Active Ranking without Strong Stochastic Transitivity

Anonymous Author(s)
Affiliation
Address
email

Abstract

Ranking from noisy comparisons is of great practical interest in machine learning. In this paper, we consider the problem of recovering the exact full ranking for a list of items under ranking models that do *not* assume the Strong Stochastic Transitivity property. We propose a δ-correct algorithm, Probe-Rank, that actively learns the ranking of the items from noisy pairwise comparisons. We prove a sample complexity upper bound for Probe-Rank, which only depends on the preference probabilities between items that are adjacent in the true ranking. This improves upon existing sample complexity results that depend on the preference probabilities for all pairs of items. Probe-Rank thus outperforms existing methods over a large collection of instances that do not satisfy Strong Stochastic Transitivity. Thorough numerical experiments in various settings are conducted, demonstrating that Probe-Rank is significantly more sample-efficient than the state-of-the-art active ranking method.

1 Introduction

Ranking from noisy comparisons has a wide range of applications including voting [7, 5], identifying the winner/full ranking of teams in sport leagues, ranking players in online gaming systems [17], crowdsourcing services [6], web search [8], and recommendation systems [2, 23]. In practice, comparisons usually contain certain levels of "noise". For example, duels in a game are not always won by the more proficient player, and preferences between movies/restaurants can also vary among different individuals. The presence of noise is commonly studied using a probabilistic comparison model [12, 25], where an item has a certain probability to win the comparison over another (pairwise) or a group of items (listwise).

We are interested in estimating the total ranking. To guarantee that the ranking is consistent with the preference probabilities, it is often assumed [14, 21, 12, 25] that if i ranks higher than j, then i wins a comparison between j with probability $p_{i,j} > \frac{1}{2}$. It is clear that the closer $p_{i,j}$ is to $1/2$, the more difficult it becomes to compare i and j. This assumption is referred to as *Weak Stochastic Transitivity (WST)*. A more strict assumption, *Strong Stochastic Transitivity (SST)*, is also often made [10, 24, 26]. SST requires items that have closer ranks to be more difficult to compare, i.e., if $i \succ j \succ k$, then $p_{i,k} \geq \max(p_{i,j}, p_{j,k}) > \frac{1}{2}$. Formal definitions of WST and SST are stated in Section 2.

However, SST can be too strong for scenarios where preference probabilities are not based on comparing a single quantifiable attribute. For instance, in sports, match outcomes are usually affected by team tactics. Team k may play a tactic that counters team i, resulting in a higher winning rate against team i compared with team j. Furthermore, items usually have multidimensional features and people may compare different pairs based on different features. A close pair in the overall ranking is thus not necessarily harder to compare than a pair that has a large gap. For example, when comparing cars, people might compare a given pair based on their interior design and another pair based on performance. As another example, in an experiment with games of chance with different probabilities

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

of winning and payoffs [30], it was observed that "people chose between adjacent gambles according to the payoff and between the more extreme gambles according to probability, or expected value."

Motivated by such applications, in this paper, we are interested in the problem of recovering the full ranking of n items under a more general setting, where only WST holds, while SST is not assumed to hold. We focus on only pairwise queries as they are easier to obtain and less prone to error in practice. Furthermore, as many applications [6, 22] allow interactions between users/annotators, we consider comparisons collected in an adaptive manner. Our goal is to use as few comparisons as possible and achieve a high confidence.

Existing algorithms [21, 25] cannot avoid comparing every item i with the item i^* that is the most similar to i, i.e., $\left|p_{i,i^*} - \frac{1}{2}\right| = \min_{j \neq i}\{\left|p_{i,j} - \frac{1}{2}\right|\}$. Further, [25] pointed out that comparing item pairs that are adjacent in the true ranking are necessary. When SST holds, adjacent pairs are also the most difficult pairs to distinguish, existing methods thus achieve sample-efficiency. For example, the Iterative-Insertion-Ranking (IIR) algorithm proposed in [25] maintains a preference tree and performs ranking by inserting items one after another. During the insertion process, every item is possible to be compared with every other items (and thus the most similar one), depending on the relative order of insertion and the true ranking. IIR was shown to be sample complexity optimal under SST and some other conditions.

However, when SST does not hold, comparing nonadjacent items harms the performance. Consider an extreme scenario where the true ranking is $1 \succ 2 \succ 3$ and $p_{1,2} = p_{2,3} = 0.8, p_{1,3} = \frac{1}{2} + 2^{-10}$. If item 1 is directly compared to item 3, then it takes $\Theta\left(2^{20}\right)$ comparisons[1]. For instance, in IIR, this can happen during the insertion process of item 3 when item 1 happens to be the root of the preference tree. On the other hand, a simple fix exists as we can let the three pairs to be compared simultaneously. The comparisons between items 1 and 2, items 2 and 3 will terminate much earlier and provide us with the information $1 \succ 2$, $2 \succ 3$, which is enough to recover the total ranking. Therefore, it is important to devise an algorithm whose sample complexity will not be harmed when SST fails to hold.

Contribution. In this paper, we propose an active algorithm, termed *Probe-Rank*, that ranks n items based on pairwise comparisons. Probe-Rank is a maxing-based algorithm, i.e., it ranks items by performing $n-1$ steps of maxing. We show that as long as the WST condition is satisfied, with probability at least $1-\delta$, Probe-Rank returns the correct ranking after conducting at most

$$O\left(n \sum_{i=1}^{n} \left(\widetilde{\Delta}_i^{-2}\right)\left(\log\log\left(\widetilde{\Delta}_i^{-1}\right) + \log\left(n/\delta\right)\right)\right) \tag{1}$$

comparisons, where $\widetilde{\Delta}_i = \min_{j:j \text{ and } i \text{ are adjacent}} \left|p_{i,j} - \frac{1}{2}\right|$. Probe-Rank is the first algorithm whose sample complexity only depends on comparison probabilities of adjacent items instead of all pairs of items [29, 21, 25, 31]. Theoretical analysis and numerical experiments under various settings are provided and show that Probe-Rank is more efficient than the state-of-the-art when comparing nonadjacent items is more difficult than comparing adjacent items. We also present a preliminary analysis on the sample complexity lower bound in the worst case scenario when SST does not hold. Further, we present a variant of Probe-Rank, named Probe-Rank-SE, in appendix B. Numerical experiments show that the variant is more sample-efficient under various settings.

2 Preliminaries

Notation Without loss of generality, let $[n] = \{1, 2, \ldots, n\}$ denote the set of n items. We write $p \sim Uni(a, b)$ to denote that p is sampled uniformly at random from the interval (a, b), and use Ber(p) to denote a Bernoulli random variable which equals 1 with probability p. We use (i, j) to denote the unordered item pair, i.e., $(i, j) = (j, i)$. Comparisons between items are probabilistic. Whenever two items i, j are compared, i (respectively, j) is preferred with probability $p_{i,j}$ (respectively, $p_{j,i}$), independent of any other quantities. For all $i \neq j$, $p_{i,j} + p_{j,i} = 1$. A probabilistic comparison model over $[n]$ is thus defined by the set of probabilities $\mathbf{P} = \{p_{i,j}\}_{1 \leq i < j \leq n}$.

[1] In fact, according to [13], we need $\Theta\left((p_{i,j} - 1/2)^{-2}\right)$ comparisons to be confident enough about the order between any two items i and j , $i, j \in [n]$.

In this paper, asymptotic notation including $O(\cdot), \Omega(\cdot), \Theta(\cdot)$ are defined in the standard sense, with $\widetilde{O}(\cdot), \widetilde{\Omega}(\cdot), \widetilde{\Theta}(\cdot)$ denoting corresponding weaker forms by allowing logarithmic factors.

Problem setup We assume that there exists a total ordering '$\succ$' over $[n]$ such that $\sigma_1 \succ \sigma_2 \succ \cdots \succ \sigma_n$ for some permutation σ of $[n]$. The permutation σ is referred to as the true ranking. Two items are called adjacent if they are adjacent in σ, i.e., one ranks right next to the other. To ensure that the true ranking σ is consistent with comparisons, we also assume that i has a higher rank than j if and only if $p_{i,j} > \frac{1}{2}$. In other words, if an item i is more preferred than j in σ, then i has a better chance to win the comparison with j. This assumption is known as *Weak Stochastic Transitivity (WST)*. A more strict assumption, *Strong Stochastic Transitivity (SST)*, is also frequently adopted. In addition to WST, SST assumes that whenever $i \succ j \succ k$, $p_{i,k} \geq \max(p_{i,j}, p_{j,k})$. In this paper, we assume only WST and our goal is to recover the true ranking σ with a given confidence level δ by taking pairwise comparisons and minimize the sample complexity. Problem instances are uniquely determined by the permutation σ representing the true ranking and the comparison probabilities $\mathbf{P}$.

Definition 1 (δ-correct algorithm)**.** *An algorithm is said to be δ-correct if for any input instance, with probability at least $1-\delta$, it returns a correct result in finite time.*

It is clear that the closer $p_{i,j}$ is to $\frac{1}{2}$, the more difficult it becomes to obtain the ordering between i and j. Therefore, the probability gap $\Delta_{i,j}$, defined as $\Delta_{i,j} = \left|p_{i,j} - \frac{1}{2}\right|$, provides a characterization of the ranking task difficulty and will be used as a parameter for measuring sample complexities of algorithms. For instance, [25, lemma 12] shows that for any δ-correct algorithm $\mathcal{A}$, $\limsup_{\Delta\to 0} \frac{T_{\mathcal{A}}[\Delta]}{\Delta^{-2}(\log\log\Delta^{-1}+\log\delta^{-1})} > 0$, where $T_{\mathcal{A}}[\Delta]$ is the expected number of samples taken by $\mathcal{A}$ on two items with probability gap Δ. Further, for each item i, we define $\Delta_i = \min_{j:j\neq i} \Delta_{i,j}$, the minimum probability gap between item i and any other item j, and

$$\widetilde{\Delta}_i = \min_{j:j \text{ and } i \text{ are adjacent in } \sigma} \Delta_{i,j}, \tag{2}$$

the minimum probability gap between i and its adjacent items in the true ranking. Note that $\Delta_i \leq \widetilde{\Delta}_i$ by definition and the equality holds when SST is satisfied.

3 Related work

The problem of ranking under coherent probabilistic comparisons dates back to 1994 [14]. Feige et al. [14] studied the comparison model assuming that $i \succ j \Leftrightarrow p_{i,j} = \frac{1}{2} + \Delta$ for some known Δ. It was shown that any δ-correct algorithm finds the true ranking with at least $\Theta(n\Delta^{-2}\log(n/\delta))$ comparisons in the worst case. Later in [21], a δ-correct algorithm TOP was proposed to rank the top-k elements by assuming only the existence of a total ranking (WST). The state-of-the-art IIR algorithm was proposed in [25], as discussed in Section 1. A comparison of related algorithms are presented in Table 1.

Table 1: δ-correct algorithms for exact ranking with sample complexity guarantee. Definitions of $\Delta_{i,j}, \Delta_i, \widetilde{\Delta}_i$ can be found in Section 2.

