Ranking with Multiple Oracles: From Weak to Strong Stochastic Transitivity

Tao Jin [*1] Yue Wu [*2] Quanquan Gu [†2] Farzad Farnoud [†1]

Abstract

We study the problem of efficiently aggregating the preferences of items from multiple information sources (oracles) and infer the ranking under both the weak stochastic transitivity (WST) and the strong stochastic transitivity (SST) conditions. When the underlying preference model satisfies the WST condition, we propose an algorithm named *Rank-with-Multiple-Oracles* (`RMO-WST`), which has a bi-level design: at the higher level, it actively allocates comparison budgets to all undetermined pairs until the full ranking is recovered; at the lower level, it attempts to compare the pair of items and selects the more accurate oracles simultaneously. We prove that the sample complexity of `RMO-WST` is $\widetilde{O}(N \sum_{i=2}^{N} H_{\sigma^{-1}(i),\sigma^{-1}(i-1)})$, where N is the number of items to rank, $H_{i,j}$ is a problem-dependent hardness factor for correctly comparing item i and item j, and $\sigma^{-1}(i)$ represents the i-th best item. We also provide a tight lower bound that matches the upper bound of approximate ranking under the WST condition, answering a previously open problem. Additionally, when the SST condition is satisfied, we propose an algorithm named `RMO-SST`, which can achieve a sample complexity of $\widetilde{O}(\sum_{i=1}^{N} H_i \log(N))$, where $H_i = \max_{j\in[N],j\neq i} H_{i,j}$. This outperforms the best-known sample complexity by a factor of $\log(N)$. The theoretical advantages of our algorithms are verified by empirical experiments in a simulated environment.

[*]Equal contribution [†] Co-corresponding Authors [1]Department of Computer Science, University of Virginia [2]Department of Computer Science, University of California, Los Angeles. Correspondence to: Quanquan Gu <qgu@cs.ucla.edu>, Farzad Farnoud <farzad@virginia.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

1. Introduction

The problem of learning to rank with multiple oracles, sometimes known as rank aggregation, is ubiquitous in many applications, such as the recommendation system (Oliveira et al., 2020; Bałchanowski and Boryczka, 2023), user classification (Minka et al., 2018; Ebtekar and Liu, 2021) and drug discovery (Agarwal et al., 2010; Ru et al., 2021). More recently, learning-to-rank has regained attention in the context of evaluating large language models (LLM). For example, human feedback is used to rank the performance of LLMs across different tasks. Each task can be seen as an oracle, where two LLMs generate responses to a given prompt for that task, and the human annotator indicates which one is more helpful or informative.

In many applications, the oracles that provide preference feedback are usually human annotators, who may provide inherently noisy feedback. Moreover, oracles may show varying accuracy for different pairs of responses. For example, the performance gap between two LLMs may be negligible in one task but quite obvious in another. One natural question is then how to identify the underlying ranking of different LLMs efficiently, especially by taking advantage of those more accurate oracles (i.e., tasks). For example, when creating an LLM leaderboard, we have two different criteria or "tasks": honesty and helpfulness. When ranking two LLMs, we may ask a human annotator to rate which LLM is more honest and which LLM is more helpful. The two tasks/criteria may exhibit different preference behaviors.

Our goal is to accurately determine the ranking with as few queries as possible. To ensure the problem is well defined and an underlying ranking exists, it is commonly postulated (Falahatgar et al., 2017; Ren et al., 2018; Saha and Gopalan, 2019) that a ranking exists and it is harder to differentiate between items that rank in adjacent compared to those that rank apart. Formally speaking, if $i \succ j \succ k$, then $p_{i,k} \geq \max\{p_{i,j}, p_{j,k}\} > \frac{1}{2}$, where $p_{i,j}$ denotes the winning probability of i over j. This principle is known as *Strong Stochastic Transitivity (SST)*. Notably, Saad et al. (2023) solves the multi-oracle learning-to-rank tasks under a special case of the SST condition. In particular, they assume numerical feedback with sub-Gaussian noise, which satisfies the SST condition. Saad et al. (2023) established tight results for comparing two items, and directly swapped it with the deterministic comparison in a classic binary insertion sort algorithm to establish an upper bound for the noisy ranking problem.

Nevertheless, the application of SST can be overly restrictive in contexts where preference probabilities do not hinge on a single numerical attribute. Items can possess multifaceted attributes, leading people to evaluate different pairs based on different attributes. For example, LLM A is favored over LLM B because A produces longer responses, although both are equally informative. LLM B is preferred over LLM C because B's response is short and informative, while C's response is long and less informative. While human annotators can identify $A \succ B$ and $B \succ C$ easily, they may find it difficult to compare A and C for the long responses from both LLMs.

This phenomenon is pervasive in human behaviors and is defined as *Weak Stochastic Transitivity (WST)* (Feige et al., 1994; Mohajer et al., 2017; Falahatgar et al., 2018; Lou et al., 2022). It only assumes that if item $i \succ j$, then $p_{i,j} > \frac{1}{2}$. Therefore, a pair that is closely ranked might not always be more challenging to compare than one with a wider disparity. Driven by these real-world scenarios, this paper mainly focuses on the challenge of identifying the complete ranking of N items in a broader context, where only WST is applicable and SST is not a requirement. Our primary goal is to minimize the number of comparisons while ensuring a high level of confidence.

In this paper, we study the problem of learning-to-rank with multiple oracles, where a set of oracles provides noisy pairwise comparisons for the items. We propose an algorithm that queries comparisons for pairs of items from an adaptively chosen user set. We summarize our contributions as follows:

1. We study learning-to-rank with multiple preference oracles. We assume all oracles obey the same underlying ranking but demonstrate various accuracy and there is no 'best' oracle for all pairs of comparisons. We propose an algorithm named *Rank-with-Multiple-Oracles* (RMO-WST) that can efficiently rank N candidates given that each oracle satisfies the weak stochastic transitivity (WST) condition. The achieved sample complexity upper bound is $\widetilde{O}(N \sum_{i=2}^{N} H_{\sigma^{-1}(i),\sigma^{-1}(i-1)})$, where N is the number of items to rank, $H_{i,j}$ is a problem-dependent hardness factor for correctly comparing item i and item j, and $\sigma^{-1}(i)$ represents the i-th best item.

2. We provide a lower bound on the query complexity of solving multi-oracle ranking under the WST condition. In particular, Lou et al. (2022) studied the single-oracle ranking under the WST condition and conjectured that the lower bound is of order $\widetilde{\Omega}(N^2/\Delta^2)$, where $\Delta := \min_{(i,j)\in[N]\times[N]} |p_{i,j} - 1/2|$ is the gap of the comparison probability from $1/2$. Our lower bound answers this open question raised by Lou et al. (2022) and shows that our algorithm can attain the lower bound in terms of the number of candidates N.

3. We also propose an algorithm for learning to rank with multiple preference oracles in the presence of SST condition, namely RMO-SST. It can achieve a sample complexity of $\widetilde{O}(\sum_{i=1}^{N} H_i \log(N))$, where $H_i = \max_{j\in[N], j\neq i} H_{i,j}$. Compared with the existing result of Saad et al. (2023), our algorithm enjoys an improved sample complexity by a factor of $\log N$ and matches the lower bound of the sample complexity.

2. Related work

Rank aggregation with preference data. Rank aggregation generally refers to the combination of multiple stochastic pairwise or listwise comparison results into a ranking that is considered a consensus (Negahban et al., 2012; Thonet et al., 2022). Each aggregation method usually assumes an underlying model. Maystre and Grossglauser (2015) discovers the connection between the maximum likelihood estimate of the Plackett-Luce model with the stationary distribution of a Markov chain, vastly accelerating the inference time given the pairwise preference data. For general estimation that also contains listwise comparisons, maximum likelihood is a common practice (Vojnovic and Yun, 2016). For Mallow's model, a recent work proposed using a partition scheme for a ranked item to discover the ranker quality and produce a ranking simultaneously (Zhu et al., 2023).

Heterogeneous ranking and multi-task ranking. With the rise of crowdsourcing, data scientists are motivated to design algorithms adapted to this scenario to account for the variable quality of workers to achieve cost-efficient data acquisition rather than assigning tasks uniformly to workers regardless of their performance (Niu et al., 2015). When data is already given and the algorithm is unable to affect the collection process, due to the varying precision of the workers, a model that considers or estimates the quality of the source while ranking shows a significant benefit over those that do not (Takanobu et al., 2019; Jin et al., 2020). In practical cases, the data collection has not happened yet; then an adaptive algorithm can be chosen to optimize query collection. Existing methods usually maintain two sets of estimates: one for ranking and one for worker quality (Wu et al., 2022; Saad et al., 2023). Low-accuracy workers are usually gradually eliminated, leaving high-quality responses to be collected more efficiently. A low-rank assumption can be made when the similarity between two parties within a subset of tasks can also be extrapolated to a broader set of tasks. Methods derived from probability matrix factorization

Table 1: A comparison among the related works and the proposed method. The table is divided into two major sections. The upper section mainly shows the sample complexities of three related algorithms under the *SST* condition. The lower section of the table shows the result under the *WST* condition.

Algorithm	Sample Complexity	Multi-Oracle
`IIR` (Ren et al., 2019)	$O\Big(\sum_{i\in[N]} \Delta_i^{-2}\Big(\log\log\big(\Delta_i^{-1}\big)+\log(N/\delta)\Big)\Big)$	No
`Binary-Search` (Saad et al., 2023)	$\widetilde{O}\big(\sum_{i\in[N]} H_i\big(\log\log(H_i)+\log^2(N)+\log(1/\delta)\big)\big)$	Yes
`RMO-SST` (this work, Algorithm 7)	$\widetilde{O}\Big(\sum_{i\in[N]} H_i\Big(\log\log\big(H_i\big)+\log(N/\delta)\Big)\Big)$	Yes
`Probe-Max` (Lou et al., 2022)	$\widetilde{O}\Big(N\sum_{i=2}^{N} \Delta^{-2}_{\sigma^{-1}(i),\sigma^{-1}(i-1)}\Big)$	No
`RMO-WST` (this work, Algorithm 1)	$\widetilde{O}\Big(N\sum_{i=2}^{N} H_{\sigma^{-1}(i),\sigma^{-1}(i-1)}\Big)$	Yes
Lower Bound for WST (this work, Theorem 5.8)	$\widetilde{\Omega}\big(N^2 \min_i H_{\sigma^{-1}(i),\sigma^{-1}(i-1)}\big)$	Yes

are usually adopted in this case (Wang et al., 2016; Jun et al., 2019).

