the same way when dealing with changes in the environments i.e. when the $y_i$ can evolve with time. So this method could be used to implement our method in case of a very large number of ads which would urges us to cluster them.
Conjugate Prior

The goal is to estimate an hidden parameter $\theta$ given observations $x$. Moreover, in Bayesian probability theory, an additional information is provided (or arbitrarily asserted) : the probability distribution of $\theta$, called the prior. Hence, from Bayes’ theorem, the posterior distribution is calculated from the prior and the likelihood function.

$$p(\theta|x) \propto p(x|\theta) \ p(\theta)$$

But, because of the normalisation factor, this simple equation can lead to intractable calculations. And as the likelihood function is often imposed by the structure of the problem, the choice of the prior is a momentous issue.

A conjugate prior is a class of distributions, conjugated to a class of likelihood functions such that the resulting posterior belongs to the same class of functions as the prior. Simple relations are derived from this fact and therefore the tedious work is avoided.

Example Bernoulli Beta Conjugacy. Having a Beta distributed prior on the expectation $\theta$ of a coin flipping experience (Bernoulli process) leads to a Beta distributed posterior:

$$\text{Beta}(\theta, \alpha + \sum_{i=1}^{n} x_i, \beta + n - \sum_{i=1}^{n} x_i) = \text{Bernoulli}(x_1, \ldots, x_n, \theta) \ \text{Beta}(\theta, \alpha, \beta)$$

*See [10] for a full detailed explanation

3.5 Multi-armed bandit

The multi-armed bandit problem (MAB) is a game where, at each step, the player has to choose which lever to pull among $K$ slot machines. He then receives a reward drawn from the reward distribution of the machine he has picked and his goal is to maximize his income. To do so, the player has to satisfy two opposite desires : playing the arm he believes is the best at this state of the game or improve his knowledge about the others arms because one of them is perhaps better than its current favorite. This is referred to as the famous exploration versus exploitation trade-off.

The Multi-armed bandit was first introduced by Robbins in his seminal work [27] in 1952. Since then, this problem has been a very active field of research and has led to various extensions. For instance the reward can been drawn from a restless distribution or chosen by an adversary. The set of arms could be continuous...

In our present work we could see the choice and printing of an ad to a web-visitor as a bandit problem. In fact, as every visitor has his own taste, we would play a different bandit problem for each visitor. Such an individual case approach, would not be very precise because it doesn’t take into account the budget associated to each advertisement (This point will be discussed in 4.7).
3. REVIEW OF EXISTING MODEL

3.5. MULTI-ARMED BANDIT

The stochastic K multi-armed bandit is the first and simplest version of the MAB. In these settings Auer et al. (2002) have proposed the UCB policy [5]. This policy associates an index to each slot machine and pick at each step the one with the highest index. This index is simply the sum of the observed mean reward and a positive confidence term (see the box page 20). The theoretical results and practical performances of UCB has made it the reference policy in the domain.

In our work we will stick most of the time to this basic MAB. However, some other variants will be now reviewed as they let us model in their way our problem or help us get more intuition about how to tackle it.

3.5.1 Bayesian Bandit

Right from the beginning, MAB studies have often assumed prior knowledge on the reward distributions of the arms. Moreover, in our case, the reward is 1 for a click and 0 otherwise which is basically modeled by a Bernoulli distribution. Then the prior information is restricted to the mean of the distribution. A convenient choice is to take a Beta prior which is the conjugate prior of the Bernoulli distribution. It permits to ripen iteratively our knowledge in a very handy way. Such an approach was explicitly proposed by Ole-Christoffer in [13] but failing to give theoretical guarantees. On the other hands, new results from Goel et al. [11] bound the performance of Bayesian bandits. But as they do not update the prior during the game, the results are quite deceptive.

In our model we adopt the same strategy as [13] rather than UCB. In his article, Ole-Christoffer showed that his strategy was working better than UCB (UCBtuned, in fact) in most cases. But what makes this argument even more correct is that in our problem we are working with extremely small probabilities (the order of magnitude is $10^{-4}$ or less). In this case, the confidence term of UCB takes a huge time to fade compared to the observed mean reward of the arms, compelling the player to play almost randomly for a very long time. This is well-known, UCB can be really slow to find the best arm when the best arms have very close expected rewards. But this is not tragic when the mistake is between 0.8001 and 0.8007 for instance because it leads only to a loss less than 1%. But a mistake between 0.0001 and 0.0007 divides your income by 7.

In Figure 3.5, the sensitivity of four classical MAB policies to very low reward expectations is studied. Besides UCB-tuned and Beta algorithms, we added $\varepsilon$-greedy and $\varepsilon_n$-greedy strategies. The $\varepsilon$-greedy algorithm picks at each step the best observed arm with probability $1 - \varepsilon$ or a uniformly sampled arm with probability $\varepsilon$. In the $\varepsilon_n$-greedy, the parameter $\varepsilon$ becomes the $\varepsilon_n$ sequence which decreases to zero so that the strategy converges to the best arm. We notice that Beta and the $\varepsilon_n$-greedy strategies perform similarly well in both cases with Beta being slightly better. On the other hand UCB performs very badly when facing very low “click rate”. The time it requires to discover, in this case, the best arm and stick to it is way longer than the other strategies. And, as we will pursue our study with those realistic range of probabilities and simulation time, UCB will no longer be used.

In addition, for the $\varepsilon$-greedy algorithms, parameters have to be tuned to reach an optimal performance. Plus, they are dependent of the reward distributions. For instance, in Figure 3.5, the displayed $\varepsilon$-greedy results were found after such a tuning and were different in both cases.
3.5 MULTI-ARMED BANDIT

3.5.1 UCB

**Initialisation:** Play each machine once.

**Loop:** Play machine that maximizes

\[
B_j = \bar{x}_j + \sqrt{\frac{2\ln n}{n_j}}
\]

where \( \bar{x}_j \) is the average reward obtained from machine \( j \), \( n_j \) is the number of times machine \( j \) has been played so far and \( n \) is the overall number of plays so far.

---

3.5.1 Beta

**Initialisation:** Play each machine once.

**Loop:**

For each machine, sample from the posterior Beta distribution with \( \alpha = \text{Success} + 1 \) and \( \beta = \text{Failure} + 1 \). Success being the number of 1 observed and Failure the number of 0. Choose the arm with highest sampled value.

