MergeRUCB: A Method for Large-Scale Online Ranker Evaluation

Masrour Zoghi Shimon Whiteson Maarten de Rijke
University of Amsterdam, Amsterdam, The Netherlands
{m.zoghi, s.a.whiteson, derijke}@uva.nl

ABSTRACT

A key challenge in information retrieval is that of *on-line ranker evaluation*: determining which one of a finite set of rankers performs the best in expectation on the basis of user clicks on presented document lists. When the presented lists are constructed using *interleaved comparison methods*, which interleave lists proposed by two different candidate rankers, then the problem of minimizing the total *regret* accumulated while evaluating the rankers can be formalized as a *K-armed dueling bandit problem*. In the setting of web search, the number of rankers under consideration may be large. Scaling effectively in the presence of so many rankers is a key challenge not adequately addressed by existing algorithms.

We propose a new method, which we call *mergeRUCB*, that uses "localized" comparisons to provide the first provably scalable K-armed dueling bandit algorithm. Empirical comparisons on several large learning to rank datasets show that mergeRUCB can substantially outperform the state of the art K-armed dueling bandit algorithms when many rankers must be compared. Moreover, we provide theoretical guarantees demonstrating the soundness of our algorithm.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Evaluation; implicit feedback; on-line learning

1. INTRODUCTION

An important challenge in information retrieval is that of *ranker evaluation*: determining which of a finite set of rankers performs best in expectation. In *off-line* ranker evaluation, which goes back to the early Cranfield experiments [9], rankers are assessed based on a fixed set of queries and documents manually judged by human assessors. Unfortunately, obtaining such judgments is expensive and error prone. Furthermore, because the assessors typically did not formulate the queries on which they judge documents, their assessments may not accurately reflect how well those documents meet the needs of real users.

These difficulties can be addressed by *on-line* ranker evaluation, in which rankers are assessed using click feedback from actual users. A common approach is to use *interleaved comparison methods* [7, 8, 13, 15, 18, 23, 25], in which the document lists proposed by two candidate rankers for a given query are interleaved and the resulting list presented to the user, whose clicks are used to infer a noisy preference for one ranker over the other. Interleaving methods have been successfully applied in large-scale settings [7, 8].

The use of interleaved comparison methods gives rise to a key challenge in ranker evaluation: how to efficiently determine the best ranker from a set using only pairwise comparisons. This challenge can be formalized as a *K-armed dueling bandit* problem [32], wherein the best ranker is defined as the one that in expectation wins an interleaved comparison against every other ranker. Several algorithms have been proposed for this problem, including *interleaved filter* (IF) [32], *beat the mean* (BTM) [31], *sensitivity analysis of variables for generic exploration* (SAVAGE) [28], *relative upper confidence bound* (RUCB) [34] and *relative confidence sampling* (RCS) [33].

One challenge that such algorithms face is that the number of parameters that must be learned grows quadratically with K, the number of rankers. This in turn results in excessive exploration. The challenge is especially relevant in the case of web search, where a large number of rankers may be under consideration.[1] Some algorithms, such as SAVAGE, RUCB and RCS have difficulty scaling to large K. Other algorithms such as IF and BTM avoid this problem by making more restrictive assumptions, such as a total ordering of the rankers, that make it possible to identify the best ranker without explicitly considering all parameters. However, this approach is problematic because the required assumptions often do not hold in web search settings.

In this paper, we remedy these shortcomings by bridging the gap between these two approaches. Specifically, we propose and evaluate a new method for evaluating rankers, called *mergeRUCB*, that makes only weak assumptions about the K-armed dueling bandit problem, but provably requires only $\mathcal{O}(K)$ comparisons and therefore performs well when many rankers must be compared, as is typically the case in web search. As the name suggests, mergeRUCB uses a divide and conquer strategy to reduce the number of exploratory comparisons carried out by the evaluation process. It proceeds by grouping rankers into small batches so that fewer comparisons are needed before rankers can be eliminated.

To validate mergeRUCB, we evaluate its performance on large

WSDM'15, February 2–6, 2015, Shanghai, China.

ACM 978-1-4503-3317-7/15/02 ...$15.00.
http://dx.doi.org/10.1145/2684822.2685290.

[1] Kohavi et al. [21] report that on any given day over 200 concurrent experiments are being run at Bing, with users ending up in one of billions of possible variants of the site.

Microsoft, Yandex and Yahoo! learning to rank datasets. Our results show that mergeRUCB significantly and substantially outperforms multiple state of the art K-armed dueling bandit algorithms. Moreover, we provide theoretical performance guarantees that bound how much regret can be accumulated by mergeRUCB.

The main contributions of this paper are thus:

1. Proposing a new K-armed dueling bandit algorithm that scales well with the number of rankers (cf. §4).
2. Experimentally comparing its performance against existing K-armed dueling bandit algorithms (cf. §5 and §6).
3. Providing theoretical results guaranteeing the proper functioning of the algorithm (cf. §7); moreover, our regret bounds are the first that are completely linear in K, without imposing impractical assumptions.

2. PROBLEM SETTING

One approach to ranker evaluation is to use manual expert annotations in a TREC-like setting [29]. However, collecting the necessary annotations is expensive and, because they are not based on real users, such annotations may not reflect real users' actual information needs. An attractive alternative is thus to evaluate rankers *on-line* using feedback from real users. One way to obtain such feedback is to measure the *click-through rate* [17]. However, such feedback is often unreliable, since click-through rates can have substantial variance, particularly across users and topics [19, 20, 25, 26].

Fortunately, on-line feedback can also be obtained using *interleaved comparison methods*, which give *relative* feedback about how one ranker compares to another and have been shown to be more reliable [7, 23]. To compare two rankers on a given query, an interleaved comparison constructs a ranking that is an amalgamation of the rankings proposed by the two rankers for that query. Schemes for constructing the amalgamation include *balanced interleave* [18], *team draft* [25], *document constraints* [13], *probabilistic interleave* [15], and *optimized interleave* [24].

However, since interleaved comparison methods require feedback from real users, each comparison has significant real-world costs, i.e., if either of the compared rankers is poor, the interleaved ranking may also be poor, leaving the user dissatisfied. An important question is thus how to find the best ranker in a way that minimizes these costs.

This problem can be formalized as a K-armed dueling bandit problem [32], which is itself an extension of the K-*armed bandit* problem [3]. In the K-armed bandit problem, there are K rankers, $\{\rho_1, \ldots, \rho_K\}$. At each *time-step*, a ranker ρ_i can be tried, generating a *reward* drawn from an unknown stationary distribution with expected value μ_i, which might be a quantity like click-through rate [6].

The K-armed *dueling* bandit problem is a variation that models the relative feedback available in settings like ranker evaluation with interleaved comparison methods. The problem is defined by a matrix $\mathbf{P} = [p_{ij}]$ of *preference probabilities*. At each *time-step*, two rankers (ρ_i, ρ_j) are compared, e.g., using an interleaved comparison method, and with probability p_{ij} ranker ρ_i beats ρ_j. In this paper, we assume that there exists a *Condorcet winner* [28]: a ranker, which without loss of generality we label ρ_1, such that $p_{1i} > \frac{1}{2}$ for all $i > 1$. In other words, when interleaved with any other ranker, the Condorcet winner is expected to win.

The goal of a K-armed dueling bandit algorithm is to minimize the total *regret* it accumulates. Regret is a measure of the lost opportunity incurred by performing interleaved comparisons between suboptimal rankers. More specifically, the regret resulting from a comparison between rankers ρ_i and ρ_j is defined to be $\frac{p_{1i}+p_{1j}}{2} - 0.5$, i.e., the average suboptimality of the two rankers with respect to the Condorcet winner. Furthermore, *cumulative regret at time* T, R_T, is the sum of regret accumulated in the first T time-steps. In the context of online ranker evaluation for web search, we are interested in algorithms that minimize cumulative regret, since that means that fewer poor-quality rankings are shown to the users, hence lowering the risk of user frustration.

