Reinforcement Learning for Online Optimization of Banner Format and Delivery

Benoit Baccot[1,2]
Romulus Grigoras[2]
Vincent Charvillat[2]
[1]*Sopra Group, Division Midi-Pyrénées, Toulouse, France*
[2]*University of Toulouse, IRIT-ENSEEIHT, France*

ABSTRACT

In our Internet-connected world, online advertising has grown into one of the most successful advertising channels, since users spend an important amount of time browsing the web. Among the different types of online advertising (emails, games, etc.), we are particularly interested in contextual ads using rich media banners that display motion and exploit sensory information such as video, audio, animation etc. Once the various banners of an ad campaign are produced, a legitimate question arises for a web marketer: among various options, and for the same banner content, what is the optimal banner format and delivery policy?

In this chapter, we deal with three main problems a web marketer can be confronted with: the right format among those available (e.g. text, image, video, interactive, etc.), the right time to display a banner during a user browsing session (e.g. at the beginning, at the end or when salient events are detected, etc.) and the right sequence of banners to display (that takes into account the format and the time problem). We show that these problems share common points. These points fit well within the reinforcement learning framework: a "trial-and-error" process can be used to dynamically determine an advertising policy that optimizes a criterion based on an impact measure (e.g. the click-through rate or the session duration of a user). Two stochastic models based on Markov Decisional Processes and Multi-Armed Problems are presented in order to solve the three problems.

Results, showing the power and the efficiency of the two models to solve our problems, are also given. By comparing to a “ground truth” acquired by observing user browsing session on a test site, we conclude that our models are able to determine optimal advertising policies concerning banner formats and delivery.

INTRODUCTION

Today, online advertising has grown into one of the most successful advertising channels, since browsing the web has become a daily activity for a majority of users. Website owners call on the experience of marketers or online advertising agencies in order to design, produce and deploy ad campaigns (see for example (Marketing Sherpa, 2008) or (McCoy et al., 2007) if you are interested in this process).

Among the different types of online advertising (emails, games, etc.), we are particularly interested in contextual ads using rich media banners. In fact, on a commercial website, contextual ads are used to drive users and transform their navigation into a transaction. Traditionally, the banner's content has been presented in the form of text and hyperlinks. Recent studies such as (Rosenkrans, 2009) have shown the benefits of using rich media for displaying motion and exploit sensory information such as video, audio, animation etc. Rich media content is considered more attractive since it can grab users' attention easily and can also leave stronger memories (Mei et al., 2007). Rich media naturally fights the banner blindness problem, when users tend to completely ignore banners. These are some of the reasons that made rich media advertising very popular.

Unfortunately, some sites make excessive use of it, leading to the commonly called *ad overload problem*. The overabundance of banners or the poor targeting of ad campaigns make sometimes the user navigation on a web site difficult or unenjoyable. Moreover, users tend to learn (by reinforcement...) how to avoid clicking on banners, since they can lead to unwanted content, which is clearly contrary to the aim of the ad campaign.

Once the various banners of an ad campaign are produced, a legitimate question arises (Baccot et al., 2009): among various options, and for the same banner content, what is the optimal banner format and delivery policy?

In this chapter, we intentionally put aside the banner content issue and consider it (as other authors, like Hauser et al. in 2009) as a separate question. Thus, the problem is how to take into account three classical dimensions of a banner seen as a hypermedia document:

- the logical and spatial layout. The logical layout includes the different elements (and their links) that can be inserted in a banner (e.g. an image, or an image with a caption, a video etc.) whereas the spatial layout gives the way these elements are presented (e.g. a caption above or below the image).
- the level of interactivity of banners. Basic or more complex banners may be available, enabling users to click, scroll, type etc.
- the timing of delivery: the instants when banners are added. Adding them at very specific times (e.g. when the user level of interest falls) can increase their effectiveness and therefore their click-through rate (CTR).

A variety of options for designing and delivering the banners are available. Obviously, the various possibilities do not have the same impact, therefore what should we (or should we not) do, what are the criteria that allow us to choose?

The remaining of this chapter is organized in three sections. The first one (section 2) introduces the three banner problems we will deal with: the right format for a banner, the right time to display it and the right sequence of banners. While presenting these problems, we will introduce step by step all the ingredients that enable us to work in the reinforcement learning framework. In the next section (section 3), we formalize these problems and present more precisely the reinforcement learning framework. Two stochastic models that will help to solve the previous problems are also detailed. The third section (section 4) presents results obtained by solving the banner problems using reinforcement learning. The results prove the strength and the efficiency of the models. Finally, a discussion about the benefits of reinforcement learning is conducted. The chapter ends with a conclusion and some perspectives.

1,2,3… BANNERS PROBLEMS

Among the various possible use cases related to banner optimization of format and delivery, we choose to address three problems. These problems are important to solve, since they handle essential aspects of the banners: the right format, the right time to insert and the right sequence of banners.

The right banner format

Today advertisers produce a wide variety of banner formats that compete for capturing users' attention and fulfill advertisers' requirements... This includes various banners sizes (from leaderboard to skyscraper), rich in interactivity or not, or using various technologies (from standard text or images to a complex highly "dynamic" Flash application).

Marketers are confronted with a difficult question (Cole et al., 2009): among this large variety of banner formats, which one to use? The answer is not straightforward, since many parameters impact the effectiveness of a banner. Obviously, the usage context is important, since the same banner cannot be deployed on terminals or browsers with very different characteristics: mobile access requires small and lightweight banners whereas rich media banners can be easily displayed on a desktop computer. Marketers have also stressed the importance of fine targeting the banners according to users by taking into account users preferences, intentions and various other behavior parameters.

In brief, this first problem is to decide the right banner format according to what is generally called *context*. It is actually an instantiation of a more general multimedia adaptation problem: how to adapt a multimedia document (in our case the banners) to the context of usage (taking into account both human and non-human factors)?

Commonly, the problem is solved empirically by defining rules (expressed by marketers) which provide adaptation instructions (banner formats) upon detection of a salient event during users' navigation on a website. In order to pretest the campaign (the set of advertising rules), multivariate or A/B testing is used: different versions of the banner (i.e. different formats) are deployed to selected distinct groups of visitors and their effectiveness is measured, using an impact measure (various examples of such measures are given in (Cole, 2008)). When the test period is finished, the format which gave the best results is chosen. Obviously, this is a tedious process, made even more complicated if the set of banners is large.

A major drawback of this solution is that suboptimal versions waste advertising efficiency during the tests. The testing phase can be called exploration, since the various possibilities are explored, even the ones that will be revealed as inefficient later. Once the best format is identified, a second phase starts: the exploitation phase. Clearly, the exploration phase should be as short as possible in order to rapidly put the system into effective operation.

Instead of having the exploration and exploitation periods in a sequence, a better solution would be to combine them. This means that we may mix exploration and exploitation and make a trade-off between them: *explore* more in the beginning and *exploit* more and more with time.

