Online Learning to Rank in Stochastic Click Models

Mohammad Ghavamzadeh 1  Branislav Kveton 1  Csaba Szepesvári 2  Tomas Tunys 3  Zhong Wen 1  Masrour Zoghi 4

Abstract
Online learning to rank is an important problem in machine learning, information retrieval, recommender systems, and display advertising. Many provably efficient algorithms have been developed for this problem recently, under specific click models. A click model is a model of how users click on a list of documents. Though these results are significant, the proposed algorithms have limited application because they are designed for specific click models, and do not have guarantees beyond them. To overcome this challenge, we propose a novel algorithm, which we call MergeRank, for learning to rank in a class of click models that satisfy mild technical assumptions. This class encompasses two most fundamental click models, the cascade and position-based models. We derive a gap-dependent upper bound on the expected n-step regret of MergeRank and evaluate it on web search queries. We observe that MergeRank performs better than ranked bandits and is more robust than CascadeKL-UCB, an existing algorithm for learning to rank in the cascade model.

1. Introduction
Learning to rank is a core problem in machine learning and information retrieval (Manning et al., 2008). This problem has many practical applications, for instance in web search (Radlinski & Joachims, 2005; Agichtein et al., 2006). In web search, in response to a query by the user, the goal is to present a list of $K$ documents out of $L$ that maximizes the satisfaction of the user. Millions of users interact with popular search engines daily and this vast amount of data can be used to learn better lists of documents over time.

A lot of literature has been dedicated to the problem of online learning to rank. Arguably the most popular approach to learning to rank is ranked bandits (Radlinski et al., 2008; Slivkins et al., 2013). The key idea in ranked bandits is to model each position in the recommended list as a separate bandit problem, and then solve it independently of the other positions. Because the distribution of clicks on a position is not stationary and depends on higher ranked items, each position is typically solved by an adversarial bandit algorithm (Auer et al., 1995). This approach is quite general, but suffers from one obvious and major limitation. Ranked bandits do not generalize across positions, and thus are statistically inefficient in many practical applications (Kveton et al., 2015a; Katariya et al., 2016).

Click models are models of how users click on a list of documents (Chuklin et al., 2015), and many such models have been proposed in the literature (Becker et al., 2007; Richardson et al., 2007; Craswell et al., 2008; Chapelle & Zhang, 2009; Guo et al., 2009a;b). Different click models are motivated by different practical scenarios and make different simplifying assumptions. Several recent papers proposed online learning to rank algorithms for various click models, such as the cascade model (CM) (Kveton et al., 2015a; Combes et al., 2015; Kveton et al., 2015b; Zong et al., 2016; Li et al., 2016), the dependent-click model (DCM) (Katariya et al., 2016), and the position-based model (PBM) (Lagree et al., 2016). In contrast to ranked bandits, these algorithms generalize across different positions by exploiting the structure of the click model, and thus are more statistically efficient in that model.

A major limitation of the existing algorithms for learning to rank in click models is that they may perform poorly in the click models that they are not designed for. This severely limits their application, since the underlying click model is often unknown. Therefore, it is important to develop algorithms that are statistically efficient in multiple click models. We note that this may be possible. In particular, in many click models, the item is clicked only if it is attractive and its position is examined, and these two events are independent. Therefore, it may be possible to estimate the attractiveness of items from clicks, which is necessary for ranking them; while averaging out model-dependent factors, such as examination.
This paper initiates the study of this topic. We make several contributions. First, we propose *stochastic click bandits*, a common framework for learning to rank in a general class of click models, which includes both the CM (Craswell et al., 2008) and the PBM (Richardson et al., 2007); arguably two most fundamental click models. Second, we design a novel algorithm for solving our class of problems, which we call MergeRank. Third, we analyze this algorithm and bound its regret. Finally, we evaluate the algorithm on both CM and PBM queries. We observe that MergeRank is more robust than CascadeKL-UCB (Kveton et al., 2015a), which is an optimal algorithm for learning to rank in the CM, and performs better than ranked bandits with Exp3 (Radlinski et al., 2008).

We denote random variables by boldface letters and define \([n] = \{1, \ldots, n\}\). For any two sets \(A\) and \(B\), we denote by \(A^B\) the set of all vectors whose entries are indexed by \(B\) and take values from \(A\).

2. Background

This section reviews two fundamental click models (Chuklin et al., 2015), which explain the click patterns of users who are presented an ordered list of \(K\) documents out of \(L\). The universe of all documents is represented by ground set \(E = [L]\) and we refer to each document in \(E\) as an item. The user is presented an ordered list of \(K\) items out of \(L\), \(A = (a_1, \ldots, a_K) \in \Pi_K(E)\), where \(\Pi_K(E) \subset E^K\) is the set of all \(K\)-tuples with distinct elements from \(E\), which we call the set of \(K\)-permutations of \(E\). Both reviewed models are parameterized by \(L\) item-dependent attraction probabilities \(\alpha \in [0, 1]^E\), where \(\alpha(e)\) is the probability that item \(e\) is attractive. For simplicity and without loss of generality, we assume that \(\alpha(1) \geq \ldots \geq \alpha(L)\). The items attract the user independently. The two models differ in how the user examines items, which then leads to clicks.

2.1. Cascade Model

In the cascade model (CM) (Craswell et al., 2008), the user scans a list of items \(A = (a_1, \ldots, a_K)\) from the first item \(a_1\) to the last \(a_K\). If item \(a_k\) is attractive, the user clicks on it and does not examine any of the remaining items. If item \(a_k\) is not attractive, the user examines item \(a_{k+1}\). The first item is examined with probability one.

Since the items attract the user independently, the probability that item \(a_k\) is examined in list \(A\) is equal to the probability that none of the first \(k - 1\) items in \(A\) are attractive,

\[
\gamma(A, k) = \prod_{i=1}^{k-1} (1 - \alpha(a_i)) .
\]  

Then the expected number of clicks on list \(A\) is at most one, and is equal to the probability of observing any click,

\[
r(A) = \sum_{k=1}^{K} \gamma(A, k)\alpha(a_k) = 1 - \prod_{k=1}^{K} (1 - \alpha(a_k)) .
\]

This function is maximized by the list of \(K\) most attractive items,

\[
A^* = (1, \ldots, K) ;
\]  

though in the CM any ordering of items \(1, \ldots, K\) is optimal.

2.2. Position-Based Model

In the position-based model (PBM) (Richardson et al., 2007), each position is associated with an examination probability. In particular, in addition to \(L\) item-dependent attraction probabilities, the model is parameterized by \(K\) position-dependent examination probabilities \(\gamma \in [0, 1]^K\), where \(\gamma(k)\) is the probability that position \(k\) is examined.

The items are examined independently of each other. The user clicks on the item if and only if its position is examined and the item at that position is attractive. So, the expected number of clicks on list \(A\) in this model is

\[
r(A) = \sum_{k=1}^{K} \gamma(k)\alpha(a_k) .
\]

In practice, it is often observed that \(\gamma(1) \geq \ldots \geq \gamma(K)\) (Chuklin et al., 2015), and we adopt this assumption in this work. Under this assumption, the above function is maximized by the list of \(K\) most attractive items \(A^*\) in (2), where the \(k\)-th most attractive item is placed at position \(k\). Therefore, the CM and PBM have the same sets of optimal lists.

3. Learning to Rank in Stochastic Click Models

The CM (Section 2.1) and PBM (Section 2.2) are similar in many aspects. First, both models are parameterized by \(L\) item-dependent attraction probabilities. The attractiveness of any item is independent of the attractiveness of any other item. Second, the item is clicked only if it is attractive and its position is examined. The events of examination and attractiveness are independent, given the recommended list. Therefore, the probability of clicking on an item is the product of its attraction probability and the examination probability of its position in the list. Finally, the optimal list in both models is the list of \(K\) most attractive items \(A^*\) in (2), where the \(k\)-th most attractive is placed at position \(k\).

This suggests that it may be possible to design a single learning algorithm that learns the optimal solution in both
models from observed clicks, without knowing the underlying model. We propose such algorithm in Section 4. Before we discuss it, we propose a bandit model that allows us to learn in both the CM and PBM.

3.1. Stochastic Click Bandit

We refer to our learning problem as a stochastic click bandit (SCB). An instance of this problem is defined by a tuple \((K, L, P_w, P_c)\), where \(K > 0\) is the number of positions, \(L \geq K\) is the number of items, \(P_w\) is a distribution over \([0, 1]^E\), and \(P_c\) is a distribution over \([0, 1]^{\Pi_K(E) \times K}\).

