The Limits of Maxing, Ranking, and Preference Learning

Moein Falahatgar [1] **Ayush Jain** [1] **Alon Orlitsky** [1] **Venkatadheeraj Pichapati** [1] **Vaishakh Ravindrakumar** [1]

Abstract

We present a comprehensive understanding of three important problems in PAC preference learning: maximum selection (maxing), ranking, and estimating *all* pairwise preference probabilities, in the adaptive setting. With just Weak Stochastic Transitivity, we show that maxing requires $\Omega(n^2)$ comparisons and with slightly more restrictive Medium Stochastic Transitivity, we present a linear complexity maxing algorithm. With Strong Stochastic Transitivity and Stochastic Triangle Inequality, we derive a ranking algorithm with optimal $\mathcal{O}(n \log n)$ complexity and an optimal algorithm that estimates all pairwise preference probabilities.

1. Introduction

1.1. Background and motivation

Maximum selection (maxing) and sorting (ranking) are fundamental problems in Computer Science with numerous important applications. Deterministic versions of these problems are well studied.

In practical applications, comparisons are rarely deterministic. For example in soccer, when Real Madrid plays Barcelona the outcome is not always the same. Similarly, individual preferences in restaurants vary a lot. Other practical applications are in areas such as social choice (Caplin & Nalebuff, 1991; Soufiani et al., 2013), web search and information retrieval (Radlinski & Joachims, 2007; Radlinski et al., 2008), crowdsourcing (Chen et al., 2013; gif), recommender systems (Baltrunas et al., 2010) and several others.

These practical applications and the intrinsic theoretical interest, has led to significant work on the probabilistic version of maxing and ranking. Yet the most general model for which maxing can be done using near-linear comparisons is not known. We consider the most general transitive model that guarantees the existence of maximum and show that under this model any maxing algorithm requires quadratic many comparisons. We also consider a slightly more restrictive transitive model and propose a linear complexity maxing algorithm, making it the most general model known for which linear complexity maxing is possible. Also, for the most general known model with sub-quadratic complexity for ranking, we improve the complexity, making it orderwise optimal. We also propose an optimal algorithm that can simulate all pairwise comparisons.

1.2. Notation and problem formulation

Without loss of generality, let $[n] \stackrel{\text{def}}{=} \{1, 2, ..., n\}$ be the set of n elements. We consider probabilistic noisy comparisons i.e., whenever two elements i and j are compared, i is returned with an unknown probability $p_{i,j}$. There are no "ties" i.e., $p_{j,i} = 1 - p_{i,j}$. Let $\tilde{p}_{i,j} \stackrel{\text{def}}{=} p_{i,j} - \frac{1}{2}$ be the centered preference probability.

A maximal is an element i that is preferable to every other element i.e., $\tilde{p}_{i,j} \ge 0\ \forall j$. A ranking is a permutation $\sigma_1, \sigma_2, ..., \sigma_n$ of $[n]$ such that $\tilde{p}_{\sigma_i,\sigma_j} \ge 0$ whenever $i > j$.

But sometimes maximal and ranking might not even exist. For example, consider the popular Rock-Paper-Scissor game i.e., $p_{1,2} = p_{2,3} = p_{3,1} = 1$. Notice that under this model there is neither a maximal nor a ranking. Hence we need additional constraints on pairwise probabilities $p_{i,j}$.

Notice that for ranking to exist, there must exist an ordering ($\succ$) among elements s.t. whenever $i \succ j$, $\tilde{p}_{i,j} \ge 0$. The models that have such an ordering are said to satisfy *Weak Stochastic Transitivity (WST)*. Observe that WST is sufficient for existence of both maximal and ranking.

More restrictive notions of transitivity are motivated and used in different contexts. *Strong Stochastic Transitivity (SST)* which assumes that whenever $i \succ j \succ k$, $\tilde{p}_{i,k} \ge \max(\tilde{p}_{i,j}, \tilde{p}_{j,k})$, as its name suggests is a stronger notion of transitivity that confines the model more than WST, hence less general. *Medium Stochastic Transitivity (MST)* (Skorepa, 2010) sitting in between WST and SST, assumes that whenever $i \succ j \succ k$, $\tilde{p}_{i,k} \ge \min(\tilde{p}_{i,j}, \tilde{p}_{j,k})$. From WST to MST to SST, the model becomes more restrictive.