In order to model the banner impact problem, it is natural to consider the following elements:

- a set of **states** representing the state of the context at instants when a decision about the banner can be issued,
- a set of **actions** that reflect these decisions,
- an **impact measure** that allows to evaluate the result of the actions.

In any state of the system an action (inserting a banner in a predefined format) can be taken and its impact is measured in the form of a reward. By trying the various possible actions, an optimal mapping between a state and an action may be learnt. This mapping is called an *optimal policy*. Later in this chapter we will formalize this problem within the reinforcement learning framework as a Multi-Armed Bandit Problem. For now, we only provide a quick intuition: it is like going to the casino and using slot machines. You try to optimize the revenues by choosing the machine that provides you with the best money return.

The right time to insert

Banner effectiveness is not only about the format, but also about timing. When should a banner be displayed during a user's browsing session?

Once again, the answer is not obvious. Displayed too early (in the beginning of a browsing session), the banner can be ignored by the user since his or her attention is much more focused on the content of the website than on ads. The phenomenon is called the "blindness problem" (e.g. it is studied in (Calisir & Karaali, 2008)). Displayed too late, the impact can be null, since the user may have already left the site. Therefore, if we are able to detect the precise moment during a user navigation to insert a banner, the impact will be more important.

In our previous work (Baccot et al., 2009), we have shown that adding a banner close to the end of a browsing session can lead to an increase of users' interest levels and lengthen sessions' duration.

Thus, our second problem is to choose the optimal instant for inserting the banner. Trying various instants (chosen at the beginning of a user browsing session) is a natural solution. After each try the system observes the impact and dynamically learns which action is the best. Therefore the system operates in a closed-loop.

Figure 1 generalizes this idea. The system is composed of a learning agent and its environment. The agent observes the context (i.e. users' preferences, terminals, network conditions etc.). Based on this observation, it builds a state. By following a policy, it decides whether or not to add a banner. After the banner is inserted, the system observes the impact of this action and collects a reward, measuring the effectiveness of the action. The policy is updated continuously according to the reward: fruitful actions are reinforced, whereas unfruitful ones are avoided. By following this trial-and-error process, the system is able to dynamically learn an optimal time insertion policy.

The level of dynamism goes further than the previous problem. However, the three main ingredients are still present: **states**, **actions** and **rewards**.

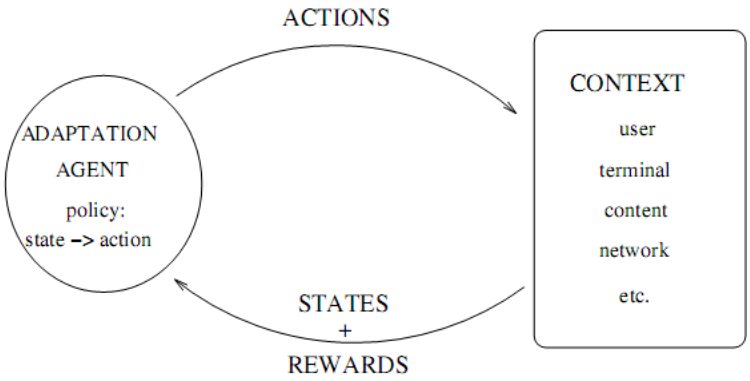

Figure 1. A decision-taking agent operating in a closed-loop.

The right sequence

Another advertisers' need is to decide the sequence of banner formats to be displayed during a user browsing session. At each step, the format of a banner that can be displayed needs to be chosen.

Intuitively, the sequence counts, since it may be counterproductive to visually stimulate heavily (and too much) the user with the first banner. Instead, an increasing power of stimulus may result in a better absorption of the advertisements message. Section 3 presents results from an experiment we led. The experiment was carried out on a real web site using three types of banner formats (figure 2):

- the basic version, only composed of a text (including links to recommended content),
- the video/avatar version. In that case, an avatar in a video serves as a teaser,
- the animated version. It uses a ``carousel''. Recommendations are included in the different facets.

For this experiment, we choose to determine which banner format sequence (basic, animated, video or video, animated, basic or etc.) is best according to various impact criteria.

More generally, we can state our third problem as a problem of sequentially deciding a set of banners. Like in the previous problems we propose to identify the **states**, the **actions** (insert a banner of a certain format) and the **rewards** (the CTR or the session duration). Users' navigations are closely monitored (the context is observed) and, at particular navigation moments, various banner format sequences are tested (possible actions are tried). Naturally, observed rewards (e.g. session duration) influence the subsequent taken actions and make the system learnt in a closed-loop manner.

Expressing a problem as a closed-loop system (see figure 1 previously presented) of such states, actions and rewards, fits well within the reinforcement learning (RL) framework, as we will see later.

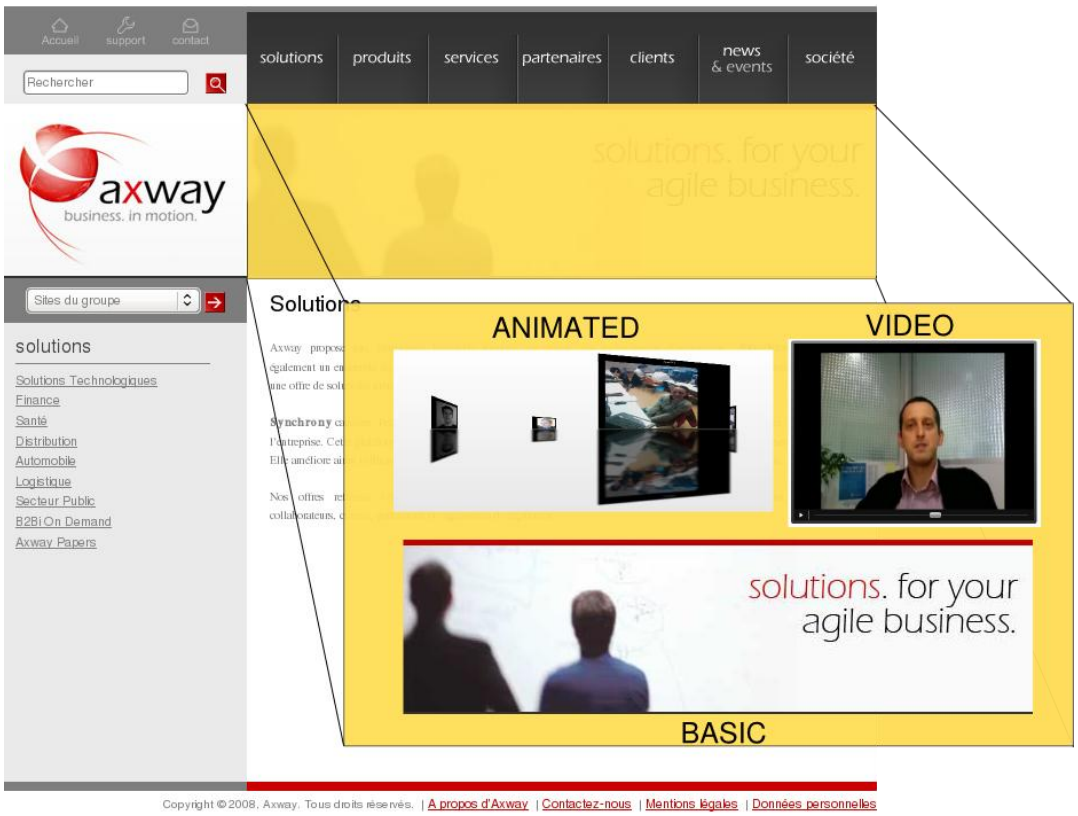

Figure 2. The different types of banner formats available for our test site.