Active ranking. Online ranking from noisy comparisons requires a careful algorithm design to ensure that the estimations of the relative order of the two items are correct to ensure accuracy and make as few comparisons as possible. With the objective of ranking items according to their Borda score, the Active Ranking (AR) algorithm (Heckel et al., 2019) divides items into partitions according to the estimated score of uniformly sampling an opponent item. This sampling method matches directly that of the definition of Borda score. Partitions group items into the same set if their confidence interval overlaps. Afterwards, items in disjoint sets are considered correctly ordered with high confidence. With a slightly stronger imposed assumption of stochastic transitivity. Ren et al. (2019) propose the `Iterative-Insertion-Ranking` (IIR) algorithm that uses preference interval trees to perform binary search to sequentially insert items into an already ranked list. It achieves a provable optimal upper bound when the instance satisfies the SST condition. However, if only WST is assumed, the algorithm may fail to terminate in extreme cases.

In light of this, the `Probe-Rank` algorithm (Lou et al., 2022) solves this problem by sampling uniformly over the unranked pairs until the order of a pair is revealed. Afterwards, this order information is added as an edge of a directed acyclic graph. Then a transitive closure is computed to take advantage of stochastic transitivity when the order of unranked pairs can be inferred so that there can be some savings in sample complexity.

Dueling bandits. In the context of dueling bandits, the agent can only request a pair to be compared and get preferential feedback as opposed to stochastic bandits where the feedback is a single value tied to the pulled arm (Yue et al., 2012). As in the common case in bandits, not only must the rankings be inferred correctly, but also the optimization of regret (reward) must be taken into account. Contextual dueling bandits usually imply strong stochastic transitivity if the reward is defined on the actual utility gained from arm pulls (Dudík et al., 2015). Methods developed for these settings are ideal for online user interaction systems, where the collection of the reward is on-the-fly and cannot be postponed until the whole ranking is derived. Stochastic transitivity assumptions rarely hold in real-world situations, since the "rock-scissor-paper" relationship happens quite often in tournaments and preferences in real life. To cover these conditions, methods targeted for rewards defined on Copeland score $\frac{1}{N}\sum_{j\in[N]} \mathbb{1}(p_{i,j} > \frac{1}{2})$ or the Borda score $\frac{1}{N}\sum_{j\in[N]} p_{i,j}$ are widely studied in the dueling bandits literature (Wu and Liu, 2016; Zoghi et al., 2015; Saha et al., 2021).

We summarize our contributions and compare them with the related work in Table 1. Due to the lengthy form of the exact result of `Binary-Search` and `RMO-SST` that does not fit in one line. We use $\widetilde{O}$ in the table to omit insignificant terms. In addition, we keep relevant $\log$ terms inside the $\widetilde{O}$ to showcase the improvement obtained by the proposed method.

3. Preliminaries

In this paper, we consider actively ranking N items, with M oracles (also known as users or experts). We assume there exists an ordering '$\succ$' over these items which is characterized as a mapping $\sigma(\cdot) : [N] \to [N]$ indicating the position of a given item in the ranking in descending order. Equivalently, the inverse mapping $\sigma^{-1}(\cdot)$ lists the items in order: $\sigma^{-1}(1) \succ \sigma^{-1}(2) \succ \cdots \succ \sigma^{-1}(n)$. For two items i and

j, we assume that the result of the comparison is sampled independently from the Bernoulli distribution with mean $p^u_{i,j}$. More specifically, we denote $p^u_{i,j}$ as the probability that the response is "i is preferred over j" when a query is sent to oracle u. In particular, $p^u_{i,j} > \frac{1}{2}$ is considered as the item i is preferred over the item j by the oracle u. We will omit u in $p^u_{i,j}$ if the discussion is restricted to a single oracle.

For a fixed pair of items i and j, each oracle exhibits its preference, represented by the probability $p^u_{i,j}$. It is yet to decide how to aggregate them into a 'consensus preference'. If the consensus is defined as an average over $p^u_{i,j}$ or a majority vote over $\operatorname{sign}(p^u_{i,j} - 1/2)$, it is required that all oracles must be queried for each pair. In this case, there is no point to identify a more accurate expert to save queries. Instead, we make the following assumption that all oracles show a consistent preference for any item pair.

Assumption 3.1 (Consistency)**.** For any item pair (i, j), the preferences of all oracles are the same. More formally, for any two oracles u and v, we always have:

$$\operatorname{sign}(p^u_{i,j} - 1/2) = \operatorname{sign}(p^v_{i,j} - 1/2).$$

This assumption states that the oracles can show different levels of noise but must agree with the same underlying true ranking. This assumption also enables us to only select the more accurate oracles to recover the ranking with fewer comparisons. An equivalent assumption named the 'monotonicity' assumption is made by Saad et al. (2023).

We study the ranking problem under two different assumptions: Weak Stochastic Transitivity (WST) and Strong Stochastic Transitivity (SST) (Falahatgar et al., 2017):

Assumption 3.2 (Weak Stochastic Transitivity)**.** For a given oracle u, for any item pairs (i, j) and (j, k), if $p^u_{i,j} \geq \frac{1}{2}$ and $p^u_{j,k} \geq \frac{1}{2}$ then $p^u_{i,k} \geq \frac{1}{2}$.

WST implies that if $i \succ j$ and $j \succ k$, then $i \succ k$, which eliminates the possible cyclic order dependency and guarantees that an exact order of items can be inferred.

Assumption 3.3 (Strong Stochastic Transitivity)**.** For each oracle u and for any pair of items (i, j) and (j, k), if $p^u_{i,j} \geq \frac{1}{2}$ and $p^u_{j,k} \geq \frac{1}{2}$, then $p^u_{i,k} \geq \max\{p^u_{i,j}, p^u_{j,k}\} \geq \frac{1}{2}$.

SST implies that items that are more distant in the ranking are easier to compare. Combining the consistency assumption and either of the stochastic transitivity assumptions, we conclude that there must exist a ground-truth ranking σ on which all oracles agree. We use $\sigma(i)$ to denote the rank of item i and $\sigma^{-1}(k)$ to denote the index of the k-th best item.

Ranking two items The hardness of estimating the preference of two items i and j under oracle u can be captured by the *gap* between their preferential probability and $1/2$: $\Delta^u_{i,j} = |p^u_{i,j} - 1/2|$. Intuitively, the closer the preferential probability to $1/2$, the harder it is to estimate the preference of the two items since the collected responses are more noisy. In our multi-oracle setting, a trivial method would be querying one oracle at a time in a uniformly random fashion and aggregating them as if from a single oracle, which leads to an *average gap* of $\bar{\Delta}_{i,j} := \sum_{u=1}^{M} \Delta^u_{i,j}/M$. In other words, any single-oracle algorithm can be trivially applied to the multi-oracle setting as if one oracle has a gap of $\bar{\Delta}_{i,j}$.

Ranking under WST condition Given that the hardness of inferring the rank of a pair is inversely proportional to the (squared) gap, we introduce the *average hardness* for a given pair (i, j) as $\bar{H}_{i,j} := 1/(\bar{\Delta}_{i,j})^2$. Another definition that we will use frequently is $H_{i,j} := M/\sum_{u=1}^{M}(\Delta^u_{i,j})^2$, which is the *hardness* of a given pair (i, j). It is straightforward to prove that $H_{i,j} \leq \bar{H}_{i,j}$ by the Cauchy-Schwartz inequality, so any algorithm that can achieve a sample complexity related to $H_{i,j}$ instead of $\bar{H}_{i,j}$ is a strict improvement over the trivially adapted single-oracle algorithm.

Ranking under SST condition For ranking under the SST condition, the hardness depends on each item's most difficult pair. Here, for each item i, we define the minimum gap $\Delta_i := \min_{j \in [N], j \neq i} \Delta_{i,j}$ and maximum hardness $H_i := \max_{j \in [N], j \neq i} H_{i,j}$.

Finally, we define a δ-correct ranking algorithm as an algorithm that returns the true ranking with probability $1 - \delta$ within finite number of queries.

Additional notations Let $a \vee b := \max(a, b)$. We use standard asymptotic notations, including $O(\cdot), \Omega(\cdot), \Theta(\cdot)$, and $\widetilde{O}(\cdot), \widetilde{\Omega}(\cdot), \widetilde{\Theta}(\cdot)$ will hide logarithmic factors. For a positive integer N, $[N] := \{1, 2, \ldots, N\}$. For a set of items S, denote the set of all possible pairs of items as $S^2 := \{(i, j) : i, j \in S, i \neq j\}$.

4. Main algorithms

In this section, we propose algorithms called *Rank-with-Multiple-Oracles (RMO)* under both WST and SST conditions.

4.1. Proposed algorithm under WST condition

The ranking algorithm under the WST condition is called `RMO-WST`, which is displayed in Algorithm 1 and has a bi-level design. At the high level, it calls `Probe-Max` (Algorithm 2) to select the maximal item from the pool of candidates repeatedly. After $N - 1$ rounds of finding the maximal item, the algorithm finds the ranking of all the items. At the low level, the `Compare` algorithm (Algorithm 3) perform comparisons that are necessary to rank a pair of items i and j, which also accounts for the heterogeneous quality of oracles that provide preferential feedback. `Try-Compare` (Algorithm 4) is where the actual comparison takes place and the order of pairs of items is determined.

In detail, `RMO-WST` (Algorithm 1, a variant of `Probe-Sort` in Lou et al. (2022)) takes a set of items labeled by $1, 2, \cdots, N$ as input and outputs a δ-correct ranking of them.

The set S_t contains the items to be ranked, each of which corresponds to a node in the directed graph T. It is also called "partial order preserving graph" in prior work (Lou et al., 2022). The graph starts with empty, where each node represents an item (Line 3). A directed edge is created between the two items, originating from the winning item, once the pair's order is determined. The maximal items are nodes of the current graph T such that there is no incoming edge towards them, which means they have not yet lose to any other items in comparison.

The WST assumption can also be employed to introduce additional edges during this process by getting the transitive closure of the graph. Furthermore, $\tau_{i,j}$ records a factor that determines the number of comparisons required to confidently determine the direction of a pair (i, j). It is initialized at 1 and its value will increase by one each time to determine the number of comparisons required throughout the algorithm. Inside each loop, the maximal item is found by `Probe-Max` with a confidence level of $2\delta/N^2$ and then removed from the graph T. This process repeats and finds the top items in the set of unranked items S_t sequentially.