Algorithm	Assumptions on $\mathbf{P}$	Sample complexity
Single Elimination Tournament [21]	WST	$O\left(\frac{n(\log n)^2\log(1/\delta)}{\min_{1\leq i<j\leq n}\Delta_{i,j}^2}\right)$
PLPAC-AMPR [29]	The Plackett-Luce model	$O\left(n\log n\max_{i\in[n]}\{\frac{1}{\Delta_i^2}\log(\frac{n}{\delta\Delta_i})\}\right)$
Iterative-Insertion-Ranking [25]	WST	$O\left(\sum_{i=1}^{n}\frac{1}{\Delta_i^2}\left(\log\log\frac{1}{\Delta_i}+\log\frac{n}{\delta}\right)\right)$
Probe-Rank (this paper)	WST	$O\left(n\sum_{i=1}^{n}\frac{1}{(\widetilde{\Delta}_i)^2}\left(\log\log\frac{1}{\widetilde{\Delta}_i}+\log\frac{n}{\delta}\right)\right)$

Ranking or maxing have also been widely studied under more strict assumptions, e.g., SST, RST[2] and STI[3] and usually in the probably approximately correct (PAC) setting [10, 11, 12, 24, 26, 27, 29, 32].

[2]Under relaxed stochastic transitivity (RST), it is assumed that for all $i \succ j \succ k$, $\Delta_{i,k} \geq \gamma\max\{\Delta_{i,j}, \Delta_{j,k}\}$ for some $0 < \gamma < 1$.

[3]Under stochastic triangle inequality (STI), it is assumed that for all $i \succ j \succ k$, $\Delta_{i,k} \leq \Delta_{i,j} + \Delta_{j,k}$.

In particular, [24, 26, 27, 29] considered parametric comparison models such as the multinomial logit (MNL) model. Note that parametric models are often more restrictive and can imply SST/STI conditions. In the PAC setting, the goal is to find an ϵ-ranking $r_1 \succ r_2 \succ \cdots \succ r_n$ such that $p_{r_i,r_j} > \frac{1}{2} - \epsilon$ for all $i < j$. Although ϵ-rankings become closer to the true ranking as ϵ goes to 0, it is pointed out by [25] that PAC ranking algorithms cannot be easily extended to the case when $\epsilon = 0$. Among all, [12] is the most relevant work to this paper. In [12], PAC ranking and maxing were studied for both SST and WST settings. For WST, an instance-independent lower bound $\Theta(n^2)$ was proved, and a brute-force algorithm which compares each pair to an accuracy of ϵ and thus conducts $O((n^2/\epsilon^2)\log(n/\delta))$ comparisons was proposed. Note that in this paper, we are aiming at recovering the exact ranking instead of an ϵ-ranking. An exact ranking is preferred over an epsilon-ranking in competitive applications like voting and sport games, where people are not satisfied with an approximate winner. Furthermore, as suggested by [25], analyzing the exact ranking helps us to gain a better understanding about the instance-wise upper and lower bounds. A trivial extension of the brute-force algorithm can lead to sample complexity $\widetilde{O}\left(\frac{n^2}{\min_{i,j}\Delta_{i,j}^2}\right)$, which is substantially worse than our proposed algorithm.

Although we believe WST can be considered a natural and reasonably weak assumption, there are situations that WST does not hold as a ranking over items may not exist or, if it does, all comparison probabilities are not necessarily consistent with that ranking. So another line of research is to allow comparison probabilities $p_{i,j}$ take any values in $(0,1)$ as long as $p_{i,j} + p_{j,i} = 1$. In such scenarios, rankings can be defined and derived based on various criteria including Borda score [16, 19, 28] and Copeland score [4, 33]. The ranking problem has also been studied from a heterogeneous perspective [31, 18], where queries are made by multiple agents with different comparison probabilities. In [15], the problem of testing whether the WST condition holds was studied. More broadly, the problems of ranking, maxing or selection can be formulated in the context of dueling bandits. A comprehensive survey can be found in [3].

4 Proposed algorithm

In this section, we propose a δ-correct algorithm for exact ranking of all problem instances that satisfy the WST condition. As mentioned previously, our algorithm is designed to take into account situations where nonadjacent items can be more difficult to compare than adjacent items.

To avoid spending unnecessary samples on item pairs with small probability gaps, we propose a subroutine named *Successive-Comparison* (SC) (see Subroutine 1). SC uses a parameter τ for controlling to what extent the comparison should last. Specifically, SC compares the given item pair for a fixed number $b_\tau = \lceil (2/\epsilon_\tau^2)\log(1/\delta_\tau) \rceil$ times with an accuracy level $\epsilon_\tau = 2^{-\tau}$ and confidence level $\delta_\tau = 6\delta/(\tau^2\pi^2)$. If the empirical probability that i (respectively, j) wins is over $1/2$ by more than $\epsilon_\tau/2$, then SC returns i (respectively, j) as the more preferred item. Otherwise, SC will return 'unsure' to inform us that more samples are needed.

For two items i and j, SC (i, j, δ, τ) will be called successively with τ increasing by 1 at a time. We show in Appendix A that after τ gets large enough such that $\epsilon_\tau \le \Delta_{i,j}$, the correct ordering between i and j will be returned with high probability.

Subroutine 1 Successive-Comparison(i, j, δ, τ) (SC)

Input: items i, j, confidence level δ, probing parameter τ
$w_i = 0, \epsilon_\tau = 2^{-\tau}, \delta_\tau = \frac{\delta}{c\tau^2}, c = \frac{\pi^2}{6}, b_\tau = \left\lceil \frac{2}{\epsilon_\tau^2}\log\frac{1}{\delta_\tau} \right\rceil$;
For $t = 1$ to b_τ **do**
    compare i and j once; if i wins, $w_i = w_i + 1$;
$\widehat{p}_i = w_i/b_\tau$;
return $[i, j]$ **if** $\widehat{p}_i - \frac{1}{2} > \frac{1}{2}\epsilon_\tau$; **return** $[j, i]$ **if** $\widehat{p}_i - \frac{1}{2} < -\frac{1}{2}\epsilon_\tau$; and **return** 'unsure' **else**;

Partial order preserving graph During the ranking process, we maintain a directed graph T to store the partial orders we have obtained from SC instances so far. The graph T is initialized with n nodes $V_1, \ldots, V_n$ and no edge exists between any two nodes. Nodes $V_1, V_2, \ldots, V_n$ represent items $1, 2, \ldots, n$, respectively. In our algorithm, T is involved with three types of operations, *edge update*,

node removal and *maximal set selection*. Every time an instance of SC returns a pairwise order, e.g., $i \succ j$, we add a directed edge from V_i to V_j, written as $T = T \cup (i \succ j)$. Moreover, we also complete all edges in the transitive closure of the existing edges. In other words, if the edge between V_i and V_j induces a directed path from V_{k_1} to V_{k_2}, then a directed edge from V_{k_1} to V_{k_2} is also added to T. By completing the transitive closure, we can avoid comparing pairs whose ordering can be inferred from current knowledge and keep T acyclic. In the ranking process, we only run comparisons on item pairs that are not connected by edges and hence no contradictions in orderings will be returned by SC. By removing node V_i, we remove V_i and all edges of V_i from T. The maximal elements of T are the nodes which do not have any incoming edges. Since edges represent comparison results returned by SC, maximal elements correspond to items that have not lost to any other items. Note that since T is acyclic, maximal elements always exist.

Algorithm 2 Probe-Rank

1: **Input:** items $[n]$, confidence level δ
2: $S_1 = [n], Ans = [0]^n$, initialize T, $\tau_{i,j} = 1$ for all pairs of items $i \neq j$;
3: **For** $t = 1$ to $n-1$ **do**
4: $\quad i_{max} =$ Probe-Max$(S_t, 2\delta/n^2)$;
5: $\quad$ remove i_{max} from T; $Ans[t-1] = i_{max}$; $S_{t+1} = S_t \setminus \{i_{max}\}$;
6: $Ans[n-1] = S_n[0]$; **return** Ans;

Next, we establish our ranking algorithm *Probe-Rank* (see Algorithm 2). Probe-Rank finds the true ranking by performing maxing for $n-1$ rounds. In every round t, subroutine *Probe-Max* returns an item in S_t as the most preferred item (the maximum), where S_t denotes the set of remaining unranked items right before round t. The strategy of Probe-Max is to repeatedly apply SC on all item pairs. For every item pair (i, j), we initialize a global variable $\tau_{i,j}$ as the probing parameter for all SC instances that run over i, j. The graph T storing obtained partial orders is also viewed as a global variable. Parameters $\tau_{i,j}$ and graph T will be accessed and altered in Probe-Max.

In Probe-Max(S, δ) (see Subroutine 3), SC instances are performed only on items that are possible to be the actual maximum. Let U be the set of maximal elements in T. By definition, every item in U has not lost to any other item in S yet. Assuming all previous comparison results are correct, to find the actual maximum, it suffices to focus on items in U. We use S^2 to denote the set of all unordered item pairs in S, i.e., $S^2 = \{(a, b) : a, b \in S, a \neq b\}$. All 'legitimate' pairs that can potentially provide us with information about the maximum item in S are thus

$$P = \{(i, j) : (i \in U \text{ or } j \in U), (i, j) \in S^2, (i, j) \notin T\}, \tag{3}$$

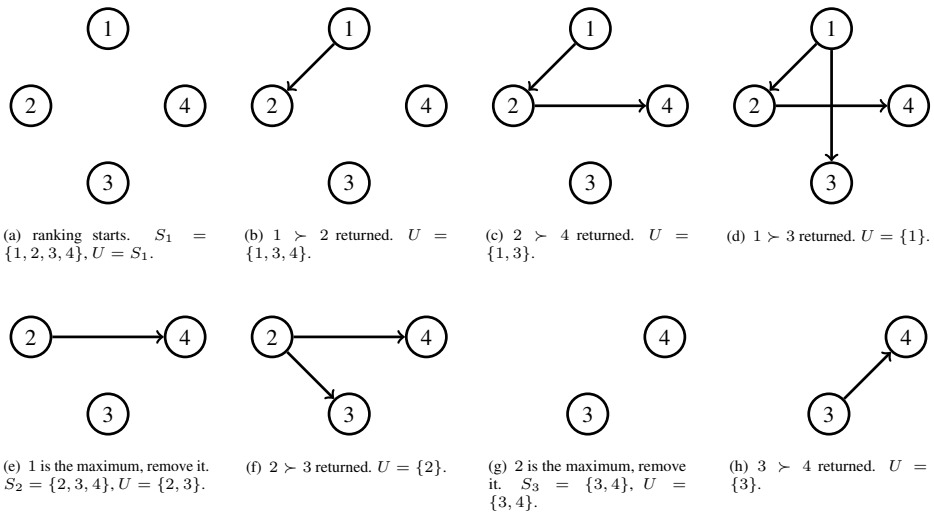
where $(i, j) \notin T$ means that nodes V_i and V_j are not connected in T. While U contains more than one items, Probe-Max keeps applying SC on item pairs in P. If an item in U loses a comparison, then we remove it from U. In every iteration of the while loop, the pair (i^*, j^*) in P with the smallest τ value is chosen (break ties arbitrarily) and SC $(i^*, j^*, \delta, \tau_{i^*,j^*})$ is performed. Note that the τ value increases by one after each call of SC. To start with item pairs with small τ values guarantees that we do not miss any useful information that can be obtained by paying only a small amount of comparisons.

