Fifty-fourth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
September 27 - 30, 2016

Active Top-K Ranking from Noisy Comparisons

Soheil Mohajer
ECE, University of Minnesota
Email: soheil@umn.edu

Changho Suh
EE, KAIST
Email: chsuh@kaist.ac.kr

***Abstract*—We explore the active top-K sorting problem, in which the goal is to recover the top-K items in order out of n items, from *adaptive* pairwise comparisons that are collected possibly in a sequential manner as per our design choice. 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 an upper bound on the sample size required for reliable top-K sorting. 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 runs linearly in n and which achieves the sample complexity bound. Our theoretical findings are corroborated via numerical experiments.**

I. Introduction

Ranking is one of the central problems that have proved crucial in a wide spectrum of contexts: social choice [1], [2], web search and information retrieval [3], recommendation systems [4], and crowd sourcing [5], to name a few. The task aims to bring a consistent ordering to a collection of items, given only partial preference information. The two main paradigms among a large volume of works on ranking include spectral ranking algorithms [6], [3], [7] and maximum likelihood estimation (MLE) [8]. While these ranking schemes yield reasonably good estimates which are faithful globally w.r.t. the latent preferences (i.e., low ℓ_2 loss), the minimum ℓ_2 loss does not necessarily imply a high ranking accuracy. Also these methods focus on recovery of the entire-item ordering, not well customized for many realistic scenarios in which only a few significant items, say top-K items, are often desired to be retrieved.

In an effort to exploit the more practically relevant scenario, Chen-Suh [9] studied the top-K ranking problem, and characterized the minimax limit on the sample size needed (i.e., the sample complexity) for reliable top-K ranking, assuming a prominent statistical model called the Bradley-Terry-Luce (BTL) model [10], [11]. However, this development is intended for a *passive* measurement setting in which pairwise data are collected prior to analysis.

In many applications which often allow for interaction with users, we may be able to choose comparison pairs of items in an adaptive manner. This adaptive way of ranking can possibly save for a large number of blindly collected measurements and/or yield a higher ranking accuracy [12]. Motivated by these applications, this work investigates the problem of retrieving top-K items under an *adaptive* measurement setting in which pairwise comparisons are gathered interacting with a ranker (termed *active ranking*). In particular, we seek to answer the following questions: (a) how much can active ranking provide performance improvements over passive ranking? (b) how does the limit on the adaptive sample size for feasible top-K sorting scale with K?

Main contributions. In this work, we make some progress towards addressing these questions for a general measurement model in which the pairwise comparison probabilities are *arbitrary* subject to a mild condition (see (4) in Section II for details) and thus which includes as special cases various models like the BTL model, Strong Stochastic Transitivity (SST) model [13], and uniform noise model [14].

Our main contribution lies in establishing an upper bound on the adaptive sample size necessary for identifying top-K items in order:

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

where $\Delta_K = \min_{i\in[K]}\min_{j:j\geq i}(P_{ij}-0.5)^2$. Here P_{ij} indicates the probability of item i being preferred over item j. Observe that the sample complexity bound well scales with K: $O(n\log\log n/\Delta_K)$ in the small K regime (e.g, $K=O(\log n)$); $O(K\log^2 K/\Delta_K)$ in the large K regime (e.g., $K=\Theta(n)$). Another consequence of our result is that active ranking can provide significant multiplicative gains over passive ranking. For instance, 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 I in Section III for details.