Wrap up

The three problems have common ingredients. In each problem, states are defined. In a given state, different actions are available and one of them is chosen. The impact of the action is measured and rewards are obtained. Moreover, by using the closed-loop framework (figure 1) and a "trial-and-error" process, an optimal mapping between actions and states (called policy) can be dynamically learnt. The closed-loop also enables to take decision in sequence, allowing us to deal with the third problem.

All these elements are at the heart of the reinforcement learning framework. In the next section, we will present more precisely this framework and see how it can help to solve the three problems.

STOCHASTIC MODELS FOR BANNER OPTIMIZATION

As previously suggested, the dynamics of the web navigation context are heavily influenced by the variability of human and non-human factors. These dynamics are stochastic. In this section, we first introduce the reinforcement learning framework. Then we present two stochastic models that will be used to solve the banner problems.

The Reinforcement Learning Framework

According to (Sutton & Barto, 1998), reinforcement learning is "learning what to do (i.e. how to map situations to actions) so as to maximize a numerical reward signal". An agent, in a given situation, should try every possible action and discover by itself which are the ones that yield the most important reward. Action taking process is really important since actions affect immediate rewards, and, by modifying the environment, the next situations the agent will be in, and therefore, the subsequent rewards. Therefore the agent is able to maximize the long term (cumulative) reward.

Reinforcement learning is a sub-area of machine learning. Supervised and unsupervised learning are the two main types of machine learning. Reinforcement learning is close to unsupervised learning in that solutions do not necessarily need help from a human expert, nor sub-optimal actions are explicitly corrected. Further, there is a focus on on-line performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). The exploration vs. exploitation trade-off in reinforcement learning has been mostly studied through the multi-armed bandit problem.

Reinforcement learning has been successfully used in many applications, from robotics and control to game playing (the survey (Kaelbling et al., 1996) gives more details about the possible uses of RL).

In the context of the Web, we propose here to solve our banner problems using two stochastic models that make use of reinforcement learning: Markov Decision Processes (MDP) and Multi-Armed Bandit Problems (MABP). At some moments during a Web navigation, the system needs to decide "advertising actions" that maximize the utility criteria defined by a web marketer. A MDP sequentially decides of an advertising action at each step in the navigation. Bandit models are simpler and lighter, since only specific navigation states are considered and advertisement actions can only be computed for those states.

Markov Decision Processes

Markov Decision Processes are commonly used for solving sequential decision problems under stochastic conditions. A sequence of decisions is called a policy. In our case, users' interactions and network's performance are unpredictable, making our rich media banners operate under stochastic conditions.

A MDP is a stochastic controlled process that assigns rewards to transitions between states (Sutton & Barto, 1998). It is defined as a quintuple $(S; A; T; p_t; r_t)$ where S is the state space, A is the action space, T is the discrete temporal axis of instants when actions are taken, $p_t()$ are the probability distributions of the transitions between states and $r_t()$ is a given function of reward on the transitions. We rediscover in a formal way the previous ingredients (figure 1): at each instant

$t \in T$, the decisional agent (the advertising controller component) observes its state $\sigma \in S$, applies on the system the action $a \in A$ that brings the system (randomly, according to $p_t(\sigma'|\sigma,a)$) to a new state σ', and receives a reward $r_t(\sigma,a)$.

In this framework, we are looking for the best policy with respect to the accumulated rewards. A policy is a function π that associates an action $a \in A$ with each state $\sigma \in S$. Our aim is to find the best one, π^*. In each state σ, the optimal action $\pi^*(\sigma)$ maximizes the expected cumulated reward on the remaining temporal horizon. This cumulated reward may be defined as the sum of local rewards associated with the actual transitions. Other similar cumulative measures such as discounted sum or mean value are possible as well. In our case, since the model probabilities (i.e. $p_t()$) are not known, a reinforcement learning algorithms such as Q-learning can be used to find π^*.

In the Q-learning algorithm, Q-values are defined: $Q(\sigma,a)$ corresponds to the expected cumulated reward in state σ when action a is chosen. The principle of Q-learning (algorithm 1) is the following: after each observed transition (σ_n, a_n, σ_{n+1}, r_n) the current value function Q_n for the couple (σ_n,a_n) is updated, where σ_n represents the current state, a_n the chosen and applied action, σ_{n+1} the resulted state, r_n the immediate reward and R_n the accumulated reward for this experience.

The updating formula trades off previous estimation of Q_n and accumulated reward R_n for the current experience. The learning rate $\alpha_n(\sigma,a)$ is particular to each pair state-action and controls how fast we modify our estimates. One expects to start with a high learning rate (e.g. 1), which allows fast changes, and lowers the learning rate as time progresses.

Initialize Q_0
for $n = 0$ **to** $N_{tot} - 1$ **do**
    σ_n = chooseState
    a_n = chooseAction
    (σ'_n, r_n) = simulate(σ_n,a_n)
    /* update Q_{n+1} */
    $Q_{n+1} \leftarrow Q_n$
    /* compute the current expectation reward R_n */
    $R_n = r_n + \gamma \max_b Q_n(\sigma'_n,b)$
    $Q_{n+1}(\sigma_n,a_n) \leftarrow (1 - \alpha_n(\sigma_n,a_n))Q_n(\sigma_n,a_n) + \alpha_n(\sigma_n,a_n)R_n$
end for
return Q_{Ntot}

Algorithm 1. The Q-learning algorithm.

In this algorithm, N_{tot} is an initial parameter that represents the number of iterations. The function *simulate* returns a new state and its associated reward according to the dynamics of the system. The choice of the current state and of the action to execute is made by the functions *chooseState* and *chooseAction*. The function *initialize* is used most of the time to initialize the Q_0 values to *0*.

After N_{tot} iterations, it is easy to determine the best policy π^*: in each state σ, the action to take is the one that maximizes the Q-values, i.e. $\max_a(Q(\sigma,a))$.

A Markov Decisional Process is a good model for our banner problems, since:

- we can define a set of decisional states, representing states in the browsing session context,
- in each state we can choose among different banner actions (e.g. choice of the banner format),
- rewards are given by an impact/utility measure defined by a marketer (e.g. the session duration). The reward is cumulative: the impact of each banner format decision influences the user behavior and adds up to the total session duration.

