Handling Advertisements of Unknown Quality in Search Advertising

Sandeep Pandey
Carnegie Mellon University
spandey@cs.cmu.edu

Christopher Olston
Yahoo! Research and Carnegie Mellon University
olston@yahoo-inc.com

Abstract

We consider how a search engine should select advertisements to display with search results, in order to maximize its revenue. Under the standard "pay-per-click" arrangement, revenue depends on how well the displayed advertisements appeal to users. The main difficulty stems from new advertisements whose degree of appeal has yet to be determined. Often the only reliable way of determining appeal is *exploration* via display to users, which detracts from *exploitation* of other advertisements known to have high appeal. Budget constraints and finite advertisement lifetimes make it necessary to explore as well as exploit.

In this paper we study the tradeoff between exploration and exploitation, modeling advertisement placement as a multi-armed bandit problem. We extend traditional bandit formulations to account for budget constraints that occur in search engine advertising markets, and derive theoretical bounds on the performance of a family of algorithms. We measure empirical performance via extensive experiments over real-world data.

1 Introduction

Search engines are invaluable tools for society. Their operation is supported in large part through advertising revenue. Under the standard "pay-per-click" arrangement, search engines earn revenue by displaying appealing advertisements that attract user clicks. Users benefit as well from this arrangement, especially when searching for commercial goods or services.

Successful advertisement placement relies on knowing the appeal or "clickability" of advertisements. The main difficulty is that the appeal of new advertisements that have not yet been "vetted" by users can be difficult to estimate. In this paper we study the problem of placing advertisements to maximize a search engine's revenue, in the presence of uncertainty about appeal.

1.1 The Advertisement Problem

Consider the following *advertisement problem* [9], illustrated in Figure 1. There are m advertisers $A_1, A_2 \ldots A_m$ who wish to advertise on a search engine. The search engine runs a large auction where each advertiser submits its bids to the search engine for the query phrases in which it is interested. Advertiser A_i submits advertisement $a_{i,j}$ to *target* query phrase Q_j, and promises to pay $b_{i,j}$ amount of money for each click on this advertisement, where $b_{i,j}$ is A_i's bid for advertisement $a_{i,j}$. Advertiser A_i can also specify a daily budget (d_i) that is the total amount of money it is willing to pay for the clicks on its advertisements in a day. Given a user search query on phrase Q_j, the search engine selects a constant number $C \geq 1$ of advertisements from the candidate set of advertisements $\{a_{*,j}\}$, targeted to Q_j. The objective in selecting advertisements is to maximize the search engine's total daily revenue. The arrival sequence of user queries is not known in advance. For now we assume that each day a new set of advertisements is given to the search engine and the set remains fixed through out the day; we drop both of these assumptions later in Section 4.

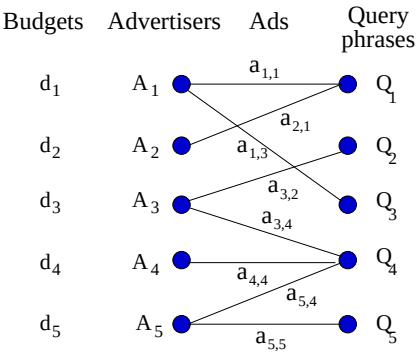

Figure 1: Advertiser and query model.

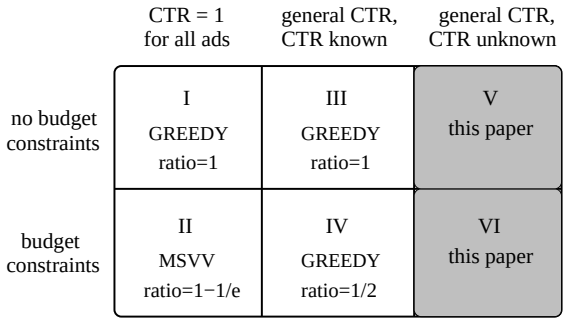

Figure 2: Problem variants.

High revenue is achieved by displaying advertisements that have high bids as well as high likelihood of being clicked on by users. Formally, the *click-through rate* (CTR) $c_{i,j}$ of advertisement $a_{i,j}$ is the probability of a user to click on advertisement $a_{i,j}$ given that the advertisement was displayed to the user for query phrase Q_j. In the absence of budget constraints, revenue is maximized by displaying advertisements with the highest $c_{i,j} \cdot b_{i,j}$ value. The work of [9] showed how to maximize revenue in the presence of budget constraints, but under the assumption that all CTRs are known in advance. In this paper we tackle the more difficult but realistic problem of maximizing advertisement revenue when CTRs are not necessarily known at the outset, and must be learned on the fly.

We show the space of problem variants (along with the best known advertisement policies) in Figure 2. GREEDY refers to selection of advertisements according to expected revenue (*i.e.*, $c_{i,j} \cdot b_{i,j}$). In Cells I and III GREEDY performs as well as the optimal policy, where the optimal policy also knows the arrival sequence of queries in advance. We write "ratio=1" in Figure 2 to indicate that GREEDY has the competitive ratio of 1. For Cells II and IV the greedy policy is not optimal, but is nevertheless 1/2 competitive. An alternative policy for Cell II was given in [9], which we refer to as MSVV; it achieves a competitive ratio of $1 - 1/e$. In this paper we give the first policies for Cells V and VI, where we must choose which advertisements to display while simultaneously estimating click-through rates of advertisements.

1.2 Exploration/Exploitation Tradeoff

The main issue we face while addressing Cells V and VI is to balance the exploration/exploitation tradeoff. To maximize short-term revenue, the search engine should *exploit* its current, imperfect CTR estimates by displaying advertisements whose estimated CTRs are large. On the other hand, to maximize long-term revenue, the search engine needs to *explore*, *i.e.*, identify which advertisements have the largest CTRs. This kind of exploration entails displaying advertisements whose current CTR estimates are of low confidence, which inevitably leads to displaying some low-CTR ads in the short-term. This kind of tradeoff between *exploration* and *exploitation* shows up often in practice, e.g., in clinical trials, and has been extensively studied in the context of the *multi-armed bandit* problem [4].

In this paper we draw upon and extend the existing bandit literature to solve the advertisement problem in the case of unknown CTR. In particular, first in Section 3 we show that the unbudgeted variant of the problem (Cell V in Figure 2) is an instance of the multi-armed bandit problem. Then, in Section 4 we introduce a new kind of bandit problem that we termed the *budgeted multi-armed multi-bandit* problem (BMMP), and show that the budgeted unknown-CTR advertisement problem (Cell VI) is an instance of BMMP. We propose policies for BMMP and give performance bounds. We evaluate our policies empirically over real-world data in Section 5. In Appendix A we show how to extend our policies to address various practical considerations, *e.g.*, exploiting any prior information available about the CTRs of ads, permitting advertisers to submit and revoke advertisements at any time, not just at day boundaries.

2 Related Work

We have already discussed the work of [9], which addresses the advertisement problem under the assumption that CTRs are known. There has not been much published work on estimating CTRs. Reference [8] discusses how contextual information such as user demographic or ad topic can be used to estimate CTRs, and makes connections to the recommender and bandit problems, but stops short of presenting technical solutions. Some methods for estimating CTRs are proposed in [6] with the focus of thwarting click fraud.

Reference [1] studies how to maximize user clicks on banner ads. The key problem addressed in [1] is to satisfy the contracts made with the advertisers in terms of the minimum guaranteed number of impressions (as opposed to the budget constraints in our problem). Reference [10] looks at the advertisement problem from an advertiser's point of view, and gives an algorithm for identifying the most profitable set of keywords for the advertiser.

