A multiple-play bandit algorithm applied to recommender systems

Jonathan Louëdec, Max Chevalier, Josiane Mothe, Aurélien Garivier, Sébastien Gerchinovitz

A Multiple-Play Bandit Algorithm Applied to Recommender Systems

Jonathan Louëdec
Institut de Mathématiques de Toulouse
Institut de Recherche en Informatique de Toulouse
UMR5505 CNRS, Université de Toulouse
118 route de Narbonne 31062 Toulouse, France
jonathan.louedec@irit.fr

Max Chevalier and **Josiane Mothe**
Institut de Recherche en Informatique de Toulouse
UMR5505 CNRS, Université de Toulouse
118 route de Narbonne 31062 Toulouse, France
max.chevalier@irit.fr, josiane.mothe@irit.fr

Aurélien Garivier and **Sébastien Gerchinovitz**
Institut de Mathématiques de Toulouse
118 route de Narbonne 31062 Toulouse, France
aurelien.garivier@math.univ-toulouse.fr, sebastien.gerchinovitz@math.univ-toulouse.fr

Abstract

For several web tasks such as ad placement or e-commerce, recommender systems must recommend multiple items to their users—such problems can be modeled as bandits with multiple plays. State-of-the-art methods require running as many single-play bandit algorithms as there are items to recommend. On the contrary, some recent theoretical work in the machine learning literature designed new algorithms to address the multiple-play case directly. These algorithms were proved to have strong theoretical guarantees. In this paper we compare one such multiple-play algorithm with previous methods. We show on two real-world datasets that the multiple-play algorithm we use converges to equivalent values but learns about three times faster than state-of-the-art methods. We also show that carefully adapting these earlier methods can improve their performance.

1 Introduction

When a user interacts with a recommender system (RS), several items, such as ads, movies, news or songs, are recommended to him. The user can select recommended items, or not. The click-through rate allows to estimate the performance of a RS (Ricci, Rokach, and Shapira 2011).

In this paper we make the hypothesis that a recommended item selected by a user is relevant to her and that abandonment occurs when none of the recommended items is clicked through. In order to optimize RS, we thus consider the problem of minimizing abandonment.

To be as adaptive as possible, a RS should consider previous interactions with different users before providing recommendations to the current user. The problem is then to set up a strategy allowing to learn from item relevance while continuing to recommend relevant items to users. When the whole dataset is available at once, we can estimate all items' relevance : this is a supervised learning environment (Hastie, Tibshirani, and Friedman 2009). This is usually not the case in real RS: new users and new items occur continuously; moreover, the choice of the items to be recommended at each interaction must be decided with the information from past interactions only. Such an environment is called "reinforcement learning" (Sutton and Barto 1999). It requires implementing a strategy to gain information on the relevance of each item (exploration) while ensuring that the RS will continue to recommend relevant items (exploitation). This problem is known as the exploration/exploitation dilemma. Bandit algorithms are known to offer solutions to this dilemma (Bubeck and Cesa-Bianchi 2012).

Related work falls into two groups of models. The first one gathers feature-based models which deduce new information on unseen items or users from user-feedback. Some of them are designed to recommend a single item, such as the linear stochastic bandit approaches (Li et al. 2010) and (Chu et al. 2011), while others deliver several recommendations. In this sense, Yue and Guestrin (2011) propose the *LSBGreedy*, using upper confidence bounds on the estimated gain in utility, with a linear bandit approach. The second groups of models are feature-free models which deduce no information on unseen items or users. In this direction, Radlinski, Kleinberg, and Joachims (2008) developed the *Ranked Bandit Algorithm* (*RBA*) using as many single-play bandit algorithms as the number of recommendations to be supplied. *UCB*-type algorithms (Auer, Cesa-Bianchi, and Fischer 2002), the ϵ-*greedy* algorithm (Sutton and Barto 1999) or the *Exp3* algorithm (Auer et al. 2002) can be used as subroutine for *RBA*. More recently Kohli, Salek, and Stoddard (2013) created the *Independent Bandit Algorithm*(*IBA*). *RBA* and *IBA* are described in Section 3.