Let \((w_t, X_t)_{t=1}^n\) be i.i.d. random variables drawn from distribution \(P_w \otimes P_c\). In particular, at any time \(t \in [n]\), \(w_t\) and \(X_t\) are independent. We refer to \(w_t\) as the attraction weights at time \(t\) and to \(w_t(e)\) as the attraction of item \(e\). We refer to \(X_t\) as the examination matrix at time \(t\) and to \(X_t(A, k)\) as the examination indicator of position \(k\) in list \(A \in \Pi_K(E)\).

The learning agent interacts with our problem as follows. At time \(t\), the agent chooses a list \(A_t = (a_1^t, \ldots, a_K^t) \in \Pi_K(E)\), which can depend arbitrarily on the past observations of the agent, and observes clicks. The clicks are a function of \(w_t\), \(X_t\), and \(A_t\). Let \(c_t \in \{0, 1\}^K\) be the indicator vector of clicks at all positions at time \(t\). Then

\[
c_t(k) = X_t(A_t, k) w_t(a_k^t)
\]

for any \(k \in [K]\), the item at position \(k\) is clicked only if both \(w_t(a_k^t)\) and \(X_t(A_t, k)\) are one.

The goal of the learning agent is to maximize the number of clicks. Therefore, the number of clicks at time \(t\) is the reward \(r_t\) of the agent at time \(t\), and is given by

\[
r_t = r(A_t, w_t, X_t);
\]

where \(r : \Pi_K(E) \times [0, 1]^E \times [0, 1]^{\Pi_K(E) \times K} \to [0, K]\) is a reward function, which we define as

\[
r(A, w, X) = \sum_{k=1}^K X(A, k) w(a_k)
\]

for any \(A = (a_1, \ldots, a_K) \in \Pi_K(E)\), \(w \in [0, 1]^E\), and \(X \in [0, 1]^{\Pi_K(E) \times K}\).

We make the same independence assumptions as in Section 2. First, we assume that items attract the user independently.

**Assumption 1.** For any \(w \in \{0, 1\}^E\),

\[
P(w_t = w) = \prod_{e \in E} \text{Ber}(w(e); \alpha(e)).
\]

We denote by \(\text{Ber}(\cdot; \theta)\) the probability mass function of the Bernoulli distribution with mean \(\theta \in [0, 1]\), and define it as \(\text{Ber}(y; \theta) = \theta^y (1 - \theta)^{1-y}\) for \(y \in \{0, 1\}\).

Second, we assume that in any list \(A\), the attraction of any item is independent of its examination.

**Assumption 2.** For any list \(A \in \Pi_K(E)\) and position \(k\),

\[
\mathbb{E}[c_t(k) | A_t = A] = \mathbb{E}[X_t(A, k) w_t(a_k)] = \gamma(A, k) \alpha(a_k),
\]

where \(\gamma \in [0, 1]^{|\Pi_K(E)\times K}|\) and \(\gamma(A, k) = P(X_t(A, k) = 1)\) is the probability that position \(k\) is examined in list \(A\).

Note that we do not make any specific independence assumption on the entries of \(X_t\). They can be heavily correlated. For instance, when the first \(k - 1\) items in lists \(A\) and \(A'\) are identical, it may happen that \(X_t(A, k) = X_t(A', k)\) for any \(X_t\).

From our independence assumptions and the definition of rewards, it follows that the expected reward of list \(A\) is

\[
\mathbb{E}[r(A, w_t, X_t)] = \sum_{k=1}^K \mathbb{E}[X_t(A, k) w_t(a_k)]
\]

\[
= \sum_{k=1}^K \gamma(A, k) \alpha(a_k)
\]

\[
= r(A, \alpha, \gamma).
\]

We evaluate the performance of a learning agent by its expected cumulative regret

\[
R(n) = \mathbb{E}\left[ \sum_{t=1}^n R(A_t, w_t, X_t) \right],
\]

where \(R(A_t, w_t, X_t) = r(A^*, w_t, X_t) - r(A_t, w_t, X_t)\) is the instantaneous regret of the agent at time \(t\) and

\[
A^* = \arg \max_{A \in \Pi_K(E)} r(A, \alpha, \gamma)
\]

is the optimal list of items, the list that maximizes the expected reward. To simplify exposition, we assume that the optimal solution, as a set, is unique.

3.2. Position Bandit

The learning variant of the PBM can be formulated in our setting. It is an instance of our problem when

\[
\forall A, A' \in \Pi_K(E) : X_t(A, k) = X_t(A', k)
\]

at any position \(k \in [K]\). Under this assumption, the probability of clicking on item \(a_k^t\) at time \(t\) is

\[
\mathbb{E}[c_t(k) | A_t = \gamma(k) \alpha(a_k^t)],
\]

where \(\gamma(k)\) is defined in Section 2.2, while the expected reward of list \(A_t\) at time \(t\) is

\[
\mathbb{E}[r_t | A_t] = \sum_{k=1}^K \gamma(k) \alpha(a_k^t).
\]
3.3. Cascading Bandit

The learning variant of the CM can also be formulated in our setting. It is an instance of our problem when

\[ X_i(A, k) = \prod_{i=1}^{k-1} (1 - w_i(a_i)) \]

for any list \( A \in \Pi_K(E) \) and position \( k \in [K] \). Under this assumption, the probability of clicking on item \( a_k \) at time \( t \) is

\[ \mathbb{E}[c_k(k) | A_i] = \left[ \prod_{i=1}^{k-1} (1 - \alpha(a_i)) \right] \alpha(a_k), \]

while the expected reward of list \( A_i \) at time \( t \) is

\[ \mathbb{E}[r | A_i] = \sum_{k=1}^{K} \left[ \prod_{i=1}^{k-1} (1 - \alpha(a_i)) \right] \alpha(a_k). \]

3.4. Additional Assumptions

The above assumptions are not sufficient to guarantee that the optimal list is \( A^* \) in (2). To guarantee this, and that \( A^* \) is learnable, we make four additional assumptions.

Assumption 3 (Order-independent examination). For any \( A = (a_1, \ldots, a_K) \in \Pi_K(E) \) and \( A' = (a'_1, \ldots, a'_K) \in \Pi_K(E) \) and any position \( k \in [K] \) such that \( a_k = a'_k \) and \( \{a_1, \ldots, a_{k-1}\} = \{a'_1, \ldots, a'_{k-1}\} \), \( X_i(A, k) = X_i(A', k) \).

This assumption says that \( X_i(A, k) \) only depends on the identities of \( a_1, \ldots, a_{k-1} \). It does not depend on their order or \( a_k \). Both the CM and PBM satisfy this assumption, which can be validated from (3) and (4).

Assumption 4 (Decreasing examination). For any list \( A \in \Pi_K(E) \) and positions \( 1 \leq i \leq j \leq K \), \( \gamma(A, i) \geq \gamma(A, j) \).

The above assumption is natural and says that in any list \( A \), lower ranked items are less likely to be examined than the higher ranked ones. Assumption 4 is satisfied by both the CM and PBM.

Assumption 5 (Examination scaling). For any list \( A \in \Pi_K(E) \) and positions \( 1 \leq i \leq j \leq K \), let \( \alpha(a_i) \leq \alpha(a_j) \) and \( A' \in \Pi_K(E) \) be the same list as \( A \) except that \( a_i \) and \( a_j \) are exchanged. Then \( \gamma(A, j) \geq \gamma(A', j) \).

The above assumption is natural and says that in any list \( A \), a more attractive item is more likely to be examined after a less attractive item than vice versa. Note that both the CM and PBM satisfy this assumption. In the CM, the inequality follows from the definition of the examination in (1). In the PBM, \( \gamma(A, j) = \gamma(A', j) \).

Assumption 6 (Most attractive items are optimal). For any list \( A \in \Pi_K(E) \) and positions \( 1 \leq i \leq j \leq K \), let \( \alpha(a_i) \leq \alpha(a_j) \) and \( A' \in \Pi_K(E) \) be the same list as \( A \) except that \( a_i \) and \( a_j \) are exchanged. Then \( r(A, \alpha, \gamma) \leq r(A', \alpha, \gamma) \).

The above assumption implies that \( A^* \) in (2) is optimal. This follows from the linearity of the expected reward in \( \alpha \), Assumption 3, and Assumption 6.

