Active Learning for Top-K Rank Aggregation from Noisy Comparisons

Soheil Mohajer [1] **Changho Suh** [2] **Adel Elmahdy** [1]

Abstract

We explore an active top-K ranking problem based on pairwise comparisons that are collected possibly in a sequential manner as per our design choice. We consider two settings: (1) *top-K sorting* in which the goal is to recover the top-K items in order out of n items; (2) *top-K partitioning* where only the set of top-K items is desired. Under a fairly general model which subsumes as special cases various models (e.g., Strong Stochastic Transitivity model, BTL model and uniform noise model), we characterize upper bounds on the sample size required for top-K sorting as well as for top-K partitioning. As a consequence, we demonstrate that active ranking can offer significant multiplicative gains in sample complexity over passive ranking. Depending on the underlying stochastic noise model, such gain varies from around $\frac{\log n}{\log\log n}$ to $\frac{n^2 \log n}{\log\log n}$. We also present an algorithm that is applicable to both settings.

1. Introduction

Ranking is prevalent in a wide variety of applications: social choice (Caplin & Nalebuff, 1991; Azari Soufiani et al., 2013), web search and information retrieval (Dwork et al., 2001), recommendation systems (Baltrunas et al., 2010), and crowd sourcing (Chen et al., 2013), to name a few. The goal of the problem is to bring a consistent ordering to a collection of items, given partial preference information. The two main paradigms among a large volume of works on ranking include spectral ranking algorithms (Negahban et al., 2016; Dwork et al., 2001; Brin & Page, 1998) and maximum likelihood estimation (Ford, 1957). These methods focus on finding the entire ordering, not being tailored for many practical applications in which only a few significant items, say top-K, are often desired to be retrieved.

In an effort to exploit the more practically relevant scenario, (Chen & Suh, 2015) investigated the top-K rank aggregation which aims to recover the correct set of top-ranked items only. It characterized the minimax limit on the sample size (i.e., sample complexity) under the Bradley-Terry-Luce (BTL) model (Bradley & Terry, 1952; Luce, 1959) where pairs of two items are compared. However, this development is limited to a *passive* measurement setting in which pairwise data are simply given prior to analysis.

Many applications of interest often admit interaction with users. This enables us to select comparison pairs of items in an *adaptive* manner. This way of ranking provides the potential to reduce a large number of blindly collected measurements while maintaining a ranking accuracy (Tschopp et al., 2011). This motivates us to examine an *adaptive* measurement setting, in which pairwise comparisons are gathered interacting with a ranker (termed *active ranking*). In particular, we intend to address the following two questions: (a) how much can active ranking offer performance improvements over passive ranking? (b) how does the limit on the sample size for top-K ranking scale with K?

To answer this question, we consider a general model in which the pairwise comparison probabilities are *arbitrary* subject to a mild condition (see (5) in Section 2 for details) and thus which includes as special cases various models like the BTL model, Strong Stochastic Transitivity (SST) model (Fishburn, 1973; Shah et al., 2016), and uniform noise model (Braverman & Mossel, 2008). Two ranking tasks are taken into consideration: (i) top-K sorting which takes care of detailed ordering within top-K items; (ii) top-K partitioning which concerns only the correct set of them.

Contribution. Our contributions are two-folded. The first lies in deriving an upper bound on the sample size:

$$O\left((n + K\log K)\frac{\max(\log\log n, \log K)}{\Delta_K}\right) \tag{1}$$

where[1]

$$\Delta_K = \begin{cases} \Delta_{K,\mathsf{S}} = \min\limits_{i\in[K]} \min\limits_{j:j\geq i} (P_{ij} - 0.5)^2, & \text{Sorting;} \\ \Delta_{K,\mathsf{P}} = (P_{K,K+1} - 0.5)^2, & \text{Partitioning.} \end{cases} \tag{2}$$

[1]ECE, University of Minnesota, Twin Cities, MN, USA. [2]EE, KAIST, Daejeon, South Korea. Correspondence to: Soheil Mohajer <soheil@umn.edu>, Changho Suh <chsuh@kaist.ac.kr>, Adel Elmahdy <adel@umn.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

[1]Note that the notation Δ_K here is defined slightly differently from the one in (Chen & Suh, 2015).

Here $P_{ij} = P_{i,j}$ indicates the probability of item i being preferred over item j, and $\Delta_{K,\mathsf{S}}$ (or $\Delta_{K,\mathsf{P}}$) denotes the parameter w.r.t. top-K sorting (or partitioning). Without loss of generality, we assume that the ground truth ranking is the order of $1 \succ 2 \succ \cdots \succ n$. Notice that the sample complexity bound reads $O(n \log\log n/\Delta_K)$ for the small K regime (e.g, $K = O(\log n)$), and $O(K \log^2 K/\Delta_K)$ for the large K regime (e.g., $K = \Theta(n)$). For the regime $K = O(\log n)$ of practical interest, this exhibits significant multiplicative gains of active ranking over passive ranking. For instance, in the case of top-K sorting, when specializing our result into the uniform noise model and BTL model, one can demonstrate that the factor gains are $\Omega\left(\frac{n^2 \log n}{\log\log n}\right)$ and $\Omega\left(\frac{\log n}{\log\log n}\right)$, respectively. See Table 1 for further details.

Our second contribution is to develop a computationally-efficient (nearly)-linear-time algorithm that can achieve the above bound promised. The algorithm is based on standard algorithms in TCS literature for the *noiseless* sorting and partitioning, where each pairwise order can be retrieved using a single comparison or the transitive property of the ranking. However, in a noisy setting of our interest, single comparison results are not reliable. A key distinction in our work is to employ *repeated pairwise comparisons* for each comparison to combat the noise effect.

Here is how our algorithm works in details. It builds upon a `HEAP` data-structure and can be applied to both settings of sorting and partitioning. The main idea of identifying top-K items in a dataset of n items is to partition the set of n items into K subsets, and then identify the top item in each subset using single-elimination tournament. Next, a max-`HEAP` is built to single out the top-1 item among the K winners of the tournaments. After that, this item is removed from the system and the max-`HEAP` is updated by replacing it by the second top item of the subset this top item belongs to. We repeat this process until we identify top-K items. See Fig. 2 for an explanatory example.