By using as many bandits as items to recommend, *RBA* and *IBA* do not fully exploit the combinatorial aspect of the problem, because single-play bandit algorithms are independent of each other. As a result, at each time, each single-play bandit algorithm learns about one item only. We argue that learning on all recommended items at the same time using a single algorithm is better; this addresses the multiple case directly.

Some recent theoretical works in the machine learning literature designed new algorithms to address this prob-

lem. Bubeck and Cesa-Bianchi (2012) proposed an algorithm capable of exploiting the combinatorial aspect of the bandit problem with multiple plays: *Exp3.M* (see also (Uchiya, Nakamura, and Kudo 2010)). While this algorithm has strong theoretical guarantees, it is a general algorithm for a wide class of complex bandit problems. For the convenience of the reader, we detail in Section 3.3 how we suggest to implement the algorithm of Bubeck and Cesa-Bianchi (2012) (Sections 5.3 and 5.4) in an efficient fashion. Our work takes place in the feature-free context, because at each time *t*, we consider the actual user as a new user and all items are independent.

The three approaches mentioned above (*RBA*, *IBA*, *Exp3.M*) do not exactly target the same solution. Ideally we would like to minimize abandonment. *RBA* aims at delivering a solution based on the diversity principle. *Exp3.M* and *IBA* aim at maximizing the sum of the clicks at each time. These solutions are suboptimal with respect to abandonment but are easier to find than the optimal solution.

In this paper, we briefly describe *RBA*, *IBA* and *Exp3.M* and their theoretical guarantees. Furthermore, we evaluate the *Exp3.M* algorithm on MovieLens and Jester collections. We show that the *Exp3.M* learns faster than the *RBA* and *IBA* approaches. Additionally, we show that using *Exp3* as a subroutine for *RBA* improves significantly the performance of the latter, but has no effect on *IBA* results.

This paper is organized as follows: in Section 2, we formalize the recommendation problem and define several suboptimal solutions that algorithms should target to solve it. In Section 3, we first detail the *RBA* and *IBA* algorithms, then the *Exp3.M* algorithm and the implementation that we propose, and compare the theoretical guarantees of each algorithm. In Section 4, we present the experiment framework and results. Finally we conclude this paper and indicate some directions for future work.

2 Problem Formalization

We consider a collection of K items noted I_i with $i \in \{1, \dots, K\}$. At each time t, m items are recommended to the user. If the user clicks through at least one item, we get a global payoff of 1, otherwise the payoff is 0. The objective is to minimize the fraction of 0's, i.e., to minimize abandonment. P_m^K is the set of all the combinations of m items that it is possible to get with K items. A user can be represented by a relevance vector $X = \{0,1\}^K$ where $X_i = 1$ if the user selects the item i. At each time t, the user is represented by X_t. A set of m items $A_t \in P_m^K$ is submitted to this user.

Z_t is the reward obtained for the set A_t with the relevance vector X_t. It is defined by :

$$Z_t = \max_{i \in A_t} X_{i,t}$$

Each component $i \in \{1, \dots, K\}$ of the vector X follows a Bernoulli distribution of unknown parameter p_i. The probability p_i can be estimated at time t by :

$$\hat{p}_i(t) = \frac{1}{N_i(t)} \sum_{t\,:\,i \in A_t} X_{i,t}$$

with

$$N_i(t) = \sum_{t\,:\,i \in A_t} 1$$

The fraction of users who consider at least one item of A_t as relevant is $E[Z_t]$, the expectation of the variable Z_t. Maximizing $E[Z_t]$ is equivalent to minimizing the abandonment. In this case an optimal set A^* is a set that leads to at least one click for a majority of users. A^* is defined as :

$$A^* = \operatorname*{argmax}_{A \in P_m^K} E[Z]$$

The purpose of RS in an online context can be defined as the minimization of the difference between $\sum_{t=1}^{T} Z^*$ (the sum of obtained rewards if we use the optimal set A^*) and $\sum_{t=1}^{T} Z_t$ (the sum of the rewards obtained with the algorithm). This difference is called the cumulative expected regret, denoted by $R(T)$ and is defined as :

$$R(T) = T \times E[Z^*] - \sum_{t=1}^{T} E[Z_t]$$

Finding an optimal set A^* is a NP-hard problem (Radlinski, Kleinberg, and Joachims 2008). A suboptimal solution, but easier to find, could be more appropriate. In this context Kohli, Salek, and Stoddard (2013) used two others solutions : the greedy solution and the independent solution.