Subroutine 3 Probe-Max(S, δ)

1: **Input:** set of unranked items S, SC confidence level δ
2: Let U be the set of maximal elements according to T
3: **While** $|U| > 1$ **do**
4: $\quad$ Let $P = \{(i, j) : (i \in U \text{ or } j \in U), (i, j) \in S^2, (i, j) \notin T\}$;
5: $\quad (a, b) = \text{argmin}_{(x,y) \in P} \tau_{x,y}$; $Ans =$ SC $(a, b, \delta, \tau_{a,b})$; $\tau_{a,b} = \tau_{a,b} + 1$;
6: $\quad$ **If** Ans is not 'unsure' **then**
7: $\quad\quad w, l = Ans$; $T = T \cup (w \succ l)$; **If** $|U| > 1$ and $l \in U$ **then** $U = U \setminus \{l\}$;
8: **return** $U[0]$;

We provide an example demonstrating the ranking process.

Example 1. Consider items $\{1, 2, 3, 4\}$ with true ranking $1 \succ 2 \succ 3 \succ 4$. Figure 1 shows the status of T, U, S_t throughout the ranking process. In particular, we assume the pairwise comparison results are all correct and returned in order $1 \succ 2$, $2 \succ 4$, $1 \succ 3$, $2 \succ 3$, $3 \succ 4$.



(a) ranking starts. $S_1 = \{1, 2, 3, 4\}, U = S_1$.

(b) $1 \succ 2$ returned. $U = \{1, 3, 4\}$.

(c) $2 \succ 4$ returned. $U = \{1, 3\}$.

(d) $1 \succ 3$ returned. $U = \{1\}$.

(e) 1 is the maximum, remove it. $S_2 = \{2, 3, 4\}, U = \{2, 3\}$.

(f) $2 \succ 3$ returned. $U = \{2\}$.

(g) 2 is the maximum, remove it. $S_3 = \{3, 4\}$, $U = \{3, 4\}$.

(h) $3 \succ 4$ returned. $U = \{3\}$.

Figure 1: An illustration of the steps by Probe-Ranking, assuming true ranking as $1 \succ 2 \succ 3 \succ 4$.

5 Upper bound on the sample complexity of Probe-Rank

In this section, we provide a sample complexity upper bound for the proposed algorithm Probe-Rank.

Theorem 2. *Let $\delta > 0$ be an arbitrary constant. For all problem instances satisfying the Weak Stochastic Transitivity (WST) property, with probability at least $1 - \delta$, Probe-Rank returns the true ranking of n items and conducts at most*

$$O\left(n \sum_{i=1}^{n} \left(\widetilde{\Delta}_i^{-2}\right) \left(\log\log\left(\widetilde{\Delta}_i^{-1}\right) + \log\left(n/\delta\right)\right)\right) \tag{4}$$

comparisons, where $\widetilde{\Delta}_i$ is defined as in (2)*.*

The proof of Theorem 2 is deferred to Appendix A.

By the preceding theorem, the sample complexity of Probe-Rank is upper bounded by the sum of terms $(\widetilde{\Delta}_i)^{-2}(\log\log(\widetilde{\Delta}_i)^{-1} + \log(n/\delta))$ with an additional multiplicative factor of n. Recall from Section 2 that the term $(\widetilde{\Delta}_i)^{-2}(\log\log(\widetilde{\Delta}_i)^{-1} + \log(n/\delta))$ can be viewed as a lower bound on the number of comparisons that is needed for obtaining the order between i and its adjacent items with confidence level δ/n. Theorem 2 thus suggests that in Probe-Rank, every item is compared until it can be distinguished from its neighbors and no further. This matches with our intuition that only comparisons between adjacent items are necessary, and a single nonadjacent pair being extremely hard to distinguish should not harm the overall sample complexity. In contrast, sample complexities of existing algorithms are determined by the smallest probability gap between items, which can lead to a substantially large amount of unnecessary comparisons.

However, Probe-Rank achieves the dependence on $\widetilde{\Delta}_i$ instead of Δ_i at the cost of an additional multiplicative factor of n. Intuitively, because we have zero prior information about which items are adjacent and which are not, Probe-Rank pays $\Theta\left(n\right)$ attempts for each item i in order to 'identify' its neighbors and get the ordering feedback.

We compare Probe-Rank with the state-of-the-art IIR algorithm. Let $\mathcal{C}\left(\text{Probe}\right)$ and $\mathcal{C}\left(\text{IIR}\right)$ denote the sample complexities of two algorithms. From Table 1 and Theorem 2,

$$\mathcal{C}\left(\text{Probe}\right) = \sum_{i=1}^{n} \widetilde{\Theta}\left(n(\widetilde{\Delta}_i)^{-2}\right), \quad \mathcal{C}\left(\text{IIR}\right) = \sum_{i=1}^{n} \widetilde{\Theta}\left((\Delta_i)^{-2}\right), \tag{5}$$

noting that from the proofs, the sample complexity upper bounds are both tight in the worst case.

Under WST with no other conditions assumed, $\Delta_i \leq \widetilde{\Delta}_i$. In particular, when $\widetilde{\Delta}_i/\Delta_i = \Theta(\sqrt{n})$ for all i, then $\mathcal{C}$ (Probe) and $\mathcal{C}$ (IIR) are of the same asymptotic order with respect to n; if $\widetilde{\Delta}_i/\Delta_i = \omega(\sqrt{n})$, then Probe-Rank is asymptotically more sample-efficient than IIR. These phenomena are also reflected in our numerical experiments in Section 6 (see Figure 3).

Remark. It is worth noting that IIR is optimal if the more strict assumption SST as well as some other conditions are made, as shown in [25]. When SST holds, $\widetilde{\Delta}_i = \Delta_i$. Probe-Rank thus suffers from an additional factor of n. This case is also included in our numerical experiment (see Figure 2(a)).

6 Experiments

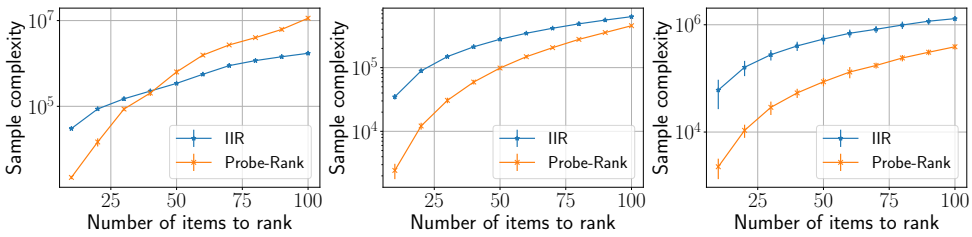
In this section, we present numerical experiments demonstrating the practical performance of Probe-Rank. We compare Probe-Rank with the IIR algorithm, which was shown to outperform all the other baseline algorithms both theoretically and numerically [25].

We study different settings where SST is satisfied, not guaranteed, or violated, but WST always holds, which is consistent with our theory. Specifically, we want to rank n items with the true ranking $\sigma_1 \succ \sigma_2 \succ \cdots \succ \sigma_n$, where n varies over $[10, 100]$. The probabilistic comparison model p_{ij} is generated in different ways to satisfy different assumptions. Note that Δ and Δ_d are tuning parameters in all the following settings.

- `SST`: SST is satisfied. p_{ij} is generated from the MNL model, where $p_{\sigma_i,\sigma_j} = (\exp(s_{\sigma_i} - s_{\sigma_j}) + 1)^{-1}$, and $s_{\sigma_1}, \ldots, s_{\sigma_n}$ is a decreasing sequence where $s_{\sigma_i} = 100\Delta_d \cdot \frac{(n+1-i)}{n}$.
- `WST`: SST does not necessarily hold. Let $p_{i,j} \sim Uni(\frac{1}{2} + \Delta_d, 1)$ for all items $i \succ j$.
- `NON-SST`: SST does not hold. For adjacent items, we have $p_{\sigma_i,\sigma_{i+1}} \sim Uni\left(\frac{1}{2} + \Delta_d, 1\right)$. Otherwise, we have $p_{\sigma_i,\sigma_j} \sim Uni\left(\frac{1}{2} + \frac{\Delta_d}{10}, \frac{1}{2} + \Delta_d\right)$ for $j > i + 1$.
- `ADJ-ASYM`: SST does not hold. This setting is used to verify the asymptotic analysis in Section 5. For adjacent items, we set $p_{\sigma_i,\sigma_{i+1}} = \frac{1}{2} + \Delta_d$. Otherwise, we set $p_{\sigma_i,\sigma_j} = \frac{1}{2} + \frac{\Delta_d}{n^\alpha}$ for $j > i + 1$.
- `ADJ-CNST`: SST does not hold. For adjacent items, we set $p_{\sigma_i,\sigma_{i+1}} = \frac{1}{2} + \Delta$. Otherwise $p_{\sigma_i,\sigma_j} = \frac{1}{2} + \Delta_d$ for $j > i + 1$. Here $\Delta > \Delta_d$.

All experiments are averaged over 100 independent trials. For each trial, the ground truth ranking σ is generated uniformly at random and the comparison probabilities are assigned accordingly. The confidence level δ is fixed to be 0.1. Throughout the experiment, every trial for every algorithm successfully recovered the correct ranking.

We use internal clusters of intel "Skylake" generation CPUs. Each job contains a single model type for item numbers ranging from 10 to 100 with a step size of 10. Models are generated from a job unique random seed shared among the two algorithms. Most jobs with sample complexity smaller than 10^7 terminate in 3 minutes. For $\Delta_d = 0.1$ under the `ADJ-ASYM` model, 3 hours are needed due to high sample complexity. Due to the space limit, more detailed experimental setups and thorough ablation studies can be found in Appendix C.

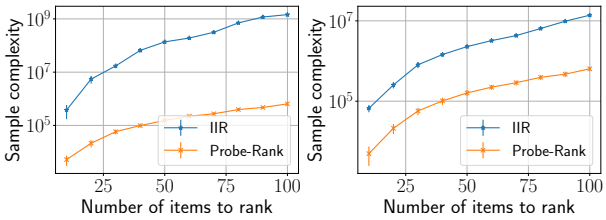


(a) SST: $\Delta_d = 0.3$ (b) WST: $\Delta_d = 0.3$ (c) NON-SST: $\Delta_d = 0.3$

Figure 2: Comparison of sample complexities of Probe-Rank and IIR under various settings. In each subfigure, Δ_d is fixed while the number of items varies.