Algorithm 1 MergeRank

1: // Initialization
2: for all \( b = 1, \ldots, 2K \) do
3: for all \( \ell = 1, \ldots, n \) do
4: for all \( e \in E \) do
5: \( C_{b, \ell}(e) \leftarrow 0 \), \( n_{b, \ell}(e) \leftarrow 0 \)
6: \( B \leftarrow \{1\} \), \( b_{\text{max}} \leftarrow 1 \)
7: \( I_1 \leftarrow (1, K), E_{b_0} \leftarrow E, \ell_1 \leftarrow 0 \)
8: for all \( t = 1, \ldots, n \) do
9: for all \( b \in B \) do
10: for all \( e \in E \) do
11: DisplayItems(b, t)
12: for all \( b \in B \) do
13: UpdateBatch(b, t)

4. Algorithm MergeRank

The design of MergeRank, whose pseudocode is in Algorithm 1, builds on two key ideas. First, we randomize the placement of items to avoid biases due to the click model. Second, we divide and conquer: divide batches of items into batches of more and less attractive items, and then proceed recursively. The result is an ordered list of items, where the \( k \)-th most attractive item is placed at position \( k \).

Our basic data structure is a batch. MergeRank creates batches dynamically and we index them by integers \( b > 0 \). Each batch \( b \) is associated with a range of positions \( I_b \subseteq [K] \), where \( I_b(1) \) is the highest position in batch \( b \), \( I_b(2) \) is the lowest position in batch \( b \), and \( \text{len}(b) = I_b(2) - I_b(1) + 1 \) is the number of positions in batch \( b \). Each batch \( b \) is also associated with the initial set of items \( E_{b_0} \subseteq E \). The items are explored in stages, which are indexed by integers \( \ell > 0 \). The remaining items in stage \( \ell \) of batch \( b \) are \( E_{b, \ell} \subseteq E_{b_0} \). At the end of the stage, some items can be eliminated and the batch can also split into two new batches.

Each batch \( b \) is explored in stages that quadruple in length. In particular, each item \( e \in E_{b, \ell} \) in stage \( \ell \) is explored \( n_\ell \) times, where \( n_\ell = \left\lceil 16 \Delta_{\ell}^{-2} \log n \right\rceil \) and \( \Delta_{\ell} = 2^{-\ell} \). The current stage of batch \( b \) is \( \ell_b \). The exploration is conducted in method DisplayItems, whose pseudocode is in Algorithm 3; which randomly displays items. For each item \( e \in E_{b, \ell} \), the method counts the number of clicks on that item, \( C_{b, \ell}(e) \); and the number of times that this item is observed, \( n_{b, \ell}(e) \). The method is designed such that any item \( e \in E_{b, \ell} \) is displayed in any list from \( E_{b, \ell} \) with that item with the same probability. This is critical to avoid biases due to the click model. At the end of stage \( \ell \), the probability of clicking on item \( e \in E_{b, \ell} \) is estimated as

\[ \hat{C}_{b, \ell}(e) = C_{b, \ell}(e)/n_\ell. \]

Because of the randomized placement strategy, this esti-
Algorithm 2 UpdateBatch
1: Input: batch index \( b \), time \( t \)
2: \( \ell \leftarrow \ell_b \)
3: // End-of-stage elimination
4: \( \ell \leftarrow \ell_b \)
5: if \( \min_{e \in \mathcal{E}_b} n_{b,\ell}(e) = n_\ell \) then
6: Let \( \mathbf{U}_{b,\ell}(e) \) be the UCB on \( \mathbf{C}_{b,\ell}(e) \)
7: Let \( \mathbf{L}_{b,\ell}(e) \) be the LCB on \( \mathbf{C}_{b,\ell}(e) \)
8: Let \( \mathcal{E}^+_{\ell} \subseteq \mathcal{E}_{b,\ell} \) be the items with \( k \) largest LCBs
9: Let \( \mathcal{E}^-_{\ell} = \mathbf{L}_{b,\ell} \setminus \mathcal{E}^+_{\ell} \)
10: Let \( \ell^+_{k} \leftarrow \min_{e \in \mathcal{E}^+_{\ell}} \mathbf{L}_{b,\ell}(e) \)
11: // Find a split at the position with the highest index
12: if \( s = 0 \) or \( \mathcal{E}^-_{\ell} \) then
13: \( \ell_b \leftarrow \ell_b + 1 \)
14: for all \( k = 1, \ldots, \text{len}(b) - 1 \) do
15: if \( \ell^+_{k} > \max_{e \in \mathcal{E}^+_{b}} \mathbf{U}_{b,\ell}(e) \) then
16: \( s \leftarrow k \)
17: if \( (s = 0) \) and \( (\mathcal{E}^-_{\ell} > \text{len}(b)) \) then
18: // Next elimination stage
19: \( \mathbf{E}_{b,\ell+1} \leftarrow \mathbf{E}_{b,\ell} \setminus \{ e \in \mathcal{E}_{b,\ell} : \mathbf{L}_{\text{len}(b)} > \mathbf{U}_{b,\ell}(e) \} \)
20: \( \ell_b \leftarrow \ell_b + 1 \)
21: else if \( s > 0 \) then
22: // Split
23: \( \mathcal{B} \leftarrow \mathcal{B} \cup \{ \mathbf{b}_{\text{max}} + 1, \mathbf{b}_{\text{max}} + 2 \} \setminus \{ b \} \)
24: \( \mathbf{I}_{b_{\text{max}} + 1} \leftarrow (\mathbf{I}_{b}(1), \mathbf{I}_{b}(1) + s - 1) \)
25: \( \mathbf{E}_{b_{\text{max}} + 1, 0} \leftarrow \mathcal{E}^+_{s} \)
26: \( \ell_{b_{\text{max}} + 1} \leftarrow 0 \)
27: \( \mathbf{I}_{b_{\text{max}} + 2} \leftarrow (\mathbf{I}_{b}(1) + s, \mathbf{I}_{b}(2)) \)
28: \( \mathbf{E}_{b_{\text{max}} + 2, 0} \leftarrow \mathcal{E}^-_{s} \)
29: \( \ell_{b_{\text{max}} + 2} \leftarrow 0 \)
30: \( \mathbf{b}_{\text{max}} \leftarrow b_{\text{max}} + 2 \)

mate can be viewed as the scaled attraction probability of item \( e \), \( \alpha(e) \), such that the scaling factor is “similar” for all items \( e \in \mathcal{E}_{b,\ell} \) (Section 5.2). This permits elimination based on the UCBS and LCBS of \( \mathbf{C}_{b,\ell}(e) \).

At the end of stage \( \ell \) of batch \( b \), any item \( e \in \mathcal{E}_{b,\ell} \) is observed \( n_\ell \) times, \( n_{b,\ell}(e) = n_\ell \). Then we update batch \( b \) in method UpdateBatch, whose pseudocode is in Algorithm 2. This method works as follows. First, we compute KL-UCB (Garivier & Cappé, 2011) upper and lower confidence bounds on \( \mathbf{C}_{b,\ell}(e) \) (lines 6–7) as

\[
\mathbf{U}_{b,\ell}(e) = \arg\max_{q \in [\mathbf{C}_{b,\ell}(e), 1]} \left\{ n_\ell d(\mathbf{C}_{b,\ell}(e), q) \leq \delta_n \right\},
\]

\[
\mathbf{L}_{b,\ell}(e) = \arg\min_{q \in [0, \mathbf{C}_{b,\ell}(e)]} \left\{ n_\ell d(\mathbf{C}_{b,\ell}(e), q) \leq \delta_n \right\},
\]

for any \( e \in \mathcal{E}_{b,\ell} \), where \( d(p, q) \) is the Kullback-Leibler divergence between Bernoulli random variables with means \( p \) and \( q \), and \( \delta_n = \log n + 3 \log \log n \). Then we test whether \( \mathcal{E}_{b,\ell} \) can be safely divided into some \( s \) more attractive and \( |\mathcal{E}_{b,\ell}| - s \) less attractive items (lines 8–16). If the items can be divided, we split batch \( b \) into two batches (lines 22–30). The first batch contains \( s \) items and is defined on positions \( \mathbf{I}_b(1), \ldots, \mathbf{I}_b(s) \). The second batch contains \( |\mathcal{E}_{b,\ell}| - s \) items and is defined on positions \( \mathbf{I}_b(s+1), \ldots, \mathbf{I}_b(2) \). The stage indices of the new batches are initialized to 0. If multiple splits are possible, we choose the largest value of \( s \). If the batch does not split, we still eliminate items that cannot be at position \( \mathbf{I}_b(2) \) or higher with a high probability (lines 18–20).