Another model restriction used in some of the previous

[1]University of California, San Diego. Correspondence to: Venkatadheeraj Pichapati <dheerajpv7@ucsd.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

works *Stochastic Triangle Inequality (STI)*, assumes that whenever $i \succ j \succ k$, $\tilde{p}_{i,k} \le \tilde{p}_{i,j} + \tilde{p}_{j,k}$. In this paper we propose maxing and ranking algorithms for models under various set of constraints.

There is also a concern with finding an exact maximal and ranking. Consider the case of $n = 2$ and $\tilde{p}_{1,2} \approx 0$. Notice that in this case where n is just 2, finding maximal and ranking could take arbitrarily many comparisons. Easy fix to alleviate this problem is to consider *Probably Approximately Correct (PAC)* formulation which we also adopt.

An element i is said to be ϵ-preferable to j if $\tilde{p}_{i,j} \ge -\epsilon$. For $\epsilon \in (0, 1/2)$, an ϵ-maximal is an element i that is ϵ-preferable to all elements i.e., $\tilde{p}_{i,j} \ge -\epsilon \; \forall j$. Given $0 < \epsilon < 1/2$, $0 < \delta \le 1/2$, a PAC maxing algorithm must output an ϵ-maximal with probability $\ge 1-\delta$. Similarly, an ϵ-ranking is a permutation $\sigma_1, \sigma_2, ..., \sigma_n$ of $[n]$ such that σ_i is ϵ-preferable to σ_j whenever $i > j$. Given $0 < \epsilon < 1/2$, $0 < \delta \le 1/2$, a PAC ranking algorithm must output an ϵ-ranking with probability $\ge 1 - \delta$.

1.3. Related work

Researchers initially considered more restrictive models. (Feige et al., 1994) considered constant noise model i.e., $\tilde{p}_{i,j} = \alpha > 0$ if $i \succ j$ and presented a maxing algorithm that uses $\mathcal{O}\left(\frac{n}{\alpha^2} \log \frac{1}{\delta}\right)$ comparisons and outputs maximal with probability $\ge 1 - \delta$. It also presented a ranking algorithm that uses $\mathcal{O}\left(\frac{n \log n}{\alpha^2}\right)$ comparisons and outputs ranking with probability $\ge 1 - 1/n$.

Another set of widely-studied restrictive models are parametric ones. (Szörényi et al., 2015) considered one of the most popular parametric models, Plackett-Luce (Plackett, 1975; Luce, 2005) and presented PAC maxing and ranking algorithms that use $\mathcal{O}\left(\frac{n}{\epsilon^2} \log \frac{n}{\epsilon\delta}\right)$ and $\mathcal{O}\left(\frac{n \log n}{\epsilon^2} \log \frac{n}{\epsilon\delta}\right)$ comparisons respectively.

Researchers also considered models that are more general than parametric models, yet still more restrictive than WST. (Yue & Joachims, 2011) considered models that satisfy both SST and STI and derived a PAC maxing algorithm that uses $\mathcal{O}\left(\frac{n}{\epsilon^2} \log \frac{n}{\epsilon\delta}\right)$ comparisons. Later (Falahatgar et al., 2017b) considered same model and proposed an optimal PAC maxing algorithm that uses $\mathcal{O}\left(\frac{n}{\epsilon^2} \log \frac{1}{\delta}\right)$ comparisons. It also proposed a PAC ranking algorithm that with probability $\ge 1 - 1/n$, outputs an ϵ-ranking using $\mathcal{O}\left(\frac{n \log n (\log \log n)^3}{\epsilon^2}\right)$ comparisons, $(\log \log n)^3$ times the known lower bound. Until now, it was not known if the additional $(\log \log n)^3$ factor is necessary for PAC ranking.

(Falahatgar et al., 2017a) considered models that satisfy only SST but not necessarily STI and proposed an optimal PAC maxing algorithm that uses $\mathcal{O}\left(\frac{n}{\epsilon^2} \log \frac{1}{\delta}\right)$ comparisons. They also showed that there exists a model which satisfies SST and yet no algorithm can find an ϵ-ranking for this model using $o(n^2)$ comparisons, establishing a lower bound of $\Omega(n^2)$ comparisons once STI property is dropped.