Performance comparison Figure 2 with y-axis in log-scale shows comparison of IIR and Probe-Ranking under the `SST`, `WST` and `NON-SST` settings. The parameter Δ_d is set to be 0.3. It can be seen that under the `SST` and `WST` settings (Figures 2(a), 2(b)), Probe-Rank consumes less samples than IIR for small n. As n gets larger, however, IIR becomes more sample-efficient due to that Probe-Rank has an additional factor of n in its sample complexity compared with IIR for instances satisfy SST. However, under the `NON-SST` setting where SST does not hold, Probe-Rank has a clear advantage over IIR, as shown in Figure 2(c).

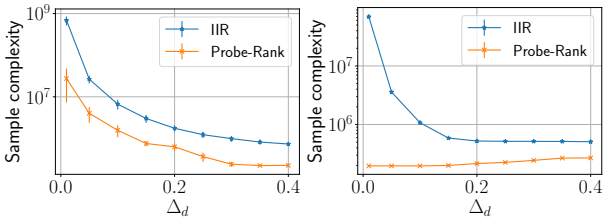
Dependence on n and the probability gaps Following Theorem 2, we verify that the sample complexity of Probe-Rank is lower than IIR when the number of items n gets larger. We use the `ADJ-ASYM` setting to simulate situations where nonadjacent items can be much more difficult to compare. In particular, we choose $\alpha = 1$ (see Figure 3(a)) and $\alpha = 1/2$ (see Figure 3(b)). It can be seen from Figure 3(a) that as the number of items n gets larger, the gap between the two curves also gets larger. This matches our analysis that when $\widetilde{\Delta}_i/\Delta_i = \omega(\sqrt{n})$, then the sample complexity of IIR is of higher order than that of Probe-Rank. When $\widetilde{\Delta}_i/\Delta_i = \Theta(\sqrt{n})$, Figure 3(b) shows that the gap between the two sample complexities varies little as n increases. Our analysis also suggests that sample complexities of two algorithms are of the same order.



(a) ADJ-ASYM: $\Delta_d = 0.3, \alpha = 1$ (b) ADJ-ASYM: $\Delta_d = 0.3, \alpha = 0.5$

Figure 3: Relationship between n and gap Δ_d

Furthermore, we show through the `NON-SST` and `ADJ-CNST` settings that when the probability gaps of nonadjacent item pairs decrease, the advantage of our algorithm will be more and more prominent. In Figure 4, we fix $n = 80$ and let Δ_d vary. Clearly, Probe-Rank has an advantage over IIR in both settings. In particular, Figure 4(b) shows the comparison of two algorithms in the `ADJ-CNST` setting with the probability gaps between adjacent items Δ fixed as 0.4. As the probability gap between nonadjacent items Δ_d varies from 0.01 to 0.4, it can be seen that the sample complexity of Probe-Rank does not vary much. However, the sample complexity of IIR has a positive correlation with $\frac{1}{\Delta_d^2}$. This numerical result matches our analysis that Probe-Ranking is not affected by the comparison probability of nonadjacent items, which does not hold for IIR.



(a) NON-SST: $n = 80$ (b) ADJ-CNST: $n = 80, \Delta = 0.4$

Figure 4: Ablation study on the dependence of the sample complexity on the probability gap Δ_d.

7 Discussion on the lower bound

In this section, we provide some insights about the lower bound for pairwise ranking by proposing a conjecture for a particularly hard instance that satisfies the WST condition.

This problem instance $\mathcal{I}_{WST}$ is constructed as follows. For all $i \succ j$,

$$p_{i,j} = \begin{cases} \frac{1}{2} + \Delta, & \text{if } i \text{ and } j \text{ are adjacent,} \\ \frac{1}{2} + cn^{-10}\Delta^2/\log(1/\delta), & \text{otherwise,} \end{cases}$$

where c and Δ are constants and n^{-10} can be replaced by any other quantity that is smaller than n^{-2}.

By a reduction, any δ-correct algorithm that finds the maximum item for $\mathcal{I}_{WST}$ can be constructed to solve Problem 1, described below. Therefore, a lower bound on the sample complexity for Problem 1 will imply a lower bound of the same order for the maxing (and thus, ranking) problem for $\mathcal{I}_{WST}$.

This lower bound is also a worst-case lower bound for ranking under WST. In the following, we provide an analysis for Problem 1. The reduction technique will be deferred to Appendix D.

Problem 1. Consider vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n$. Each $\boldsymbol{q}_i$ is of length n. Let $q_{i,k}$ denote the k-th entry of $\boldsymbol{q}_i$. Every entry of $\boldsymbol{q}_1$ is Ber$(1/2)$. For $i > 1$, the i-th entry of $\boldsymbol{q}_i$, $q_{i,i}$, is Ber$(1/2 - 2\Delta)$, and other entries are all Ber$(1/2)$. Given the n vectors, but randomly permuted, the agent is asked to predict which vector is $\boldsymbol{q}_1$, i.e., the one with all entries Ber$(1/2)$ by sampling the n^2 entries. All entries are independent.

We conjecture that any δ-correct algorithm for Problem 1 has a sample complexity at least

$$\Omega\big(n^2 \Delta^{-2} \log(1/\delta)\big). \tag{6}$$

We start from viewing it as a hypothesis testing problem. Let $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ be the vectors after permutation. Consider that the agent is asked to determine if $\boldsymbol{p}_1$ satisfies hypothesis H_0:

$$H_0 : \boldsymbol{p}_1 = \boldsymbol{q}_1, \text{ i.e., } p_{1,k} \sim \text{Ber}(1/2), \forall k \in [n].$$

or H_j, in which the j-th entry is biased:

$$H_j : \boldsymbol{p}_1 = \boldsymbol{q}_j, \text{ i.e., } p_{1,j} \sim \text{Ber}(1/2 - 2\Delta); p_{1,k} \sim \text{Ber}(1/2), \forall k \neq j.$$

Suppose the hypothesis testing algorithm $\mathcal{A}$ is δ-correct and stops within T rounds of interactions. We denote $\mathcal{A}(T)$ as the output at the T-th round, which is either 0 (accept H_0) or 1 (reject H_0). For any given $j \neq 1$, by the Bretagnolle–Huber inequality, we have

$$2\delta \geq \mathbb{P}_0(\mathcal{A}(T) \neq 0) + \mathbb{P}_j(\mathcal{A}(T) = 0) \geq \frac{1}{2} e^{-\text{KL}(\mathbb{P}_0^{\mathcal{A}} || \mathbb{P}_j^{\mathcal{A}})}, \tag{7}$$

where $\mathbb{P}_0$ is the probability measure under H_0, and $\mathbb{P}_0^{\mathcal{A}}$ is the probability measure of the canonical bandit model under H_0. In fact, we have the divergence decomposition lemma [20, Lemma 15.1]:

$$\text{KL}(\mathbb{P}_0^{\mathcal{A}} || \mathbb{P}_j^{\mathcal{A}}) = \sum_{k=1}^{n} \mathbb{E}_0[N_k(T)] \text{KL}(\mathbb{P}_{0,k} || \mathbb{P}_{j,k}) = \mathbb{E}_0[N_j(T)] \text{KL}(\text{Ber}(1/2) || \text{Ber}(1/2 - 2\Delta)), \tag{8}$$

where $\mathbb{E}_0$ denotes the expectation under H_0; $\mathbb{E}_0[N_k(T)]$ denotes under H_0, the expected number of queries for the entry $p_{1,k}$ within T rounds.; $\mathbb{P}_{0,k}$, $\mathbb{P}_{j,k}$ are the Bernoulli distributions specified by $p_{1,k}$ under H_0, H_j, respectively. The second equality is due to the fact that the only difference between H_0 and H_j is that the j-th entry has different Bernoulli distributions.

Combining the two inequality above gives:

$$\mathbb{E}_0[N_j(T)] \text{KL}(\text{Ber}(1/2) || \text{Ber}(1/2 - 2\Delta)) \geq \log(1/4\delta). \tag{9}$$

Since KL(Ber$(1/2)$||Ber$(1/2 - x)) < (4x)^2$ for all $x < 2/9$, we get $\mathbb{E}_0[N_j(T)] = \Omega(\Delta^{-2} \log(1/\delta))$. Thus, the total expected number of queries under H_0 will be $\Omega(n\Delta^{-2} \log(1/\delta))$.

In Problem 1, there are in total n vectors. To identify which vector satisfies H_0 requires at least $\Omega(n)$ attempts, with each attempt costs $\Omega(n\Delta^{-2} \log(1/\delta))$. Thus, we conjecture any δ-correct algorithm requires $\Omega\big(n^2 \log(1/\delta)/\Delta^2\big)$ queries.

8 Conclusion and future work

In this paper, we studied the problem of exact ranking under the most general assumption WST. We proposed a δ-correct algorithm Probe-Rank, and derived an instance-wise upper bound on its sample complexity. The upper bound shows that the performance of Probe-Rank only depend on the comparison probabilities of adjacent items and thus improves existing results when SST does not hold. Numerical results also suggest that our ranking algorithm outperforms state-of-the-art. A discussion over the lower bound for pairwise ranking is also provided. We propose a conjecture that in the worst case, any algorithm has sample complexity n times the number of comparisons needed for comparing all adjacent items. However, it remains an open problem whether our conjecture holds and will be left to future work.

References

[1] Nir Ailon, Zohar Karnin, and Thorsten Joachims. Reducing dueling bandits to cardinal bandits. In *International Conference on Machine Learning*, pages 856–864. PMLR, 2014.

[2] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 119–126, 2010.

[3] Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: A survey. *J. Mach. Learn. Res.*, 22:7–1, 2021.

[4] Róbert Busa-Fekete, Balazs Szorenyi, Weiwei Cheng, Paul Weng, and Eyke Hüllermeier. Top-k selection based on adaptive sampling of noisy preferences. In *International Conference on Machine Learning*, pages 1094–1102. PMLR, 2013.

[5] Andrew Caplin and Barry Nalebuff. Aggregation and social choice: A mean voter theorem. *Econometrica: Journal of the Econometric Society*, pages 1–23, 1991.

[6] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202, 2013.

[7] Vincent Conitzer and Tuomas Sandholm. Communication complexity of common voting rules. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 78–87, 2005.

[8] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622, 2001.

[9] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*, pages 255–270. Springer, 2002.

[10] Moein Falahatgar, Yi Hao, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. Maxing and ranking with few assumptions. *Advances in Neural Information Processing Systems*, 30, 2017.

[11] Moein Falahatgar, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Maximum selection and ranking under noisy comparisons. In *International Conference on Machine Learning*, pages 1088–1096. PMLR, 2017.

[12] Moein Falahatgar, Ayush Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. The limits of maxing, ranking, and preference learning. In *International conference on machine learning*, pages 1427–1436. PMLR, 2018.

[13] Roger H Farrell. Asymptotic behavior of expected sample size in certain one sided tests. *The Annals of Mathematical Statistics*, pages 36–72, 1964.

[14] Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.

[15] Björn Haddenhorst, Viktor Bengs, and Eyke Hüllermeier. On testing transitivity in online preference learning. *Machine Learning*, 110(8):2063–2084, 2021.