Our second contribution is the development of a computationally-feasible linear-time algorithm that returns correct top-K ordered items as soon as the number of adaptive comparisons exceeds the above bound promised. The proposed scheme is based on a *divide-&-conquer* concept which also forms the basis of a well-known merge-sorting algorithm. It runs in multiple rounds. In Round 1, we divide the set of n items into multiple groups. We then identify top-item for each group and form a short list containing the local winners. The top-item in the short list is then selected. In Round 2, we backtrack the path along which the top-item was passing, and identify the second-top in the home group where the top-item was originated from. Competing the newly selected item with those in the short list, we choose

the second-top item. We repeat this procedure in subsequent rounds until we identify top-K items in order. One significant distinction w.r.t. *noiseless* sorting algorithms that have been well studied especially in the TCS literature is that, in our noisy measurement setting, we employ *repeated comparisons* to combat the noise effect, and also identify the minimum number of repeated comparisons required to ensure that the retrieved items are correct winners. The carefully chosen number for repeated comparisons together with a couple of bounding techniques plays a key role in characterizing the above sample complexity bound. Finally, we conduct numerical experiments to corroborate our main results.

Related work. To the best of our knowledge, Chen-Suh [9] was the first work to focus on top-K identification under the non-adaptive random comparison model for the first time. They developed a nearly linear-time algorithm, called `SpectralMLE`, that achieves the order-wise optimal sample complexity for feasible top-K ranking under the BTL model. Subsequently, sample complexity analyses were made with regard to different yet popular ranking paradigms such as simple counting methods [15] and spectral methods [16] (e.g., `RankCentrality` [6]), under passive measurement settings. 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-Mao-Weinberg [17] developed an active ranking algorithm, which is limited to the top-selection case. Interestingly, we found that for the $K = 1$ case and the uniform noise model, their algorithm can achieve the same sample complexity as ours for a certain target error rate. Szörényi *et.al.* [18] also focused on the $K = 1$ case under the BTL model, thus developing an active ranking algorithm which however yields a larger sample complexity than ours. Most recently, Heckel *et.al.* [19] proposed an active ranking algorithm for a general problem setting. We found that their algorithm is outperformed by ours when specializing it to top-K sorting setting of our interest. See III-A for details.

There has been a proliferation of active ranking algorithms [20], [21], [22], [23], [14], [24]. While interesting ranking schemes are developed for perfect ranking [20], [21], [22] and approximate ranking [21], [23], [14], [24], they are not customized for top-K selection and hence the sample complexity for top-K retrieval is not analyzed therein.

On the other hand, the best-K identification with adaptive sampling has been extensively explored under the name of the multi-armed bandit problem [25], [26], [27], [28] 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 [2], [29], [30], [31].

Notation. Unless specified otherwise, we use $[n]$ to represent $\{1, 2, \ldots, n\}$, and $\log$ to represent a logarithm in base 2. The standard notation $f(n) \gtrsim g(n)$ (res. $f(n) = O(g(n))$ or $f(n) \lesssim g(n)$) means there exists a constant $c > 0$ such that $f(n) \geq cg(n)$ (resp. $f(n) \leq cg(n)$).

II. 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 as per the following rule:

$$Y_t = \begin{cases} 1 & \text{with probability } P_{ij} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

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

$$Y_{ij} := \sum_{t: e_t = (i,j), e_t \in \mathcal{E}} Y_t; \quad \boldsymbol{Y} := \{Y_{ij} : (i, j) \in \mathcal{E}\}. \tag{3}$$

Ground-truth ranking. 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 [13] suggests one way to relate the ranking to the model parameters P_{ij}'s by putting the following constraint[1]: $P_{ik} > P_{jk}$ for all $k \neq \{i, j\}$ whenever item $i \succ$ item j. In this paper, we consider a more general setting by relaxing the constraint as:

$$P_{ij} > \frac{1}{2} \qquad \text{whenever } i \succ j. \tag{4}$$

Notice that the new constraint (4) 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 $i \succ j$,

$$P_{ij} = \begin{cases} \frac{1}{2} + \gamma > \frac{1}{2}, & \text{(uniform noise model);} \\ \frac{w_i}{w_i + w_j} > \frac{1}{2}, & \text{(BTL model).} \end{cases} \tag{5}$$

where γ denotes an arbitrary constant $\in (0, 0.5)$ and w_i indicates the importance score w.r.t. item i that determines a ranking.