Among other related works we can point out (Busa-Fekete et al., 2014b; Lee et al., 2014; Dudík et al., 2015; Hüllermeier et al., 2008), who considered models more general than WST under different definitions of maximum and ranking. More discussion about these models can be found in Appendix G. (Busa-Fekete et al., 2014a; Mohajer et al., 2017) considered the non-PAC version and (Rajkumar & Agarwal, 2014; Negahban et al., 2012; 2016; Jang et al., 2016) considered the non-adaptive version of this problem. Also (Acharya et al., 2016; Ajtai et al., 2015) considered the deterministic adversarial version of maxing and ranking. (Shah et al., 2016b; Chatterjee et al., 2015; Shah et al., 2016a) studied the problem of estimating pairwise probabilities in non-adaptive setting.

2. New results and Outline

Maxing Linear-complexity maxing algorithm under SST by (Falahatgar et al., 2017a) encourages the search for a linear-complexity maxing algorithm for models with only WST properties. Two questions then arise: 1a) Is a linear complexity PAC maxing algorithm possible for models with only WST property? 1b) If not, does there exist a model more general than SST and less general than WST for which a linear complexity PAC maxing is possible?

We resolve both questions in this paper: 1a) No. Theorem 1 in Section 3 shows that there are WST models for which any PAC maxing algorithm requires $\Omega(n^2)$ comparisons. 1b) Yes. In Theorem 8 in Section 4, we derive a PAC maxing algorithm for MST model that uses $\mathcal{O}\left(\frac{n}{\epsilon^2} \log \frac{1}{\delta}\right)$ comparisons for $\delta \ge \min(1/n, e^{-n^{1/4}})$.

Ranking Motivated by the previous results of ranking under SST + STI, three questions arise: 2a) For models with SST + STI, is the additional $(\log \log n)^3$ factor necessary for PAC ranking algorithms? 2b) Since the near-linear complexity of ranking under SST + STI changes to quadratic complexity by dropping STI (Falahatgar et al., 2017a), is there a sub-quadratic algorithm for ranking under MST + STI? 2c) For models with SST + STI, since PAC ranking is possible with near linear complexity, is it also possible to approximate all pairwise probabilities to accuracy of ϵ using near linear number of comparisons?

We essentially resolve all three questions. 2a) No. In Theorem 9 in Section 5, we improve the PAC ranking algorithm for models with SST + STI removing additional $(\log \log n)^3$ factor and hence making it optimal. 2b) No. Theorem 10 in Section 6 shows that there is a model with MST+STI, for which any PAC ranking algorithm requires

Model	Maxing	Ranking	Finding $p_{i,j}$
SST with STI	$\Theta\left(\frac{n}{\epsilon^2}\log\frac{1}{\delta}\right)$ (Falahatgar et al., 2017b)	$\Theta\left(\frac{n\log n}{\epsilon^2}\right)^*$ Section 5	$\Theta\left(\frac{n\min(n,1/\epsilon)\log n}{\epsilon^2}\right)^*$ Section 7
SST	$\Theta\left(\frac{n}{\epsilon^2}\log\frac{1}{\delta}\right)$ (Falahatgar et al., 2017a)	$\Omega(n^2)$ (Falahatgar et al., 2017a)	$\Omega(n^2)$
MST with STI and MST	$\Theta\left(\frac{n}{\epsilon^2}\log\frac{1}{\delta}\right)^{**}$ Section 4	$\Omega(n^2)$ Section 6	$\Omega(n^2)$
WST with STI and WST	$\Omega(n^2)$ Section 3	$\Omega(n^2)$ Section 6	$\Omega(n^2)$

Table 1. Comprehensive results for maxing, ranking and finding $p_{i,j}$
$*$: for $\delta \ge \frac{1}{n}$, $**$: for $\delta \ge \min(1/n, e^{-n^{1/4}})$

$\Omega(n^2)$ comparisons. 2c) Yes. For models with SST + STI, in Theorems 11 and 12 in Sections 7, we present an optimal algorithm that uses $\mathcal{O}\left(\frac{n\min(n,1/\epsilon)\log n}{\epsilon^2}\right)$ comparisons and approximates all pairwise probabilities to accuracy of ϵ with probability $\ge 1-1/n$.