---

3.5.1 \( \varepsilon \)-greedy

**Initialisation:** Play each machine once.

**Parameters:** \( 0 < \varepsilon < 1 \)

**Loop:** With probability

\[
\begin{aligned}
1 - \varepsilon & \quad \text{Play best observed arm.} \\
\varepsilon & \quad \text{Play an arm randomly.}
\end{aligned}
\]

---

3.5.1 \( \varepsilon_n \)-greedy

**Initialisation:** Play each machine once.

**Parameters:** \( c > 0 \) and \( 0 < d < 1 \)

\[
\varepsilon_n = \min\{1, \frac{cK}{d^{\varepsilon_n}}\}
\]

**Loop:** At step \( n \), with probability

\[
\begin{aligned}
1 - \varepsilon_n & \quad \text{Play best observed arm.} \\
\varepsilon_n & \quad \text{Play an arm randomly.}
\end{aligned}
\]

---

Table 3.1: Four classical multi-armed bandit strategies.

Non optimal parameters would have led to poor results. On the contrary, Beta is a very handy non parametric solution. It only needs a prior. But this prior can be the uniform distribution (Beta\((x, 1, 1)\)) which seems to give excellent experimental results. In fact in the case of the very low probabilities this prior plays the role of an upper bound because it has an optimistic effect on the posterior distribution urging the algorithm to explore the arms until the probability of the arms are more carefully shaped.

3.5.2 Multi-armed bandit with side information

Multi-armed bandit with side information is an extension of the original MAB problem with applications in on-line advertisement. It was born from the need to model the external information the player may have access to while having to make his choice. In the ad serving settings, those would typically be the information available on the Internet user, the page he is visiting, the time of day or the season, etc... Nevertheless, in this scope of research, each analyst has model this additional information in his own way, depending on his fields of application or theoretical purpose.

During his thesis, Wang (2005) was one of the first to bring such issues. In [30], he introduced a very theoretical analysis of a two armed bandits with a separated process \( X_t \) giving information.
Figure 3.5: 7-armed bandits with Bernoulli distributed rewards. The arms expectations are 
(0.9, 0.8, 0.6, 0.5, 0.4, 0.2, 0.1) (left) and $10^{-3} \times (0.9, 0.8, 0.6, 0.5, 0.4, 0.2, 0.1)$ (right). Averaging respectively 1000 and 100 simulations. The cumulative rewards are displayed in the shaded area bounded by the “always pick the best arm” policy and the uniform random policy.

about the underlying parameters of the two reward distributions. Four different cases of the relation linking $X_t$ to the expected reward are addressed. This first theoretical work has been followed by more application driven papers. In [24] (2007) Pandey et al. add taxonomies of the website pages and of the ads. Under a bandit feedback the goal is to match those two feature spaces. This taxonomy is a family of clusters on pages and on ads. Therefore this model is informative as it shows a way to infer clicks rate of people through the gathering of similar persons. Moreover, they build a two stages bandit method (select the cluster first, then select in this cluster the ad) which is more deeply broken down in [25]. For Kakade et al. [18], the side information is a vector $x \in \mathbb{R}^d$. Their “Banditron” inspired by the Perceptron is an algorithm whose goal is to classify those vectors into K labels, i.e. the K bandit arms. Here, the ad serving problem is reduced to an on-line multiclass prediction problem with bandit feedback.

Other reference papers are due to Langford [21][20] where he proposes an epoch-greedy algorithm and analyse the possibility to assess a new policy from results coming from other policies in a very general contextual bandit setting with abstract hypothesis space.

Those methods are among the most accurate for our problem and it is not a coincidence if most of their authors are from Yahoo!. They should be considered as the base techniques. In our worked we tried to see if we could go even further dealing with the limited budget of the ad and the fact that we should play each ad uniformly during a given time period.
3.6. LINEAR PROGRAMMING

3.5.3 On-line linear optimization

On-line linear optimization is a generalization of the multi-armed bandit problem where the player has to choose at each turn a point $x \in D$ with $D$ a subset of $\mathbb{R}^n$. Moreover the cost functions are linear. Its results are often motivated by the need to design adaptive routing algorithm in networks but is also often referred to as a method to tackle the on-line ad placement.

Auer (2002) [4] and Abe et al. (1999) [1] analyse very similar models. In those, the player has a finite number of choices, each represented by a vector $z_i(t) \in D$. Then, the cost functions being linear, the expectation of the reward $x_i$ of arm $i$ is $E[x_i] = f \cdot z_i(t)$, with $f$ an unknown vector to approximate. This model is applied to Internet ads serving, regarding arms as the ads to display and the vector $z_i(t)$ as an array of features characterizing the ad, environment, user or a combination of the former ones. This approach is very classical in reinforcement learning and could be efficiently implemented in a final model to try to predict (as a prior?) the click probabilities. But it makes the strong assumption of linear cost functions which has to be tested with real data. And, in our settings, this method could be critically curbed because of the lack of information we dispose from Orange. Moreover it does not take into account the limited budget of the ad which is a key constraint of our study. Finally, the algorithm of Auer would be preferred in terms of regret guarantees but it uses complex linear algebra computation which make it not suited for industrial implementations.

But the two former examples don’t capture all the complexities of playing a bandit problem in $\mathbb{R}^n$. This more general problem could be very interesting for us. Indeed, our budgeted ads and our knowledge about the number of visitors on the website turn our problem into a planning problem where we have to output a vector $x$ representing the number of visitors of each profile we want to allocate to each ad. Thus $x \in \mathbb{R}^{K \times N}$ with $K$ the number of ads and $N$ the number of visitor profiles. In addition, the reward function is linear and proportional to the hidden click rate of each ad-profile pair. Dani et al. (2008) [8] proposed such an algorithm in the stochastic case. In addition they treat the bandit feedback setting where the player is only revealed the reward associated to his choice and not the linear function drawn from a fixed distribution. However, some differences prevent us from applying their algorithm. In our problem the taken action changes the environment. As we play there is less and less visitors to come and ad to click on making $D$ vary with time. In fact our problem can even be treated as a Markov decision process. Furthermore, in our settings we dispose of more information about where the reward comes from. We know we tried to make this profile click on this ad. So we are able to learn more from each experience, improving each time our knowledge on a particular projection of the reward distribution.

3.6 Linear Programming

Linear programming (LP) is an optimization technique of a linear objective function, subject to linear equality and linear inequality constraints. It was developed as a discipline in the 1940’s, rapidly finding applications during the second world war. In the postwar period, with the publication of George B. Dantzig’s Simplex Method (1947), linear programming was rapidly used in many industries in order to maximise profit.
Linear programming is usually introduced giving the example of a manufacturer finding the best assignment of people to jobs, of raw material to different products or of the farmer cultivating wheat and barley on different fields. Here, our task is to allocate ads with a limited budget of clicks to a certain number of web-visitors. Furthermore, we will rapidly deal with clusters of similar people to avoid too large problems.