3 Unbudgeted Unknown-CTR Advertisement Problem

In this section we address Cell V of Figure 2, where click-through rates are initially unknown and budget constraints are absent (*i.e.,* $d_i = \infty$ for all advertisers A_i). Our unbudgeted problem is an instance of the multi-armed bandit problem [4], which is the following: we have K *arms* where each arm has an associated reward and payoff probability. The payoff probability is not known to us while the reward may or may not be known (both versions of the bandit problem exist). With each *invocation* we *activate* exactly $C \leq K$ arms. [1] Each activated arm then yields the associated reward with its payoff probability and nothing with the remaining probability. The objective is to determine a *policy* for activating the arms so as to maximize the total reward over some number of invocations.

To solve the unbudgeted unknown-CTR advertisement problem, we create a multi-armed bandit problem instance for each query phrase Q, where ads targeted for the query phrase are the arms, bid values are the rewards and CTRs are the payoff probabilities of the bandit instance. Since there are no budget constraints, we can treat each query phrase independently and solve each bandit instance in isolation. [2] The number of invocations for a bandit instance is not known in advance because the number of queries of phrase Q in a given day is not known in advance.

A variety of policies have been proposed for the bandit problem, *e.g.*, [2, 3, 7], any of which can be applied to our unbudgeted advertisement problem. The policies proposed in [3] are particularly attractive because they have a known performance bound for any number of invocations not known in advance (in our context the number of queries is not known a priori). In the case of $C = 1$, the policies of [3] make $O(ln\ n)$ number of *mistakes*, on expectation, in n invocations (which is also the asymptotic lower bound on the number of mistakes [7]). A mistake occurs when a suboptimal arm is chosen by a policy (the optimal arm is the one with the highest expected reward).

We consider a specific policy from [3] called UCB and apply it to our problem (other policies from [3] can also be used). UCB is proposed under a slightly different reward model; we adapt it to our context to produce the following policy that we call *MIX* (for mixing exploration with exploitation). We prove a performance bound of $O(ln\ n)$ mistakes for MIX for any $C \geq 1$ in Appendix D.

Policy MIX :

Each time a query for phrase Q_j arrives:

1. *Display the C ads targeted for Q_j that have the highest priority. The priority $P_{i,j}$ of ad $a_{i,j}$ is a function of its current CTR estimate ($\hat{c}_{i,j}$), its bid value ($b_{i,j}$), the number of times it has been displayed so far ($n_{i,j}$), and the number of times phrase Q_j has been queried so far in the day (n_j). Formally, priority $P_{i,j}$ is defined as:*

$$P_{i,j} = \begin{cases} \left(\hat{c}_{i,j} + \sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}\right) \cdot b_{i,j} & \text{if } n_{i,j} > 0 \\ \infty & \text{otherwise} \end{cases}$$

[1] The conventional multi-armed bandit problem is defined for $C = 1$. We generalize it to any $C \geq 1$ in this paper.

[2] We assume CTRs to be independent of one another.

2. *Monitor the clicks made by users and update the CTR estimates $\hat{c}_{i,j}$ accordingly. $\hat{c}_{i,j}$ is the average click-through rate observed so far,* i.e.*, the number of times ad $a_{i,j}$ has been clicked on divided by the total number of times it has been displayed.*

Policy MIX manages the exploration/exploitation tradeoff in the following way. The priority function has two factors: an exploration factor $\left(\sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}\right)$ that diminishes with time, and an exploitation factor ($\hat{c}_{i,j}$). Since $\hat{c}_{i,j}$ can be estimated only when $n_{i,j} \geq 1$, the priority value is set to ∞ for an ad which has never been displayed before.

Importantly, the MIX policy is practical to implement because it can be evaluated efficiently using a single pass over the ads targeted for a query phrase. Furthermore, it incurs minimal storage overhead because it keeps only three numbers ($\hat{c}_{i,j}$, $n_{i,j}$ and $b_{i,j}$) with each ad and one number (n_j) with each query phrase.

4 Budgeted Unknown-CTR Advertisement Problem

We now turn to the more challenging case in which advertisers can specify daily budgets (Cell VI of Figure 2). Recall from Section 3 that in the absence of budget constraints, we were able to treat the bandit instance created for a query phrase independent of the other bandit instances. However, budget constraints create dependencies between query phrases targeted by an advertiser. To model this situation, we introduce a new kind of bandit problem that we call **B**udgeted **M**ulti-armed **M**ulti-bandit **P**roblem (BMMP), in which multiple bandit instances are run in parallel under overarching budget constraints. We derive generic policies for BMMP and give performance bounds.

4.1 Budgeted Multi-armed Multi-bandit Problem

BMMP consists of a finite set of multi-armed bandit instances, $\mathcal{B} = \{B_1, B_2 \ldots B_{|\mathcal{B}|}\}$. Each bandit instance B_i has a finite number of arms and associated rewards and payoff probabilities as described in Section 3. In BMMP each arm also has an associated *type*. Each type $T_i \in \mathcal{T}$ has budget $d_i \in [0, \infty]$ which specifies the maximum amount of reward that can be generated by activating all the arms of that type. Once the specified budget is reached for a type, the corresponding arms can still be activated but no further reward is earned.

With each invocation of the bandit system, one bandit instance from $\mathcal{B}$ is invoked; the policy has no control over which bandit instance is invoked. Then the policy activates C arms of the invoked bandit instance, and the activated arms generate some (possibly zero) total reward.

It is easy to see that the budgeted unknown-CTR advertisement problem is an instance of BMMP. Each query phrase acts as a bandit instance and the ads targeted for it act as bandit arms, as described in Section 3. Each advertiser defines a unique type of arms and gives a budget constraint for that type; all ads submitted by an advertiser belong to the type defined by it. When a query is submitted by a user, the corresponding bandit instance is invoked.

We now show how to derive a policy for BMMP given as input a policy POL for the regular multi-armed bandit problem such as one of the policies from [3]. The derived policy, denoted by *BPOL* (**B**udget-aware POL), is as follows:

- Run $|\mathcal{B}|$ instances of POL in parallel, denoted $\text{POL}_1, \text{POL}_2, \ldots \text{POL}_{|\mathcal{B}|}$.
- Whenever bandit instance B_i is invoked:
  1. Discard any arm(s) of B_i whose type's budget is newly depleted, *i.e.*, has become depleted since the last invocation of B_i.
  2. If one or more arms of B_i was discarded during step 1, restart POL_i.
  3. Let POL_i decide which of the remaining arms of B_i to activate.

Observe that in the second step of BPOL, when POL is restarted, POL loses any state it has built up, including any knowledge gained about the payoff probabilities of bandit arms. Surprisingly, despite this seemingly imprudent behavior, we can still derive a good performance bound for BPOL, provided that POL has certain properties, as we discuss in

the next section. In practice, since most bandit policies can take prior information about the payoff probabilities as input, when restarting POL we can supply the previous payoff probability estimates as the prior (as done in our experiments).

4.2 Performance Bound for BMMP Policies

Let S denote the sequence of bandit instances that are invoked, *i.e.*, $S = \{S(1), S(2) \ldots S(N)\}$ where $S(n)$ denotes the index of the bandit instance invoked at the n^{th} invocation. We compare the performance of BPOL with that of the optimal policy, denoted by OPT, where OPT has advance knowledge of S and the exact payoff probabilities of all bandit instances.

We claim that $bpol(N) \geq opt(N)/2 - O(f(N))$ for any N, where $bpol(N)$ and $opt(N)$ denote the total expected reward obtained after N invocations by BPOL and OPT, respectively, and $f(n)$ denotes the expected number of mistakes made by POL after n invocations of the the regular multi-armed bandit problem (for UCB, $f(n)$ is $O(ln\ n)$ [3]). Our complete proof is rather involved. Here we give a high-level outline of the proof (the complete proof is given in Appendix C). For simplicity we focus on the $C = 1$ case; $C \geq 1$ is a simple extension thereof.