Using a trial and error mechanism, Q-learning can be iteratively used to try the various banner actions and compute the optimal policy.

Multi-Armed Bandit Problems

A Multi-Armed Bandit Problem (MABP) is named by analogy to a slot machine. For example, in the *K*-Armed Bandit Problem, a gambler has to choose which of the *K* slot machines to play with. At each time step, he pulls the arm of one machine and receives a reward. His aim is to maximize the sum of rewards he perceives over time. This clearly shows the "exploration vs. exploitation" dilemma: the purpose of the gambler is to find, as rapidly as possible, the arm that gives the best expected reward.

In order to solve a Bandit problem (i.e. to find the best arm to play), various strategies can be used (Sutton & Barto gives a good overview in their book). Recent research has proposed various solutions to solve optimally and online this dilemma, minimizing the number of errors over time. One of the most efficient is called Upper Confidence Bound (UCB, by Auer et al., 2003). At each play, it computes a priority index for each arm, based on the previous rewards and the number of times it has already been invoked. The index p_j for arm j is given by

$$p_j = \overline{x_j} + \sqrt{\frac{2\ln(n)}{n_j}} \quad \text{(eq 1)}$$

where $\overline{x_j}$ is the average rewards obtained from arm j, n_j the number of times arm j has been chosen and n the overall number of plays done so far. The best arm to choose is the one with the highest priority.

In equation 1, $\overline{x_j}$ can be seen as the exploitation term (it uses the previous observed information about the arm) whereas the second term is an exploration term since it takes into account the number of times an arm has already been played with respect to the others.

For our banner problems, we can use MABPs as follows:

- we strategically define a set of decisional states,
- we associate one MABP to each state of the context,
- each arm corresponds to a possible advertising action in a given state,
- rewards are given by an impact/utility measure, defined by a marketer.

Figure 9 depicts a 6-armed Bandit problem associated to a state. Pushing arm *i* at a play means performing action *i* at this step.

As a result, we get a collection of independent multi-armed Bandit problems. Each MAPB can be solved independently using the UCB technique. One can notice that, compared to the MDP, this is done at the expense of neglecting potential relations between states.

EXPERIMENTS

A set of experiments was conducted on a website to test the various proposed models, focusing on the three banner problems.

In order to conduct these test rounds, we have built a generic software platform (figure 3). Our system is able to dynamically adapt website pages during a browsing session, and deploy either heuristic (fixed) policies or MDP/MABPs based model.

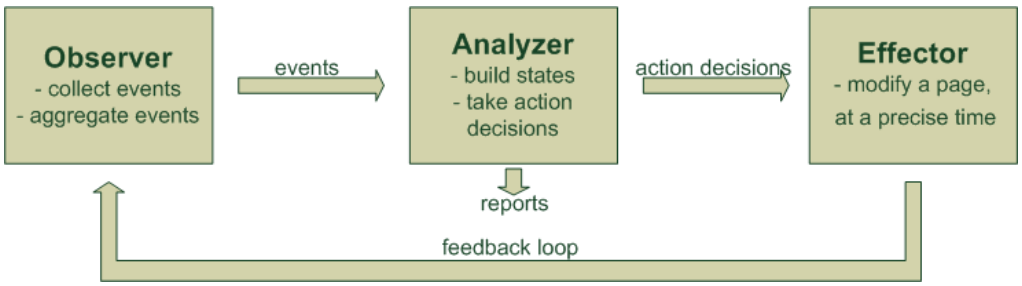

Figure 3. The adaptation platform that allows us to deploy both MDP and MABPs based models.

It has three main components. First, the user behavior is observed and his interactions with the web browser are collected (Observer). Interactions (mouse clicks, mouse movements...), called events, are sent back to a server and are analyzed in order to decide whether an adaptation should be done or not (Analyzer). The last component (Effector) performs the web page adaptation as decided by the Analyzer. The system works in a closed loop: when an adaptation is performed by the Effector, the user continues to be observed and the effect of the adaptation is measured by the Analyzer.

In this section, we describe how to use reinforcement learning techniques on a real website using this software platform. The site we choose is a showcase for a company that sells "Collaborative Business Solutions". The website has a section/subsection structure; each subsection presents a product or know-how of the company.

Case study 1: the right time to insert a banner

In this section, we study the problem of the right time when a banner should be inserted in a page. In this set of experiments, we look for the optimal instants for insertion during a user browsing session.

Concerning the displayed banner, we choose to focus on the delivery problem only and put aside the content problem. However, in order to present the users with consistent banners, we use the existing RSS feed of the test site as content provider (the feed is composed of links to the new products or white papers; these links are included in the banners). We also deliberately choose one banner format, by taking the basic one (presented in figure 2).

We first conduct a round of experiments where banners were inserted at different instants during the navigation. We used 170 testers and collected the insertion instants and their browsing session duration. What we realized is that adding the banner close to the end of the session can lengthen the session duration.

Then, we exploited these data to seed a simulator that was used to test our models. We can also use these data as "ground-truth" and compare them with the results obtained with the models.

A MDP solution

A MDP to model the problem is shown in figure 4.

The first thing we have to do is to model the different states. We chose each state to be composed of two variables: *T* and *s*. The state is therefore denoted as <*T*,*s*>. *T* is a positive integer that represents the current session duration time for a given user, after choosing a quantization for the browsing time of a user. *s* can have three values: *N* when the banner has not been displayed yet, *B* when the banner has been displayed but has not been followed (clicked on) by the user yet and *B+* when the banner has been displayed and has been followed by the user.