It is worth mentioning that the performance of our algorithm depends on the separability parameter Δ_K. The algorithm is supposed to correctly distinguish between items K and $(K+1)$ (in partitioning), or the top K items and the rest (in sorting). The smaller Δ_K, the less reliable the results of comparison between to-be-distinguished items; hence, more repetitions are needed for a decision. We characterize the minimum number of repeated comparisons required to ensure that the retrieved items are the correct winners. The carefully chosen number for repeated comparisons together with a couple of bounding techniques play a key role to derive the above sample complexity bound. Finally, we conduct several experiments to corroborate our main results.

Related work. (Chen & Suh, 2015) explored the top-K partitioning problem under the *non-adaptive* comparison model, and characterized the optimal sample complexity. Subsequently, sample complexity analyses were made with regard to different yet popular ranking paradigms such as simple counting methods (Shah & Wainwright, 2016) and spectral methods (Jang et al., 2016) (e.g., `RankCentrality` (Negahban et al., 2016)). In this work, we examine an *adaptive* measurement setting under a fairly general model, thereby showing that active ranking can significantly outperform passive ranking for a variety of scenarios.

Recently, (Braverman et al., 2016) developed an active ranking algorithm. Interesingly, for the $K = 1$ case and under the uniform noise model, their algorithm can achieve the same sample complexity as ours for a certain target error rate. (Szörényi et al., 2015) also focused on the $K = 1$ case but under the BTL model, thus developing an algorithm which however yields a larger sample complexity than ours. Most recently, (Heckel et al., 2016) proposed an algorithm for a general problem setting which encompasses top-K sorting and partitioning of our interest. We found that their algorithm is outperformed by ours when specializing it to our settings. See 3.2 for detailed discussion.

There has been a proliferation of active ranking algorithms (Busa-Fekete et al., 2014; Jamieson & Nowak, 2011; Maystre & Grossglauser, 2015; Ailon, 2012; Braverman & Mossel, 2008; Wauthier et al., 2013). While interesting ranking schemes are developed for perfect ranking (Busa-Fekete et al., 2014; Jamieson & Nowak, 2011; Maystre & Grossglauser, 2015) and approximate ranking (Jamieson & Nowak, 2011; Ailon, 2012; Braverman & Mossel, 2008; Wauthier et al., 2013), they are not customized for top-K ranking of our interest.

Moreover, the best-K identification with adaptive sampling has been extensively explored under the name of the multi-armed bandit problem (Gabillon et al., 2011; Bubeck et al., 2013; Jamieson et al., 2014; Yue et al., 2012) for a so-called value-based model in which the observation on each item is drawn only from the distribution underlying this individual. Also there are many related yet different problem settings considered in prior literature (Azari Soufiani et al., 2013; Hajek et al., 2014; Lu & Boutilier, 2011; Eriksson, 2013).

2. Problem Formulation

Comparison model. We denote by $\mathcal{G} = ([n], \mathcal{E})$ a comparison graph in which items i and j are compared if and only if (i, j) belongs to the edge set $\mathcal{E}$. More precisely, a *multi-edge* graph is taken into consideration to accommodate repeated measurements for an observed pair. We take into account an adaptive comparison graph in which the edge set is dynamically selected interacting with a ranker. Specifically, for a sample instance $t \in [1 : S]$ where S indicates the total sample size, an edge $e_t = (i_t, j_t)$ is chosen based on the pairwise outcomes obtained up to $(t-1)$.

Pairwise comparisons. Given $e_t = (i, j)$, the outcome of the t^{th} comparison, denoted by Y_t, is generated according to

$$Y_t = \begin{cases} 1 & \text{with probability } P_{ij} \\ 0 & \text{with probability } 1 - P_{ij}, \end{cases} \tag{3}$$

where $Y_t = 1$ indicates that item i is preferred over item j. The outcomes Y_t's are independent across t. We also represent the collection of sufficient statistics as

$$Y_{ij} := \sum_{t: e_t = (i,j), e_t \in \mathcal{E}} Y_t; \quad \boldsymbol{Y} := \{Y_{ij} : (i, j) \in \mathcal{E}\}. \tag{4}$$

Sorted-by-Probabilities (SP) model. Without loss of generality, assume that the ground truth ranking is the order of $1 \succ 2 \succ \cdots \succ n$. In fact, the SST model (Fishburn, 1973; Shah et al., 2016) suggests one way to relate the ranking to the model parameters P_{ij}'s by putting the following constraint[2]: $P_{ik} > P_{jk}$ for all $k \neq \{i, j\}$ whenever item $i \succ$ item j. In this work, we introduce a more general model, which we call *Sorted-by-Probabilities (SP)* model, by relaxing the constraint as:

$$P_{ij} > \frac{1}{2} \qquad \text{whenever } i \succ j. \tag{5}$$

Notice that the new constraint (5) is weaker, thus spanning a larger parameter space. One can readily verify that our model also subsumes as special cases other prominent models. Observe that whenever $i \succ j$,

$$P_{ij} = \begin{cases} \frac{1}{2} + \gamma > \frac{1}{2}, & \text{(uniform noise model);} \\ \frac{w_i}{w_i + w_j} > \frac{1}{2}, & \text{(BTL model),} \end{cases} \tag{6}$$

where γ denotes an arbitrary constant $\in (0, 0.5)$ and w_i indicates the score of item i.

Performance metric and goal. Given the pairwise comparisons, one wishes to know whether or not the top-K ordered items (or the top-K set) are identifiable. In light of this, we consider the probability of error P_{e}:

$$P_e(\psi) = \begin{cases} \mathbb{P}\left\{\psi(\boldsymbol{Y}) \neq (1 \succ \cdots \succ K)\right\}, & \text{sorting;} \\ \mathbb{P}\left\{\psi(\boldsymbol{Y}) \neq [K]\right\}, & \text{partitioning,} \end{cases}$$

where ψ is any ranking scheme that returns an order of K indices. Our goal in this work is to characterize the *sample complexity* S_K^*, defined as the minimum sample size above which top-K ranking is feasible, in other words, P_e can be vanishingly small as n grows.