In the following, the first model of LP for our on-line advertisement will be built. Yet, this modelling assumes knowledge of the appetite of the customers for the different ads. This is a little bit hasty as those click probabilities are precisely what we are trying to estimate. That’s why, our final model should incorporate the tools reported in the MAB section (3.5). Another solution could be to solve the linear program by replacing the unknown values by distributions depicting our believes. This is called stochastic linear programming. Finally, to close the review, we refer to adaptive probing, an optimization method which has to compute the optimal probing sequence of unknown parameters before playing.

### 3.6.1 Deterministic Linear Programming

Our goal is to allocate $N$ various profiles of customers in $K$ ads. If we are working during one day, we know then that $C_i$ persons from the profile $i$ will come to our website. Each ad $j$ is associated with its click budget $A_j$. And let’s call $p_{i,j}$ the probability of the profile $i$ to click on the ad $j$ (here we assume we know this probability). Eventually, we need to compute the $x_{i,j}$ which represents the number of persons from the Profile$_i$ to which we are willing to assign the Ad$_j$.

This problem can be formalized with the following equations where we try to maximize the number of clicks in expectation.

Given $p \in \mathbb{R}^{K \times N}$, $A \in \mathbb{R}^{K}$ and $C \in \mathbb{R}^{N}$ find $x$ such that (and note $x = LP(p, A, C)$):

\[
\begin{align*}
\text{Maximise} & \quad \sum_{1 \leq i \leq N} \sum_{1 \leq j \leq K} x_{i,j} p_{i,j} & \quad (3.4) \\
\text{Subject to} & \quad \sum_{1 \leq i \leq N} x_{i,j} p_{i,j} \leq A_j & \quad \forall j \in \{1, \ldots, K\} \quad (3.5) \\
& \quad \sum_{1 \leq j \leq K} x_{i,j} \leq C_i & \quad \forall i \in \{1, \ldots, N\} \quad (3.6) \\
& \quad x_{i,j} \geq 0 & \quad \forall i \in \{1, \ldots, N\}, \forall j \in \{1, \ldots, K\} \quad (3.7)
\end{align*}
\]

Equation 3.4 maximizes the number of clicks. Equations 3.5 are the constraints on the click budget of the ads. Equations 3.6 are the constraints on the number of persons in the profiles.
To solve a linear program, the standard method is to use the Simplex method. Though it is not a polynomial algorithm, and can even be exponential in certain critical cases, it has been largely adopted in the industry for its very good experimental speed. And even the more recent interior point method which is a polynomial algorithm is experimentally slower. That’s why we chose to use the Simplex method in our simulations relying on glpk library.

Linear programming comes with sensibility analysis. Sensitivity analysis is a systematic study of how sensitive solutions are to changes in the data \[7\][17]. One can, for instance, compute on what interval each \( p_{i,j} \) can belong so that the \( x_{i,j} \) stays in the same state (being or not equal to zero). Based on this analysis, an experimental strategy was tested but did not scored so well. The idea was to select an ad with probability proportional to the posterior probability given the data that \( p_{i,j} \) was equal to the switching value of the sensibility analysis (more details in the following chapter).

3.6.2 Stochastic Linear Programming

Stochastic Linear Programming \[19\][28] is a variant of Linear Programming where the parameters of the program are replaced by distributions. Indeed, most of the time our knowledge about some parameters of the problem is partial. Typically, we can represent our prediction for the number of web visitors to come as a distribution instead of fixed parameters. This distinction permits to make a more subtle work. The maximisation is now on the expectation of the rewards. But modifications are necessary in order to keep the constraint meaningful. Penalty costs are introduced to punish exceedances. Another approach is the “Probabilistic Constraint” which computes a policy of allocation whose probability of satisfying the constraint is less than a given precision parameter chosen by the user.
Minimize \[ cx + \sum g_i \mathbb{E}[|b_i - a_i x|] \] \hspace{1cm} (3.8)

subject to

\[ A_1 x = b_1 \] \hspace{1cm} (3.9)

\[ L_1 \leq x \leq U_1. \] \hspace{1cm} (3.10)

The above equations are an example of a simple Recourse problem. Equations 3.10 and 3.9 are standard linear constraints. In equation 3.8, the first term is the objective function and the second is the expected accumulation of the penalties. Those penalties are proportional to the expected exceedance on some additional stochastic constraints.

In our context such tools could be used to model our uncertainty on the number of visits on the website. Moreover one could think about modeling our knowledge on the click probability by a posterior distribution given the observed data. But this method is a more complex version of the linear method problem leading to complex algorithm solution. They can rapidly become intractable and hence urge to use sub-optimal routines. Moreover, new parameters are introduced by these methods (“penalty cost” in the Recourse problem case or “precision” in the probabilistic constraint case) burdening the analysis. For those reasons, stochastic programming was not exploited in our model.

### 3.6.3 Model driven optimization

Here is a quick review of model driven optimization which is a neighbouring domain to the former ones. Besides, Figure 4.8 is an attempt to structure those fields around our forthcoming model. In model driven optimization, the goal is again to find the best policy which would maximize an objective function. However, some of its parameters are only known through their prior probability distribution. But we are allowed to probe a few of them before playing. Here, probing means to reveal the true value of the variable. The question is then to choose an appropriate sequence of probing. Thus, we have seen clear connections to our problem where we try to maximize benefits by playing and probing uncertain values. Yet, in our settings, the probing does not completely reveal the hidden value but only gives us a realization of an unknown random variable.

This domain was initiated by Goel et al. in [12] (2008). In [14] (2008) Guha et al. analysed the importance of the adaptivity of the probing sequence.
Building our own model

4.7 Aggregating ideas

Based on our previous discoveries of state of the art methods to tackle the Internet banner advertisements problem, we decided to create our own model. All in all, we just attempted to aggregate the techniques which appear to have the best characteristics. First, linear programming permits us to schedule our display and handle the limited budget of the ads. Moreover this formulation naturally urges us to cluster our web-visitors into various profiles (see Figure 3.6). Thus we match a native modeling of the problem by Orange (dividing up people according to their age, gender...) and inherit from the simplification of this method and the possibility to predict by looking to the behaviour of similar customers. And finally, we integrated the multi-armed bandits approach to discover the click probabilities of a given profile on a given ad.