Since bandit arms generate rewards stochastically, it is not clear how we should compare BPOL and OPT. For example, even if BPOL and OPT behave in exactly the same way (activate the same arm on each bandit invocation), we cannot guarantee that both will have the same total reward in the end. To enable meaningful comparison, we define a *payoff instance*, denoted by I, such that $I(i, n)$ denotes the reward generated by arm i of bandit instance $S(n)$ for invocation n in payoff instance I. The outcome of running BPOL or OPT on a given payoff instance is deterministic because the rewards are fixed in the payoff instance. Hence, we can compare BPOL and OPT on per payoff instance basis. Since each payoff instance arises with a certain probability, denoted as $\mathbb{P}(I)$, by taking expectation over all possible payoff instances of execution we can compare the expected performance of BPOL and OPT.

Let us consider invocation n in payoff instance I. Let $B(I, n)$ and $O(I, n)$ denote the arms of bandit instance $S(n)$ activated under BPOL and OPT respectively. Based on the different possibilities that can arise, we classify invocation n into one of three categories:

- *Category 1:* The arm activated by OPT, $O(I, n)$, is of smaller or equal expected reward in comparison to the arm activated by BPOL, $B(I, n)$. The expected reward of an arm is the product of its payoff probability and reward.
- *Category 2:* Arm $O(I, n)$ is of greater expected reward than $B(I, n)$, but $O(I, n)$ is not available for BPOL to activate at invocation n due to budget restrictions.
- *Category 3:* Arm $O(I, n)$ is of greater expected reward than $B(I, n)$ and both arms $O(I, n)$ and $B(I, n)$ are available for BPOL to activate, but BPOL prefers to activate $B(I, n)$ over $O(I, n)$.

Let us denote the invocations of category k (1, 2 or 3) by $\mathcal{N}^k(I)$ for payoff instance I. Let $bpol_k(N)$ and $opt_k(N)$ denote the expected reward obtained during the invocations of category k (1, 2 or 3) by BPOL and OPT respectively. In Appendix C we show that

$$bpol_k(N) = \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^k(I)} I(B(I, n), n) \Big)$$

Similarly,

$$opt_k(N) = \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^k(I)} I(O(I, n), n) \Big)$$

Then for each k we bound $opt_k(N)$ in terms of $bpol(N)$. In Appendix C we provide proof of each of the following bounds:

Lemma 1 $opt_1(N) \leq bpol_1(N)$.

Lemma 2 *$opt_2(N) \leq bpol(N) + (|\mathcal{T}| \cdot r_{max})$, where $|\mathcal{T}|$ denotes the number of arm types and r_{max} denotes the maximum reward.*

Lemma 3 *$opt_3(N) = O(f(N))$.*

From the above bounds we obtain our overall claim:

Theorem 1 *$bpol(N) \geq opt(N)/2 - O(f(N))$, where $bpol(N)$ and $opt(N)$ denote the total expected reward obtained under BPOL and OPT respectively.*

Proof:

$$\begin{aligned} & opt(N) \\ = \;& opt_1(N) + opt_2(N) + opt_3(N) \\ \leq \;& bpol_1(N) + bpol(N) + \big(|\mathcal{T}| \cdot r_{max}\big) + O(f(N)) \\ \leq \;& 2 \cdot bpol(N) + O(f(N)) \end{aligned}$$

Hence, $bpol(N) \geq opt(N)/2 - O(f(N))$. ■

If we supply MIX (Section 3) as input to our generic BPOL framework, we obtain *BMIX*, a policy for the budgeted unknown-CTR advertisement problem. Due to the way MIX structures and maintains its internal state, it is not necessary to restart a MIX instance when an advertiser's budget is depleted in BMIX, as specified in the generic BPOL framework (the exact steps of BMIX are given in Appendix B).

So far, for modeling purposes, we have assumed the search engine receives an entirely new batch of advertisements each day. In reality, ads may persist over multiple days. With BMIX, we can carry forward an ad's CTR estimate ($\hat{c}_{i,j}$) and display count ($n_{i,j}$) from day to day until an ad is revoked, to avoid having to re-learn CTR's from scratch each day. Of course the daily budgets reset daily, regardless of how long each ad persists. In fact, with a little care we can permit ads to be submitted and revoked at arbitrary times (not just at day boundaries). We describe this extension, as well as how we can incorporate and leverage prior beliefs about CTR's, in Appendix A.

5 Experiments

From our general result of Section 4, we have a theoretical performance guarantee for BMIX. In this section we study BMIX empirically. In particular, we compare it with the greedy policy proposed for the known-CTR advertisement problem (Cells 1-IV in Figure 2). GREEDY displays the C ads targeted for a query phrase that have the highest $\big(\hat{c}_{i,j} \cdot b_{i,j}\big)$ values among the ads whose advertisers have enough remaining budgets; to induce a minimal amount of exploration, for an ad which has never been displayed before, GREEDY treats $\hat{c}_{i,j}$ as ∞ (our policies do this as well). GREEDY is geared exclusively toward *exploitation*. Hence, by comparing GREEDY with our policies, we can gauge the importance of *exploration*.

We also propose and evaluate the following variants of BMIX that we expect to perform well in practice:

1. Varying the Exploration Factor. Internally, BMIX runs instances of MIX to select which ads to display. As mentioned in Section 4, the priority function of MIX consists of an exploration factor $\big(\sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}\big)$ and an exploitation factor ($c_{i,j}$). In [3] it was shown empirically that the following heuristical exploitation factor performs well, despite the absence of a known performance guarantee:

$$\sqrt{\frac{ln\ n_j}{n_{i,j}} \cdot \min\Big\{\frac{1}{4}, V_{i,j}(n_{i,j}, n_j)\Big\}} \quad \text{where} \quad V_{i,j}(n_{i,j}, n_j) = \Big(\hat{c}_{i,j} \cdot (1 - \hat{c}_{i,j})\Big) + \sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}$$

Substituting this expression in place of $\sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}$ in the priority function of BMIX gives us a new (heuristical) policy we call *BMIX-E*.

2. Budget Throttling. It is shown in [9] that in the presence of budget constraints, it is beneficial to display the ads of an advertiser less often as the advertiser's remaining budget decreases. In particular, they propose to multiply bids from advertiser A_i by the following *discount factor*:

$$\phi(d_i') = 1 - e^{-d_i'/d_i}$$

where d_i' is the current remaining budget of advertiser A_i for the day and d_i is its total daily budget. Following this idea we can replace $b_{i,j}$ by $\left(\phi(d_i') \cdot b_{i,j}\right)$ in the priority function of BMIX, yielding a variant we call *BMIX-T*. Policy *BMIX-ET* refers to use of heuristics 1 and 2 together.

5.1 Experiment Setup

We evaluate advertisement policies by conducting simulations over real-world data. Our data set consists of a sample of 85,000 query phrases selected at random from the Yahoo! query log for the date of February 12, 2006. Since we have the frequency counts of these query phrases but not the actual order, we ran the simulations multiple times with random orderings of the query instances and report the average revenue in all our experiment results. The total number of query instances is 2 million. For each query phrase we have the list of advertisers interested in it and the ads submitted by them to Yahoo!. We also have the budget constraints of the advertisers. Roughly 60% of the advertisers in our data set impose daily budget constraints.

In our simulation, when an ad is displayed, we decide whether a click occurs by flipping a coin weighted by the true CTR of the ad. Since true CTRs are not known to us (this is the problem we are trying to solve!), we took the following approach to assign CTRs to ads: from a larger set of Yahoo! ads we selected those ads that have been displayed more than thousand times, and therefore we have highly accurate CTR estimates. We regarded the distribution of these CTR estimates as the true CTR distribution. Then for each ad $a_{i,j}$ in the dataset we sampled a random value from this distribution and assigned it as CTR $c_{i,j}$ of the ad. (Although this method may introduce some skew compared with the (unknown) true distribution, it is the best we could do short of serving live ads just for the purpose of measuring CTRs).