Next, in `Probe-Max`, let U be the set that contains all the possible maximal items (i.e., all maximal items). Each item in U is paired with items whose order between them is not yet revealed (Line 2-Line 6). After revealing the order of a new pair using the `Compare` method, the losing item is removed from the set of possible maximal items U, and the graph T is also updated to include this directional edge and any other possible edges according to transitivity by running a standard method to compute the transitive closure of the graph (Line 9). The sub-routine `Compare` (Algorithm 3, modified from Saad et al. (2023)), is designed to account for the multiple-oracles situation. The high-level idea is to enumerate different possible parameters: the subset size s_r and the gap width h_r. In Line 3-Line 10, the guessed subset size s_r and the gap width h_r will be sent to `Try-Compare` (Algorithm 4), until both reach the actual quantities. Then `Try-Compare` will return the correct comparison result with high probability (Line 8).

`Try-Compare` (Algorithm 4) is where the actual query and estimation for the pair direction takes place. In Line 2, a query size of m is determined by the subset size s given from the argument. Note that this value halves each time since s is doubled each time this subroutine is called on the same pair.

Then, a total number of $n_0 m$ comparisons is evenly assigned to the set of oracles that has not been eliminated yet (Line 6). Note that in our setting, the feedback is a binary indicator for a pair of items rather than a bounded scalar value for a single item as described in Saad et al. (2023). After comparison, a Bernoulli parameter estimate is calculated for each individual oracle as $\widehat{\mu}_{i,j}^{(u,\ell)}$ and a joint estimate as $\widehat{\mu}_{i,j}^{(\ell)}$ (Line 7). In Line 8-Line 9, the order of the pair is called when the confidence threshold is reached. If not, it continues to the elimination phase (Line 10), where oracles with accuracy lower than the medium of the group in the common statistical sense according to the estimate are removed from the active set S_ℓ^{ij} and S_ℓ^{ij}.

Algorithm 1 `RMO-WST` (N, δ): Rank-with-Multiple-Oracles

```
1: input: number of items to rank $N$, confidence level $\delta$
2: initialize: $S_1 = [N], ans = [0]^N$, a directed graph $T$
   with $N$ nodes and no edges, for $(i,j) \in S_1^2$ set $\tau_{i,j} = 1$
3: define: $\tau = \{\tau_{i,j}\}_{(i,j)\in[N]^2}$
4: for $t = 1$ to $N - 1$ do
5:     $i_{\max}, T, \tau \leftarrow$ Probe-Max$(S_t, 2\delta/N^2, T, \tau)$
6:     remove $i_{\max}$ from $T$
7:     $ans[t-1] = i_{\max}$
8:     $S_{t+1} = S_t \setminus \{i_{\max}\}$
9: end for
10: $ans[N-1] = S_N[0]$, return $ans$
```

Algorithm 2 `Probe-Max`(S, δ, T, τ)

```
1: input: set of unranked items $S$, confidence level $\delta$,
   partial order preserving graph $T$, exponential factors
   $\{\tau_{i,j}\}_{(i,j)\in[N]^2}$ as $\tau$.
2: Let $U$ be the set of possible maximal items in $T$.
3: while $|U| > 1$ do
4:     $P = \{(i,j) | (i \in U \vee j \in U), (i,j) \in S^2, (i,j) \notin T\}$
5:     for $(i,j)$ in $\arg\min_{(x,y)\in P} \tau_{x,y}$ do
6:         $ans =$ Compare$(i, j, \frac{6\delta}{\pi^2\tau_{i,j}^2}, \tau_{i,j}), \tau_{i,j} = \tau_{i,j} + 1$
7:         if $ans \neq$ unsure then
8:             $w, l = ans, T =$ TransClosure$(T \cup (w \succ l))$
9:             if $|U| > 1$ and $l \in U$ then $U = U \setminus \{l\}$
10:        end if
11:    end for
12: end while
13: return $U[0], T, \tau$
```

Algorithm 3 Compare(i, j, δ, τ)

```
1: input: pair $(i, j)$, confidence level $\delta$, precision factor $\tau$
2: initialize: $r_{\max} = 1$, $\epsilon_\tau = 2^{-\tau}$, $ans = \texttt{unsure}$.
3: while $ans = \texttt{unsure}$ and $\epsilon_\tau^2 < 4\log(2M)M2^{-r_{\max}}$ do
4:     for $r = 0, \cdots, r_{\max}$ do
5:         $s_r = \frac{2^r M}{2^{r_{\max}}}$, $h_r = 2^{-\frac{r}{2}}$
6:         $\delta_{r_{\max}} = \delta/(10(r_{\max})^3 \log(M))$
7:         $ans = \texttt{Try-Compare}\ (i, j, \delta_{r_{\max}}, s_r, h_r)$
8:         if $ans \neq \texttt{unsure}$, break.
9:     end for
10:    $r_{\max} = r_{\max} + 1$
11: end while
12: return $ans$
```

Algorithm 4 Try-Compare(i, j, δ, s, h)

```
1: input: pair to query $(i, j)$, confidence level $\delta$, subset size $s$, estimated gap width $h$.
2: $m = 2^{\lceil \log_2(26\log(1/\delta)M/s) \rceil}$, $n_0 = 64/h^2$.
3: Sample a set of $m$ oracles with replacement from all $M$ oracles as $S_1$.
4: Let $S_1^{ij} = S_1^{ji} = S_1$ and $L = \lceil \log_{4/3}(M/s) \rceil$.
5: for $\ell = 0, \cdots, L$ do
6:     Request $t_\ell = n_0 m/|S_\ell^{ij}|$ comparisons for pair $(i, j)$ from each oracle $u \in S_\ell^{ij} \cup S_\ell^{ji}$. Denote $c_u^{ij}$ as the number of times $i \succ j$.
7:     $\widehat{\mu}_{i,j}^{(u,\ell)} = \frac{c_{u,\ell}^{ij}}{t_\ell}$, $\widehat{\mu}_{j,i}^{(u,\ell)} = 1 - \widehat{\mu}_{i,j}^{(u,\ell)}$, $\widehat{\mu}_{i,j}^{(\ell)} = \frac{1}{|S_\ell^{ij}|}\sum_{u \in S_\ell^{ij}} \widehat{\mu}_{i,j}^{(u,\ell)}$.
8:     if $\widehat{\mu}_{i,j}^{(\ell)} - \frac{1}{2} \geq \sqrt{2\log(2/\delta)/n_0 m}$ then return $i \succ j$
9:     if $\widehat{\mu}_{i,j}^{(\ell)} - \frac{1}{2} < -\sqrt{2\log(2/\delta)/n_0 m}$ then return $i \prec j$
10:    $S_{\ell+1}^{ij} \leftarrow \{v \in S_\ell^{ij} | \widehat{\mu}_{i,j}^{(v,\ell)} \geq$ medium of $\widehat{\mu}_{i,j}^{(u,\ell)}, u \in S_\ell^{ij}\}$
11:    $S_{\ell+1}^{ji} \leftarrow \{v \in S_\ell^{ji} | \widehat{\mu}_{j,i}^{(v,\ell)} \geq$ medium of $\widehat{\mu}_{j,i}^{(u,\ell)}, u \in S_\ell^{ji}\}$
12: end for
13: return unsure.
```

4.2. Proposed algorithm under SST condition

We propose another algorithm that works under the SST condition called RMO-SST, which is derived from the Iterative-Insertion-Ranking (IIR) routine by Ren et al. (2019). It also has a bi-level design like the algorithm discussed in the previous subsection. The low-level component is Compare (Algorithm 3). The detailed algorithm description is deferred to Appendix D due to space limit.

5. Main theoretical results

In this section, we present the upper bound on the sample complexity of the proposed algorithms, as well as a new lower bound.

5.1. Upper bound of sample complexity under WST condition

To start with, we introduce the following theorem, which guarantees the performance of Compare (Algorithm 3).

Theorem 5.1. *(Restatement of Theorem 4.1 from Saad et al. (2023)) For any given τ, δ and any pair (i, j), with probability at least $1 - \delta$, Algorithm 3 satisfies:*

1. *It outputs the correct order or unsure for any given τ and $\delta > 0$.*
2. *If $\tau > -\frac{1}{2}\log(M/H_{i,j})$, the correct order is returned.*
3. *When the correct order is returned, the sample complexity is $\widetilde{O}(\log(1/\delta)H_{i,j})$.*

To obtain the above theorem, we replace the ϵ in the original theorem with $2^{-\tau}$ to suit our application. A detailed reasoning is available in Appendix B.1. This theorem guarantees the pairwise comparisons are correct with desired accuracy.

With a robust routine to return the correct order for each queried pair, a carefully designed ranking algorithm (Algorithm 1) orchestrates such pairwise comparisons to recover the ranking with high probability. In RMO-WST, the maximal item in the candidate set is identified and removed iteratively to rank all items. Thus, the total sample complexity is the summation of the sample complexity to identify each maximal item. And such cost can be upper bounded by N times the hardness to compare it with the item immediately smaller than it. We present the following theorem to characterize the total sample complexity upper bound of Algorithm 1. The detailed proof is deferred to Appendix B.2.

Theorem 5.2. *(Instance-dependent sample complexity upper bound for RMO-WST) For a given set of items $[N]$ and desired confidence level δ, Algorithm 1 terminates with sample complexity bounded by*

$$\widetilde{O}\Big(N\textstyle\sum_{i=2}^{N} H_{\sigma^{-1}(i),\sigma^{-1}(i-1)}\Big).$$

With probability at least $1 - \delta$, Algorithm 1 will output a ranking that exactly matches the true ranking.

Remark 5.3. A baseline algorithm is to uniformly randomly choose one oracle and apply the Probe-Rank algorithm with the average oracle. This leads to an averaged oracle with average gap $\bar{\Delta}_{i,j}$ or average hardness $\bar{H}_{i,j}$ as described near the end of Section 3. The resulting sample complexity is $\widetilde{O}\big(N\sum_{i=2}^{N} \bar{H}_{\sigma^{-1}(i),\sigma^{-1}(i-1)}\big)$, which is strictly worse than our sample complexity, since $H_{i,j} \leq \bar{H}_{i,j}$.

5.2. Upper bound of sample complexity under SST condition

We state the sample complexities resulting from running the `RMO-SST` as follows:

Theorem 5.4. *(Instance-dependent sample complexity upper bound for `RMO-SST`) For a given set of items $[N]$ and desired confidence level δ, Algorithm 7 in Appendix D terminates with sample complexity bounded by $\widetilde{O}\Big(\sum_{i\in[N]} H_i\big(\log\log(H_i)+\log(N/\delta)\big)\Big)$.*

A sketch of proof of the theorem is deferred to Appendix B.3.

