The Sample Complexity of Best-k Items Selection from Pairwise Comparisons

Wenbo Ren [1] **Jia Liu** [2] **Ness B. Shroff** [1 3]

Abstract

This paper studies the sample complexity (aka number of comparisons) bounds for the active best-k items selection from pairwise comparisons. From a given set of items, the learner can make pairwise comparisons on every pair of items, and each comparison returns an independent noisy result about the preferred item. At any time, the learner can adaptively choose a pair of items to compare according to past observations (i.e., active learning). The learner's goal is to find the (approximately) best-k items with a given confidence, while trying to use as few comparisons as possible. In this paper, we study two problems: (i) finding the probably approximately correct (PAC) best-k items and (ii) finding the exact best-k items, both under strong stochastic transitivity and stochastic triangle inequality. For PAC best-k items selection, we first show a lower bound and then propose an algorithm whose sample complexity upper bound matches the lower bound up to a constant factor. For the exact best-k items selection, we first prove a worst-instance lower bound. We then propose two algorithms based on our PAC best items selection algorithms: one works for $k = 1$ and is sample complexity optimal up to a loglog factor, and the other works for all values of k and is sample complexity optimal up to a log factor.

[1]Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, USA [2]Department of Computer Science, Iowa State University, Ames, Iowa, USA [3]Department of Electrical and Computer Engineering, The Ohio State University, Columbus, Ohio, USA. Correspondence to: Wenbo Ren <ren.453@osu.edu>, Jia Liu <jialiu@iastate.edu>, Ness B. Shroff <shroff.11@osu.edu>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

1. Introduction

1.1. Background and Motivation

Ranking from pairwise comparisons (or pairwise ranking) is a fundamental problem that has been widely applied to various areas, such as recommender systems, searching, crowd-sourcing, and social choices. In a pairwise ranking system, the learner wants to learn the full or partial ranking (e.g., best-k items) of a set of items from noisy pairwise comparisons, where items can refer to various things such as products, posts, choices, and pages; and comparisons refer to processes or queries that indicate qualities or users' preferences over the items. In this paper, for simplicity, we use the terms "item", "comparison", and "users' preference".

A *noisy* pairwise comparison is a query over two items that returns a noisy result about the preferred one. Here, "noisy" simply means that the comparison could return the less preferred one, which may be the result of the uncertain nature of physics, machines, or humans. Since the comparisons can reveal some information about the users' preferences, by repeatedly comparing these items, the learner may find a reasonable global ranking (e.g., (Hunter, 2004)) or local ranking (e.g., (Park et al., 2015)) of these items.

Based on when the comparisons are generated, the ranking problems can be divided into two classes: passive ranking (e.g., (Park et al., 2015; Shah et al., 2017)) and active ranking (e.g., (Pfeiffer et al., 2012; Chen et al., 2013; Falahatgar et al., 2017a; Ren et al., 2019)). In passive ranking, the learner first has all the comparison data and then develops a reasonable ranking. In active ranking, the learner does not have all the comparison data at the beginning, and can adaptively choose items to compare during the learning process. This paper studies the *fully active ranking (or active learning)*, where for each comparison, the learner can adaptively choose two items to compare according to past observations. Chen et al. (2013) showed that in a crowd-sourcing dataset, their active ranking algorithm uses only 3% comparisons and achieves almost the same performance as passive ranking.

This paper focuses on the *best-k-items-selection* problem. For many applications, ranking all the items may be neither efficient nor necessary. For instance, in a video sharing website, filters may generate hundreds of candidate videos, but the website may only want to present 30 videos to the user. Thus, it is not necessary to rank all these videos, and a more efficient way can be to first select the best 30 videos and then rank them. The best-k items selection can be of interest to many different applications.

In previous works (e.g., (Yue and Joachims, 2011; Busa-Fekete et al., 2014; Szörényi et al., 2015; Falahatgar et al., 2017b; 2018; Saha and Gopalan, 2019a;b)), the problem of best item selection has been studied in different settings. However, the problem of best-k items selection has been less investigated. We note that best-k items selection is not a naive extension to the best item selection. For instance, in the deterministic case, finding the max number is easy by sequentially doing $n-1$ comparisons and eliminating the smaller ones, while finding the largest k numbers in $O(n \log k)$ time needs more complex algorithms (e.g., quick select (Hoare, 1961)). The same is true in non-deterministic settings. We do not find a method to extend best item selection algorithms to an efficient best-k one.

This paper studies both the *exact* and *probably approximately correct (PAC)* best-k items selection. Exact selection simply means finding the exact best-k items. PAC selection is to find k items that are approximately best or good enough (see Section 1.2 for details), which can avoid the cases where the preferences over two items are extremely close, making exactly ranking them too costly.

In summary, this paper studies the problem of using fully active ranking (active learning) to find the exact or PAC best-k items from noisy pairwise comparisons with a certain confidence and use as few comparisons as possible.

1.2. Problem Formulation and Notations

Assume that there are n items, indexed by $1, 2, 3, ..., n$, and we use $[n] = \{1, 2, 3, ..., n\}$[1] to denote the set of these items. For these items, we make the following assumptions:

A1) Time-invariance. For any items i and j in $[n]$, we assume that the distributions of the comparison outcomes over items i and j are time-invariant, i.e., there is a number $p_{i,j}$ in $[0, 1]$ independent of time such that for any comparison over items i and j, item i wins the comparison with probability $p_{i,j}$, where "item i wins the comparison" means that the comparison returns item i as the preferred one.

A2) Tie Breaking. We assume that for every comparison, exactly one item wins. If a tie does happen, we randomly assign one item as the winner. Thus, for any items i and j in $[n]$, $p_{i,j} + p_{j,i} = 1$.

A3) Independence. We assume that the comparison results are independent across time, items, and sets.

[1]For any positive integer m, we define $[m] := \{1, 2, 3, ..., m\}$.

We note that assumptions A1) to A3) are common in the literature (e.g., (Szörényi et al., 2015; Shah and Wainwright, 2017; Falahatgar et al., 2017a;b; 2018; Heckel et al., 2018; Katariya et al., 2018; Heckel et al., 2019; Saha and Gopalan, 2019a;b; Ren et al., 2019)). In this paper, we make two more assumptions to restrict our problems to specific conditions.

Before making these two assumptions, we introduce some notations. For two items i and j in $[n]$, we define $\Delta_{i,j} := |p_{i,j} - 1/2|$ as the gap of $p_{i,j}$ and $1/2$, which can measure how difficult to order items i and j by comparing them. Also, we define $p_{i,i} := 1/2$ for all items i. For real numbers a, b, we define $a \vee b := \max\{a, b\}$, and $a \wedge b := \min\{a, b\}$.

A4) Strong stochastic transitivity (SST) (Shah et al., 2017; Falahatgar et al., 2018). In this paper, the items are said to satisfy SST if and only if (i) there is a strict order over these n items, (ii) if $i \succ j$[2], then $p_{i,j} > 1/2$,[3] and (iii) for any three items i, j, and l with $i \succ j \succ l$, $p_{i,l} \geq p_{i,j} \vee p_{j,l}$.

A5) Stochastic triangle inequality (STI) (Falahatgar et al., 2018). The items are said to satisfy STI if for any three items i, j, and l, $\Delta_{i,l} \leq \Delta_{i,j} + \Delta_{j,l}$.

We note that many widely used parametric models such as the Bradley-Terry-Luce (Bradley and Terry, 1952; Luce, 2012) (BTL) and Thurstone's model (Thurstone, 1927) satisfy SST and STI, and thus, the algorithms in this paper can be directly used under these models. In this paper, we do not restrict our results to specific parametric models. Without loss of generality, we use $r_1 \succ r_2 \succ \cdots \succ r_n$ to denote the unknown true ranking.