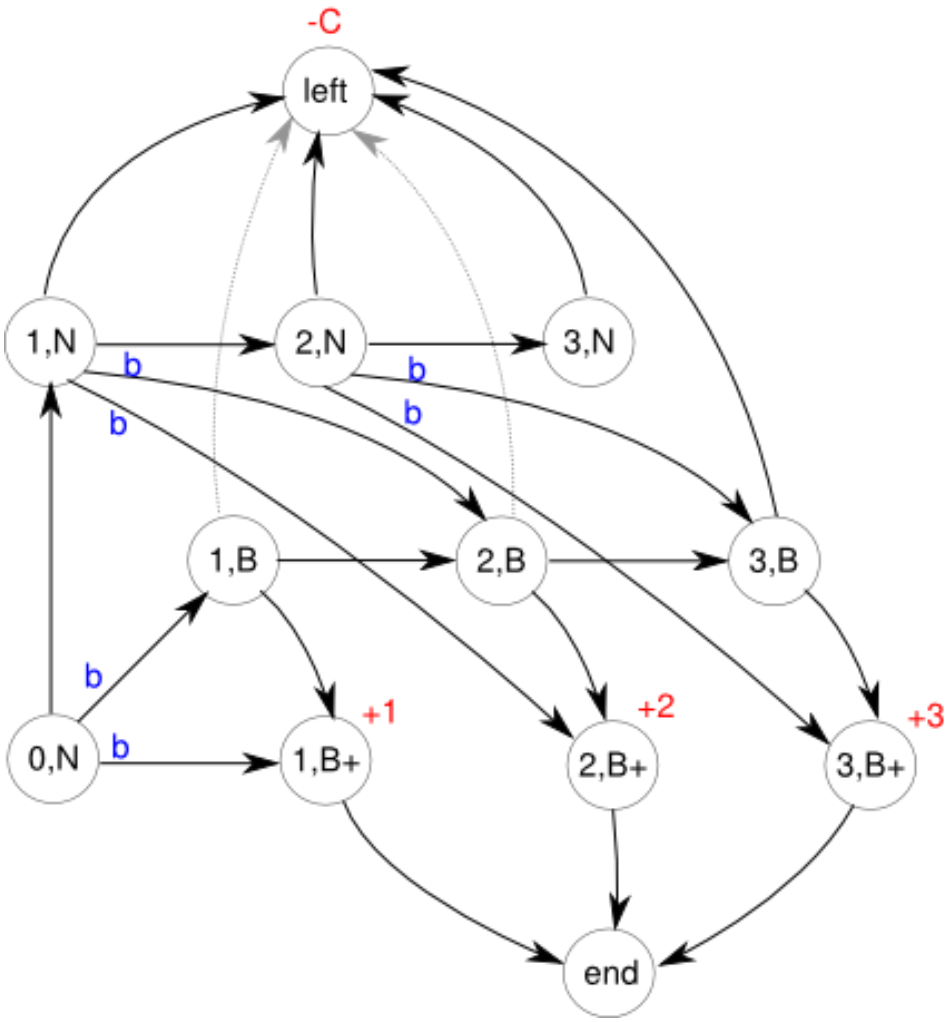

Figure 4. A MDP model to solve the optimal time problem. Rewards are written in red, actions in blue. Action Φ has been omitted for clarity reason.

Two states have a particular definition: *left*, reached when a user left the site without having seen or followed a banner and *end*, reached when a user has followed a banner.

Concerning the transitions between the states, there are three types, presented in figure 5. When the banner is not displayed at time *i*, it can be not displayed at time *i+1* (*N*), displayed (*B*) or directly followed by the user (*B+*). The user can also leave the website.

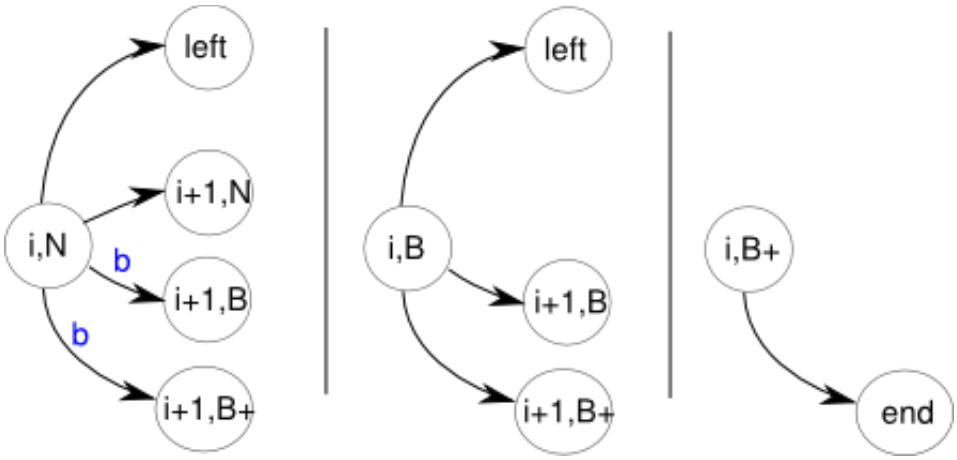

Figure 5. The three "generic" types of transition at time i.

As for the actions, they are of two types: *b*, the action that consists in adding the banner and *Φ*, an action that corresponds to the null action (nothing is done).

Thus, for example, being in state *<i,N>* means that the user has spent *i* time units on the website, and that the banner has not been displayed yet. In that state, two actions can be taken: adding a banner (*b*) that leads to *<i+1,B>*, *<i+1,B+>* or *left*, and *Φ* that leads to *<i+1,B>* or *left*.

Finally, let us present the rewards:

- *r(<i,B+>,a)=i*, for all actions *a*.
- *r(left,a)=-C*, for all actions *a*, where *C* is a positive integer.
- *r(s,a)=0* otherwise.

The rewards, that are higher if the banner is inserted at the latest possible instant, encourage the agent to try to insert the banner just before the end of the session, in order to get the maximum rewards. However, this is balanced by the fact that if the user prematurely left the website, the reward will be negative.

The model is not deterministic since when an action is taken in a given state, the resulting state does not only depend on the chosen action. In fact, the user can leave the site in any states, whatever the chosen action.

Table 1 shows the results (by setting the value of *C* to 4, which corresponds to the highest reward). They were obtained using the simulator and 10.000 Q-learning iterations. Only the actions taken at time 1, 2 and 3 are mentioned.

These results confirm the preliminary observed results and our intuition: adding the banner at the end of the user browsing session leads to longer session duration. Practically, it allows the system to identify the optimal, latest moment for the banner insertion.

Sequence	Average session duration
Φ, Φ, Φ	102s
b, Φ, Φ	120s
Φ, b, Φ	121s
Φ, Φ, b	146s

Table 1. Average session duration obtained for the different sequences of the PDM.

A MABP solution

We choose to use a very simple model, based on a single state. The associated Bandit problem has two arms (figure 6): *0s* and *120s*.

We choose an arm to pull at the beginning of user navigation. When arm *0s* is chosen, the ad is included immediately in the page (i.e. at the beginning of a user session) whereas when the other one is chosen, it is inserted 120 seconds after the beginning of the session. 120 seconds is chosen based on our preliminary results: average session duration was about 140 seconds.

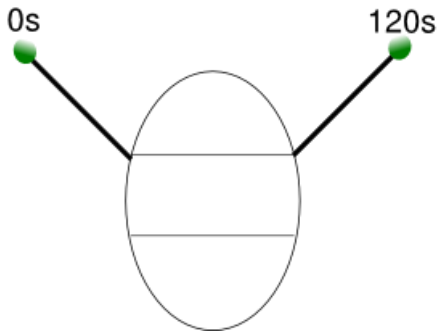

Figure 6. A Bandit model to solve the optimal time problem.

In table 2, the values of the priority indexes (computed according the UCB method, equation 1) for the two arms are presented at different iterations. The chosen action is the one with the greatest index, that is to say *120s* most of the time.

iteration	*0s*	*120s*
0	∞	∞
10	1.22	1.58
20	1.14	1.38
100	0.85	1.12

Table 2. Priority indexes for the two arms of the Bandit problem.

In figure 7, the percentage of optimal chosen action is depicted in function of the number of times the Bandit has been invoked. Quickly, after a few dozen browsing sessions, the chosen action is optimal (120 seconds), showing the efficiency of this Bandit problem.

