Explore/Exploit Schemes for Web Content Optimization

Deepak Agarwal
Yahoo! Research
dagarwal@yahoo-inc.com

Bee-Chung Chen
Yahoo! Research
beechun@yahoo-inc.com

Pradheep Elango
Yahoo! Labs
elango@yahoo-inc.com

***Abstract*— We propose novel multi-armed bandit (explore/exploit) schemes to maximize total clicks on a content module published regularly on Yahoo! Intuitively, one can "explore" each candidate item by displaying it to a small fraction of user visits to estimate the item's click-through rate (CTR), and then "exploit" high CTR items in order to maximize clicks. While bandit methods that seek to find the optimal trade-off between explore and exploit have been studied for decades, existing solutions are not satisfactory for web content publishing applications where dynamic set of items with short lifetimes, delayed feedback and non-stationary reward (CTR) distributions are typical. In this paper, we develop a Bayesian solution and extend several existing schemes to our setting. Through extensive evaluation with nine bandit schemes, we show that our Bayesian solution is uniformly better in several scenarios. We also study the empirical characteristics of our schemes and provide useful insights on the strengths and weaknesses of each. Finally, we validate our results with a "side-by-side" comparison of schemes through *live experiments* conducted on a random sample of real user visits to Yahoo!**

I. Introduction

Selecting the best items to display for a given user visit (i.e., page view) from an available set is a fundamental problem in applications like web search, online advertising and content publishing. For instance, items are documents for web search, ads for online advertising and articles for content publishing. Typically, the goal is to maximize a readily available yield metric that acts as a proxy for long-term success, click-rate (CTR) is often used in practice. If the CTR of items are known, one could maximize clicks by simply displaying high CTR items to user visits. Of course, CTR's are unknown and estimated from data obtained through a sequential process. Multi-armed bandit schemes provide an attractive and tractable framework to construct such sequential processes in an optimal fashion ([1], [2], [3]). Originally motivated by problems in a gambling scenario, these schemes construct sequential designs to play different *arms* of a slot machine with the goal of maximizing the expected total *reward* for the gambler. By now, multi-armed bandit schemes have found wide applicability in diverse areas like clinical trials [4], marketing [5], management [6], web search [7] and online advertising [8]. However, existing solutions to the "standard" bandit problem that assumes a fixed set of arms with no delayed feedback are sub-optimal for our application scenario. We propose a new set of schemes that significantly outperform the standard bandit solutions and have wider applicability.

Motivating application: We consider maximizing total clicks on a content module published by Yahoo! The module is a panel with several slots, where each slot displays an editorially programmed item (story) selected from a content pool of several items. For simplicity, we focus on maximizing clicks on the most prominent slot of the module that obtains a large fraction of clicks. More precisely, we construct schemes that select an item from the available pool for every user visit to maximize overall CTR.

In this paper, we primarily focus on explore/exploit methods that quickly converge to the *most popular* item (item with highest CTR at any point in time). In Section IV-C, we discuss an extension to personalized recommendation based on user clusters. Bandit solutions for general machine-learning models are beyond the scope of this paper.

We now discuss the system constraints and characteristics of our application that violate assumptions underlying standard bandit solutions, but are typical in the context of content publishing on the web. The *standard* bandit problem considers a *fixed* set of arms (items), each of which is assumed to have an unknown but *fixed* reward (CTR). After pulling an arm (showing an item to a user visit), one *immediately* obtains an observation (click or no click) that reduces uncertainty about the reward (CTR) distribution of the arm. The goal is to find a sequential scheme that maximizes the expected total clicks after a certain number of arm pulls. However, in our application, we have:

- **Dynamic set of items**: Items usually have short lifetimes (6-24 hours) and the set of available ones changes over time. Editors add new items to keep up with current news and eliminate old and fading ones.
- **Non-stationary CTR**: As shown in [9], the CTR of each item changes over time, with strong diurnal pattern (high points are several times higher than low points) and CTR decay caused by repeated exposure. We predict *future* item CTR distribution through a dynamic time-series model that is incorporated in our explore/exploit schemes.
- **Batch Serving**: Click and view observations are delayed due to system performance constraints and delayed user feedback. The latter occurs due to time lag between an item view and subsequent user click (usually within minutes). Our solutions provide a sampling plan (fraction of views allocated to each item in the next time interval),

instead of item priorities that are typical of standard bandit schemes.

A. Problem Definition

Throughout, index i and t denote item and time interval (we used 5 minute intervals) respectively. Let p_{it} denote the *unobserved* time-varying CTR of item i at time t, N_t denote the total number of user visits (i.e., page views), and $\mathcal{I}_t$ denote the set of available items. Note that the available itemset is dynamic, hence the suffix t on $\mathcal{I}$. If p_{it}'s were known, the optimal solution is to serve all N_t user visits to item $i_t^* = \arg\max_i p_{it}$ at t. Since p_{it}'s are unknown, they need to be estimated by showing item i to some visits. We assume N_t's are *known* (usually obtained through a forecasting model).

Serving scheme: A serving scheme (also called *policy*) π is an algorithm that, for each interval t, decides what fraction of user visits should be allocated to each item based on all data observed before t. Let x_{it}^π (and $x_{it}^\pi N_t$) denote the fraction (and number) of user visits that π allocates to item i in interval t, where $\sum_i x_{it}^\pi = 1$, for each t. Note that these policies are different from standard multi-armed bandit ones where feedback is assumed instantaneous and item states change after each user visit. We drop superscript π and subscript i when they are clear from the context.

Observation: We observe c_{it}^π clicks after serving $x_{it}^\pi N_t$ user visits to item i in interval t; c_{it}^π is a random variable. Following [9], we assume $c_{it}^\pi \sim$ *Poisson*$(p_{it} x_{it}^\pi N_t)$, which has also been validated for our application. We note that it is straightforward to replace *Poisson* with *Binomial*. Let $R(\pi, T) = \sum_{t=1}^{T} \sum_{i \in \mathcal{I}_t} c_{it}^\pi$ denote the total number of clicks (also called *reward*) over T intervals (typically, a few months) obtained through scheme π.

Definition 1: **Oracle optimal scheme:** Assume an oracle that knows p_{it} exactly. The oracle scheme π^+ picks the item ($i_t^* = \arg\max_i p_{it}$) having the highest p_{it} for each time t.

Definition 2: **Regret:** The regret of scheme π is the difference between the oracle optimal reward and its reward; i.e., $E[R(\pi^+, T)] - E[R(\pi, T)]$.

Definition 3: **Bayes optimal scheme:** Assume a prior distribution $\mathcal{P}$ over p_{it}. Given N_t and $\mathcal{I}_t$ (for $1 \le t \le T$), a serving scheme π^* is Bayes optimal under $\mathcal{P}$ if $E_{\mathcal{P}}[R(\pi^*, T)] = \max_\pi E_{\mathcal{P}}[R(\pi, T)]$, where $E_{\mathcal{P}}$ denote expectation computed under $\mathcal{P}$.

Our goal is to find a Bayes optimal scheme. Note that even Bayes optimal schemes have non-zero regrets, because they need to explore items to estimate item CTR's, while the oracle optimal scheme does not explore at all.

