Bandit Algorithm for Both Unknown Best Position and Best Item Display on Web Pages

Camille-Sovanneary Gauthier, Romaric Gaudel, Elisa Fromont

Bandit Algorithm for Both Unknown Best Position and Best Item Display on Web Pages

Camille-Sovanneary Gauthier[1,3], Romaric Gaudel[2], and Elisa Fromont[3]

[1] Louis Vuitton, Paris (FR) camille-sovanneary.gauthier@louisvuitton.com
[2] Univ. Rennes, Ensai, CNRS, CREST, Rennes (FR) romaric.gaudel@ensai.fr
[3] Univ. Rennes, IUF, Inria, IRISA, Rennes (FR) elisa.fromont@irisa.fr

Abstract. Multiple-play bandits aim at displaying relevant items at relevant positions on a web page. We introduce a new bandit-based algorithm, PB-MHB, for online recommender systems which uses the Thompson sampling framework with Metropolis-Hastings approximation. This algorithm handles a display setting governed by the position-based model. Our sampling method does not require as input the probability of a user to look at a given position in the web page which is difficult to obtain in some applications. Experiments on simulated and real datasets show that our method, with fewer prior information, delivers better recommendations than state-of-the-art algorithms.

Keywords: Multi-armed Bandit, Position-Based Model, Thomson Sampling, Metropolis-Hasting.

1 Introduction

An online recommender systems (ORS) chooses an item to recommend to a user among a list of N potential items. The relevance of the item is measured by the users' feedback: clicks, time spent looking at the item, rating, etc. Since a feedback is only available when an item is presented to a user, the ORS needs to present both attractive items (a.k.a. *exploit*) to please the current user, and some items with an uncertain relevance (a.k.a. *explore*) to reduce this uncertainty and perform better recommendations to future users. It faces the *exploration-exploitation dilemma* expressed by the *multi-armed bandit setting* [3].

On websites, online recommender systems select L items per time-stamp, corresponding to L specific positions in which to display an item. Typical examples of such systems are (i) a list of news, visible one by one by scrolling; (ii) a list of products, arranged by rows; or (iii) advertisements spread in a web page. To be selected (clicked) by a user in such context, an item neds to be relevant by itself, but also to be displayed at the right position. Several models express the way a user behaves while facing such a list of items [23,9] and they have been transposed to the bandit framework [15,14].

Retailers often spread their commercials over a web page or display their items on several rows all at once. Thus, in this paper, we will focus on the *Position-Based Model* (PBM) [23]. This model assumes that the probability to

click on an item i in position ℓ results only from the combined impact of this item and its position: items displayed at other positions do not impact the probability to consider the item at position ℓ. PBM also gives a user the opportunity to give more than one feedback: she may click on all the items relevant for her. It means we are facing the so-called *multiple-play* (semi-)bandit setting [5]. PBM setting is particularly interesting when the display is dynamic, as often on modern web pages, and may depend on the reading direction of the user (which varies from one country to another) and on the ever-changing layout of the page.

Contribution. We introduce PB-MHB (Position Based Metropolis-Hastings Bandit), a bandit algorithm designed to handle PBM with a Thompson sampling framework. This algorithm does not require the knowledge of the probability of a user to look at a given position: it learns this probability from past recommendations/feedback. This is a strong improvement w.r.t. previous attempts in this research line [13,17] as it allows the use of PB-MHB in contexts where this information is not obvious. This improvement results from the use of the Metropolis-Hastings framework [22] to sample parameters given their true *a posteriori* distribution, even thought it is not a usual distribution. While Markov Chain Monte Carlo methods are well-known and extensively used in Bayesian statistics, they were rarely used for Thomson Sampling [12], [10], [24], [21] and it is the first time that the Metropolis-Hastings framework is used in the PBM setting. Besides this specificity, we also experimentally show that PB-MHB suffers a smaller regret than its competitors.

The paper is organized as follows: Section 2 presents the related work and Section 3 precisely defines our setting. PB-MHB is introduced in Section 4 and is experimentally compared to state-of-the-art algorithms in Section 5. We conclude in Section 6.

2 Related Work

PBM [9,23] relies on two vectors of parameters: $\boldsymbol{\theta} \in [0,1]^N$ and $\boldsymbol{\kappa} \in [0,1]^L$, where $\boldsymbol{\theta}_i$ is the probability for the user to click on item i when she observes that item, and $\boldsymbol{\kappa}_\ell$ is the probability for the user to observe the position ℓ. These parameters are unknown, but they may be inferred from user behavior data: we need to first record the user feedback (click vs. no-click per position) for each set of displayed items, then we may apply an *expectation-maximization* framework to compute the maximum a posteriori values for $(\boldsymbol{\theta}, \boldsymbol{\kappa})$ given these data [7].

PBM is transposed to the bandit framework in [13,14,17]. [13] and [17] propose two approaches based on a Thompson sampling (TS) framework, with two different sampling strategies. [17] also introduce several approaches based on the *optimism in face of uncertainty* principle [3]. However, the approaches in [13,17] assume $\boldsymbol{\kappa}$ known beforehand. [14] proposes the only approach learning both $\boldsymbol{\theta}$ and $\boldsymbol{\kappa}$ while recommending but it still requires the $\boldsymbol{\kappa}_\ell$ values to be organized in decreasing order, which we do not require. Note also that the corresponding approach is not based, as ours, on Thompson sampling.