The first problem is the PAC best-k items selection. We follow the definition of PAC best item of Falahatgar et al. (2017a;b; 2018) to define the PAC best-k items. We note that when $k = 1$, our definition of PAC best items is the same as that of Falahatgar et al. (2017a;b; 2018).

Definition 1 ((ϵ, k)-optimal subsets)**.** *For a set S, given $k \leq |S|$, and $\epsilon \in [0, 1]$, a set $U \subset S$ is said to be an (ϵ, k)-optimal subset of S if $|U| = k$ and $p_{i,j} \geq 1/2 - \epsilon$ for any items i in U and j not in U.*

If $\epsilon < \min_{i \in [n]: r_k \succ i} \Delta_{i,r_k}$, an (ϵ, k)-optimal subset of S is exactly the set of the best-k items of S. However, if we do not have a priori knowledge about the gaps, we cannot use the PAC algorithms to find the exact best items. The number ϵ is called the error tolerance. We note that in an (ϵ, k)-optimal subset, every item i has $p_{i,r_k} \geq 1/2 - \epsilon$.

Problem 1 (PAC best-k items selection (PAC k-selection))**.** *Given n items $[n]$, $k \leq n/2$, and $\delta, \epsilon \in (0, 1/2)$, we want to find an (ϵ, k)-optimal subset of S with probability at least*

[2]Term $i \succ j$ means that i ranks higher than j in the true order.

[3]In some works, we may have $p_{i,j} = 1/2$ for items $i \neq j$. However, in this paper, we do not allow $p_{i,j} = 1/2$ to avoid the case where the term "best-k items" is not well defined.

$1-\delta$, and use as few comparisons as possible.

The second problem is the exact best-k items selection. Under SST, since there is a strict order over these n items, the best-k items are unique. The best-k items are $r_1, r_2, ..., r_k$, and finding the best-k items is to find the set $\{r_1, r_2, ..., r_k\}$. We do not need to order these best-k items but only need to find a k-sized set that contains all the best-k items.

Problem 2 (Exact best-k items selection (exact k-selection))**.** *Given n items, $k \le n/2$, and $\delta \in (0, 1/2)$, we want to find the best-k items with probability at least $1-\delta$, and use as few comparisons as possible.*

We define the gap of item i as

$$\Delta_i = \mathbb{1}_{i \succ r_{k+1}} \cdot \Delta_{i,r_{k+1}} + \mathbb{1}_{r_k \succ i} \cdot \Delta_{r_k,i}, \qquad (1)$$

and our sample complexity (aka number of comparisons) bounds for the exact k-selection depends on these gaps.

1.3. Main Contributions

For the PAC k-selection problem, we first prove an $\Omega(n\epsilon^{-2}\log(k/\delta))$ lower bound on the expected number of comparisons, and then propose an algorithm with sample complexity $O(n\epsilon^{-2}\log(k/\delta))$, which implies that our upper bound matches the lower bound up to a constant factor.

For the exact k-selection problem, we first prove a worst-instance sample complexity lower bound $\Omega(\sum_{i\in[n]}[\Delta_i^{-2}\log\delta^{-1}] + \log\log\Delta_{r_k}^{-1})$. We then propose an algorithm for $k = 1$ with sample complexity $O(\sum_{i\neq r_1}[\Delta_i^{-2}(\log\delta^{-1} + \log\log\Delta_i^{-1})])$ based on our PAC k-selection algorithm, which is optimal up to a loglog factor. Finally, we propose another algorithm for general values of k with sample complexity $O(\sum_{i\in[n]}[\Delta_i^{-2}(\log(n/\delta) + \log\log\Delta_i^{-1})])$, which is optimal up to a log factor.

2. Related Works

An early work that has studied the exact k-selection was done by Feige et al. (1994). Feige et al. (1994) have shown that if $\Delta_{i,j} \ge \Delta > 0$ for all items i and j where $\Delta > 0$ is a priori known, then to find the best-k items of $[n]$ with probability at least $1-\delta$, $\Theta(\Delta^{-2}\log(k/\delta))$ comparisons are sufficient and necessary for worst instances. However, the work of Feige et al. (1994) requires a priori knowledge of a lower bound of the values of $\Delta_{i,j}$'s to run, which may not be possible in practice. This paper does not assume this knowledge. Further, the sample complexity in Feige et al. (1994) depends on the minimal gaps, i.e., $\min_{i\neq j}\Delta_{i,j}$, while the sample complexity in this paper depends on Δ_{r_i,r_k} or $\Delta_{r_i,r_{k+1}}$, which exploits unequal gaps better.

Chen and Suh (2015); Negahban et al. (2017); Chen et al. (2019) studied the exact k-selection problem under the Plackett-Luce (Plackett, 1975; Luce, 2012) (PL) model[4], which is a parametric model that satisfies SST and STI. They proposed algorithms with adaptivity[5] one, which can find the best-k items of $[n]$ with high probability[6] by $O(n\Delta_{r_k,r_{k+1}}^{-2}\log n)$ comparisons. In contrast, this paper focuses on fully active algorithms (i.e., the number of adaptivity is unlimited) and the algorithms are not restricted to parametric models. Another work that has focused on the exact k-selection problem under the MNL model is Chen et al. (2018). Chen et al. (2018) proposed an exact k-selection algorithm from pairwise comparisons with sample complexity $O(n\log^{14}(n))$. They also studied ranking from multi-wise comparisons, which is beyond the scope of this paper.

Busa-Fekete et al. (2014) studied the best item selection problem under Mallows model, and proposed an algorithm with samples complexity $O(n\log(n/\delta))$. Saha and Gopalan (2019b) studied the exact best item selection problem under the PL model with subset-wise feedbacks, and proposed an algorithm with $O(\sum_{i\in[n]}[\Delta_i^{-2}(\log\delta^{-1} + \log\log\Delta_i^{-1})])$ sample complexity for confidence $1-\delta$, which is of the same order as the algorithm in this paper. Compared to the work of Saha and Gopalan (2019b), our algorithms work for all instances satisfying SST and STI, while the PL model is a special case in our setting.

Another focus of this paper is the PAC k-selection problem. To the best of our knowledge, we are the first to propose PAC k-selection algorithms. Prior to this paper, there are works that focused on the PAC best item selection problem. Falahatgar et al. (2017a;b) proved that under SST, to find an item i from $[n]$ with $p_{i,r_1} \ge 1/2 - \epsilon$ with probability at least $1-\delta$, $\Theta(n\epsilon^{-2}\log\delta^{-1})$ comparisons are sufficient and necessary. Earlier to this, Yue and Joachims (2011) proved the same result for cases under the SST and the STI. The works of Saha and Gopalan (2019a) also proved the same sample complexity bounds under the PL model. When $k = 1$, our upper bound and lower bound for the PAC k-selection problem is the same as that of Falahatgar et al. (2017a;b) (ignoring constant factors).

There are also many works that studied the ranking problems under other models, which are beyond the scope of this paper. Shah and Wainwright (2017); Heckel et al. (2018); Katariya et al. (2018); Heckel et al. (2019) studied the active ranking problems under the Borda-Score (BS) model, which can be viewed as a superset of SST and STI in some sense. However, we note that, for instances satisfying SST and STI, BS ranking algorithms may not be as efficient as their perfor-

[4]We note that the PL model, the BTL model, and the multinomial logit (MNL) model (McFadden, 1973; Luce, 2012) share equivalent mathematical formula for pairwise comparisons.

[5]See Agarwal et al. (2017); Braverman et al. (2019) for details about learning with limited adaptivity.