The independent solution is based on the probability ranking principle (Robertson 1977). It is defined as the set consisting of the m items with the highest CTR.

$$A^{independent} = \operatorname*{argmax}_{A \in P_m^K} \sum_{i \in A} E[X_i]$$

This solution is targeted by the *IBA* and *Exp3.M* algorithms, both of them are designed to maximize :

$$Z_t^{independent} = \sum_{i \in A_t} X_{i,t}$$

The diversity principle is essential in recommendation, but it is not taken into account with the independent solution (Chen and Karger 2006). Let us take the example of movie recommendation : the most popular movies are those of the hexalogy "Star Wars". In this context, the most popular 6 items can be liked by users who have similar taste, representing the majority of users. But if a user has different tastes, none of the proposals will satisfy him. The optimal set A^* is based on the diversity principle, because it is composed of m items such that at least one of them is relevant to a majority of users.

The greedy solution is defined as the solution which ranks the most popular item at the first position and most popular items in the following positions given that the user has not selected previous items in the recommendation list. More formally, this solution can be defined as:

$$A_1^{greedy} = \operatorname*{argmax}_{i \in K} E[X_i]$$

and for $k \geq 2$,

$$A_k^{greedy} = \underset{i \in K / \left\{A_{1,\dots,k-1}^{greedy}\right\}}{\operatorname{argmax}} E\left[X_i | X_j = 0 \ \ \forall j \in \left\{A_{1,\dots,k-1}^{greedy}\right\}\right]$$

By taking most popular items given that the user has not clicked through any earlier items, the greedy solution takes into account the principle of diversity. As the greedy solution chooses in the first position the most popular item, this item does not necessarily occur in the optimal solution A^*, that is why A^{greedy} is a suboptimal solution.

3 Algorithms

In this section, we first present the two methods that will serve as benchmarks: *RBA* and *IBA*. The *RBA* approach targets the greedy solution while *IBA* targets the independent solution. Both methods use as a subroutine a single-play bandit algorithm; we detail the case when the subroutine is *Exp3* (Auer et al. 2002). This will enable a fair comparison with the multiple-play algorithm, *Exp3.M*, which detailed description is given in Section 3.3 in order to allow the reader to efficiently implement the algorithm of Bubeck and Cesa-Bianchi (2012). This algorithm targets the independent solution.

3.1 Ranked Bandits Algorithm (*RBA*)

RBA (Algorithm 1) was developed by Radlinski, Kleinberg, and Joachims (2008) and requires running m single-play K-arm bandit algorithms in parallel. At each time t, the m items chosen by the m bandits are recommended to the user. The way the bandits' information is updated works as follows : the bandit corresponding to the first clicked item gets a reward of 1, while all the remaining bandits get a reward of 0. This way, the first bandit tends to recommend the item with the highest CTR. Besides, since the second bandit can only get a reward when the first item is not clicked, it tends to recommend the item with the highest CTR when the first item is not relevant. And so on for the 3rd,..., m^{th} bandits.

As shown by Radlinski, Kleinberg, and Joachims (2008), *RBA* combined with the subroutine *Exp3* has a regret of $O(m\sqrt{TKlogK})$ with respect to the greedy solution and the rewards $Z_t = \max_{i \in A_t} X_{i,t}$.

3.2 Independent Bandits Algorithm (*IBA*)

IBA (Algorithm 2) was later developed by Kohli, Salek, and Stoddard (2013). Similarly to *RBA*, it requires to run m single-play K-arm bandit algorithms in parallel. The main difference between the two algorithms is how user clicks are taken into account: while for *RBA* only the first clicked item is considered as relevant, all clicked items are considered relevant by *IBA*. As a consequence, the k^{th} single-play bandit of *IBA* tends to recommend the item with the k^{th} highest CTR. The *IBA* approach thus targets the independent solution. Adapting the proof of Kohli, Salek, and Stoddard (2013), one can show that *IBA* combined with the *Exp3* subroutine has a regret of $O(m\sqrt{TKlog(K)})$ with respect to the independent solution and the rewards $Z_t = \sum_{i \in A_t} X_{i,t}$ (sum of clicks).