The set of active batches is tracked in \( \mathcal{B} \), and we explore and update them in parallel. The highest index of the latest added batch is \( \mathbf{b}_{\text{max}} \). Note that \( \mathbf{b}_{\text{max}} \leq 2K \), because each pair of new batches is the result of a split at a unique position \( k \in [K] \). MergeRank is initialized with a single batch over all positions and items (lines 6–7). By the design of splits in UpdateBatch (lines 22–30), the following invariants hold. First, at any time \( t \), all batches in \( \mathcal{B} \) form a partition of \( [K] \). Second, each batch contains at least as many items as is the number of positions in that batch. Finally, in any batch \( b \) such that \( \mathbf{I}_b(2) < K \), the number of items is equal to the number of positions, \( |\mathcal{E}_{b,\ell}| = \text{len}(b) \).

5. Analysis

Fix batch \( b \), positions \( \mathbf{I}_b \), stage \( \ell \), and items \( \mathcal{E}_{b,\ell} \). Then our estimate of the probability of clicking on item \( e \in \mathcal{E}_{b,\ell} \) in (5) is

\[
\hat{\mathbf{C}}_{b,\ell}(e) = \frac{1}{n_\ell} \sum_{t \in T} \sum_{k = \mathbf{I}_b(1)}^{\mathbf{I}_b(2)} \mathbf{c}_t(k) 1 \{ a_k^t = e \},
\]
where $\mathcal{T}$ is the set of time steps where item $e$ is displayed in stage $\ell$ of batch $b$. Although the above quantity is well defined, it is not clear if observations $c_\ell(k)I\{a^k_b = e\}$ are i.i.d. in time, and hence if we can easily argue that $\mathcal{C}_{b,\ell}(e)$ concentrates at

$$\mathcal{C}_{b,\ell}(e) = \mathbb{E}\left[\mathcal{C}_{b,\ell}(e)\right] \tag{7}$$

as the number of observations $n_\ell$ increases. Moreover, even if $\mathcal{C}_{b,\ell}(e)$ concentrates, it is unclear if the elimination based on $\mathcal{C}_{b,\ell}(e)$, which is only a proxy for $\alpha(e)$, is sound. In this work, we show that $\mathcal{C}_{b,\ell}(e)$ is a scaled attraction probability that satisfies

$$\mu^* = \frac{\mathcal{C}_{b,\ell}(e^*)}{\alpha(e^*)} \geq \frac{\mathcal{C}_{b,\ell}(e)}{\alpha(e)} = \mu \tag{8}$$

for any items $e, e^* \in E_{b,\ell}$ such that $\alpha(e^*) \geq \alpha(e)$. This is a sufficient condition for sound elimination, and we apply it in Lemma 3 in Appendix.

In the rest of this section, we argue that the above properties are satisfied by any click model that satisfies all assumptions in Section 3. Then we present our upper bound.

### 5.1. I.I.D. Observations

Fix batch $b$, positions $I_b$, stage $\ell$, and items $E_{b,\ell}$. For any item $e \in E_{b,\ell}$ and position $k \in I_b$, it may seem that observations

$$\{X_t(A_t,k)w_t(e)I\{a^k_b = e\}\}_{t \in \mathcal{T}} \tag{9}$$

are not i.i.d. in time because MergeRank explores batches in parallel, and hence the first $I_b(1) - 1$ entries of $A_b$ can change arbitrarily in time. However, by Assumption 3, we have that $X_t(A_t,k)$ is independent of the order of the first $I_b(1) - 1$ items for any $A \in \Pi_K(E)$ and $k \in I_b$. Since all other quantities are random in time, (9) is indeed an i.i.d. sample.

### 5.2. Correct Examination Scaling

Fix batch $b$, positions $I_b$, stage $\ell$, and items $E_{b,\ell}$. By the definition of $\mathcal{C}_{b,\ell}(e)$ in (7) and the fact that any item $e \in E_{b,\ell}$ is displayed in any list from $E_{b,\ell}$ with that item with the same probability, we have that

$$\mathcal{C}_{b,\ell}(e) = \frac{\alpha(e)}{|A_e|} \sum_{A \in A_e} \sum_{k=1}^{\text{len}(b)} \gamma(A + k)I\{a_k = e\} ,$$

where $A_e = \{A \in \Pi_{\text{len}(b)}(E_{b,\ell}) : e \in A\}$ is the set of lists of length $\text{len}(b)$ over $E_{b,\ell}$ with item $e$. $A$ is a vector whose entries are the items at the first $I_b(1) - 1$ positions in stage $\ell$ of batch $b$, and $A + A$ is the concatenation of $A$ and $A$.

Now we argue that (8) holds for any $e, e^* \in E_{b,\ell}$ such that $\alpha(e^*) \geq \alpha(e)$. Before we proceed, note that for any list $A \in A_e$, there exists one and only one list in $A_e'$ that differs from $A$ only in that items $e$ and $e^*$ are exchanged. We denote this list by $A' \in A_e'$. We analyze three cases. First, suppose that list $A$ does not contain item $e^*$. Then by Assumption 3, the examination probabilities of $e$ in $A$ and $e^*$ in $A'$ are the same. Second, suppose that item $e^*$ is placed before item $e$ in $A$. Then by Assumption 5, the examination probability of $e$ in $A$ is not higher than that of $e^*$ in $A'$. Third, suppose that item $e^*$ is placed after item $e$ in $A$. Then by Assumption 3, the examination probabilities of $e$ in $A$ and $e^*$ in $A'$ are the same, because they do not depend on lower ranked items. Finally, note that $|A_e| = A_e'$. It follows that (8) holds.

### 5.3. Regret Bound

For simplicity of exposition, let $\alpha(1) > \ldots > \alpha(L) > 0$. We also define $\alpha_{\max} = \alpha(1)$, and $\gamma'(k) = \gamma(A^*,k)$ for any $k \in [K]$. Our upper bound on the $n$-step regret of MergeRank is presented below.

**Theorem 1.** For any stochastic click bandit in Section 3.1 that satisfies Assumptions 1 to 6, the expected $n$-step regret of MergeRank is bounded as

$$R(n) \leq \frac{128K^3L}{(1 - \alpha_{\max})\Delta_{\min}} \log n + 2KL(6e + 2K)$$

for any $n \geq 5$, where

$$\Delta_{\min} = \min_{k \in [K]} \{\alpha(k) - \alpha(k + 1)\} . \tag{10}$$

**Proof.** The key idea is to bound the expected $n$-step regret in any batch (Lemma 6 in Appendix). Since the number of batches is at most $2K$, the regret of MergeRank is at most $2K$ times larger than that of in any batch.

The regret due to any batch is bounded as follows. Let $b$ be over positions $I_b$, and $\Delta_{\max} = \alpha(s) - \alpha(s + 1)$ be the maximum gap in batch $b$, where

$$s = \arg\max_{e \in \{I_b(1), \ldots, I_b(2) - 1\}} [\alpha(e) - \alpha(e + 1)] .$$

We analyze two cases. If the expected per-step regret of item $e$ in batch $b$ is $O(K\Delta_{\max})$, its regret is dominated by the time that the batch splits, and we bound this time in Lemma 5 in Appendix. Otherwise, the item is likely to be eliminated before the split, and we bound this time in Lemma 4 in Appendix. Finally, we take the maximum of these two upper bounds.

Our upper bound in Theorem 1 is logarithmic in the number of steps $n$, linear in the number of items $L$, and polynomial in the number of positions $K$. To the best of our
knowledge, this is the first gap-dependent upper bound on the regret of a learning algorithm that has sublinear regret in both the CM and PBM. The gap in (10) characterizes the hardness of sorting $K + 1$ most attractive items, which is sufficient for solving our problem. In practice, no item is attractive with probability one, or even close to it. Therefore, the dependence on $1/(1 - \alpha_{\max})$ is not critical.

The cubic dependence on $K$ is likely suboptimal. In particular, one $K$ is because $\text{MergeRank}$ has separate click counters $C_{b,e}(t)$ in each batch $b$. We believe that $\text{MergeRank}$ would be sound even if the batches shared the counters, and this would reduce the regret by a factor of $K$. Then, by a standard reduction to a gap-free bound, the regret of $\text{MergeRank}$ would be $O(K \sqrt{Ln \log n})$ and comparable to that of $\text{RankedExp3}$ (Radlinski et al., 2008). Note that ranked bandits do not have a gap-dependent regret bound in either the CM or PBM.