Remark 5.5. Theorem 5.1 in Saad et al. (2023) presented an algorithm with upper bound of $\widetilde{O}\Big(\sum_{i\in[N]} H_i\big(\log\log(H_i)\ +\ \log^2(N)\ +\ \log(1/\delta)\big)\Big)$, which suffers an extra $\log N$ factor compared with ours. The two $\log N$ factors come from assigning a failure probability of δ/N to insert one item into the sorting tree and adding the number of queries over $\log N$ comparisons. This gives approximately $H_i\log(N\log(N)/\delta)\times\log N$ total comparisons. The last $\log N$ is removable by applying a more sophisticated insertion procedure we presented here. Also, Ren et al. (2019, Theorem 2) suggests that the lower bound is of order $\Omega\Big(\sum_{i\in[N]} H_i\big(\log\log(H_i)+\log(N/\delta)\big)\Big)$, which tightly matches our result.

5.3. Lower bound of sample complexity

Since the lower bound for the SST case is well-established, this section will focus on the lower bound for the WST case. This remains an open problem to the best of our knowledge.

To solve the single-oracle ranking problem, Lou et al. (2022) proposed a class of hard instances for any (ϵ,δ)-correct algorithm. The definition of (ϵ,δ)-correctness is that, with probability at least $1-\delta$, the algorithm will output a ranking $\widehat{\sigma}$ such that for all $i\succ_{\widehat{\sigma}} j$, $p(i,j)\ge\frac{1}{2}-\epsilon$. Intuitively, this means the algorithm will only mis-rank those pairs satisfying $|p(i,j)-1/2|<\epsilon$.

Lou et al. (2022) conjectured that any (ϵ,δ)-correct algorithm must query at least $\Omega(N^2\log(1/\delta)/\epsilon^2)$, but did not formally prove it. We will first present a proof that fills this gap, and then extend the results into the multi-oracle setting.

Single Oracle Similar to Lou et al. (2022), we construct a class of hard instances for the single-oracle setting. Each instance is indexed by a ranking σ.

Problem 1 ($\mathcal{I}_{\text{WST}}$). *Consider N items with an underlying ordering σ. For all $i\succ_\sigma j$,*

$$p^\sigma(i,j)=\begin{cases}\frac{1}{2}+\epsilon, & \textit{if } \sigma(i)=1 \textit{ and } \sigma(j)=2,\\ \frac{1}{2}, & \textit{otherwise},\end{cases}$$

and for $i\prec_\sigma j$, $p(i,j)=1-p(j,i)$.

Solving the instance class above can be reduced to solving the one-sided instance class described below in Problem 2. The reduction is done by query (i,j) and (j,i) on $\mathcal{I}_{\text{WST'}}$ equally likely to simulate the same environment as in $\mathcal{I}_{\text{WST}}$. Therefore, $\mathcal{I}_{\text{WST}}$ is at least as hard as $\mathcal{I}_{\text{WST'}}$, up to constants.

Problem 2 ($\mathcal{I}_{\text{WST'}}$). *Consider N items with an underlying ordering σ. For all i,j,*

$$p^\sigma(i,j)=\begin{cases}\frac{1}{2}+2\epsilon, & \textit{if } \sigma(i)=1 \textit{ and } \sigma(j)=2,\\ \frac{1}{2}, & \textit{otherwise}.\end{cases}$$

For any $(2\epsilon,\delta)$-correct ranking algorithm that outputs a 2ϵ-correct ranking under $\mathcal{I}_{\text{WST'}}$ with probability at least $1-\delta$, we have that the algorithm must correctly rank between the largest item $\sigma^{-1}(1)$ and the second-largest one $\sigma^{-1}(2)$. Intuitively, this implies that the algorithm has to go over almost all pairs to correctly identify $\sigma^{-1}(1)$ and $\sigma^{-1}(2)$, which is signified by a biased coin among N^2 fair coins. We have the following result:

Theorem 5.6. *For any (ϵ,δ)-correct algorithm $\mathcal{A}$, there exist a ranking σ and corresponding $p(i,j)$ such that with probability at least δ, $\sum_{i,j}C_{i,j}=\Omega\left(\frac{N^2\log(1/\delta)}{\epsilon^2}\right)$, where $C_{i,j}$ denotes the queries made at (i,j).*

The lower bound is tight up to logarithmic factors because a simple algorithm that allocates comparisons evenly to each pair will guarantee an ϵ-approximate estimation of $p(i,j)$, thus ensuring the ranking is (ϵ,δ)-correct.

Further, we define the (ϵ,δ)-correctness for multiple oracles:

Definition 5.7. An algorithm $\mathcal{A}$ is called (ϵ,δ)-correct, if with probability at least $1-\delta$, $\mathcal{A}$ will output a ranking $\widehat{\sigma}$ such that for all $i\succ_{\widehat{\sigma}} j$ but $j\succ_\sigma i$, $\sum_{u=1}^{M}(\Delta^u_{i,j})^2<\epsilon^2$.

Intuitively, the equivalent probability margin $\sqrt{\sum_{u=1}^{M}(\Delta^u_{i,j})^2}$ must be small for any mis-ranked pair (i,j).

We have the following result:

Theorem 5.8. *For any (ϵ,δ)-correct algorithm $\mathcal{A}$, there exist a ranking σ and corresponding $\{p^u(i,j)\}_{u\in[M]}$ such that with probability at least δ,*

$$\textstyle\sum_{i,j}C_{i,j}=\Omega\big(N^2M\log(1/\delta)/\epsilon^2\big)=\widetilde{\Omega}\big(N^2H_{\sigma^{-1}(1),\sigma^{-1}(2)}\big),$$

where $C_{i,j}$ denotes the queries made at (i,j).

Again, the lower bound is tight and can be reached by allocating the comparison budget evenly to each pair and call Algorithm 3.

6. Experiments

We study the practical performance of the proposed algorithm and compare it with existing methods. We compare

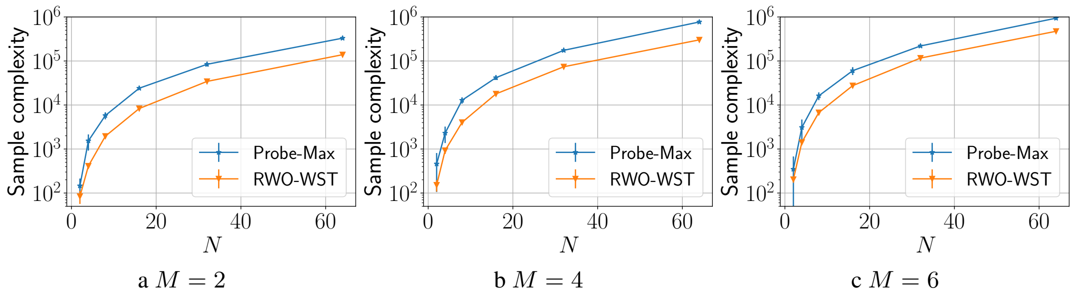

Figure 1: Sample complexities of ranking $N \in \{2, 4, 8, 16, 32, 64\}$ items with $M - 1$ oracles have low accuracy and one oracle has high accuracy.

two methods in the experiment [1]:

`Probe-Max`: the main algorithm proposed (Lou et al., 2022), however, their algorithm does not account for multiple oracles. In this case, as a naive implementation, whenever a pair is requested, it chooses an oracle from $[M]$ uniformly at random.

`RMO-WST`: Algorithm 1 proposed in this work. However, we notice that due to multiple uses of the union bound, excessive sampling can occur, which is unrealistic in real-world scenarios.

For instance, repetitive sampling of m tasks as seen in Algorithm 4 at Line 2 can enhance the precision of the overall estimate $\widehat{\mu}_{i,j}^{(u,\ell)}$. However, since s diminishes at an exponential rate, the quantities m and t_l also increase exponentially. Our hypothesis is that setting $S_1 = [M]$—in other words, maintaining a constant size of m at M—can lead to greater efficiency. Additionally, the distribution of the accuracy of active candidates is the same with or without the sampling without replacement in Line 6. Furthermore, note that the confidence interval used from Line 8 to Line 9 still holds after the change. We also notice that, in Line 7 the individual estimate and the global estimate depend only on the samples collected within a single iteration of the for loop starting Line 4. This can also be improved by reusing the statistics collected in previous rounds. We present the improved algorithm in Algorithm 5 in supplementary material.

To start with, we randomly generate the comparison matrix according to the following rules: a) There are N items to rank and a random permutation of $[N]$ is generated as the ground truth ranking. There are M oracles that can conduct pairwise evaluation each with a comparison matrix for $p_{i,j}$. b) One of the M oracles is very accurate and we assign a probability value for $p_{i,j}$ sampled from $[0.85, 0.95]$ uniformly at random if $i \succ j$ for every pair of items. c) For the rest of $M - 1$ oracles, we assign a value to $p_{i,j}$ sampled uniformly at random from $[0.55, 0.65]$ if $i \succ j$.

[1]Upon acceptance of the paper, the code will be posted on GitHub.

We tested $N \in \{2, 4, 8, 16, 32, 64\}$ to explore how the samples grow with the size of the problem. The mix of accurate responses can also affect the performance gain. The average sample complexity of 32 runs with one standard deviation error bar calculated by Python `numpy` library is plotted in Figure 1a, Figure 1b and Figure 1c for the case where $M = 2$, $M = 4$ and $M = 6$, respectively. In general, it is harder to derive estimated rankings while the majority of the information is noisy ($M = 6$), as in this case the sample complexity is much higher for both algorithms. However, regardless of the noisiness of oracles, the proposed method always beats the baseline. In addition, if the proportion of the noisy oracles are high, then our proposed method benefits more which is illustrated by the wider gap compared to the baseline method from $M = 2$ to $M = 6$.

7. Conclusion and future work

In this work, we introduce an active ranking algorithm that works in the WST setting when there are multiple experts with different degrees of precision in multiple tasks. An instance-independent upper bound is proven for this case. We also solved an open problem for the lower bound in this setting. In addition, we extend a state-of-the-art active ranking algorithm under SST conditions into the multi-oracle setting. Whether there is an instance-dependent lower bound remains an open question.

We also conjecture that for the single oracle WST algorithm, its search efficiency could be improved with a better worse case sample complexity by redesigning the graph construction method. After which this can be extended into the multi-oracle case. It is also of great interest to design an algorithm that works for both WST and SST conditions to achieve a best-of-both-worlds result.

Limitations The monotonic assumption might not hold in general. The empirical experiments are performed assuming only one such user is more accurate. If the set of experts have uniform accuracy level, our method may have additional overhead.

Impact Statement

This work can potentially increase the efficiency of data collection. With the constantly rising cost of data acquisition and model training, savings on data make sense not only financially but also environmentally, and thus this reduction can thereby raise its own societal value.