Performance metric and goal. Given the pairwise comparisons, one wishes to know whether or not the top-K ordered items are identifiable. In light of this, we consider the probability of error P_{e} in decoding the correct top-K order, namely, $P_e(\psi) := \mathbb{P}\{\psi(\boldsymbol{Y}) \neq (1 \succ \cdots \succ K)\}$, 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^*, 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.

[1]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 [13].

III. Main Results

As noted in the passive ranking setup [9], 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 setting of our interest, 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:

$$\Delta_K = \min_{i\in[K]} \min_{j:j>i} (P_{ij} - 0.5)^2. \tag{6}$$

Observe 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 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 IV for the proof of Theorem 1 and algorithm description.

Notice that in (6), the term $P_{ij} - 0.5$ captures how noisy the comparison data is. It 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 small Δ_K means the difficulty of ranking which results in an increase of sample complexity. We also provide several interesting remarks in order.

Penalty due to noisy measurements: As mentioned earlier, the top-K sorting problem has been extensively explored in the TCS literature, but only the noiseless setting has been the main focus, thus yielding sample complexity characterization:

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

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 poly$(\log n)$, as long as Δ_K is not too small (scales at most with poly$(\log n)$).

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 see that the sample complexity bound scales with K in a graceful manner. Here one interesting observation that one can make in the $K = O(\log n)$ regime of practical interest is that the sample complexity bound can be irrelevant to K under some measurement model. One such example is the uniform noise model where $\Delta_K = \gamma^2$ and thus

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

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 interested regime where $K = O(\log n)$ and consider two special measurement models: (1) uniform noise model; (2) BTL model. In the uniform noise model, Shah-Wainwright [15] made some progress on 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 $\Omega\left(\frac{n^2 \log n}{\log\log n}\right)$, which is quite substantial. In the BTL model, Chen-Suh [9] 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 (13), 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 I.

Computational complexity: A noticeable feature of our proposed algorithm is its low computational complexity. It runs in time $O(n)$ in the above practically-relevant regime where $K = O(\log n)$. For general K, it runs in time $O(n + K \log K)$, being nearly linear in n. Here, it should be noted that this complexity assumes that the input fed to our ranking algorithm is the sufficient statistic of the outcome comparisons: Y_{ij} (see (3)), rather than the entire collection of Y_t's associated with the pair (i, j). This will be evident later when describing the algorithm.

A. Comparison to Related Work

Braverman-Mao-Weinberg [17] developed an active ranking algorithm for the $K = 1$ case and analyzed the order-wise tight sample complexity in terms of target error rate under the uniform noise model. More concretely, suppose we want

	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 I
MULTIPLICATIVE GAINS OF ACTIVE RANKING OVER PASSIVE RANKING.

the error probability not to exceed a target error rate δ. Then, the result of [17] implies

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

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 $\delta \leq \frac{1}{\log n}$. 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.* [18] also developed a top-selection active ranking algorithm and analyzed sample complexity under the BTL model. Their sample complexity bound reads around the order of $n \log n$, thus yielding a larger sample complexity compared to ours.

On the other hand, Heckel *et.al.* [19] proposed an active ranking algorithm for a general problem setting which subsumes top-K sorting as a special case. They also characterized sample complexity under a fairly general measurement model which is slightly different from ours, but still includes several popular models as special cases. However, their algorithm does not outperform ours. For instance, in the uniform noise model, their algorithm achieves around the order of $\frac{Kn^2}{\gamma^2}$ in sample complexity (up to a logarithmic factor gap). Hence, comparing this to (10), our algorithm outperforms by a factor of Kn.