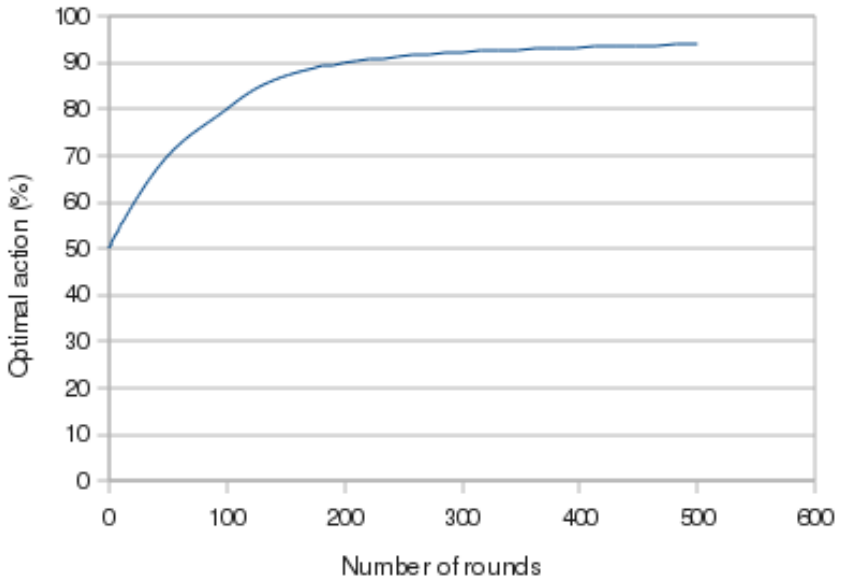

Figure 7. Percentage of optimal action chosen.

Again, results confirm what we already know: adding a banner at the end of a user browsing session leads to an increase in the session duration. This proves that MABPs may be used to solve such problems.

Comparison

The MDP solution to model the right insertion time is more complex than the Bandit Problem, but it allows a finer time granularity, at the expense of an increase in the number of states. If the number of states gets too large, it will be difficult to solve the MDP using methods such as Q-Learning (it requires an important amount of memory and needs a lot of training data, i.e. user navigations on the web site).

On the contrary, the Bandit based model is very simple but much more static since the decision to insert the banner is taken at the beginning of the session. Nevertheless, it is easy to add other arms corresponding to other instants of insertion.

Case study 2: the right sequence of banners

During navigation, a user may be presented, sequentially, with a set of banners. This is the third problem exposed in the introduction of this chapter. For each banner, we only focus on the format of the banner and put aside the content problem. Like in the previous section, in order to get an appropriate content for our test users, we choose to use the existing RSS feed of the web site as the content provider for recommendations.

Concerning the format of the banner, three types of banner formats are available (see figure 2):

- the basic version, only composed of a text (including links to recommended content),
- the video/avatar version. In that case, an avatar in a video serves as a teaser,
- the animated version. It uses a “carousel” component written in Adobe Flash. Recommendations are included in the different facets.

We choose to display banners in a sequence. A sequence contains exactly one banner of each format. Therefore, six sequences are available (basic/video/animated, video/basic/animated, etc.). Thus, adaptations will consist in displaying the banners in a certain order.

Among many possibilities, we chose to use as an objective/target of the adaptation to get users stay longer on the site, i.e. increase their session duration. We use the session duration as an impact measure (reward) for a given sequence.

In order to experimentally validate our approach, like in the previous case study, we use a navigation simulator that has been seeded with the data collected from 170 users from our previous work (Baccot et al., 2009) in which we also have sequences of three similar banner types. We have already concluded which sequence of banners is the best (basic/animated/video). Using this "ground truth", we want to determine whether our stochastic models rediscover or not this conclusion.

A MDP solution

We define a simple Markov Decision Process in order to model the problem (figure 8).

17 states are defined. Each state is composed of the previously added banner format. For instance, when state <*ABV*> is reached, an animated banner (*A*), a basic banner (*B*) and a video banner (*V*) have been displayed, in this order. There are also 2 specific states: *init* and *end*. At the beginning of a user session, the agent is in the state *init* and will reach the state *end* when a sequence of 3 banner formats have been displayed or when the user has prematurely left the website.

As for the adaptation actions, they consist in displaying a banner in a specific format. 3 actions are defined: *a* consists in adding an animated banner to the web page currently visited by a user, *b* a basic banner and *v* a video banner.

This model is also non-deterministic since knowing the chosen action is not sufficient to determine the next state (e.g. the user can leave the site whatever the action is).

Concerning the temporal axis, actions are taken when the user activity decreases. The user activity on the website corresponds to the number of events (mouse clicks, moves, scroll, etc...) done in a given time slot. We choose to add banners on activity decreases since users may be more receptive at these instants.

The last element to define concerns the rewards. When state *end* is reached, the given reward corresponds to the session duration of the user.

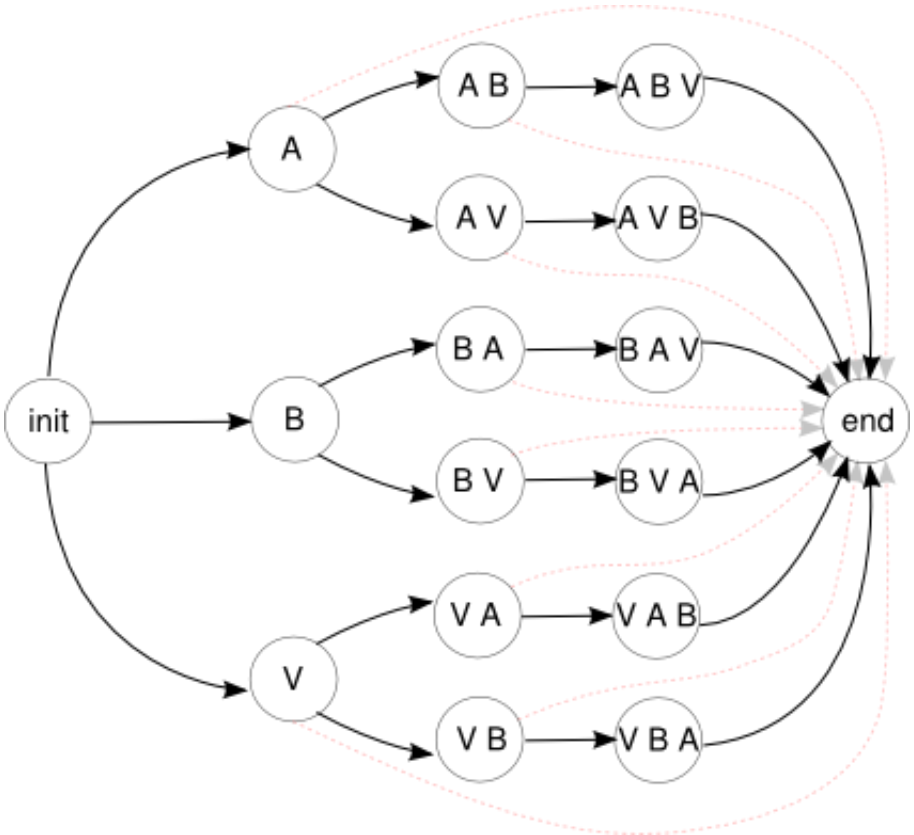

Figure 8. A PDM to solve the optimal banner format sequence problem.