[16] Reinhard Heckel, Nihar B Shah, Kannan Ramchandran, and Martin J Wainwright. Active ranking from pairwise comparisons and when parametric assumptions do not help. *The Annals of Statistics*, 47(6):3099–3126, 2019.

[17] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*, 19, 2006.

[18] Tao Jin, Pan Xu, Quanquan Gu, and Farzad Farnoud. Rank aggregation via heterogeneous thurstone preference models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[19] Sumeet Katariya, Lalit Jain, Nandana Sengupta, James Evans, and Robert Nowak. Adaptive sampling for coarse ranking. In *International Conference on Artificial Intelligence and Statistics*, pages 1839–1848. PMLR, 2018.

[20] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[21] Soheil Mohajer, Changho Suh, and Adel Elmahdy. Active learning for top-k rank aggregation from noisy comparisons. In *International Conference on Machine Learning*, pages 2488–2497. PMLR, 2017.

[22] Thomas Pfeiffer, Xi Alice Gao, Yiling Chen, Andrew Mao, and David G Rand. Adaptive polling for information aggregation. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[23] Chris Piech, Jonathan Huang, Zhenghao Chen, Chuong Do, Andrew Ng, and Daphne Koller. Tuned models of peer assessment in moocs. *arXiv preprint arXiv:1307.2579*, 2013.

[24] Wenbo Ren, Jia Liu, and Ness B Shroff. Pac ranking from pairwise and listwise queries: Lower bounds and upper bounds. *arXiv preprint arXiv:1806.02970*, 2018.

[25] Wenbo Ren, Jia Kevin Liu, and Ness Shroff. On sample complexity upper and lower bounds for exact ranking from noisy comparisons. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] Aadirupa Saha and Aditya Gopalan. Active ranking with subset-wise preferences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3312–3321. PMLR, 2019.

[27] Aadirupa Saha and Aditya Gopalan. From pac to instance-optimal sample complexity in the plackett-luce model. In *International Conference on Machine Learning*, pages 8367–8376. PMLR, 2020.

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

[29] Balázs Szörényi, Róbert Busa-Fekete, Adil Paul, and Eyke Hüllermeier. Online rank elicitation for plackett-luce: A dueling bandits approach. *Advances in Neural Information Processing Systems*, 28, 2015.

[30] Amos Tversky. Intransitivity of preferences. *Psychological review*, 76(1):31, 1969.

[31] Yue Wu, Tao Jin, Hao Lou, Pan Xu, Farzad Farnoud, and Quanquan Gu. Adaptive sampling for heterogeneous rank aggregation from noisy pairwise comparisons. In *International Conference on Artificial Intelligence and Statistics*, pages 11014–11036. PMLR, 2022.

[32] Yisong Yue and Thorsten Joachims. Beat the mean bandit. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 241–248. Citeseer, 2011.

[33] Masrour Zoghi, Zohar S Karnin, Shimon Whiteson, and Maarten De Rijke. Copeland dueling bandits. *Advances in neural information processing systems*, 28, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We mentioned that the proposed algorithm has an additional factor of n compared with state-of-the-art. We also illustrated that the proposed lower bound is a conjecture.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We believe that our work will not cause any negative societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Due to space limitation, a part of the proofs is deferred to in Appendices A and D.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code is provided as supplementary material for the review process for anonymity. Upon acceptance of the paper, it will be uploaded online for public access with GPLv3 license.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A] We do not use any existing assets. For algorithms, we implement our version of algorithms in cited resources. For dataset, it is generated purely by code.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of the sample complexity upper bound on Probe-Rank

In this section, we prove our theoretical upper bound presented in Theorem 2, Section 5. Before stating the proof, we make the correction that in line 5 of Subroutine 3, SC should be called over all item pairs (a, b) such that $\tau_{a,b} = \min_{(x,y)\in P} \tau_{x,y}$, instead of over just one pair and breaking ties arbitrarily. All such $\tau_{a,b}$ should be increased by 1, and outcomes of all SC instances should be recorded and used to update T.

We first show in the following lemma that the subroutine Successive-Comparison returns desired outcomes with high probability. Given an item pair (i, j) with probability gap $\Delta_{i,j} > 0$ and a positive integer τ, we say $\mathrm{SC}\,(i, j, \delta, \tau)$ is successful if one of the following two events holds,

$$\mathcal{E}_1 = \{\Delta_{i,j} \geq \epsilon_\tau \text{ and SC correctly returns } [i, j]\}, \tag{10}$$

$$\mathcal{E}_2 = \{\Delta_{i,j} < \epsilon_\tau \text{ and SC returns 'unsure' or } [i, j]\}. \tag{11}$$

Lemma 3. *For an item pair (i, j) with probability gap $\Delta_{i,j} > 0$ and a positive integer τ, $\mathrm{SC}\,(i, j, \delta, \tau)$ is successful with probability at least $1 - \frac{\delta}{c\tau^2}$, where $c = \frac{\pi^2}{6}$.*

Proof of Lemma 3. Hoeffding's inequality gives that

$$\Pr\left(\widehat{p}_i - p_{i,j} \leq -\frac{1}{2}\epsilon_\tau\right) \leq \exp\left(-2b_\tau\left(\frac{1}{2}\epsilon_\tau\right)^2\right) \leq \frac{\delta}{c\tau^2}. \tag{12}$$

Therefore, the probability that SC outputs $[j, i]$ is at most

$$\Pr\left(\widehat{p}_i - \frac{1}{2} < -\frac{1}{2}\epsilon_\tau\right) \leq \Pr\left(\widehat{p}_i - p_{i,j} \leq -\frac{1}{2}\epsilon_\tau\right) \leq \frac{\delta}{c\tau^2}, \tag{13}$$

and the probability that SC returns $[i, j]$ or 'unsure' is at least $1 - \frac{\delta}{c\tau^2}$.

Further, if $\Delta_{i,j} \geq \epsilon_\tau$, the probability that SC returns $[i, j]$ is at least

$$\Pr\left(\widehat{p}_i - \frac{1}{2} > \frac{1}{2}\epsilon_\tau\right) = \Pr\left(\widehat{p}_i > \frac{1}{2} + \frac{1}{2}\epsilon_\tau\right) \geq \Pr\left(\widehat{p}_i > p_{i,j} - \frac{1}{2}\epsilon_\tau\right) \geq 1 - \frac{\delta}{c\tau^2}. \tag{14}$$

This completes the proof. □

By Lemma 3, with high probability, SC do not return the incorrect ordering. Further, if τ is large enough, then SC is guaranteed to return the correct ordering. We use Lemma 3 to show the theoretical performance of Probe-Rank.

Theorem 2. *Let $\delta > 0$ be an arbitrary constant. For all problem instances satisfying the Weak Stochastic Transitivity (WST) property, with probability at least $1 - \delta$, Probe-Rank returns the true ranking of n items and conducts at most*

$$O\left(n\sum_{i=1}^{n}\left(\widetilde{\Delta}_i^{-2}\right)\left(\log\log\left(\widetilde{\Delta}_i^{-1}\right) + \log\left(n/\delta\right)\right)\right) \tag{4}$$

comparisons, where $\widetilde{\Delta}_i$ is defined as in (2)*.*

Proof of Theorem 2. Define events

$$\mathcal{E}_{i,j}(\tau) = \{\mathrm{SC}\left(i, j, 2\delta/n^2, \tau\right) \text{ is successful}\}. \tag{15}$$

Define the bad event

$$\mathcal{E}^{bad} = \cup_{(i,j)\in[n]^2} \cup_{\tau=1}^{\infty} \left(\mathcal{E}_{i,j}(\tau)\right)^c. \tag{16}$$

By the union bound and Lemma 3

$$\Pr\left(\mathcal{E}^{bad}\right) \leq \sum_{(i,j)\in[n]^2}\sum_{\tau=1}^{\infty}\frac{2\delta}{cn^2\tau^2} \leq \sum_{\tau=1}^{\infty}\frac{\delta}{c\tau^2} \leq \delta. \tag{17}$$

In the following, we assume that $\mathcal{E}^{bad}$ does not happen.

Correctness. We show that when $\mathcal{E}^{bad}$ does not happen, in every round t, Probe-Max$(S_t, 2\delta/n^2)$ (line 4 of Algorithm 2) correctly returns the most preferred item in the set of remaining items S_t. Since the probability of $\mathcal{E}^{bad}$ is upper bounded by δ, the correctness of Probe-Rank thus follows.

Let x be the most preferred items in S_t. When $\mathcal{E}^{bad}$ does not happen, all comparison results returned by SC are correct and T is always consistent with the true ranking. Thus, no item in S_t is known to rank higher than x, i.e., at the beginning of Subroutine 3, $x \in U$. Moreover, x will not be eliminated from U since x will not lose to any other item in S_t during calls of SC.

We show that any other item in U will be eliminated from U after a finite number of iterations of the while loop in Probe-Max. Let $y \neq x$ be an item in U. Since x is the maximum, $y \prec x$ in the true ranking. Whenever $\epsilon_{\tau_{y,x}} \leq \Delta_{x,y}$, a successful call of SC $\left(x, y, 2\delta/n^2, \tau_{x,y}\right)$ will return the result $x \succ y$ and remove y from U if $\mathcal{E}^{bad}$ does not happen. Since $\epsilon_{\tau_{y,x}}$ converges to 0, there must exist $\tau^*_{x,y}$ such that $\epsilon_{\tau^*_{x,y}} \leq \Delta_{x,y}$. After each execution of SC, the corresponding τ value increases by one, therefore after at most $\binom{n}{2}\tau^*_{x,y}$ iterations of the while loop, SC $\left(x, y, 2\delta/n^2, \tau^*_{x,y}\right)$ must have been called. The same argument holds for any $y \in U, y \neq x$.

Sample complexity. We first note the asymptotic behavior that for any $N > 0$,

$$\sum_{\tau=1}^{N} b_\tau \leq \sum_{\tau=1}^{N} \frac{2}{4^{-\tau}} \log \frac{c\tau^2 n^2}{\delta} \leq \sum_{\tau=1}^{N} \frac{2}{4^{-\tau}} \log \frac{cN^2 n^2}{\delta} = O\left(4^N \log \frac{cN^2\delta^2}{\delta}\right) = O\left(b_N\right). \quad (18)$$

Without loss of generality, we assume the true ranking is $1 \succ 2 \succ \cdots \succ n$. When $\mathcal{E}^{bad}$ does not happen, all comparison results returned by SC coincide with the true ranking. Therefore, for every $i \in [n-1]$, item i belongs to $S_1, S_2, \ldots, S_i$ and gets eliminated during the execution of Probe-Max$\left(S_i, 2\delta/n^2\right)$.

Recall that SC is only called over item pairs in which at least one of them is a maximal element. For every SC called on items a, b, if a is maximal, we say item a initializes the comparison and we charge the number of comparisons taken by SC to item a (if both a and b are maximal, we charge the number of samples to both). Let $c(a)$ denote the number of comparisons charged to a. The total sample complexity of Probe-Rank is thus at most $\sum_{a\in[n]} c(a)$.