B. Related Work and Our Contributions

There is a rich literature on the bandit problem; a proper review of this area is not possible here. The standard k-armed bandit assumes the arms are static over time. Although lifetime constraints can be modeled through restless bandits [2], [10], [11], [12], finding the solution is computationally infeasible (PSPACE-complete [13]) for even modest-scale applications. Recently in Machine Learning, Chakrabarti et al. [14] introduced mortal bandits where they study regret (defined differently from ours) bounds for dynamic arms. They work in a model agnostic framework and do not exploit an available predictive distribution as we do through a Bayesian model. Several bandit schemes for non-stationary reward distribution have been proposed. For instance, Slivkins and Upfal [15] reported a study of restless bandits with reward following a Brownian motion; this however, does not apply to our setting. Adversarial bandit schemes [16] also assume non-stationarity; the solutions are however sub-optimal in our scenario as we shall see in Section IV. While delayed outcomes [17] and parallel bandits [18] (which select multiple items but each one is picked at most once in one action) have been studied in the literature, to the best of our knowledge, bandit schemes that provide a batch serving plan have not been carefully analyzed before.

It is important to note that most of the known optimality results are for always-available arms (e.g., [1], [19], [20]), where regret is defined based on the "optimal scheme" that plays the *single* best arm *at all times*. Our setting is very different where arms have short lifetimes of non-zero rewards and different start times (common in many web applications, e.g., news, advertising, etc.). Thus, we define regret based on the oracle optimal scheme that plays the best available arm at each individual time point; the best available arms at different time points may be different. Playing the single best arm (which has a short period of non-zero reward) at all times would perform poorly in our setting. Optimality bounds for our regret definition are generally unknown and a challenging area for future research.

Our approach: Our goal is to find effective solutions to real-world explore/exploit problems. Deriving theoretically optimal regret bounds with large sample size is not the focus of this paper. As we shall show in Section IV, schemes with optimal regret bounds may have poor empirical performance due to inappropriate assumptions and large constants associated with regret bounds. Our approach is to appropriately model the characteristics of our application, develop Bayes optimal solutions in simplified scenarios and near-optimal solutions to the general case by appropriate approximations, and then empirically evaluate a large number of schemes using real log data and finally conduct *live* experiments in the target application that serves real users.

Our contributions: We provide an extensive study of *batch* schemes in a multi-armed bandit framework for applications with dynamic item pools, non-stationary reward distributions and delayed feedback (common in web applications). (1) We propose two *novel* classes of bandit schemes: (a) schemes that are developed from first principles in a Bayesian framework, and (b) schemes that are adaptations of standard bandit methods. (2) We show the effectiveness of our solutions through extensive evaluation on simulated and real data obtained from a content publishing application, and also results from *live* experiments with real user visits to a major internet portal. Such live experiments are important to validate applicability of bandit schemes, but rarely reported in the literature. (3) We

show that our Bayesian solution is both computationally efficient and better than other other schemes in our experiments. Such an empirical comparison between Bayesian and non-Bayesian schemes on a real application has not been reported before. (4) We show the empirical characteristics of a variety of schemes (in Section IV-B), explaining the reasons behind the strength and weakness of each scheme.

II. Bayesian Solution

In this section, we describe our Bayesian solution with approximations that ensure computational feasibility. We develop our Bayesian solution in stages, beginning with optimal solutions for simple scenarios that assume a fixed set of items. This is followed by our near-optimal solution to the general case similar to an index policy (solving a K-dimensional problem through K one-dimensional ones).

Basics: To simplify notations, here we consider a single item and drop index i. The item CTR in interval t has prior $p_t \sim \mathcal{P}(\boldsymbol{\theta}_t)$, where vector $\boldsymbol{\theta}_t$ is the *state* or parameter of the distribution. After we serve the item $x_t N_t$ times to obtain c_t clicks, we obtain the posterior (updated prior) at time $t+1$, $p_{t+1} \sim \mathcal{P}(\boldsymbol{\theta}_{t+1})$. Note that c_t is a random variable; to emphasize that $\boldsymbol{\theta}_{t+1}$ is a function of c_t and x_t, we sometimes write $\boldsymbol{\theta}_{t+1}(c_t, x_t)$. We consider the following model assuming a stationary CTR (dynamic models will be discussed later).

Gamma-Poisson model (GP): Following [9], we assume, at time t, the prior distribution $\mathcal{P}(\boldsymbol{\theta}_t)$ is $Gamma(\alpha_t, \gamma_t)$ with mean α_t/γ_t and variance α_t/γ_t^2. Suppose we serve $x_t N_t$ user visits with the item and get c_t clicks, where the click count distribution is $(c_t \,|\, p_t, x_t N_t) \sim Poisson(p_t x_t N_t)$. By conjugacy, $\mathcal{P}(\boldsymbol{\theta}_{t+1}) = Gamma(\alpha_t + c_t, \gamma_t + x_t N_t)$. Note that, α_t and γ_t intuitively represent the numbers of clicks and views (respectively) observed so far. When computing the fraction of user visits to be allocated to the item in interval t, item state $\boldsymbol{\theta}_t = [\alpha_t, \gamma_t]$ is known. However, $\boldsymbol{\theta}_{t+1}(c_t, x_t)$ is random since we have *not* observed c_t, the number of clicks that would be obtained with an allocation of $x_t N_t$ user visits.

One-step lookahead: We consider the optimal scheme with only one remaining interval (call this interval 1). That is, we want to find x_{i1}'s that maximize the expected total number of clicks:

$$\max_{x_{i1}} E[\sum_{i \in \mathcal{I}_1} x_{i1} N_1 p_{i1}] = \max_{x_{i1}} \sum_{i \in \mathcal{I}_1} x_{i1} N_1 E[p_{i1}],$$

subject to $\sum_{i \in \mathcal{I}_1} x_{i1} = 1$ and $0 \le x_{i1} \le 1$, for all i. It is easy to see that the maximum is attained if we assign 100% user visits to the item with the highest expected CTR; i.e, $x_{i^*1} = 1$ if $E[p_{i^*1}] = \max_i E[p_{i1}]$, and $x_{i1} = 0$ otherwise.

A. 2×2 Case: Two Items, Two Intervals

Next simplified case where we can efficiently find the optimal solution is the 2×2 case, i.e., we have two items and two remaining intervals. In fact, the two-armed case have been studied in the literature before (e.g., [21], [22]) but under different assumptions. Let us also assume we know the CTR of one item without any uncertainty. We use 0 and 1 as time indices for the two intervals. Since there are only two items, we simplify our notations.

- N_0, N_1: number of user visits at time 0 and 1.
- q_0, q_1: CTR of the *certain* item at time 0 and 1.
- $p_0 \sim \mathcal{P}(\boldsymbol{\theta}_0)$, $p_1 \sim \mathcal{P}(\boldsymbol{\theta}_1)$ are CTR distributions of the *uncertain* item at time 0 and 1 respectively.
- x, x_1 are the fractions of user visits allocated to the uncertain item at time 0 and 1; $(1-x)$ and $(1-x_1)$ are the fractions allocated to the certain item.
- c, a random variable, denotes the number of clicks that the uncertain item gets at time 0.
- Let $\hat{p}_0 = E[p_0]$ and $\hat{p}_1(x,c) = E[p_1 \,|\, x, c]$. In fact, for the GP model, $\hat{p}_0 = \alpha/\gamma$ and $\hat{p}_1(x,c) = (c + p_0\gamma)/(\gamma + xN_0)$.