While more theoretical results are obtained when applying the *optimism in face of uncertainty* principle to the bandit setting, approaches based on TS are known to deliver more accurate recommendations [1,4,6,12]. The limitation of TS is its requirement to draw from 'exotic' distributions when dealing with complex models. By limiting themselves to a setting where $\boldsymbol{\kappa}$ is known, [13] and [17] face simpler distributions than the one which arises from $\boldsymbol{\kappa}$ being unknown. In the following, we propose to use Metropolis-Hastings framework to handle this harder distribution. [10,24] investigate a large range of distribution approximation strategies to apply TS framework to the distributions arising from the *contextual bandit* setting, and [12,21] apply approximate sampling to other settings. Overall, these articles handle a pure bandit setting while we are in a semi-bandits setting: for each recommendation we receive as reward a list of 1 or 0 (click or not). As most of commercial website can track precisely on which product each client clicks, we aim at exploiting that fine-grain information.

The *cascading model* (CM) [9] is another popular model of user behavior. It assumes that the positions are observed in a known order and that the user leaves the website as soon as she clicks on an item[4]. More specifically, if she clicks on the item in position ℓ, she will not look at the following positions: $\ell+1,\dots,L$. This setting has been extensively studied within the bandit framework [6,8,11,15,16,20,27]. However, the assumption of CM regarding the order of observation is irrelevant when considering items spread in a page or on a grid (especially as reading direction of the user may varies from one country to another).

A few approaches simultaneously handle CM and PBM [18,19,26]. Their genericity is their strength and their weakness: they do not require the knowledge of the behavioral model, but they cannot use that model to speed-up the process of learning the user preferences. Moreover, these algorithms assume that the best recommendation consists in ordering the items from the more attractive to the less attractive ones. In the context of PBM, this is equivalent to assuming $\boldsymbol{\kappa}$ to be sorted in decreasing order. Our algorithm does not make such assumption. Anyhow, we also compare PB-MHB to TopRank [18] in Section 5 in order to ensure that our model benefits from the knowledge of the click model.

3 Recommendation Setting

The proposed approach handles the following online recommendations setting: at each time-stamp t, the ORS chooses a list $\boldsymbol{i}(t) = (\boldsymbol{i}_1(t),\dots,\boldsymbol{i}_L(t))$ of L distinct items among a set of N items. The user observes each position ℓ with a probability $\boldsymbol{\kappa}_\ell$, and if the position is observed, the user clicks on the item $\boldsymbol{i}_\ell$ with a probability $\boldsymbol{\theta}_{\boldsymbol{i}_\ell}$. We denote $\boldsymbol{r}_\ell(t) \in \{0,1\}$ the reward in position ℓ obtained by proposing $\boldsymbol{i}$ at time t, namely 1 if the user did observe the position ℓ and clicked on item $\boldsymbol{i}_\ell(t)$, and 0 otherwise. We assume that each draw is independent,

[4] Some refined models assume a probability to leave. With these models, the user may click on several items.

meaning $\boldsymbol{r}_\ell(t) \mid \boldsymbol{i}_\ell(t) \overset{iid.}{\sim} \text{Bernoulli}\left(\boldsymbol{\theta}_{\boldsymbol{i}_\ell}\boldsymbol{\kappa}_\ell\right)$, or in other words

$$\begin{cases} \mathbb{P}\left(\boldsymbol{r}_\ell(t) = 1 \mid \boldsymbol{i}_\ell(t)\right) = \boldsymbol{\theta}_{\boldsymbol{i}_\ell}\boldsymbol{\kappa}_\ell, \\ \mathbb{P}\left(\boldsymbol{r}_\ell(t) = 0 \mid \boldsymbol{i}_\ell(t)\right) = 1 - \boldsymbol{\theta}_{\boldsymbol{i}_\ell}\boldsymbol{\kappa}_\ell. \end{cases}$$

The ORS aims at maximizing the *cumulative reward*, namely the total number of clicks gathered from time-stamp 1 to time-stamp T: $\sum_{t=1}^{T}\sum_{\ell=1}^{L}\boldsymbol{r}_\ell(t)$.

Without loss of generality, we assume that $\max_\ell \boldsymbol{\kappa}_\ell = 1$.[5] To keep the notations simple, we also assume that $\boldsymbol{\theta}_1 > \boldsymbol{\theta}_2 > \cdots > \boldsymbol{\theta}_N$, and $\boldsymbol{\kappa}_1 = 1 > \boldsymbol{\kappa}_2 > \cdots > \boldsymbol{\kappa}_L$.[6] The best recommendation is then $\boldsymbol{i}^* = (1, 2, \ldots, L)$, which leads to the expected instantaneous reward $\mu^* = \sum_{\ell=1}^{L}\boldsymbol{\theta}_\ell\boldsymbol{\kappa}_\ell$.