Fix $i \in [n]$. We use τ_i° to denote the value of $\tau_{i,i-1}$ when the order between i and $i-1$ is revealed. Define $\tau_1^\circ = 0$ for completeness. We note that the order between i and $i-1$ can not be inferred from any other comparison results therefore can only be returned by SC $\left(i, i-1, 2\delta/n^2, \tau_i^\circ\right)$. When $\mathcal{E}^{bad}$ does not happen, $\tau_i^\circ \leq \left\lceil \log \frac{1}{\Delta_{i,i-1}} \right\rceil$ since a successful call of SC $\left(i, i-1, 2\delta/n^2, \left\lceil \log \frac{1}{\Delta_{i,i-1}} \right\rceil\right)$ will return the order.

For each $j \neq i$, we use $\tau^*_{i,j}$ to denote the value of $\tau_{i,j}$ when the last time SC is initialized by i and called over i, j before the beginning of Probe-Max$\left(S_i, 2\delta/n^2\right)$. In other words, for any $\tau > \tau^*_{i,j}$, if SC $\left(i, j, 2\delta/n^2, \tau\right)$ is called in Probe-Max$\left(S_t, 2\delta/n^2\right)$ for some $t < i$, then it must not be initialized by i. Moreover, we use $\tau^t_{i,j}$ to denote the value of $\tau_{i,j}$ right after Probe-Max$(S_t, 2\delta/n^2)$ terminates. Since i is ranked and removed from T after Probe-Max$(S_i, 2\delta/n^2)$ is called, $\tau^i_{i,j}$ is also the value of $\tau_{i,j}$ when Probe-Rank terminates. It is clear that

$$c(i) \leq \sum_{j\neq i} \sum_{\tau=1}^{\tau^*_{i,j}} b_\tau + \sum_{j\neq i} \sum_{\tau=\tau^{i-1}_{i,j}+1}^{\tau^i_{i,j}} b_\tau. \quad (19)$$

We consider the first term on the right-hand side of (19). Before Probe-Max$\left(S_{i-1}, 2\delta/n^2\right)$ terminates, item $i-1$ is in T. Therefore, whenever i is a maximal element, the order between i and $i-1$ must have not been revealed. So when i initializes the comparison SC $\left(i, j, 2\delta/n^2, \tau^*_{i,j}\right)$, 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 further no larger than τ_i°. The same argument holds for any j. It follows that

$$\sum_{j\neq i} \sum_{\tau=1}^{\tau^*_{i,j}} b_\tau \leq \sum_{j\neq i} \sum_{\tau=1}^{\tau^*_{i,j}} b_\tau \leq \sum_{\tau=1}^{\tau_i^\circ} n b_\tau. \quad (20)$$

Next, we bound the second term on the right-hand side of (19). Note that if there is no SC called during Probe-Max$(S_i, 2\delta/n^2)$, then $\sum_{j\neq i}\sum_{\tau=\tau_{i,j}^{i-1}+1}^{\tau_{i,j}^i} b_\tau = 0$. So it suffices to consider the case when at least one instance of SC is called during Probe-Max$(S_i, 2\delta/n^2)$. Consider the last group of SC called in Probe-Max$(S_i, 2\delta/n^2)$, here group means that there might be multiple item pairs whose τ values are the minimum in P. Denote their τ values by τ^i. There must be some SC $\left(a_i, b_i, 2\delta/n^2, \tau^i\right)$ 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 $\mathcal{E}^{bad}$ does not happen, a_i is not the maximum in S_i so $a_i > i$. Thus, item $a_i - 1$ is also in S_i and before the call of SC $\left(a_i, b_i, 2\delta/n^2, \tau^i\right)$, the ordering between $a_i - 1$ and a_i is not revealed, i.e., $\tau^i \leq \tau_{a_i}^\circ$. Moreover, $\tau_{i,j}^i \leq \tau^i$ by the fact that we always compare item pairs with the smallest τ values. It follows that

$$\sum_{j\neq i}\sum_{\tau=\tau_{i,j}^{i-1}+1}^{\tau_{i,j}^i} b_\tau \leq n\sum_{\tau=1}^{\tau^i} b_\tau = O\left(nb_{\tau^i}\right). \tag{21}$$

The same argument holds for all $i \in [n-1]$.

Consider the sets

$$\mathcal{D}_1 = \{b_{\tau^i} : i = 1, 2, \ldots, n-1\}, \quad \mathcal{D}_2 = \cup_{i=2}^n \mathcal{D}_2^i = \cup_{i=2}^n \{b_\tau : \tau = 1, 2, \ldots, \tau_i^\circ\}. \tag{22}$$

We claim that if $i_1 \neq i_2$, then the pairs (a_{i_1}, τ^{i_1}) and (a_{i_2}, τ^{i_2}) do not equal. With the facts that $a_i > i$ and $\tau^i \leq \tau_{a_i}^\circ$, there is an injective mapping from $\mathcal{D}_1$ to $\mathcal{D}_2$ given by b_{τ^i} is mapped to the element b_{τ^i} in $\mathcal{D}_2^{a_i}$. It follows that

$$\sum_{i=1}^{n-1} O\left(nb_{\tau^i}\right) = O\left(\sum_{x\in\mathcal{D}_1} nx\right) \leq O\left(\sum_{x\in\mathcal{D}_2} nx\right) = O\left(\sum_{i=2}^{n}\sum_{\tau=1}^{\tau_i^\circ} nb_\tau\right) \tag{23}$$

The reason for pairs (a_{i_1}, τ^{i_1}) and (a_{i_2}, τ^{i_2}) equal if and only if $i_1 = i_2$ is as follows. Let $i_2 > i_1$ and suppose $a_{i_1} = a_{i_2} = a$. When SC $\left(a, b_{i_1}, 2\delta/n^2, \tau^{i_1}\right)$ is called, SC $\left(a, b, 2\delta/n^2, \tau^{i_1}\right)$ for all b such that $(a, b) \notin T$ and $\tau_{a,b} = \tau^{i_1}$ are also called. It follows that $\tau_{a,b} > \tau^i$ for all such b after this point. When SC $\left(a, b_{i_2}, 2\delta/n^2, \tau^{i_2}\right)$ is called, the order between a and b_{i_2} is not know and thus also not known when SC $\left(a, b_{i_1}, 2\delta/n^2, \tau^{i_1}\right)$ was called. So τ^{i_2} must be larger than τ^{i_1}.

Combining (19), (20) and (23) gives,

$$\sum_{i=1}^{n} c(i) \leq \sum_{i=2}^{n}\sum_{j\neq i}\sum_{\tau=1}^{\tau_{i,j}^*} b_\tau + \sum_{i=1}^{n-1}\sum_{j\neq i}\sum_{\tau=\tau_{i,j}^{i-1}+1}^{\tau_{i,j}^i} b_\tau \tag{24}$$

$$\leq O\left(\sum_{i=2}^{n}\sum_{\tau}^{\tau_i^\circ} nb_\tau\right) = O\left(n\sum_{i=2}^{n} b_{\tau_i^\circ}\right). \tag{25}$$

The desired sample complexity follows from $\tau_i^\circ \leq \left\lceil \log \frac{1}{\Delta_{i,i-1}} \right\rceil$ and

$$b_{\lceil \log \frac{1}{\Delta} \rceil} = O\left(\frac{1}{\Delta^2}\left(\log\log\frac{1}{\Delta} + \log\frac{n}{\delta}\right)\right), \tag{26}$$

which completes the proof. □

B A sample-efficient variant of Probe-Rank

In this section, we present a variant of Probe-Rank, named *Probe-Rank-SE*. When demonstrating more detailed experiments in Appendix C, Probe-Rank-SE is also included and is shown to have better practical performance. However, we will not prove its correctness due to the high similarity it shares with Probe-Rank.

Compared with Probe-Rank, the variant Probe-Rank-SE finds the ranking also by performing $n-1$ steps of maxing and differs only in the subroutine for collecting comparison samples. Specifically,

Probe-Rank-SE takes queries from all unknown item pairs simultaneously. Comparison results for pairs that terminate earlier are still collected and stored in the graph T, which represents our current knowledge about the ranking. We use T to decide whether to pause, drop or resume comparisons of remaining item pairs.

We adopt the Successive Elimination (SE) algorithm from [9], shown in Algorithm 4, as a procedure to perform comparisons.

Subroutine 4 Successive Elimination (modified for comparing two items)

Input: items i, j, confidence level δ
$t = 1$;
while true **do**
  Compare i and j for 2^t times; Let $\widehat{p}_i^t$ be the winning rate of i;
  Let $\alpha_t = \sqrt{\frac{\log(ct^2/\delta)}{2^t}}$, $c = \frac{\pi^2}{3}$;
  return $i \succ j$ **if** $\widehat{p}_i^t - \frac{1}{2} > \alpha_t$; **return** $j \succ i$ **if** $\widehat{p}_i^t - \frac{1}{2} < -\alpha_t$; $t = t + 1$ **else**;
end while

It was shown that with probability at least $1 - \delta$, Subroutine 4 correctly returns the more preferred item between i and j using at most $O\left(\frac{1}{\Delta_{i,j}^2}\left(\log\frac{1}{\delta} + \log\log\frac{1}{\Delta_{i,j}}\right)\right)$ comparisons [9, Remark 1].

In Probe-Rank-SE, we do not call SE directly, rather, SE is used as a black-boxed unit that repeatedly collects query samples from the input pair i, j. Moreover, after every sample, it generates feedback which is either Null, $i \succ j$ or $j \succ i$, where Null corresponds to that the number of samples has not accumulated to 2^t or $\left|\widehat{p}_i^t - \frac{1}{2}\right| < \alpha_t$; feedback $i \succ j$ and $j \succ i$ correspond to that inside the black box, SE actually terminates and returns the order between i and j. Note that the SE procedure can be replaced by any algorithm that can rank two items, including all best-arm-identification algorithms.

Denote the instance of Successive Elimination that runs over items i, j with confidence level δ as $\mathrm{SE}_{i,j}(\delta)$. When the value of δ is given without ambiguity, we will drop the dependence and write $\mathrm{SE}_{i,j}$ as a shorthand. We define two operations on $\mathrm{SE}_{i,j}$, named $\mathrm{advance}$ and feed. The $\mathrm{advance}$ operation returns one of the three possible internal outcomes, Null, $i \succ j$ or $j \succ i$. The feed operation is used for simulating the sampling process. We write $\mathrm{feed}\left(\mathrm{SE}_{i,j}, Y_{i,j}\right)$ to represent that $\mathrm{SE}_{i,j}$ is fed with a comparison sample $Y_{i,j}$. As a black-boxed unit, before $\mathrm{advance}$ returns one of $i \succ j$ and $j \succ i$, $\mathrm{advance}$ and feed operations are invoked in an alternating fashion. The idea of viewing a sampling subroutine as a black-box controlled by artificial operations was also used in [1], but for a different problem setting.