We apply the Q-Learning algorithm (algorithm 1) to solve the problem and use our simulator. Table 3 presents the Q-values obtained after 10.000 iterations.

In a given state, the chosen action is the one with the highest Q-value. When we look more precisely at the table, the first action to choose (in the *init* state) is *b*. As a result, state <*B*> is reached, the action to choose is *a*; state <*BA*> is reached and the final action to choose is *v*, leading to *state* <*BAV*>.

Thus, the optimal sequence of banner format to choose is basic (*b*), animated (*a*) and video (*v*). This confirms the results we drew while studying the data collected during the preliminary experiment.

	0	a	b	v
init	0	140	**209**	170
A	37	0	145	176
B	65	**328**	0	145
V	39	173	167	0
AB	60	0	0	132
AV	95	0	35	0
BA	53	0	0	**423**
BV	54	166	0	0
VA	0	0	145	0
VB	136	165	0	0
ABV	230	0	0	0
AVB	115	0	0	0
BAV	**277**	0	0	0
BVA	112	0	0	0
VAB	230	0	0	0
VBA	148	0	0	0

Table 3. Q-values obtained after 10.000 iterations of the Q-Learning algorithm. In a given state, the optimal action to choose is the one with the highest Q-value.

A MAPB solution

First, we need to define the different states and the associated Bandit problems.

For the state, the simplest solution is to use a single state. It only contains an inferred information: the user "activity" (the number of events produced by the user in a given time window) on the website.
When the user activity decreases, a banner is displayed on the site according to the chosen sequence of banners.

As for the associated Bandit problems, actions are the different sequences to display, and thus we consider a 6-armed Bandit problem (figure 9).

The (stochastic) reward is given by the session duration.

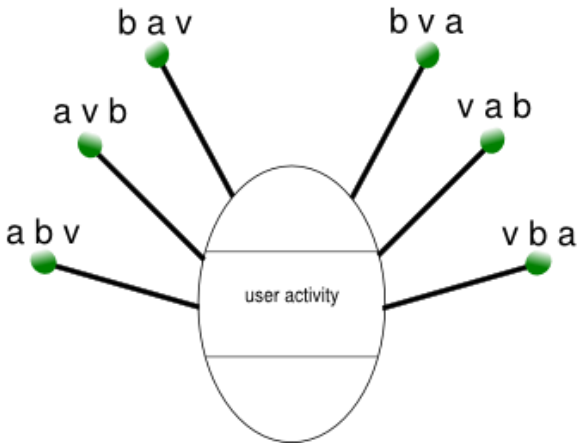

Figure 9. A Bandit-based model to solve the optimal banner format sequence problem. The six arms describe the possible six actions that can be done.

Figure 10 presents the evolution of priority indexes values for each arm of the associated Bandit. As the number of times the Bandit is invoked gets higher, priority indexes decreases. However, if we zoom into, we notice that the one for the sequence video/animated/basic (bold black line) is often greater than the others. It means that the associated arm is pulled more frequently. Using the ground truth, we realize that, indeed, this adaptation action is better that the others with respect to the session duration.

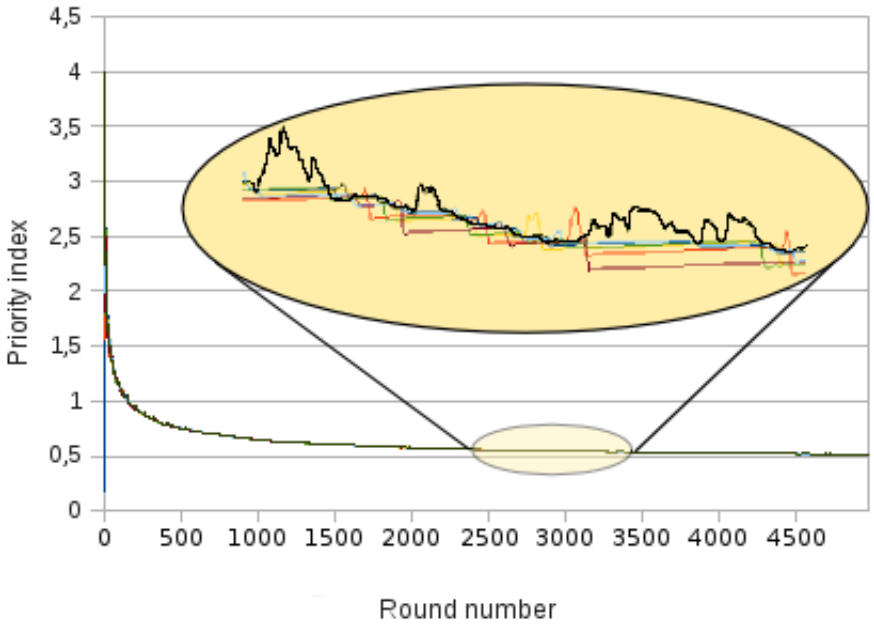

Figure 10. Evolution of priority indexes for each arm of the Bandit problem.

Figure 11 shows the percentage of the optimal action the Bandit has chosen as a function of the number of plays. Let us recall that we know which action is the best thanks to the original dataset. As the number of plays gets higher, the percentage increases, indicating the efficiency of the Bandit strategy. Interesting results are reached after around *10.000* sessions. On our test website, according to its popularity, this can be realized in less than a month.

Bandit problems allow us to draw a conclusion similar to the preliminary results: the usefulness of considering sequence for improving the effectiveness of banners. These results demonstrate the power and simplicity of a Bandit-based adaptation strategy.

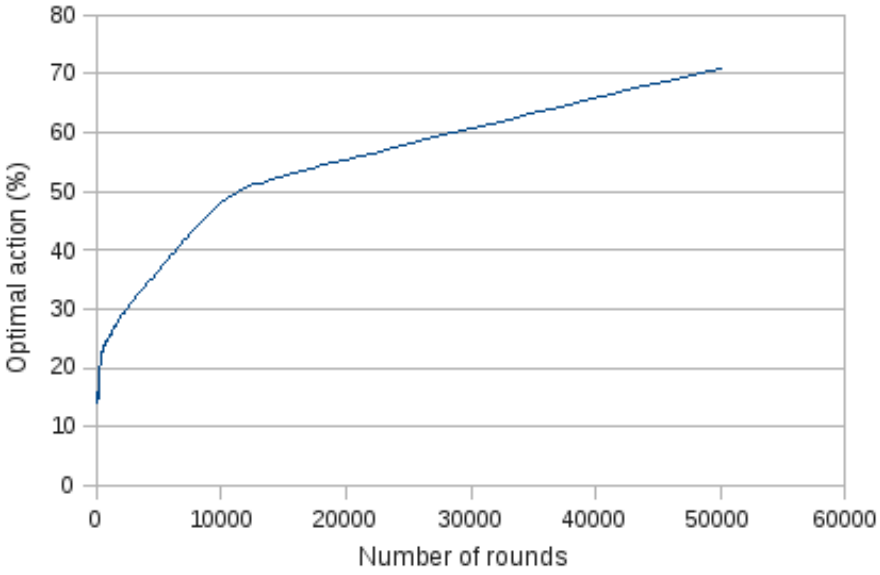

Figure 11. Percentage of optimal action chosen.