Remark 1 *It should be noted that there are other frameworks in which different performance metrics are taken into consideration: "regret" (Yue et al., 2012), and "PAC-learning" (Szörényi et al., 2015), both introduced in the bandit literature. Actually our work focuses on the worst case scenario as the "error rate (0/1 loss)" that we considered is the most stringent criterion among others. Hence, the sample complexity of our model provides an upper bound for other criteria. Interestingly, the sample complexity of the proposed algorithm is superior (lower) to that of the PLPAC algorithm (Szörényi et al., 2015) under the PAC criterion. See Section 3.2 for details.*

3. Main Results

As noted in the passive ranking setup (Chen & Suh, 2015), the most crucial part of top-K partitioning under the BTL model hinges on separating the two items near the boundary, being reflected in $\left(\frac{w_K - w_{K+1}}{w_K + w_{K+1}}\right)^2$. Similarly for top-$K$ sorting, one can easily show that the key measure would be: $\min_{i \in [K]} \left(\frac{w_i - w_{i+1}}{w_i + w_{i+1}}\right)^2$. We find that in our general model, the corresponding key measure is the one defined in (2). Observe in (2) that $P_{ij} - 0.5 = \frac{w_i - w_j}{2(w_i + w_j)}$ under the BTL model. Hence, we will use this measure to express our upper bound on sample complexity as below.

Theorem 1 *With probability exceeding $1 - (\log n)^{-c_0}$, the top-K order (or top-K set[3]) can be identified provided that*

$$S_K \geq c_1 (n + K \log K) \frac{\max(\log \log n, \log K)}{\Delta_K}. \tag{7}$$

Here, (c_0, c_1) are some universal positive constants.

See Section 4 for the proof of Theorem 1 and algorithm description. There are three points to make. The first is that the above bound is w.r.t. a target error rate that scales like $\frac{1}{\text{poly}(\log n)}$. Aiming at a smaller target error, we need a larger sample size. Secondly, the term Δ_K, affected by $(P_{ij} - 0.5)$ (see (2)), captures how noisy the comparison data is, i.e., $(P_{ij} - 0.5)$ is a sort of the difficulty level of separating item i from item j. So the result in Theorem 1 coincides with our intuition because smaller Δ_K means more difficult to rank, which results in an increase of sample complexity. The last point is regarding the performance of our algorithm. Since sorting naturally produces a partitioning, our algorithm is tailored for the sorting, and hence favors the sorting performance relative to partitioning. Notice for partitioning that when $K = n$, the sample complexity bound reads the order of $n(\log n)^2$, which is certainly far from optimality. Hence, in the next subsection, we provide several interesting remarks with an emphasis on top-K sorting in which we advocate the performance of our algorithm.

[2] We ignore the tie situation as we consider a strict order of ranking. Precisely speaking, the constraint is called Strict Strong Stochastic Transitivity (SSST) property (Fishburn, 1973).

[3] For top-K partitioning, we assume monotonicity in P_{ij}: $P_{ij} \geq P_{ik}$ whenever $i \leq j \leq k$, which holds still under a fairly general model like SST.

3.1. Top-K Sorting

Penalty due to noisy measurements: As mentioned earlier, top-K sorting has been extensively explored in the TCS literature, but only the noiseless setting has been the main focus, in which sample complexity is characterized as (Cormen et al., 2009)

$$S_{\mathsf{noiseless},K} = \Theta(n + K\log K). \tag{8}$$

Comparing (8) to (7), we see that the penalty factor in sample complexity due to noisy measurements is:

$$O\left(\frac{\max(\log\log n, \log K)}{\Delta_K}\right). \tag{9}$$

Actually it is not clear whether or not this penalty factor is *fundamental* due to the lack of the optimality result. However, the gap, if any, is up to $\mathrm{poly}(\log n)$, as long as Δ_K is not too small (scales at most with $\mathrm{poly}(\log n)$). This implies that the degradation over the noiseless setting is low and therefore our algorithm performs very close to optimum.

How the limit scales with K: Observe in (7) that:

$$S_K = \begin{cases} O\left(\frac{n\log\log n}{\Delta_K}\right), & K = O(\log n); \\ O\left(\frac{K\log^2 K}{\Delta_K}\right), & K = \Theta(n). \end{cases}$$

We make one interesting observation in the $K = O(\log n)$ regime of practical interest: the bound is irrelevant to K under some measurement model. One such example is the uniform noise model where $\Delta_K = \gamma^2$:

$$S_{\mathsf{uniform},K} = O\left(\frac{n\log\log n}{\gamma^2}\right), \tag{10}$$

for every K. However, this phenomenon does not carry over to other noisy models, like the BTL model in which the noise quality varies according to associated preference scores. Note that

$$S_{\mathsf{BTL},K} = O\left(\frac{n\log\log n}{\min_{i\in[K]}\left(\frac{w_i - w_{i+1}}{w_i + w_{i+1}}\right)^2}\right). \tag{11}$$

But our result still suggests that the phenomenon may hold universally for a variety of statistical models as long as K is small enough.

Active vs. passive ranking: For illustrative purpose, let us focus on the interesting regime of $K = O(\log n)$, and consider two models: (1) uniform noise model; (2) BTL model. In the uniform noise model, Shah-Wainwright (Shah & Wainwright, 2016) characterized the passive ranking sample complexity for a certain observation model:

$$S^{\mathsf{passive}}_{\mathsf{uniform},K} = \Theta\left(\frac{n^3\log n}{\gamma^2}\right), \tag{12}$$

for every choice of K. This together with (10) demonstrates that the factor gain due to active measurements is quite substantial: $\Omega\left(\frac{n^2\log n}{\log\log n}\right)$. In the BTL model, Chen-Suh (Chen & Suh, 2015) characterized the sample complexity under passive ranking as:

$$S^{\mathsf{passive}}_{\mathsf{BTL},K} = O\left(\frac{n\log n}{\min_{i\in[K]}\left(\frac{w_i - w_{i+1}}{w_i + w_{i+1}}\right)^2}\right). \tag{13}$$

Comparing this to (11), we see that the factor gain is $\Omega\left(\frac{\log n}{\log\log n}\right)$, which is not quite significant but still scales with n and hence exhibits respectful improvements in high dimensional regimes. The comparisons are summarized in Table 1. See Section 5 for experimental results on this.