We are now ready to present our results. Due to lack of space we consider a simple setting here where the set of ads is fixed and no prior information about CTR is available. We study the more general setting in Appendix A.

5.2 Exploration/Exploitation Tradeoff

We ran each of the policies for a time horizon of ten days; each policy carries over its CTR estimates from one day to the next. Budget constraints are renewed each day. For now we fix the number of displayed ads (C) to 1. Figure 3 plots the revenue generated by each policy after a given number of days (for confidentiality reasons we have changed the unit of revenue). All policies (including GREEDY) estimate CTRs based on past observations, so as time passes by their estimates become more reliable and their performance improves. Note that the exploration factor of BMIX-E causes it to perform substantially better than that of BMIX. The budget throttling heuristic (BMIX-T and BMIX-ET) did not make much difference in our experiments.

All of our proposed policies perform significantly better than GREEDY, which underscores the importance of balancing exploration and exploitation. GREEDY is geared exclusively toward exploitation, so one might expect that early on it would outperform the other policies. However, that does not happen because GREEDY immediately fixates on ads that are not very profitable (*i.e.*, low $c_{i,j} \cdot b_{i,j}$).

Next we vary the number of ads displayed for each query (C). Figure 4 plots total revenue over ten days on the y-axis, and C on the x-axis. Each policy earns more revenue when more ads are displayed (larger C). Our policies outperform GREEDY consistently across different values of C. In fact, GREEDY must display almost twice as many ads as BMIX-E to generate the same amount of revenue.

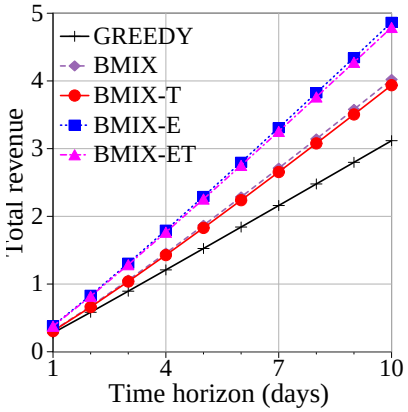

Figure 3: Revenue generated by different advertisement policies (C=1).

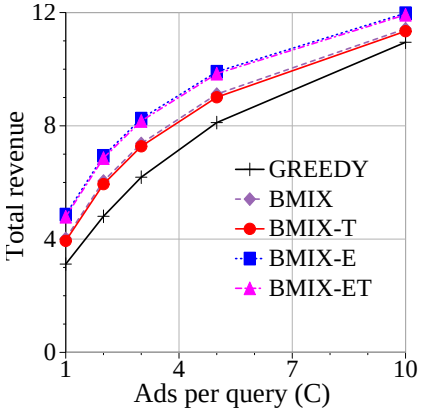

Figure 4: Effect of C (number of ads displayed per query).

6 Summary and Future Work

In this paper we studied how a search engine should select which ads to display in order to maximize revenue, when click-through rates are not initially known. We dealt with the underlying exploration/exploitation tradeoff using multi-armed bandit theory. In the process we contributed to bandit theory by proposing a new variant of the bandit problem that we call budgeted multi-armed multi-bandit problem (BMMP). We proposed a policy for solving BMMP and derived a performance guarantee. Practical extensions of our advertisement policies are given in the extended version of the paper. Extensive experiments over real ad data demonstrate substantial revenue gains compared to a greedy strategy that has no provision for exploration.

Several useful extensions of this problem can be conceived. One such extension would be to exploit similarity in ad attributes while inferring CTRs, as suggested in [8], instead of estimating the CTR of each ad independently. Also, an adversarial formulation of this problem merits study, perhaps leading to general consideration of how to manage exploration versus exploitation in game-theoretic scenarios.

References

[1] N. Abe and A. Nakamura. Learning to Optimally Schedule Internet Banner Advertisements. In *ICML*, 1999.

[2] R. Agrawal. Sample Mean Based Index Policies with O(log n) Regret for the Multi-Armed Bandit Problem. *Advances in Applied Probability*, 27:1054–1078, 1995.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multi-Armed Bandit Problem. *Machine Learning*, 47:235–256, 2002.

[4] D. A. Berry and B. Fristedt. Bandit Problems: Sequential Allocation of Experiments. Chapman and Hall, London, 1985.

[5] G. Casella and R. L. Berger. Statistical Inference. Thomson Learning, 2001.

[6] N. Immorlica, K. Jain, M. Mahdian, and K. Talwar. Click Fraud Resistant Methods for Learning Click-Through Rates. In *WINE*, 2005.

[7] T. Lai and H. Robbins. Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[8] O. Madani and D. Decoste. Contextual Recommender Problems. In *Proceedings of the 1st International Workshop on Utility-based Data Mining*, 2005.

[9] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. AdWords and Generalized On-line Matching. In *FOCS*, 2005.

[10] P. Rusmevichientong and D. Williamson. An Adaptive Algorithm for Selecting Profitable Keywords for Search-Based Advertising Services. In *EC*, 2006.

A Practical Extensions of BMIX

In Section 4 we studied BMIX in a simple setting where the set of ads is fixed and no prior information about CTR is available. We consider the more general setting now.

A.1 Exploiting Prior Information About CTRs

In practice, search engines may have some prior information available about the CTRs of ads even before displaying them and gauging user response. The prior information may come from various sources such as textual relevance of the ad to the query phrase or trustworthiness of the advertiser who submitted the ad. We do not propose any method of deriving the prior information in this paper; instead we focus on studying how the prior information, if it is available, can be used in the advertisement policies and what difference it makes on their performance. For instance, it would be interesting to find out whether our policies perform any better than GREEDY if the prior estimates of CTRs are reasonably correct.

A.1.1 Modeling Prior Information

We use the following model of prior information. Suppose the true CTR of ad $a_{i,j}$ is $c_{i,j}$. We assume that the search engine does not know the CTR value a priori, but has a prior distribution on the CTR. We set the form of prior distribution to a beta distribution [3] $beta_{i,j}(\alpha_{i,j}, \beta_{i,j})$ where $\alpha_{i,j}$ and $\beta_{i,j}$ are its parameters. We denote the mean and the variance of $beta_{i,j}$ by $\hat{\mu}_{i,j}$ and $\hat{\sigma}_{i,j}$.

In our experiments we synthetically generate the prior distributions of ads. While generating these distributions, we vary two parameters: (a) the fraction of ads for which the prior distribution is available, denoted by p, and (b) the accuracy of prior information, denoted by v. To synthesize a prior distribution, we take the following two steps: (a) given the true CTR value $c_{i,j}$ we create a beta distribution with mean $c_{i,j}$ and variance $c_{i,j} \cdot (1 - v)$ and (b) we then sample the mean of prior distribution, $\hat{\mu}_{i,j}$, from the created beta distribution and set the variance, $\hat{\sigma}_{i,j}$, to $\hat{\mu}_{i,j} \cdot (1 - v)$.

To give an intuition of how far the initial CTR estimate $\hat{\mu}_{i,j}$ can be from the actual CTR $c_{i,j}$ for different values of v, we consider an ad of CTR equal to 0.2. When $v = 0.9$, $\hat{\mu}_{i,j}$ is set between 0.1 and 0.3 with 0.58 probability. When $v = 0.95$, this probability increases to 0.68 and when $v = 0.98$, it is almost 0.90.

A.1.2 Exploiting Prior Information

Next we show how we use the prior distributions of CTRs in our advertisement policies. All our policies including GREEDY use CTR estimates ($\hat{c}_{i,j}$'s) in deciding which ads to display. We use the prior distributions to find these CTR estimates. Initially, for each ad $a_{i,j}$ the estimate of its CTR is the mean of its prior distribution $beta_{i,j}(\alpha_{i,j}, \beta_{i,j})$, hence, $\hat{c}_{i,j} = \hat{\mu}_{i,j} = \frac{\alpha_{i,j}}{\alpha_{i,j}+\beta_{i,j}}$.