[6]In this paper, "with high probability" means that with probability at least $1 - n^{-p}$, where $p > 0$ is a sufficiently large constant.

mance on BS problems[7]. Agarwal et al. (2017); Braverman et al. (2019) studied the problem of ranking (or finding) the best-k items with limited adaptivity. Feige et al. (1994); Szörényi et al. (2015); Falahatgar et al. (2017a;b; 2018); Ren et al. (2019) studied the (PAC) full ranking problems in various settings, which is less related to this paper.

3. PAC k-Selection

This section studies the sample complexity lower bound and upper bound for PAC k-selection. We first prove that for the worst instances, to find an (ϵ, k)-optimal subset of $[n]$ needs $\Omega(n\epsilon^{-2}\log(k/\delta))$ number of comparisons in expectation. Then, we design an algorithm that solves all instances with at most $O(n\epsilon^{-2}\log(k/\delta))$ number of comparisons in expectation, which shows that both our lower bound and upper bounds are tight (up to a constant factor).

3.1. Lower Bound

We first analyze the lower bound for PAC k-selection, which is stated in Theorem 2. We prove this bound by reducing the pure exploration multi-armed bandit (PEMAB) problem (e.g., (Mannor and Tsitsiklis, 2004; Kalyanakrishnan et al., 2012)) to the PAC k-selection problem under the MNL model and using the lower bounds for the PEMAB problem of Mannor and Tsitsiklis (2004); Kalyanakrishnan et al. (2012) to get the desired lower bound for PAC k-selection. We note that Ren et al. (2018) used a similar method and proved a similar lower bound. However, its definition of PAC k-selection is different from that in this paper. Thus, we need to independently find a lower bound in this paper.[8] Later in subsection 3.2, we show that the lower bound stated in Theorem 2 is tight up to a constant factor.

Theorem 2 (Lower bound for PAC k-selection)**.** *Given $\epsilon \in (0, 1/128)$, $\delta \in (0, e^{-4}/4)$, $n \geq 2$, and $1 \leq k \leq n/2$, there is an n-sized instance satisfying SST and STI such that to find an (ϵ, k)-optimal subset of $[n]$ with probability $1-\delta$, any algorithm needs to conduct $\Omega(n\epsilon^{-2}\log(k/\delta))$ number of comparisons in expectation.*

3.2. Upper Bound and the Algorithm

We develop an optimal algorithm in two steps. Step one is to design a PAC k-selection algorithm with $O(n\epsilon^{-2}\log(n/\delta))$ sample complexity. Step two is to develop another algorithm with $O(n\epsilon^{-2}\log(k/\delta))$ sample complexity through the above algorithm. We note that Falahatgar et al. (2018) proposed an algorithm for finding the PAC full ranking with high probability, and has sample complexity $O(n\epsilon^{-2}\log n)$. In a PAC ranking, the top-k items form an (ϵ, k)-optimal subset of $[n]$, and thus, this PAC full ranking algorithm can be used as a PAC k-selection algorithm. However, the algorithm of Falahatgar et al. (2018) can only guarantee to return correct results with confidence $1 - 1/n$, while in the construction of the k-selection algorithm with sample complexity $O(n\epsilon^{-2}\log(k/\delta))$, we need the confidence to be larger than $1 - 1/n$. Thus, this algorithm is not sufficient for us to obtain the $O(n\epsilon^{-2}\log(k/\delta))$ sample complexity. In this paper, we propose a k-selection algorithm with sample complexity $O(n\epsilon^{-2}\log(n/\delta))$ to achieve this purpose.

3.2.1. Step One: Epsilon-Quick-Select

Our first PAC k-selection algorithm is similar to a classical deterministic k-selection algorithm, Quick Select (Hoare, 1961). In each round, Quick Select randomly picks (some versions may have different picking strategies) an item as a pivot and splits the other items into two piles: one contains items no less than the pivot and the other contains items less than the pivot. After the splitting, according to the sizes of these two piles, we do Quick Select again on one pile. This will be repeated until we find the k-th best item. The expected time complexity of Quick Select is $O(n)$.

When the comparisons are noisy, we need more effort to find the (PAC) best-k items, but the basic idea is similar to Quick Select. For each round t, we randomly pick an item v_t as the pivot, and compare every other item with the pivot for certain times. According to these comparisons, we distribute each item i into one of the following three piles: (i) S_{up}:={item i is "sure" to be better than v_t, i.e., $p_{i,v_t} > 1/2$ with a large probability}; (ii) S_{mid}:={item i is "close to" v_t, i.e., $1/2 - \epsilon \leq p_{i,v_t} \leq 1/2 + \epsilon$ with a large probability}; and (iii) S_{down}:={item i is "sure" to be worse than v_t, i.e., $p_{i,v_t} < 1/2$ with a large probability}. After the splitting, there can be three cases. If S_{up} contains at least k items, then we run our algorithm again on S_{up}. If S_{up} contains less than k items, and $S_{up} \cup S_{mid}$ contains at least k items, then the items in S_{up} along with $(k - |S_{up}|)$ arbitrary items in S_{mid} form an (ϵ, k)-optimal subset. If $S_{up} \cup S_{mid}$ contains less than k items in total (say the number is k'), then we run the algorithm on S_{down} to find the PAC best $(k - k')$ items, and the returned items along with S_{up} and S_{mid} form an (ϵ, k)-optimal subset. The properties of SST and STI guarantee the correctness, and the choice of input confidence for each round guarantees the sample complexity.

The "Quick-Select-like" algorithm is described in Algorithm 2 Epsilon-Quick-Select (EQS). Subroutine 1 Distribute-Item (DI) is a subroutine, which splits the items into three piles. DI is called by EQS with two shifts s_u and s_d being equal to zero, and later in Section 4, the algo-

[7]The BS of an item i is $\frac{1}{n-1}\sum_{j\neq i} p_{i,j}$. When $p_{i,j} = 2/3$ for all $i \succ j$, the gap of the BSs between the best two items is $\Theta(n^{-1})$, and thus, the sample complexity to order them by BS algorithms (e.g., Active Ranking (Heckel et al., 2019) is $\Omega(n^2)$.

[8]Due to space limitation, all proofs in this paper are relegated to the supplementary material.

Subroutine 1 Distribute-Item (DI) $(i, v, \epsilon, s_u, s_d, \delta, S_{up}, S_{mid}, S_{down})$

```
1: Set $t_{max} := \lceil \frac{2}{\epsilon^2} \log \frac{4}{\delta} \rceil$, $\forall t \in \mathbb{Z}$, $b_t := \sqrt{\frac{1}{2t} \log \frac{\pi^2 t^2}{3\delta}}$;
2: $t \leftarrow 0$, and $w_0 \leftarrow 0$;
3: **repeat**
4:     $t \leftarrow t + 1$ and compare $i$ and $v$ once;
5:     if $i$ wins, $w_t \leftarrow w_{t-1} + 1$; otherwise $w_t \leftarrow w_{t-1}$;
6:     **if** $\frac{w_t}{t} - b_t > \frac{1}{2} + s_u$ **then**
7:         Add $i$ to $S_{up}$ and **return**;
8:     **else if** $\frac{w_t}{t} + b_t < \frac{1}{2} - s_d$ **then**
9:         Add $i$ to $S_{down}$ and **return**;
10:    **end if**
11: **until** $t = t_{max}$;
12: **if** $\frac{w_{t_{max}}}{t_{max}} > \frac{1}{2} + \frac{1}{2}\epsilon + s_u$ **then**
13:     Add $i$ to $S_{up}$;
14: **else if** $\frac{w_{t_{max}}}{t_{max}} < \frac{1}{2} - \frac{1}{2}\epsilon - s_d$ **then**
15:     Add $i$ to $S_{down}$;
16: **else**
17:     Add $i$ to $S_{mid}$;
18: **end if**
```