The current state $\boldsymbol{\theta}_0 = [\alpha, \gamma]$ is known but $\boldsymbol{\theta}_1$ is a function of c, hence random. Also, the decision x_1 is a function of c. To emphasize this, we write $x_1(c)$ and let $\mathcal{X}_1$ denote the set of all such functions. Our goal is to find $x \in [0,1]$ and $x_1 \in \mathcal{X}_1$ that maximize the expected total number of clicks in the two intervals given by

$$\begin{aligned} &E[N_0(xp_0 + (1-x)q_0) + N_1(x_1 p_1 + (1-x_1)q_1)] \\ &= E[N_0 x(p_0 - q_0) + N_1 x_1 (p_1 - q_1)] + q_0 N_0 + q_1 N_1. \end{aligned}$$

Since $q_0 N_0$ and $q_1 N_1$ are constants, we only need to maximize the expectation term. That is, we find x and x_1 that maximize:

$$Gain(x, x_1) = E[N_0 x(p_0 - q_0) + N_1 x_1 (p_1 - q_1)], \quad (1)$$

Note that $Gain(x, x_1)$ is the difference in the expected number of clicks between: (a) a scheme that distributes user visits between two items (xN_0 and $x_1 N_1$ are allocated to the uncertain item at time 0 and 1) and (b) a scheme that always serves the certain item. Intuitively, it quantifies the gain obtained by exploring the uncertain item that could be *potentially* better than the certain one.

Proposition 1: Given $\boldsymbol{\theta}_0$, q_0, q_1, N_0 and N_1,

$$\max_{x \in [0,1], x_1 \in \mathcal{X}_1} Gain(x, x_1) = \max_{x \in [0,1]} Gain(x),$$

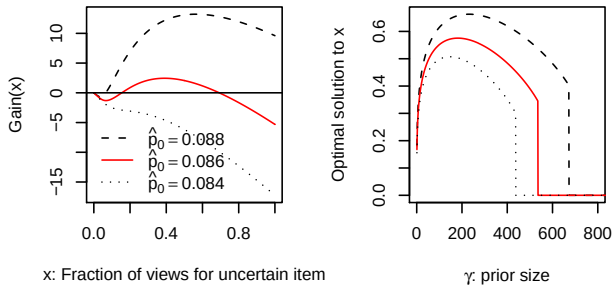
where $Gain(x) = Gain(x, \boldsymbol{\theta}_0, q_0, q_1, N_0, N_1) =$

$$N_0 x(\hat{p}_0 - q_0) + N_1 E_c[\max\{\hat{p}_1(x,c) - q_1 \ , \ 0\}].$$

Note that $\hat{p}_0$ (= α/γ for Gamma) and $\hat{p}_1(x,c)$ (= $(c + p_0\gamma)/(\gamma + xN_0)$ for Gamma) are functions of $\boldsymbol{\theta}_0$ (= $[\alpha, \gamma]$ for Gamma), the tail expectation $E_c[\max\{\hat{p}_1(x,c) - q_1 \ , \ 0\}]$ with respect to the *marginal* distribution of c appears since interval 1 is the last interval and from the one-step lookahead case, when the gain is maximized, $x_1(c)$ would either be 0 or 1 depending on whether $\hat{p}_1(x,c) - q_1 > 0$.

Optimal solution: $\max_{x \in [0,1]} Gain(x)$ is the optimal number of clicks in the 2×2 case. For a given class of distribution $\mathcal{P}$, the optimal x can be obtained numerically. For computational efficiency, we use a Normal approximation.

Normal approximation: Assume $\hat{p}_1(x,c)$ is approximately Normal. Note that here we only approximate the posterior $(p_1 \,|\, x, c)$ by Normal; the prior p_0 is still assumed to be Gamma. Let σ_0^2 denote the variance of p_0; $\sigma_0^2 = \alpha/\gamma^2$ for the Gamma distribution. By a derivation using iterated



(a) Gain function (b) Optimal solution

Fig. 1. (a) shows the $Gain(x)$ for different $\hat{p}_0$ when γ=500, N_0=2K, N_1=40K, q_0=q_1=0.1. (b) shows the optimal solution to x ($\arg\max_x Gain(x)$) as a function of γ.

expectations with respect to distributions $(c \mid p_1)$ and p_1, we obtain:

$$E_c[\hat{p}_1(x,c)] = \hat{p}_0 = \alpha/\gamma,$$
$$Var_c[\hat{p}_1(x,c)] = \sigma_1^2(x) = \tfrac{xN_0}{\gamma + xN_0}\sigma_0^2.$$

Using the Normal approximation, the tail expectation is obtained in closed form and given below.

Proposition 2: Let ϕ and Φ denote the density and distribution functions of the standard normal.

$$Gain(x, \boldsymbol{\theta}_0, q_0, q_1, N_0, N_1) \approx N_0 x(\hat{p}_0 - q_0) + N_1\left[\sigma_1(x)\phi\left(\tfrac{q_1-\hat{p}_0}{\sigma_1(x)}\right) + \left(1 - \Phi\left(\tfrac{q_1-\hat{p}_0}{\sigma_1(x)}\right)\right)(\hat{p}_0 - q_1)\right]$$

The Normal approximation makes $Gain(x)$ a differentiable function with some nice properties. Figure 1(a) shows three Gain functions with three different prior means. In particular, it can be shown that $Gain(x)$ has at most one minimum and at most one maximum (excluding the boundaries). It can also be shown that $\frac{d^2}{dx^2}Gain(x) = 0$ has at most one solution for $0 < x < 1$. Let C denote the solution, if it exists. One can also show that $\frac{d^2}{dx^2}Gain(x) > 0$ for $0 < x < C$, and $\frac{d^2}{dx^2}Gain(x) < 0$; thus, $\frac{d}{dx}Gain(x)$ is decreasing for $C < x < 1$. Now, binary search can be used to efficiently find $C < x^* < 1$ such that $\frac{d}{dx}Gain(x) = 0$, if it exists. Then, the optimal solution is $x = 0$, x^* or 1.

Proposition 3: Let x^* denote the optimal solution to the normal approximation. The time complexity of finding solution x such that $|x - x^*| < \epsilon$ is $O(\log 1/\epsilon)$.

Properties of the gain function: We note some interesting properties of the gain function as shown in Figure 1. In particular, Figure 1(b) shows the optimal amount of exploration as a function of uncertainty (small γ means high uncertainty) for different mean values of the uncertain item. Contrary to what one might expect, we note that the amount of exploration is not monotonically decreasing as uncertainty goes down (i.e., γ goes up); we in fact should not explore too much when the degree of uncertainty is too high (i.e., small γ). The scheme is cautious and does not allocate too many observations to items that have high degree of uncertainty.

B. K×2 Case: K Items, 2 Intervals

We now consider the case of K items (without distinguishing between certain and uncertain ones), but still use two intervals. The optimal solution to this problem can be defined exactly (details omitted), but solving for it is computationally hard. Thus, we apply the Lagrange relaxation technique to find a near optimal solution. Although the Lagrange relaxation [2] is well-known, the application to our problem is novel.