To sum up: We possess clusters of people visiting our websites and of Internet banners. Our goal is to maximize the total number of clicks. To do that the click probabilities of the different profiles on a the different ads have to be carefully estimated while playing to get optimal results.

We know: The proportion appearance of each profile. The number of clicks asked by the advertisers for each ad. The time period of the ads.

We infer: The expected click rate of each profile on each ad.

Then, we will try to make our model more and more realistic by introducing step by step more features into it. Thus permitting us to study their different characteristics one after another. Our first shot will work with a fixed number of ads and a fixed and known number of visitors in the website. Then, our linear program approach will be tested in a dynamic context with the sequential arrival of new ads. Finally mortality of the ad will be modeled.

4.7.1 The relevance of linear programming

Using linear programming permits to obtain better results than just playing blind parallel multi-armed bandits. Blind parallel bandits means that the various profiles are not aware that they share
the same resources. Therefore the associated algorithm just tries to give to each profile the ad for which its probability of click is the highest, then just running parallel bandits. Here is an example.

<table>
<thead>
<tr>
<th>Budget</th>
<th>Ad 1</th>
<th>Ad 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>100</td>
<td>0.8</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Explanations of the above table: The Budget indicates the maximum number of clicks for each ads. We put on the left of the table the number of visitors for each profile. Finally in the table are the click rate probabilities.

<table>
<thead>
<tr>
<th>Budget</th>
<th>100</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>50</td>
<td>62.5</td>
</tr>
<tr>
<td>100</td>
<td>3.75</td>
<td>37.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Budget</th>
<th>100</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>75</td>
</tr>
</tbody>
</table>

Planning of parallel multi-armed bandits.

In the two plannings, the numbers in blue are the expected clicks, and the numbers in red are the expected number of persons allocated to a given ad. The bandit approach chooses for every profile its favorite ad. So by definition the result is less or equal, in expectation, to the optimal one found by linear programming. In this example LP collects 137.5 clicks whereas the blind bandits only reach 122.5 clicks. So when the budget of the ad is limited a blind bandit method is not optimal (but is still fairly good). And linear programming is more “clever” as it sees that we should prevent profile 1 from facing Ad 2 and therefore gives him priority on Ad 1.

4.7.2 Mixing bandits and linear programming

Our formulation:

\[ N \]: the number of clusters of the profiles.
\[ C_i \]: the number of persons in Profile\(_i\).
\[ K \]: number of ads (or clusters of ads).

An ad \( j \) is defined by

\[ \Rightarrow A_j : \text{the click budget.} \]

\[ \Rightarrow p_j : [0, 1]^N \text{ where } p_{i,j} \text{ is the click probability of Profile}_i \text{ on ad } Ad_j. \]

In the previous examples we compared policies that had knowledge of the click probabilities of every profile-ad pair. This is cheating because those probabilities are of course unknown and all our task is to estimate them. In fact we have to estimate them while playing, raising again
4.7. AGGREGATING IDEAS

4. BUILDING OUR OWN MODEL

<table>
<thead>
<tr>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current estimators: ( \hat{p}_{i,j} )</td>
</tr>
<tr>
<td>Compute the linear program: ( x = LP(\hat{p}, A, C) )</td>
</tr>
<tr>
<td>Compute the probability allocations: ( s_{i,j} = \frac{x_{i;j}}{|x_i|} \forall i \in 1 \ldots N, \forall j \in 1 \ldots K )</td>
</tr>
<tr>
<td>with ( |x_i| = \sum_{j=1}^{K} x_{i,j} )</td>
</tr>
<tr>
<td>A visitor from profile ( i ) arrives.</td>
</tr>
<tr>
<td>( Ad_j ) is showed with probability ( s_{i,j} ).</td>
</tr>
<tr>
<td>( \hat{p}_{i,j} ) is updated.</td>
</tr>
</tbody>
</table>

Table 4.2: The basic Simplex routine.

the exploration vs exploitation dilemma. The basic option is to compute the linear program (see Section 3.6.1) with our current estimators \( \hat{p} \) of \( p \). With \( p \in \mathbb{R}^{K \times N} \) and \( p_{i,j} \) the probability that Profile \( i \) clicks on \( Ad_j \). The linear programming solution is a vector \( x \in \mathbb{R}^{K \times N} \) where \( x_{i,j} \) is the number of persons from Profile \( i \) to be allocated to \( Ad_j \). A new web-visitor then arrives. Let’s say he is from Profile \( i \). We have \( K \) possible choices of ad to display. Our algorithm displays an ad \( j \) for which \( x_{i,j} \neq 0 \) and we observe whether a click occurs or not. Then, \( \hat{p}_{i,j} \) can be updated thanks to the new observation and a new schedule \( x \) can be computed by running another LP problem. This is summarized in Table 4.7.2. Notice that if we execute the planning of the LP (following the proportion \( s \) from Table 4.7.2), the clicks will be spread out uniformly during the simulation.

So, our algorithm plans at each time step what is the best schedule to follow based on its past observations. But this planning is not followed until the end of the experience. Its role is just to give us what is supposed to be the best action for the next visitor. Then the response of the visitor permits us to improve our estimations and to compute the next planning which should only be exploited for the next visitor etc... Still, computing at every time step the schedule with LP can be very costly. A straightforward simplification is then to perform each new planning after having seen a significant number of visits in the website. Such a simplification is addressed in Figure 4.7 to see its repercussions on the performance.

Moreover, we cannot content ourselves with following to strictly the planning of the LP. Indeed, such an algorithm performs no exploration of the other possible displays. Then, if a click rate which was excellent has been undervalued in the first step, the corresponding pair will probably never be played and thus the algorithm will perform sub-optimally. That’s why we will try to review possible mixtures of LP-bandits and compare them experimentally.

Some of those issues were already addressed in [2] (1999) and [22] (2005) by Naoki Abe and Atsuyoshi Nakamura. In those articles they created an equivalent model to tackle a similar Internet banner advertisement problem. However they did not take into account a budget constraint on ad clicks but on impression proportions. Our study will enrich the analysis of the basic model and then will propose extensions in which we have to play our problem sequentially with a dynamic environment of new ads arriving while others pass away.
In [26] (2006), Pandey et al. tackle a similar problem with budget constraints on the ad revenues. Nevertheless, they study it in the context of multiple ad display urging them to estimate the click probability conditionally to the other ads and so to restart the estimation when ads disappear. Moreover they did not try to use the linear program approach.

4.7.3 Methods

As we've just said, the algorithm has to explore all the profile-ad pairs. To do so, it has to differ slightly from the plan given by the linear program. We can see two different methods do to so.