3. RELATED WORK

To our knowledge, the earliest method for the K-armed dueling bandit problem is *interleaved filter* (IF) [32], which proceeds as follows: a ranker $\hat{\rho}$ is randomly chosen to be compared against all other rankers; these comparisons are repeated until another ranker ρ' either loses to or beats $\hat{\rho}$ by a wide margin, i.e., the winner scores so many more wins over the loser that one can conclude with high confidence that the loser can be eliminated. If $\hat{\rho}$ is the winner, ρ' is eliminated from the pool of rankers and not compared against any other rankers. If $\hat{\rho}$ is the loser, it is eliminated from the pool of rankers and ρ' becomes the new $\hat{\rho}$. This process continues until all but one of the rankers is eliminated.

More recently, the *beat the mean* (BTM) algorithm has been shown to outperform IF [31]. BTM works by focusing exploration on the rankers that have been involved in the fewest comparisons. When it determines that a ranker fares on average too poorly in comparison to the remaining rankers, it removes it from consideration. More precisely, BTM considers the performance of each ranker against the *mean ranker* by averaging the ranker's scores against all other rankers and uses these estimates to decide which ranker should be eliminated. BTM is currently the state of the art for K-armed dueling bandit problems for the web search setting in which we are interested, that is, with large numbers of rankers K. However, so far, the performance of BTM has not been evaluated on large-scale online evaluation tasks using large learning to rank datasets. In this paper, we demonstrate that our proposed algorithm outperforms BTM on such datasets.

IF and BTM require the comparison probabilities p_{ij} to satisfy conditions that are difficult to verify without specific knowledge about the dueling bandit problem at hand. Specifically, IF and BTM require a *total ordering* $\{\rho_1, \ldots, \rho_K\}$ of the rankers to exist such that $p_{ij} > \frac{1}{2}$ for all $i < j$. In [33, 34], the authors show that, in the case of the LETOR and MSLR datasets, the probability of a total ordering existing decreases quickly as the number of rankers increases, due to the presence of cyclical relationships among rankers other than the Condorcet winner.

Sensitivity analysis of variables for generic exploration (SAVAGE) [28] is a more recent algorithm that outperforms both IF and BTM by a wide margin when the number of rankers is small. However, in cases with large numbers of rankers, SAVAGE was significantly outperformed by BTM. One version of SAVAGE, which we call *Condorcet SAVAGE*, assumes only the existence of a Condorcet winner and performed the best experimentally [28]. Condorcet SAVAGE compares pairs of rankers uniformly randomly until there exists a pair for which one of the rankers beats the other by a wide margin, in which case the loser is removed from the pool of rankers under consideration.

Relative upper confidence bound (RUCB) [34] and *relative confidence sampling* (RCS) [33] are two related algorithms that select rankers to compare by first choosing a ranker that is believed to be a good candidate for the Condorcet winner, and then choosing a second ranker that has the best chance of disproving the hypothesis

that the first ranker is indeed the Condorcet winner. RUCB uses confidence intervals to carry this out, while RCS uses sampling.

RUCB has strong theoretical guarantees, while RCS currently has none. However, RCS was shown to outperform RUCB on small-scale ranker evaluation problems. So far, neither algorithm has been tested on large-scale evaluation problems. Our extensive experimentation shows that, when there are few rankers, RCS tends to perform better than RUCB but, for the case with many rankers, RUCB performs better. However, for the case with many rankers, our proposed algorithm mergeRUCB outperforms both of them. Moreover, the regret bounds proven for RUCB take the form

$$\mathcal{O}(K^2) + \mathcal{O}(K \log T),$$

while our bound is linear in K, taking the form $\mathcal{O}(K \log T)$.

However, these bounds are not directly comparable because RUCB does not require δ, the probability of failure, to be passed to the algorithm explicitly. Instead, we let the user of the algorithm specify how risk averse they want the algorithm to be, which is not an option with RUCB or RCS.

4. METHOD

In this paper, we propose mergeRUCB, shown in Algorithm 1, to deal with online ranker evaluation problems involving many rankers. As with sorting algorithms, most naive approaches to the K-armed dueling bandit problem suffer from quadratic dependence on K because they require every ranker to be compared against every other ranker. However, this quadratic dependence can be avoided by a mergesort-style algorithm that carries out comparisons only "locally," i.e., items are placed in small batches that are processed separately and then merged together.

The same principle underlies mergeRUCB. The crucial difference is that, unlike in sorting, one comparison is not sufficient to determine which of a pair of rankers is better, since feedback is stochastic. Furthermore, the number of times two rankers must be compared is larger if the rankers are more similar. In the worst case, we have $p_{ij} = 0.5$ and the two rankers cannot be distinguished. This case is problematic because ρ_i and ρ_j might be weak rankers overall (i.e., lose badly to other rankers), in which case comparing them to each other many times will incur large regret. MergeRUCB deals with this difficulty by using the best ranker in the batch to eliminate the rest. If a batch contains only similar rankers and is thus too slow in eliminating rankers, it is combined with other batches that have more variety.

In the following, we explain the components of mergeRUCB, which proceeds in *stages* (Line 4). Before the first stage, the algorithm groups rankers into small batches $\mathcal{B}_i$ (Line 2). Then, within each stage, mergeRUCB carries out interleaved comparisons among rankers that reside in the same batch. At any given time, the choice of rankers to compare against each other inside a given batch is guided by a matrix $\mathbf{U}$ of upper confidence bounds (Line 7), which is obtained by optimistically estimating the preference probabilities p_{ij}: the optimism is included to ensure sufficient exploration among the rankers. The matrix $\mathbf{U}$ is used both to eliminate rankers if they lose to other rankers by a wide margin (Line 8) and to choose the ranker ρ_d (Line 10) that is selected so as to hasten the elimination of ρ_c, which is chosen randomly. The algorithm proceeds in this fashion until the number of remaining rankers becomes small (Line 12), at which point the stage is concluded by merging pairs of batches together to form bigger batches (Line 13). This initiates the next stage, and the process repeats until a single ranker remains. Our theoretical results state that the probability that this remaining ranker is the Condorcet winner is greater than $1 - \delta$.

Algorithm 1 mergeRUCB(δ)

Input: A set of rankers $\rho_1, \ldots, \rho_K$;
an oracle that can take a pair of rankers and return one as the winner (e.g., an interleaved comparison method);
the size of each partition, $p \geq 4$;
the maximum probability of failure, δ;
$\alpha > \frac{1}{2}$.

1: $\mathbf{W} \leftarrow \mathbf{0}_{K \times K}$ // *2D array of wins:* $\mathbf{W}_{ij}$ *is the number of times* ρ_i *has beaten* ρ_j

2: $\mathcal{B}_1 = \Big\{ \underbrace{\{\rho_1, \ldots, \rho_p\}}_{B_1}, \ldots, \underbrace{\{\rho_{(b_1-1)p+1}, \ldots, \rho_K\}}_{B_{b_1}} \Big\}$, a set of disjoint batches of rankers, with $b_1 = \lfloor \frac{K}{p} \rfloor$

3: $C(\delta) = \left\lceil \left(\frac{(4\alpha-1)K^2}{(2\alpha-1)\delta} \right)^{\frac{1}{2\alpha-1}} \right\rceil$

4: $S = 1$ // *The stage that the algorithm is in.*

5: **for** $t = 1, 2, \ldots$ **do**

6: $\quad i = t \mod b_S$

7: $\quad \mathbf{U} = \frac{\mathbf{W}}{\mathbf{W}+\mathbf{W}^T} + \sqrt{\frac{\alpha \ln(t+C(\delta))}{\mathbf{W}+\mathbf{W}^T}}$, where all operations are element-wise.

8: $\quad$ For any $\rho_k \in B_i$ if $\mathbf{U}_{kl} < \frac{1}{2}$ for any $\rho_l \in B_i$, remove ρ_k from B_i.

9: $\quad$ Select $\rho_c \in B_i$ randomly.

10: $\quad$ Set $d := \underset{\{l \mid \rho_l \in B_i \setminus \{\rho_c\}\}}{\arg\max} \mathbf{U}_{lc}$.