Robustness: Theorem 1 suggests that the performance gap between sorting and partitioning is not significant. For instance, under the uniform noise model, Δ_K's are the same, yielding the same sample complexity bound. For the BTL model, Δ_K's would be similar if w_i's are equidistant. Experimental results on this are provided in the supplemental.

Computational complexity: A noticeable feature of our algorithm is its low computational complexity. It runs in time $O(n)$ in the practically-relevant regime of $K = O(\log n)$. For general K, it is nearly linear in n, i.e., $O(n + K\log K)$. Here, this complexity assumes that the input fed to our algorithm is the sufficient statistic of the outcome comparisons: Y_{ij} (see (4)), rather than the entire collection of Y_t's associated with the pair (i, j). This will be evident later when describing the algorithm.

3.2. Comparison to Related Work

(Braverman et al., 2016) developed an active ranking algorithm for the $K = 1$ case and derived the order-wise tight sample complexity in terms of target error rate under the uniform noise model. More concretely, suppose we want the error probability not to exceed a target error rate δ. Then, the result of (Braverman et al., 2016) implies

$$S_{1,\delta} = \Theta\left(\frac{n\log(1/\delta)}{\gamma^2}\right). \tag{14}$$

Notice that our result (7) admits the target error rate that scales like $\frac{1}{\log n}$, implying that their algorithm can achieve the same sample complexity as ours for a certain scenario in which $\delta \le \frac{1}{\log n}$ (see Section 5 for experimental results). For a relaxed target error like $\frac{1}{\log\log n}$, their algorithm achieves a slightly smaller sample complexity by a factor of $\frac{\log\log n}{\log\log\log n}$.

(Szörényi et al., 2015) also developed a top-selection algorithm and analyzed sample complexity under the BTL model as well as a less-stringent PAC criterion. Their sample complexity bound reads around $O(n\log n)$, thus yields a larger one compared to ours.

In another relevant paper, (Heckel et al., 2016) proposed an active ranking algorithm for a general setting, which subsumes top-K sorting (as well as top-K partitioning)

	Noise Model	Active Measurement	Passive Measurement	Gain
Uniform Noise	$P_{ij} = 0.5 + \gamma(-1)^{\mathbb{1}\{i \prec j\}}$	$O\left(\frac{n \log\log n}{\gamma^2}\right)$	$\Theta\left(\frac{n^3 \log n}{\gamma^2}\right)$	$\Omega\left(\frac{n^2 \log n}{\log\log n}\right)$
BTL model	$P_{ij} = \frac{w_i}{w_i + w_j}$	$O\left(\frac{n \log\log n}{\min_{i\in[K]}\left(\frac{w_i - w_{i+1}}{w_i + w_{i+1}}\right)^2}\right)$	$\Theta\left(\frac{n \log n}{\min_{i\in[K]}\left(\frac{w_i - w_{i+1}}{w_i + w_{i+1}}\right)^2}\right)$	$\Omega\left(\frac{\log n}{\log\log n}\right)$

Table 1. Top-K sorting: Multiplicative gains of active ranking (this work) over passive ranking for the uniform noise model (Shah & Wainwright, 2016) and BTL model (Chen & Suh, 2015) in the practically relevant regime $K = O(\log n)$.

as a special case. They also provided a lower bound on the sample complexity for a class of parametric models, which is only $O(\log n)$ away from the achievable sample complexity. It is worth mentioning that under the uniform noise model, the algorithm in (Heckel et al., 2016) requires $O\left(\frac{n^2 \log(1/\delta)}{\gamma^2}\right)$ pairwise comparisons to achieve an error rate of δ, which is very expensive compared to our algorithm. Here a key to note is that the uniform noise model does not fit to the class of parametric models considered by (Heckel et al., 2016), in which the CDF function Φ in $P_{ij} = \Phi(w_i - w_j)$ is assumed to be differentiable, which does not hold in the uniform noise model where $\Phi(t) = \frac{1}{2} + \gamma\,\mathrm{sign}(t)$.

On the other hand, applying the result of (Heckel et al., 2016) on the BTL model, we get lower and upper bounds on the sample complexity. To see this in details, consider a special case of the top-K partitioning problem, where $w_1 = w_2 = \cdots = w_K$, and $w_{K+1} = w_{K+2} = \cdots = w_n$. In this case, (Heckel et al., 2016) implies

$$c_{\mathrm{l}} \frac{n}{\Delta_K} \log\left(\frac{1}{2\delta}\right) \le S_{\mathsf{BTL},K} \le c_{\mathrm{u}} \frac{n}{\Delta_K} \log\left(\frac{n}{\delta}\right) \log\log\left(\frac{4}{\Delta_K}\right)$$

where $c_{\mathrm{l}} = 1/16$ and $c_{\mathrm{u}} \approx 654$. Notice that the asymptotic multiplicative gap between the lower and upper bounds are on the order of $\log(n)\log\log(1/\Delta_K)$. Moreover, the large constant-factor gap yields a significant performance gap in the actual experiment. For instance, see Fig. 3(b) where $n = 1024$, $\Delta_K = 0.0225$, and $\delta = 0.1$. Observe that the lower and upper bounds are 6.6×10^3 and 4.5×10^8 comparisons, respectively, which are far apart.

4. Proposed Ranking Algorithm

In this section we present our algorithms for sorting and partitioning tasks, and provide upper bounds for the sample complexity. We use $\mathcal{N}(\cdot)$ to denote the number of pairwise comparisons required in the algorithm.