Once ad $a_{i,j}$ has been displayed for query phrase Q_j, we condition its prior distribution using the click observation of the ad and obtain the posterior distribution of its CTR. In particular, if the prior distribution for ad $a_{i,j}$ is $beta_{i,j}(\alpha_{i,j}, \beta_{i,j})$ and suppose that $s_{i,j}$ denotes the number of times the ad was clicked on when it was displayed for Q_j while $f_{i,j}$ denotes number of times it was not, then the posterior distribution of CTR is simply $beta_{i,j}(\alpha_{i,j} + s_{i,j}, \beta_{i,j} + f_{i,j})$. Given the posterior distribution, the CTR estimate (or the mean) is $\frac{\alpha_{i,j}+s_{i,j}}{\alpha_{i,j}+\beta_{i,j}+s_{i,j}+f_{i,j}}$. We use this CTR estimate in all the advertisement policies (GREEDY, BMIX and its variants).

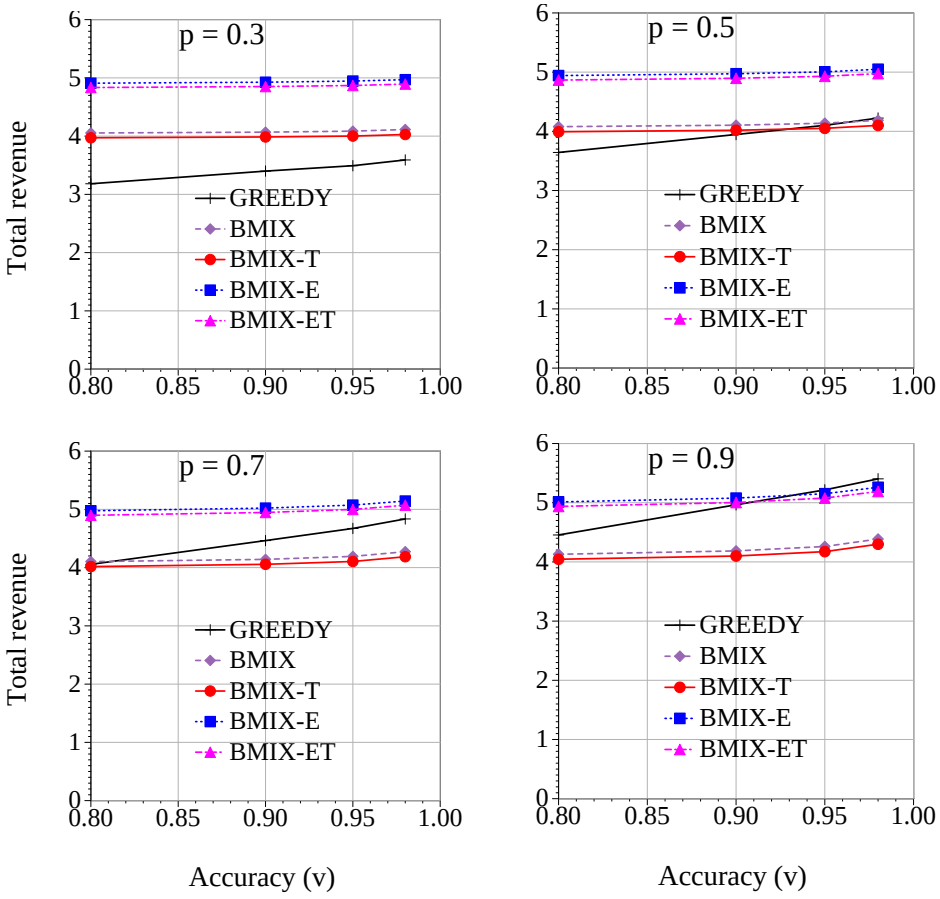

Figure 5: Effect of the prior information.

A.1.3 Performance Comparison

For a given p and v we simulate the advertisement policies for a time horizon of ten days and measure the total revenue generated. The results are shown in Figure 5, with v plotted on the x-axis and the total revenue plotted on the y-axis. The four graphs are for different values of p.

For a given value of p, if we increase v the prior estimates of CTRs ($\hat{\mu}_{i,j}$) get closer to the actual CTRs ($c_{i,j}$), hence, all of the policies perform better. Similarly, if we increase p for a fixed v, the policies get the prior distributions for more ads and they perform better. Note that unlike GREEDY our policies are not affected significantly by the prior distribution of CTRs. GREEDY does not have any provision for exploration, so it relies heavily on the prior distributions. On the other hand, our policies only use the prior distributions to start with (they keep low confidence in the prior distributions due to small $\alpha_{i,j} + \beta_{i,j}$) and once in steady state they largely rely on their own CTR estimates.

Except when the amount (p) and accuracy (v) of prior information is exceptionally high, our policies significantly outperform GREEDY. Furthermore, our policies are never substantially worse than GREEDY.

[3]The event of clicking on an ad is a Bernoulli random variable. It is standard to use a beta distribution for modeling the prior of Bernoulli event [5].

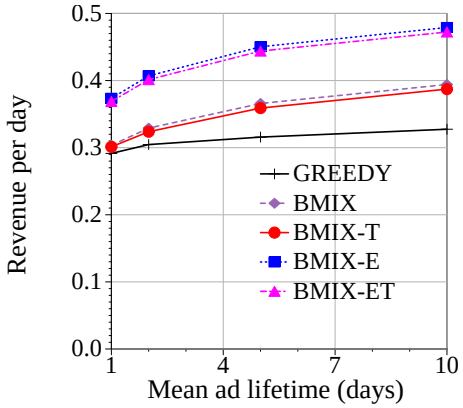

Figure 6: Effect of ad lifetime.

A.2 Allowing Submission/Revocation of Ads at Any Time

We now consider the scenario where advertisers can submit or revoke ads at any time. Observe that BMIX (and its variants) seamlessly extends to this scenario. We make BMIX to look at all the ads that are available at the time a query phrase is being answered, hence any deleted ad is not considered while every newly submitted is.

Next we evaluate our policies empirically in this scenario. We use the following model of submission and revocation of ads: an ad stays with the search engine for a lifetime that is distributed according to a Poisson random variable with the mean set to λ. The ad is revoked once its life is over. When the ad is revoked, we submit a new ad with identical characteristics to the just revoked one. Hence, the rates of submission and revocation of ads are kept the same.

Since the ads are in flux in this experiment, we ran our experiment for a long enough time horizon (100 days) to reach steady state. Figure 6 shows the result, with mean lifetime plotted on the x-axis and the revenue generated per day in the steady state on the y-axis. As expected, as the average lifetime of ads (λ) increases, the performance gap between our policies and GREEDY increases. When ads tend to remain in the system for a long time, the exploration done by our policies pays off the most. Even for a reasonably short lifetime of ads, *e.g.*, one day, our policies still outperform GREEDY.

B Policy for Budgeted Unknown-CTR Advertisement Problem

In Section 4 we proposed our method of deriving a policy for BMMP given as input a conventional multi-armed bandit policy. Below we use that method to derive BMIX, a policy for the budgeted unknown-CTR advertisement problem which is an instance of BMMP. We use MIX (from Section 3) as the input policy.

Policy BMIX :

- *Each time a query for phrase Q_j arrives:*
  1. *For ads whose advertisers have not depleted their budgets yet, compute the priorities as defined in Policy MIX, and display the C ads of highest priority.*
  2. *Update the CTR estimates ($\hat{c}_{i,j}$) of the displayed ads by monitoring user clicks.*

Note that it is not necessary to restart the MIX instance for Q_j when an advertiser's budget is depleted as done in the generic BPOL (Section 4.1). The reason is that MIX maintains

state (*i.e.*, $n_{i,j}$, $\hat{c}_{i,j}$'s) on a per-ad basis, so it can continue from where it left off if some ads are removed from consideration "in-flight".