The pair $(\boldsymbol{\theta}, \boldsymbol{\kappa})$ is unknown from the ORS. It has to infer the best recommendation from the recommendations and the rewards gathered at previous time-stamps, denoted $D(t) = \{(\boldsymbol{i}(1), \boldsymbol{r}(1)), \ldots, (\boldsymbol{i}(t-1), \boldsymbol{r}(t-1))\}$. This corresponds to the *bandit setting* where it is usual to consider the *(cumulative pseudo-)regret*

$$R_T \overset{def}{=} \sum_{t=1}^{T}\sum_{\ell=1}^{L}\mathbb{E}\left[\boldsymbol{r}_\ell \mid \boldsymbol{i}_\ell^*\right] - \sum_{t=1}^{T}\sum_{\ell=1}^{L}\mathbb{E}\left[\boldsymbol{r}_\ell(t) \mid \boldsymbol{i}_\ell(t)\right] = T\mu^* - \sum_{t=1}^{T}\sum_{\ell=1}^{L}\boldsymbol{\theta}_{\boldsymbol{i}_\ell(t)}\boldsymbol{\kappa}_\ell. \quad (1)$$

The regret R_T denotes the cumulative expected loss of the ORS w.r.t. the oracle recommending the best items at each time-stamp. Hereafter we aim at an algorithm which minimizes the expectation of R_T w.r.t. its choices.

4 PB-MHB Algorithm

We handle the setting presented in the previous section with the online recommender system depicted by Algorithm 1 and referred to as PB-MHB (for Position Based Metropolis-Hastings Bandit). This algorithm is based on the Thompson sampling framework [25,2]. First, we look at rewards with a fully Bayesian point of view: we assume that they follow the statistical model depicted in Section 3, and we choose a uniform prior on the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\kappa}$. Therefore the posterior probability for these parameters given the previous observations $D(t)$ is

$$\mathbb{P}\left(\boldsymbol{\theta}, \boldsymbol{\kappa} | D(t)\right) \propto \prod_{i=1}^{N}\prod_{\ell=1}^{L}\left(\boldsymbol{\theta}_i\boldsymbol{\kappa}_\ell\right)^{S_{i,\ell}(t)}\left(1 - \boldsymbol{\theta}_i\boldsymbol{\kappa}_\ell\right)^{F_{i,\ell}(t)}, \quad (2)$$

where $S_{i,\ell}(t) = \sum_{s=1}^{t-1}\mathbb{1}_{\boldsymbol{i}_\ell(s)=i}\mathbb{1}_{\boldsymbol{r}_\ell(s)=1}$ denotes the number of times the item i has been clicked while being displayed in position ℓ from time-stamp 1 to $t-1$, and $F_{i,\ell}(t) = \sum_{s=1}^{t-1}\mathbb{1}_{\boldsymbol{i}_\ell(s)=i}\mathbb{1}_{\boldsymbol{r}_\ell(s)=0}$ denotes the number of times the item i has not been clicked while being displayed in position ℓ from time-stamp 1 to $t-1$.

Second, we choose the recommendation $\boldsymbol{i}(t)$ at time-stamp t according to its posterior probability of being the best arm. To do so, we denote $(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\kappa}})$ a sample of parameters $(\boldsymbol{\theta}, \boldsymbol{\kappa})$ according to their posterior probability, we keep the best items given $\tilde{\boldsymbol{\theta}}$, and we display them in the right order given $\tilde{\boldsymbol{\kappa}}$.

[5] As stated in [14], we may replace $(\boldsymbol{\theta}, \boldsymbol{\kappa})$ by $(\boldsymbol{\theta}.\max_\ell \boldsymbol{\kappa}_\ell, \boldsymbol{\kappa}/\max_\ell \boldsymbol{\kappa}_\ell)$.

[6] Our algorithm and the experiments only assume $\boldsymbol{\kappa}_1 = 1$.

Algorithm 1 PB-MHB, Metropolis-Hastings based bandit for Position-Based Model

$D(1) \leftarrow \{\}$
for $t = 1, \dots$ **do**
  draw $(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\kappa}}) \sim \mathbb{P}\left(\boldsymbol{\theta}, \boldsymbol{\kappa} | D(t)\right)$ using Algorithm 2
  display the L items with greatest value in $\tilde{\boldsymbol{\theta}}$, ordered by decreasing values of $\tilde{\boldsymbol{\kappa}}$
  get rewards $\boldsymbol{r}(t)$
  $D(t+1) \leftarrow D(t) \cup (\boldsymbol{i}(t), \boldsymbol{r}(t))$
end for

4.1 Sampling w.r.t. the Posterior Distribution

The posterior probability (2) does not correspond to a well-known distribution. [13], [17] tackle this problem by considering that $\boldsymbol{\kappa}$ is known in order to manipulate N independent simpler distributions $\mathbb{P}_i\left(\boldsymbol{\theta}_i | \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)$. By having $\boldsymbol{\kappa}$ and $\boldsymbol{\theta}$ both unknown, we have to handle a law for which the components $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N$ and $\boldsymbol{\kappa}_1, \dots, \boldsymbol{\kappa}_L$ are correlated (see Equation 2). We handle it thanks to a carefully designed Metropolis-Hastings algorithm [22] (cf. Algorithm 2). This algorithm consists in building a sequence of m samples $(\boldsymbol{\theta}^{(1)}, \boldsymbol{\kappa}^{(1)}), \dots, (\boldsymbol{\theta}^{(m)}, \boldsymbol{\kappa}^{(m)})$ such that $(\boldsymbol{\theta}^{(m)}, \boldsymbol{\kappa}^{(m)})$ follows a good approximation of the targeted distribution. It is based on a Markov chain on parameters $(\boldsymbol{\theta}, \boldsymbol{\kappa})$ which admits the targeted probability distribution as its unique stationary distribution.