4.1. Top-1 Selection: Single-Elimination Tournament

We first focus on the special case of identifying the top item, i.e., $K = 1$. The proposed algorithm is essentially a customized *single-elimination tournament*, which consists of multiple layers. In each layer items are paired in a random manner, and one of the items in each pair is selected to proceed to the next layer, while the other one is eliminated. This decision is made based on pairwise comparisons between the two items. The distinctive feature relative to conventional single-elimination tournament is that in order to combat against the uncertainty of the observations, we repeat each binary comparison multiple (say m) times. It turns out that for a sufficiently large m, the algorithm will output the index of the top item with overwhelming probability.

Algorithm 1 SELECT$(X; m)$

Input: m
Data: $X = \{x[1], x[2], \ldots, x[n]\}$.
Output: a^*: the index of item with the highest score.
(Assume $|X|$ is a power of 2 for simplicity)
$n \leftarrow |X|$
for $i \leftarrow 1$ to $|X|$ **do** $a(i) \leftarrow i$ **end for**
for $\ell \leftarrow 1$ **to** $\log n$ **do**
  for $i \leftarrow 1$ **to** $n/2^\ell$ **do**
    $T \leftarrow 0$
    for $t \leftarrow 1$ **to** m **do**
      $e_t \leftarrow (a(2i-1), a(2i))$
      $T \leftarrow T + Y_t$ (Y_t is defined in Eq. 3)
    end for
    if $T \ge \frac{m}{2}$ **then** $a(i) \leftarrow a(2i-1)$
    else $a(i) \leftarrow a(2i)$
    end if
  end for
end for
$a^* \leftarrow a(1)$

The algorithm builds a binary tree of depth $\lceil \log |X| \rceil$ (see Fig. 1). We denote the i-th item index in layer ℓ by $x_{\ell,i}$. Initially, we randomly locate items on the leaves of the tree, that are denoted by $x_{1,i}$ for $i = 1, \ldots, n$. Then, in each iteration, a pair in layer ℓ is tested, and the winner will proceed to layer $(\ell + 1)$. Hence, half of the existing items will be eliminated in each iteration, until we get to the *root* layer in which there would be only one surviving item. The algorithm is formally presented in Algorithm 1.

Number of measurements: The algorithm consists of $\lceil \log |X| \rceil$ layers, and $|X|/2^\ell$ pairs are being tested in layer ℓ.

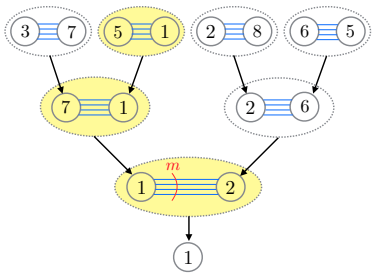


Figure 1. The single-elimination tournament with repeated comparisons for $n = 8$ items. In each round (layer), items are paired. A decision between a pair of items is made based on the majority rule among m comparisons. Note that a wrong decision in a pair that does not include item 1 (e.g., when 7 beats 3 in spite of $3 \succ 7$) will not affect the ultimate output of the algorithm.

Hence, the total number of tests is $\sum_{\ell=1}^{\lceil \log |X| \rceil} n2^{-\ell} \leq |X|$. Each test requires m binary comparisons, yielding a total number of measurements given by

$$\mathcal{N}(\texttt{SELECT}(X; m)) = O(m|X|) = O(mn), \tag{15}$$

for a dataset of size $|X| = n$. It can be shown that if

$$m \geq \frac{(1+\varepsilon)\ln 2}{2} \frac{\log\log |X|}{\Delta_{1,\mathsf{S}}}, \tag{16}$$

then $\mathbb{P}_e(\mathsf{TopSelection}) \leq (\log n)^{-\varepsilon}$ for any $\varepsilon > 0$. This implies that, with high probability, the SELECT algorithm can successfully select the top item using a total of

$$\mathcal{N}(\texttt{SELECT}(X; m)) = O\Big(\frac{|X| \log\log |X|}{\Delta_1}\Big) \tag{17}$$

pairwise comparisons.

Remark 2 *Parallel to this work, the top-selection problem is studied in (Braverman et al., 2016) under the uniform noise model. The algorithm in (Braverman et al., 2016) first identifies the top item in a subset of size $n/\log n$ items, and then iteratively refines the estimate by further comparing that to other items in the set. It is shown that the number of measurements required grows linearly with n, when a* constant (non-vanishing) *error probability is desired. However, in order to achieve a vanishing error probability scaling as $1/poly(\log n)$, the algorithm of (Braverman et al., 2016) requires the same number (up to a constant factor) of pairwise comparisons, as the one presented above.*

4.2. Top-K Sorting: A Heap-based Algorithm

In this section we generalize our proposed algorithm to find the top K items along with their order. The proposed algorithm is built based on the single-elimination algorithm, which can find the the single top item with high probability. A trivial generalization is to repeat the SELECT algorithm for K times, which requires a large number of comparisons

Algorithm 2 $\texttt{TOP}(X; K, m)$

Input: Integers K and m
Data: Array $X = \{x[1], x[2], \ldots, x[n]\}$.
Output: Indices of top-K: $\pi(1), \pi(2), \ldots, \pi(K)$
$n \leftarrow |X|$, $Q \leftarrow \lceil n/K \rceil$
for $i \leftarrow 0$ **to** $K-1$ **do**
  $C_{i+1} \leftarrow \{x[iQ+1], x[iQ+2], \ldots, x[\min\{(i+1)Q, n\}]\}$
  $b(i+1) \leftarrow iQ + \texttt{SELECT}(C_{i+1}; m)$
end for
$Z \leftarrow \{b(1), b(2), \ldots, b(K)\}$
$\texttt{BuildHeap}(Z, X; m)$
for $i \leftarrow 1$ **to** K **do**
  $\pi(i) \leftarrow Z[1]$
  $j \leftarrow \lfloor Z[1]/K \rfloor$
  $C_j \leftarrow C_j \setminus \{x[Z[1]]\}$
  $Z[1] \leftarrow \texttt{SELECT}(C_i; m)$
  $\texttt{Heapify}(Z, X, 1; m)$
end for