Algorithm 1: Ranked Bandit Algorithm

SPB$_i$: single-play bandit algorithm for recommendation i
for $t = 1, \dots, T$ **do**
  for $i = 1, \dots, m$ **do**
    $a_i \leftarrow$ selectItem(SPB$_i$, K)
    if $a_i \in \{a_1, \dots, a_{i-1}\}$ **then**
      $a_i \leftarrow$ arbitrary unselected item
    end
  end
  $A_t \leftarrow \cup_i a_i$
  Display A_t to user, receive feedback vector X_t
  for $i = 1, \dots, m$ **do**
    Feedback:

$$z_i = \begin{cases} 1 & \text{if item } a_i \text{ was the first clicked on} \\ 0 & \text{otherwise} \end{cases}$$

    update(SPB_i, z_i)
  end
end

Algorithm 2: Independent Bandit Algorithm

SPB$_i$: single-play bandit algorithm for recommendation i
for $t = 1, \dots, T$ **do**
  for $i = 1, \dots, m$ **do**
    $a_i \leftarrow$ selectItem(SPB$_i$, $K \backslash A_{t,i-1}$)
    $A_{t,i} \leftarrow A_{t,i-1} \cup a_i$
  end
  Display A_t to user, receive feedback vector X_t
  for $i = 1, \dots, m$ **do**
    Feedback:

$$z_i = \begin{cases} 1 & \text{if item } a_i \text{ was clicked on} \\ 0 & \text{otherwise} \end{cases}$$

    update(SPB_i, z_i)
  end
end
—

Usually[1], the fraction of users who select at least one item is smaller for the independent solution than for the greedy solution. Thus, in the long run, *RBA* is usually better than *IBA*. However Kohli, Salek, and Stoddard (2013) showed in their simulations that *IBA* learns faster at the beginning (the fraction of abandonment decreases faster than with *RBA*), which is an important feature for RS.

In both approaches all m single-play bandit algorithms are (almost) independent from one another. At each time t, each bandit learns about one item only. Thus, we can only expect that the overall method converges when all m single-play bandit algorithms have converged, which might yield an unnecessary large regret. On the contrary, the multiple-play bandit algorithm presented below manages all m items

[1]This is generally true, in some cases counter-examples can be cooked where the probability of click on the independent solution is higher.

Algorithm 3: *Exp3.M*

Init : $p_1 = (\frac{m}{K}, \dots, \frac{m}{K}) \in \mathbb{R}^K$

for *each time* $t \geq 1$ **do**

- draw $A_t \subset \{1, \dots, K\}$ at random such that $\mathbb{P}[i \in A_t] = p_{i,t}$ for all $i = 1, \dots, K$
- submit A_t to the user and receive rewards $X_{i,t}$ for all $i \in A_t$
- compute $q_{t+1} \in \mathbb{R}^K$ as

$$q_{i,t+1} = m \, \frac{p_{i,t} \, \exp(\eta \tilde{X}_{i,t})}{\sum_{1 \leq j < K} p_{j,t} \, \exp(\eta \tilde{X}_{j,t})} \quad (1)$$

  where $\tilde{X}_{i,t} = \frac{X_{i,t}}{p_{i,t}} \mathbb{1}_{i \in A_t}$

- update: compute $p_{t+1} \in \mathbb{R}^K$ by

$$p_{i,t+1} = \min\{C q_{i,t+1}, 1\} \quad (2)$$

  where C is such that $\sum_{i=1}^K \min\{C q_{i,t+1}, 1\} = m$

end

simultaneously, hence it should reach a solution much faster.