IV. PROPOSED RANKING ALGORITHM

A. Top Selection ($K = 1$): Binary Search Tree

We first focus on the case $K = 1$, in which we are only interested in determining the single top item. The adaptive algorithm proposed below can be viewed as a customized version of *binary tree search* in which decisions in each layer are made in accordance with *random measurements*. Here one distinctive feature relative to conventional binary search is that in order to combat against the uncertainty of the observations, we repeat each binary measurement multiple (say m) times. It is shown that for a carefully designed m (to be detailed soon), the algorithm will output the index of the maximum item with overwhelming probability.

The algorithm builds a binary tree of depth $\lceil \log n \rceil$. 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.

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)

```
initialization;
n ← |X|;
for i ← 1 to |X| do
    a(i) ← i;
end
comparison;
for ℓ ← 1 to log n do
    for i ← 1 to n/2^ℓ do
        T ← 0;
        for t ← 1 to m do
            e_t ← (a(2i − 1), a(2i));
            T ← T + Y_t;
        end
        if T ≥ m/2 then
            a(i) ← a(2i − 1);
        else
            a(i) ← a(2i);
        end
    end
end
a* ← a(1)
```

Algorithm 1: SELECT$(X; m)$.

Number of measurements: The algorithm consists of $\lceil \log n \rceil$ layers, and $n/2^\ell$ pairs are being tested in layer ℓ. Hence, the total number of tests will be $\sum_{\ell=1}^{\lceil \log n \rceil} n2^{-\ell} \leq n$. Each test requires m binary measurements, which implies the total number of measurements for a population of size

n is given by

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

for a dataset of size n.

Error analysis: The algorithm will successfully return[2] item 1 if and only if the top item survives all the comparisons *it is involved in*. Since the tree has $\lceil \log n \rceil$ layers, each element will be tested against at most $\lceil \log n \rceil$ other items. Let j_ℓ be the item that is tested against item 1 in layer ℓ of the tree. We have

$$\begin{aligned}
P(\text{error}) &= \mathcal{E}[\texttt{SELECT}(X;m)] \\
&= \mathbb{P}\left[\bigcup_{\ell=1}^{\lceil \log |X| \rceil} \{\text{item 1 gets eliminated in layer } \ell\}\right] \\
&\overset{(a)}{\leq} \sum_{\ell=1}^{\lceil \log |X| \rceil} \mathbb{P}\left[\sum_{t=1}^{m} Y_{1j_\ell}^{(t)} < \frac{m}{2}\right] \\
&\overset{(b)}{\leq} \sum_{\ell=1}^{\lceil \log |X| \rceil} 2^{-m\mathsf{KL}(\frac{1}{2}\|P_{1j_\ell})} \\
&= \sum_{\ell=1}^{\lceil \log |X| \rceil} 2^{-\frac{m}{2}\log \frac{1}{4P_{1j_\ell}(1-P_{1j_\ell})}} \\
&\overset{(c)}{\leq} \log|X| \cdot \max_{j\neq 1} 2^{-\frac{m}{2}\log \frac{1}{4P_{1j}(1-P_{1j})}} \\
&= 2^{\log\log|X| - \frac{m}{2}\min_{j\neq 1}\log \frac{1}{4P_{1j_\ell}(1-P_{1j_\ell})}}
\end{aligned} \tag{16}$$

where $\mathsf{KL}(\cdot \,\|\, \cdot)$ indicates the Kullback-Leibler divergence defined w.r.t. the logarithm in base 2. Note that we used the union bound in (a), and Chernoff bound in (b). It is worth mentioning that the first term in (c) should read $\lceil \log |X| \rceil$ instead of $\log |X|$ for a mathematical rigor. However, for illustrative simplicity, we drop the ceiling, which turns out not to affect the entire analysis to be followed.