In Appendix D we show that for MIX, $f(n)$ is $O(ln\ n)$ for any $C \geq 1$. Hence, using our general result of Section 4, we know that the average revenue generated by BMIX is at least $opt(N)/2 - O(ln\ N)$ for any $C \geq 1$ where $opt(N)$ denotes the optimal revenue generated from answering N user queries.

C Performance Bound for BMMP Policies

We prove the lemmas of Section 4 here. First we give some background. Recall that we have defined payoff instance I such that $I(i,n)$ denotes the reward for arm i of bandit instance $S(n)$ for invocation n in payoff instance I. Since $I(i,n)$ takes a particular reward value with a certain probability, say $\mathbb{P}(I(i,n))$, we can get the probability with which payoff instance I arises by multiplying the probabilities of all $I(i,n)$'s, hence $\mathbb{P}(I) = \prod_{n=1}^{N} \prod_{i \in S(n)} \mathbb{P}(I(i,n))$. Let $\mathcal{I}$ denote the space consisting of all payoff instances, then $\sum_{I \in \mathcal{I}} \mathbb{P}(I) = 1$.

The total expected reward obtained under BPOL, $bpol(N)$, is:

$$bpol(N) \quad = \quad \sum_{I \in \mathcal{I}} \big(\mathbb{P}(I) \cdot bpol(I,N) \big)$$

where $bpol(I,N)$ denotes the total reward obtained in payoff instance I. Also,

$$bpol(I,N) \quad = \quad \sum_{n=1}^{N} Z(B(I,n),n,I)$$

where $Z(i,n,I)$ denotes the reward obtained by activating arm i of bandit instance $S(n)$ for invocation n in payoff instance I.

Since BPOL activates the arms of only those types whose budgets have not depleted yet, $Z(B(I,n),n,I) = I(B(I,n),n)$. Hence,

$$\begin{aligned} bpol(N) \quad &= \quad \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n=1}^{N} Z(B(I,n),n,I) \right) \\ &= \quad \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n=1}^{N} I(B(I,n),n) \right) \end{aligned}$$

Similarly,

$$opt(N) = \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n=1}^{N} I(O(I,n),n) \right)$$

Some further notation: let $\mu_{i,B}$ denote the expected reward of arm i of bandit instance B. Let $d_I(T,n)$ denote the remaining budgets of type T at invocation n under BPOL in payoff instance I. As mentioned in Section 4, we classify each invocation n into the following three categories.

- *Category 1:* If $\mu_{B(I,n),S(n)} \geq \mu_{O(I,n),S(n)}$.
- *Category 2:* If $\{\mu_{B(I,n),S(n)} < \mu_{O(I,n),S(n)}\} \ \wedge \{d_I(T,n) < I(O(I,n),n)\}$.
- *Category 3:* If $\{\mu_{B(I,n),S(n)} < \mu_{O(I,n),S(n)}\} \ \wedge \{d_I(T,n) \geq I(O(I,n),n)\}$.

For payoff instance I, let us denote the invocations of category k (1, 2 or 3) by $\mathcal{N}^k(I)$. Let

$$bpol_k(N) = \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^k(I)} I(B(I,n),n) \right)$$

It is easy to see that $bpol(N) = \sum_{k=1}^{3} bpol_k(N)$. Similarly, $opt(N) = \sum_{k=1}^{3} opt_k(N)$ where

$$opt_k(N) = \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^k(I)} I(O(I,n),n) \Big)$$

Lemma 1 *$opt_1(N) \leq bpol_1(N)$.*

Proof: Recall that:

$$bpol_1(N) = \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^1(I)} I(B(I,n),n) \right)$$

For any predicate Π we define $\{\Pi(\mathrm{x})\}$ to be the indicator function of the event $\Pi(\mathrm{x})$; i.e., $\{\Pi(\mathrm{x})\} = 1$ if $\Pi(\mathrm{x})$ is true and $\Pi(\mathrm{x}) = 0$ otherwise. Using the definition of category 1,

$$\begin{aligned} & bpol_1(N) \\ &= \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n=1}^{N} \Big(\{\mu_{B(I,n),S(n)} \geq \mu_{O(I,n),S(n)}\} \cdot I(B(I,n),n) \Big) \right) \\ &= \sum_{n=1}^{N} \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \{\mu_{B(I,n),S(n)} \geq \mu_{O(I,n),S(n)}\} \cdot I(B(I,n),n) \Big) \end{aligned}$$

For a given n we divide payoff instance I into two parts I_1 and I_2 where I_1 consists of $I(i,n')$'s for $n' < n$ and I_2 consist of $I(i,n')$'s for $n' \geq n$. By definition, the arm selected by BPOL (and OPT) at the n^{th} invocation only depends on I_1. Hence, we denote $B(I,n)$ and $O(I,n)$ by $B(I_1,n)$ and $O(I_1,n)$ for the rest of this proof. Clearly, payoff instance space $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2$ where $\mathcal{I}_1$ and $\mathcal{I}_2$ denote the payoff instance spaces for I_1 and I_2 respectively and $\times$ denotes the cross product.

$$\begin{aligned} & bpol_1(N) \\ &= \sum_{n=1}^{N} \sum_{I_1 \in \mathcal{I}_1} \sum_{I_2 \in \mathcal{I}_2} \Big(\mathbb{P}(I_1) \cdot \mathbb{P}(I_2) \cdot \{\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}\} \cdot I_2(B(I_1,n),n) \Big) \\ &= \sum_{n=1}^{N} \sum_{I_1 \in \mathcal{I}_1} \left(\mathbb{P}(I_1) \cdot \{\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}\} \cdot \sum_{I_2 \in \mathcal{I}_2} \Big(\mathbb{P}(I_2) \cdot I_2(B(I_1,n),n) \Big) \right) \\ &= \sum_{n=1}^{N} \sum_{I_1 \in \mathcal{I}_1} \left(\mathbb{P}(I_1) \cdot \{\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}\} \cdot \mu_{B(I_1,n),S(n)} \right) \end{aligned}$$

Similarly,

$$opt_1(N) = \sum_{n=1}^{N} \sum_{I_1 \in \mathcal{I}_1} \Big(\mathbb{P}(I_1) \cdot \{\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}\} \cdot \mu_{O(I_1,n),S(n)} \Big)$$

Since $\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}$ in the terms contributing to the above summations, we get $bpol_1(N) \geq opt_1(N)$.

■

Lemma 2 *$opt_2(N) \leq bpol(N) + (|\mathcal{T}| \cdot r_{max})$ where $|\mathcal{T}|$ denotes the number of arm types and r_{max} denotes the maximum reward.*

Proof: Recall that $\mathcal{N}^2(I)$ denotes the sequence of invocations of category 2 for payoff instance I. Let us denote the set of $O(I,n)$'s for $n \in \mathcal{N}^2(I)$ by $\mathcal{O}^2(I)$, *i.e.*, $\mathcal{O}^2(I) = \{O(I,n) \mid n \in \mathcal{N}^2(I)\}$. Furthermore, let $\mathcal{T}^2(I)$ denote the set of types covering the arms of set $\mathcal{O}^2(I)$. Consider any type T from set $\mathcal{T}^2(I)$. By definition of category 2, we know that the remaining budget of type T drops below r_{max} at some point in BPOL. Therefore, $d_I(T, N+1) < r_{max}$ (here $d_I(T, N+1)$ denotes the remaining budget of type T in BPOL after all N bandit instances of sequence S have been invoked).