Algorithm 2 Epsilon-Quick-Select(S, k, ϵ, δ) (EQS)

```
1: Randomly pick an item from $S$ and denote it by $v$;
2: $S_{up}, S_{down} \leftarrow \emptyset$; $S_{mid} \leftarrow \{v\}$; $\delta_1 \leftarrow \frac{\delta}{|S|(|S|-1)}$;
3: **for** item $i$ in $S$ and $i \neq j$ **do**
4:     DI$(i, v, \frac{\epsilon}{2}, 0, 0, \delta_1, S_{up}, S_{mid}, S_{down})$.
5: **end for**
6: **if** $|S_{up}| > k$ **then**
7:     **return** EQS$(S_{up}, k, \epsilon, \frac{(n-1)\delta}{n})$;        # $n = |S|$.
8: **else if** $|S_{up}| + |S_{mid}| \geq k$ **then**
9:     **return** $S_{up} \cup (k - |S_{up}|)$ random items of $S_{mid}$;
10: **else**
11:     $k' \leftarrow k - |S_{up}| - |S_{mid}|$;
12:     **return** $S_{up} \cup S_{mid} \cup$ EQS$(S_{down}, k', \epsilon, \frac{(n-1)\delta}{n})$;
13: **end if**
```

rithms for exact k-selection will also call DI as a subroutine. Lemma 3 states the theoretical performance of DI, and Theorem 4 states the theoretical performance of EQS.

Lemma 3 (Theoretical Performance of DI). *DI terminates after at most $O(\epsilon^{-2} \log \delta^{-1})$ comparisons, and with probability at least $1 - \delta$, one the following five events happens: (i) $p_{i,v} \geq 1/2 + \epsilon + s_u$ and item i is added to S_{up}; (ii) $p_{i,v} \in (1/2 + s_u, 1/2 + \epsilon + s_u)$ and item i is not added to S_{down}; (iii) $p_{i,v} \in [1/2 - s_d, 1/2 + s_u]$ and item i in added to S_{mid}; (iv) $p_{i,v} \in (1/2 - \epsilon - s_d, 1/2 - s_d)$ and item i is not added to S_{up}; and (v) $p_{i,v} \leq 1/2 - \epsilon - s_d$ and item i is added to S_{down}.*

Theorem 4 (Theoretical Performance of EQS). *Given an input set S with $|S| = n$, $1 \leq k \leq n/2$, and $\epsilon, \delta \in (0, 1/2)$, EQS$(S, k, \epsilon, \delta)$ terminates after $O(n\epsilon^{-2} \log(n/\delta))$ number of comparisons in expectation, and with probability at least $1 - \delta$, returns an (ϵ, k)-optimal subset of S.*

3.2.2. STEP TWO: TOURNAMENT-k-SELECTION

In this section, we use EQS to develop a PAC k-selection algorithm with sample complexity $O(n\epsilon^{-2} \log(k/\delta))$. The algorithm runs like a tournament and consists of rounds. At each round t, we split the remaining items (use R_t to denote the set of the remaining items at the beginning of round t) into subsets with size around $2k$, and for each subset we use EQS to find an (ϵ_t, k)-optimal subset with confidence $1 - \delta_t/k$. We then keep the items in these (ϵ_t, k)-optimal subsets, and remove all the other items. We can show that with probability at least $1 - \delta_t$, the items kept in round t (i.e., R_{t+1}) contain an (ϵ_t, k)-optimal subset of R_t, which implies that for any t, R_{t+1} contains a subset U_{t+1} such that for any item i in U_{t+1} and item j in $R_t - U_{t+1}$, $p_{i,j} \geq 1/2 - \epsilon_t$. We can also show that with probability at least $1 - \delta_t - \delta_{t-1}$, for any item i in U_{t+1} and j in $R_{t-1} - U_{t+1}$, $p_{i,j} \geq 1/2 - \epsilon_t - \epsilon_{t-1}$. Repeating this, we can show that with probability at least $1 - \sum_{r=1}^{t} \delta_r$, for any item i in U_{t+1} and item j in $[n] - U_{t+1}$, $p_{i,j} \geq 1/2 - \sum_{r=1}^{t} \epsilon_r$. Thus, by repeating the rounds until only k items remain, we have that with probability at least $1 - \sum_{t=1}^{\infty} \delta_t$, for any item i in the returned set and j not in the returned set, $p_{i,j} \geq 1/2 - \sum_{t=1}^{\infty} \epsilon_t$, which implies that the returned set is a $(\sum_{t=1}^{\infty} \epsilon_t, k)$-optimal subset of $[n]$. Choosing $\sum_{t=1}^{\infty} \epsilon_t \leq \epsilon$ and $\sum_{t=1}^{\infty} \delta_t \leq \delta$, we can get that with probability at least $1 - \delta$, the returned set is an (ϵ, k)-optimal subset of $[n]$. The algorithm is described in Algorithm 3, and its theoretical performance is stated in Theorem 5.

Algorithm 3 Tournament-k-Selection$([n], k, \epsilon, \delta)$ (TKS)

```
1: For any $t \in \mathbb{Z}^+$, set $\epsilon_t := \frac{1}{4}(\frac{4}{5})^t$ and $\delta_t := \frac{6\delta}{\pi^2 t^2}$;
2: Initialize $t \leftarrow 0$, $R_1 \leftarrow [n]$;
3: **repeat**
4:     $t \leftarrow t + 1$;
5:     Split $R_t$ into $m_t = \lceil \frac{|R_t|}{2k} \rceil$ sets $(S_{t,i}, i \in [m_t])$,
       where $\forall i \in [m_t]$, $|S_{t,i}| \leq 2k$;
6:     **for** $i \in [m_t]$ **do**
7:         $A_{t,i} \leftarrow$EQS$(S_{t,i}, \min\{k, |S_{t,i}|\}, \epsilon_t, \frac{\delta_t}{k})$;
8:     **end for**
9:     $R_{t+1} \leftarrow A_{t,1} \cup A_{t,2} \cup \cdots \cup A_{t,m_t}$;
10: **until** $|R_{t+1}| = k$;
11: **return** $R_{t+1}$;
```

Theorem 5 (Theoretical Performance of TKS). *Given input $1 \leq k \leq n/2$, and $\epsilon, \delta \in (0, 1/2)$, TKS terminates after $O(n\epsilon^{-2} \log(k/\delta))$ number of comparisons in expectation, and with probability at least $1 - \delta$, returns an (ϵ, k)-optimal subset of $[n]$.*

Remark. i) The sample complexity upper bound of TKS matches the lower bound stated in Theorem 2 up to a constant factor. Thus, in order sense, our upper and lower bounds for PAC k-selection are tight. ii) When $k = 1$, our upper bound is the same as that of Falahatgar et al. (2017a;b). We note that the algorithms given by Falahatgar et al. (2017a;b) only work for $k = 1$, and it is not obvious how to generalize them to cases with general k-values.

4. Exact k-Selection

4.1. Lower Bound

In this subsection, we prove a lower bound for the exact k-selection problem. We note that the sample complexity lower bound not only depends on the gaps between items i and items r_k or r_{k+1} as in PEMAB problems (e.g., (Jamieson et al., 2014; Chen et al., 2017)), but also depends on other comparisons probabilities. In fact, even if the values of Δ_i's are the same, different instances may have different lower bounds on the sample complexity for finding the best-k items. For some instances, even the $\Omega(\Delta_i^{-2})$ lower bound for ordering two items stated in Theorem 6 and Ren et al. (2019) may not hold if there are more than two items. For instance, Example 13 in Ren et al. (2019) states an instance with three items such that $O(\Delta_{r_1,r_2}^{-1}\log(\Delta_{r_1,r_2}^{-1}\delta^{-1}))$ comparisons are sufficient to find the best item with probability $1-\delta$, which indicates the difficulty in finding an instance-wise lower bound for all instances.