We present experiments over simulated data in Section 8 and end with our conclusions in Section 9.

Interpretation Table 1 summarizes all known results for problems of maxing, ranking, and finding pairwise probabilities under different transitive properties. Notice that under the most general model WST, all these problems require quadratic many comparisons and under the most restrictive model SST + STI, all problems have optimal algorithms with near-linear complexity. For MST and WST models adding STI property does not influence complexity for any problem. But for SST model adding STI property facilitates near-linear complexity algorithms for PAC ranking and approximating pairwise probabilities.

It is easy to see that once all pairwise probabilities are approximated to accuracy of $\epsilon/2$, one can find an ϵ-maximum and an ϵ-ranking. Hence approximating pairwise probabilities is harder than PAC ranking and lower bound for PAC ranking implies a lower bound for problem of approximating pairwise probabilities. Therefore in Table 1 lower bounds for finding p_{ij} follow from lower bounds for ranking. Further in Appendix B.1, under WST model, we present a trivial algorithm that with probability $\ge 1-\delta$, estimates all pairwise probabilities to accuracy of ϵ using $\mathcal{O}\left(\frac{n^2}{\epsilon^2}\log\frac{n}{\delta}\right)$ comparisons. Hence upper bound of $\mathcal{O}\left(\frac{n^2}{\epsilon^2}\log\frac{n}{\delta}\right)$ follows for all problems.

3. PAC maxing for WST

We show the lower bound of $\Omega(n^2)$ for maxing under WST by presenting an example for which any algorithm requires $\Omega(n^2)$ comparisons to output a $1/4$-maximum for $\delta \le 1/8$.

To establish the lower bound, we reduce the problem of finding a $1/4$-maximum to finding the left most piece of a linear jigsaw puzzle. We consider the following model with n elements $S = \{a_1, a_2, \ldots, a_n\}$: $\tilde{p}_{a_i,a_{i+1}} = \frac{1}{2}\forall i < n$, and $\tilde{p}_{a_i,a_j} = \mu(0 < \mu < 1/n^{10}), \forall j > i+1$. This model satisfies WST since there exists an underlying order $\succ$, $a_i \succ a_j$ if $i < j$ (because $\tilde{p}_{a_i,a_j} > 0$) and a_1 is the only $1/4$-maximum under this model.

Observe that a_i is always preferred to a_{i+1}, but for every non consecutive pair, comparison output is almost a fair coin flip. We make the problem simpler by giving the extra information of whether two non consecutive elements are being compared. Notice that this only makes the problem easier, namely, complexity for modified problem is smaller than that of original problem.

The modified problem is similar to a linear jigsaw puzzle where if we compare two pieces we will know if pieces are adjacent or not and if adjacent, which piece is on the left, the goal is to find the left most piece. We show that w.h.p., any algorithm neither finds more than $n/32$ connections (a set of neighbors) nor asks $\Omega(n)$ comparisons for the left most piece. We use this to show the lower bound. The proof is in Appendix A.

Theorem 1. *There exists a model that satisfies WST for which any algorithm requires $\Omega(n^2)$ comparisons to find a $1/4$-maximum with probability $\ge 7/8$.*

4. PAC maxing for MST

Outline In this section, we propose OPT-MAX, a linear complexity maxing algorithm for MST. In the process, we present two other suboptimal maxing algorithms SOFT-SEQ-ELIM, NEAR-OPT-MAX and use them as building blocks in OPT-MAX. SOFT-SEQ-ELIM finds an ϵ-maximum with quadratic complexity. Its performance depends on the starting element (anchor). NEAR-OPT-MAX first finds a good anchor and then uses SOFT-SEQ-ELIM, guaranteeing near linear comparison complexity. OPT-MAX builds on NEAR-OPT-MAX and finds an ϵ-maximum in linear-complexity for $\delta \geq \min(1/n, e^{-n^{1/4}})$.

4.1. SOFT-SEQ-ELIM

Before presenting SOFT-SEQ-ELIM, we first present the subroutine COMPARE we use to compare two elements.