Recall that $p_{it} \sim \mathcal{P}(\boldsymbol{\theta}_{it})$ denotes the CTR of item i at time $t \in \{0,1\}$. Let $\mu(\boldsymbol{\theta}_{it}) = E[p_{it}]$. To simplify notations, we use vectors: $\boldsymbol{\theta}_t = [\boldsymbol{\theta}_{1t}, ..., \boldsymbol{\theta}_{Kt}]$, $\mathbf{x_t} = [x_{1t}, ..., x_{Kt}]$ and $\mathbf{c_0} = [c_{10}, ..., c_{K0}]$, where x_{it} is the fraction of views allocated to item i at time $t \in \{0,1\}$, and c_{i0} denotes the random number of clicks item i obtained at time 0. Our goal is the obtain $\mathbf{x_0}$ and $\mathbf{x_1}$ that maximizes the total expected number of clicks in the two intervals. When we make this decision, $\boldsymbol{\theta}_0$ is known and $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_1(\mathbf{x_0}, \mathbf{c_0})$ depends on $\mathbf{x_0}$ and $\mathbf{c_0}$. Also, $\mathbf{x_0}$ is a vector of numbers, but $\mathbf{x_1} = \mathbf{x_1}(\boldsymbol{\theta}_1)$ is a function of $\boldsymbol{\theta}_1$. Let $\mathbf{x} = [\mathbf{x_0}, \mathbf{x_1}]$

Now, the expected total number of clicks is

$$R(\mathbf{x}, \boldsymbol{\theta}_0, N_0, N_1) = N_0 \textstyle\sum_i x_{i0}\mu(\boldsymbol{\theta}_{i0}) + N_1 \sum_i E_{\boldsymbol{\theta}_1}[x_{i1}(\boldsymbol{\theta}_1)\mu(\boldsymbol{\theta}_{i1})].$$

Our goal is to find

$$R^*(\boldsymbol{\theta}_0, N_0, N_1) = \max_{0 \le \mathbf{x} \le 1} R(\mathbf{x}, \boldsymbol{\theta}_0, N_0, N_1), \text{ subject to } \textstyle\sum_i x_{i0} = 1 \text{ and } \sum_i x_{i1}(\boldsymbol{\theta}_1) = 1, \text{for all possible } \boldsymbol{\theta}_1.$$

Lagrange relaxation: To make the above optimization computationally feasible, we relax the constraints on interval 1. Instead of requiring $\sum_i x_{i1}(\boldsymbol{\theta}_1) = 1$, *for all possible* $\boldsymbol{\theta}_1$, we only require $\sum_i x_{i1}(\boldsymbol{\theta}_1) = 1$ on average. Thus, the optimization problem becomes:

$$R^+(\boldsymbol{\theta}_0, N_0, N_1) = \max_{0 \le \mathbf{x} \le 1} R(\mathbf{x}, \boldsymbol{\theta}_0, N_0, N_1), \text{ subject to } \textstyle\sum_i x_{i0} = 1 \text{ and } E_{\boldsymbol{\theta}_1}\left[\sum_i x_{i1}(\boldsymbol{\theta}_1)\right] = 1.$$

Next, we define the value function V as:

$$V(\boldsymbol{\theta}_0, q_0, q_1, N_0, N_1) = \max_{0 \le \mathbf{x} \le 1} \{R(\mathbf{x}, \boldsymbol{\theta}_0, N_0, N_1) - q_0 N_0(\Sigma_i\ x_{i0} - 1) - q_1 N_1(E[\Sigma_i\ x_{i0}] - 1)\},$$

where q_0 and q_1 are the Lagrange multipliers. Under mild regulatory conditions,

$$R^+(\boldsymbol{\theta}_0, N_0, N_1) = \min_{q_0, q_1} V(\boldsymbol{\theta}_0, q_0, q_1, N_0, N_1).$$

We now state two important properties of the V function that simplifies computation.

Proposition 4: **Convexity:** $V(\boldsymbol{\theta}_0, q_0, q_1, N_0, N_1)$ is convex in (q_0, q_1).

Since V is convex in (q_0, q_1), standard non-differential convex optimization tools can be used to find the minimum solution. However, we need to compute V efficiently given (q_0, q_1). Fortunately, this can be done due to the separability property stated below.

Proposition 5: **Separability:** $V(\boldsymbol{\theta}_0, q_0, q_1, N_0, N_1)$=

$$\sum_i \left(\max_{0 \le x_{i0} \le 1} Gain(x_{i0}, \boldsymbol{\theta}_{i0}, q_0, q_1, N_0, N_1)\right) + q_0 N_0 + q_1 N_1,$$

where $Gain$ is defined in Proposition 1.

Due to separability, the V function can be computed by maximization (over x_{i0}) for each item i *independently*. This independent maximization reduces to the gain maximization discussed in Section II-A and can be solved efficiently. Thus, we are able to solve a *joint* maximization (over $x_{10}, ..., x_{K0}$) problem in a K-dimensional space through K independent one dimensional optimization. This *decoupling* is similar in spirit to Gittins (1979) index policy calculation.

Near optimal solution: To compute the fraction of user visits allocated to each item i in the coming interval (interval 0), we use a standard convex optimizer to compute $\min_{q_0,q_1} V(\boldsymbol{\theta}_0, q_0, q_1, N_0, N_1)$. Let q_0^* and q_1^* be the minimum solution. Then,

$$x_{i0}^* = \arg \max_{0 \le x_{i0} \le 1} Gain(x_{i0}, \boldsymbol{\theta}_{i0}, q_0^*, q_1^*, N_0, N_1)$$

is the fraction to be given to item i. Note that the Langrange relaxation technique was first applied by [2] to bandit problems. Studies of several related (but different) problems suggest that Lagrange relaxation usually provides near optimal solutions as [11] noted "a developing body of empirical evidence testifies to the strong performance of Whittle's index [Lagrange-relaxation based] policies."

C. General Solution

We now describe the solution to the general case.

Dynamic set of items: We extend the near-optimal solution for the K×2 case to a dynamic set of items with multiple future intervals. Now, the set $\mathcal{I}_t$ of items is allowed to change over time. Let $s(i)$ and $e(i)$ denote the start time and end time of item i. The end time of an item may be stochastic; this can be incorporated in our framework by marginalizing over the value function. Here, we assume a deterministic $e(i)$ for ease of exposition. $\mathcal{I}_0$ is the set of items i such that $s(i) \le 0$ and $e(i) \ge 0$, called *live items* which start before the current time $t = 0$. Let $T = \max_{i \in \mathcal{I}_0} e(i)$ denote the end time of the live item having the longest remaining lifetime. Let $\mathcal{I}^+$ be the set of items i with $1 \le s(i) \le T$, called *future items*. Let $T(i) = \min\{T, e(i)\}$.

We note that, after we apply Lagrange relaxation, the convexity and separability properties still hold (with slightly modified formulae), but computational complexity increases exponentially in the number of intervals. For efficiency, we propose to approximate the multi-interval case by only considering two stages for each item i: The first interval of the item ($\max\{0, s(i)\}$) is the exploration stage, while the remaining intervals ($\max\{0, s(i)\}+1$ to $T(i)$) of the item are the exploitation stage. These two stages correspond to $t = 0$ and $t = 1$ in the K×2 case, respectively. It is important to note that the two-stage approximate is only used to compute the sampling plan for interval $t = 0$; here we do not decide to actually do pure exploitation for $t = 1, ..., T(i)$. At $t = 1$, we would consider $t = \max\{1, s(i)\}$ as the exploration stage for item i and compute the sampling plan for it based on the data observed at $t = 0$ and before.