Thus, in this paper, we prove a lower bound for a specific model: Thurstone's model. In Thurstone's model, each item i holds a real number θ_i representing the users' preference for this item. We name these numbers as scores. The higher the score, the more preferred the item, and thus, the scores imply a true order of these items. Under Thurstone's model with variance σ^2, for any two items i and j, we have

$$p_{i,j} = \mathbb{P}\{\theta_i + Z_1 > \theta_j + Z_2\} = \frac{1}{\sqrt{4\pi\sigma^2}}\int_{-\infty}^{\theta_i-\theta_j} e^{-\frac{x^2}{4\sigma^2}}\,\mathrm{d}x,$$

where Z_1 and Z_2 are two independent Gaussian$(0,\sigma^2)$ random variables. The definitions of the gaps $\Delta_{i,j}$'s and Δ_i's remain the same as in Section 1.2. It can be verified that Thurstone's model satisfies SST and STI. Under Thurstone's model, we prove the following lower bound for exact k-selection, which can be viewed as a worst-instance lower bound. Here, the worst-instance lower bound means that under the same values of gaps δ_i's, the lower bound for the Thurstone's model is no higher than the actual worst-instance lower bound. In the proof, we invoke the results shown by Jamieson et al. (2014); Chen et al. (2017).

Theorem 6 (Lower bound for exact k-selection under Thurstone's model)**.** *Under Thurstone's model with variance one, given $\delta \in (0, 1/100)$, n items with scores $\theta_1, \theta_2, ..., \theta_n \in [0, 1]$, and $1 \le k \le n/2$, to find the best-k items with probability at least $1-\delta$, any algorithm must conduct at least $\Omega(\sum_{i\in[n]}[\Delta_i^{-2}\log\delta^{-1}] + \log\log\Delta_{r_k}^{-1})$ number of comparisons in expectation.*

4.2. Algorithm for Best Item Selection

We first use the PAC algorithm TKS to establish a best item selection algorithm called Sequential-Elimination-Exact-Best-Selection (SEEBS). SEEBS runs in rounds. In each round t, it chooses a threshold α_t, uses TKS to choose a PAC best item v_t with error tolerance $\alpha_t/3$, and uses DI to identify items i with $p_{i,r_1} \le 1/2 - \alpha_t$ and removes them. By choosing a proper confidence δ_t for each round t, the properties of DI and TKS stated in Lemma 2 and Theorem 3 guarantee that with probability at least $1-\delta$, the best item r_1 will not be removed. If α_t is diminishing so that $\lim_{t\to\infty}\alpha_t = 0$ and the confidences satisfy $\sum_{t=1}^{\infty}\delta_t \le \delta$, the algorithm will, with probability at least $1-\delta$, discard all items other than r_1 and keep the best item r_1. TKS is described in Algorithm 4, and its theoretical performance is stated in Theorem 7.

Algorithm 4 Sequential-Elimination-Exact-Best-Selection $([n], \delta)$ (SEEBS)

```
 1: For all $t \in \mathbb{Z}^+$, set $\alpha_t := 2^{-t}$ and $\delta_t := \frac{6\delta}{\pi^2 t^2}$;
 2: Initialize $t \leftarrow 1$, $R_1 \leftarrow [n]$;
 3: repeat
 4:     $\{v_t\} \leftarrow$TKS$(R_t, 1, \frac{\alpha_t}{3}, \frac{2\delta_t}{3})$;
 5:     $S_{up} \leftarrow \emptyset$, $S_{mid} \leftarrow \{v_t\}$, $S_{down} \leftarrow \emptyset$;
 6:     for items $i$ in $R_t - \{v_t\}$ do
 7:         DI$(i, v_t, \frac{\alpha_t}{3}, 0, \frac{\alpha_t}{3}, \frac{\delta_t}{3}, S_{up}, S_{mid}, S_{down})$;
 8:     end for
 9:     $R_{t+1} \leftarrow R_t - S_{down}$;
10:     $t \leftarrow t + 1$;
11: until $|R_t| = 1$
12: return the only item in $R_t$;
```

Theorem 7 (Theoretical Performance of SEEBS)**.** *With probability at least $1-\delta$, SEEBS terminates after $O(\sum_{i\ne r_1}[\Delta_i^{-2}(\log\delta^{-1} + \log\log\Delta_i^{-1})])$ number of comparisons in expectation and returns the best item in $[n]$.*

Remark. i) According to the lower bound stated in Theorem 6, SEEBS is worst-instance optimal up to a loglog factor. If Δ_i's are not too small, the term $\log\log\Delta_i^{-1}$ will be dominated by $\log\delta^{-1}$, i.e., if $\Delta_i^{-1} \le e^{1/\delta}$, then our upper bound is worst-instance optimal up to a constant factor. ii) The phrase "in expectation" in Theorem 7 does not only come from the sample complexity of TKS, but also comes from the choice of input confidences of DI. At each round t, by inputting $\delta_t/3$ to DI, one cannot guarantee that the executions of DI correctly assign all non-best items i with $p_{i,r_1} \le 1/2 - \alpha_t$ to S_{down} with probability $1-\delta_t$, and thus, more rounds may be needed to remove these non-best items. Therefore, in expectation, the number of comparisons over item i is upper bounded by $O(\Delta_i^{-2}(\log\delta^{-1}+\log\log\Delta_i^{-1}))$.

4.3. Algorithm for Best-k Items Selection

In this subsection, we develop an exact best-k items selection algorithm called Sequential-Elimination-Exact-k-Selection (SEEKS). The basic idea of SEEKS is similar to SEEBS. SEEKS runs in rounds. At each round t, it calls TKS and TKS2 (where TKS2 is almost the same as TKS except that it finds the PAC worst items) to find a pivot v_t such that $\Delta_{v_t,r_k}\le\alpha_t/3$. Then it uses DI to distribute the items such that with probability at least $1-\delta_t$, (i) all items i with $p_{i,r_k}\ge 1/2+\alpha_t$ are added to S_{t+1}; (ii) all items i with $p_{i,r_k}\le 1/2-\alpha_t$ are discarded (i.e., not added to S_{t+1} or R_{t+1}); (iii) none of the items with $p_{i,r_k}\ge 1/2$ is discarded; and (iv) all items added to S_{t+1} are of the best-k items. By choosing proper confidence δ_t for each round t, we guarantee that with probability at least $1-\delta$, none of the best-k items is discarded, and all items added to S_{t+1} are of the best-k items. Thus, with probability at least $1-\sum_{t=1}^{\infty}\delta_t=1-\delta$, in all rounds, none of the best items is discarded, and S_t only contains the best-k items. When $|S_t|\le k$ or $|S_t\cup R_t|\le k$, the algorithm terminates, and thus, if the algorithm returns, with probability at least $1-\delta$, it returns the set of the best-k items. Since $\lim_{t\to\infty}\alpha_t=0$, there is a large enough t such that either all of the best-k items have been added to some S_t, or all items except the best-k are discarded. Therefore, the algorithm terminates in finite time. The sample complexity follows from the choice of α_t's and δ_t's. SEEKS is described in Algorithm 5. Its theoretical performance is stated in Theorem 8.