Let $\Delta_{ij} = (P_{ij} - 0.5)^2$. Then we have

$$\begin{aligned}
\frac{\ln 2 \log\log n}{\Delta_{ij}} = \frac{4\ln 2 \log\log n}{(2P_{ij}-1)^2} &\geq \frac{4\ln 2 \log\log n}{-\ln[1-(2P_{ij}-1)^2]} \\
&= \frac{4\log\log n}{\log \frac{1}{4P_{ij}(1-P_{ij})}}.
\end{aligned}$$

It is clear that the probability of error vanishes at rate $(\log n)^{-\epsilon}$ as n grows, provided that

$$m \geq \frac{(1+\epsilon)\ln 2}{2} \frac{\log\log n}{\min_{j\neq 1}\Delta_{1j}} = \frac{(1+\epsilon)\ln 2}{2} \frac{\log\log n}{\Delta_1},$$

where $\Delta_1 = \min_{j>1}\Delta_{1j}$.

Plugging this choice for m into (15), we can obtain the number of comparison required for SELECT algorithm. This implies

$$S_1 = O\left(\frac{n\log\log n}{\Delta_1}\right), \tag{17}$$

[2]Recall that we assumed that the ground truth ranking is the order of $1 \succ \cdots \succ n$.

for $K = 1$.

Example 1: Consider a set of $n = 8$ items, in which item number 1 is top ranked item. This item can be identified by the binary tree in Figure 1, in which each binary decision is made based on the majority rule among m comparisons. It is worth mentioning that a wrong decision 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.

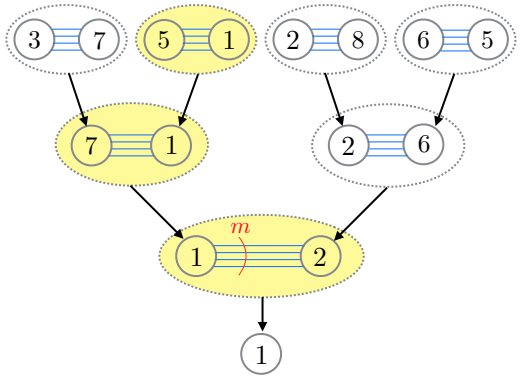


Fig. 1. An illustrative example for Binary Search Tree with repeated comparisons.

Remark 1: Parallel to this work, the top-selection problem is studied in [17] under the uniform noise model. The algorithm in [17] 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 $1/\text{poly}(\log n)$, the algorithm of [17] requires the same number (up to a constant factor) of pair-wise comparisons, as the one presented above.

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

In this section we present an algorithm to identify the top K items along with their order. The algorithm is built based on SELECT, which can find the the single top item with high probability. We first split the dataset with n items into K groups each of size n/K. Then we identify the top item in each sub-group using SELECT, and form a short list that includes all top items from the sub-groups. Then we build a (max-)HEAP data-structure for the short list of K items. 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, we go back to its home sub-group, identify the second top item in that sub-group, and add 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 elements. The main algorithm, TOP is presented in Algorithm 2.

Example 2: In the example illustrated in Figure 2, the goal is to identify the top $K = 10$ items, along with their

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 $i \leftarrow 1$ **to** K **do**
  $C_i \leftarrow \{x[(i-1)Q+1], x[(i-1)Q+2], \ldots, x[\min\{iQ, n\}]\}$;
  $b(i) \leftarrow (i-1)Q + \texttt{SELECT}(C_i; m)$;
end
$Z \leftarrow \{x[b(1)], x[b(2)], \ldots, x[b(K)]\}$;
$\texttt{BuildHeap}(Z; m)$;
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

Algorithm 2: `TOP`

order. We first split the items into $K = 10$ groups of equal size, namely $C_1, C_2, \ldots, C_{10}$. The top item in each sub-group can be found using the binary search tree. Then we build a heap data-structure on the short list, obtained using the top items. As it is clear in the figure, heap is also a binary tree, with the property that both children of each node have a rank lower than their parent. Then the root will be reported as the top item.

Next, we go back to the home-subgroup of the root, to find the second-top item of that group. The top item 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, we can identify items $1, 2, \ldots, 10$.