when K scales with $n = |X|$. Rather, we first split the dataset X of n items into K groups each of size n/K, namely groups $C_1, C_2, \ldots, C_K$. Then we identify the top item in each sub-group using SELECT, and form a short list that includes all winners from the sub-groups. Then we build a (max-)HEAP data-structure for the short list obtained from the K winners. The HEAP structure allows us to easily extract the top item from the short list. Once the top item of the short list is identified and removed from the list, we go back to its home sub-group, identify the second top item in that sub-group, and insert it to the short list. We maintain the HEAP structure of the short list during the process, to be able to easily extract the *next* top item of the list. We repeat this procedure for $(K-1)$ rounds until we retrieve all the top K items. The main algorithm, TOP is presented in Algorithm 2. For the sake of completeness, we also present the algorithms required to build the heap structure and to insert a new item to an existing heap in the supplemental.

Sample complexity: The sample complexity of the algorithm can be simply evaluated in terms of the input parameters as follows. We first identify the top entry of each of the K sub-groups. Later, during the iterative phase of the algorithm, we need to repeat SELECT algorithm on the remaining elements of sub-groups for another K iterations, which results in $2K$ runs of SELECT, each on subgroups of size at most n/K items. Each run of algorithm SELECT with parameter m on a dataset C_i requires $\mathcal{N}[\texttt{SELECT}(C_i; m)]$ pairwise comparison, as given in (16).

In order to build the heap structure on K sub-winners we need to make $O(K)$ binary decisions, where we repeat each comparison m times to deal with the noise. Moreover, in each iteration one new item is added to the short list. We need to make $O(\log K)$ binary decisions to maintain the

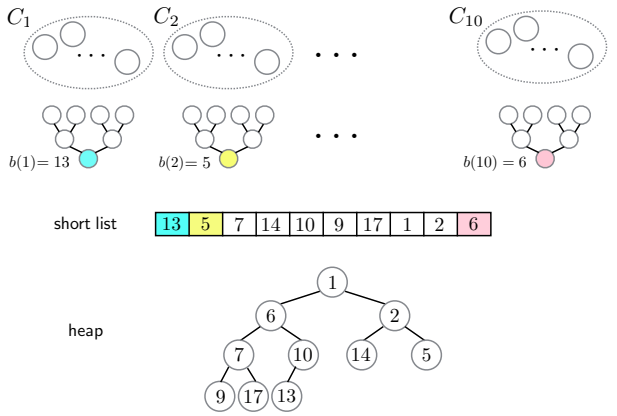


Figure 2. Sorting algorithm for $K = 10$ using heap data structure. Items are split into $K = 10$ groups of equal size, namely $C_1, C_2, \ldots, C_{10}$. The top items of each sub-groups is identified using single-elimination tournament. A heap data-structure is then built for the short list of the winners of the tournaments. This provides a binary tree, with the property that children of each node have a rank lower than their parent. Then the root will be reported as the top item. Next, one goes back to the home-subgroup of the root, to find the second-top item of that group. The top item found in the heap will be replaced by the second top item, and heap will be re-arranged to maintain its property. Iterating on this for $(K-1) = 9$ times, one can identify items $1, 2, \ldots, 10$.

heap structure. Similar to `BuildHeap`, we repeat each comparison for m times, and then decide based on a majority rule. Therefore, we have

$$\begin{aligned}\mathcal{N}[\texttt{TOP}(X,K;m)] &\triangleq 2K\cdot\mathcal{N}[\texttt{SELECT}(C_i;m)]\\ &+\mathcal{N}[\texttt{BuildHeap}(Z;m)]+K\mathcal{N}[\texttt{InsertHeap}(Z;m)]\\ &= 2K\cdot O(mn/K)+m\cdot O(K)+m\cdot O(K\log K)\\ &= O(mn+mK\log K). \qquad (18)\end{aligned}$$

It turns out, from the analysis of probability of error, that the proposed algorithm can successfully sort the top K items if

$$m = O\left(\frac{\max\{\log K, \log\log n\}}{\Delta_K}\right).$$

Plugging this into (18), we can find the sample complexity of the `TOP` algorithm as

$$S_K = O\left(\frac{(n+K\log K)\max\{\log K, \log\log n\}}{\Delta_K}\right).$$

Remark 3 *The repetition parameter used in `SELECT` algorithm, and the one used in `HEAP` can be potentially different, and accordingly optimized. However, our analysis shows that a choice of distinct parameters can provide a marginal gain only for a tiny range of K. Thus, we rather choose the same parameter for the sake of simplicity.*

4.3. Partitioning

The algorithm we propose for *partitioning* is exactly identical to that of the sorting. We (randomly) split the items into K groups, and run the single-elimination tournament in each group. The winners will proceed to the `HEAP` algorithm, and then the `HEAP` outputs items one by one. The only difference lies in the performance metric which concerns only the *set* of K items reported by the algorithm, regardless of the order. It turns our that a similar analysis holds for this problem, except the fact that the number of repeated comparisons, m, has a weaker dependency on the data. More precisely, the algorithm is robust against wrong binary decision between two items from $[K]$, and similarly between two items from $X \setminus [K]$. One needs to choose m such that the algorithm (with high probability) can correctly distinguish when $s \in [K]$ is compared against $q \in X \setminus [K]$. Consequently, we show that m depends only on $\Delta_{K,\mathsf{P}}$, which captures the gap between $P_{K,K+1}$ and $1/2$.