Algorithm 5 Sequential-Elimination-Exact-k-Selection ($[n],k,\delta$) (SEEKS)

```
1: For all $t\in\mathbb{Z}^+$, set $\alpha_t:=2^{-t}$ and $\delta_t:=\frac{6\delta}{\pi^2t^2}$;
2: Initialize $t\leftarrow 1$, $R_1\leftarrow[n]$, $S_1\leftarrow\emptyset$, $k_1\leftarrow k$;
3: repeat
4:     $A_t\leftarrow$TKS$(R_t,k_t,\frac{\alpha_t}{3},\frac{\delta_t}{3})$;
5:     $\{v_t\}\leftarrow$TKS2$(A_t,1,\frac{\alpha_t}{3},\frac{\delta_t}{3})$
6:     $S_{up}\leftarrow\emptyset$, $S_{mid}\leftarrow\{v_t\}$, $S_{down}\leftarrow\emptyset$;
7:     for items $i$ in $R_t-\{v_t\}$ do
8:         DI$(i,v_t,\frac{\alpha_t}{3},\frac{\alpha_t}{3},\frac{\alpha_t}{3},\frac{\delta_t}{3(|R_t|-1)},S_{up},S_{mid},S_{down})$;
9:     end for
10:    $S_{t+1}\leftarrow S_t\cup S_{up}$;
11:    $R_{t+1}\leftarrow R_t-S_{up}-S_{down}$;
12:    $k_{t+1}\leftarrow k_t-|S_{up}|$;
13:    $t\leftarrow t+1$;
14: until $|S_t|\ge k$ or $|S_t\cup R_t|\le k$
15: return $S_t\cup\{k-|S_t|$ items in $R_t\}$;
```

Theorem 8 (Theoretical Performance of SEEKS)**.** *With probability at least $1-\delta$, SEEKS terminates after $O(\sum_{i\in[n]}[\Delta_i^{-2}(\log(n/\delta)+\log\log\Delta_i^{-1})])$ number of comparisons in expectation, and returns the best-k items.*

Remark. i) According to the lower bound stated in Theorem 6, SEEKS is worst-instance optimal up to a log factor. We conjecture that the true lower bound and upper bound of the exact k-selection depend on $\log(k/\delta)$, just as that of the PAC k-selection, but it remains an open problem for future studies. ii) Different from Theorem 4, the phrase "in expectation" in Theorem 8 comes from the sample complexity of TKS (stated in Theorem 5). If one can find a PAC k-selection algorithm that uses no more than $O(n\epsilon^{-2}\log(n/\delta))$ comparisons with probability $1-\delta$, then by replacing TKS and TKS2 with this algorithm, we can remove "in expectation" in Theorem 8.

5. Numerical Results

In this section, we perform experiments on the synthetic dataset with equal noise-levels (i.e., $\Delta_{i,j}$ is a constant) and public election datasets provided by PrefLib (Mattei and Walsh, 2013). In the supplementary material, we present the results of the synthetic dataset with unequal noise-levels and the numerical illustrations of the growth rates of the exact best-k items selection bounds. The codes and datasets can be found in our GitHub page.[9]

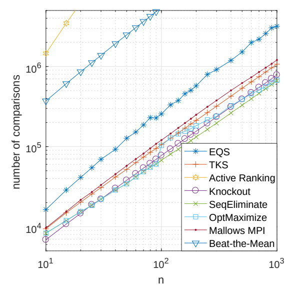
5.1. Numerical Results on Synthetic Data

In this subsection, we provide numerical simulations for our algorithms and those in related works under equal noise levels, i.e., we set $p_{i,j}=0.6$ for all items i and j with $i\succ j$. This dataset has also been used in previous works (Yue and Joachims, 2011; Busa-Fekete et al., 2014; Falahatgar et al., 2017a;b; 2018). The results are presented in Figure 1, and every data point of it is averaged over 100 independent trials.

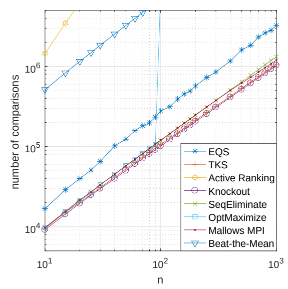
5.1.1. PAC Best Item Selection

For PAC best item selection, the algorithms we compare with our EQS and TKS algorithms are: i) Knockout (Falahatgar et al., 2017b), ii) Seq-Eliminate (Falahatgar et al., 2017a), iii) Opt-Maximize (Falahatgar et al., 2017a), iv) Active Ranking (Heckel et al., 2019), v) Beat-the-Mean (Yue and Joachims, 2011), and vi) MallowsMPI (Busa-Fekete et al., 2014). Knockout and Opt-Maximize are two PAC best item selection algorithms, and their sample complexities are upper bounded by $O(n\epsilon^{-2}\log\delta^{-1})$, which is of the same order as TKS. Seq-Eliminate and Beat-the-Mean are also PAC best item selection algorithms, but their sample complexities are $O(n\epsilon^{-2}\log(n/\delta))$, higher than that of TKS by a log factor. Active Ranking (Heckel et al., 2019) and MallowsMPI are exact selection algorithms with sample

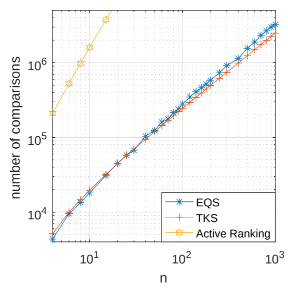
[9]https://github.com/WenboRen/Topk-Ranking-from-Pairwise-Comparisons.git



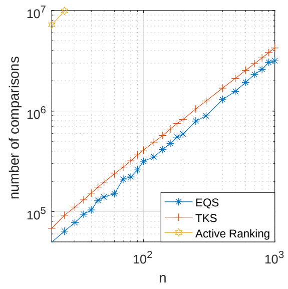
(a) PAC best one selection with $\epsilon = 0.08$ and $\delta = 0.01$.



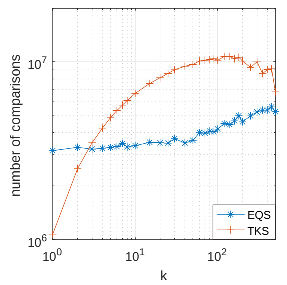
(b) PAC best one selection with $\epsilon = 0.001$ and $\delta = 0.01$.



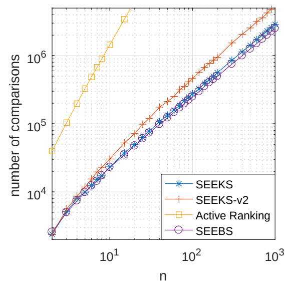
(c) PAC k-selection with $k = 2$, $\epsilon = 0.08$, and $\delta = 0.01$.



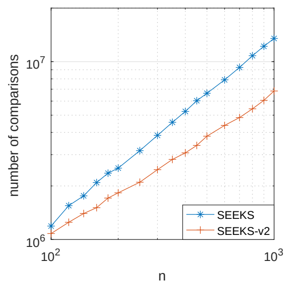
(d) PAC k-Selection with $k = 4$, $\epsilon = 0.08$, and $\delta = 0.01$.



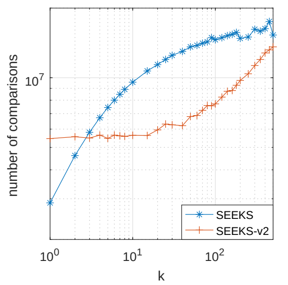
(e) PAC k-selection with $n = 1000$, $\epsilon = 0.08$, and $\delta = 0.01$.



(f) Exact k-selection with $k = 1$ and $\delta = 0.01$.