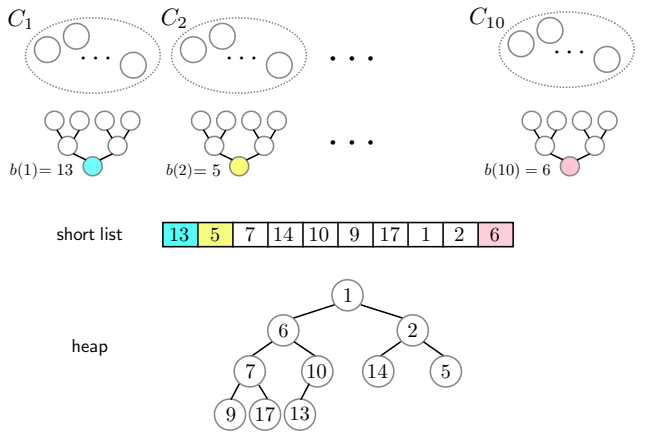


Fig. 2. The top items of the sub-groups form a short list, which will be used to build a heap.

For the sake of completeness, we also present a modified version of `Heapify` and `BuildHeap` algorithms that work based on noisy observations in Algorithm 3 and Algorithm 4,

Input: Integers i and m
Data: Array Z of indices and Data X
Output: Array Z with sub-tree at node i being a max-heap

left $\leftarrow 2i$;
right $\leftarrow 2i+1$;
$T \leftarrow 0$;
for $t \leftarrow 1$ **to** m **do**
  $e_t \leftarrow (Z[\text{left}], Z[i])$;
  $T \leftarrow T + Y_t$;
end
if left $\leq |Z|$ **and** $T \geq \frac{m}{2}$ **then**
  max $\leftarrow$ left;
else
  max $\leftarrow i$;
end
$T \leftarrow 0$;
for $t \leftarrow 1$ **to** m **do**
  $e_t \leftarrow (Z[\text{right}], Z[\text{max}])$;
  $T \leftarrow T + Y_t$;
end
if right $\leq |Z|$ **and** $T \geq \frac{m}{2}$ **then**
  max $\leftarrow$ right;
end
if max $\neq i$ **then**
  $\texttt{swap}(Z[i], Z[\text{max}])$;
  $\texttt{Heapify}(Z, X, \text{max}; m)$;
end

Algorithm 3: `Heapify`

Input: Integers i and m
Data: Array Z of indices and Data X
Output: max-heap Z

for $i \leftarrow \lfloor |Z|/2 \rfloor$ **downto** 1 **do**
  $\texttt{Heapify}(Z, X, i; m)$
end

Algorithm 4: `BuildHeap`

respectively. It is worth mentioning the sample complexity of building a `HEAP` over N items using noiseless measurements is $O(N)$, and insertion of a new item to an existing `HEAP` with N items requires $O(\log N)$ comparisons. Finally, `HEAP` maintains the maximum item at the beginning of the list, so that max-extraction can be done for free. Here, in the modified version of the algorithms, we repeat each comparison used for `HEAP` for m times, and decide based on the majority of the observation. The variable m is a design parameter, which will be determined later.

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 (15).

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

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

Error analysis: We expect to discover the top item correctly in each run of SELECT, only when the sub-group includes one of the top K items of the original population, and we don't care about the validity of the result otherwise. On the other hand, during the iterative part of the algorithm, every single wrong decision that includes a top-K item may propagate to the final result, and hence, we consider that as a source of error. Hence, using the union bound we have