KL-UCB confidence intervals in $\text{MergeRank}$ are necessary. Without them, our upper bound would be $O(K^4L \log n)$ and depend on the reciprocal of the examination probabilities of lower ranked positions, which can be exponentially small in the CM. The elimination step in lines 18–20 of $\text{UpdateBatch}$ is also necessary. Without it, the $n$-step regret of $\text{MergeRank}$ would be $O(L^2 \log n)$.

6. Experiments

We evaluate $\text{MergeRank}$ on the Yandex dataset (Yandex), a search log of 35M search sessions. Each session includes at least one search query, the resulting list of displayed documents at positions 1 to 10, and the clicks on those documents. We experiment with 60 most frequent queries from our dataset, and learn their CMs and PBM using PyClick (Chuklin et al., 2015). PyClick is an open-source library of click models for web search. In each query, our problem is to rerank $L = 10$ most attractive items on the first $K = 5$ positions to maximize the expected number of clicks on those first $K$ items. This resembles a real-world setting, where the learning algorithm would only be allowed to rerank highly attractive items, and not allowed to explore aggressively unattractive items (Zoghi et al., 2016).

We compare $\text{MergeRank}$ to two baselines, CascadeKL-UCB (Kveton et al., 2015a) and $\text{RankedExp3}$ (Radlinski et al., 2008). CascadeKL-UCB is the optimal algorithm for learning to rank in the cascade model, which matches a lower bound under reasonable assumptions. $\text{RankedExp3}$ is a variant of ranked bandits where the base bandit algorithm is Exp3 (Auer et al., 1995). This is arguably the most popular algorithm for learning to rank that does not make any independence assumptions.

Because of our setting, many solutions in our queries are near optimal. Therefore, we decided to evaluate the performance of all algorithms by their expected per-step regret in up to step $n = 10M$. If a solution is near-optimal and

Figure 1. The expected per-step regret of $\text{MergeRank}$ (red), $\text{CascadeKL-UCB}$ (blue), and $\text{RankedExp3}$ (gray) on three problems. The results are averaged over 10 runs.

Figure 2. The comparison of $\text{MergeRank}$ (red), $\text{CascadeKL-UCB}$ (blue), and $\text{RankedExp3}$ (gray) in the CM and PBM. The top plots show the expected per-step regret as a function of time $n$, averaged over all 60 queries and 10 runs per query. The bottom plots show the distribution of the per-step regret at $n = 10M$. 
Online learning to rank in click models (Craswell et al., 2008; Chuklin et al., 2015) was recently studied in multiple papers (Kveton et al., 2015a; Combes et al., 2015; Kveton et al., 2015b; Katariya et al., 2016; Zong et al., 2016; Li et al., 2016; Lagree et al., 2016). These papers assume that recommended items attract the user independently. When the click model is known, the attraction probability of the item can be estimated from clicks and the dynamics of the model. This results in more statistically efficient learning algorithms than those in ranked bandits (Kveton et al., 2015a; Katariya et al., 2016). The problem is that all above algorithms are designed for a specific click model, and do not have guarantees beyond them.

Click models (Chuklin et al., 2015) explain the click patterns of users who are presented a list of documents, and many such models have been proposed (Becker et al., 2007; Richardson et al., 2007; Craswell et al., 2008; Chapelle & Zhang, 2009; Guo et al., 2009a;b). Two most fundamental models are the cascade (Craswell et al., 2008) and position-based (Richardson et al., 2007) models. These models have been traditionally studied separately. In this work, we bring them together and show that learning-to-rank problems in both models can be solved by the same algorithm.

Our problem can be also viewed as an instance of partial monitoring, where the attraction indicators of items are unobserved. General algorithms for partial monitoring (Agrawal et al., 1989; Bartok et al., 2012; Bartok & Szepesvari, 2012; Bartok et al., 2014) are unsuitable for our setting because their complexity is polynomial in the number of actions, which is exponential in $K$.

8. Conclusions

We propose stochastic click bandits. Stochastic click bandits are a framework for online learning to rank in a class of click models that encompasses two most fundamental click models, the cascade and position-based models. We propose a computationally and sample efficient algorithm for solving our problems, which we call $\text{MergeRank}$; and derive a gap-dependent upper bound on its $n$-step regret. We evaluate the algorithm on web search queries. The algorithm outperforms ranked bandits (Radlinski et al., 2008), a popular approach to online learning to rank, and is more robust than CascadeKL-UCB (Kveton et al., 2015a), an existing algorithm for the cascade model.

The goal of this work is to design an online algorithm for learning to rank whose regret is provably bounded in multiple click models. As such, many of our design decisions are conservative (Section 5.3) and we plan to relax them in our future work. One such decision is that $\text{MergeRank}$ resets its click estimators between batches, which results in an order of $K$ slower learning.

7. Related Work

The first work on online learning to rank are ranked bandits (Radlinski et al., 2008; Slivkins et al., 2013). The key idea in ranked bandits is to model each position in the recommended list as a separate bandit problem, which is then solved by some base bandit algorithm. The base algorithm is typically adversarial (Auer et al., 1995) because the distribution of clicks beyond the first position depends on higher ranked items, and thus is not stationary. Generally speaking, the algorithms for ranked bandits learn $(1 - 1/e)$ approximate solutions and their $n$-step regret is $O(\sqrt{n})$. We compare $\text{MergeRank}$ to ranked bandits in Section 6.

Online learning to rank in stochastic click models (Craswell et al., 2008; Chuklin et al., 2015) was recently studied in multiple papers (Kveton et al., 2015a; Combes et al., 2015; Kveton et al., 2015b; Katariya et al., 2016; Zong et al., 2016; Li et al., 2016; Lagree et al., 2016). These papers assume that recommended items attract the user independently. When the click model is known, the attraction probability of the item can be estimated from clicks and the dynamics of the model. This results in more statistically efficient learning algorithms than those in ranked bandits (Kveton et al., 2015a; Katariya et al., 2016). The problem is that all above algorithms are designed for a specific click model, and do not have guarantees beyond them.

Click models (Chuklin et al., 2015) explain the click patterns of users who are presented a list of documents, and many such models have been proposed (Becker et al., 2007; Richardson et al., 2007; Craswell et al., 2008; Chapelle & Zhang, 2009; Guo et al., 2009a;b). Two most fundamental models are the cascade (Craswell et al., 2008) and position-based (Richardson et al., 2007) models. These models have been traditionally studied separately. In this work, we bring them together and show that learning-to-rank problems in both models can be solved by the same algorithm.

Our problem can be also viewed as an instance of partial monitoring, where the attraction indicators of items are unobserved. General algorithms for partial monitoring (Agrawal et al., 1989; Bartok et al., 2012; Bartok & Szepesvari, 2012; Bartok et al., 2014) are unsuitable for our setting because their complexity is polynomial in the number of actions, which is exponential in $K$.

8. Conclusions

We propose stochastic click bandits. Stochastic click bandits are a framework for online learning to rank in a class of click models that encompasses two most fundamental click models, the cascade and position-based models. We propose a computationally and sample efficient algorithm for solving our problems, which we call $\text{MergeRank}$; and derive a gap-dependent upper bound on its $n$-step regret. We evaluate the algorithm on web search queries. The algorithm outperforms ranked bandits (Radlinski et al., 2008), a popular approach to online learning to rank, and is more robust than CascadeKL-UCB (Kveton et al., 2015a), an existing algorithm for the cascade model.

The goal of this work is to design an online algorithm for learning to rank whose regret is provably bounded in multiple click models. As such, many of our design decisions are conservative (Section 5.3) and we plan to relax them in our future work. One such decision is that $\text{MergeRank}$ resets its click estimators between batches, which results in an order of $K$ slower learning.

7. Related Work

The first work on online learning to rank are ranked bandits (Radlinski et al., 2008; Slivkins et al., 2013). The key idea in ranked bandits is to model each position in the recommended list as a separate bandit problem, which is then solved by some base bandit algorithm. The base algorithm is typically adversarial (Auer et al., 1995) because the distribution of clicks beyond the first position depends on higher ranked items, and thus is not stationary. Generally speaking, the algorithms for ranked bandits learn $(1 - 1/e)$ approximate solutions and their $n$-step regret is $O(\sqrt{n})$. We compare $\text{MergeRank}$ to ranked bandits in Section 6.
References

Agichtein, Eugene, Brill, Eric, and Dumais, Susan. Improving web search ranking by incorporating user behavior information. In Proceedings of the 29th Annual International ACM SIGIR Conference, pp. 19–26, 2006.