**Deflect from the schedule of the LP**: As in the MAB settings, \( \varepsilon \)-greedy methods can be applied. This means following the LP with high probability and with low probability choose randomly an ad. This approach is equivalent to changing the LP by putting lower bound on the \( x_{i,j} \) in [2].

**Modify the LP**: The \( \hat{p}_{i,j} \) can be modified to compel the planning to include exploration. Abe and Nakamura used Gittins indices. Here we have investigated the upper confidence bounds of UCB. But, as discussed in section 3.5.1 they are not at all suited to low income probabilities. The Beta algorithm was also tested out. Here the idea is to replace the \( \hat{p}_{i,j} \) by values drawn from their posterior Beta distributions. As said in section 3.5.1, this Beta distribution with uniform prior can be viewed as optimistic in the context of very low success probabilities.

Keeping the Beta posterior representation, we have now a probability distribution for each parameter of our optimization and we are able to probe each of them to ripen our knowledge. So it seems we are at the intersection of the stochastic linear programming and the adaptive probing review in the previous chapter. This picturing is showed in the Figure 4.8.

Figure 4.9 compares different policies for our problem. Those simulations are based on the Orange toy representation of the global problem (see section 2.1). There were seven ads to display and the web-visitors were grouped according to their characteristics and the page they were visiting. First three “Anytime” policies performance are compared all along the x axis. When using the linear programming method one has to specify the number of web-visitors expected. For those three curves, for each abscissa point x giving the number of web visitors to come, the complete simulation is run with this number of web visitors and the performance recorded. On the contrary, the “16M” curves are just the follow up of one simulation which believes 16 million people will come.

In the “Anytime policies” : “Best” which plays knowing the real \( p_{i,j} \) and scheduling according to a LP, Simplex-\( \varepsilon \) merging linear programming and \( \varepsilon \) exploration, Blind-\( \varepsilon \) running parallel blind bandits on every profile. As expected “Best” is the best and Simplex-\( \varepsilon \) outperforms the blind bandits. But this has to be moderated, because the difference is at most of 5%. If the number of visitors is small (inferior to 8 million), Simplex-\( \varepsilon \) and Blind results are the same because the resources of Ad clicks is large enough for every profile to have its favorite ad. If the number of visitors is massive (superior to 30 million), every policy performs the same because they have had
4.7. AGGREGATING IDEAS

Figure 4.7: Sensitivity to the frequency of the schedule. As Simplex-$\varepsilon$ always reserves a part of its play for the exploration, it is less affected by the reduction of the scheduling frequency. On the contrary, Beta strictly follows, between each new planning, a given schedule and makes no exploration. Thus, exploration will be done by fits and starts. More generally this different sensitivities distinguish the two types of methods introduced above, the ones which deflect from the LP, less sensitive to low frequency re-scheduling and the “upper confidence bound” LP ones which definitely are.

Figure 4.8: Contextualization of our Model among some optimization methods.
Table 4.3: Comparison of the average number of clicks obtained by different policies with a fixed number of visitors (16 million). Averaging 100 simulations.

<table>
<thead>
<tr>
<th>Policies</th>
<th>Best</th>
<th>Simplex-$\varepsilon^3$</th>
<th>Robust Simplex-$\varepsilon$</th>
<th>Beta</th>
<th>Blind $\varepsilon$Bandits</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results at 16M</td>
<td>5983</td>
<td>5756</td>
<td>5739</td>
<td>5720</td>
<td>5531</td>
<td>4933</td>
</tr>
</tbody>
</table>

enough time to empty all the ads’ clicks. In between, Simplex-$\varepsilon$ results prove that the LP approach can be fruitful.

The difference of performance between “Best” and Simplex-$\varepsilon$ gives information about our ability to learn the hidden parameters ($p_{i,j}$). In the end, the gap is small (just a loss of 3%). “Best” is still a LP based policy, it is not the optimal treatment of the problem. So here we do not question the relevance of our model but this comparison helps us discover the price of having to accomplish an on-line estimation. This low price is linked with the fact that our clusters of similar visitors are big enough to infer the clicks rate precisely and then have time to exploit our knowledge. Thereof, an equilibrium has to be found for the number of clusters. A small number of large clusters permits to learn more easily the click rate whereas numerous little clusters permit to split more accurately the data but makes it harder to estimate the parameters. This key issue goes beyond the frame of our present study as it would need to be based on real data from Orange. That is why this is left as future work.

The three “16M” are presented to show how one simulation really happens. By definition the simulations join the “Anytime” Simplex-$\varepsilon$ in 16 million but before that they are not always optimal. This means that if the algorithm had forecast 16 million people to come and that only 8 million showed up this algorithm would perform poorly likened to a Simplex-$\varepsilon$ which would have forecast 8 million visitors. And in some cases they could even perform worse than the random policy. In fact an Anytime policy is impossible to design (remember that our “Anytime” policies are just aggregations of simulations results of algorithms having forecast well for each abscissa). So it is paramount to have a precise idea of the number of visitors to come. Comparing the “16M” curves permits to see that in the end they perform similar but that their way to achieve their results is not the same. Thus “Robust Simplex-$\varepsilon$” was designed to have a better anytime behavior. Normally the planning outputs a distribution over ads for each profile. “Robust Simplex-$\varepsilon$” does not draw its choice from those distributions but deterministically picks the ad with highest probability among those selected by the planning for a given profile.

**Regret** To judge the policies on the multi-armed bandits game, one can produce a bound on the “regret” which is often defined as the expected difference at each time step between the best possible total cumulative reward and the cumulative reward of the policy. In the case of MAB, the best behavior is to always pull the best arm. Thus the computation of the regret is equivalent, here, to estimate the expected time passed in the sub-optimal arms.

But in our problem an arm is not played because it has the highest estimated click rate but because it is selected by the linear program. So to determine the probability of pulling one arm, we have to know the estimation of all the arms (this is related to sensitivity analysis) whereas in
Comparison of policies

Figure 4.9: Comparison of policies on the Bandit Linear programming problem. Averaging for each curves 100 simulations. Here the hidden probabilities are the same in each simulations and derived from the Orange’s problem formulation.
the MAB we would just compare it to the mean of the best arm. This make the analysis harder. And we were unable during the internship to produce bound for this problem.