COMPARE COMPARE takes 5 parameters : two elements i, j that need to be compared, lower bias ϵ_l, upper bias ϵ_u, confidence δ and deems if $\tilde{p}_{i,j} < \epsilon_l$ or $\tilde{p}_{i,j} > \epsilon_u$. It compares i and j for $\frac{8}{(\epsilon_u-\epsilon_l)^2}\log\frac{2}{\delta}$ times. Let $\hat{p}_{i,j}$ be the fraction of times i won and $\hat{\tilde{p}}_{i,j} = \hat{p}_{i,j} - 1/2$. If $\hat{\tilde{p}}_{i,j} < \frac{3\epsilon_l}{4} + \frac{\epsilon_u}{4}$, then COMPARE deems $\tilde{p}_{i,j} < \epsilon_l$ (returns 1), if $\hat{\tilde{p}}_{i,j} > \frac{\epsilon_l}{4} + \frac{3\epsilon_u}{4}$, then COMPARE deems $\tilde{p}_{i,j} > \epsilon_u$ (returns 3) and for other ranges of $\hat{\tilde{p}}_{i,j}$, COMPARE not able to take a decision, returns 2.

Lemma 2 bounds comparisons used by COMPARE and proves its correctness. COMPARE and its analysis is presented in Appendix C.2.

Lemma 2. *For $\epsilon_u > \epsilon_l$, COMPARE$(i, j, \epsilon_l, \epsilon_u, \delta)$ uses $\leq \frac{8}{(\epsilon_u-\epsilon_l)^2}\log\frac{2}{\delta}$ comparisons and if $\tilde{p}_{i,j} < \epsilon_l$, then w.p.$\geq 1-\delta$, it returns 1, else if $\tilde{p}_{i,j} > \epsilon_u$, w.p.$\geq 1-\delta$, it returns 3. Further if $\tilde{p}_{i,j} \leq (\epsilon_l+\epsilon_u)/2$, w.p.$\geq 1-\delta$, it does not return 3 and if $\tilde{p}_{i,j} > (\epsilon_l+\epsilon_u)/2$, w.p.$\geq 1-\delta$, it does not return 1.*

SOFT-SEQ-ELIM SOFT-SEQ-ELIM takes 5 parameters: input set S, starting anchor element r, lower bias ϵ_l, upper bias ϵ_u and confidence δ. SOFT-SEQ-ELIM happens in rounds. In each round, it compares the current anchor a with remaining elements one by one using COMPARE. Due to probabilistic nature, we cannot exactly compare if $\tilde{p}_{e,a} > \epsilon_u$ vs $\tilde{p}_{e,a} \leq \epsilon_u$. Hence we compare if $\tilde{p}_{e,a} > \epsilon_u$ vs $\tilde{p}_{e,a} < \epsilon_l$. For an element e, if COMPARE deems $\tilde{p}_{e,a} < \epsilon_l$, then SOFT-SEQ-ELIM eliminates that element and if COMPARE deems $\tilde{p}_{e,a} > \epsilon_u$, then SOFT-SEQ-ELIM updates current anchor element to e and eliminates a. This process is continued until the current anchor element is not updated after comparing with all remaining elements and then SOFT-SEQ-ELIM outputs final anchor element.

If $\tilde{p}_{e,a} < \epsilon_l$ or $\tilde{p}_{e,a} > \epsilon_u$, COMPARE deems correctly. If $\epsilon_l \leq \tilde{p}_{e,a} \leq \epsilon_u$, then COMPARE can sometimes fail to output any decision and in that case, SOFT-SEQ-ELIM neither eliminates that element nor updates the anchor element, it just moves to next remaining element in S.

Theoretically, performance of SOFT-SEQ-ELIM strongly depends on the starting anchor element r. To define a good anchor element, similar to (Falahatgar et al., 2017a), an element a is called an (ϵ, m)-*good anchor* if a is not ϵ-preferable to at most m elements, i.e., $|\{e : e \in S \text{ and } \tilde{p}_{e,a} > \epsilon\}| \leq m$. We show that every element for which initial anchor r is ϵ_l-preferable is deemed bad and gets eliminated after its first comparison round and hence comparisons spent on all such elements is $\mathcal{O}(|S|)$. Since initial anchor r is an (ϵ_l, m)-*good anchor* element, there are only m elements for which r is not ϵ_l-preferable. We later show that only these elements can become anchors, leading to at most m changes of anchors. Therefore each such element gets compared in at most m rounds and hence we can bound total comparison rounds by $\mathcal{O}(|S| + m^2)$. Lemma 3 bounds comparisons used by SOFT-SEQ-ELIM and proves its correctness. Proof is in Appendix C.3.