11: $\quad$ Compare ρ_c against ρ_d and increment $\mathbf{W}_{cd}$ if c won and $\mathbf{W}_{dc}$ otherwise.

12: $\quad$ **if** $\sum_i |B_i| \leq \frac{K}{2^I}$ **then**

13: $\quad\quad$ Combine pairs of batches of rankers so that each new batch has between $p/2$ and $3p/2$ rankers in it, pairing the smallest batches with the largest ones, making sure that each batch contains at least two rankers. Update the sets B_i, putting them all in the set $\mathcal{B}_S$, and define $b_S := |\mathcal{B}_S|$.

14: $\quad\quad S = S + 1$

15: $\quad$ **end if**

16: **end for**

5. EXPERIMENTAL SETUP

Our experiments address the following research questions:

RQ1 Does mergeRUCB outperform BTM, the state of the art online ranker evaluation algorithm for large-scale evaluation problems?

RQ2 How does mergeRUCB scale as the number of rankers increases in comparison to existing algorithms?

RQ3 How does the click model affect the scalability of the various algorithms?

RQ4 How does the performance of mergeRUCB depend on the parameters α and p? In particular, how do our default parameters perform?

We conduct experiments on four large-scale learning to rank datasets, namely the Microsoft Learning to Rank dataset (MSLR), the Yandex Internet Mathematics 2009 dataset and the Yahoo! Learning to Rank (YLR) Challenge datasets 1 and 2. Basic information about these datasets is included in Table 1.

Using these datasets, we create a finite set of rankers, each of which corresponds to a ranking feature provided in the dataset, e.g. PageRank or BM25. From this set of rankers, we choose subsets on which we test our algorithms. Therefore, in the experiments carried out here, the ranker evaluation task corresponds to determining which single feature constitutes the best ranker. From one

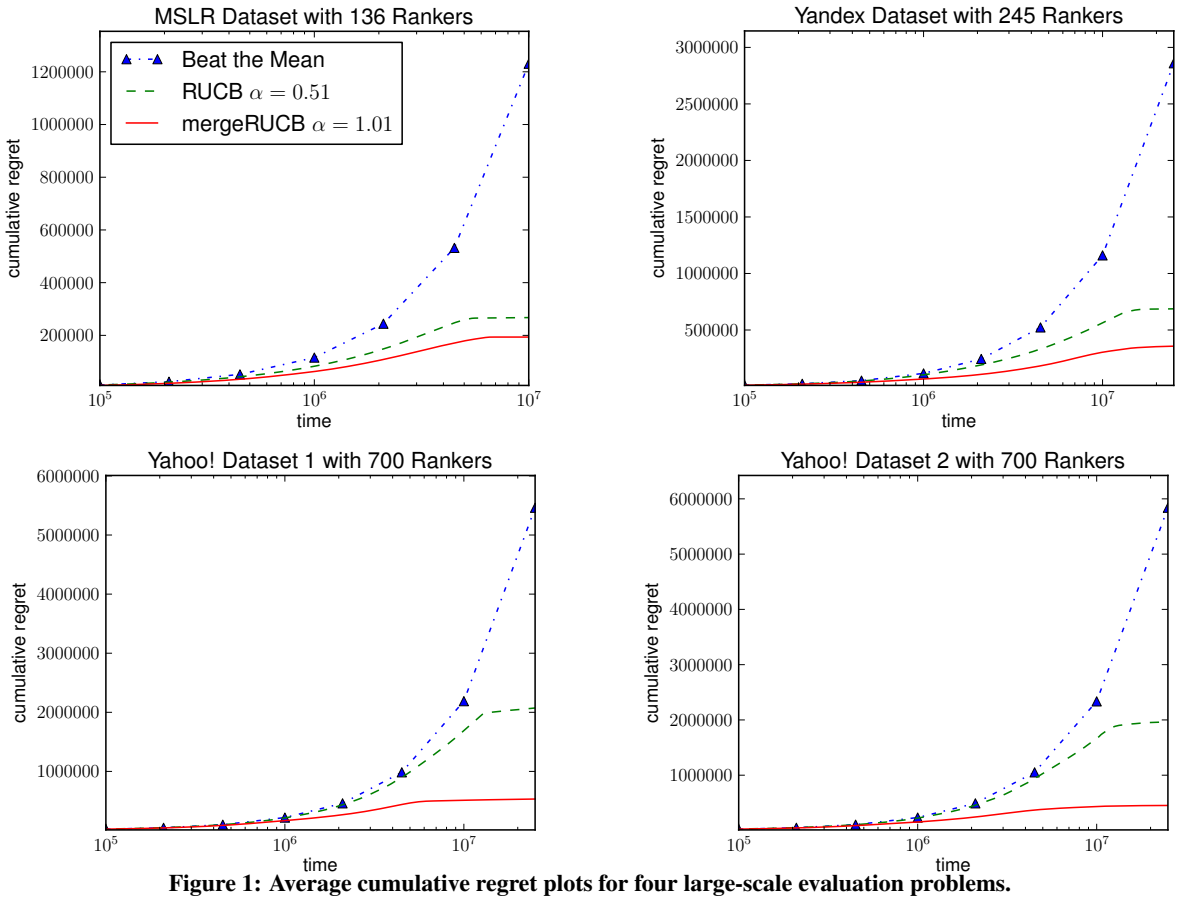

Figure 1: Average cumulative regret plots for four large-scale evaluation problems.

Table 1: The specifics of the datasets used.

Datasets	Queries	URLs	Features	Reference
MSLR-WEB30K	31,531	3,771,125	136	[22]
Yandex	9,124	97,290	245	[30]
YLR Set 1	19,944	473,134	700	[5]
YLR Set 2	1,266	34,815	700	[5]

point of view, this is a rather artificial setup because when optimizing a search engine one is rarely (if ever) interested in determining which single feature performs the best; instead, what is often of greater interest is comparing different rankers that were produced as the outcome of different learning to rank algorithms. However, what determines the difficulty of a dueling bandit problem is not the quality of the individual rankers but rather the difference in quality between pairs of rankers. So, for instance, evaluating K rankers that are very small modifications on the random ranker is as difficult as, say, evaluating K rankers that are small variations on LambdaMART [4]. More precisely, given a preference matrix $\mathbf{P}$, what determines the difficulty it poses for the dueling bandit algorithm is how close to 0.5 its entries are, and indeed, from our experience, the preference matrices of the feature rankers of the datasets listed in Table 1 does contains many entries that are very close to 0.5.

Given the above observation, we opted to use feature rankers in our experiments rather than, for instance, rankers produced by different learning to rank algorithms in order to both facilitate the reproducibility of our experimental results and to avoid having our results depend upon arbitrary choices made in the process of training such rankers. Nonetheless, comparing various dueling bandit algorithm on better performing rankers is an important research question that we postpone to future work.