Still, the regret could be written as follows:

\[
E\left[ \sum_{t=1}^{T} R_{LP(\hat{p}, A, (T-t)\times C)}(t) - \sum_{t=1}^{T} R_{LP-\varepsilon(\hat{p}, A, (T-t)\times C)}(t) \right]
\]

The first term is the cumulative reward of a policy performing at each time step the LP planning knowing the hidden click probabilities \( p \). And the second term is the cumulative reward of the Simplex-\( \varepsilon \). The expectation is taken here on the arrival of the profiles which is stochastic and on Simplex-\( \varepsilon \) which is a stochastic policy.

**Sensitivity Analysis** In linear programming, sensitivity analysis permits to evaluate the sensitivity of the solution to perturbation in the parameters of the LP such as the objective function or the constraints (see Chvatal [7], chapter 10). Let’s focus on the case where the objective function parameters (the \( p_{i,j} \)) can change. The C library glpk, used to implement our simulations, comes with those tools. \( \hat{p} \) is our current click rate estimator. Glpk provides the interval around \( p_{i,j} \) on which each parameter \( p_{i,j} \) can run without changing the solution. Here, we move one parameter at a time and let the others fixed to discover the interval. Moreover we say that the solution doesn’t change if the corresponding \( x_{i,j} \) does not switch between zero state, \( x_{i,j} = 0 \) (non basic variable) to non zero state \( x_{i,j} \neq 0 \) (basic variable). One could take advantage of this information because it indicates how far is one pair of ad-profile from being played. For example if we have estimated after some trials that the click rate associated with this ad-profile pair is 0.2 and suppose that the sensitivity analysis tells us that it would need to be above 0.5 to be selected by the schedule, then we could compute the probability of such case (with Beta distributions). Thus we could measure the relevance of playing each couple as the probability that the real click rate is the requested click rate given our past observations:

**Loop**

1. Current estimate : \( \hat{p}_{i,j} \)
2. Compute Linear program \( x = LP(\hat{p}, A, C) \)
3. Compute Sensitivity analysis \( [a_{i,j}, b_{i,j}] = SV(A, C, \hat{p}) \forall i \in 1 \ldots N, \forall j \in 1 \ldots K \)
4. if \( x_{i,j} = 0 \)
   - \( u_{i,j} = P(p_{i,j} > b_{i,j} \mid \text{trials}) \)
   - \( u_{i,j} = 1 - P(p_{i,j} < a_{i,j} \mid \text{trials}) \)
5. Then if a profile i comes show him ad j with probability proportional to \( u_{i,j} \). Observe and update the “trials”.

Unfortunately, our attempt to code this method did not scored as good as the other ones (around 5550 on the 16M simulation). But, while arguing about it, the idea of the beta distribution came out. In the LP, the \( p_{i,j} \) are replaced by a value drawn form their posterior distribution given the past observations. Thus, we approximate the probability of each variable to be a basic solution of the LP given the past.
4.8 Playing sequentially

In the last section we have shown that on a particular version of the problem, Linear Programming could produce an enhancement of the performances. What makes this version particular is that we played it with a limited number of web-visitors known by the algorithm whereas in reality there is just a constant flow of visitors and new ads appearing.

4.8.1 What horizon for our planning?

Our model is changed so that our algorithm has to play a never ending game with new ads appearing randomly according to a generative model of ads arrival. Applying linear programming requires to specify the number of web visitors that will connect to the website. Hence, when we just play sequentially with a never ending Internet users traffic, this parameter seems to lose its meaning. But that’s not the case. In fact it is related to our prediction of what will occur in the future.

The new formulation:

\[ N: \text{the number of clusters of the profiles.} \]
\[ C: \text{the proportion of appearance of the cluster,} \]
\[ C_i: \text{the probability that the next visitor is from } \text{Profile}_i. \]
\[ K(t): \text{number of ads, depends of the time.} \]
\[ \text{An ad } j \text{ is defined by} \]
\[ \rightarrow A_j: \text{the click budget.} \]
\[ \rightarrow p_j: \text{a } [0, 1]^N \text{ where } p_{i:j} \text{ is the click probability of } \text{Profile}_i \text{ on ad } \text{Ad}_j. \]
\[ M \text{ the generative ad model where :} \]
\[ \rightarrow u: \text{probability of appearance of a new ad at each time step.} \]
\[ \text{H, the horizon, the number of people} \]
\[ \text{with which we compute the LP.} \]

The strategy we will study changes just a little compared to the last section. We are still planning with our current \( K(t) \) ads with no idea of what will happen next. We compute the LP problem with a chosen parameter \( H \) which is the number of persons we are expecting to see. And then LP gives us the policy to apply to the next visitor represented by the vector \( s \) defined in table 4.7.2 with here \( x = LP(\hat{p}, A, C^*H) \).

First, notice that if we fix \( H = 1 \), then our linear program reduces to blind bandits because allocating ads to just one person cannot consume an entire ad. And in this case we just give to this guy its favorite ad because he is alone, and does not have to share with anybody else. But if we increase this horizon above a certain point, LP will differ from the blind bandit because the limited resources of an ad will be reached and then a clever schedule is needed. This planning induces that each profile will not necessarily be given their favorite ads. But LP will provide the optimal schedule for that number of people to come. And the bigger the horizon, the larger the number of ads to be played.

But of course, this schedule is not necessarily optimal in the case where new ads arrive. Indeed it seems that in the short term, if we possess two ads, one good and one bad, and if we know that
a lot of very good ads will appear really soon, then we should set a small horizon to play only our current good ad in the short time before the others arrive. On the contrary if the new ads only appear late, then we should fix a bigger $H$ to better manage our current resources. And a mistake in the choice of the parameter $H$ leads to bad performances. So we see that $H$ reveals our confidence in the future which can be qualify by the quality of the upcoming ads and their frequency of appearance.

<table>
<thead>
<tr>
<th>Budget</th>
<th>100</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad 1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>profile 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ad 2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>profile 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Planning with $H=20$
Profile 1 : 100% on $Ad_1$
Profile 2 : 100% on $Ad_1$

<table>
<thead>
<tr>
<th>Budget</th>
<th>100</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad 1</td>
<td>125</td>
<td>25</td>
</tr>
<tr>
<td>profile 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ad 2</td>
<td>0</td>
<td>150</td>
</tr>
<tr>
<td>profile 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Planning with $H=300$
Profile 1 : 73% on $Ad_1$ and 17% on $Ad_2$
Profile 2 : 100% on $Ad_2$

Role of $H$ in the planning. Made with the table printed in page 27. We observe two different behaviors according to the value of $H$. With a small horizon ($H=20$), the planning selects only the best arms for the profiles. If $H$ is large, the program becomes far-sighted.