Probe-Rank-SE is presented in Algorithm 5. We initialize $\binom{n}{2}$ independent instances of $\mathrm{SE}_{i,j}\left(2\delta/n^2\right)$, each for obtaining the order between an item pair $(i, j), 1 \leq i < j \leq n$. The probability of being unable to recover the true ranking is thus upper bounded by probability that at least one of the SE instances fails, which is at most δ. Same as Probe-Rank, we use T to denote the transitive closure composed of results returned by the SE instances.

Algorithm 5 Probe-Rank-SE

Input: items $[n]$, confidence level δ
$S_1 = [n], Ans = [0]^n$; initialize T;
initialize $\mathrm{SE}_{i,j}\left({}^{2\delta}/_{n^2}\right)$ for all $1 \leq i < j \leq n$;
for t from 1 to $n - 1$ **do**
  i_{max} =Probe-Max-SE(S_t);
  remove i_{max} from T; $Ans[t-1] = i_{max}$; $S_{t+1} = S_t \setminus \{i_{max}\}$;
end for
$Ans[n-1] = S_n[0]$; **return** Ans;

The procedure Probe-Max-SE serves as a switch for the SE instances. Let S_t^2 denote the set of unordered item pairs $\{(i, j) : i, j \in S_t, i \neq j\}$. In each round t, all SE instances for 'legitimate' pairs in S_t^2 are turned on and take queries in a round-robin fashion. 'legitimate' pairs are similarly

defined as in Probe-Rank. A pair (i, j) is 'legitimate' if the order between i, j is unknown, i.e., not in T, and at least one of i and j is a maximal element in S_t.

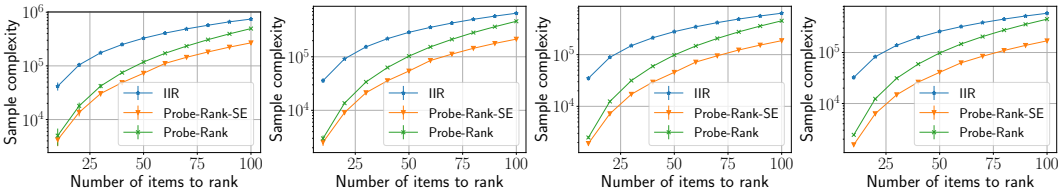
Algorithm 6 Probe-Max-SE(S_t)

```
Let U be sets of maximal elements according to T
while |U| ≥ 1 do
    C = [ ]
    for (i, j) in S_t^2 do
        if (i ∈ U or j ∈ U) and (i, j) ∉ T then
            compare i with j once and get result Y_{i,j}; feed (SE_{i,j}(δ/n^2), Y_{i,j})
            if advance(SE_{i,j}(2δ/n^2)) == i ≻ j then C.append([i, j]);
            else if advance(SE_{i,j}(2δ/n^2)) == j ≻ i then C.append([j, i]);
            end if
        end if
    end for
    for w, l in C do
        if (w, l) ∉ T then
            T = T ∪ (w ≻ l);
            if |U| > 1 and l ∈ U then U = U \ {l};
            end if
        end if
    end for
end while
return U[0];
```

C Additional experiments

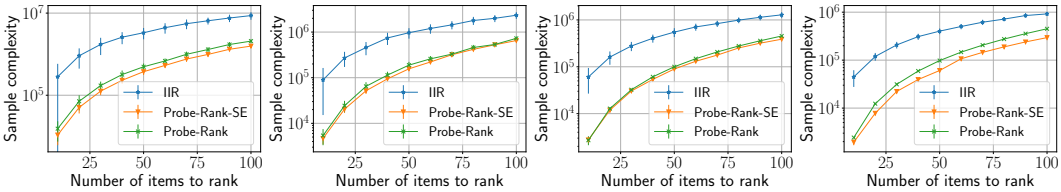
In this section, we present more detailed numerical experiments comparing the sample complexities of Probe-Rank, Probe-Rank-SE and the state-of-the-art algorithm IIR by Ren et al. [25]. In particular, we focus on the `WST`, `NON-SST` and `ADJ-ASYM` settings and perform these three algorithms with various parameters. Same as the results presented in Section 6, all experiments are averaged over 100 independent trials. For each trial, the ground truth ranking σ is generated uniformly at random and the comparison probabilities are assigned according to the chosen setting. The confidence level δ is fixed to be 0.1. Throughout the experiment, every trial for every algorithm successfully recovered the correct ranking. Moreover, for IIR, if the rank has not been recovered after the sample complexity reaches 10^9, we manually stop the ranking process and record the sample complexity as 10^9 to avoid extremely large running times. Note that the extreme cases happen in Figures 7(a), 7(b) 7(c) and 10(d).

Figures 5, 6 and 7 compare the three algorithms under different settings where the difficulty parameter Δ_d is fixed and the number of items n varies from 10 to 100. Figures 8, 9 and 10 compare the three algorithms under different settings where the number of items n is fixed and the difficulty parameter Δ_d varies from 0.1 to 0.4. It can be seen that Probe-Rank and its variant always consumes less samples than IIR to recover the true ranking. Note that in the WST setting, comparison probabilities are all identically distributed and thus on average, adjacent items are as hard as nonadjacent items to compare. When Δ_d is fixed, as n gets larger and larger, IIR will eventually outperforms Probe-Rank. This is consistent with our theoretical results presented in Section 5. Moreover, as indicated by the experimental results, Probe-Rank-SE can further reduce the sample complexity compared with Probe-Rank.



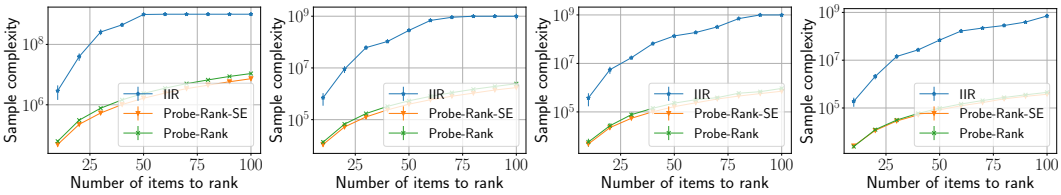
(a) WST: $\Delta_d = 0.1$ (b) WST: $\Delta_d = 0.2$ (c) WST: $\Delta_d = 0.3$ (d) WST: $\Delta_d = 0.4$

Figure 5: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the WST setting. In each subfigure, Δ_d is fixed while the number of items varies.

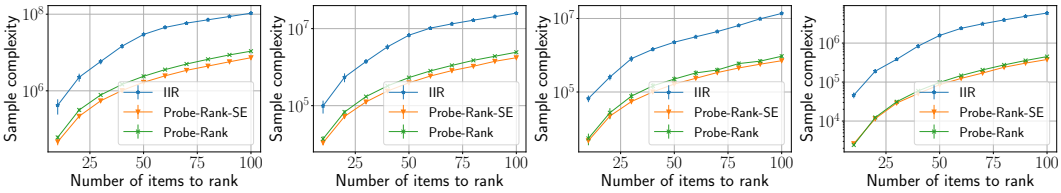


(a) NON-SST: $\Delta_d = 0.1$ (b) NON-SST: $\Delta_d = 0.2$ (c) NON-SST: $\Delta_d = 0.3$ (d) NON-SST: $\Delta_d = 0.4$

Figure 6: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the NON-SST setting. In each subfigure, Δ_d is fixed while the number of items varies.

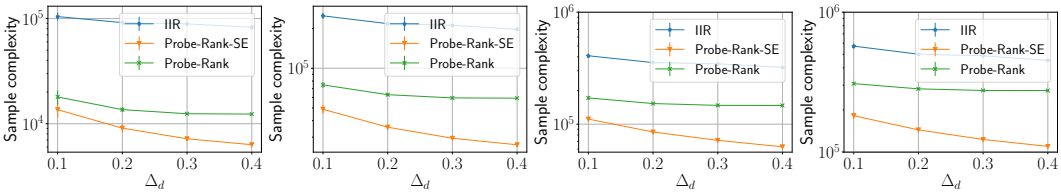


(a) $\Delta_d = 0.1, \alpha = 1$ (b) $\Delta_d = 0.2, \alpha = 1$ (c) $\Delta_d = 0.3, \alpha = 1$ (d) $\Delta_d = 0.4, \alpha = 1$



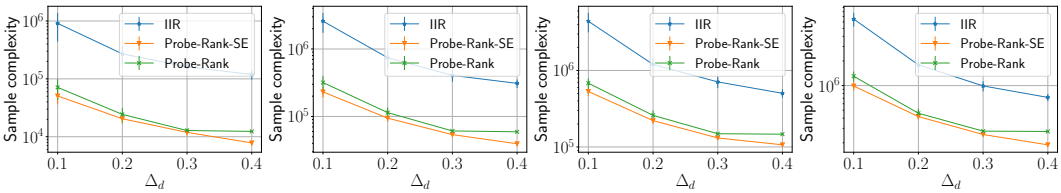
(e) $\Delta_d = 0.1, \alpha = 0.5$ (f) $\Delta_d = 0.2, \alpha = 0.5$ (g) $\Delta_d = 0.3, \alpha = 0.5$ (h) $\Delta_d = 0.4, \alpha = 0.5$

Figure 7: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the ADJ-ASYM setting. In each subfigure, Δ_d and α are fixed while the number of items varies.



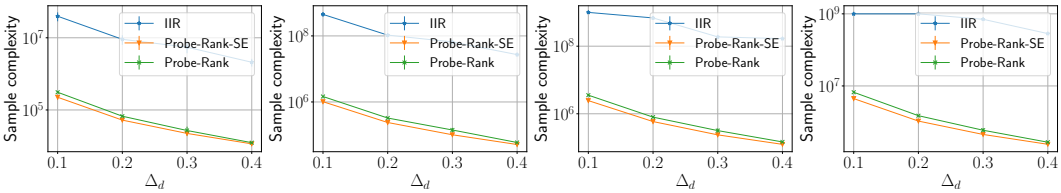
(a) WST, $n = 20$ (b) WST, $n = 40$ (c) WST, $n = 60$ (d) WST, $n = 80$

Figure 8: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the WST setting. In each subfigure, n is fixed while Δ_d varies.

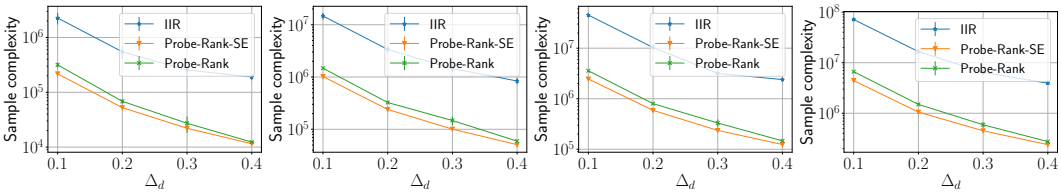


(a) NON-SST, $n = 20$ (b) NON-SST, $n = 40$ (c) NON-SST, $n = 60$ (d) NON-SST, $n = 80$

Figure 9: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the NON-SST setting. In each subfigure, n is fixed while Δ_d varies.