After this two-stage approximation, the objective function V (in Proposition 5) becomes:

$$\begin{aligned} &V(\boldsymbol{\theta}_0, q_0, q_1, N_0, ..., N_T) = \\ &\quad \sum_{i \in \mathcal{I}_0} \max_{0 \le x_{i0} \le 1} Gain(x_{i0}, \boldsymbol{\theta}_{i0}, q_0, q_1, N_0, \sum_{t=1}^{T(i)} N_t) \\ &\quad + \sum_{i \in \mathcal{I}^+} \max_{0 \le y_i \le 1} Gain(y_i, \boldsymbol{\theta}_{i0}, q_1, q_1, N_{s(i)}, \sum_{t=s(i)+1}^{T(i)} N_t) \\ &\quad + q_0 N_0 + q_1 \textstyle\sum_{t \in [1,T]} N_t. \end{aligned}$$

We apply standard convex minimization techniques to find q_0^* and q_1^* that minimize the V function. The x_{i0} that maximizes the above $Gain$ function (on the 2nd line) with $q_0 = q_0^*$ and $q_1 = q_1^*$ is the fraction of user visits to be given to item i in the next time interval. We now explain the V function:

- *Live items vs. future items*: Live items $\mathcal{I}_0$ (on the 2nd line) requires a different treatment from future items $\mathcal{I}^+$ (on the 3rd line). For each item, we apply the two-stage approximation. For live item $i \in \mathcal{I}_0$, the first stage is time 0 having N_0 views, while the second stage is from 1 to $T(i)$ having $\sum_{t \in [1,T(i)]} N_t$ views. For future item $i \in \mathcal{I}^+$, the first stage is $s(i) > 0$, and the second stage is from $s(i)+1$ to $T(i)$. Since our goal is to decide how to serve live items at time 0 (i.e., x_{i0}), we use different variables y_i for future items i that enter the system later than time 0.
- *Lagrange multipliers*: q_0 and q_1 are used to ensure the optimizer-allocated number of views matches the specified total number of views. In fact, q_0 ensures $\sum_i x_{i0} N_0$ = N_0, for time 0; and q_1 ensures $E[\sum_{t \in [1,T]} \sum_i x_{it} N_t]$ = $\sum_{t \in [1,T]} N_t$, for time 1 to T. Note that, future items $\mathcal{I}^+$ only occur in time 1 to T. Thus, their $Gain$ functions (on the 3rd line) have two occurrences of q_1 and no q_0.
- *State*: $\boldsymbol{\theta}_{i0}$ represents our current belief about the CTR of item i. For live item $i \in \mathcal{I}_0$, $\boldsymbol{\theta}_{i0}$ is the current state of the item's CTR model. For future item $i \in \mathcal{I}^+$, $\boldsymbol{\theta}_{i0}$ is the prior belief of the item's CTR (possibly predicted using item features).

Dynamic Gamma-Poisson (DGP): We incorporate non-stationarity into our solution by using time-series models to update the distribution of item CTR over time. In general, any model that estimates the predictive distribution of CTR accurately can be plugged into our Bayesian solution. Following [9], we use the Dynamic Gamma-Poisson model. Assume CTR $p_{i,t-1}$ of item i at time $t-1$ follows *Gamma*(α, γ). We capture temporal variation in CTR by giving more weight to recent data. One simple way is to down-weight the effective sample size γ after each interval, i.e., the prior at t is centered around the posterior mean at $t-1$ with variance obtained by inflating the posterior variance at $t-1$ (the variance depends on the effective sample size) through a "discounting" factor w [23]. More specifically, after observing c clicks and v views for item i at time t, the prior at time t is $p_{i,t} \sim$ *Gamma*$(w\alpha+c, w\gamma+v)$, where $w \in (0, 1]$ is a pre-specified discount factor, tuned on training data. Incorporating the DGP model into the solution

to the 2×2 case is straightforward. When computing the $Gain$ function, we down-weight α and γ for the second interval by w. Specifically, in the Normal approximation, we redefine:

$$Var_c[\hat{p}_1(x,c)] = \sigma_1(x)^2 \equiv \frac{xN_0}{w\gamma + xN_0}\sigma_{0w}^2, \text{where } \sigma_{0w}^2 = \frac{\alpha}{w\gamma^2}$$

BayesGeneral and Bayes2×2: We call the above general solution the *BayesGeneral* scheme, which requires solving a two-dimensional convex, non-differentiable minimization problem (for q_0 and q_1). To ensure the constraints are satisfied, the minimization needs high precision, which leads to long execution time. To achieve better efficiency, we consider the following approximation and call the resulting scheme *Bayes2×2*. We approximate q_0 and q_1 by the CTR of item i^* that has the highest estimated CTR $\max_i \mu(\boldsymbol{\theta}_{i,0})$, and then find the optimal solution to each x_{i0} with fixed q_0, q_1. The intuition is that we compare each item i with the best item i^* using the 2×2 case assuming we are certain about the CTR of i^*. Since $\sum_i x_{i0}$ may not sum to unity, we set the fraction allocated to item i as $\rho\, x_{i0}$, where ρ is a global tuning parameter. Also, since comparing i^* to itself is not appropriate (which makes x_{i^*0} always 1), we set the fraction given to i^* to be $\max\{1 - \sum_{i\neq i^*} \rho\, x_{i0},\ 0\}$. In Section IV, we show that the performance of *Bayes2×2* is close to *BayesGeneral*. We note that ρ is tuned based on simulations, and we find the performance of *Bayes2×2* to be not very sensitive to the setting of ρ.

III. Non-Bayesian Solutions

In this section, we adapt two existing schemes from the standard multi-armed bandit literature (popular in Machine Learning) to our scenario and describe baseline methods which are empirically compared in Section IV. For ease of exposition, let $\hat{p}_{it} = E[p_{it}]$ estimated by the DGP model.

B-UCB1: UCB1 [3] is a popular scheme designed for one-at-a-time serving – it serves an incoming page view with item i that has the highest *priority*; the item priorities are updated after each feedback that is assumed to be instantaneous. The priority for item i is given by $\hat{p}_{it} + (\frac{2\ln n}{n_i})^{\frac{1}{2}}$, where n_i is the total number of views given to item i so far and $n = \sum_i n_i$. This does not work in our scenario. We modify this scheme as follows.

- To incorporate non-stationary CTR, we use the DGP model with state $[\alpha_{it}, \gamma_{it}]$ for item i to estimate $\hat{p}_{it} = \alpha_{it}/\gamma_{it}$ and replace n_i with the down-weighted sample size γ_{it}.
- To adapt to time-varying $\mathcal{I}_t$, we replace the n with $\sum_{i\in\mathcal{I}_t} \gamma_{it}$.
- For batch serving, we propose a *hypothetical run* technique. The idea is to pretend that we run UCB1 to serve every page view in the next interval one by one. Since we have not actually served any page, we would not observe any reward after each "hypothetical" serve, which is required to run UCB1. A reasonable strategy is to pretend the reward of serving item i to be its current CTR estimate. Then, after hypothetically running through all the user visits in the next interval, we decide the fraction of views given to item i in the interval to be the fraction of hypothetical serves with item i.

Putting pieces together, we sequentially go through each of the N_t user visits in the next interval as follows. For k = 1 to N_t, we hypothetically give the kth page view to item i^* that has the highest priority: Let m_i be a counter that counts the number of views that have been given to item i so far (before k) updated during the hypothetical run, $n_i = \gamma_{it} + m_i$ and $n = \sum_{i\in\mathcal{I}_t} n_i$; the priority for item i is:

$$\text{Untuned:}\quad \hat{p}_{it} + \left(\frac{2\ln n}{n_i}\right)^{\frac{1}{2}},\ \text{or}$$

$$\text{Tuned:}\quad \hat{p}_{it} + \left(\frac{\ln n}{n_i}\min\left\{\frac{1}{4},\ \mathit{Var}(i) + \sqrt{\frac{2\ln n}{n_i}}\right\}\right)^{\frac{1}{2}},$$

where $\mathit{Var}(i) = \hat{p}_{it}(1-\hat{p}_{it})$. After the hypothetical run finishes, m_i is the number of user visits to be given to item i for time t. Thus, we set $x_{it} = m_i/\Sigma_i\, m_i$. We call this scheme Batch-UCB1 or B-UCB1. We note that the tuned version uniformly outperforms the untuned version in our experiments. Thus, we only report the former.