Since the total reward given by the arms of a type is the difference of its initial budget $d_I(T,1)$ and the final budget $d_I(T, N+1)$,

$$\begin{aligned}
bpol(I,N) &= \sum_{T \in \mathcal{T}} \Big(d_I(T,1) - d_I(T,N+1) \Big) \\
&\geq \sum_{T \in \mathcal{T}^2(I)} \Big(d_I(T,1) - d_I(T,N+1) \Big) \\
&\geq \sum_{T \in \mathcal{T}^2(I)} \Big(d_I(T,1) - r_{max} \Big) \\
&= \sum_{T \in \mathcal{T}^2(I)} \big(d_I(T,1) \big) - \big(|\mathcal{T}^2(I)| \cdot r_{max} \big) \\
&\geq \sum_{T \in \mathcal{T}^2(I)} \big(d_I(T,1) \big) - \big(|\mathcal{T}| \cdot r_{max} \big)
\end{aligned}$$

By rearranging the terms,

$$\sum_{T \in \mathcal{T}^2(I)} d_I(T,1) \leq bpol(I,N) + \big(|\mathcal{T}| \cdot r_{max} \big)$$

Now we derive a bound for $opt_2(N)$. Recall that:

$$opt_2(N) = \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^2(I)} \big(I(O(I,n),n) \big) \Big)$$

Since we know that the total reward given by the arms of a type can never exceed its initial budget,

$$\begin{aligned}
opt_2(N) &\leq \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{T \in \mathcal{T}^2(I)} d_I(T,1) \Big) \\
&\leq \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \Big(bpol(I,N) + \big(|\mathcal{T}| \cdot r_{max} \big) \Big) \Big) \\
&= \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot bpol(I,N) \Big) + \Big(|\mathcal{T}| \cdot r_{max} \Big) \\
&= bpol(N) + \Big(|\mathcal{T}| \cdot r_{max} \Big)
\end{aligned}$$

Hence, $opt_2(N) \leq bpol(N) + \big(|\mathcal{T}| \cdot r_{max} \big)$. ■

Lemma 3 *$opt_3(N) = O(f(N))$ where $f(n)$ denotes the expected number of mistakes made by POL for any finite n.*

Proof: Recall that:

$$opt_3(N) = \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^3(I)} \big(I(O(I,n),n) \big) \Big)$$

Since $I(i,n) \leq r_{max}$ for all i and n,

$$\begin{aligned} opt_3(N) &\leq \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^3(I)} r_{max} \Big) \\ &= \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^3(I)} \big(r_{max} \cdot \sum_{i=1}^{|\mathcal{B}|} \{S(n) = i\} \big) \Big) \\ &= r_{max} \cdot \sum_{i=1}^{|\mathcal{B}|} C_i(N) \end{aligned}$$

where $C_i(N)$ denotes the expected number of times bandit instance B_i happens to be invoked during the invocations of category 3:

$$C_i(N) = \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^3(I)} \{S(n) = i\} \Big)$$

In Lemma 4 we show that for every $B_i \in \mathcal{B}$, $C_i(N) = O(f(N))$. Hence, $opt_3(N) = O(f(N))$. ■

Lemma 4 *For every bandit instance B_i in $\mathcal{B}$, $C_i(N) = O(f(N))$.*

Proof: Let $\mathcal{S}^i$ denote the sequence of invocations at which bandit instance B_i is invoked in sequence S, *i.e.*, $\mathcal{S}^i = \{n \mid S(n) = i\}$. We analyze BPOL now. Recall that in BPOL as the arms of a bandit instance run out of budget, they are being successively discarded. Let $\mathcal{S}^i_d(I)$ denote the sequence of those invocations at which an arm(s) of B_i is discarded in payoff instance I. We call the sequence of invocations of B_i between two successive invocations of sequence $\mathcal{S}^i_d(I)$ a *batch*. The number of batches is upper bounded by the number of arms of B_i (which is finite). Clearly, within a batch the set of available arms of bandit instance B_i remains fixed and POL operates on them independently and uninterruptedly.

Consider a batch in payoff instance I. Now pick an invocation n of category 3 in the batch when bandit instance B_i is invoked. By definition of category 3, both arms $B(I,n)$ and $O(I,n)$ are available to choose for POL at n. By choosing arm $B(I,n)$ over $O(I,n)$, POL makes a *mistake* of choosing suboptimal arm since $\mu_{B(I,n),S(n)} < \mu_{B(I,n),S(n)}$. Hence, we have shown that in a given batch, each invocation of category 3 is caused by a mistake of POL. Given the performance bound of POL, the expected number of such mistakes in a batch is $f(\textit{batch length})$, hence $O(f(N))$. Since the number of batches is finite, $C_i(N)$ is $O(f(N))$. ■

D Performance Bound for MIX

The optimal policy for the unbudgeted unknown-CTR advertisement problem is to display the C ads of the highest expected reward $(c_{i,j} \cdot b_{i,j})$ for each query phrase. We prove that MIX makes $O(ln\ N)$ mistakes, on expectation, for any $C \geq 1$ where N denotes the number of queries answered. A mistake occurs when an ad of less expected reward $(c_{i,j} \cdot b_{i,j})$ is displayed for a query phrase while keeping an ad of higher expected reward out. Since MIX is adapted from UCB, our proof is largely inherited from [3].

Consider query phrase $Q_j \in \mathcal{Q}$. Let $\mathcal{A}_j$ denote the set of ads for phrase Q_j and let $\mathcal{G}_j$ denote the set of C ads of the highest expected rewards. For simplicity, we assume that each ad has a unique expected reward. Clearly, a mistake occurs when an ad from set $\mathcal{A}_j - \mathcal{G}_j$ is displayed for Q_j. We denote the number of times ad $a_{i,j}$ is displayed by MIX by $m_{i,j}(n_j)$ where n_j denotes the number of times query phrase Q_j has been answered so far.

Theorem 2 *For any ad $a_{i,j} \in \{\mathcal{A}_j - \mathcal{G}_j\}$, $\mathbb{E}(m_{i,j}(N_j)) = O(ln\ N_j)$ where $\mathbb{E}$ denotes the expectation.*

Proof: Recall the priority function of MIX:

$$P_{i,j} = \begin{cases} \left(\hat{c}_{i,j} + \sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}\right) \cdot b_{i,j} & \text{if } n_{i,j} > 0 \\ \infty & \text{otherwise} \end{cases}$$

Here $\hat{c}_{i,j}$ denote the current CTR estimate of ad $a_{i,j}$ based on the past observations, $b_{i,j}$ is its bid value, $n_{i,j}$ denotes the number of times $a_{i,j}$ has been displayed so far for phrase Q_j and n_j denotes the number of times phrase Q_j has been queried so far in the day. We denote the CTR estimated after $n_{i,j}$ display of ads by $\hat{c}_{i,j}(n_{i,j})$. For notation convenience, we denote $\sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}$ in the priority function by $g(n_j, n_{i,j})$.

Some further notation: For any predicate Π we define $\{\Pi(\mathrm{x})\}$ to be the indicator function of the event $\Pi(\mathrm{x})$; i.e., $\{\Pi(\mathrm{x})\} = 1$ if $\Pi(\mathrm{x})$ is true and $\Pi(\mathrm{x}) = 0$ otherwise. For the n_j^{th} occurrence of query phrase Q_j, let $L_j(n_j)$ denote the ad of lowest priority value in G_j and let $\mathcal{U}_j(n_j)$ denote the set of C ads displayed by MIX. Consider $a_{i,j} \in \{\mathcal{A}_j - \mathcal{G}_j\}$, then:

$$\begin{aligned} m_{i,j}(N_j) &= 1 + \sum_{n_j=|\mathcal{A}_j|+1}^{N_j} \{a_{i,j} \in \mathcal{U}_j(n_j)\} \\ &\qquad \{\text{since each ad from } \mathcal{A}_j \text{ is displayed once initially}\} \\ &\leq l + \sum_{n_j=|\mathcal{A}_j|+1}^{N_j} \{a_{i,j} \in \mathcal{U}_j(n_j),\ m_{i,j}(n_j - 1) \geq l\} \\ &\qquad \{\text{where } l \text{ is an arbitrary positive integer}\} \end{aligned}$$