Remark 4 *Note that the proposed algorithms do not require the knowledge of Δ_K, and can be executed with any m. It depends only on the measurement budget. However, dependency of the performance of the algorithms on Δ_K is inevitable, since the success rate depends on the separability parameter as discussed earlier.*

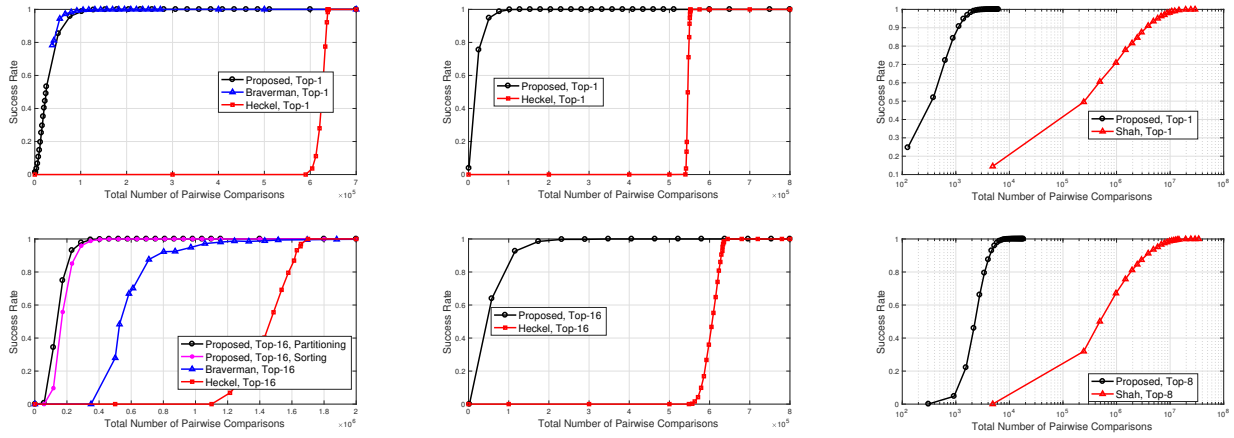
5. Simulation Results

In this section, we empirically evaluate the performance of the proposed algorithm by conducting Monte Carlo simulations on synthetic data and developing a benchmark comparison against the state-of-the-art active and passive ranking algorithms in the literature. In an effort to guarantee fairness in the performance comparison, we jointly investigate the average number of pairwise comparisons and the corresponding empirical success rate of identifying top-K items. The source code of our algorithm[4] is provided to allow for reproducible research. In addition, we present a more detailed discussion on the performance of our algorithm under various simulation parameters in the supplemental.

5.1. Comparison to Prior Active Ranking Algorithms

We assess the performance of the proposed algorithm against two recent active ranking algorithms; the first is proposed by (Braverman et al., 2016), coined "Braverman", while the second algorithm is proposed by (Heckel et al., 2016), coined "Heckel". For Braverman algorithm, we sweep over an algorithm parameter (denoted by c in the paper), such that $c \geq 10$, to measure the corresponding success rate. Heckel algorithm employs a confidence interval during the process, and we select the following empirical confidence interval model, that is a function of the algorithm round, t, and the tolerance parameter $\delta \in (0, 0.14]$: $\alpha_t = \sqrt{\log(n/3(\log(t)+1)/\delta)/16t}$.

[4]The source code is accessible via GitHub at https://github.com/a-elmahdy/Active-Learning-from-Noisy-Comparisons.git



(a) Uniform noise model. (b) BTL model. (c) Active vs. Passive algorithms.

Figure 3. Performance Evaluation of various active and passive ranking algorithms: (a) Active algorithms under the uniform noise model for different values of K when $n = 1024$, $\Delta_{K,\mathrm{S}} = \Delta_{K,\mathrm{P}} = \gamma^2 = 0.0225$, and $\delta = 0.1$ (Heckel algorithm parameter); (b) Active algorithms under BTL model for different values of K when $n = 1024$, $\Delta_K = 0.0225$, $w_i \in (0, 3]$, $1 \leq i \leq n$, and $\delta = 0.1$ (Heckel algorithm parameter); (c) Performance Comparison between the proposed Top-K algorithm versus the Copeland Counting algorithm (with $\alpha = 8$ and $p = 0.6$) under the uniform noise model for different values of K when $n = 128$, $\Delta_K = \gamma^2 = 0.1$.

The plots in Fig. 3(a) indicate the performance of different active ranking algorithms under the uniform noise model to identify the top-1 and top-16 items, respectively, when $n = 2^{10} = 1024$, $\Delta_K = \gamma^2 = 0.0225$ and $\delta = 0.1$. The first plot shows slight improvement in the total number of pairwise comparisons required to achieve a target success rate for Braverman algorithm over our Top-1 algorithm. We also observe that the performance gap between the two algorithms is negligible at a higher success rate. On the other hand, the second plot in Fig. 3(a) depicts the significant improvement of our Top-K algorithm over Braverman algorithm[5] when $K = 16$. It is evident that the sample complexity of Braverman algorithm scales with K when it is employed for top-K identification. We can also see that the performance gap between Partitioning and Sorting is insignificant. These findings are consistent with our analysis of the sample complexity of both algorithms. Hence, the two prime merits of the proposed Top-K algorithm are: (1) its superior performance compared to Braverman algorithm when $K > 1$; and (2) error and sample complexity of Braverman algorithm is limited to uniform noise model, while our algorithm provably performs well for a wide class of pairwise comparison models, defined by (5). We refer the reader to the supplemental for an insightful remark about our algorithm.

[5]Braverman algorithm is extended to top-K ranking. We first run the algorithm to retrieve the top-item. Then, we remove it from the set of items and rerun the algorithm to find the second top item. We keep doing that until we find the top-K items.

The plots in Fig. 3(a) also depict performance comparison between our and Heckel algorithm for Top-K ranking under the uniform noise model. As it is clear from the figure, the proposed algorithm in this work performs significantly better than Heckel algorithm for this noise model. Furthermore, Fig. 3(b) shows that our algorithm for identifying top-1 and top-16 ranked items also achieves a better performance compared to Heckel algorithm under the BTL model.