3.3 *Exp3.M*

The *Exp3.M* algorithm, which we state as Algorithm 3 below, is a recent bandit algorithm specifically designed to address the multiple-play case. Its target is the independent solution defined in Section 2. It was first proposed and theoretically studied by Uchiya, Nakamura, and Kudo (2010). More recently Bubeck and Cesa-Bianchi (2012) analyzed a generalization of this algorithm for a wide class of complex bandit problems known as *combinatorial bandits*. In this section we describe the *Exp3.M* algorithm and our adaptation of some components in order to allow the readers to implement efficiently this algorithm.

Main mathematical intuitions. The Algorithm 3 relies on the *online mirror descent* principle, a general methodology to minimize regret in online convex optimization. More precisely, Algorithm 3 can be seen as a particular case of the *Online Stochastic Mirror Descent* algorithm of (Bubeck and Cesa-Bianchi 2012, Sections 5.3 and 5.4) used with the *negative entropy* function $F(p) = \sum_{i=1}^K (p_i \log(p_i) - p_i)$ on the convex set $\mathcal{C}_m = \{p \in [0,1]^K : \sum_i p_i = m\}$. Indeed, steps (1) and (2) of Algorithm 3 are equivalent to the two online mirror descent updates:

$$\nabla F(q_{t+1}) = \nabla F(p_t) + \eta \tilde{X}_t$$
$$p_{t+1} = \operatorname*{argmin}_{p \in \mathcal{C}_m} D_F(p, q_{t+1}) \, ,$$

where $D_F(p,q) = \sum_{i=1}^K (p_i \log(p_i/q_i) + q_i - p_i)$ is the Bregman divergence associated to the function F. In other words, to compute the new vector p_{t+1}, the *Exp3.M* algorithm starts from p_t, then moves along the direction of the estimated reward vector $\tilde{X}_t = (\tilde{X}_{i,t})_{1 \leq i \leq K}$, and finally projects it back onto the convex set $\mathcal{C}_m$.

Subalgorithm 1: Dependent Rounding

Input : Vector $p \in \mathbb{R}^K$ of probabilities with $\sum_{i=1}^K p_i = m$

Output : Subset of $[K]$ with m items

while *there is an* i *with* $0 < p_i < 1$ **do**

- Choose distinct i, j with $0 < p_i < 1$, $0 < p_j < 1$
- Set $\alpha = \min\{1 - p_i, p_j\}$ and $\beta = \min\{p_i, 1 - p_j\}$
- Update p_i and p_j as

$$(p_i, p_j) = \begin{cases} (p_i + \alpha, p_j - \alpha) \text{ with probability } \frac{\beta}{\alpha+\beta} \\ (p_i - \beta, p_j + \beta) \text{ with probability } \frac{\alpha}{\alpha+\beta} \end{cases}$$

end

return $\{i : p_i = 1, 1 \leq i \leq K\}$

Efficient implementation. As explained below, the proposed implementation requires only $\mathcal{O}(K \log K)$ elementary operations (flops) at each time t. Two points can be highlighted:

- There are several ways to draw a set $A_t \subset \{1, \dots, K\}$ at random in order that $\mathbb{P}[i \in A_t] = p_{i,t}$ for all $i = 1, \dots, K$. A naive approach consists in noting that $p_t \in \mathcal{C}_m$ can be decomposed as a convex combination of the form $\sum_A \alpha_A \mathbb{1}_A$ (where the subsets $A \subset \{1, \dots, K\}$ have cardinality m), so that it is enough to draw a set A at random with probability α_A. Unfortunately, the computational complexity of this step would be at least of the order of $\binom{K}{m}$ flops. Instead we use the *Dependent Rounding* function*(Subalgorithm 1)* designed by Gandhi et al. (2006) and Uchiya, Nakamura, and Kudo (2010) since it only requires $\mathcal{O}(K)$ flops.
- The second issue is how to compute the numerical constant C defined implicitly by the equation $\sum_{i=1}^K \min\{C q_{i,t+1}, 1\} = m$. It turns out that after sorting the components $q_{i,t+1}$, $i = 1, \dots, K$, in decreasing order, this problem boils down to successively solving elementary one-variable linear equations of the form:

$$k + \sum_{i=k+1}^{K} C v_i = m \, , \quad \text{with } C \in [1/v_k, 1/v_{k+1}]$$

  where $v_1 \geq v_1 \geq \dots \geq v_K$ are the values $q_{i,t+1}$ sorted in decreasing order. The overall associated computational complexity is thus only of order $\mathcal{O}(K \log K)$.