At iteration s, the sample $(\boldsymbol{\theta}^{(s)}, \boldsymbol{\kappa}^{(s)})$ moves toward sample $(\boldsymbol{\theta}^{(s+1)}, \boldsymbol{\kappa}^{(s+1)})$ by applying $(N + L - 1)$ transitions: one per item and one per position except for $\boldsymbol{\kappa}_1$. Let's start by focusing on the transition regarding item i (Lines 5–9) and denote $(\boldsymbol{\theta}, \boldsymbol{\kappa})$ the sample before the transition.

The algorithm aims at sampling a new value for $\boldsymbol{\theta}_i$ according to its posterior probability given other parameters and the previous observations $D(t)$:

$$\mathbb{P}_i\left(\boldsymbol{\theta}_i | \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right) \propto \prod_{\ell=1}^{L} {\boldsymbol{\theta}_i}^{S_{i,\ell}(t)} \left(1 - \boldsymbol{\theta}_i \boldsymbol{\kappa}_\ell\right)^{F_{i,\ell}(t)}, \tag{3}$$

where $\boldsymbol{\theta}_{-i}$ denotes the components of $\boldsymbol{\theta}$ except for the i-th one. This transition consists in two steps:

1. *draw a candidate* value $\tilde{\boldsymbol{\theta}}$ after a *proposal* probability distribution $q\left(\tilde{\boldsymbol{\theta}} \mid \boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)$ discussed later on;
2. *accept* that candidate or keep the previous sample:

$$\boldsymbol{\theta}_i \leftarrow \begin{cases} \tilde{\boldsymbol{\theta}} & \text{, w. prob. } \min\left(1, \frac{\mathbb{P}_i\left(\tilde{\boldsymbol{\theta}} | \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)}{\mathbb{P}_i\left(\boldsymbol{\theta}_i | \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)} \frac{q\left(\boldsymbol{\theta}_i | \tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)}{q\left(\tilde{\boldsymbol{\theta}} | \boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)}\right) \\ \boldsymbol{\theta}_{i-1} & \text{, otherwise} \end{cases}.$$

This acceptance step yields two behaviours:

– $\frac{\mathbb{P}_i\left(\tilde{\boldsymbol{\theta}} | \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)}{\mathbb{P}_i\left(\boldsymbol{\theta}_i | \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)}$ measures how likely the candidate value is compared to the previous one, w.r.t. the posterior distribution,

Algorithm 2 Metropolis-Hastings applied to the distribution of Equation (2)

Require: $D(t)$: previous recommendations and rewards
Require: $\sigma = c/\sqrt{t}$: Gaussian random-walk steps width
Require: m: number of iterations
1: draw $(\boldsymbol{\theta}, \boldsymbol{\kappa})$ after uniform distribution
2: $\boldsymbol{\kappa}_1 \leftarrow 1$
3: **for** $s = 1, \ldots, m$ **do**
4: **for** $i = 1, \ldots, N$ **do**
5: **repeat**
6: draw $\tilde{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\theta}_i, \sigma)$
7: **until** $0 \leqslant \tilde{\boldsymbol{\theta}} \leqslant 1$
8: **with prob.** $\min\left(1, \frac{\mathbb{P}_i\left(\tilde{\boldsymbol{\theta}} | \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)}{\mathbb{P}_i\left(\boldsymbol{\theta}_i | \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)} \frac{\Delta\Phi_\sigma(\boldsymbol{\theta}_i)}{\Delta\Phi_\sigma\left(\tilde{\boldsymbol{\theta}}\right)}\right)$
9: $\boldsymbol{\theta}_i \leftarrow \tilde{\boldsymbol{\theta}}$
10: **end for**
11: **for** $\ell = 2, \ldots, L$ **do**
12: **repeat**
13: draw $\tilde{\boldsymbol{\kappa}} \sim \mathcal{N}(\boldsymbol{\kappa}_\ell, \sigma)$
14: **until** $0 \leqslant \tilde{\boldsymbol{\kappa}} \leqslant 1$
15: **with prob.** $\min\left(1, \frac{\mathbb{P}_\ell\left(\tilde{\boldsymbol{\kappa}} | \boldsymbol{\theta}, \boldsymbol{\kappa}_{-\ell}, D(t)\right)}{\mathbb{P}_\ell\left(\boldsymbol{\kappa}_\ell | \boldsymbol{\theta}, \boldsymbol{\kappa}_{-\ell}, D(t)\right)} \frac{\Delta\Phi_\sigma(\boldsymbol{\kappa}_\ell)}{\Delta\Phi_\sigma\left(\tilde{\boldsymbol{\kappa}}\right)}\right)$
16: $\boldsymbol{\kappa}_\ell \leftarrow \tilde{\boldsymbol{\kappa}}$
17: **end for**
18: **end for**
19: **return** $(\boldsymbol{\theta}, \boldsymbol{\kappa})$

– $\frac{q\left(\boldsymbol{\theta}_i | \tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)}{q\left(\tilde{\boldsymbol{\theta}} | \boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)}$ downgrades candidates easily reached by the proposal q.