To compare a pair of rankers, we use *probabilistic interleave* (PI) [14], though any other interleaved comparison method could be used instead. To model the user's click behavior on the resulting interleaved lists, we employ a probabilistic user model [10, 14] that uses as input the manual labels (classifying documents as relevant or not for given queries) provided with both datasets. Queries are sampled randomly and clicks are generated probabilistically by conditioning on these assessments using a user model that resembles the behavior of an actual user [12]. This approach follows an experimental paradigm that has previously been used for assessing the performance of rankers [13–16]. We used a software package called Lerot [27] to carry out these comparisons.[2]

For the large-scale experiments in §6.1, aimed at answering RQ1, we use all of the feature rankers available in these datasets and perform the comparisons between rankers by directly using Lerot to simulate interleaved comparisons. In this case, our assumption that there exists a Condorcet winner happens to be satisfied in the case of all four datasets. For all other experiments, for each value of K tested, we choose 10 subsets of rankers of size K and apply each algorithm to each subset: this choice is made by sampling subsets of size K at random and keeping the first 10 that have Condorcet winners. As illustrated in [33], the probability that a subset has a Condorcet winner depends on K, but is generally very high. In addition, since the Lerot-based experiments for RQ1 took three months to complete,[3] we use a faster proxy setup for

[2]The interested reader can find the repository at the following URL: `https://bitbucket.org/ilps/lerot`

[3]This was primarily due to shortcomings of the competing algo-

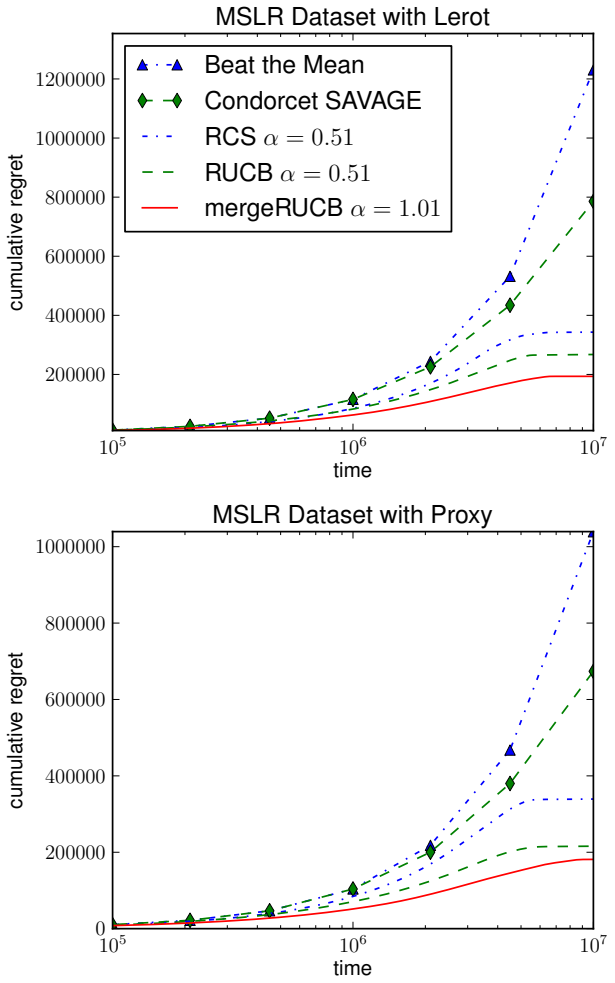

Figure 2: Average cumulative regret on the 136-ranker evaluation problem arising from the MSLR dataset using Lerot (top) or the proxy approach (bottom).

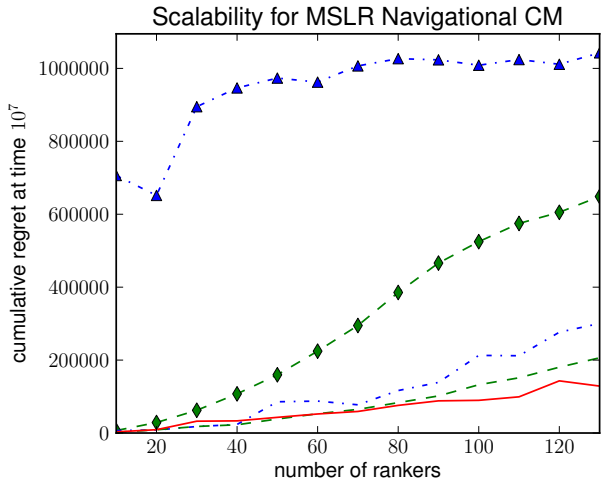

Figure 3: Average cumulative regret after 10^7 iterations on K-ranker evaluation problems with K ranging from 10 to 130.

the other experiments: for each pair of feature rankers ρ_i, ρ_j in the MSLR dataset, we estimate the probability p_{ij} that ρ_i beats ρ_j by simulating 400,000 interleaved comparisons between the two using Lerot.[4] Given these numbers p_{ij}, in the remaining experiments, we perform comparisons between rankers ρ_i and ρ_j for each pair (i, j) by drawing a sample from the Bernoulli distribution with mean p_{ij}, i.e., by flipping a biased coin. This is a standard approach to evaluating dueling bandit algorithms (cf. [31, 32, 34]). We verify the accuracy of the proxy approach in §6.2.

In principle, the proxy approach could also be used for the Yandex and the Yahoo! datasets. However, given the higher number of feature rankers in those datasets, obtaining the same level of precision on the preference probabilities p_{ij} would require orders of magnitude more computational resources. Indeed, it is more efficient to use Lerot directly for those experiments. Given this constraint, our experiments using the proxy method cannot go beyond 136 rankers.

In all experiments other than those in §6.5, we use the following parameter settings: $\alpha = 1.01$ and $p = 4$. In fact, the only constraint on α is that it should be greater than 0.5 in order for $C(\delta)$ (cf. Line 3 in Algorithm 1) to be well-defined and for our theoretical results in §7 to hold. However, as α approaches 0.5, the expression for $C(\delta)$ grows super-exponentially as a function of α, and so the benefits of having more slowly growing confidence intervals (cf. Line 7 of Algorithm 1) are outweighed by the added exploration caused by starting with larger confidence intervals. Indeed, as demonstrated in §6.5, there is little or no gain from changing these parameters from the above values. Moreover, for all of our experiments, we chose the probability of failure, δ to be 0.01. Finally, all experiments other than those in §6.4 used the navigational click model (cf. Table II of [15]) to simulate user click behavior.

6. RESULTS AND DISCUSSION

In this section, we present our experimental results.

6.1 Large scale experiments

We first address our main research question, **RQ1**. We tested mergeRUCB on the full set of feature vectors of the four large learning to rank datasets described in Table 1, directly using Lerot instead of the proxy approach. The MSLR results, shown in Fig. 1 (top-left), were carried out for 10 million time-steps, since two of the three algorithms converge to the Condorcet winner within that time frame. For the remaining datasets, we extended the horizon to 25 million time-steps, again to make sure two of the three algorithms converge. These results are shown in the remaining plots in Fig. 1. Note that, in these plots and those that follow, the time axis uses a log scale, while the vertical axis uses a linear scale.

For these experiments, we tested three algorithms: mergeRUCB and RUCB, which had the best performance in the scalability experiments in §6.3, together with BTM, which is the state of the art K-armed dueling bandit algorithm for large K, according to [28, 31]. These plots show that, as the number of rankers increases (going from 136 to 245 to 700), so does the difference between the performance of mergeRUCB and the remaining algorithms.

6.2 Lerot simulation vs Bernoulli samples

The remaining results presented in this work use the proxy approach described in §5. So, before proceeding further, we validate the proxy approach by showing that it provides qualitatively similar results to those generated with Lerot. To do so, we compare the performances of five K-armed dueling bandit algorithms on

rithms, which need to be run sequentially. By contrast, mergeRUCB can easily be parallelized across different batches.

[4]The resulting matrices can be found here (as Numpy matrices): `http://ilps.science.uva.nl/sites/ilps.science.uva.nl/files/PrefMats.zip`

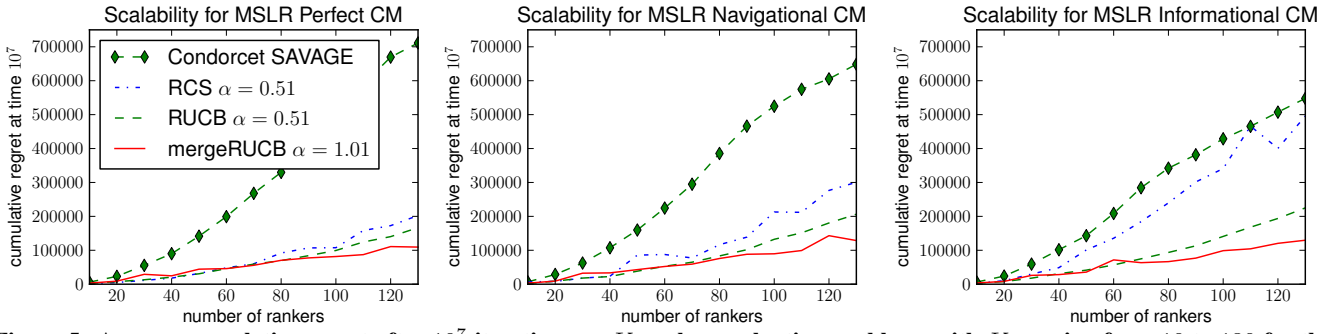

Figure 5: Average cumulative regret after 10^7 iterations on K-ranker evaluation problems with K ranging from 10 to 130 for the perfect (left), navigational (middle), and informational (right) click models.

the MSLR dataset using both approaches. The results for Lerot are shown in Fig. 2 (top), while those of the proxy approach are shown in Fig. 2 (bottom). Comparing the two plots shows that there is no qualitative difference in the relative performance of the various dueling bandit algorithms under consideration here. Consequently, we use the proxy method to conduct the experiments described in the rest of this section.