(g) Exact k-selection with $k = 50$ and $\delta = 0.01$.



(h) Exact k-selection with $n = 1000$ and $\delta = 0.01$.

Figure 1. Numerical results on the equal noise-level dataset, i.e., $p_{i,j} = 0.6$ for any items $i \succ j$.

complexity $O(n \log(n/\delta))$.

The numerical results are summarized in Figure 1 (a) (b). We set $\delta = 0.01$, and examine how the number of comparisons conducted increases with n. In Figure 1 (a), we set $\epsilon = 0.08$, and in Figure 1 (b), we set $\epsilon = 0.001$.

According to the illustrated results, we can see that when ϵ is small (i.e., $\epsilon = 0.001$), the performance of our algorithm TKS is almost the same as those of Knockout and MallowsMPI, the best of previous works. We note that Knockout and MallowsMPI are only designed for best item selection and it is not obvious how to extend them to cases with $k > 1$. Thus, although our TKS works for all values of k, its performance is close to the best of the state-of-the-art when $k = 1$.

5.1.2. PAC k-Selection

For the PAC k-selection, we provide the simulation results for EQS, TKS, and Active Ranking.

The results are summarized in Figure 1 (c)-(e). In Figure 1 (c)-(d), we set $\epsilon = 0.08$ and $\delta = 0.01$, vary the values of n, and compare EQS of TKS with $k = \{2, 4\}$. In Figure 1 (e), we set $\epsilon = 0.08, \delta = 0.01$, and $n = 1000$, and compare EQS and TKS with different values of k.

As presented in Figure 1 (c)-(e), we can see that when k is small (i.e., $k \leq 2$), TKS outperforms EQS, but when k is not too small, EQS uses fewer comparisons. The sample complexity upper bound of TKS is $O(n\epsilon^{-2} \log(k/\delta))$, which is lower than the $O(n\epsilon^{-2} \log(n/\delta))$ complexity of EQS. However, in practice, for most values of k, EQS consumes fewer comparisons. One explanation is that the constant factor of TKS is larger than that of EQS. There may be two reasons: First, in each call of EQS on S, the sub-call of EQS is executed on S_{up} or S_{down}, whose expected sizes are less than $|S|/2$, while in TKS, each iteration removes no more than a half of the items. Second, in TKS, the value ϵ_t input to DI is less than ϵ, which is used in EQS.

5.1.3. Exact k-Selection

For the exact k-selection algorithm, we only provide numerical results for the algorithms proposed in this paper: SEEBS, SEEKS, and SEEKS-v2, a variation of SEEKS. Here, SEEKS-v2 is almost the same as SEEKS. But in Line 4, TKS is replaced with EQS, since EQS has a better empirical performance than TKS when k is not too small. We note that the sample complexity upper bound of SEEKS-v2 is of the the same order as SEEKS (ignoring constant factors). We do not compare the algorithm proposed by Chen et al. (2018) because it is unclear how to choose the parameters to let the confidence be $1-\delta$. We do not compare the algorithm given by Saha and Gopalan (2019b) since it

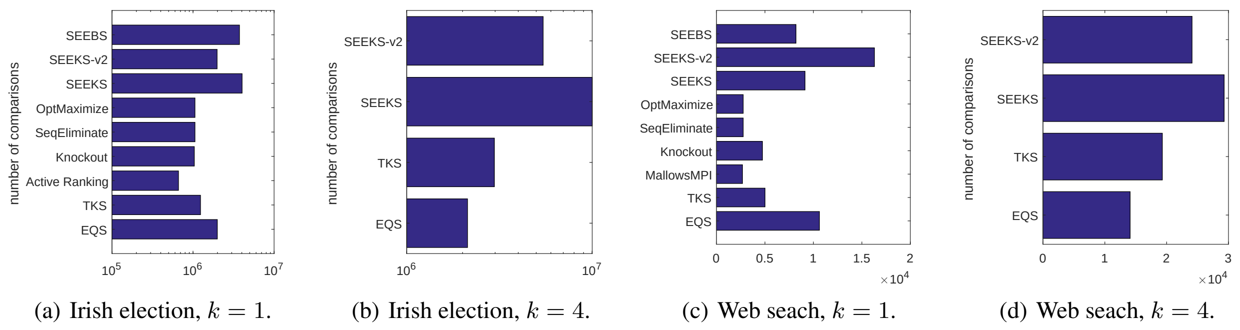


Figure 2. Numerical results on public election datasets. MallowsMPI is not in (1) because its correct probability does not reach $1 - \delta$ for the Irish election dataset. Beat-the-mean and Active Ranking are not in some subfigures because they do not return in a reasonable time.

requires the system to be able to conduct comparisons over more than two items, which is not assumed in this paper.

In Figure 1 (f), we compare SEEBS, SEEKS, and SEEKS-v2 with $k = 1$ and $\delta = 0.01$. In Figure 1 (g), we fix $k = 50$ and $\delta = 0.01$, vary n, and compare the two versions of SEEKS. In Figure 1 (h), we fix $n = 1000$ and $\delta = 0.01$, vary k, and compare the two versions of SEEKS.

From Figure 1 (f), we can see that SEEBS is slightly better than SEEKS, which is due to the choices of confidences input to the calls of DI in these two algorithms. Also, we can see that SEEBS and SEEKS are better than SEEKS-v2, especially when n is large. This is because the empirical performance of EQS is worse than TKS when $k = 1$. According to Figure 1 (g) and (h), SEEKS-v2 consumes fewer comparisons when k is not too small. An explanation is that in practice, EQS uses fewer comparisons than TKS when k is not too small.

5.2. Numerical Results on Public Election Data

In this subsection, we perform numerical experiments on public election datasets provided in PrefLib (Mattei and Walsh, 2013). To be specific, we use the Irish election dataset "ED-00001-00000001.pwg" (Lu and Boutilier, 2011) and the clean web search dataset "ED-00015-00000047.pwg" (Betzler et al., 2014). Both datasets are included in the supplementary material.

The Irish Election dataset contains $n = 12$ candidates and 43,942 votes on them. The web search dataset contains $n = 28$ pages and 1134 samples of pairwise preferences on them. For every pair of items i and j in each dataset, the dataset records the number of votes or samples $N_{i,j}$ that show preference on item i to item j. From these records, we extract $p_{i,j} := N_{i,j}/(N_{i,j} + N_{j,i})$ for any two items i and j. We note that these two dataset do not satisfy the SST or the STI and do not imply a strict order. Thus, we use the Borda-Scores for them to get the true rankings.

In the experiments, we set $\epsilon = 0.001$, $\delta = 0.01$, and $k = \{1, 4\}$. Surprisingly, although these two datasets do not satisfy SST or STI, our algorithms EQS, TKS, SEEBS, and SEEKS can still return correct results with correct probability at least $1 - \delta$ (in the experiments, all runs of them return correct results). In fact, we have done experiments on more datasets and find that if there is a small number $\gamma > 1$ (e.g., $\gamma < 5$) such that for any $i \succ j \succ k$, $p_{i,k} \geq \gamma^{-1} \max\{p_{i,j}, p_{j,k}\}$ and $\Delta_{i,k} \leq \gamma(\Delta_{i,j} + \Delta_{j,k})$, then our algorithms can guarantee at least $1 - \delta$ correct probability.

From the results presented in Figure 2, we can see that for the Irish election dataset, the performances of our algorithms EQS and TKS are close to the best of the previous works, which indicates that even if they are not designed for $k = 1$ and these types of datasets, they still have promising performances on some real-world datasets. The results also show positive evidence on our theoretical results, i.e., TKS (SEEKS) performs better than EQS (SEEKS-v2) when k is small ($k = 1$) and performs worse when k is large ($k = 4$).