Algorithm 1 SOFT-SEQ-ELIM

```
1: inputs
2:   Set $S$, element $r$, lower bias $\epsilon_l$, upper bias $\epsilon_u$, confidence $\delta$
3: $Q = S \setminus \{r\}$
4: while $Q \neq \emptyset$ do
5:   $r' = r$, $Q' = \emptyset$
6:   for $c \in Q$ do
7:     $k = $ COMPARE$(c, r, \epsilon_l, \epsilon_u, \frac{2\delta}{|S|^2})$
8:     if $k == 1$ then
9:       $Q' = Q' \bigcup \{c\}$.
10:    else if $k == 3$ then
11:      $r \leftarrow c$
12:      $Q' = Q' \bigcup \{c\}$
13:      break
14:    end if
15:  end for
16:  if $r == r'$ then
17:    break
18:  end if
19:  $Q = Q \setminus Q'$
20: end while
21: return r
```

Lemma 3. *If r is an (ϵ_l, m)-good anchor element, w.p.$\geq 1-\delta$, SOFT-SEQ-ELIM$(S, r, \epsilon_l, \epsilon_u, \delta)$ uses $\mathcal{O}\left(\frac{|S|+m^2}{(\epsilon_u-\epsilon_l)^2}\log\frac{|S|}{\delta}\right)$ comparisons and outputs $\hat{r}$, an ϵ_u maximum of S, such that either $\hat{r} = r$ or $\tilde{p}_{\hat{r},r} > \frac{\epsilon_l+\epsilon_u}{2}$.*

Corollary 4 bounds comparisons used by SOFT-SEQ-ELIM for any starting anchor. Proof follows from Lemma 3

Corollary 4. *For any r, w.p.$\geq 1-\delta$, SOFT-SEQ-ELIM$(S, r, \epsilon_l, \epsilon_u, \delta)$ uses $\mathcal{O}\left(\frac{|S|^2}{(\epsilon_u-\epsilon_l)^2}\log\frac{|S|}{\delta}\right)$ comparisons and outputs $\hat{r}$, an ϵ_u maximum of S, such that either $\hat{r} = r$ or $\tilde{p}_{\hat{r},r} > \frac{\epsilon_l+\epsilon_u}{2}$.*

Now we build on SOFT-SEQ-ELIM and propose a near linear algorithm NEAR-OPT-MAX.

4.2. NEAR-OPT-MAX

NEAR-OPT-MAX(S, ϵ, δ) w.p.$\geq 1-\delta$, uses $\mathcal{O}\left(\frac{|S|}{\epsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right)$ comparisons and outputs an ϵ-maximum of S.

Since complexity of SOFT-SEQ-ELIM depends on the initial anchor element, if we can pick a good initial anchor element, then we can reduce the number of comparisons. One way to pick a good initial anchor element is to find an $\epsilon/2$-maximum of a randomly picked subset.

Lemma 5 shows that an ϵ-maximum of a randomly picked subset is a good anchor element. Proof in Appendix C.4.

Lemma 5. *If r is an ϵ-maximum of a set Q, formed by picking m elements randomly from S, then w.p.$\geq 1-\delta$, r is an $\left(\epsilon, \frac{|S|}{m}\log\frac{|S|}{\delta}\right)$-good anchor element of S.*

NEAR-OPT-MAX(S, ϵ, δ) first picks a random subset Q of size $\sqrt{|S|\log\frac{4|S|}{\delta}}$ and uses SOFT-SEQ-ELIM to find an $\epsilon/2$-maximum of Q.

By Lemma 5, w.p.$\geq 1-\delta/4$, an $\epsilon/2$-maximum of Q will be an $(\epsilon/2, \sqrt{|S|\log\frac{4|S|}{\delta}})$-*good anchor* element. NEAR-OPT-MAX then uses SOFT-SEQ-ELIM with $\epsilon/2$-maximum of Q as initial anchor to find an ϵ-maximum of S. Since the initial anchor is provably good, we are able to bound the comparisons.