6.3 Dependence on K

To address **RQ2**, we compare 5 dueling-bandit algorithms on K-ranker evaluation experiments with K ranging from 10 to 130 in increments of 10 with the K rankers chosen randomly from the 136 feature rankers in the MSLR dataset.

Fig. 3 shows the results: the horizontal axis measures K, the number of rankers, while the vertical axis shows the regret accumulated after 10^7 iterations. As this plot demonstrates, for $K \geq 70$, mergeRUCB outperforms all other dueling bandit algorithms.

Of course, while Fig. 3 shows performance across different values of K, it does so for only one moment in time: after 10^7 iterations. However, comparing Fig. 3 to Fig. 2 confirms that, for $K = 136$, the regret accumulated by the algorithm after 10^7 timesteps is a good indication of the overall performance of the algorithm over time. Fig. 4 confirms that the same is true when $K = 70$.

6.4 Effect of click models

To address **RQ3**, we conducted the same scalability test as in §6.3, using three different click models proposed in [15], namely the *perfect*, *navigational* and *informational* click models. The perfect click model represents the behavior of a persistent user, who inspects every single document in the retrieved list and clicks on each document with a probability proportional to the document's relevance to the given query. The navigational click model simulates the behavior of a user who is trying to satisfy a specific information need and is likely to stop inspecting the items in the list upon viewing a relevant document. Finally, the informational click model mimics the behavior of a user whose information need is not satisfied by a single document and is trying to gather information about a general topic. Accordingly, the informational click model is more likely to continue inspecting the items retrieved by the ranker even after encountering a relevant document.

The results, shown in Fig. 5, demonstrate that RCS is affected more severely by the click model than either mergeRUCB or RUCB. This is because, in our experience, RCS tends to be sensitive to the margins by which the Condorcet winner beats the remaining rankers: as these gaps shrink, the performance of RCS degrades dramatically. This is precisely what takes place when one replaces the perfect click model with the navigational one and the latter with the informational click model, since doing so increases the number of clicks in interleaved comparisons, making them noisier.

6.5 Parameter dependence

To address **RQ4**, we repeated the experiments in §6.3, using the following grid of parameters:

$$(p, \alpha) \in \{4, 6, 8, 10\} \times \{0.71, 0.81, 0.91, 1.01, 1.11, 1.21\}.$$

Fig. 6 shows, for each number of rankers, the minimum and maximum cumulative regrets accumulated by mergeRUCB across the above set of parameters, as well as the regret for the default parameters used in the other experiments. Note that the vertical axis uses a log scale, which is chosen to facilitate comparing the three curves for small values of K, since in a linear plot they would be too close to distinguish from each other. As can be seen from the plots, the

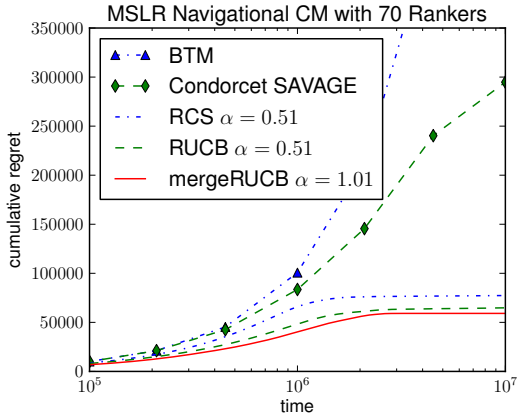

Figure 4: Average cumulative regret on the 70-ranker evaluation problem arising from MSLR.

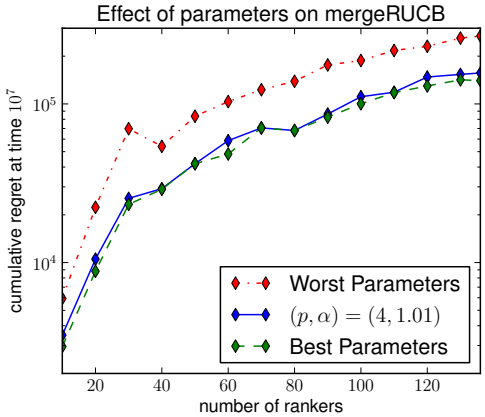

Figure 6: Effect of parameters on mergeRUCB's with MSLR dataset and navigational click model.

regret accumulated by mergeRUCB, using the default parameters $(p, \alpha) = (4, 1.01)$, is consistently close, if not equal, to the regret accumulated by the best choice of parameters, which validates our intuition that the default parameters are sensible.

7. PERFORMANCE GUARANTEES

In this section, we provide theoretical guarantees for the proper functioning and scalability of mergeRUCB. We begin by listing a number of reasonable assumptions that we impose upon the problem in order to guarantee the proper functioning of the algorithm:

A1. We assume that there is no repetition of rankers, i.e., any pair of rankers ρ_i and ρ_j are different and thus $p_{ij} \neq 0.5$, unless both rankers are *uninformative*: they provide no useful information and so lose to all other rankers, i.e., $p_{ki} \geq 0.5$ and $p_{kj} \geq 0.5$ for all k.

A2. We assume that at most a third of the rankers are uninformative.

In online ranker evaluation in web search settings, assumption **A1** is reasonable because the rankers ρ_i under evaluation are typically the result of substantial deliberation and research and so the chances of the same informative ranker appearing twice are slim. The second assumption is motivated by the Yahoo! Learning to Rank challenge dataset, in which either 104 or 181 (depending on the dataset) out of 700 feature rankers always return zero. More generally, it is plausible that some uninformative rankers are inadvertently included in the evaluation process. However, if there are too many of them, the evaluation task will be lengthened.

Here, we provide a high probability bound on the regret accumulated by mergeRUCB; Table 2 lists our notation.

THEOREM 1. *Given a K-armed dueling bandit problem with rankers $\rho_1, \ldots, \rho_K$ with ρ_1 the Condorcet winner, then if we apply mergeRUCB(δ), with probability $1-\delta$ we have the following bound on cumulative regret at time T:*

$$R_T \leq \frac{16\alpha pK \ln(T + C(\delta)) \max_j \Delta_{1j}}{\min_{S=1,\ldots,\lceil \log_2 K \rceil} \widehat{\Delta}_S^2} \leq \frac{8\alpha pK \ln(T + C(\delta))}{\min_{\{(i,j) \mid p_{ij} \neq 0.5\}} \Delta_{ij}^2}.$$

This theorem says that if mergeRUCB is run for T time-steps with probability of failure set to δ, then with probability $1 - \delta$, the total regret accumulated by the algorithm is bounded by an expression that is logarithmic in T and linear in K. This in turn tells us that the number of suboptimal interleaved comparisons grows linearly in K, since accumulating non-zero regret corresponds to suboptimal comparisons. Unlike existing results in the literature, the strongest of which take the form $\mathcal{O}(K^2) + \mathcal{O}(K \log T)$, Theorem 1 is the first regret bound that is completely linear in K.