Acknowledgments

We thank the anonymous reviewers and area chair for their helpful comments. YW and QG are supported in part by the NSF grant CIF-1911168. TJ and FF are supported in part by the NSF grant CIF-1908544. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

AGARWAL, S., DUGAR, D. and SENGUPTA, S. (2010). Ranking chemical structures for drug discovery: A new machine learning approach. *Journal of chemical information and modeling* .

BAŁCHANOWSKI, M. and BORYCZKA, U. (2023). A comparative study of rank aggregation methods in recommendation systems. *Entropy* **25**.

DUDÍK, M., HOFMANN, K., SCHAPIRE, R. E., SLIVKINS, A. and ZOGHI, M. (2015). Contextual dueling bandits. In *Conference on Learning Theory*. PMLR.

EBTEKAR, A. and LIU, P. (2021). Elo-mmr: A rating system for massive multiplayer competitions. *Proceedings of the Web Conference 2021* .

FALAHATGAR, M., HAO, Y., ORLITSKY, A., PICHAPATI, V. and RAVINDRAKUMAR, V. (2017). Maxing and ranking with few assumptions. *Advances in Neural Information Processing Systems* **30**.

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*. PMLR.

FEIGE, U., RAGHAVAN, P., PELEG, D. and UPFAL, E. (1994). Computing with noisy information. *SIAM Journal on Computing* **23** 1001–1018.

HECKEL, R., SHAH, N. B., RAMCHANDRAN, K. and WAINWRIGHT, M. J. (2019). Active ranking from pairwise comparisons and when parametric assumptions do not help. *The Annals of Statistics* **47** 3099–3126.

JIN, T., XU, P., GU, Q. and FARNOUD, F. (2020). Rank aggregation via heterogeneous thurstone preference models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

JUN, K.-S., WILLETT, R., WRIGHT, S. and NOWAK, R. (2019). Bilinear bandits with low-rank structure. In *International Conference on Machine Learning*. PMLR.

LOU, H., JIN, T., WU, Y., XU, P., GU, Q. and FARNOUD, F. (2022). Active ranking without strong stochastic transitivity. *Advances in neural information processing systems* .

MAYSTRE, L. and GROSSGLAUSER, M. (2015). Fast and accurate inference of plackett–luce models. *Advances in neural information processing systems* **28**.

MINKA, T. P., CLEVEN, R. and ZAYKOV, Y. (2018). Trueskill 2: An improved bayesian skill rating system.

MOHAJER, S., SUH, C. and ELMAHDY, A. (2017). Active learning for top-k rank aggregation from noisy comparisons. In *International Conference on Machine Learning*. PMLR.

NEGAHBAN, S., OH, S. and SHAH, D. (2012). Iterative ranking from pair-wise comparisons. *Advances in neural information processing systems* **25**.

NIU, S., LAN, Y., GUO, J., CHENG, X., YU, L. and LONG, G. (2015). Listwise approach for rank aggregation in crowdsourcing. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*.

OLIVEIRA, S. E., DINIZ, V., LACERDA, A., MERSCHMANM, L. and PAPPA, G. L. (2020). Is rank aggregation effective in recommender systems? an experimental analysis. *ACM Transactions on Intelligent Systems and Technology (TIST)* **11** 1–26.

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. K. and SHROFF, N. (2019). On sample complexity upper and lower bounds for exact ranking from noisy comparisons. *Advances in Neural Information Processing Systems* **32**.

RU, X., YE, X., SAKURAI, T. and ZOU, Q. (2021). Application of learning to rank in bioinformatics tasks. *Briefings in bioinformatics* .

SAAD, E. M., VERZELEN, N. and CARPENTIER, A. (2023). Active ranking of experts based on their performances in many tasks .

SAHA, A. and GOPALAN, A. (2019). Active ranking with subset-wise preferences. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR.

SAHA, A., KOREN, T. and MANSOUR, Y. (2021). Adversarial dueling bandits. In *International Conference on Machine Learning*. PMLR.

TAKANOBU, R., ZHUANG, T., HUANG, M., FENG, J., TANG, H. and ZHENG, B. (2019). Aggregating e-commerce search results from heterogeneous sources via hierarchical reinforcement learning. In *The World Wide Web Conference*.

THONET, T., CINAR, Y. G., GAUSSIER, E., LI, M. and RENDERS, J.-M. (2022). Listwise learning to rank based on approximate rank indicators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36.

VOJNOVIC, M. and YUN, S. (2016). Parameter estimation for generalized thurstone choice models. In *International Conference on Machine Learning*. PMLR.

WANG, S., HUANG, S., LIU, T.-Y., MA, J., CHEN, Z. and VEIJALAINEN, J. (2016). Ranking-oriented collaborative filtering: A listwise approach. *ACM Transactions on Information Systems (TOIS)* **35** 1–28.

WU, H. and LIU, X. (2016). Double thompson sampling for dueling bandits. *Advances in neural information processing systems* **29**.

WU, Y., JIN, T., LOU, H., XU, P., FARNOUD, F. and GU, Q. (2022). Adaptive sampling for heterogeneous rank aggregation from noisy pairwise comparisons. In *International Conference on Artificial Intelligence and Statistics*. PMLR.

YUE, Y., BRODER, J., KLEINBERG, R. and JOACHIMS, T. (2012). The k-armed dueling bandits problem. *Journal of Computer and System Sciences* **78** 1538–1556.

ZHU, W., JIANG, Y., LIU, J. S. and DENG, K. (2023). Partition–mallows model and its inference for rank aggregation. *Journal of the American Statistical Association* **118** 343–359.

ZOGHI, M., KARNIN, Z. S., WHITESON, S. and DE RIJKE, M. (2015). Copeland dueling bandits. *Advances in neural information processing systems* **28**.

A. Proof of technical lemmas

We first show that `Compare` (Algorithm 3) returns the correct outcomes between the given two items with high probability. In the following, without loss of generality, we assume the correct ordering between item i and item j is $i \succ j$. The proof follows those done similarly by (Saad et al., 2023), except that we deal with binary feedback representing the preference instead of numerical feedback for i and j respectively.

We first present the following lemma that characterizes the behavior of `Try-Compare` (Algorithm 4).

Lemma A.1. *In Algorithm 4, for each iteration ℓ, for any fixed pair of items $i \succ j$, we have that probability of getting incorrect probability estimation bounded as:*

$$\mathbb{P}\left(\widehat{\mu}_{i,j}^{(\ell)} - \frac{1}{2} < -\sqrt{\frac{2\log(1/\delta)}{|S_\ell| n_0}}\right) \le \delta,$$

where S_ℓ is the subset of active oracles and n_0 is the number of repeated comparisons.

Proof. According to the assumption that $i \succ j$, so $\mathbb{E}[\widehat{\mu}_{i,j}^{(\ell)}] = \frac{1}{|S_\ell|}\sum_{u \in S_\ell} p_{i,j}^u \ge \frac{1}{2}$. Then we have

$$\mathbb{P}\left(\widehat{\mu}_{i,j}^{(\ell)} - \frac{1}{2} < -\sqrt{\frac{2\log(1/\delta)}{|S_\ell| n_0}}\right) \le \mathbb{P}\left(\widehat{\mu}_{i,j}^{(\ell)} - \mathbb{E}[\widehat{\mu}_{i,j}^{(\ell)}] < -\sqrt{\frac{2\log(1/\delta)}{|S_\ell| n_0}}\right) \le \delta.$$

The last inequality is due to Chernoff's inequality given that $\widehat{\mu}_{i,j}^{(\ell)}$ is a summation of $|S_\ell| n_0$ bounded independent random variables in $[0, 1]$. □

Then we have the following lemma stating the correctness of `Try-Compare` (Algorithm 4).

Lemma A.2. *In Algorithm 4, the probability of returning an incorrect result or order is bounded by:*

$$\mathbb{P}(\text{return } i \prec j) \le 1.75\log(M)\delta.$$

Proof. If $i \prec j$ is returned, then in Algorithm 4, there exists some ℓ such that the condition in Line 9 is true. Formally, we have

$$\begin{aligned}\mathbb{P}(\text{return } i \prec j) &\le \mathbb{P}\Big(\exists \ell \in [\log_{4/3}(M/m)] : \widehat{\mu}_{i,j}^{(\ell)} - 1/2 < -\sqrt{2\log(2/\delta)/n_0 m}\Big)\\ &\le \log_{4/3}(M/m)\delta/2\\ &\le 1.75\log(M)\delta.\end{aligned}$$

The first inequality holds due to the reasoning above; the second inequality comes from the union bound and Lemma A.1; in the last inequality we drop m and rearrange terms. □

Lemma A.3. *Assume $i \succ j$. In Algorithm 3, the wrong result will appear with probability* $\mathbb{P}(\text{return } i \prec j) \le 0.6\delta$.

Proof. Let $ans_{r_{\max},r}$ denote the output of Algorithm 4 when it is called with arguments $(\delta_{r_{\max}}, s_r, h_r)$. We have

$$\begin{aligned}\mathbb{P}(\text{return } i \prec j) &\le \mathbb{P}\Big(\exists r_{\max} \ge 1, \exists r \le r_{\max} : ans_{r_{\max},r} = (i \prec j)\Big)\\ &\le \sum_{r_{\max}=1}^{\infty}\sum_{r=0}^{r_{\max}} \mathbb{P}(ans_{r_{\max},r} = (i \prec j))\\ &\overset{(1)}{\le} \sum_{r_{\max}=1}^{\infty}\sum_{r=0}^{r_{\max}} \frac{1.75\log(M)\delta}{10(r_{\max})^3\log(M)}\\ &= \sum_{r_{\max}=1}^{\infty} \frac{1.75\delta}{10}\frac{r_{\max}+1}{{r_{\max}}^3} \overset{(2)}{\le} 0.6\delta,\end{aligned}$$

where (1) is due to Lemma A.2 and the definition $\delta_{r_{\max}} = \delta/(10(r_{\max})^3\log(M))$, and (2) holds because $\sum_{r_{\max}=1}^{\infty}\frac{1+r_{\max}}{{r_{\max}}^3} \le 3$. □

B. Proofs of theorems

B.1. Full statement of Theorem 5.1

Our restatement also follows the structure of three conclusion claims towards the end of the proof by Saad et al. (2023, Theorem 4.1, Section B).

Note that one difference between our version of `Compare` (Algorithm 3) and their original algorithm is that their ϵ is replaced with τ by us to control the desired accuracy gap of one pairwise comparison. More specifically, in Line 2 of Algorithm 3, we calculated an equivalent ϵ as $\epsilon_\tau = 2^{-\tau}$ based on the input argument τ.