Algorithm 2 uses a truncated Gaussian random-walk proposal for the parameter $\boldsymbol{\theta}_i$, with a Gaussian step of *standard deviation* σ (see Lines 5–7). Note that due to the truncation, the probability to get the proposal $\tilde{\boldsymbol{\theta}}$ starting from $\boldsymbol{\theta}_i$ is $q\left(\tilde{\boldsymbol{\theta}} \mid \boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right) = \phi(\tilde{\boldsymbol{\theta}} \mid \boldsymbol{\theta}_i, \sigma)/\Delta\Phi_\sigma(\boldsymbol{\theta}_i)$, where $\phi(\cdot \mid \boldsymbol{\theta}_i, \sigma)$ is the probability associated to the Gaussian distribution with mean $\boldsymbol{\theta}_i$ and standard deviation σ, $\Phi(\cdot \mid \boldsymbol{\theta}_i, \sigma)$ is its cumulative distribution function, and $\Delta\Phi_\sigma(\boldsymbol{\theta}_i) = \Phi(1 \mid \boldsymbol{\theta}_i, \sigma) - \Phi(0 \mid \boldsymbol{\theta}_i, \sigma)$. The probability to get the proposal $\boldsymbol{\theta}_i$ starting from $\tilde{\boldsymbol{\theta}}$ is similar, which reduces the ratio of proposal probabilities at Line 8 to

$$\frac{q\left(\boldsymbol{\theta}_i \mid \tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)}{q\left(\tilde{\boldsymbol{\theta}} \mid \boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}, \boldsymbol{\kappa}, D(t)\right)} = \frac{\Delta\Phi_\sigma\left(\boldsymbol{\theta}_i\right)}{\Delta\Phi_\sigma\left(\tilde{\boldsymbol{\theta}}\right)}.$$

The transition regarding parameter $\boldsymbol{\kappa}_\ell$ involves the same framework: the proposal is a truncated Gaussian random-walk step and aims at the probability

$$\mathbb{P}_\ell\left(\boldsymbol{\kappa}_\ell | \boldsymbol{\theta}, \boldsymbol{\kappa}_{-\ell}, D(t)\right) \propto \prod_{i=1}^{N} {\boldsymbol{\kappa}_\ell}^{S_{i,\ell}(t)} \left(1 - \boldsymbol{\theta}_i \boldsymbol{\kappa}_\ell\right)^{F_{i,\ell}(t)}. \tag{4}$$

4.2 Overall Complexity

The computational complexity of Algorithm 1 is driven by the number of random-walk steps done per recommendation: $m(N+L-1)$, which is controlled by the parameter m. This parameter corresponds to the burning period: the number of iterations required by the Metropolis-Hastings algorithm to draw a point $(\boldsymbol{\theta}^{(m)}, \boldsymbol{\kappa}^{(m)})$ almost independent from the initial one. While the requirement for a burning period may refrain us from using a Metropolis-Hasting algorithm in such recommendation setting, we demonstrate in the following experiments that the required value for m remains reasonable. We drastically reduce m by starting the Metropolis-Hasting call from the point used to recommend at previous time-stamp. This corresponds to replacing Line 1 in Algorithm 2 by:

$$\boxed{\text{1: } (\boldsymbol{\theta}, \boldsymbol{\kappa}) \leftarrow (\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\kappa}}) \text{ used for the previous recommendation}}.$$

5 Experiments

In this section we demonstrate the benefit of the proposed approach both on two artificial and two real-life datasets. Note that whatever real-life data we are using, we can only use them to compute the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\kappa}$ and simulate at each time-stamp a "real" user feedback (i.e. clicks) by applying PBM with these parameters. This is what is usually done in the literature since the recommendations done by a bandit are very unlikely to match the recommendations logged in the ground truth data and without a good matching, it would be impossible to compute a relevant reward for each interaction. Code and data for replicating our experiments are available at https://github.com/gaudel/ranking_bandits.

5.1 Datasets

In the experiments, the online recommender systems are required to deliver T consecutive recommendations, their feedbacks being drawn from a PBM distribution. We consider two settings denoted *purely simulated* and *behavioral* in the remaining. With the *purely simulated* setting, we choose the value of the parameters $(\boldsymbol{\theta}, \boldsymbol{\kappa})$ to highlight the stability of the proposed approach even for extreme settings. Namely, we consider $N = 10$ items, $L = 5$ positions, and $\boldsymbol{\kappa} = [1, 0.75, 0.6, 0.3, 0.1]$. The range of values for $\boldsymbol{\theta}$ is either close to zero ($\boldsymbol{\theta}^- = [10^{-3}, 5.10^{-4}, 10^{-4}, 5.10^{-5}, 10^{-5}, 10^{-6}, \ldots, 10^{-6}]$) or close to one ($\boldsymbol{\theta}^+ = [0.99, 0.95, 0.9, 0.85, 0.8, 0.75, \ldots, 0.75]$).

With the *behavioral* setting, the values for $\boldsymbol{\kappa}$ and $\boldsymbol{\theta}$ are obtained from true users behavior as in [17], [14]. Two behavioral datasets are considered and presented hereafter. The first one is *KDD Cup 2012 track 2* dataset, which consists of session logs of *soso.com*, a Tencent's search engine. It tracks clicks and displays of advertisements on a search engine result web-page, w.r.t. the user query. For each query, 3 positions are available for a various number of ads to display.