Comparison

The two models give similar results, but they are different in the approach. The MDP allows to issue a decision at each navigation step whereas the MABP-based model simply choses the whole set of actions at the beginning of the session. In other words, on the contrary of the MABPs, MDPs can adjust the advertising strategy in real-time during the navigation. Nevertheless, MDPs are more expensive to implement and solve, since their number of states is significantly larger. Both MDPs and MABPs realize a compromise between exploration and exploitation. In the MDP case, Q-learning allows to mix the two, while in the MABP case, UCB does the job.

DISCUSSION

We proposed in this chapter models that are able to provide a fully dynamic delivery policy using the reinforcement learning framework. During a user browsing session, states define instants when an action can be taken. The impact of the action is measured and has an effect in the subsequently taken actions. This is opposed to the classic policies that are static: in ad servers such as OpenX[1], delivery parameters and format for banners are chosen at the beginning of a user navigation.

Moreover, using the reinforcement learning, framework the policy can be learnt **online**, using a "trial-and-error" process. In a given situation, different actions are tested, their impact measured and the policy is updated using these measures. It goes a step further than policies that are set up offline.

One key problem is to deal with the "**exploration vs. exploitation**" trade-off. In widely used tools such as Google Website Optimizer[2], these two phases are separated: first the exploration is done (it generally consists in testing different versions of a delivery parameter), then an expert (a web marketer) decides to end the process, chooses a policy and starts the exploitation phase. Using the reinforcement learning framework, the trade-off can be naturally done by taking into account the rewards given by the impact measure.

Moreover, if there is a need to also provide recommendation content, reinforcement learning has also proved useful for providing recommendations on a websites (e.g. Shani et al., 2005).

While many authors focus solely on spatial optimization (banner format and position, e.g. Calisir et al.), we proposed to integrate in the models a full spatio-temporal control by adding the time dimension and taking into account the sequence of banners. In previous work, premises of this general idea have also been suggested by other authors. For example, Takashi et al., in 2002, used it in the context of streaming video and Madambath in 2009 mentioned it in a personal blog.

Finally, while currently fully empirical optimization based on marketers is used for banners delivery, we are able to propose a model-driven simulation and optimization, using any of previously presented stochastic models.

[1] OpenX Ad Server, http://www.openx.org
[2] Google Website Optimizer, http://www.google.com/websiteoptimizer

CONCLUSION

Rich-media banners on the web are probably the most demanding ad content, since they are more difficult to design, deploy and evaluate. Even if the approaches presented in this paper are tailored to meet such requirements, they can naturally be applied to classic text-based ad. Therefore we believe that applications such as keyword advertising can greatly benefit from using reinforcement learning methods.

The models we used have confirmed what we get while off-line analyzing data collected from some experiments. This underlines the power of the models. Moreover, they can easily be implemented and tested on-line using our platform, previously presented in figure 3. Thus, using such platform, web marketers can choose between:

- a set of fixed policies and conduct A/B tests to get feedback on their effectiveness.
- relaxing :) while the system computes automatically (online) an optimal policy using RL.

In many real-world environments, it will not be possible for the agent to have perfect and complete perception of the state of the environment. Unfortunately, complete observability is necessary for learning methods based on MDPs. In the future, we will consider the case in which the agent makes observations of the state of the environment, but these observations may be noisy and provide incomplete information. For example, subjective, implicit factors such as user's interest level can be taken into account when issuing a decision about the optimal banner. Therefore, future work needs to consider extensions to the basic MDP or MABP framework for solving partially observable problems. The resulting formal models are called Partially Observable Markov Decision Processes (POMDP) or Bandit Problems with side information.

REFERENCES

Marketing Sherpa (Ed.). 2008 Online Advertising Handbook. Marketing Sherpa.

McCoy, S., Everard, A., Polak, P., and Galletta, D. F. (2007). The effects of online advertising. *Communications of the ACM* 50, 3 (Mar. 2007), 84-88.

Rosenkrans G. (2009). The creativeness and effectiveness of online interactive rich media advertising. *Journal of Interactive Advertising*; *9(2),* 18-31.

Mei, T., Hua, X., Yang, L., & Li, S. (2007). VideoSense: towards effective online video advertising. In *Proceedings of the 15th ACM International Conference on Multimedia* (Augsburg, Germany, September 25 - 29, 2007). ACM, New York, NY, 1075-1084.

Baccot, B., Choudary, O., Grigoras, R., & Charvillat, V. (2009). On the impact of sequence and time in rich media advertising. In *Proceedings of the Seventeen ACM international Conference on Multimedia* (Beijing, China, October 19 - 24, 2009). ACM, New York, NY, 849-852.

Hauser, J.R., Urban, G.L., Liberali, G. & Braun, M. (2009). Website Morphing. *Marketing Science, 28(2)* (Mar. 2009), 224-224.

Cole, S.G., Spalding, L., & Fayer, A. (2009). The brand value of rich media and video ads. Technical report, DoubleClick Research.

Cole, S. (2008). Creative insights on rich media. Technical report, DoubleClick Research.

Calisir, F. & Karaali, D. (2008). The impacts of banner location, banner content and navigation style on banner recognition. *Computers in Human Behavior,* 24(2) (Mar. 2008), 535-543.

Sutton R. S. & Barto A. G. (1998). *Reinforcement learning: an introduction.* MIT Press.

Kaelbling, L.P., Littman, M.L., & Moore, A.W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research, 4*, 237-285.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2003). The Nonstochastic Multiarmed Bandit Problem. *SIAM Journal on Computing, 32(1)* (Jan. 2003), 48-77.

Shani, G., Heckerman, D., & Brafman, R. I. (2005). An MDP-Based Recommender System. *Journal of Machine Learning Research, 6* (Dec. 2005), 1265-1295.

Oshiba, T., Koike, Y., Tabuchi, M., & Kamba, T. (2002). Personalized advertisement-duration control for streaming delivery. In *Proceedings of the Tenth ACM International Conference on Multimedia* (Juan-les-Pins, France, December 01 - 06, 2002). ACM, New York, NY, 21-28.

Madambath, M. (2009). The idea of sequential advertising - moving beyond the context. Retrieved May 2, 2010 from http://www.watblog.com/2009/01/30/the-idea-of-sequential-advertising-moving-beyond-the-context.