Theoretical guarantees. Uchiya, Nakamura, and Kudo (2010) proved that *Exp3.M* has a regret of $O(\sqrt{mTKlog(K/m)}))$ if the rewards are defined by $Z_t = \sum_{i \in A^t} X_{i,t}$ (total number of clicks); see also Theorem 5.8 of Bubeck and Cesa-Bianchi (2012). This regret bound improves by a factor of $\sqrt{m}$ on the regret bound we could derive for *IBA* combined with m instances of the *Exp3* algorithm.

4 Evaluation

Experimental framework. To evaluate the adapted implementation of the *Exp3.M* algorithm and compare it to

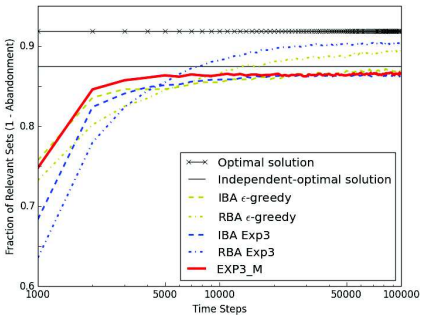

Figure 1: MovieLens dataset with relevance threshold = 2

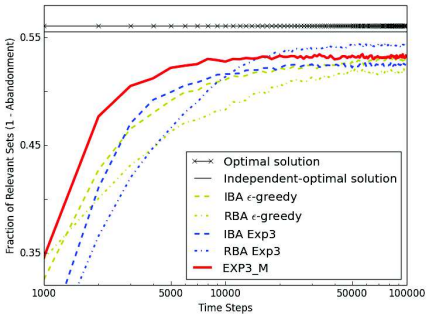

Figure 2: MovieLens dataset with relevance threshold = 4

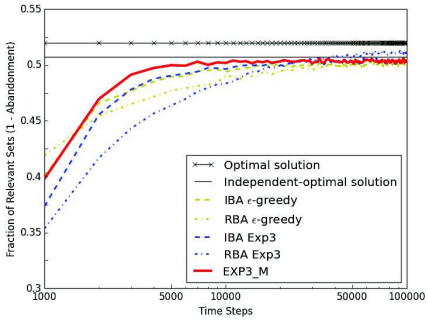

Figure 3: Jester dataset with relevance threshold = 7

state-of-the-art methods, we carried out several experiments with the MovieLens-100 and the Jester datasets which are studied in the paper of Kohli, Salek, and Stoddard (2013).

The first dataset, MovieLens-100, contains 943 users who rated 100 movies. If a movie is not rated by a user, it is considered as not liked. A movie can be rated between 1 (bad) and 5 (good). To translate these rates into user clicks, Kohli, Salek, and Stoddard (2013) chose to set up a threshold of relevance which can be set at different values. In their experiments, they used two values: 4 and 2. When the threshold is 4, for a given user, all the movies that have rates strictly greater than 4 are considered as relevant. The same method is applied when the threshold is set to 2.

The Jester dataset contains 25000 users who rated 100 jokes. If a joke is not rated, it is considered as not relevant. A joke can be rated between -10 (not funny) and 10 (very funny). Kohli, Salek, and Stoddard (2013) used the threshold of relevance 7 to identify relevant recommendation.

To simulate the online aspect of RS, at time t, one user is chosen randomly and m=5 documents are recommended to him. Moreover, the chosen user is considered as a new user. The user is said to have click through an item only if the associated rate is strictly greater than the threshold. If the user clicks on one document or more, we obtain $Z_t = 1$, otherwise $Z_t = 0$. Our objective is to maximize the sum of Z_t, i.e., minimize abandonment as described in Section 2. Each experiment is carried out on a time interval of length T = 100000. The final results are arithmetic averages of the results of 200 Monte-Carlo experiments. Every 1000 steps, the average of Z_t on the past 1000 rounds is displayed.