To follow previous works, instead of looking for the probability to be clicked per display, we target the probability to be clicked per session. This amounts

to discarding the information *Impression*. We also filter the logs to restrict the analysis to (query, ad) couples with enough information: for each query, ads are excluded if they were displayed less than 1,000 times at any of the 3 possible positions. Then, we filter queries that have less than 5 ads satisfying the previous condition. We end up with 8 queries and from 5 to 11 ads per query. Finally, for each query q, the parameters $(\boldsymbol{\theta}^{[q]}, \boldsymbol{\kappa}^{[q]})$ are set from the *Singular Value Decomposition* (SVD) of the matrix $\boldsymbol{M}^{[q]} \in \mathbb{R}^{N \times L}$ which contains the probability to be clicked for each item in each position. By denoting $\zeta^{[q]}$, the greatest singular value of $\boldsymbol{M}^{[q]}$, and $\boldsymbol{u}^{[q]}$ (respectively $\boldsymbol{v}^{[q]}$) the left (resp. right) singular vector associated to $\zeta^{[q]}$, we set $\boldsymbol{\theta}^{[q]} \stackrel{def}{=} \boldsymbol{v}_1^{[q]} \zeta^{[q]} \boldsymbol{u}^{[q]}$ and $\boldsymbol{\kappa}^{[q]} \stackrel{def}{=} \boldsymbol{v}^{[q]} / \boldsymbol{v}_1^{[q]}$, such that $\boldsymbol{\kappa}_1^{[q]} = 1$, and ${\boldsymbol{\theta}^{[q]}}^T \boldsymbol{\kappa}^{[q]} = \zeta {\boldsymbol{u}^{[q]}}^T \boldsymbol{v}^{[q]}$. This leads to $\boldsymbol{\theta}_i$ values ranging from 0.004 to 0.149, depending on the query.

The second behavioral dataset is Yandex[7]. As in [18], we select the 10 most frequent queries, and for each query the ORS displays 5 items peeked among the 10 most attractive ones. As for KDD dataset, the parameters $(\boldsymbol{\theta}^{[q]}, \boldsymbol{\kappa}^{[q]})$ are set from an SVD. This leads to $\boldsymbol{\theta}_i$ values ranging from 0.070 to 0.936, depending on the query.

5.2 Competitors

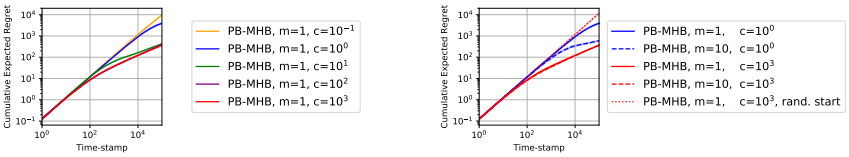
We compare the performance of PB-MHB with the performance of PMED [14], TopRank [18], and the standard baseline ε_n-greedy [3].

PMED is designed to match a lower-bound on the expected regret of any reasonable algorithm under the PBM assumption. Let us recall that it assumes $\boldsymbol{\kappa}_\ell$ values to be decreasing, which means the ORS knows in advance which is the most observed position, which is the second most observed position, and so on. PB-MHB does not require this ordering, it learns it from interactions with users. PMED uniform-exploration parameter α is fixed to 1. Due to its very high time complexity, experiments with PMED are stopped after 5 hours for each dataset (which corresponds to about 10^5 recommendations).

TopRank handles a wider range of click models than PB-MHB, but it also assumes the knowledge of the order on positions. TopRank hyper-parameter δ is set to $1/T$ as recommended by Theorem 1 in [18].

Finally, we compare PB-MHB to ϵ_n-Greedy. At each time-stamp t, an estimation $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\kappa}})$ of parameters $(\boldsymbol{\theta}, \boldsymbol{\kappa})$ is obtained applying SVD to the collected data. Let us denote $\hat{\boldsymbol{i}}(t)$ the recommendation with the highest expected reward given the inferred values $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\kappa}})$. A greedy algorithm would recommend $\hat{\boldsymbol{i}}(t)$. Since this algorithm never explores, it may end-up recommending a sub-optimal affectation. ϵ_n-Greedy counters this by randomly replacing each item of the recommendation with a probability $\varepsilon(t) = c/t$, where c is a hyper-parameter to be tuned. In the following, we plot the results obtained with the best possible value

[7] Yandex personalized web search challenge, 2013. https://www.kaggle.com/c/yandex-personalized-web-search-challenge



(a) Impact of c (b) Impact of m and random start

Fig. 1: Cumulative regret w.r.t. time-stamp on Yandex data. Impact of the width $c/\sqrt{t}$ of Gaussian random-walk steps (left). Impact of the use of the parameters from the previous time-stamp to warm-up the Metropolis-Hasting algorithm and of the number m of Metropolis-Hastings iterations per recommendation (right). The shaded area depicts the standard error of our regret estimates.

for c, while trying c in $\{10^0, 10^1, \ldots, 10^6\}$. Note that the best value for c varies a lot from a dataset to another.

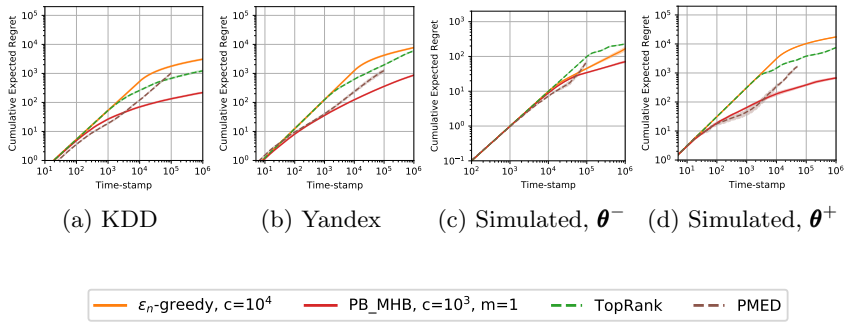
In the experiments presented hereafter the requirements of each algorithm are enforced. Namely, $\boldsymbol{\kappa}_\ell$ values are decreasing when running experiments with PMED and TopRank.