The difficulty is to clearly imagine what would happen with a generative model of ads which would manage the arrival of the ads. With experiments we can liken policies with different horizons. The idea would be to compare their expected mean income per unit time when those systems have treated a sufficient number of ads to be in kind of stationary state.

Figure 4.10, compares the performances of policies playing each with a fixed horizon. To improve the speed of the simulations, we made drastic simplifications. The cluster of the profiles and the generative model of the Ads are now completely randomized and no longer follow the Orange formulation. Moreover they were made with 3 profile clusters. However, the click rate probabilities, time step and clicks constraints are still of the right order of magnitude. Finally we completely put aside the on-line learning part to quicken computation. Here each policy is playing knowing the hidden probabilities. We assume the role of the horizon can still be analysed in this context. In fact, as shown in the previous section, the ads where played sufficiently so that the algorithm has a precise idea of the hidden parameters.

The result is a little bit surprising. Our intuition made us expect that we would find an optimal horizon depending on the generative model which here only changes as $u$ is modified. But the result is that the bigger the horizon, the better the final mean click rate leading to conjecture 4.8.1. Quite disappointingly, the difference of performance are not that important (3% for the most). But still the LP approach seems to do no harm in any case.
Figure 4.10: Role of the Horizon in a dynamic ad environment. Averaging 1000 simulations. In each simulation the click probabilities are drawn randomly and are known by the player. $N = 3$. The frequency of ad appearance is proper to each experience. The Blind Bandit policy scores as the lowest horizon. In those experiences the random policy has a stationary mean click rate of: 192 for $u = 0.2$, 234 for $u = 0.2$, 239 for $u = 0.2$. The dashed line is the maximum mean number of click in a stationary regime which depends on $u$ the frequency of appearance of the new ads. The simulations begin with a pool of 30 ads. Compared to the stationary regime, this start is an “opulence period”. Thus low horizon strategies perform better in the beginning. A new Simplex is computed at each time step (10K).
Conjecture 4.8.1 Let $M$ be an ad-profile generative model, $P_t$ the current existing ads at time $t$, $\pi_H(P_t)$ the policy schedule by LP, and $R_{\pi_H}(P_t)(t,i)$ the reward of the display of an add chosen by $\pi$ to the profile $i$, then:

$$H \mapsto \mathbb{E}_{(P_t,i) \sim M} \left[ \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} R_{\pi_H}(P_t)(t,i) \right] \text{ is non decreasing.}$$

Still, we have some intuition to explain that phenomenon. When $H$ is large, the planning believes it has an important number of persons to deal with and consequently it will play a lot of ads at the same time. In the case where the generative model creates more ads than possible to consume with the visitors traffic, this is an advantage in the long term. Indeed, after a finite time, the model will have to deal with far more ads than his horizon tell him to play. And as we have not yet modeled the mortality of the ads, it has the choice among all ads born since the beginning and not yet consumed. Then he will combine them more smartly than a LP with small horizon which would play them one after another.

But this policy also produces in this context, some strange behaviors. Indeed, with no notion of mortality the schedule does not produce a planning with time periods, it just outputs a distribution on the ads to play for each profile. As our policy plans to play quite uniformly each ad, then it will plan, for instance to finish ad $A$ which asks for $m$ clicks in $H$ time steps. But this planning will not be followed for $H$ time steps. Suppose the planning is computed each $H \times 0.5$ time steps. Then after $H \times 0.5$ time steps, $m \times 0.5$ clicks remain to be done for ad $A$. And the planning will try make them the next $H$ time step. So after $H \times 0.5$ more steps $m \times 0.5 \times 0.5$ clicks remain to be done for ad $A$. This is the story of the arrow that at each step travels half of the remaining distance to the target and which in our case will really never achieve its goal.

The necessity to model mortality is therefore all the more strong...

In passing, we can notice that our linear programming methods can be not suited for a dynamic ad environment. In section 4.7.3 we distinguished two LP methods. The one which deflects from the LP planning and the one which introduces upper confidence bound in it. The first one will not try to explore in priority the new ads about which we know nothing. Whereas, with an upper confidence, the second one will play them as soon as they appear until it has a more precise idea about their characteristics. However, due to lack of time, we have not performed the simulations which would illustrate this idea. This is left as future work.

4.9 Mortality

The previous section has shown how the linear program approach could still be applied in a dynamic environment with the appearance of new ads. Nevertheless, our first dynamic model was not enough realistic and even generated weird behaviors. The remedy is to introduce the mortality of the ads in the model. Indeed, the contracts stimulate the period during which the ads have to be displayed.
4.9. MORTALITY

4. BUILDING OUR OWN MODEL

Figure 4.11: Scheme of ads arrival and lifetime.

\[ N \] the number of clusters of the profiles.
\[ C \] the proportion of appearance of the clusters,
\[ C_i \] is the probability that the next visitor is from Profile\(_i\).
\[ K(t) \] number of ads, depends of the time.
An ad \( A_j \) is defined by
\[ \rightarrow A_j : \text{the click budget.} \]
\[ \rightarrow [D_j, E_{dj}] : \text{the beginning and the end of the display period.} \]
\[ \rightarrow p_{ij} \text{ a } [0,1]^N \text{ where } p_{ij} \text{ is the click probability of Profile}_i \text{ on ad } Ad_j. \]
\[ M \] the generative ad model where :
\[ \rightarrow u : \text{probability of appearance of a new ad at each time step.} \]
\[ H, \text{ the horizon, the number of people with which we compute the LP.} \]

In this new context our goal will still be to find what would be the best fixed horizon. The main new information added in the life expectancy of the ads. And our strategy is aware of the moment at which each ad will disappear. But we still know nothing about the ads which will come in the future. In fact an even more elaborated model could incorporate information about the upcoming contracts and infer their quality (or play them a little before their beginning?). This is left as future work.

This attempt to analyse the role of the mortality of the ads was studied by Chakrabarti et al. [6] (2008) for the multi-armed bandit game. In this work, lifetime could be budgeted and revealed to the player or even be stochastic to model uncertain events which are beyond the control of the ad system. However, their results is based on the knowledge of a prior on the arms’ reward distribution and their work did not tackle the issue of allocated the limited clicks (not only in time) to several clusters of customers.