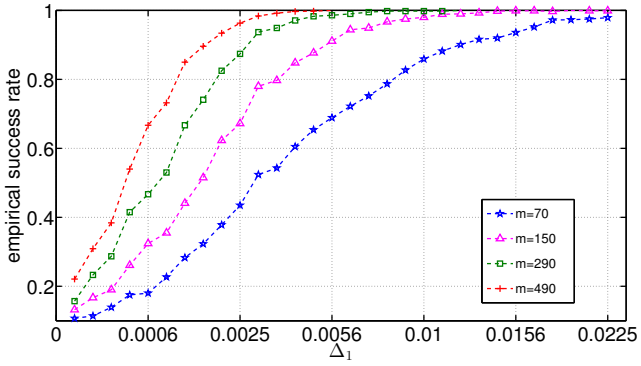
$$\begin{aligned}
&P(\text{error}) = \mathcal{E}[\texttt{TOP}(X, K; m)] \\
&\quad \le \sum_{i=1}^{2K} \mathcal{E}[\texttt{SELECT}(C_i; m)] + \mathcal{E}[\texttt{BuildHeap}(Z; m)] \\
&\qquad + K\mathcal{E}[\texttt{InsertHeap}(Z; m)] \\
&\quad \overset{(a)}{\le} 2K \log \frac{n}{K} 2^{-2m \min_{i\in[K], j>i} \Delta_{ij}} \\
&\qquad + K(1 + \log K) 2^{-2m \min_{i\in[K], j>i} \Delta_{ij}} \\
&\quad \overset{(b)}{=} 2^{1+\log K + \log\log(n/K) - 2m\Delta_K} \\
&\qquad + 2^{1+\log K + \log\log K - 2m\Delta_K}. \qquad (19)
\end{aligned}$$

where (a) holds since in each of $2K$ runs of algorithm SELECT the sub-groups include at most n/K items, and an error in SELECT affects the ultimate result only if the corresponding subgroup includes (at least) one of the top-K items. We have also used the fact $\Delta_K = \min_{i\in[K], j>i} \Delta_{ij}$ in (b).

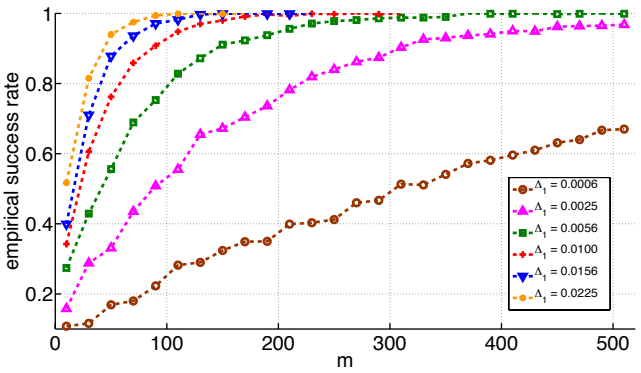
By choosing m so that it satisfies [3]

$$\begin{aligned}
m &\ge (1+\epsilon) \frac{\max\{\log K, \log\log(n/K)\}}{\Delta_K} \\
m &\ge (1+\epsilon) \frac{\max\{\log K, \log\log n\}}{\Delta_K}, \qquad (20)
\end{aligned}$$

[3]Note that condition $m \ge (1+\epsilon)\frac{\log\log n}{\Delta_K}$ is needed to guarantee decay of the error at rate $1/\text{poly}(\log n)$, specially when $K = O(\log n)$.



(a) success rate vs. Δ_1



(b) success rate vs. m

Fig. 3. Evolution of empirical success rate with respect to Δ_1 and m when $n = 100$ and $K = 1$.

we have

$$\mathcal{E}[\texttt{TOP}(X, K; m)] \le 2^{-c \log\log n} = (\log n)^{-c}, \qquad (21)$$

in which $c > 0$ is a positive constant depending of ϵ. Hence

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

is sufficient to have vanishing error probability. This implies

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

Remark 2: The repetition parameter used for the SELECT algorithm, and the one used for HEAP can be potentially different, and accordingly optimized. However, our analysis shows that individual choice of repetition parameter can provide a minor gain only for a tiny range of K. So, we rather choose the same parameter for the sake of simplicity.