B-POKER: The POKER scheme [24] is similar to UCB1 but has a different priority function. Let $K = |\mathcal{I}_t|$. Without loss of generality, assume $\hat{p}_{1t} \geq ... \geq \hat{p}_{Kt}$. We adapt POKER to our setting by following the B-UCB1 procedure, and only replace the priority function with

$$\hat{p}_{it} + \Pr(p_{it} \geq \hat{p}_{1t} + \delta)\ \delta H,$$

where $\delta = (\hat{p}_{1t} - \hat{p}_{\sqrt{K},t})/\sqrt{K}$, H is a tuning parameter, and the tail probability is computed by assuming $p_{it} \sim \mathcal{P}(\alpha_{it} + m_i\hat{p}_{it},\ \gamma_{it} + m_i)$, where $\mathcal{P}$ is Gamma.

Baseline schemes: *ϵ-Greedy* is a simple scheme that, for each interval, allocates a fixed fraction ϵ of user visits to explore all live items uniformly, and gives the rest of traffic to the item having the highest estimated CTR. *SoftMax* is another simple scheme that sets $x_{it} \propto e^{\hat{p}_{it}/\tau}$, where temperature τ is a tuning parameter. *Exp3* [16] is designed for adversarial, non-stationary reward distributions. Let G_i (initially 0) denote the total number of "adjusted" clicks that item i has received so far. Let $\epsilon \in (0,1]$ and $\eta > 0$ be two tuning parameters. For each time interval t, we (1) give item i fraction x_{it} of the user visits at time t, where $x_{it} = (1-\epsilon)r_{it} + \epsilon/|\mathcal{I}_t|$ and $r_{it} \propto e^{\eta G_i}$, and (2) at the end of interval t, assuming item i receives c_{it} clicks, update G_i by $G_i = G_i + c_{it}/x_{it}$. For comparison purposes, we also include non-batch UCB1 and POKER, called *WTA-UCB1* and *WTA-POKER*, where WTA stands for "winner takes all", which means we allocate entire traffic in an interval to the single item that has the highest priority value.

IV. Experimental Results

In this section, we report on a series of extensive empirical evaluation for the proposed serving schemes. We start by evaluating the current application scenario with a set of roughly 20 live items to choose from at any point in time, we then evaluate hypothetical scenarios where number of live items range from 10 to 1000 followed by analysis that demonstrates

the benefit of applying multi-armed bandit schemes on user segments. This is followed by results obtained from bucket tests conducted on small random fraction of traffic from the actual system.

A. Comparative Analysis

Real application scenario: We study performance in our current application scenario that has a set of roughly 20 live items in each 5-minute interval. To setup the experiment, we collected four months of past data to obtain the set of live items and the number of user visits for each interval retrospectively, and estimated the *ground truth* CTR of each item in each interval through a *loess* fit with bandwidth selected to minimize the autocorrelations in the residuals as proposed in [9]. It is important to note that our data is collected from a *random bucket*, which is a set of users, to whom we show every live item with an equal number of user visits. The bucket was setup to be large enough to reliably estimate the CTR of *every* live item in *every* time interval. This kind of randomized data is very different from regular log data collected from a system that runs some serving method, which causes serving bias in data. To evaluate the performance of schemes under different traffic conditions, we set $N'_t = a \cdot N_t$ for different a values. Here, N_t is the actual number of views observed in the data in time interval t. The number of clicks c_{it} on item i at time t for an allocation volume N'_{it} calculated through the scheme is simulated from *Poisson*$(p_{it}\, N'_{it})$ (the ground-truth CTR is unobservable, each scheme only obtains the clicks). All schemes except Exp3 use the posterior mean estimate from the DGP model to estimate the non-stationary CTR from the observed clicks; the Bayesian schemes, B-UCB1 (where n_i is the effective sample size from the model that is equivalent to knowing the variance), WTA-UCB1, B-POKER, WTA-POKER also use the variance estimate from the model. We set aside first-month's data to determine the tuning parameters for schemes (if they have some), and test all schemes on the remaining three months. To tune the parameters for a scheme, we tried a number (10-20) of parameter settings; for each of the settings, we ran a simulation over the first-month data, and then pick the setting that gives the best performance. Figure 2(a) shows the percentage regret of each scheme as a function of the amount of traffic (in terms of the average number of user visits per interval). The percentage regret of a scheme S is defined as $\frac{\#Clicks(Opt)-\#Clicks(S)}{\#Clicks(Opt)}$, where Opt is the oracle optimal scheme assuming the ground truth is completely known. We note that Opt may pick different items across intervals and provides a stronger notion of regret than the ones used in standard bandit problems (pick a single best item for all intervals, see [16] for more details).

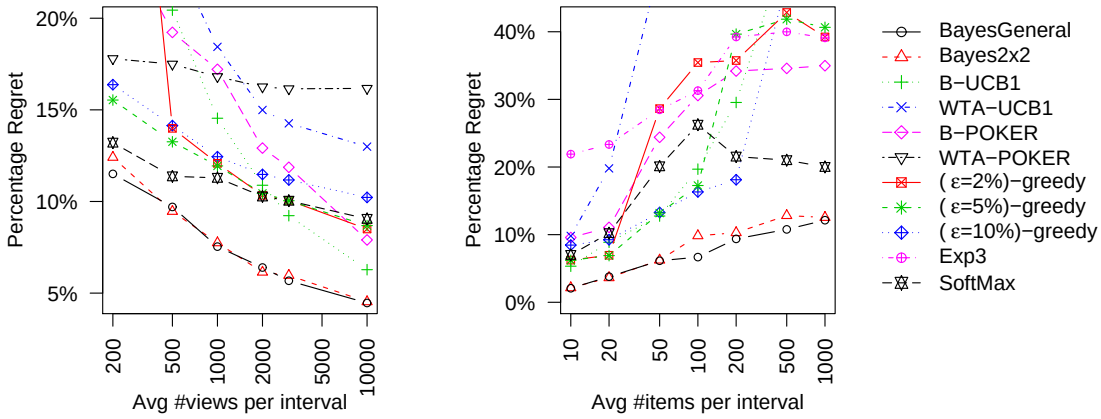
Hypothetical scenario: Several hypothetical scenarios were created by varying the number of live items in an interval. For each scenario, we fix the traffic volume to 1000 views per interval, the lifetime of each item is sampled from Poisson with mean 20 intervals, and the ground-truth CTR of each item is sampled from Gamma with mean and variance estimated from the real application data. Figure 2(b) shows the percentage regret of each scheme as a function of number of live items in each interval. We note that since the data here is synthetically generated, the regret numbers may not match those in Figure 2(a).

Summary of Results: We summarize the results from our experiments. (1) BayesGeneral and Bayes2×2 are uniformly better than all other schemes; differences in performance get larger with increasing data sparseness. Note that the more computationally efficient Bayes2×2 (with tuned ρ) closely approximates BayesGeneral. (2) Batch schemes (B-UCB1 and B-POKER) generally have better performance than their non-batch versions respectively, especially when the number of user visits per interval is large. However, with extreme data sparseness, WTA-POKER outperforms B-POKER. (3) ϵ-Greedy schemes generally provide reasonable performance, but the right ϵ depends on the application. (4) Exp3 usually has the worst performance probably because it was designed for the adversarial setting; SoftMax on the other hand provides reasonable performance with a carefully tuned temperature τ. All assertions made above are statistically significant and confirmed by experiments on several additional datasets with several replications on each.