To get a fair comparison of the multiple-play bandit algorithm *Exp3.M* with *RBA* and *IBA*, we choose to implement them with the *Exp3* algorithm (Auer et al. 2002) and the ϵ-*greedy* algorithm (Auer et al. 2002).

With the *Exp3* algorithm, a probability of $1/K$ is initially allocated to each item. According to these probabilities, an item is randomly recommended. Based on the user-feedback, probabilities are updated more or less aggressively depending on a parameter η. We chose to use the *Exp3* algorithm because it is the algorithm we used to compare theoretically *Exp3.M* with *RBA* and *IBA*. The value of the η parameter needs to be tuned. For this, we tested a grid of values 2^i with $i \in [-7, -6, \ldots, 0]$ and we chose the value with the best compromise between the learning rate and the average value of abandonment at the end of experiments. The value of η has been tuned for *RBA* and *IBA* independently. We used $\eta = 2^{-6}$ for both approaches.

The second one, the ϵ-*greedy* algorithm, adds a degree of randomness for the chosen item. At time t, a uniformly random item is recommended with a probability of ϵ or, with a probability of $(1 - \epsilon)$, the item with best CTR is recommended. This algorithm is the subroutine used in *RBA* and *IBA* that gives the best results in the paper by (Kohli, Salek, and Stoddard 2013). In their paper, the authors indicate that $\epsilon = 0.05$ give the best average during their initial tests.

The *Exp3.M* algorithm also needs to be tuned. Using the same method as for the *Exp3* algorithm, we chose $\eta = 2^{-5}$.

Our experimental results are displayed in Figures 1, 2 and 3. In these Figures, we can observe the optimal solution and the independent solution. The greedy solution is the same as the optimal solution in our experimentation. So all *RBA* applications target the optimal solution. *IBA* applications and the *Exp3.M* algorithm aim at the independent solution. The difference between the independent solution and the optimal solution is small (see Figures 2 and 3), and is about 1%. However, in Figure 1, this difference is about 4%.

Results. Let us first compare the *Exp3.M* algorithm with state-of-the-art methods. The *Exp3.M* converges to equivalent values than the *IBA*-ϵ-*greedy* approach, the state-of-the-art method with the best learning rate. Furthermore, the *Exp3.M* algorithm learns faster. To highlight the learning rate improvement, we use the one-sided version of the Wilcoxon test (see Table 1). The *Exp3.M* algorithm learns 1.5 - 6 times faster than state-of-the-art methods. To assess the improvement of the learning rate, we compare the num-

		95 %	98 %
collection	approach	p-value	p-value
MovieLens threshold 2	**IBA-Egreedy**	0.054	0.0012
	IBA-Exp3	4.2e-09	<2.2e-16
	RBA-Egreedy	2.9e-13	2.9e-15
	RBA-Exp3	<2.2e-16	<2.2e-16
MovieLens threshold 4	**IBA-Egreedy**	1.0e-08	0.0016
	IBA-Exp3	9.4e-15	7.6e-07
	RBA-Egreedy	<2.2e-16	1.0e-10
	RBA-Exp3	<2.2e-16	1.1e-07
Jester threshold 7	**IBA-Egreedy**	8.8e-06	8.7e-09
	IBA-Exp3	2.4e-14	1.9e-14
	RBA-Egreedy	<2.2e-16	<2.2e-16
	RBA-Exp3	<2.2e-16	<2.2e-16

Table 1: Learning rate improvement: One-sided Wilcoxon test. Alternative hypothesis: the *Exp3.M* algorithm learns faster.

collection	approach	ratio (95%)	ratio (98%)
MovieLens threshold 2	**IBA-Egreedy**	1	3
	IBA-Exp3	1.5	2
	RBA-Egreedy	2	2
	RBA-Exp3	2	1
MovieLens threshold 4	**IBA-Egreedy**	7	—
	IBA-Exp3	—	—
	RBA-Egreedy	—	—
	RBA-Exp3	2	—
Jester threshold 7	**IBA-Egreedy**	1.5	3
	IBA-Exp3	1.5	2
	RBA-Egreedy	2.5	6
	RBA-Exp3	3	4

Table 2: Approximate ratios between the number of steps required to reach 95% and 98% of the independent solution by state-of-the-art methods and the number of steps required to reach 95% and 98% of the independent solution by *Exp3.M*.

ber of steps required by curves in Figures 1, 2 and 3 to reach 95% and 98% (see Table 2).