5.3 Results

We compare the previously presented algorithms on the basis of the *cumulative regret* (see Equation (1)), which is the sum, over T consecutive recommendations, of the difference between the expected reward of the best possible answer and of the answer of a given recommender system. The regret will be plotted with respect to T on a log-scale basis. The best algorithm is the one with the lowest regret. The regret plots are bounded by the regret of the oracle (0) and the regret of a recommender system choosing the items uniformly at random. We average the results of each algorithm over 20 independant sequences of recommendations per query.

PB-MHB Hyper-parameters PB-MHB behavior is affected by two hyper-parameters: the width $c/\sqrt{t}$ of the Gaussian random-walk steps, and the number m of Metropolis-Hastings iterations per recommendation. Overall, when Metropolis-Hastings runs start from the couple $(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\kappa}})$ from the previous time-stamp, we show in Figure 1a that PB-MHB exhibits the smallest regret on Yandex as soon as $c > 1000$ and $m = 1$. Note that $m = 1$ is also the setting which minimizes the computation time of PB-MHB. These hyper-parameter choices also hold for the 3 other datasets (not shown here for lack of space).

We now discuss the impact of choosing other hyper-parameter values on Yandex data. The regret is the smallest as soon as c is large enough ($c >= 100$). In Figure 1b, we illustrate the impact of m. It yields a high regret only when c and m are both too small (full blue curve): when the random-walk steps are too small the Metropolis-Hasting algorithm requires more iterations to get uncorrelated



(a) KDD (b) Yandex (c) Simulated, $\boldsymbol{\theta}^-$ (d) Simulated, $\boldsymbol{\theta}^+$

Fig. 2: Cumulative regret w.r.t. time-stamp on five different settings for all competitors. The plotted curves correspond to the average over 20 independant sequences of recommendations per query (in total: 20 sequences for simulated data, 160 sequences for KDD and 200 sequences for Yandex). The shaded area depicts the standard error of our regret estimates. For ϵ_n-Greedy, c is set to 10^4 for KDD and Yandex settings, to 10^5 when $\boldsymbol{\theta}$ is close to 0, and to 10^3 when $\boldsymbol{\theta}$ is close to 1.

samples $(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\kappa}})$. For reasonable values of c, m has no impact on the regret. Figure 1b also shows the impact of keeping the parameters from the previous time-stamp compared to a purely random start. Starting from a new randomly drawn set of parameters would require more than $m = 10$ iterations to obtain the same result, meaning a computation budget more than 10 times higher. This behavior is explained by the gap between the uniform law (which is used to draw the starting set of parameters) and the targeted law (*a posteriori* law of these parameters) which concentrates around its MAP. Even worse, this gap increases while getting more and more data since the *a posteriori* law concentrates with the increase of data. As a consequence, the required value for m increases along time when applying a standard Metropolis-Hasting initialisation, which explains why the dotted red line diverges from the solid one around time-stamp 30.

Comparison with Competitors Figure 2 compares the regret obtained by PB-MHB and its competitors on datasets with various click and observation probabilities. Up to time-stamp 10^4, PMED and PB-MHB exhibit the smallest regret on all settings. Thereafter, PMED is by far the algorithm with the smallest regret. Regarding computation time, apart from PMED all the algorithms require less than 20 ms per recommendation which remains affordable: ϵ_n-Greedy is the fastest with less than 0.5ms per recommendation[8]; then TopRank and PB-MHB require 10 to 40 times more computation time than ϵ_n-Greedy; finally,

[8] Computation time for a sequence of 10^7 recommendations vs. the first query of Yandex data, on an Intel Xeon E5640 CPU2.67GHz with 50 GB RAM. The algorithms are implemented in Python.

PMED requires more than 150ms per recommendation which makes it impractical regardless of its low regret.

6 Conclusion

We have introduced a new bandit-based algorithm, PB-MHB, for online recommender systems in the PBM which uses a Thompson sampling framework to learn the $\boldsymbol{\kappa}$ and $\boldsymbol{\theta}$ parameters of this model instead of assuming them given. Experiments on simulated and real datasets show that our method (i) suffers a smaller regret than its competitors having access to the same information, and (ii) suffers a similar regret as its competitors using more prior information.

These results are still empirical but we plan to formally prove them in future work. Indeed, [21] upper-bounded the regret of a Thompson Sampling bandit algorithm, while using Langevin Monte-Carlo to sample posterior values of parameters. That could be a good starting point for theoretically studying the convergence of PB-MHB. We also would like to improve our algorithm by further working on the proposal law to draw candidates for the sampling part. The proposal is currently a truncated random walk. By managing it differently (with a logit transformation for instance) we could improve both the time and precision performance. On the other hand, with a better understanding of the evolution of the target distribution, we could also improve the sampling part. Moreover, we would like to apply PB-MHB to environments where $\boldsymbol{\kappa}$ is evolving with time (with new marketing trends) and where our learning setting could develop its full potential.