To take into account the limited lifetime of the ads, our linear program has to be modified. The mortality of each ad will be expressed as a new constraint in the LP. Remember that our policy only schedules with the current ads. The idea is to constrain, for each profile, the allocation on an ad to be not superior to the number of people that the ad will see during its life. But the allocation on ad \( A \) has also to take into account the number of persons already taken by the ads which will die before \( A \). Note that we made another simplification here: there is a linear relation between
Figure 4.12: Role of the Horizon in a dynamic ad environment and with ads only available during a given lifetime. Here we compare three scenarios with different ad lifetimes. Averaging 1000 simulations. In each simulation the click probabilities are drawn randomly and are known by the player. $N = 3$. The frequency of appearance of the ads $u$ was chosen so that the stationary regime is saturated (too much ads to display). But still, to fit to the reality we have elected a regime which is not so much saturated. The experiences seem to show that the optimal horizon is no longer the bigger and that mortality urges us to be more greedy than before. This optimal horizon seems to be linked with the lifetime of the ads. A new Simplex is computed at each time step (10K).

time and the number of persons that arrive. For instance, 1 day is equivalent to 4 million people visiting the website. $C$ gives us the proportion of each profile. And our modeling of the visitors arrival is to draw the profile of the new visitor from the categorical distribution given by $C$. So our model does not take into account some sort of seasonality or variations of the proportion according to the time of the day. This lead to the following additive constraints:

\[
\text{Sort } Ad_1, \ldots, Ad_{K(t)} \text{ so that } Ad_1 \text{ will die first and } Ad_{K(t)} \text{ die last.} \\
\forall i \in 1, \ldots, N \forall j \in 1, \ldots, K(t) \sum_{k=1}^{j} x_{i,k} < D_k * Profile_i
\]

The resulting planning gives us for each profile the allocations to the different ads. Then a little computing permits us to find in what order those ads have to be displayed in order to be played while they are still available. Notice than here the uniform use of the click resources of the ads is no longer guaranteed and could be the subject of further studies.

Figure 4.12 displays the performance of the different fixed horizon in three contexts where the lifetime of the ad is deterministically increasing. This new time limitation changes the result
compared to the former section. Now, the optimal fixed horizon is not the biggest. Now the best horizon is the one with has the best balance between being too greedy and too far-sighted. Still as the lifetime of the ads increases the system is getting closer to the former one with eternal ads. Then, we notice that with bigger lifetime, the optimal horizon is shifting, and the bigger horizon are getting the best one. The “1200K” experiment corresponds to ads with a lifetime of one month. This value seems to be a corner stone between carefully tuning the horizon and just taking the highest one (corresponding to the highest lifetime of all the current ads). So now we have to determine in which case the Orange Problem is.

**Get rid of the horizons?**  All our study has compared policies with fixed horizons. Then one would ask for more adaptivity to the environment. Or would be bored with the search for the best tuned fixed parameter. Indeed this would need a statistical test which would be had to face the high variance of such process. So, to end our work, we propose a last policy which would not need any extra parameter as the horizon. From the beginning, our policies were only performing their schedule with the present ads, not even trying to predict the future. And the issue of learning the hidden ad generative model are left as future work. One first idea was to estimate a mean ad of the generative model, see the effect of its appearance on every planning and tune them accordingly. Another simple idea is to simulate this generative model by reproducing the ads we have seen in the close past. A first implementation would be to record the ads seen in the last $T$ times (sliding window) and compute our planning, imagining that this arrival will happen again in the next $T$ time (or twice the same in the next $2T$ tile). This principle is quickly illustrated in the Figure 4.11. Here we have complicated things again, increasing our optimization just to have, in theory a planning for the next few visitors. This complexity prevented us from implementing it as it seems to need a cutting of the ads on all the distinct moments without any ad’s birth or death. Finally this model could be refined by introducing stochasticity in it. Then our planning would be an average of the results of those optimizations.
Conclusion and prospects

After having reviewed existing models for maximizing the income from Internet banner advertisement, we tried to study the relevance of introducing a linear program approach. All the study was led in the context of France Télécom formulation of the problem. This model was motivated by the constraint imposed on the number of clicks of each contract and fits with clustered data. We first tried to examine how it could be combined with the on-line learning of the ad characteristics. Then its performance were studied in the dynamic context of never-ending visitors traffic and ad’s birth and death. At first, LP seems to be not suited for this as it requires to be specified the number of visitors it has to allocate. This number, then, became a parameter of our algorithm. We have shown through experiences that its tuning was linked to the unknown generative model of the ads, and more precisely to the number of ads we have to deal with. Compared to a policy which would be not aware of the budgeted resources, Linear Program approach seems to improve the performance even if the horizon parameter is not carefully tuned. Still this improvement is quite moderated (4%).

This study is therefore encouraging but needs deeper analysis. Our experimentations, computationally greedy, due to the intensive calls to the Simplex optimization, urged us to make a flurry of simplifications which we hope did not bias the simulations. Thus we are now trying to use the power of distributed computing with INRIA’s Grid 5000 platform. This will permit us to explore more the kind of model we introduced last : models which try to simulate the generative model itself. Then we could explore here methods to learn the characteristics of the hidden ads generative model and make better predictions.

More theoretically, as we quickly addressed it, analyses should be performed in order to have deeper knowledge about the regret of the various policies. First the study could try to tackle this problem with a finite number of visitors and ads. Thus we would benefit of a strong tool to compare the efficiency of our policies.

A momentous issue left for future work was the analysis of the dependencies between our on-line learning and the building of the clusters. We have seen that a trade-off arises between the sharpness of the clusters and the ability of learning quickly their characteristics. Moreover our predictions, could be improved by introducing a more complex topology on the clusters. For example in the Orange problem, a realistic model should incorporate hierarchy in the groups because some characteristics about the visitors can be known partially. Thus the cluster of the men and the one of young men, for instance, can enrich each other.
Another way of development is the extension to multiple ad display, i.e. the possibility to display several ads at the same time on one web-page. This problem was tackled in [26] where they reset their policy each time an ad runs out of budget. We interpret this as a way to avoid the distorsion of the click probability estimators which is linked to the context (here the set of the other ads displayed at the same time). One could, for instance introduce change point detection to restart the policies when really needed (see [16]) or take into account the attention decay for the ads due to its localization in the web-page [20] (one ad printed at the bottom of the page has no chance of being clicked whatever its qualities).

Finally I would like to make a brief assessment of my internship. This internship has been for me a real engineering task where the real world problem have to be solved thanks to scientific tools. However, this internship makes me realize where my limits were and how I could try to challenge them. The goal being to acquire more self confident in order to be more autonomous. Moreover this work was the occasion of a transition in my status. As I followed the birth of this industrial project, its deroulment, the constant evaluation and its finalization. I now feel more comfortable with the leading of a project.
Bibliography