Furthermore, even though as stated the above theorem is a high probability bound, by setting $\delta = 1/T$, we obtain a bound on the expected regret of mergeRUCB at time T as follows: since the maximum amount of regret that the algorithm can accumulate in the first T time-steps is bounded by T, we have

$$\begin{aligned}
\mathbf{E}R_T &\leq \delta T + (1-\delta)\frac{8\alpha pK \ln(T + C(\delta))}{\Delta_{\min}^2} \\
&\leq 1 + \frac{8\alpha pK \ln T}{\Delta_{\min}^2} + \frac{8\alpha pKC(1/T)}{T\Delta_{\min}^2}, \\
&\leq 1 + \frac{8\alpha pK \ln T}{\Delta_{\min}^2} + \frac{8\alpha pK \left(\frac{T(4\alpha-1)K^2}{(2\alpha-1)}\right)^{\frac{1}{2\alpha-1}}}{T\Delta_{\min}^2},
\end{aligned}$$

Table 2: List of notation used in Section 7.

Symbol	Definition
K	Number of rankers
α	Exploration parameter in Algorithm 1
δ	Probability of failure
p	Initial size of the batches
S	Stage of the algorithm
$\mathcal{B}_S$	Set of batches in stage S
b_S	Number of batches in stage S
R_T	Cumulative regret at time T
$w_{ij}(t)$	Number of times ρ_i beat ρ_j in the first t time-steps
$N_{ij}(t)$	$w_{ij}(t) + w_{ji}(t)$
$u_{ij}(t)$	$\mathbf{U}_{ij} := \frac{w_{ij}(t)}{N_{ij}(t)} + \sqrt{\frac{\alpha \ln t}{N_{ij}(t)}}$
$l_{ij}(t)$	$1 - u_{ji}(t)$
$C(\delta)$	$\left\lceil \left(\frac{(4\alpha-1)K^2}{(2\alpha-1)\delta}\right)^{\frac{1}{2\alpha-1}} \right\rceil$
Δ_{ij}	$p_{ij} - 0.5$
$\Delta_{B,\min}$	$\min_{i,j \in B} \Delta_{ij}$
T_B	$\frac{4\alpha\binom{q-1}{2}\log(T + C(\delta))}{\Delta_{B,\min}^2}$
T_i	T_{B_i}
$\widehat{\Delta}_S$	$\left(\frac{2b_S}{3} + 1\right)^{th}$ largest element of $\{\Delta_{B,\min} \mid B \in \mathcal{B}_S\}$
$\widehat{T}_S$	$\frac{8\alpha pK \ln(T + C(\delta))}{\widehat{\Delta}_S^2}$
$\ln t$	Natural logarithm of t

where $\Delta_{\min} = \min_{\{(i,j) \mid p_{ij} \neq 0.5\}} \Delta_{ij}^2$ and the second inequality is obtained by using a Taylor expansion of $\ln t$ at $t = T$. Now, if $\alpha \geq 1$, the last summand in the right-hand side of the above inequality is in $\mathcal{O}(1)$, and so we have a finite-horizon expected regret bound of the form $\mathcal{O}(K \ln T)$. Moreover, this finite horizon bound can be turned into an infinite horizon one (up to $\ln \ln T$ factors) using the 'squaring trick' [2]. We would like to emphasize that these results hold under very general assumptions that do not preclude the existence of cyclical relationships among the rankers.

The proof of Theorem 1 relies on the following lemma.

LEMMA 2. *In mergeRUCB(δ), consider a batch B of size q, at least one of whose rankers is informative. Let $\Delta_{B,\min}$ denote the smallest nonzero gap $\Delta_{kl} := |p_{kl} - \frac{1}{2}| \neq 0$, with $\rho_k, \rho_l \in B$. Then, the number of comparisons N_B that could have happened between pairs of rankers in B before it is merged with another batch is bounded with probability $1 - \delta$ as follows:*

$$N_B < T_B := \frac{4\alpha\binom{q-1}{2}\ln(T + C(\delta))}{\Delta_{B,\min}^2}.$$

The proof of this lemma follows directly from the fact that the number of comparisons between any pair of rankers in the batch is at most $\frac{4\alpha \ln(T+C(\delta))}{\Delta_{B,\min}^2}$, as proven in Lemma 3 below, since there are $\binom{q-1}{2}$ distinct pairs of rankers in B.

LEMMA 3. *Given any pair of distinct rankers $\rho_i, \rho_j \in B$, the maximum number of comparisons that could have been carried out between these two rankers in the first T time-steps of Algorithm 1 before a merger between B and another batch occurs, is bounded by $\frac{4\alpha \ln(T+C(\delta))}{\Delta_{B,\min}^2}$.*

The proof of Lemma 3 considers the two possible cases: either at least one of the rankers under consideration is informative or both are uninformative. In the first case, if the two rankers have

been compared more times than the above number, we show that one of the two must have eliminated the other, while in the second case, if the two rankers have been compared too many times, then a third, informative ranker (whose existence is guaranteed by the assumption of the lemma) must have eliminated one of them.

The proof of the above lemma relies on the following result, which we repeat here for the reader's convenience:

LEMMA 4 (LEMMA 1 IN [34]). *Let $\mathbf{P} := [p_{ij}]$ be the preference matrix of a K-armed dueling bandit problem with arms $\{a_1, \ldots, a_K\}$. Then, for any dueling bandit algorithm and any $\alpha > \frac{1}{2}$ and $\delta > 0$, we have*

$$P\Big(\forall\, t > C(\delta), i, j,\ p_{ij} \in [l_{ij}(t), u_{ij}(t)]\Big) > 1 - \delta.$$

PROOF OF LEMMA 3. Let us begin by assuming that the number of comparisons between ρ_i and ρ_j is greater than $\frac{4\alpha \ln(T+C(\delta))}{\Delta^2_{B,\min}}$, and let us distinguish between two cases:

1. *At least one of ρ_i and ρ_j is informative*: in this case, by assumption **A1**, we know that $p_{ij} \neq 0.5$, and moreover by Lemma 4, we know that with probability $1-\delta$ we have $p_{ij} \in [l_{ij}(t), u_{ij}(t)]$, with t being the last time that ρ_i was compared against ρ_j and $l_{ij} := 1 - u_{ji}$. However, this tells us that one of the two rankers should have been eliminated already, since we have

$$u_{ij}(t) - l_{ij}(t) = 2\sqrt{\frac{\alpha \ln(t + C(\delta))}{N_{ij}(t)}} \leq 2\sqrt{\frac{\alpha \ln(T + C(\delta))}{N_{ij}(t)}}$$
$$< 2\sqrt{\frac{\alpha \ln(t + C(\delta))}{\frac{4\alpha \ln(T + C(\delta))}{\Delta^2_{B,\min}}}} = \Delta_{B,\min} \leq \Delta_{ij}, \qquad (1)$$

   where the last inequality is due to our assumption that $N_{ij}(t) > \frac{4\alpha \ln(T+C(\delta))}{\Delta^2_{B,\min}}$. Therefore, the confidence interval $[l_{ij}(t), u_{ij}(t)]$ does not contain 0.5, which is the criterion used by Algorithm 1 to eliminate rankers.

2. *Rankers ρ_i and ρ_j are both uninformative*: by assumption **A1** and Lemma 4, uninformative rankers cannot eliminate informative rankers, so no matter how many rankers have been eliminated from B, there must be an uneliminated third ranker ρ_k that is informative in the batch together with ρ_i and ρ_j, and by assumption **A1**, we have $p_{ki} > 0.5$ and $p_{kj} > 0.5$. Again, applying Lemma 4 as in the previous case, we know that with probability $1 - \delta$ we have $0.5 = p_{ij} \in [l_{ij}(t), u_{ij}(t)]$; on the other hand, using the same chain of inequalities as in (1), we can deduce that

$$u_{ij}(t) - l_{ij}(t) < \Delta_{B,\min} \leq \min\{\Delta_{ki}, \Delta_{kj}\}. \qquad (2)$$

   Now, in order for ρ_i to have been compared to ρ_j at time t, we must have had one of the following two scenarios:

   (a) mergeRUCB chose $c = i$ and $d = j$ at time t: this requires the satisfaction of the following two conditions:
   - $u_{ij}(t) \geq 0.5$, by Line 8 of Algorithm 1.
   - $l_{ij}(t) \leq p_{ik}$: this is because in order to have $d = j$, we must have $u_{ji}(t) \geq u_{ki}(t)$ and by Lemma 4, we have $u_{ki}(t) \geq p_{ki}$, and so $l_{ij}(t) := 1 - u_{ji}(t) \leq 1 - p_{ki} = p_{ik}$.