Agrawal, Rajeev, Teneketzis, Demosthenis, and Anantharam, Venkatachalam. Asymptotically efficient adaptive allocation schemes for controlled i.i.d. processes: Finite parameter space. IEEE Transactions on Automatic Control, 34(3):258–267, 1989.

Auer, Peter, Cesa-Bianchi, Nicolo, Freund, Yoav, and Schapire, Robert. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 322–331, 1995.

Bartok, Gabor and Szepesvari, Csaba. Partial monitoring with side information. In Proceedings of the 23rd International Conference on Algorithmic Learning Theory, pp. 305–319, 2012.

Bartok, Gabor, Zolghadr, Navid, and Szepesvari, Csaba. An adaptive algorithm for finite stochastic partial monitoring. In Proceedings of the 29th International Conference on Machine Learning, 2012.

Bartok, Gabor, Foster, Dean, Pal, David, Rakhlin, Alexander, and Szepesvari, Csaba. Partial monitoring - classification, regret bounds, and algorithms. Mathematics of Operations Research, 39(4):967–997, 2014.

Becker, Hila, Meek, Christopher, and Chickering, David Maxwell. Modeling contextual factors of click rates. In Proceedings of the 22nd AAAI Conference on Artificial Intelligence, pp. 1310–1315, 2007.

Chapelle, Olivier and Zhang, Ya. A dynamic Bayesian network click model for web search ranking. In Proceedings of the 18th International Conference on World Wide Web, pp. 1–10, 2009.

Chuklin, Aleksandr, Markov, Ilya, and de Rijke, Maarten. Click Models for Web Search. Morgan & Claypool Publishers, 2015.

Combes, Richard, Magureanu, Stefan, Proutiere, Alexandre, and Laroche, Cyrille. Learning to rank: Regret lower bounds and efficient algorithms. In Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2015.

Craswell, Nick, Zoeter, Onno, Taylor, Michael, and Ramsey, Bill. An experimental comparison of click position-bias models. In Proceedings of the 1st ACM International Conference on Web Search and Data Mining, pp. 87–94, 2008.

Garivier, Aurelien and Cappe, Olivier. The KL-UCB algorithm for bounded stochastic bandits and beyond. In Proceedings of the 24th Annual Conference on Learning Theory, pp. 359–376, 2011.

Guo, Fan, Liu, Chao, Kannan, Anitha, Minka, Tom, Taylor, Michael, Wang, Yi Min, and Faloutsos, Christos. Click chain model in web search. In Proceedings of the 18th International Conference on World Wide Web, pp. 11–20, 2009a.

Guo, Fan, Liu, Chao, and Wang, Yi Min. Efficient multiple-click models in web search. In Proceedings of the 2nd ACM International Conference on Web Search and Data Mining, pp. 124–131, 2009b.

Katariya, Sumeet, Kveton, Branislav, Szepesvari, Csaba, and Wen, Zheng. DCM bandits: Learning to rank with multiple clicks. In Proceedings of the 33rd International Conference on Machine Learning, pp. 1215–1224, 2016.

Kveton, Branislav, Szepesvari, Csaba, Wen, Zheng, and Ashkan, Azin. Cascading bandits: Learning to rank in the cascade model. In Proceedings of the 32nd International Conference on Machine Learning, 2015a.

Kveton, Branislav, Wen, Zheng, Ashkan, Azin, and Szepesvari, Csaba. Combinatorial cascading bandits. In Advances in Neural Information Processing Systems 28, pp. 1450–1458, 2015b.

Lagrange, Paul, Vernade, Claire, and Cappe, Olivier. Multiple-play bandits in the position-based model. In Advances in Neural Information Processing Systems 29, pp. 1597–1605, 2016.

Li, Shuai, Wang, Baoxiang, Zhang, Shengyu, and Chen, Wei. Contextual combinatorial cascading bandits. In Proceedings of the 33rd International Conference on Machine Learning, pp. 1245–1253, 2016.

Manning, Christopher, Raghavan, Prabhakar, and Schutze, Hinrich. Introduction to Information Retrieval. Cambridge University Press, 2008.

Radlinski, Filip and Joachims, Thorsten. Query chains: Learning to rank from implicit feedback. In Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 239–248, 2005.

Radlinski, Filip, Kleinberg, Robert, and Joachims, Thorsten. Learning diverse rankings with multi-armed bandits. In Proceedings of the 25th International Conference on Machine Learning, pp. 784–791, 2008.
Richardson, Matthew, Dominowska, Ewa, and Ragno, Robert. Predicting clicks: Estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 521–530, 2007.

Slivkins, Aleksandrs, Radlinski, Filip, and Gollapudi, Sreenivas. Ranked bandits in metric spaces: Learning diverse rankings over large document collections. *Journal of Machine Learning Research*, 14(1):399–436, 2013.

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

Zoghi, Masrour, Tunys, Tomas, Li, Lihong, Jose, Damien, Chen, Junyan, Chin, Chun Ming, and de Rijke, Maarten. Click-based hot fixes for underperforming torso queries. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 195–204, 2016.

Zong, Shi, Ni, Hao, Sung, Kenny, Ke, Nan Rosemary, Wen, Zheng, and Kveton, Branislav. Cascading bandits for large-scale recommendation problems. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence*, 2016.


A. Proof of Theorem 1

Let $R_{b,\ell}$ be the stochastic regret associated with batch $b$ in stage $\ell$. Then the expected $n$-step regret of MergeRank can be decomposed as

$$R(n) \leq \mathbb{E} \left[ \sum_{b=1}^{2K} \sum_{\ell=0}^{n-1} R_{b,\ell} \right]$$

because the maximum number of batches is $2K$. Let

$$E_{b,\ell} = \{ \text{Event 1: } \forall e \in E_{b,\ell} : \bar{C}_{b,\ell}(i) \in [L_{b,\ell}(e), U_{b,\ell}(e)] \},$$

Event 2: $\forall I_b \in [K]^2$, $e \in E_{b,\ell}$, $e^* \in E_{b,\ell} \cap [K]$ s.t. $\Delta = \alpha(e^*) - \alpha(e) > 0$ :

$$n_\ell \geq \frac{16K}{\gamma^*(I_b(1))(1 - \alpha_{\max})} \log n \implies \bar{C}_{b,\ell}(e) \leq \bar{C}_{b,\ell}(e) \frac{\alpha(e) + \Delta/4}{\alpha(e)};$$

Event 3: $\forall I_b \in [K]^2$, $e \in E_{b,\ell}$, $e^* \in E_{b,\ell} \cap [K]$ s.t. $\Delta = \alpha(e^*) - \alpha(e) > 0$ :

$$n_\ell \geq \frac{16K}{\gamma^*(I_b(1))(1 - \alpha_{\max})} \Delta^2 \log n \implies \bar{C}_{b,\ell}(e^*) \geq \bar{C}_{b,\ell}(e^*) \frac{\alpha(e^*) - \Delta/4}{\alpha(e^*)}$$

be “good events” associated with batch $b$ and stage $\ell$, where $\bar{C}_{b,\ell}(e)$ in (7) is the probability of clicking on item $e$ in stage $\ell$ of batch $b$. Let $\bar{E}_{b,\ell}$ be the complement of event $E_{b,\ell}$. Let $\mathcal{E}$ be the event that all events $E_{b,\ell}$ happen; and $\bar{\mathcal{E}}$ be the complement of $\mathcal{E}$, the event that at least one event $E_{b,\ell}$ does not happen. Then the expected $n$-step regret can be bounded from above as

$$R(n) \leq \mathbb{E} \left[ \sum_{b=1}^{2K} \sum_{\ell=0}^{n-1} R_{b,\ell} \mathbb{1}\{\mathcal{E}\} \right] + nP(\bar{\mathcal{E}}) \leq \sum_{b=1}^{2K} \mathbb{E} \left[ \sum_{\ell=0}^{n-1} R_{b,\ell} \mathbb{1}\{\mathcal{E}\} \right] + 2KL(6e + 2K),$$

where the second inequality is from Lemma 1. Now we apply Lemma 6 to each batch $b$ and get that

$$\sum_{b=1}^{2K} \mathbb{E} \left[ \sum_{\ell=0}^{n-1} R_{b,\ell} \mathbb{1}\{\mathcal{E}\} \right] \leq \frac{128K^3L}{(1 - \alpha_{\max})\Delta_{\min}} \log n.$$

This concludes our proof.