In order for ad $a_{i,j}$ to be displayed on the n_j^{th} occurrence of query phrase Q_j, its priority must be greater than or equal to the priority of $L_j(n_j)$, hence,

$$\begin{aligned} & m_{i,j}(N_j) \\ &\leq l + \sum_{n_j=|\mathcal{A}_j|+1}^{N_j} \sum_{a_{k,j} \in \mathcal{G}_j} \Big\{ \Big(\hat{c}_{i,j}(m_{i,j}(n_j - 1)) + g(n_j - 1, m_{i,j}(n_j - 1))\Big) \cdot b_{i,j} \geq \\ &\qquad \Big(\hat{c}_{k,j}(m_{k,j}(n_j - 1)) + g(n_j - 1, m_{k,j}(n_j - 1))\Big) \cdot b_{k,j},\quad a_{k,j} = L_j(n_j),\quad m_{i,j}(n_j - 1) \geq l \Big\} \\ &\leq l + \sum_{n_j=|\mathcal{A}_j|+1}^{N_j} \sum_{a_{k,j} \in \mathcal{G}_j} \Big\{ \max_{l \leq s_i < n_j} \Big(\hat{c}_{i,j}(s_i) + g(n_j - 1, s_i)\Big) \cdot b_{i,j} \geq \\ &\qquad \min_{0 < s_k < n_j} \Big(\hat{c}_{k,j}(s_k) + g(n_j - 1, s_k)\Big) \cdot b_{k,j},\quad a_{k,j} = L_j(n_j) \Big\} \\ &\qquad \{\text{since } \forall a_{k,j},\ 0 < m_{k,j}(n_j - 1) < n_j\} \\ &\leq l + \sum_{n_j=1}^{N_j} \sum_{a_{k,j} \in \mathcal{G}_j} \sum_{s_k=1}^{n_j} \sum_{s_i=l}^{n_j} \Big\{ \Big(\hat{c}_{i,j}(s_i) + g(n_j, s_i)\Big) \cdot b_{i,j} \geq \Big(\hat{c}_{k,j}(s_k) + g(n_j, s_k)\Big) \cdot b_{k,j}, \\ &\qquad a_{k,j} = L_j(n_j + 1) \Big\} \end{aligned}$$

Now we focus our attention on $\Big(\hat{c}_{i,j}(s_i) + g(n_j, s_i)\Big) \cdot b_{i,j} \geq \Big(\hat{c}_{k,j}(s_k) + g(n_j, s_k)\Big) \cdot b_{k,j}$ where $a_{k,j} \in \mathcal{G}_j$. Let us call it condition Y. Observe the following three terms:

$$\hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k) \tag{1}$$

$$\hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i) \tag{2}$$

$$c_{k,j} \cdot b_{k,j} \quad < \quad c_{i,j} \cdot b_{i,j} + 2 \cdot g(n_j, s_i) \cdot b_{i,j} \tag{3}$$

It is easy to see that if none of these terms are true, then condition Y can not hold true. Hence, we can replace condition Y in the above equation by condition $\{1 \vee 2 \vee 3\}$ since the replacement does not make the RHS any smaller.

$$\begin{aligned} & m_{i,j}(N_j) \\ \leq \quad & l + \sum_{n_j=1}^{N_j} \sum_{a_{k,j} \in \mathcal{G}_j} \sum_{s_k=1}^{n_j} \sum_{s_i=l}^{n_j} \Big(\Big\{ \hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k) \Big\} + \Big\{ \hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i) \Big\} \\ & \qquad + \Big\{ c_{k,j} \cdot b_{k,j} < c_{i,j} \cdot b_{i,j} + 2 \cdot g(n_j, s_i) \cdot b_{i,j} \Big\} \Big) \cdot \{ a_{k,j} = L_j(n_j + 1) \} \end{aligned} \tag{4}$$

We bound the probability of Terms 1 and 2 using Chernoff-Hoeffding bound:

$$\Pr\{\hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k)\} \leq e^{-4 \cdot (ln\ n_j)} = n_j^{-4}$$

$$\Pr\{\hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i)\} \leq e^{-4 \cdot (ln\ n_j)} = n_j^{-4}$$

Recall that we can set l to any positive integer. We set l to $l_o = \lceil (8 \cdot (ln\ N_j) \cdot b_{i,j}^2) / \Delta_{i,j}^2 \rceil$ where $\Delta_{i,j} = \min_{a_{k,j} \in \mathcal{G}_j} \left(c_{k,j} \cdot b_{k,j} - c_{i,j} \cdot b_{i,j} \right)$. For $a_{k,j} \in \mathcal{G}_j$ and $s_i \geq l_o$, Term 3 is false because:

$$\begin{aligned} c_{k,j} \cdot b_{k,j} - c_{i,j} \cdot b_{i,j} - 2 \cdot g(n_j, s_i) \cdot b_{i,j} \quad &= \quad c_{k,j} \cdot b_{i,j} - c_{i,j} \cdot b_{i,j} - 2 \cdot \sqrt{2(ln\ n_j)/s_i} \cdot b_{i,j} \\ &\geq \quad c_{k,j} \cdot b_{k,j} - c_{i,j} \cdot b_{i,j} - 2 \cdot \sqrt{2(ln\ N_j)/l_o} \cdot b_{i,j} \\ &= \quad c_{k,j} \cdot b_{k,j} - c_{i,j} \cdot b_{i,j} - \Delta_{i,j} \\ &\geq \quad 0 \end{aligned}$$

Hence, by taking expectation of Equation 4,

$$\begin{aligned} & \mathbb{E}(m_{i,j}(N_j)) \\ \leq \quad & l + \sum_{n_j=1}^{N_j} \sum_{a_{k,j} \in \mathcal{G}_j} \sum_{s_k=1}^{n_j} \sum_{s_i=l}^{n_j} \Big(\Pr\{\hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k)\} + \Pr\{\hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i)\} \\ & \qquad + \Pr\{c_{k,j} \cdot b_{k,j} < c_{i,j} \cdot b_{i,j} + 2 \cdot g(n_j, s_i) \cdot b_{i,j}\} \Big) \cdot \Pr\{a_{k,j} = L_j(n_j + 1)\} \\ \leq \quad & l_o + \sum_{n_j=1}^{N_j} \sum_{a_{k,j} \in \mathcal{G}_j} \sum_{s_k=1}^{n_j} \sum_{s_i=l_o}^{n_j} \Big(\Pr\{\hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k)\} \\ & \qquad + \Pr\{\hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i)\} \Big) \cdot \Pr\{a_{k,j} = L_j(n_j + 1)\} \\ & \qquad\qquad \{ \text{ by setting } l = l_o \ \} \\ \leq \quad & l_o + \sum_{n_j=1}^{\infty} \sum_{s_k=1}^{n_j} \sum_{s_i=1}^{n_j} \sum_{a_{k,j} \in \mathcal{G}_j} 2 \cdot n_j^{-4} \cdot \Pr\{a_{k,j} = L_j(n_j + 1)\} \\ = \quad & l_o + \sum_{n_j=1}^{\infty} \sum_{s_k=1}^{n_j} \sum_{s_i=1}^{n_j} 2 \cdot n_j^{-4} \\ \leq \quad & \left\lceil \frac{8 \cdot (ln\ N_j) \cdot b_{i,j}^2}{\Delta_{i,j}^2} \right\rceil + \left(1 + \frac{\pi^2}{3}\right) \end{aligned}$$

Hence $\mathbb{E}(m_{i,j}(N_j)) = O(ln\ N_j)$. ■

Given the above result it is clear that the total expected number of mistakes made by MIX for N queries, $\sum_{Q_j \in \mathcal{Q}} \sum_{a_{i,j} \in \mathcal{A}_j - \mathcal{G}_j} \mathbb{E}(m_{i,j}(N_j))$, is $O(ln\ N)$.