   This means that we have $u_{ij}(t) - l_{ij}(t) \geq \Delta_{ki}$. However, this contradicts inequality (2), so we could not have had $(c, d) = (i, j)$.

   (b) mergeRUCB chose $c = j$ and $d = i$ at time t: repeating the same argument as in the previous case with i and j swapped, we get $u_{ji}(t) - l_{ji}(t) \geq \Delta_{kj}$, which also contradicts inequality (2).

Therefore, our assumption that the number of comparisons between ρ_i and ρ_j is greater than $\frac{4\alpha \ln(T+C(\delta))}{\Delta^2_{B,\min}}$ cannot hold in either scenario. □

Next, we prove Theorem 1, the main idea of which is as follows. The central difficulty in the proof is that there may exist batches that consist entirely of uninformative rankers. This is problematic because in these batches no rankers are eliminated, since for each pair of rankers ρ_i, ρ_j in such a batch, we have $p_{ij} = 0.5$ and so with high probability we have neither $u_{ij} < 0.5$ nor $u_{ji} < 0.5$. The proof overcomes this difficulty by showing that such fully uninformative batches all disappear at the end of the first stage of the algorithm. This occurs because of how the batches are merged at the end of each stage (cf. Line 13 of Algorithm 1): the largest batches are combined with the smallest ones. Since uninformative batches inevitably fail to eliminate any rankers, they have the largest number of rankers, while the smallest batches are guaranteed to contain informative rankers. Therefore, from the second stage onwards, mergeRUCB is guaranteed not to compare rankers in a batch, none of whose elements will be eliminated.

PROOF OF THEOREM 1. We begin by considering the first stage of the algorithm:

$S = 1$ During the first stage, we have two types of batches: those that consist solely of uninformative rankers and those that contain at least one informative ranker. Assumption **A2** implies that at least two thirds of the batches have at least one informative ranker, so we can apply Lemma 2 to them.

To estimate number of time-steps mergeRUCB spends in its first stage, we introduce the following notation: recall from Algorithm 1 that b_1 is the number of partitions in the first stage of the algorithm and let $\widehat{\Delta}_1$ denote the $\left(\frac{2b_1}{3} + 1\right)^{th}$ largest number in the set $\{\Delta_{B,\min} | B \in \mathcal{B}_1\}$. Now, once all but one of the rankers in every batch B with $\Delta_{B,\min} \geq \widehat{\Delta}_1$ have been eliminated, the algorithm moves to the next stage. This occurs because at least half of the rankers have been eliminated, since there are $\frac{2b_1}{3} + 1$ batches, inside which $p - 1$ rankers are eliminated, and so the total number of eliminated rankers is at least

$$\left(\frac{2b_1}{3} + 1\right)(p-1) \geq \frac{2(b_1+1)}{3}(p-1) \geq \frac{2K}{3p}(p-1)$$
$$\geq \frac{2K}{3}\frac{3}{4} \geq \frac{K}{2}. \qquad \text{(since } p \geq 4\text{)}$$

Therefore, Line 12 of Algorithm 1 forces the next stage to begin. Now, applying Lemma 2 to the $\frac{2b_1}{3}$ batches B with $\Delta_{B,\min} \geq \widehat{\Delta}_1$, and using the fact that the size of the batches is at most $2p$ (cf. Line 2 of Algorithm 1), we can conclude that with probability $1 - \delta$ the number of time-steps in the first stage of mergeRUCB could not have been more than

$$\frac{K}{p} \times \frac{4\alpha\binom{2p}{2}\ln(T + C(\delta))}{\widehat{\Delta}_1^2} \leq \frac{8\alpha p K \ln(T + C(\delta))}{\widehat{\Delta}_1^2} =: \widehat{T}_1.$$

$S \geq 2$ At the end of the first stage of the algorithm, we combine the largest remaining batches with the smallest ones. The fact that exactly half of the rankers were eliminated in the first stage implies that this policy for combining batches forces every fully uninformative batch to acquire an informative ranker, since by

Assumption **A1** and Lemma 4, the probability of an uninformative ranker eliminating an informative ranker is less than δ. Hence, from this point on, we can apply Lemma 2 to every batch.

We can use a similar argument as with the first stage of the algorithm to bound the number of time-steps that mergeRUCB would spend in the S^{th} stage. To that end, let $\widehat{\Delta}_S$ denote the $\left(\frac{2b_S}{3}+1\right)^{th}$ largest number in the set $\{\Delta_{B,\min}|B \in \mathcal{B}_S\}$. Now, applying the same argument as above and using the fact that in stage S we have $K/2^{S-1}$ rankers, we get that the number of comparisons in stage S of mergeRUCB is bounded by $\frac{8\alpha pK\ln(T+C(\delta))}{2^{S-1}\widehat{\Delta}_S^2} =: \widehat{T}_S$.

After $\lceil \log_2 K \rceil$ stages, only a single ranker remains, beyond which point mergeRUCB goes on interleaving that ranker with itself. This ranker is the Condorcet winner with probability $1-\delta$ because the probability of the Condorcet winner being eliminated by another ranker is at most δ. Therefore, in order to estimate the total regret accumulated by mergeRUCB we can sum the $\widehat{T}_S$ for $S = 1, \ldots, \lceil \log_2 K \rceil$ and multiply the result by the maximum regret any comparison can result in, which is $\max_j \Delta_{1j}$. This gives the bound in the statement of Theorem 1 once we notice that

$$\sum_{S=1}^{\lceil \log_2 K \rceil} \frac{1}{2^{S-1}\widehat{\Delta}_S} \leq \frac{2}{\min\limits_{S=1,\ldots,\lceil \log_2 K \rceil} \widehat{\Delta}_S^2}$$

This concludes the proof of Theorem 1. □

A tighter result could be obtained by bounding the regret for comparisons inside batch B during the S^{th} stage of mergeRUCB with $\max_{i,j\in B} \Delta_{ij}$ rather than $\max_j \Delta_{1j}$. This would make the statement of the theorem more difficult to understand and the proof longer, so we opted for this simpler, slightly weaker bound for the sake of readability and brevity.

8. RECENT DEVELOPMENTS

Recently, three new dueling bandit algorithms were proposed: Doubler, MultiSBM and Sparring [1]. While Doubler and MultiSBM have performance guarantees, these hold only under the stringent assumption that each ranker has an underlying absolute utility, from which the preference matrix $\mathbf{P}$ is generated. This precludes the presence of cyclical preference relationships among the rankers and, as demonstrated in [33, 34], such a total ordering assumption is often violated in practice. Sparring has no performance guarantees but outperforms Doubler and MultiSBM in the experiments presented in [1].

Because of the recency of their publication, a comprehensive experimental comparison involving these new methods was not feasible, since large-scale experiments with Lerot would require months to complete and the proxy method cannot go beyond 136 rankers, as discussed in §5. However, we do offer some preliminary small-scale results here. We excluded Doubler from these results since in the experiments performed in [1], it was outperformed by all other algorithms in every single experiment. Our results, shown in Fig. 7, suggest that MultiSBM is consistently outperformed by most existing state-of-the-art algorithms, while the situation is more complicated for Sparring.

Sparring performs well in some cases, as shown in Fig. 7 (top). The fact that Sparring outperforms mergeRUCB in this setting with few rankers is not surprising, since mergeRUCB is designed to excel given many rankers and is already outperformed by both RUCB and RCS when there are only few rankers. However, even given only few rankers, there are settings in which Sparring does poorly, as shown in Fig. 7 (bottom). In our preliminary experiments, Sparring's performance degrades dramatically as the top *Borda scores* become closer to each other, where the Borda score for ranker ρ_i is defined to be $\sum_j p_{ij}$ [28]. This phenomenon occurs because Sparring chooses rankers to interleave based on its estimate of their Borda scores. Given the potential of Sparring to fail in this way, its practical utility in online ranker evaluation may be limited. More extensive empirical comparisons, especially in settings with many rankers, as well as a theoretical analysis of Sparring, are needed.