B. Scheme characterization

We now provide more intuition on the "explore/exploit" properties of each scheme, this also helps us understand why Bayesian schemes perform better than others. Note that a scheme that is too impatient would allocate more user visits to the item with highest posterior mean too quickly and is likely to lose clicks. On the other hand, a scheme that is too cautious in allocating large fraction of user visits to the higher posterior mean items may be too slow to react to opportunities. To quantify this delicate difference between explore/exploit properties of bandit schemes, we characterize a bandit scheme by three criteria. At a given time point, let us call the item having the highest estimated CTR (estimated by the scheme) the EMP (estimated most popular) item. Note that different schemes may pick different EMP items at the same time point. Let n_{it} denote the number of views that the scheme allocates to item i at time t. Let p_{it} denote the true CTR of item i at time t, and $p^*_t = \max_i p_{it}$. Without loss of generality, assume $i = 1$ is the EMP item determined by the scheme for any point in time. We define the following three metrics to characterize a scheme:

- Fraction of EMP display: $\frac{\sum_t n_{1t}}{\sum_t \sum_i n_{it}}$ is the fraction of views when the scheme displays EMP items. This number quantifies the amount of traffic in which the scheme *exploits* its current knowledge about the items.
- EMP regret: $\frac{\sum_t n_{1t}\,(p^*_t - p_{1t})}{\sum_t n_{1t}}$ is the regret of displaying the EMP item. This number quantifies the scheme's ability of identifying the optimal item (or a good item) to exploit. When a scheme explores less than necessary, its EMP regret is likely to be high because it does not have enough observations to identify the best item.
- Non-EMP regret: $\frac{\sum_t \sum_{i\neq 1} n_{it}\,(p^*_t - p_{it})}{\sum_t \sum_{i\neq 1} n_{it}}$ is the regret of



(a) Front Page scenario

(b) Hypothetical scenario

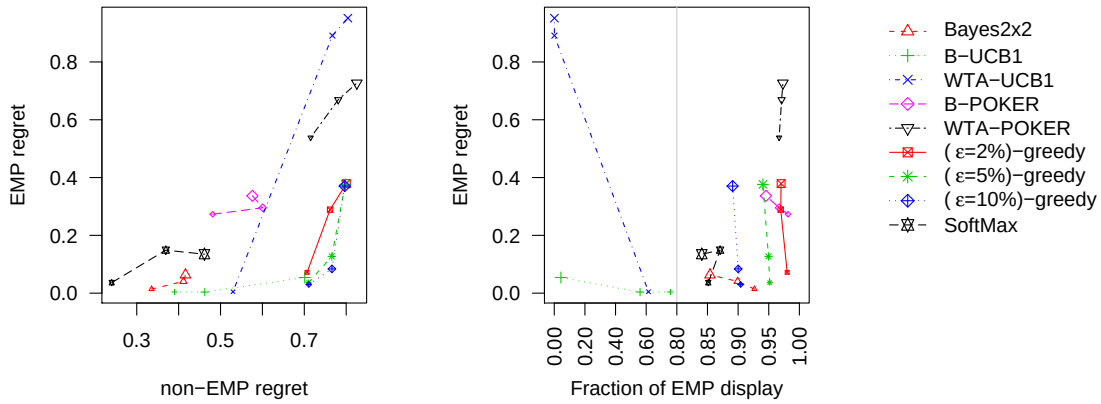
Fig. 2. Empirical comparison of explore/exploit schemes. x-axes are log-scaled. In (a), WTA-UCB1 and Exp3 have regrets more than 20%. In (b), WTA-POKER has regrets more than 40%.

displaying all the non-EMP items. This number quantifies the cost of exploration because when the scheme knows the CTRs of items exactly, it should always display the EMP item.

Figure 3(a) shows EMP regret vs. non-EMP regret for each scheme. We ran three simulations (thus, three points) for each scheme with the numbers of items per interval being 20, 100 and 1000. The simulation setup is the same as that of the hypothetical scenarios. In this plot, good schemes are at the bottom left corner. Figure 3(b) shows EMP regret vs. fraction of EMP display for each scheme. Good schemes are at the bottom right corner. Observe that: (1) The Bayesian scheme is among the best in both plots; even when the number of items is large, its performance is still good. (2) Non-batch schemes (WTA-UCB1 and WTA-POKER) usually are not able to identify the best items because they only show a *single item* for each time interval; when item lifetimes are only 20 intervals on average, they are not able to collect enough observations to identify good items before the items retire. (3) B-UCB1 has low EMP regret, indicating its capability of identifying good items with much error; however, it usually explores more than necessary. (4) The three ϵ-greedy schemes have very similar characteristics. Their fractions of EMP display are fixed at $1-\epsilon$. Their non-EMP regrets are large due to complete randomization. As the number of items gets large, their ability of identifying good items gets worse. (5) SoftMax is quite competitive especially when the number of items is small; however its overall performance is significantly worse than the Bayesian scheme for all simulation settings. Moreover, we found this scheme to be overly sensitive to the temperature parameter that has to be tweaked carefully.

C. Segmentation Analysis

We now demonstrate performance of our schemes for personalized recommendations by running them separately on user segments created based on user features, which includes age, gender, geo-location and browse behavior (search history, pages visited, ads clicked, etc.). Several other features were evaluated but found to be weakly predictive. Each item was hand labelled and assigned to one of the C content categories. Using large amount of retrospective data, we generated user segments as follows. Let y_{uit} denote the response (click or no-click) of user u on item i at time t, and x_{ut} denote the user features (browse behavior is dynamic, hence the suffix t). Then, we fit a logistic regression model $y_{uit} \mid p_{uit} \sim Bernoulli(p_{uit})$, where $\log \frac{p_{uit}}{1-p_{uit}} = x'_{ut}\boldsymbol{\beta}_{c(i)}$; $\boldsymbol{\beta}_k$'s are latent factors for category k; and $c(i)$ denote the category of item i. Using the estimated $\boldsymbol{\beta}_k$'s, we project each user with feature x_{ut} into the category space as $[x'_{ut}\boldsymbol{\beta}_1, ..., x'_{ut}\boldsymbol{\beta}_C]$. Then, we cluster these projections and select 5 user segments (cluster) after careful analysis. The segments were also analyzed and interpreted by editors who *program* stories in our application. We note that segmentation provides better interpretability for editors than the regression model in order to program targeted content items. In Figure 4, we show the CTR lift (relative to the oracle optimal scheme without segmentation) for each scheme. We note that the maximum achievable lift is about 13%. The Bayesian scheme is uniformly better than all other schemes and provides positive lift for all traffic volumes. B-UCB1 and ϵ-greedy perform reasonably well for large traffic volumes, but they breakdown when traffic volume gets small. Surprisingly, SoftMax does not perform well in this experiment. This is because we used a single tuned temperature τ for all the segments. Since τ is sensitive to the scale of item CTR's and the segments have very different item CTR distributions, a single τ value for all segments leads to bad performance.



(a) Regret: EMP vs. non-EMP

(b) EMP regret vs. fraction of EMP

Fig. 3. Characteristics of bandit schemes. Each point represent a simulation of a scheme. For each scheme, we ran three simulations with 20 (small point), 100 (medium point) and 1000 (large point) items per interval. BayesGeneral is omitted since it has almost the same characteristics as Bayes2×2. Note the x-axis of (b); we provide a zoom-in view for $x \geq 0.8$.

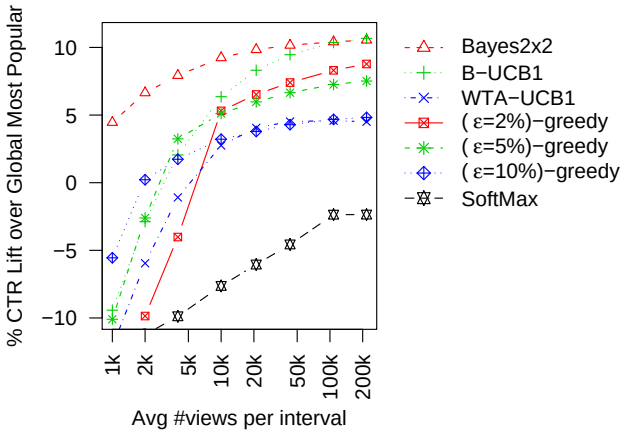


Fig. 4. Segmentation analysis

D. Bucket Test Results

We now report results of experiments that were conducted on a fraction of live traffic on a major internet portal.