References

1. Agrawal, S., Goyal, N.: Thompson sampling for contextual bandits with linear payoffs. In: proc. of the 30th Int. Conf. on Machine Learning. ICML'13 (2013)
2. Agrawal, S., Goyal, N.: Near-optimal regret bounds for thompson sampling. Jour. of the ACM, JACM **64**(5), 30:1–30:24 (Sep 2017)
3. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2), 235–256 (May 2002)
4. Chapelle, O., Li, L.: An empirical evaluation of thompson sampling. In: Advances in Neural Information Processing Systems 24. NIPS'11 (2011)
5. Chen, W., Wang, Y., Yuan, Y.: Combinatorial multi-armed bandit: General framework and applications. In: proc. of the 30th Int. Conf. on Machine Learning. ICML'13 (2013)
6. Cheung, W.C., Tan, V., Zhong, Z.: A thompson sampling algorithm for cascading bandits. In: proc. of the 22nd Int. Conf. on Artificial Intelligence and Statistics. AISTATS'19 (2019)
7. Chuklin, A., Markov, I., de Rijke, M.: Click Models for Web Search. Morgan & Claypool Publishers (2015)
8. Combes, R., Magureanu, S., Proutière, A., Laroche, C.: Learning to rank: Regret lower bounds and efficient algorithms. In: proc. of the 2015 ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems (2015)

9. Craswell, N., Zoeter, O., Taylor, M., Ramsey, B.: An experimental comparison of click position-bias models. In: proc. of the 2008 Int. Conf. on Web Search and Data Mining. WSDM '08 (2008)
10. Dumitrascu, B., Feng, K., Engelhardt, B.: Pg-ts: Improved thompson sampling for logistic contextual bandits. In: Advances in Neural Information Processing Systems 31. NIPS'18 (2018)
11. Katariya, S., Kveton, B., Szepesvári, C., Wen, Z.: Dcm bandits: Learning to rank with multiple clicks. In: proc. of the Int. Conf. on Machine Learning. ICML (2016)
12. Kawale, J., Bui, H.H., Kveton, B., Tran-Thanh, L., Chawla, S.: Efficient thompson sampling for online matrix factorization recommendation. In: Advances in Neural Information Processing Systems 28. NIPS'15 (2015)
13. Komiyama, J., Honda, J., Nakagawa, H.: Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays. In: proc. of the 32nd Int. Conf. on Machine Learning. ICML'15 (2015)
14. Komiyama, J., Honda, J., Takeda, A.: Position-based multiple-play bandit problem with unknown position bias. In: Advances in Neural Information Processing Systems 30. NIPS'17 (2017)
15. Kveton, B., Szepesvári, C., Wen, Z., Ashkan, A.: Cascading bandits: Learning to rank in the cascade model. In: proc. of the 32nd Int. Conf. on Machine Learning. ICML'15 (2015)
16. Kveton, B., Wen, Z., Ashkan, A., Szepesvári, C.: Combinatorial cascading bandits. In: Advances in Neural Information Processing Systems 28. NIPS'15 (2015)
17. Lagrée, P., Vernade, C., Cappé, O.: Multiple-play bandits in the position-based model. In: Advances in Neural Information Processing Systems 30. NIPS'16 (2016)
18. Lattimore, T., Kveton, B., Li, S., Szepesvari, C.: TopRank: A practical algorithm for online stochastic ranking. In: Advances in Neural Information Processing Systems 31. NIPS'18 (2018)
19. Li, C., Kveton, B., Lattimore, T., Markov, I., de Rijke, M., Szepesvári, C., Zoghi, M.: BubbleRank: Safe online learning to re-rank via implicit click feedback. In: proc. of the 35th Uncertainty in Artificial Intelligence Conference. UAI'19 (2019)
20. Li, S., Wang, B., Zhang, S., Chen, W.: Contextual combinatorial cascading bandits. In: proc. of the 33rd Int. Conf. on Machine Learning. ICML'16 (2016)
21. Mazumdar, E., Pacchiano, A., Ma, Y., Bartlett, P.L., Jordan, M.I.: On thompson sampling with langevin algorithms. In: proc. of the 37th Int. Conf. on Machine Learning. ICML'20 (2020)
22. Neal, R.M.: Probabilistic inference using markov chain monte carlo methods. Tech. rep., University of Zurich, Department of Informatics (09 1993)
23. Richardson, M., Dominowska, E., Ragno, R.: Predicting clicks: Estimating the click-through rate for new ads. In: proc. of the 16th International World Wide Web Conference. WWW '07 (2007)
24. Riquelme, C., Tucker, G., Snoek, J.: Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In: proc. of the Int. Conf. on Learning Representations. ICLR'18 (2018)
25. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika **25**(3/4), 285–294 (1933)
26. Zoghi, M., Tunys, T., Ghavamzadeh, M., Kveton, B., Szepesvari, C., Wen, Z.: Online learning to rank in stochastic click models. In: proc. of the 34th Int. Conf. on Machine Learning. ICML'17 (2017)
27. Zong, S., Ni, H., Sung, K., Ke, N.R., Wen, Z., Kveton, B.: Cascading bandits for large-scale recommendation problems. In: proc. of the 32nd Conference on Uncertainty in Artificial Intelligence. UAI '16 (2016)