(a) $\alpha = 1, n = 20$ (b) $\alpha = 1, n = 40$ (c) $\alpha = 1, n = 60$ (d) $\alpha = 1, n = 80$



(e) $\alpha = 1/2, n = 20$ (f) $\alpha = 1/2, n = 40$ (g) $\alpha = 1/2, n = 60$ (h) $\alpha = 1/2, n = 80$

Figure 10: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the ADJ-ASYM setting. In each subfigure, n and α are fixed while Δ_d varies.

D Lower bound analysis

In this section, we present the reduction from Problem 1 to the problem of finding the maximum item for $\mathcal{I}_{WST}$. The two problems are restated as follows.

Problem 1. Consider vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n$. Each $\boldsymbol{q}_i$ is of length n. Let $q_{i,k}$ denote the k-th entry of $\boldsymbol{q}_i$. Every entry of $\boldsymbol{q}_1$ is $\text{Ber}(1/2)$. For $i > 1$, the i-th entry of $\boldsymbol{q}_i$, $q_{i,i}$, is $\text{Ber}(1/2 - 2\Delta)$, and other entries are all $\text{Ber}(1/2)$. Given the n vectors, but randomly permuted, the agent is asked to predict which vector is $\boldsymbol{q}_1$, i.e., the one with all entries $\text{Ber}(1/2)$ by sampling the n^2 entries. All entries are independent.

Problem 2 ($\mathcal{I}_{WST}$)**.** Consider n items with an underlying ordering '$\succ$'. The comparison probabilities are defined as:

$$p_{i,j} = \begin{cases} \frac{1}{2} + \Delta, & \text{if } i \succ j \text{ and } i, j \text{ are adjacent,} \\ \frac{1}{2} - \Delta, & \text{if } i \prec j \text{ and } i, j \text{ are adjacent,} \\ \frac{1}{2} + cn^{-10}\Delta^2/\log(1/\delta), & \text{if } i \succ j \text{ and } i, j \text{ are nonadjacent,} \\ \frac{1}{2} - cn^{-10}\Delta^2/\log(1/\delta), & \text{otherwise,} \end{cases}$$

where c and Δ are constants and n^{-10} can be replaced by any other quantity that is smaller than n^{-2}.

We first consider another problem instance $\mathcal{I}_{SYM}$ and show that the maxing problem for $\mathcal{I}_{SYM}$ can be reduced to the maxing problem for $\mathcal{I}_{WST}$. The problem instance $\mathcal{I}_{SYM}$ is modified from $\mathcal{I}_{WST}$ by setting the values of $p_{i,j}$ to be exactly $\frac{1}{2}$.

Problem 3 ($\mathcal{I}_{SYM}$)**.** Consider n items with an underlying ordering '$\succ$'. The comparison probabilities are defined as:

$$p_{i,j} := \begin{cases} \frac{1}{2} + \Delta, \text{ if } i \succ j \text{ and } i, j \text{ are adjacent,} \\ \frac{1}{2} - \Delta, \text{ if } i \prec j \text{ and } i, j \text{ are adjacent,} \\ \frac{1}{2}, \text{ otherwise.} \end{cases}$$

Note that $\mathcal{I}_{SYM}$ does not satisfy the WST condition as the comparison probabilities can be $\frac{1}{2}$. However, any δ-correct algorithm that finds the maximum item for $\mathcal{I}_{WST}$ efficiently can also find the maximum item for $\mathcal{I}_{SYM}$ efficiently, shown as follows.

Reduction from $\mathcal{I}_{SYM}$ to $\mathcal{I}_{WST}$ Let $\mathcal{A}$ be any δ-algorithm that finds the maximum item for any instance that satisfies the WST condition. Algorithm $\mathcal{A}$ is also able to find the maximum item for $\mathcal{I}_{WST}$ with any $c > 0$. Consider any interaction trajectory $\mathcal{T}$ defined by the sequence of comparisons (including the choices for item pairs and their outcomes) with length smaller than $Cn^2 \log(1/\delta)/\Delta^2$ for some constant C. Under the two instances $\mathcal{I}_{SYM}$ and $\mathcal{I}_{WST}$, the probabilities of occurrences of $\mathcal{T}$, denoted $\mathbb{P}_{SYM}$ and $\mathbb{P}_{WST}$, satisfy

$$\frac{\mathbb{P}_{SYM}(\mathcal{T})}{\mathbb{P}_{WST}(\mathcal{T})} \geq \left(\frac{1/2 - cn^{-10}\Delta^2/\log(1/\delta)}{1/2} \right)^{Cn^2 \log(1/\delta)/\Delta^2} \tag{27}$$

$$> \left(\frac{1}{1 + 4cn^{-10}\Delta^2/\log(1/\delta)} \right)^{Cn^2 \log(1/\delta)/\Delta^2} \tag{28}$$

$$\geq e^{-4cn^{-10}\Delta^2/\log(1/\delta) \cdot Cn^2 \log(1/\delta)/\Delta^2} \tag{29}$$

$$= e^{-4cCn^{-8}} \geq 1 - \delta. \tag{30}$$

where the first inequality holds because the likelihood ratio for one query is upper bounded by the base and the number of queries on nonadjacent pairs is bounded by the exponent; the second inequality assumes $cn^{-10}\Delta^2/\log(1/\delta) < 1/4$, which holds for n sufficiently large; the third inequality is due to $(1 + x) \leq e^x$. The last inequality can hold by choosing a small enough c.

If $\mathcal{A}$ solves the maxing problem for $\mathcal{I}_{WST}$ with probability at least $1 - \delta$ and conducts at most $Cn^2 \log(1/\delta)/\Delta^2$ comparisons, then by inequality (30),

$$\mathbb{P}_{SYM}(\mathcal{A} \text{ finds the correct maximum}) = \sum_{\mathcal{T} \in \mathcal{E}_f} \mathbb{P}_{SYM}(\mathcal{T}) \geq (1 - \delta) \sum_{\mathcal{T} \in \mathcal{E}_f} \mathbb{P}_{WST}(\mathcal{T}) \tag{31}$$

$$= (1 - \delta)\, \mathbb{P}_{WST}(\mathcal{A} \text{ finds the correct maximum}) \tag{32}$$

$$\geq (1 - \delta)^2 > 1 - 2\delta, \tag{33}$$

where $\mathcal{E}_f$ denotes the collection of trajectories where $\mathcal{A}$ returns the correct maximum. In other words, $\mathcal{A}$ is also a 2δ-correct algorithm that solves the maxing problem for $\mathcal{I}_{SYM}$ and conducts at most $Cn^2 \log(1/\delta)/\Delta^2$ comparisons. Further, if we force $\mathcal{A}$ to terminated after $Cn^2 \log(1/\delta)/\Delta^2$ comparisons have been made, then $\mathcal{A}$ is still correct with probability at least $1 - \delta$ and with expected number of samples upper bounded by $Cn^2 \log(1/\delta)/\Delta^2$.

The second step is to reduce the maxing problem for $\mathcal{I}_{SNG}$ (defined as follows) to the maxing problem for $\mathcal{I}_{SYM}$.

Problem 4 ($\mathcal{I}_{SNG}$)**.** Consider n items with an underlying ordering '$\succ$'. One can make queries of the form 'if $i \succ j$'. The feedback $Y_{i,j}$ is a binary random variable which takes value 1 is the answer is YES and takes value 0 else. The random variables $Y_{i,j}$ are defined to follow distributions:

$$Y_{i,j} \sim \begin{cases} \text{Ber}(\frac{1}{2} - 2\Delta), \text{ if } i \prec j \text{ and } i, j \text{ are adjacent,} \\ \text{Ber}(\frac{1}{2}), \text{ otherwise.} \end{cases}$$

Reduction from Problem 1 to $\mathcal{I}_{SYM}$ Let $\mathcal{A}$ be any δ-correct algorithm that can find the maximum item for $\mathcal{I}_{SYM}$. Without loss of generality, we can assume that when comparing an item pair i, j, $i < j$, $\mathcal{A}$ takes in answer 1 representing i is more preferred and answer 0 representing j is more preferred. We construct a δ-correct algorithm $\mathcal{A}'$ that can find the maximum item for $\mathcal{I}_{SNG}$ from $\mathcal{A}$:

Given $\mathcal{A}$, whenever $\mathcal{A}$ compares item pair (i, j),
with probability $1/2$, $\mathcal{A}'$ queries 'if $i \succ j$', gets the sample Y and feeds Y to $\mathcal{A}$;
with probability $1/2$, $\mathcal{A}'$ queries 'if $j \succ i$', gets the sample Y and feeds $1 - Y$ to $\mathcal{A}$. Whenever $\mathcal{A}$ terminates and return an item, $\mathcal{A}'$ also terminates and return the same item.

It is clear that, if $\mathcal{A}$ queries an adjacent pair i, j with $i \succ j$, the feedback Y is an average over $Y_{i,j}$ (Ber($\frac{1}{2}$)) and $1 - Y_{j,i}$ (1-Ber($\frac{1}{2} - 2\Delta$)), which is Ber($\frac{1}{2} + \Delta$); if i, j are adjacent and $i \prec j$, the feedback Y is an average of Ber($\frac{1}{2} - 2\Delta$) and $1-$Ber($\frac{1}{2}$), which is Ber($\frac{1}{2} - \Delta$); if i, j are nonadjacent, the feedback Y is an average of two Ber($\frac{1}{2}$) random variables, which is still Ber($\frac{1}{2}$). Therefore, $\mathcal{A}$ gets the same feedback when it is performed over $\mathcal{I}_{SYM}$. If $\mathcal{A}$ is a δ-correct maxing algorithm for $\mathcal{I}_{SYM}$ and conducts at most $Cn^2 \log(1/\delta)/\Delta^2$ comparisons on average, then $\mathcal{A}'$ is a δ-correct maxing algorithm for $\mathcal{I}_{SNG}$ with the same sample complexity.

Finally, we consider Problem 1. We note that querying if $i \succ j$ is equivalent of taking a sample from the variable $p_{i,j}$ and the maximum item for $\mathcal{I}_{SNG}$ is equivalent of the vector $\boldsymbol{q}_1$ whose entries are all Ber($\frac{1}{2}$). So the two problems are equivalent.

To summarize, if there exists a δ-correct algorithm $\mathcal{A}$ that solves the maxing problem for $\mathcal{I}_{WST}$ and conducts at most $Cn^2 \log(1/\delta)/\Delta^2$ on average, then with the reduction, we can conclude there exists a 2δ-correct algorithm $\mathcal{A}'$ that solves Problem 1 with the same sample complexity. Since the above argument holds for any $C > 0$ and in Section 7, we argued that Example 1 requires $\Omega(n^2 \log(1/\delta)/\Delta^2)$ queries, we conjecture that any δ-correct algorithm $\mathcal{A}$ that solves the maxing (and thus ranking) problem for $\mathcal{I}_{WST}$ conducts $\Omega(n^2 \log(1/\delta)/\Delta^2)$ comparisons.