In their notation, d is the number of experts (oracles), which is equivalent to M in our notation.

The problem hardness factor $H_{i,j}$ is similarly defined. The setting is slightly different in that Saad et al. (2023) considered the difference of two 1-sub-Gaussian variables while we consider a Bernoulli variable shifted by $1/2$. The central problem is to identify the sign of the expectation of the said random variables (which determines the order between i and j), and thus the problem hardness factors are defined in the same spirit.

Now, we are ready to restate the theorem.

Claim 1: The first part of the theorem can be directly derived from Lemma A.3 or Saad et al. (2023, Lemma B.3). Indeed, since Lemma A.3 states that Algorithm 3 will return the wrong result with probability at most 0.6δ. Therefore, Algorithm 3 returns either `unsure` or the correct order with probability at least $1-\delta$.

Claim 2: The second part of the theorem states that when τ is sufficiently large, the algorithm will return the correct result with a high probability of at least $1-\delta$.

To see this, when $\epsilon_\tau^2 = 2^{-2\tau} < M/H_{i,j}$, that is, when $\tau > -\frac{1}{2}\log(M/H_{i,j})$, the algorithm returns `unsure` with probability less than 0.4δ by the same argument as in Saad et al. (2023, Eq.5 and Lemma B.8). And by Lemma A.3, it returns the wrong order with probability less than 0.6δ. In total, the probability of returning the incorrect result is less than δ.

Claim 3: The third part deals with the sample complexity. It is calculated under two conditions regarding the relationship between τ (hence ϵ_τ) and $M/H_{i,j}$, where c_1 is a constant. According to the end of the proof of Saad et al. (2023, Theorem 4.1, Section B) and replace it with our notation we have the following two cases:

1. When $\epsilon_\tau^2 \geq M/H_{i,j}$, Algorithm 3 finishes with a sample complexity of

$$c_1 \log^2(2M)\log\left(\frac{\log(2M)}{2^{-2\tau}}\right)\log\left(\frac{2\log(2M/2^{-2\tau})}{\delta}\right)\frac{M}{2^{-2\tau}} = \widetilde{O}\left(\log(1/\delta)\frac{M}{2^{-2\tau}}\right). \tag{B.1}$$

2. When $\epsilon_\tau^2 < M/H_{i,j}$, the total sample complexity is:

$$c_1 \log^2(M)\log(H_{i,j})\log\left(\log(H_{i,j})\log(M)/\delta\right)H_{i,j} = \widetilde{O}(\log(1/\delta)H_{i,j}). \tag{B.2}$$

B.2. Proof of Theorem 5.2

Proof. As stated in Theorem 5.1, for any pair (i,j) and τ, with probability $1-\delta$, the following event will hold for $\texttt{Compare}(i,j,\delta,\tau)$:

1. $\texttt{Compare}(i,j,\delta,\tau)$ outputs the correct order or `unsure`.
2. If $\tau > -\frac{1}{2}\log(M/H_{i,j})$, $\texttt{Compare}(i,j,\delta,\tau)$ outputs the correct order.
3. The sample complexity is $\widetilde{O}(H_{i,j})$.

For each i,j and τ, denote $\mathcal{E}_{i,j}(\tau)$ as the high-probability event described above regarding $\texttt{Compare}(i,j,6\delta/\pi^2\tau_{i,j}^2,\tau_{i,j})$ in Algorithm 4, which is called within $\texttt{Probe-Max}(S_t, 2\delta/N^2, T, \tau)$ in Algorithm 1. Then, we have that

$$\mathbb{P}(\mathcal{E}_{i,j}(\tau)) \geq 1 - \frac{12\delta}{\pi^2 N^2 \tau_{i,j}^2}.$$

By union bound, the probability that there exists one pair (i,j) that is compared wrongly by $\texttt{Compare}(i,j,\delta/\tau_{i,j}^2,\tau_{i,j})$ for some $\tau_{i,j}$ is

$$\mathbb{P}\left(\bigcup_{(i,j)\in[N]^2}\bigcup_{\tau=1}^{\infty}\overline{\mathcal{E}_{i,j}(\tau)}\right) \le \frac{N^2}{2}\sum_{\tau=1}^{\infty}\frac{12\delta}{\pi^2N^2\tau^2} \le \delta, \tag{B.3}$$

where the last inequality comes from $\sum_{\tau=1}^{\infty}\tau^{-2}=\pi^2/6$.

In the following proof, we assume that $\texttt{Compare}(i,j,6\delta/\pi^2\tau_{i,j}^2,\tau_{i,j})$ always runs successfully. Now, using Equation (B.1), Equation (B.2), define the following two terms:

1. When $\epsilon_\tau^2 \ge M/H_{i,j}$, for any $t>0$,

$$n^{(t)} := c_1\log^2(2M)\log\left(\frac{\log(2M)}{4^{-t}}\right)\log\left(\frac{2\pi^2N^2t^2\log(M/4^{-t})}{12\delta}\right)\frac{M}{4^{-t}} = \widetilde{O}(4^tM), \tag{B.4}$$

2. When $\epsilon_\tau^2 < M/H_{i,j}$, that is $2^{-\tau_{i,j}} < M/H_{i,j}$, which is applied in the first inequality below:

$$n_{i,j}^{(*)} := c_1\log^2(M)\log(H_{i,j})\log\left(\frac{\tau_{i,j}^2\log(H_{i,j})\log(M)N^2\pi^2}{12\delta}\right)H_{i,j} \le \widetilde{O}(H_{i,j}). \tag{B.5}$$

Given the fact that $\texttt{RMO-WST}$ only compares the pair contains at least one maximal element. In this case, for every call of $\texttt{Compare}$ on pair (i,j) if i is maximal, we say that item i *initializes* the comparison, and the number of comparisons is charged to i. If both i,j are maximal, then the cost is charged to both items. We denote the total number of charged comparisons to i as $c(i), i\in[N]$. And the sample complexity of $\texttt{RMO-WST}$ is at most $\sum_{i\in[N]}c(i)$.

Without loss of generality, assume the true ranking of items is $1\succ 2\succ\cdots\succ N$. Given $i\in[N]$, we use τ_i° to denote the value of $\tau_{i,i-1}$ when the order between i and $i-1$ is revealed. Let $\tau_1^\circ=0$. The order of adjacent items of i under WST condition can only be revealed when $\texttt{Compare}(i,i-1,2\delta/N^2,\tau_i^\circ)$ returns a value other than $\texttt{unsure}$. According to Lemma A.3, $\tau_i^\circ \le \lceil\frac{1}{2}\log\frac{H_{i,i-1}}{M}\rceil$.

Define $b_{i,j}^{(\tau)}$ as follows, where $n^{(\tau)}$ is defined in Equation (B.4):

$$b_{i,j}^{(\tau)} = \begin{cases} n^{(\tau)}, & \text{if } \tau<\tau_{i,j}^\circ \\ \sum_{t=1}^{\tau-1}n^{(t)}+n_{i,j}^{(*)}, & \text{otherwise}\end{cases}$$

For each $j\neq i$, let $\tau_{i,j}^*$ be the value of $\tau_{i,j}$ when last time $\texttt{Compare}$ is initialized by i and called before $\texttt{Probe-Max}(S_i,2\delta/N^2)$. For any $\tau>\tau_{i,j}^*$, if $\texttt{Compare}(i,j,2\delta/N^2,\tau)$ is called in $\texttt{Probe-Max}(S_t,2\delta/N^2)$ for some $t<i$, then it must not be initialized by i. In light of this, let $\tau_{i,j}^t$ be the value of $\tau_{i,j}$ after completion of $\texttt{Probe-Max}(S_t,2\delta/N^2)$. We break down $c(i)$ into two parts as follows:

$$c(i) \le \sum_{j\neq i}\sum_{\tau=1}^{\tau_{i,j}^*}b_{i,j}^{(\tau)} + \sum_{j\neq i}\sum_{\tau=\tau_{i,j}^{i-1}+1}^{\tau_{i,j}^i}b_{i,j}^{(\tau)} \tag{B.6}$$

We now move on to bound the first summation term on the right-hand side of Equation (B.6). Before $\texttt{Probe-Max}(S_{i-1},2\delta/N^2)$ terminates, item $i-1$ is in T. Therefore, whenever i is a maximal item, the order between i and $i-1$ is not revealed. So when i initializes the comparison $\texttt{Compare}(i,j,2\delta/N^2,\tau_{i,j}^*)$, the item pair $(i,i-1)$ is also in the set of legitimate pairs P. Therefore, $\tau_{i,j}^*$ is no larger than the value of $\tau_{i,i-1}$ at that point, and is further no larget than τ_i°:

$$\sum_{j\neq i}\sum_{\tau=1}^{\tau_{i,j}^*}b_{i,j}^{(\tau)} \le N\sum_{\tau=1}^{\tau_{i,i-1}}b_\tau \le N\sum_{\tau=1}^{\tau_i^\circ}b_{i,i-1}^{(\tau)}. \tag{B.7}$$

We then continue to bound the second summation term in $c(i)$ in Equation (B.6). Consider the last group of `Compare` called in `Probe-Max`$(S_i, 2\delta/N^2)$, here the groups mean that there might be multiple item pairs whose values τ are the minimum in P. Denote their τ values by τ^i. There must be some `Compare`$(a_i, b_i, 2\delta/N^2, \tau^i)$ returning $b_i \succ a_i$ such that a_i is a maximal item, otherwise no maximal item is removed from U and `Probe-Max` will not terminate. When every `Compare` call is returning the correct order, a_i is not the maximal in S_i so $a_i > i$. Thus, item $a_i - 1$ is also in S_i and before the call of `Compare`$(a_i, b_i, 2\delta/N^2, \tau^i)$, the order between a_i and $a_i - 1$ is not revealed, that is, $\tau^i \le \tau^\circ_{a_i}$. Moreover, $\tau^i_{i,j} \le \tau^i$ because we always compare pairs of items with the smallest τ values, it follows that

$$\sum_{j \ne i} \sum_{\tau=\tau^{i-1}_{i,j}+1}^{\tau^i_{i,j}} b^{(\tau)}_{i,j} \le N \sum_{\tau=1}^{\tau^\circ_i} b^{(\tau)}_{i,i-1}. \tag{B.8}$$