On the other hand we compare the implementation of *RBA* and *IBA* with the ϵ-*greedy* algorithm, as implemented in the paper by Kohli, Salek, and Stoddard, with our implementation with the *Exp3* algorithm. The *RBA-Exp3* has a proportion of (1 - abandonment) greater than *RBA-Egreedy* but only after approximately 4000 time steps for Movielens and 10000 time steps for Jester. Regarding the *IBA-Exp3* approach and *IBA-*ϵ*-greedy* approach, performances of both implementations are very close in Figures 1, 2 and 3.

In a long run, the *RBA-Exp3* approach is still the best. In Figure 1, the *RBA-Exp3* approach outperforms the independent solution after 8000 time steps, and after approximately 50000 steps in Figure 3. In Figure 2 the *RBA-Exp3* approach does not outperform the independent solution, but it is the only one approach capable of reaching 98% of the independent solution on average. Nevertheless its learning rate is slower than *IBA* approaches and the *Exp3.M* algorithm.

5 Conclusion

We have assessed a bandit algorithm designed specially for the multiple-play context, such as in recommender systems: *Exp3.M*. Usually state-of-the-art approaches use as many bandits as items to recommend. By managing all items simultaneously, the *Exp3.M* algorithm converges to equivalent values and learns significantly faster (about 3 times) than the state-of-the-art method with the best learning rate, *IBA*. The experiments were conducted on two benchmark datasets.

The *Exp3.M* is designed to maximize the sum of clicks. So, when minimizing abandonment is expected, *Exp3.M* looks for a suboptimal solution. We have shown that the *RBA* approach with a careful adaptation we propose can be more interesting in the long run. In future work we plan to investigate about an adaptation of *Exp3.M* capable of targeting the greedy solution, like *RBA* does.

References

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 2002. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing* 32(1):48–77.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2-3):235–256.

Bubeck, S., and Cesa-Bianchi, N. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *CoRR* abs/1204.5721.

Chen, H., and Karger, D. R. 2006. Less is more: probabilistic models for retrieving fewer relevant documents. In *29th international ACM SIGIR conference on Research and development in information retrieval*, 429–436.

Chu, W.; Li, L.; Reyzin, L.; and Schapire, R. E. 2011. Contextual bandits with linear payoff functions. In *International Conference on Artificial Intelligence and Statistics*, 208–214.

Gandhi, R.; Khuller, S.; Parthasarathy, S.; and Srinivasan, A. 2006. Dependent rounding and its applications to approximation algorithms. *J. ACM* 53(3):324–360.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *The Elements of Statistical Learning*. Springer, 2nd edition.

Kohli, P.; Salek, M.; and Stoddard, G. 2013. A fast bandit algorithm for recommendation to users with heterogenous tastes. In *27th AAAI Conference on Artificial Intelligence*, 1135–1141.

Li, L.; Chu, W.; Langford, J.; and E.Schapire, R. 2010. A contextual-bandit approach to personalized news article recommandation. In *Proc. of 19th International World Wide Web Conference*, 661–670.

Radlinski, F.; Kleinberg, R.; and Joachims, T. 2008. Learning diverse rankings with multi-armed bandits. In *25th International Conference on Machine Learning*, 784–791.

Ricci, F.; Rokach, L.; and Shapira, B. 2011. Introduction to recommender systems handbook. In *Recommender Systems Handbook*. Springer. 1–35.

Robertson, S. E. 1977. The probability ranking principle in ir. *Journal of documentation* 33(4):294–304.

Sutton, R. S., and Barto, A. G. 1999. Reinforcement learning. *Journal of Cognitive Neuroscience* 11(1):126–134.

Uchiya, T.; Nakamura, A.; and Kudo, M. 2010. Algorithms for adversarial bandit problems with multiple plays. In *Algorithmic Learning Theory*, LNCS Springer. 375–389.

Yue, Y., and Guestrin, C. 2011. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems*, 2483–2491.