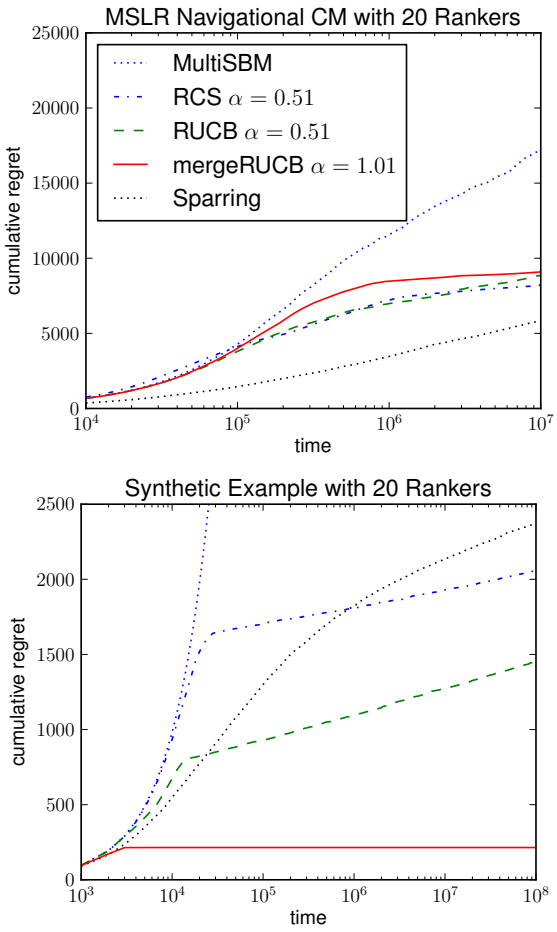

Figure 7: Comparisons against Sparring.

9. CONCLUSION

In this paper, we proposed a new algorithm, called mergeRUCB, for the online ranker evaluation problem in situations that are of particular interest for web search, i.e., with large numbers of rankers. We conducted extensive experimentation to understand the behavior of this algorithm in comparison to other online evaluations algorithms. The results of these experiments demonstrate that mergeRUCB can significantly outperform existing state of the art algorithms on large-scale evaluation problems. Moreover, we provided theoretical guarantees proving the proper functioning of mergeRUCB.

The algorithm presented in this paper makes it feasible for search engines to perform large-scale online ranker evaluation experiments that might be too costly if other K-armed dueling bandit algorithms were used. Furthermore, our theoretical results provide the necessary assurance for undertaking such large-scale evaluation tasks.

Since we care more about the worst-case performance of ranker evaluation algorithms than their average performance, our regret bounds are high probability rather than in expectation, unlike previous work such as [32, 34].

In future work, we intend to conduct more thorough comparisons with the recently proposed Sparring algorithm, whose behavior is not as well understood and whose performance can vary substantially, as we have briefly seen, depending on the online ranker evaluation at hand. Another possible direction for future research is to use the "locality" ideas in this paper together with continuous armed bandit algorithms like Branch and Bound [11] to extend online ranker evaluation to an infinite number of rankers, for instance if the rankers are defined in terms of continuous parameters.

Acknowledgments. We would like to thank the reviewers for their many helpful comments on this manuscript.

This research was partially supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.011.005, 612.001.116, HOR-11-10, 640.-006.013, 612.066.930, the Center for Creation, Content and Technology (CCCT), the Dutch national program COMMIT, the ESF Research Network Program ELIAS, the Elite Network Shifts project funded by the Royal Dutch Academy of Sciences (KNAW), the Netherlands eScience Center under project number 027.012.105, the Yahoo! Faculty Research and Engagement Program, the Microsoft Research PhD program, and the HPC Fund.

REFERENCES

[1] N. Ailon, Z. Karnin, and T. Joachims. Reducing dueling bandits to cardinal bandits. In *ICML*, 2014.

[2] K. Amin, M. Kearns, and U. Syed. Bandits, query learning, and the haystack dimension. In *COLT 2011 - The 24th Annual Conference on Learning Theory, June 9-11, 2011, Budapest, Hungary*, pages 87–106, 2011.

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

[4] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. Techn. Report MSR-TR-2010-82, Microsoft Research, June 2010.

[5] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research-Proceedings Track*, 14:1–24, 2011.

[6] O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *NIPS*, 2011.

[7] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Trans. Inf. Syst.*, 30:1–41, 2012.

[8] A. Chuklin, A. Schuth, K. Hofmann, P. Serdyukov, and M. de Rijke. Evaluating aggregated search using interleaving. In *CIKM 2013*. ACM, 2013.

[9] C. W. Cleverdon, J. Mills, and M. Keen. Factors determining the performance of indexing systems. In *ASLIB Cranfield project*. 1966.

[10] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08*, pages 87–94, 2008.

[11] N. de Freitas, A. Smola, and M. Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *ICML '12*, 2012.

[12] F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *WSDM '09*, pages 124–131, New York, NY, USA, 2009. ACM.

[13] J. He, C. Zhai, and X. Li. Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In *CIKM '09*, pages 2029–2032, 2009.

[14] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *CIKM '11*, pages 249–258, USA, 2011.

[15] K. Hofmann, S. Whiteson, and M. de Rijke. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Trans. Inf. Syst.*, 31(4), 2013.

[16] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1):63–90, 2013.

[17] S. Ji, K. Zhou, C. Liao, Z. Zheng, G.-R. Xue, O. Chapelle, G. Sun, and H. Zha. Global ranking by exploiting user clicks. In *SIGIR '09*, 2009.

[18] T. Joachims. Evaluating retrieval performance using clickthrough data. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining*, pages 79–96. 2003.

[19] S. Jung, J. L. Herlocker, and J. Webster. Click data as implicit relevance feedback in web search. *Information Processing & Management*, 43(3):791–807, 2007.

[20] J. Kamps, M. Koolen, and A. Trotman. Comparative analysis of clicks and judgments for IR evaluation. In *WSCD'09*, pages 80–87, 2009.

[21] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann. Online controlled experiments at large scale. In *KDD '13*, pages 1168–1176. ACM, 2013.

[22] Microsoft Learning to Rank Datasets, 2012. `http://research.microsoft.com/en-us/proj\-ects/mslr/default.aspx`.

[23] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *SIGIR '10*, pages 667–674, 2010.

[24] F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *WSDM '13*, 2013.

[25] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*, pages 43–52, 2008.

[26] F. Scholer, M. Shokouhi, B. Billerbeck, and A. Turpin. Using clicks as implicit judgments: expectations versus observations. In *ECIR'08*, pages 28–39, 2008.

[27] A. Schuth, K. Hofmann, S. Whiteson, and M. de Rijke. Lerot: an online learning to rank framework. In *Living Labs '13*, 2013.

[28] T. Urvoy, F. Clerot, R. Féraud, and S. Naamane. Generic exploration and k-armed voting bandits. In *ICML*, 2013.

[29] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[30] Yandex Internet Mathematics 2009 Dataset, 2009. `http://imat2009.yandex.ru/en/datasets`.

[31] Y. Yue and T. Joachims. Beat the mean bandit. In *ICML*, 2011.

[32] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The K-armed dueling bandits problem. *J. Comp. Syst. Sciences*, 78(5):1538–1556, 2009.

[33] M. Zoghi, S. Whiteson, M. de Rijke, and R. Munos. Relative confidence sampling for efficient on-line ranker evaluation. In *WSDM '14*, 2014.

[34] M. Zoghi, S. Whiteson, R. Munos, and M. de Rijke. Relative upper confidence bound for the k-armed dueling bandits problem. In *ICML*, 2014.