In summary, the total sample complexity is

$$\sum_{i=1}^{N} c(i) \le 2N \sum_{i=1}^{N} \sum_{\tau=1}^{\tau^\circ_i} b^{(\tau)}_{i,i-1} = 2N \sum_{i=2}^{N} \sum_{\tau=1}^{\tau^\circ_i} b^{(\tau)}_{i,i-1}, \tag{B.9}$$

where the last equality is due to $\tau^\circ_1 = 0$. Plug in $\tau^\circ_i \le \lceil \frac{1}{2} \log \frac{H_{i,i-1}}{M} \rceil$ into the above equation to get:

$$\begin{aligned} 2N \sum_{i=2}^{N} \sum_{\tau=1}^{\tau^\circ_i} b^{(\tau)}_{i,j} &= N \Bigg[\sum_{i=2}^{N} O\left(\log^2(M) \log(H_{i,i-1}) \log\left(\log(H_{i,i-1}) \log(M)\right) H_{i,i-1}\right) &\text{(B.10)} \\ &+ \sum_{i=2}^{N} \log^2(2M) \log\left(\log(2M) \frac{H_{i,i-1}}{M}\right) \log\left(4N^2 \log(\frac{H_{i,i-1}}{M} M)/\delta\right) O(\frac{H_{i,i-1}}{M} M) \Bigg] &\text{(B.11)} \\ &= \widetilde{O}\left(N \sum_{i=2}^{N} H_{i,i-1}\right) &\text{(B.12)} \end{aligned}$$

Given we assumed w.l.o.g. that the correct ranking is $1 \succ 2 \succ 3 \succ \cdots N$ and the sample complexity is Equation (B.12). Now we conclude without this assumption the sample complexity would be $\widetilde{O}\left(N \sum_{i=2}^{N} H_{\sigma^{-1}(i), \sigma^{-1}(i-1)}\right)$. $\square$

B.3. Sketch of Proof of Improved Sample Complexity Upper Bound with SST Assumption

The proof of the upper bound of the sample complexity is similar to Theorem 11 in (Ren et al., 2019). Notice that our algorithm listed in Appendix D differs from theirs by switching the all the usage of `ATC` to `Compare` (Algorithm 3). The rest of the procedure does not change. We establish our claim by stating the equivalence of this exchange in terms of algorithm guarantee.

First of all, we note that Lemma 8 in (Ren et al., 2019) `ATC` states that

1. With probability $1 - \delta$, `ATC` returns the correct order or `unsure`.
2. If $\epsilon < \Delta_{i,j}$, `ATC` returns the correct order.
3. The sample complexity is $O(\epsilon^{-2} \log(1/\delta))$.

In Algorithm 6, we notice that it differs with Algorithm 3 in that τ is no longer supplied through input arguments and $\epsilon = 2^{-\tau}$ is calculated outside of this subroutine. In this case, ϵ_τ in Algorithm 3 is equivalent to ϵ in Algorithm 6, which turns out to be a slight change in notation.

With this in mind, we can restate the Theorem 5.1 as follows:

1. With probability $1 - \delta$, `Compare` returns the correct order or `unsure`.
2. If $\epsilon < \sqrt{H_{i,j}/M}$, `Compare` returns the correct order.

3. The sample complexity is $\widetilde{O}(\log(1/\delta)H_{i,j})$.

Define $H_i := \max_{j\neq i, j\in[N]} H_{i,j}$. With this established claim, we plugin the condition $\epsilon < \sqrt{H_{i,j}/M}$ and sample complexity $\widetilde{O}(\log(1/\delta)H_{i,j})$ for a single pair to be correctly compared to the proof of Lemman 10 and Theorem 11 in (Ren et al., 2019) to get the sample complexity as:

$$\widetilde{O}\Big(\sum_{i\in[N]} H_i\Big(\log\log\big(H_i\big)+\log(N/\delta)\Big)\Big) \tag{B.13}$$

B.4. Proof of Theorem 5.6

Let $\mathcal{A}$ be an δ-correct algorithm. For any ranking σ, it correspond to a problem instance in $\mathcal{I}_{\text{WST}}$. We denote $\mathbb{P}^{\mathcal{A}}_{a,b}$ as the canonical bandit distribution of algorithm $\mathcal{A}$ under environment with $p(i,j)=\frac{1}{2}+\epsilon$ when $(i,j)=(a,b)$ and $p(i,j)=\frac{1}{2}$ otherwise. We also denote $\mathbb{P}^{\mathcal{A}}_0$ as the canonical bandit distribution of algorithm $\mathcal{A}$ under environment with $p(i,j)=\frac{1}{2}$ everywhere.

Since $\mathcal{A}$ is an δ-correct algorithm, its prediction on the ranking between a and b, denoted as $\widehat{\sigma}(a)$ and $\widehat{\sigma}(b)$, must align with the true ranking $\sigma(a)>\sigma(b)$ with probability at least $1-\delta$:

$$\mathbb{P}^{\mathcal{A}}_{a,b}\big(\widehat{\sigma}(a)<\widehat{\sigma}(b)\big)\leq\delta, \forall a,b\in[N], a\neq b.$$

Denote $X=\sum_{i,j}C_{i,j}$ the total number of queries made by $\mathcal{A}$ before it stops. Define the constant

$$\bar{x} := \inf\big\{x : \max_{a,b}\mathbb{P}^{\mathcal{A}}_{a,b}(X>x)\leq\delta\big\}.$$

Here, $\bar{x}$ serves as a probabilistic lower bound of the total number of queries for all instances. This is the quantity we aim to bound from below in the coming reasoning.

Lemma B.1. *For the fixed $\bar{x}$, we have that*

$$\mathbb{P}^{\mathcal{A}}_0(X>\bar{x})\geq 1-2\delta.$$

Proof of Lemma B.1. We define two new distributions $\widetilde{\mathbb{P}}^{\mathcal{A}}_{1,2}$ and $\widetilde{\mathbb{P}}^{\mathcal{A}}_{2,1}$, where $\widetilde{\mathbb{P}}^{\mathcal{A}}_{1,2}$ denotes the canonical bandit distribution of algorithm $\mathcal{A}$ under environment with $p(i,j)=\frac{1}{2}+\alpha$ when $(i,j)=(1,2)$ and $p(i,j)=\frac{1}{2}$ otherwise. $\widetilde{\mathbb{P}}^{\mathcal{A}}_{2,1}$ is defined similarly.

We have that

$$\mathbb{P}^{\mathcal{A}}_0(X\leq\bar{x})=\mathbb{P}^{\mathcal{A}}_0(\widehat{\sigma}(1)>\widehat{\sigma}(2), X\leq\bar{x})+\mathbb{P}^{\mathcal{A}}_0(\widehat{\sigma}(1)<\widehat{\sigma}(2), X\leq\bar{x}),$$

and for $\mathbb{P}^{\mathcal{A}}_0(\widehat{\sigma}(1)>\widehat{\sigma}(2), X\leq\bar{x})$, we have for any α,

$$\begin{aligned}\mathbb{P}^{\mathcal{A}}_0(\widehat{\sigma}(1)>\widehat{\sigma}(2), X\leq\bar{x}) &\leq \widetilde{\mathbb{P}}^{\mathcal{A}}_{2,1}(\widehat{\sigma}(1)>\widehat{\sigma}(2), X\leq\bar{x}) \\ &\quad+\sup_{\mathcal{F}\cap\{w:X\leq\bar{x}\}}|\mathbb{P}^{\mathcal{A}}_0(\mathcal{F}\cap\{w:X\leq\bar{x}\})-\widetilde{\mathbb{P}}^{\mathcal{A}}_{2,1}(\mathcal{F}\cap\{w:X\leq\bar{x}\})| \\ &\leq \delta+\underbrace{\sup_{\mathcal{F}\cap\{w:X\leq\bar{x}\}}|\mathbb{P}^{\mathcal{A}}_0(\mathcal{F}\cap\{w:X\leq\bar{x}\})-\widetilde{\mathbb{P}}^{\mathcal{A}}_{2,1}(\mathcal{F}\cap\{w:X\leq\bar{x}\})|}_{d_{\text{TV}}(\mathbb{P}^{\mathcal{A}}_0,\widetilde{\mathbb{P}}^{\mathcal{A}}_{2,1}|X\leq\bar{x})},\end{aligned}$$

where the first inequality comes from the definition of total variance distance; the second inequality comes from $\mathcal{A}$ being δ-correct so that $\mathbb{P}^{\mathcal{A}}_{2,1}(\widehat{\sigma}(1)>\widehat{\sigma}(2))\leq\delta$. Let α converge to 0, we have that the total variance distance will also converge to 0 when $X\leq\bar{x}$. Therefore, the above inequality implies that $\mathbb{P}^{\mathcal{A}}_0(\widehat{\sigma}(1)>\widehat{\sigma}(2), X\leq\bar{x})\leq\delta$.

Applying the same argument to $\mathbb{P}^{\mathcal{A}}_0(\widehat{\sigma}(1)<\widehat{\sigma}(2), X\leq\bar{x})$, we then conclude with $\mathbb{P}^{\mathcal{A}}_0(X>\bar{x})\geq 1-2\delta$. □

Consider a new algorithm $\mathcal{A}'$ that performs exactly the same as $\mathcal{A}$, until $\mathcal{A}$ stops or its total number of queries reaches $\bar{x}$. In the latter case, $\mathcal{A}'$ will stop and return 'null'. We have that $\mathcal{A}'$ is an algorithm such that:

$$\mathbb{P}_{a,b}^{\mathcal{A}'}\big(\widehat{\sigma} = \text{'null'}\big) \leq \delta, \forall a, b \in [N], a \neq b; \quad \mathbb{P}_0^{\mathcal{A}'}\big(\widehat{\sigma} \neq \text{'null'}\big) \leq 2\delta,$$

where 2δ comes from two cases of failure: 1. outputting a wrong ranking as $\mathcal{A}$ with probability at most δ; 2. outputting 'null' when the queries exceed limit $\bar{x}$ with probability at most δ.

By the Bretagnolle–Huber inequality, we have $\exp\big(-d_{\text{KL}}(\mathbb{P}_0^{\mathcal{A}'} \| \mathbb{P}_{a,b}^{\mathcal{A}'})\big) \leq 6\delta$. Further, denoting we have

$$\begin{aligned}
&\exp\Big(-\frac{1}{N(N-1)}\sum_{a,b}\sum_{i,j} C'_{i,j}\text{KL}\big(p_0(i,j)\|p_{a,b}(i,j)\big)\Big) \\
&\leq \frac{1}{N(N-1)}\sum_{a,b}\exp\Big(-\sum_{i,j} C'_{i,j}\text{KL}\big(p_0(i,j)\|p_{a,b}(i,j)\big)\Big) \\
&= \frac{1}{N(N-1)}\sum_{a,b}\exp\Big(-d_{\text{KL}}(\mathbb{P}_0^{\mathcal{A}'}\|\mathbb{P}_{a,b}^{\mathcal{A}'})\Big) \\
&\leq 6\delta,
\end{aligned}$$

where the first inequality comes from Jensen's inequality and $C'_{i,j}$ denotes the queries made by $\mathcal{A}'$ at (i,j); the first equation comes from the decomposition of KL-divergence for the canonical bandit model. $\text{KL}(p\|q)$ denotes the KL-divergence between two Bernoulli random variables with expectation p and q. Note that for $(i,j) \neq (a,b)$, $p_0(i,j) = p_{a,b}(i,j) = 1/2$.