V. Experimental Results

We conduct two synthetic experiments to corroborate our main result stated in Theorem 1. We examine the single top selection ($K = 1$) using Algorithm 1, under BTL model. To this end, we generate a preference score vector of size $n = 100$ which contains distinct scores subject to $\Delta_1 =$

$(P_{12} - 0.5)^2 = \left(\frac{w_1 - w_2}{2(w_1+w_2)}\right)^2$. For each case we run the algorithm for several values of m. In each case, the empirical success rate is obtained by averaging over 10000 Monte Carlo trials. Figure 3 shows how the empirical success rate varies with respect to Δ_1 and m. In particular, Figure 3(b) shows that for a fixed success rate m scales with $1/\Delta_1$, which is consistent with our theoretical analysis.

VI. Conclusion and Future Work

We investigated the active top-K sorting from noisy comparisons. We characterized an upper bound on the sample size necessary for top-K recovery, thereby demonstrating substantial multiplicative gains over passive ranking for various measurement models. We also developed a nearly linear-time algorithm that can achieve the derived bound. Some future works of interest include: (a) derivation of minimax information-theoretic lower bounds that can possibly match our upper bound, up to constant factors; (b) extension of our ranking algorithm to a variety of statistical models such as the Plackett-Luce model [32], [29] and the mixture BTL model [33]; (c) performance evaluations of our algorithm vs. prior active ranking algorithms for real-world data.

Acknowledgments

The work of C. Suh was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) [2015R1C1A1A02036561].

References

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

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

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

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

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

[6] S. Negahban, S. Oh, and D. Shah, "Rank centrality: Ranking from pair-wise comparisons," 2012.

[7] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN systems*, vol. 30, no. 1, pp. 107–117, 1998.

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

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

[10] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3-4, pp. 324–345, 1952.

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

[12] D. Tschopp, S. Diggavi, P. Delgosha, and S. Mohajer, "Randomized algorithms for comparison-based search," in *Neural Information Processing Systems*, 2011.

[13] P. Fishburn, "Binary choice probabilites: on the varieties of stochastic transitivity," *Journal of Mathematical Psychology*, vol. 10, pp. 327–352, 1973.

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

[15] N. B. Shah and M. J. Wainwright, "Simple, robust and optimal ranking from pairwise comparisons," 2015.

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

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

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

[19] R. Heckel, N. Shah, K. Ramchandran, and M. Wainwright, "Active ranking from pairwise comparisons and the futility of parametric assumptions," *arXiv:1606.08842*, 2016.

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

[21] K. G. Jamieson and R. Nowak, "Active ranking using pairwise comparisons," in *Neural Information Processing Systems*, pp. 2240–2248, 2011.

[22] L. Maystre and M. Grossglauser, "Robust active ranking from sparse noisy comparisons," in *arXiv:1502.05556*, 2015.

[23] N. Ailon, "Active learning ranking from pairwise preferences with almost optimal query complexity," *Journal of Machine Learning*, vol. 13, pp. 137–164, 2012.

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

[25] V. Gabillon, M. Ghavamzadeh, A. Lazaric, and S. Bubeck, "Multi-bandit best arm identification," in *Neural Information Processing Systems*, 2011.

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

[27] K. Jamieson, M. Malloy, R. Nowak, and S. Bubeck, "liiUCB: An optimal exploration algorithm for multi-armed bandits," in *Conference on Learning Theory*, 2014.

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

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

[30] T. Lu and C. Boutilier, "Learning Mallows models with pairwise preferences," in *International Conference on Machine Learning*, pp. 145–152, 2011.

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

[32] R. L. Plackett and R. D. Luce, "The analysis of permutations," *Applied Statistics*, pp. 193–202, 1975.

[33] S. Oh and D. Shah, "Learning mixed multinomial logit model from ordinal data," in *Neural Information Processing Systems*, pp. 595–603, 2014.