B. Technical Lemmas

Lemma 1. Let $\bar{\mathcal{E}}$ be defined as in the proof of Theorem 1. Then for any $n \geq 5$,

$$P(\bar{\mathcal{E}}) \leq \frac{2KL(6e + 2K)}{n}.$$

Proof. By the union bound,

$$P(\bar{\mathcal{E}}) \leq \sum_{b=1}^{2K} \sum_{\ell=0}^{n-1} P(\bar{E}_{b,\ell}).$$

Now we bound the probability of each event in $\bar{E}_{b,\ell}$ and then sum them up.

Event 1
The probability that event 1 in $\mathcal{E}^t_\ell$ does not happen is bounded as follows. For any $e \in E_{b,\ell}$,

$$P(\hat{C}_{b,\ell}(e) \notin [L_{b,\ell}(e), U_{b,\ell}(e)]) \leq P(\hat{C}_{b,\ell}(e) < L_{b,\ell}(e)) + P(\hat{C}_{b,\ell}(e) > U_{b,\ell}(e))$$

$$\leq 2e \left[ \log(n \log^3 n) \log n \right]$$

$$\leq 2e \left[ \frac{\log^2 n + \log(\log^3 n) \log n}{n \log^3 n} \right]$$

$$\leq \frac{2e}{n \log n} \left[ \frac{2 \log^2 n}{n \log n} \right]$$

$$\leq \frac{6e}{n \log n} ,$$

where the second inequality is from Theorem 10 of (Garivier & Cappe, 2011), the third inequality is from $n \geq n_\ell$, the fourth inequality is from $\log(\log^3 n) \leq \log n$ for $n \geq 5$, and the last inequality is from $[2 \log^2 n] \leq 3 \log^2 n$ for $n \geq 3$.

By the union bound,

$$P(\exists e \in E_{b,\ell} \text{ s.t. } \hat{C}_{b,\ell}(e) \notin [L_{b,\ell}(e), U_{b,\ell}(e)]) \leq \frac{6eL}{n \log n}$$

for any $E_{b,\ell}$. Finally, we take the expectation over $E_{b,\ell}$ and have that the probability that event 1 in $\mathcal{E}^t_\ell$ does not happen at the end of stage $\ell$ of batch $b$ is bounded as above.

**Event 2**

The probability that event 2 in $\mathcal{E}^t_\ell$ does not happen is bounded as follows. Fix $I_b$ and $E_{b,\ell}$, and let $k = I_b(1)$. If the event does not happen for items $e$ and $e^*$, then it must be true that

$$n_\ell \geq \frac{16K}{\gamma^*(k)(1 - p_{\text{max}}) \Delta^2} \log n, \quad \hat{C}_{b,\ell}(e) > \frac{\hat{C}_{b,\ell}(e)}{\alpha(e)} \frac{\alpha(e) + \Delta/4}{}. \quad \hat{C}_{b,\ell}(e) > \frac{\hat{C}_{b,\ell}(e)}{\alpha(e)} \frac{\alpha(e) + \Delta/4}{.}$$

From Hoeffding’s inequality, we have that

$$P\left( \frac{\hat{C}_{b,\ell}(e)}{\alpha(e)} \frac{\alpha(e) + \Delta/4}{.} \right) \leq \exp\left[ -n_\ell d\left(\frac{\hat{C}_{b,\ell}(e)}{\alpha(e)} \frac{\alpha(e) + \Delta/4}{.}, \hat{C}_{b,\ell}(e)\right) \right].$$

From Lemma 7, $\hat{C}_{b,\ell}(e)/\alpha(e) \geq \gamma^*(k)/K$ (Lemma 2), and Pinsker’s inequality, we have that

$$\exp\left[ -n_\ell d\left(\frac{\hat{C}_{b,\ell}(e)}{\alpha(e)} \frac{\alpha(e) + \Delta/4}{.}, \hat{C}_{b,\ell}(e)\right) \right] \leq \exp\left[ -n_\ell \frac{\hat{C}_{b,\ell}(e)}{\alpha(e)} \frac{\alpha(e) + \Delta}{.} \right] \leq \exp\left[ -n_\ell \frac{\gamma^*(k)(1 - p_{\text{max}}) \Delta^2}{8K} \right].$$

Finally, from our assumption on $n_\ell$, we conclude that

$$\exp\left[ -n_\ell \frac{\gamma^*(k)(1 - p_{\text{max}}) \Delta^2}{8K} \right] \leq \exp[-2 \log n] = \frac{1}{n^2} .$$

Now we chain all inequalities and observe that event 2 in $\mathcal{E}^t_\ell$ does not happen for any fixed $I_b$ and $E_{b,\ell}$ with probability of at most $KL/n^2$. Finally, we take the expectation over $I_b$ and $E_{b,\ell}$; and have that the probability that event 2 in $E_{b,\ell}$ does not happen at the end of stage $\ell$ of batch $b$ is at most $KL/n^2$.

**Event 3**

This bound is analogous to that of event 2.

**Total probability**
Proof. From the definition of \( U \), let \( U = \{ s \in S : \exists \alpha > 0 \} \). Then this contradicts to (11), and therefore it must be true that \( L = \alpha(e) - \alpha(e) > 0 \), let \( k = I_b(1) \) be the highest position in batch \( b \) and \( m \) be the first stage where

\[
\hat{\Delta}_m < \sqrt{\frac{\gamma^*(k)}{K}(1 - \alpha_{\max})\Delta}.
\]

Then \( U_{b,m}(e) < L_{b,m}(e^*) \).

Proof. The proof follows from two observations. First, by Assumptions 1 to 6 in Section 3, position \( k \) is least examined when the items at positions \( 1, \ldots, k - 1 \) are \( \{1, \ldots, k - 1\} \). In this case, the examination probability of this position is \( \gamma^*(k) \). Now note that item \( e \) is placed at position \( k \) with probability of at least \( 1/K \). ■

Lemma 3. Let event \( E \) happen. For any batch \( b \), positions \( I_b, \), set \( E_{b,\ell} \), and item \( e \in E_{b,\ell} \),

\[
\gamma^*(k) \leq \frac{C_{b,\ell}(e)}{\alpha(e)}
\]

where \( k = I_b(1) \) is the highest position in batch \( b \).

Proof. From the definition of \( n_m \), and our assumption on \( \hat{\Delta}_m \),

\[
n_m \geq \frac{16}{\Delta^2} \log n > \frac{16K}{\gamma^*(k)(1 - \alpha_{\max})\Delta^2} \log n .
\]

Let \( \mu = \frac{C_{b,m}(e)}{\alpha(e)} \) and suppose that \( U_{b,m}(e) \geq \mu[\alpha(e) + \Delta/2] \) happens. Then from this assumption, the definition of \( U_{b,m}(e) \), and event 2 in \( E_{b,m} \),

\[
d(\mathcal{C}_{b,m}(e), U_{b,m}(e)) \geq d^+(\mathcal{C}_{b,m}(e), \mu[\alpha(e) + \Delta/2]) \geq d(\mu[\alpha(e) + \Delta/4], \mu[\alpha(e) + \Delta/2]),
\]

where \( d^+(p, q) = d(p, q)1_{\{p < q\}}. \) From Lemma 7, \( \mu \geq \gamma^*(k)/K \), and Pinsker’s inequality, we further have that

\[
d(\mu[\alpha(e) + \Delta/4], \mu[\alpha(e) + \Delta/2]) \geq \mu(1 - \alpha_{\max})d(\alpha(e) + \Delta/4, \alpha(e) + \Delta/2) \geq \frac{\gamma^*(k)(1 - \alpha_{\max})\Delta^2}{8K}.
\]

From the definition of \( U_{b,m}(e) \) and above inequalities,

\[
n_m = \frac{2\log n}{d(\mathcal{C}_{b,m}(e), U_{b,m}(e))} \leq \frac{16K \log n}{\gamma^*(k)(1 - \alpha_{\max})\Delta^2}.
\]

This contradicts to (11), and therefore it must be true that \( U_{b,m}(e) < \mu[\alpha(e) + \Delta/2] \).