Rearranging the terms and remove those terms with $p_0(i,j) = p_{a,b}(i,j)$ gives

$$\sum_{a,b} C'_{a,b} \geq \frac{N(N-1)\log(1/(6\delta))}{\text{KL}(1/2\|1/2+\epsilon)} = \Omega\Big(\frac{N^2\log(1/\delta)}{\epsilon^2}\Big).$$

Notice that, $C'_{i,j}$ denotes the queries made by $\mathcal{A}'$ at (i,j), which satisfies that $\sum_{a,b} C'_{a,b} \leq \bar{x}$, which as defined, serves as a high-probability lower bound on the sample complexity of $\mathcal{A}$.

B.5. Proof of Theorem 5.8

We define the following multi-oracle problem class:

Problem 3 ($\mathcal{I}_{\text{WST}''}$). *Consider N items with an underlying ordering 'σ'. For any items i, j and oracle u,*

$$p_u^\sigma(i,j) = \begin{cases} \frac{1}{2} + \frac{\epsilon}{\sqrt{M}}, & \textit{if } \sigma(i) = 1 \textit{ and } \sigma(j) = 2, \\ \frac{1}{2}, & \textit{otherwise.} \end{cases}$$

For any (ϵ, δ)-correct ranking algorithm that outputs a ϵ-correct ranking under $\mathcal{I}_{\text{WST}''}$ with probability at least $1-\delta$, we have that the algorithm must correctly rank between the largest item $\sigma^{-1}(1)$ and the second-largest one $\sigma^{-1}(2)$.

Because all oracles have the same comparison probability, the problem is equivalent to ranking with a single oracle, with the lower bound being $\Omega\Big(\frac{N^2 M\log(1/\delta)}{\epsilon^2}\Big)$.

C. Improved Algorithm in Empirical Study

In this section, we present the algorithm that is modified for practical usage. Instead of using an estimator $\widehat{\mu}_{i,j}$ that only depends on the data collected in the same iteration of the for loop. A global estimator is derived from the statistics collected from multiple iterations to save sample complexity (Line 6).

D. Improved Algorithm with SST Assumption

In this section, we display the improved algorithm that can lead to a log factor improvement of the `BinarySearch` algorithm that is proposed by Saad et al. (2023) for active ranking problems. We replace the `Attempt-to-Compare`

Algorithm 5 (Improved version for practical adoption) `Try-Compare`(i, j, δ, s, h)

```
1: input: pair to query $(i,j)$, confidence level $\delta$, subset size $s$, estimated gap width $h$.
2: $m = M$.
3: $S_1 = [M]$.
4: Let $S_1^{ij} = S_1^{ji} = S_1$ and $L = \lceil \log_{4/3}(M/s) \rceil$.
5: for $\ell = 0, \cdots, L$ do
6:     Request $t_\ell = n_0 m / |S_\ell^{ij}|$ comparisons for pair $(i,j)$ from each oracle $u \in S_\ell^{ij} \cup S_\ell^{ji}$. Denote $c_u^{ij}$ as the number of times $i \succ j$.
7:     $\widehat{\mu}_{i,j}^{(u,\ell)} = \frac{\sum_{\ell' \in [\ell]} c_{u,\ell'}^{ij}}{\sum_{\ell' \in [\ell]} t_{\ell'}}$, $\widehat{\mu}_{j,i}^{(u,\ell)} = 1 - \widehat{\mu}_{i,j}^{(u,\ell)}$, $\widehat{\mu}_{i,j}^{(\ell)} = \frac{1}{|S_\ell^{ij}|} \sum_{u \in S_\ell^{ij}} \widehat{\mu}_{i,j}^{(u,\ell)}$.
8:     if $\widehat{\mu}_{i,j}^{(\ell)} - \frac{1}{2} \geq \sqrt{2\log(2/\delta)/n_0 m}$ then
9:         return $i \succ j$
10:    end if
11:    if $\widehat{\mu}_{i,j}^{(\ell)} - \frac{1}{2} < -\sqrt{2\log(2/\delta)/n_0 m}$ then
12:        return $i \prec j$
13:    end if
14:    $S_{\ell+1}^{ij} \leftarrow \{v \in S_\ell^{ij} | \widehat{\mu}_{i,j}^{(v,\ell)} \geq$ medium of $\widehat{\mu}_{i,j}^{(u,\ell)}, u \in S_\ell^{ij}\}$
15:    $S_{\ell+1}^{ji} \leftarrow \{v \in S_\ell^{ji} | \widehat{\mu}_{j,i}^{(v,\ell)} \geq$ medium of $\widehat{\mu}_{j,i}^{(u,\ell)}, u \in S_\ell^{ji}\}$
16: end for
17: return unsure.
```

(ATC) subroutine in the `Iterative-Insertion-Ranking` (IIR) by Ren et al. (2019) with the `Compare` (Algorithm 3) proposed in this work to achieve this.

Algorithm 7 Main Procedure: `RMO-SST`

```
input: $N, M, \delta$
initialize: $\mathbf{C} = \mathbf{N} = \mathbf{0}$
1: $\mathcal{L}_1 \leftarrow$ a list containing only 1
2: for $i \leftarrow 2$ to $N$ do
3:     $\mathcal{L}_i \leftarrow$ IAI$(i, \mathcal{L}_{i-1}, \delta/(N-1))$        ▷ Algorithm 8
4: end for
Output: $\mathcal{L}_N$
```

Algorithm 8 Subroutine: `Iterative-Attempt-To-Insert` (IAI)

```
input: $i, \mathcal{L}, \delta$
initialize: $t = 0, Flag \leftarrow$ unsure
1: repeat
2:     $t = t + 1$, $\epsilon_t = 2^{-(t+1)}$, $\delta_t = \frac{6\delta}{\pi^2 t^2}$
3:     $Flag, \mathcal{L} \leftarrow$ATI$(i, \mathcal{L}, \epsilon_t, \delta_t)$        ▷ Algorithm 9
4: until $Flag =$ inserted
5: return $\mathcal{L}$
```

Algorithm 6 `ATC`(i, j, ϵ, δ) (modified version of `Compare` in Algorithm 3)

```
1: input: pair $(i, j)$, confidence level $\delta$, precision factor $\epsilon$
2: initialize: $r_{\max} = 1$, $ans = \texttt{unsure}$.
3: while $ans = \texttt{unsure}$ and $\epsilon^2 < 4\log(2M)M2^{-r_{\max}}$ do
4:     for $r = 0, \cdots, r_{\max}$ do
5:         $s_r = \frac{2^r M}{2^{r_{\max}}}$, $h_r = 2^{-\frac{r}{2}}$
6:         $\delta_{r_{\max}} = \delta/(10(r_{\max})^3 \log(M))$
7:         $ans \leftarrow \texttt{Try-Compare}\ (i, j, \delta_{r_{\max}}, s_r, h_r)$
8:         if $ans \neq \texttt{unsure}$, break.
9:     end for
10:    $r_{\max} \leftarrow r_{\max} + 1$
11: end while
12: if $ans = i \succ j$ then
13:    $ans \leftarrow i$
14: end if
15: if $ans = j \succ i$ then
16:    $ans \leftarrow j$
17: end if
18: return $ans$
```

Algorithm 9 Subroutine: `Attempt-To-Insert` (ATI)

```
input: $i, \mathcal{L}, \epsilon, \delta$
initialize: Let $\mathcal{T}$ be a PIT constructed from $\mathcal{L}$, $h \leftarrow \lceil 1 + \log_2(1 + |\mathcal{L}|) \rceil$, the depth of $\mathcal{T}$
For all leaf nodes $u$ of $\mathcal{T}$, initialize $c_u \leftarrow 0$; Set $t^{\max} \leftarrow \lceil \max\{4h, \frac{512}{25} \log \frac{2}{\delta}\} \rceil$ and $q \leftarrow \frac{15}{16}$
 1: $X \leftarrow$ the root node of $\mathcal{T}$
 2: for $t \leftarrow 1$ to $t^{\max}$ do
 3:   if $X$ is the root node then
 4:     $q \leftarrow$ ATC$(i, X$.mid$, \epsilon, 1 - q)$                    ▷ Algorithm 6
 5:     if $q = i$ then
 6:       $X \leftarrow X$.rchild
 7:     else
 8:       $X \leftarrow X$.lchild
 9:     end if
10:   else if $X$ is a leaf node then
11:     $q_1 \leftarrow$ ATC$(i, X$.left$, \epsilon, 1 - \sqrt{q})$
12:     $q_2 \leftarrow$ ATC$(i, X$.right$, \epsilon, 1 - \sqrt{q})$
13:     if $q_1 = i \wedge q_2 = X$.right then
14:       $c_X \leftarrow c_X + 1$
15:       if $c_X > b^t := \frac{1}{2}t + \sqrt{\frac{t}{2} \log \frac{\pi^2 t^2}{3\delta}} + 1$ then
16:         Insert $i$ into the corresponding interval of $X$ and
17:         return *inserted*
18:       end if
19:     else if $c_X > 0$ then
20:       $c_X \leftarrow c_X - 1$
21:     else
22:       $X \leftarrow X$.parent
23:     end if
24:   else
25:     $q_1 \leftarrow$ ATC$(i, X$.left$, \epsilon, 1 - \sqrt[3]{q})$
26:     $q_2 \leftarrow$ ATC$(i, X$.right$, \epsilon, 1 - \sqrt[3]{q})$
27:     $q_3 \leftarrow$ ATC$(i, X$.mid$, \epsilon, 1 - \sqrt[3]{q})$
28:     if $q_1 = X$.left $\vee q_2 = i$ then
29:       $X \leftarrow X$.parent
30:     else if $q_3 = i$ then
31:       $X \leftarrow X$.rchild
32:     else
33:       $X \leftarrow X$.lchild
34:     end if
35:   end if
36: end for
37: if there is a leaf node $u$ with $c_u \geq 1 + \frac{5}{16} t^{\max}$ then
38:   Insert $i$ into the corresponding interval of $u$
39:   return *inserted*
40: else
41:   return *unsure*
42: end if
```