6. Conclusion

This paper studied the sample complexity bounds for selecting the PAC or exact best-k items from pairwise comparisons. For PAC k-selection, we first proved an $\Omega(n\epsilon^{-2}\log(k/\delta))$ lower bound, and then proposed an algorithm with expected sample complexity $O(n\epsilon^{-2}\log(k/\delta))$, which implies that both our upper bound and lower bound are tight up to a constant factor. For exact k-selection, we first proved a worst-instance lower bound, and then proposed an algorithm for $k = 1$ that is optimal up to a loglog factor. Finally, we proposed an algorithm for general k-values that is optimal up to a log factor. The numerical results in this paper also confirm our theoretical results.

Acknowledgements

This work has been supported in part by NSF grants CAREER CNS-1943226, CNS-1901057, CNS-1758757, CNS-1719371, CNS-1717060, ECCS-1818791, and CCF-1758736, a Google Faculty Research Award, and an IITP grant (No. 2017-0-00692).

We express our sincere gratitude to those who fought or are fighting against COVID-19.

References

Agarwal, A., Agarwal, S., Assadi, S., and Khanna, S. (2017). Learning with limited rounds of adaptivity: Coin tossing, multi-armed bandits, and ranking from pairwise comparisons. In *Conference on Learning Theory*, pages 39–75.

Betzler, N., Bredereck, R., and Niedermeier, R. (2014). Theoretical and empirical evaluation of data reduction for exact kemeny rank aggregation. *Autonomous Agents and Multi-Agent Systems*, 28(5):721–748.

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

Braverman, M., Mao, J., and Peres, Y. (2019). Sorted top-k in rounds. In *Conference on Learning Theory*, pages 342–382. PMLR.

Busa-Fekete, R., Hüllermeier, E., and Szörényi, B. (2014). Preference-based rank elicitation using statistical models: The case of mallows.

Chen, L., Li, J., and Qiao, M. (2017). Towards instance optimal bounds for best arm identification. In *Conference on Learning Theory*, pages 535–592.

Chen, X., Bennett, P. N., Collins-Thompson, K., and Horvitz, E. (2013). Pairwise ranking aggregation in a crowdsourced setting. In *ACM international conference on Web search and data mining*, pages 193–202. ACM.

Chen, X., Li, Y., and Mao, J. (2018). A nearly instance optimal algorithm for top-k ranking under the multinomial logit model. In *Annual ACM-SIAM Symposium on Algorithms*, pages 2504–2522. SIAM.

Chen, Y., Fan, J., Ma, C., and Wang, K. (2019). Spectral method and regularized MLE are both optimal for top-k ranking. *Annals of statistics*, 47(4):2204.

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

Falahatgar, M., Hao, Y., Orlitsky, A., Pichapati, V., and Ravindrakumar, V. (2017a). Maxing and ranking with few assumptions. In *Advances in Neural Information Processing Systems*, pages 7060–7070.

Falahatgar, M., Jain, A., Orlitsky, A., Pichapati, V., and Ravindrakumar, V. (2018). The limits of maxing, ranking, and preference learning. In *International Conference on Machine Learning*, pages 1427–1436. PMLR.

Falahatgar, M., Orlitsky, A., Pichapati, V., and Suresh, A. T. (2017b). Maximum selection and ranking under noisy comparisons. In *International Conference on Machine Learning*, pages 1088–1096. JMLR. org.

Feige, U., Raghavan, P., Peleg, D., and Upfal, E. (1994). Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018.

Heckel, R., Shah, N. B., Ramchandran, K., Wainwright, M. J., et al. (2019). Active ranking from pairwise comparisons and when parametric assumptions do not help. *The Annals of Statistics*, 47(6):3099–3126.

Heckel, R., Simchowitz, M., Ramchandran, K., and Wainwright, M. J. (2018). Approximate ranking from pairwise comparisons. In *International Conference on Artificial Intelligence and Statistics*, pages 1057–1066.

Hoare, C. A. (1961). Algorithm 65: find. *Communications of the ACM*, 4(7):321–322.

Hunter, D. R. (2004). MM algorithms for generalized bradley-terry models. *The annals of statistics*, 32(1):384–406.

Jamieson, K., Malloy, M., Nowak, R., and Bubeck, S. (2014). lil'UCB: An optimal exploration algorithm for multi-armed bandits. In *Conference on Learning Theory*, pages 423–439.

Kalyanakrishnan, S., Tewari, A., Auer, P., and Stone, P. (2012). PAC subset selection in stochastic multi-armed bandits. In *International Conference on Machine Learning*, volume 12, pages 655–662.

Katariya, S., Jain, L., Sengupta, N., Evans, J., and Nowak, R. (2018). Adaptive sampling for coarse ranking. In *International Conference on Artificial Intelligence and Statistics*, pages 1839–1848.

Lu, T. and Boutilier, C. (2011). Budgeted social choice: From consensus to personalized decision making. In *International Joint Conference on Artificial Intelligence*.

Luce, R. D. (2012). *Individual choice behavior: A theoretical analysis*. Courier Corporation.

Mannor, S. and Tsitsiklis, J. N. (2004). The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5(Jun):623–648.

Mattei, N. and Walsh, T. (2013). Preflib: A library of preference data HTTP://PREFLIB.ORG. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT 2013)*, Lecture Notes in Artificial Intelligence. Springer.

McFadden, D. (1973). Conditional logit analysis of qualitative choice behavior.

Negahban, S., Oh, S., and Shah, D. (2017). Rank centrality: Ranking from pairwise comparisons. *Operations Research*, 65(1):266–287.

Park, D., Neeman, J., Zhang, J., Sanghavi, S., and Dhillon, I. (2015). Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *International Conference on Machine Learning*, pages 1907–1916.

Pfeiffer, T., Xi, A., Gao, A., Mao, Y., Chen, and Rand, D. G. (2012). Adaptive polling for information aggregation. In *Twenty-sixth AAAI Conference on Artificial Intelligence*.

Plackett, R. L. (1975). The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(2):193–202.

Ren, W., Liu, J., and Shroff, N. B. (2018). PAC ranking from pairwise and listwise queries: Lower bounds and upper bounds. *arXiv preprint arXiv:1806.02970*.

Ren, W., Liu, J., and Shroff, N. B. (2019). On sample complexity upper and lower bounds for exact ranking from noisy comparisons. In *Advances in Neural Information Processing Systems*, pages 10014–10024.

Saha, A. and Gopalan, A. (2019a). Active ranking with subset-wise preferences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3312–3321.

Saha, A. and Gopalan, A. (2019b). From PAC to instance-optimal sample complexity in the Plackett-Luce model. *arXiv preprint arXiv:1903.00558*.

Shah, N. B., Balakrishnan, S., Guntuboyina, A., and Wainwright, M. J. (2017). Stochastically transitive models for pairwise comparisons: Statistical and computational issues. *IEEE Transactions on Information Theory*, 63(2).

Shah, N. B. and Wainwright, M. J. (2017). Simple, robust and optimal ranking from pairwise comparisons. *The Journal of Machine Learning Research*, 18(1):7246–7283.

Szörényi, B., Busa-Fekete, R., Paul, A., and Hüllermeier, E. (2015). Online rank elicitation for Plackett-Luce: A dueling bandits approach. In *Advances in Neural Processing Systems*, pages 604–612.

Thurstone, L. L. (1927). A law of comparative judgment. *Psychological review*, 34(4):273.

Yue, Y. and Joachims, T. (2011). Beat the mean bandit. In *International Conference on Machine Learning*, pages 241–248.