5.2. Active vs. Passive Ranking

We compare the performance of our active ranking algorithm against a passive ranking algorithm proposed by (Shah & Wainwright, 2016), coined "Shah". This simple yet robust algorithm hinges on Copeland counting algorithm that recovers the top-K items that win the maximum number of pairwise comparisons. Moreover, the algorithm makes no assumptions on the probability model of pairwise comparisons. The algorithm parameters are the probability of making a comparison on any trial, p, the number of trials, r, and $\alpha \geq 8$. Note that the number of noisy comparisons associated with each pair of items follows a binomial distribution with parameters p and r.

The plots in Fig. 3(c) illustrate the performance of the two algorithms under the uniform noise model to recover the top-1 and top-8, respectively, when $n = 2^7 = 128$, $\Delta_K = \gamma^2 = 0.1$, $\alpha = 8$ and $p = 0.6$. As predicted by our analysis of the sample complexity gain, we can observe the considerable gain due to active measurements, even when the total number of items is modest.

Acknowledgment

The authors would like to thank the reviewers who gave useful comments. C. Suh was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP; Ministry of Science, ICT & Future Planning) (No. 2015R1C1A1A02036561).

References

Ailon, N. Active learning ranking from pairwise preferences with almost optimal query complexity. *Journal of Machine Learning*, 13:137–164, 2012.

Azari Soufiani, H., Chen, W., Parkes, D. C., and Xia, L. Generalized method-of-moments for rank aggregation. In *Neural Information Processing Systems*, pp. 2706–2714, 2013.

Baltrunas, L., Makcinskas, T., and Ricci, F. Group recommendations with rank aggregation and collaborative filtering. In *ACM Conference on Recommender Systems*, pp. 119–126. ACM, 2010.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3-4):324–345, 1952.

Braverman, M. and Mossel, E. Noisy sorting without resampling. In *ACM-SIAM symposium on Discrete algorithms*, pp. 268–276, 2008.

Braverman, M., Mao, J., and Weinberg, S. M. Parallel algorithms for select and partition with noisy comparisons. *STOC*, 2016.

Brin, S. and Page, L. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN systems*, 30(1):107–117, 1998.

Bubeck, S., Wang, T., and Viswanathan, N. Multiple identification in multi-armed bandits. In *International Conference on Machine Learning*, 2013.

Busa-Fekete, R., Hüllermeier, E., and Szörényi, B. Preference-based rank elicitation using statistical models: The case of mallows. In *International Conference on Machine Learning*, pp. 1071–1079, 2014.

Caplin, A. and Nalebuff, B. Aggregation and social choice: a mean voter theorem. *Econometrica*, pp. 1–23, 1991.

Chen, X., Bennett, P. N., Collins-Thompson, K., and Horvitz, E. Pairwise ranking aggregation in a crowd-sourced setting. In *ACM Conference on Web Search and Data Mining*, pp. 193–202. ACM, 2013.

Chen, Y. and Suh, C. Spectral MLE: Top-K rank aggregation from pairwise comparisons. In *International Conference on Machine Learning*, pp. 371–380, 2015.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. MIT press, 3$^{\mathbf{rd}}$ edition, 2009.

Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. Rank aggregation methods for the web. In *International conference on World Wide Web*, pp. 613–622. ACM, 2001.

Eriksson, B. Learning to top-K search using pairwise comparisons. In *International Conference on Artificial Intelligence and Statistics*, pp. 265–273, 2013.

Fishburn, P.C. Binary choice probabilites: on the varieties of stochastic transitivity. *Journal of Mathematical Psychology*, 10:327–352, 1973.

Ford, L. R. Solution of a ranking problem from binary comparisons. *American Mathematical Monthly*, pp. 28–33, 1957.

Gabillon, V., Ghavamzadeh, M., Lazaric, A., and Bubeck, S. Multi-bandit best arm identification. In *Neural Information Processing Systems*, 2011.

Hajek, B., Oh, S., and Xu, J. Minimax-optimal inference from partial rankings. In *Neural Information Processing Systems*, pp. 1475–1483, 2014.

Heckel, R., Shah, N., Ramchandran, K., and Wainwright, M. Active ranking from pairwise comparisons and when parametric assumptions don't help. *arXiv:1606.08842*, 2016.

Jamieson, K., Malloy, M., Nowak, R., and Bubeck, S. li-iUCB: An optimal exploration algorithm for multi-armed bandits. In *Conference on Learning Theory*, 2014.

Jamieson, K. G. and Nowak, R. Active ranking using pairwise comparisons. In *Neural Information Processing Systems*, pp. 2240–2248, 2011.

Jang, M., Kim, S, Suh, C, and Oh, S. Top-k ranking from pairwise comparisons: When spectral ranking is optimal. *arXiv preprint arXiv:1603.04153*, 2016.

Lu, T. and Boutilier, C. Learning Mallows models with pairwise preferences. In *International Conference on Machine Learning*, pp. 145–152, 2011.

Luce, R. D. *Individual choice behavior: A theoretical analysis*. Wiley, 1959.

Maystre, L. and Grossglauser, M. Robust active ranking from sparse noisy comparisons. In *arXiv:1502.05556*, 2015.

Negahban, S., Oh, S., and Shah, D. Rank centrality: Ranking from pairwise comparisons. *Operations Research*, 2016.

Shah, N. B. and Wainwright, M. J. Simple, robust and optimal ranking from pairwise comparisons. 2016. URL http://arxiv.org/abs/1512.08949.

Shah, N. B, Balakrishnan, S., Guntuboyina, A., and Wainwright, M. J. Stochastically transitive models for pairwise comparisons: Statistical and computational issues. *International Conference on Machine Learning*, 2016.

Szörényi, B., Busa-Fekete, R., Paul, A., and Hüllermeier, E. Online rank elicitation for Plackett-Luce: A dueling bandits approach. In *Neural Information Processing Systems*, 2015.

Tschopp, D., Diggavi, S., Delgosha, P., and Mohajer, S. Randomized algorithms for comparison-based search. In *Neural Information Processing Systems*, 2011.

Wauthier, F., Jordan, M., and Jojic, N. Efficient ranking from pairwise comparisons. In *International Conference on Machine Learning*, pp. 109–117, 2013.

Yue, Y., Broder, J., Kleinberg, R, and Joachims, T. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 2012.