Experimental setup: To setup an experiment, we create a number of equal-sized random samples, called buckets, of users. Each user is identified by an identifier (cookie) saved in the user's web browser by the website. We exclude users who do not accept cookies from our experiment; the size of this user population is relatively small and does not affect the validity of our experiments. In each bucket, we run a different serving scheme and compare the performance of multiple schemes over the same time period. In this study, we consider three schemes, Bayes2×2, B-UCB1 and ϵ-greedy, and use the overall CTR of a bucket over a two-week period as our performance measure for all methods. Ideally, we would like a scheme to have full control over all the traffic in its allocated bucket. However, because of concerns regarding user experience being hurt through excessive exploration since we were testing several schemes simultaneously (e.g., having undesired side-effects like attrition), each scheme was only allowed a maximum of 15% of the bucket to explore at any given time; i.e., the remaining 85% of the bucket has to show the current EMP as determined by the scheme. We call 15% the *explore traffic* and the remaining 85% the *exploit traffic*, and report the performance of the two parts separately. Both Bayes2×2 and B-UCB1 assign 85% of views to the item having the highest estimated CTR and allocate the remaining 15% using the scheme but after incorporating feedback from 85% views that have already been allocated to the highest-CTR item. For ϵ-greedy, we use 15% to explore every available item randomly. For confidentiality reasons, we do not disclose the actual CTRs of the buckets and only report CTR lift relative to a bucket that serves every available item randomly.

Experimental result: Table I shows the results. In this experimental setup, all there schemes have almost the same CTR lift in the exploit traffic. However, in the explore traffic, it is clear that Bayes2×2 is significantly better than B-UCB1, which is significantly better than random. With roughly 270 views per 5-minute interval to explore roughly 20 items, all schemes can easily find the current best item, but Bayes2×2 finds it more economically. Figure 5 shows the same results but on a daily basis. The Bayes2×2 dominate others uniformly across days in the explore bucket.

V. Conclusion

We proposed several sequential schemes for systems with dynamic item pools, item lifetime constraints, non-stationary reward distributions and delayed feedback. Our schemes were

Serving Scheme	CTR Lift in Explore	CTR Lift in Exploit	#Views in Two Weeks
Bayes2×2	35.7%	38.7%	7,781,285
B-UCB1	12.2%	40.1%	7,753,184
ϵ-greedy	0.0%	39.4%	7,805,165

Note that any difference less than 5% is not statistically significant. The average number of views per interval in explore is roughly 270.

TABLE I
BUCKET TESTING RESULTS

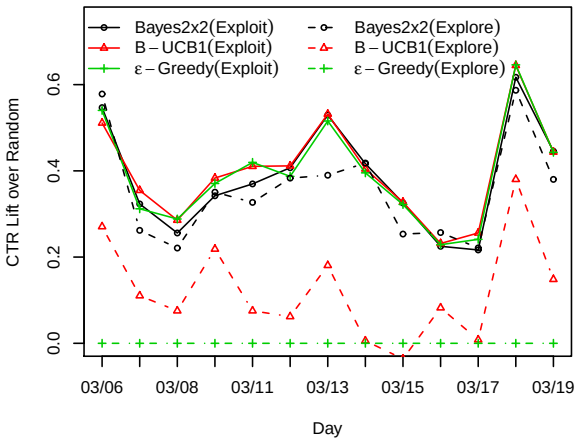


Fig. 5. Live traffic results over a two-week period

rigorously evaluated on an actual content publishing application. We find Bayesian solutions in the presence of an accurate predictive model outperform adaptations of standard bandit schemes. Sequential schemes proposed in this paper are applicable in a variety of web applications and the results obtained shows our Bayesian schemes can be potentially applied to several high-dimensional problems. For instance, in web search, one is confronted with the problem of selecting the top-k documents for every individual query. One can apply information retrieval and other offline models based on features like Page-rank, anchor-text to narrow down the list of relevant documents for a query to a small set. One can then use the techniques proposed in this paper to converge to the best documents for a given query in an economical way. Similar comments apply to advertising where one has to match the best ads to a given query (or web page). For each query, the sample size available is small and effective exploration becomes important for good performance. One can also use statistical models based on covariates to initialize the CTR priors, this helps in faster convergence. Generalizing our methods to optimally select the top-k instead of top-1 is another challenging research problem which we are currently investigating. In fact, the top-k problem ($k > 1$) is no longer a bandit problem but a control problem for which optimal solutions are harder to construct.

REFERENCES

[1] J. Gittins, "Bandit processes and dynamic allocation indices," *J. of the Royal Stat. Society, B*, vol. 41, 1979.
[2] P. Whittle, "Restless bandits: Activity allocation in a changing world," *J. of Applied Probability*, 1988.
[3] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, 2002.
[4] D. Berry and B. Fristedt, *Bandit Problems: Sequential Allocation of Experiments*. Chapman and Hall, 1985.
[5] J. Hauser, G. Urban, G. Liberali, and M. Braun, "Website morphing," *Marketing Science*, forthcoming.
[6] F. Caro and J. Gallien, "Dynamic assortment with demand learning for seasonal consumer goods," *Management Science*, 2007.
[7] F. Radlinski and T. Joachims, "Active exploration for learning rankings from clickthrough data," in *KDD*, 2007.
[8] S. Pandey, D. Chakrabarti, and D. Agarwal, "Multi-armed bandit problems with dependent arms," in *ICML*, 2007.
[9] D. Agarwal, B.-C. Chen, and P. Elango, "Spatio-temporal models for estimating click-through rate," in *WWW*, 2009.
[10] R. Weber and G. Weiss, "On an index policy for restless bandits," *J. of Applied Probability*, 1990.
[11] K. Glazebrook, P. Ansell, R. Dunn, and R. Lumley, "On the optimal allocation of service to impatient tasks," *J. of Applied Probability*, 2004.
[12] D. Bertsimas and J. Nino-Mora, "Restless bandits, linear programming relaxations, and a primal-dual index heuristic," *Operations Research*, 2000.
[13] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of optimal queueing network control," in *Structure in Complexity Theory Conference*, 1994.
[14] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal, "Mortal multi-armed bandits," in *NIPS*, 2008.
[15] A. Slivkins and E. Upfal, "Adapting to a changing environment: The brownian restless bandits," in *COLT*, 2008.
[16] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Shapire, "Gambling in a rigged casino: The adversarial multi-armed bandit problem," in *FOCS*, 1995.
[17] S. Eick, "Gittins procedures for bandits with delayed responses," *J. of the Royal Stat. Society, B*, 1988.
[18] K. Glazebrook and D. Wilkinson, "Index-based policies for discounted multi-armed bandits on parallel machines," *Annals of Applied Probability*, 2000.
[19] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, 1985.
[20] P.Auer, N.Cesa-Bianchi, and P.Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, 2002.
[21] M. DeGroot, *Optimal Statistical Decisions*. John Wiley & Sons, 2004.
[22] J. Sarkar, "One-armed bandit problems with covariates," *The Annals of Statistics*, 1991.
[23] M. West and J. Harrison, *Bayesian Forecasting and Dynamic Models*. Springer-Verlag, 1997.
[24] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *ECML*, 2005.