Now let \( \mu^* = \mathcal{C}_{b,m}(e^*)/\alpha(e^*) \) and suppose that \( L_{b,m}(e^*) \leq \mu^*[\alpha(e^*) - \Delta/2] \) happens. Then from this assumption, the definition of \( L_{b,m}(e^*) \), and event 3 in \( E_{b,m} \),

\[
d(\mathcal{C}_{b,m}(e^*), L_{b,m}(e^*)) \geq d^-(\mathcal{C}_{b,m}(e^*), \mu^*[\alpha(e^*) - \Delta/2]) \geq d(\mu^*[\alpha(e^*) - \Delta/4], \mu^*[\alpha(e^*) - \Delta/2]),
\]
Proof.
Let \( \Delta = \Delta(1) \) happen. Then the expected per-step regret of any item \( e \) is bounded as
\[
E \{ R_{b,m}(e) \} \leq \alpha(m) \Delta = \Delta(1) \Delta.
\]
This concludes our proof. ■

Lemma 4. Let event \( E \) happen. For any batch \( b \), positions \( I_b \) where \( I_b(2) = K \), set \( E_{b,m} \), and item \( e \in E_{b,m} \) such that \( e > K \), let \( k = I_b(1) \) be the highest position in batch \( b \) and \( m \) be the first stage where
\[
\Delta_m < \sqrt{\frac{\gamma^*(k)}{K} (1 - \alpha_{\text{max}}) \Delta}
\]
for \( \Delta = \alpha(K) - \alpha(e) \). Then item \( e \) is eliminated by the end of stage \( m \).

Proof. Let \( E^+ = \{ k, \ldots, K \} \). Now note that \( \alpha(e^*) - \alpha(\epsilon) \geq \Delta \) for any \( e^* \in E^+ \). By Lemma 3, \( L_{b,m}(e^*) > U_{b,m}(e) \) for any \( e^* \in E^+ \); and thus item \( e \) is eliminated by the end of stage \( m \). ■

Lemma 5. Let event \( E \) happen. For any batch \( b \), positions \( I_b \), and set \( E_{b,m} \), let \( k = I_b(1) \) be the highest position in batch \( b \) and \( m \) be the first stage where
\[
\Delta_m < \sqrt{\frac{\gamma^*(k)}{K} (1 - \alpha_{\text{max}}) \Delta_{\text{max}}}
\]
for \( \Delta_{\text{max}} = \alpha(s) - \alpha(s + 1) \), where \( s = \arg \max_{\epsilon \in \{I_b(1), \ldots, I_b(2) - 1\}} \alpha(e) - \alpha(\epsilon + 1) \). Then batch \( b \) is split by the end of stage \( m \).

Proof. Let \( E^+ = \{ k, \ldots, s \} \) and \( E^- = E_{b,m} \setminus E^+ \). Now note that \( \alpha(e^*) - \alpha(\epsilon) \geq \Delta_{\text{max}} \) for any \( (e^*, e) \in E^+ \times E^- \). By Lemma 3, \( L_{b,m}(e^*) > U_{b,m}(e) \) for any \( (e^*, e) \in E^+ \times E^- \); and thus batch \( b \) is split by the end of stage \( m \). ■

Lemma 6. Let event \( E \) happen. Then the expected \( n \)-step regret of any batch \( b \) is bounded as
\[
\mathbb{E} \left[ \sum_{t=0}^{n-1} R_{b,t} \right] \leq \frac{64K^2 |E_{b,0}|}{(1 - \alpha_{\text{max}}) \Delta_{\text{max}}} \log n.
\]

Proof. Let \( k = I_b(1) \) be the highest position in batch \( b \). Choose any item \( e \in E_{b,0} \) and let \( \Delta = \alpha(k) - \alpha(e) \).

First, we show that the expected per-step regret of any item \( e \) is bounded by \( \gamma^*(k) \Delta \) when event \( E \) happens. Since event \( E \) happens, all eliminations up to any stage \( \ell \) of batch \( b \) are correct. Therefore, the items at positions \( 1, \ldots, k - 1 \) are
{1, \ldots, k - 1}; and position $k$ is examined with probability $\gamma^*(k)$. This is the highest examination probability in batch $b$ (Assumption 4). Finally, our upper follows from the observation that the expected reward is linear in $\alpha$ and that $\alpha(k)$ is the most attractive item in batch $b$.

Our analysis has two parts. First, suppose that $\Delta \leq 2K\Delta_{\max}$, where $\Delta_{\max}$ is defined in Lemma 5. By Lemma 5, batch $b$ splits when the number of steps in a stage is at least

$$\frac{16K}{\gamma^*(k)(1 - \alpha_{\max})\Delta_{\max}^2} \log n,$$

and therefore the maximum regret due to item $e$ in the last stage before the split is

$$\frac{16K\gamma^*(k)\Delta}{\gamma^*(k)(1 - \alpha_{\max})\Delta_{\max}^2} \log n \leq \frac{32K^2\Delta_{\max}}{(1 - \alpha_{\max})\Delta_{\max}^2} \log n = \frac{32K^2}{(1 - \alpha_{\max})\Delta_{\max}} \log n.$$

Now suppose that $\Delta > 2K\Delta_{\max}$. Then this item is easy to distinguish from item $K$,

$$\Delta \alpha(K) - \alpha(e) \leq 2.$$

By Lemma 4, the maximum regret due to item $e$ before it is eliminated is

$$\frac{16K\gamma^*(k)\Delta}{\gamma^*(k)(1 - \alpha_{\max})(\alpha(K) - \alpha(e))^2} \log n \leq \frac{64K}{(1 - \alpha_{\max})\Delta} \log n \leq \frac{32}{(1 - \alpha_{\max})\Delta_{\max}} \log n,$$

where the last inequality is from our assumption that $\Delta > 2K\Delta_{\max}$.

Because the number of steps between consecutive stages quadruples, and we reset all estimators at the beginning of each stage, the maximum expected regret of item $e$ before this item is eliminated in batch $b$, or batch $b$ is split, is at most twice of that in the last stage, and hence

$$\mathbb{E}\left[\sum_{\ell=0}^{n-1} R_{b,\ell}\right] \leq \frac{64K^2|E_{b,0}|}{(1 - \alpha_{\max})\Delta_{\max}} \log n.$$

This concludes our proof. ■

**Lemma 7.** Let

$$d(p, q) = p \log \frac{p}{q} + (1 - p) \log \frac{1 - p}{1 - q}$$

be the KL divergence between Bernoulli random variables with means $p \in [0, 1]$ and $q \in [0, 1]$. Let $c \in [0, 1]$. Then

$$c(1 - \max \{p, q\})d(p, q) \leq d(cp, cq) \leq cd(p, q).$$

(12)

**Proof.** The proof is based on differentiation. The first two derivatives of $d(cp, cq)$ with respect to $q$ are

$$\frac{\partial}{\partial q} d(cp, cq) = \frac{c(q - p)}{q(1 - cq)},$$

$$\frac{\partial^2}{\partial q^2} d(cp, cq) = \frac{c^2(q - p)^2 + cp(1 - cp)}{q^2(1 - cq)^2};$$

and the first two derivatives of $cd(p, q)$ with respect to $q$ are

$$\frac{\partial}{\partial q} [cd(p, q)] = \frac{c(q - p)}{q(1 - q)},$$

$$\frac{\partial^2}{\partial q^2} [cd(p, q)] = \frac{c(q - p)^2 + cp(1 - p)}{q^2(1 - q)^2}.$$
The second derivatives show that both $d(cp, cq)$ and $cd(p, q)$ are convex in $q$ for any $p$. The minima are at $q = p$.

We fix $p$ and $c$, and prove (12) for any $q$. The upper bound is derived as follows. Since

$$d(cp, cx) = cd(p, x) = 0$$

when $x = p$, the upper bound holds if $cd(p, x)$ increases faster than $d(cp, cx)$ for any $p < x \leq q$, and that $cd(p, x)$ decreases faster than $d(cp, cx)$ for any $q \leq x < p$. This follows from the definitions of $\frac{\partial}{\partial x}d(cp, cx)$ and $\frac{\partial}{\partial x}[cd(p, x)]$. In particular, both derivatives have the same sign for any $x$, and $1/(1 - cx) \leq 1/(1 - x)$ for $x \in [\min\{p, q\}, \max\{p, q\}]$.

The lower bound is derived as follows. Note that the ratio of $\frac{\partial}{\partial x}[cd(p, x)]$ and $\frac{\partial}{\partial x}d(cp, cx)$ is bounded from above as

$$\frac{\frac{\partial}{\partial x}[cd(p, x)]}{\frac{\partial}{\partial x}d(cp, cx)} = \frac{1 - cx}{1 - x} \leq \frac{1}{1 - x} \leq \frac{1}{1 - \max\{p, q\}}$$

for any $x \in [\min\{p, q\}, \max\{p, q\}]$. Therefore, we get a lower bound on $d(cp, cx)$ when we multiply $cd(p, x)$ by $1 - \max\{p, q\}$. $\blacksquare$