Algorithm 2 NEAR-OPT-MAX

```
1: inputs
2:     Set $S$, bias $\epsilon$, confidence $\delta$
3: Form a set $Q$ by selecting $\sqrt{|S|\log\frac{4|S|}{\delta}}$ random elements from $S$ without replacement.
4: $a \leftarrow$ random element from $Q$, $Q = Q \setminus \{a\}$
5: $r \leftarrow$ SOFT-SEQ-ELIM$\left(Q, a, 0, \frac{\epsilon}{2}, \frac{\delta}{4}\right)$, $S = S \setminus \{r\}$
6: return SOFT-SEQ-ELIM$(S, r, \epsilon/2, \epsilon, \delta/2)$
```

Lemma 6 bounds the comparisons used by NEAR-OPT-MAX and proves its correctness.

Lemma 6. *With probability $\geq 1-\delta$, NEAR-OPT-MAX(S, ϵ, δ) uses $\mathcal{O}\left(\frac{|S|}{\epsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right)$ comparisons and outputs an ϵ-maximum of S.*

We build on NEAR-OPT-MAX and derive an optimal algorithm for $\delta \geq \min(1/|S|, e^{-|S|^{1/4}})$.

4.3. Optimal linear Algorithm

We first present an algorithm that is optimal for low ranges of δ i.e., $\min(e^{-|S|^{1/4}}, 1/|S|) \leq \delta \leq \frac{1}{|S|^{1/3}}$.

4.3.1. LOW RANGES OF δ

We first find a good anchor, this time using NEAR-OPT-MAX and then use SOFT-SEQ-ELIM with NEAR-OPT-MAX output as initial anchor.

OPT-MAX-LOW picks a random subset of size $|S|^{3/4}$ and finds an $\epsilon/2$-maximum of this set using NEAR-OPT-MAX. We later show that output is an $(\epsilon/2, \mathcal{O}(\sqrt{|S|}))$-*good anchor* element of S. OPT-MAX-LOW then uses SOFT-SEQ-ELIM with the previous output as initial anchor to find an ϵ-maximum of S. Since initial anchor is good, we are able to bound comparisons used by OPT-MAX-LOW.

Observe that in OPT-MAX-LOW, we call SOFT-SEQ-ELIM three times in total: two times during NEAR-OPT-MAX and once to produce the final output. Each successive call of SOFT-SEQ-ELIM acts on higher size, namely first we find $\epsilon/4$-maximum in a small set and using this element as anchor, then we find $\epsilon/2$-maximum in a larger set and finally using this new element as anchor, we find an ϵ-maximum of the whole set S.

Algorithm 3 OPT-MAX-LOW

```
1: inputs
2:     Set $S$, bias $\epsilon$, confidence $\delta$
3: Form a set $Q$ by selecting $|S|^{3/4}$ random elements from $S$ without replacement
4: $r \leftarrow$ NEAR-OPT-MAX$(Q, \frac{\epsilon}{2}, \frac{\delta}{3})$
5: return SOFT-SEQ-ELIM$(S, r, \frac{\epsilon}{2}, \epsilon, \frac{\delta}{3})$
```

Lemma 7 bounds comparisons used by OPT-MAX-LOW and proves its correctness. Proof is in Appendix C.6.

Lemma 7. *For $\frac{1}{|S|^{1/3}} \geq \delta \geq \min(1/|S|, e^{-|S|^{1/4}})$, w.p.$\geq 1-\delta$, OPT-MAX-LOW$(S, \epsilon, \delta)$ uses $\mathcal{O}(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta})$ comparisons and outputs r, an ϵ-maximum*

4.3.2. HIGHER RANGES OF CONFIDENCE δ

For low ranges of confidence δ $\left(\delta \leq \frac{1}{|S|^{1/3}}\right)$, notice that $\log\frac{1}{\delta}$ and $\log\frac{|S|}{\delta}$ are of same order and hence if we use SOFT-SEQ-ELIM with a good anchor, we can guarantee complexity of $\mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{|S|}{\delta}\right) = \mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$.

However, for high values of δ, this is not the case. We solve this problem by pruning S to a smaller set of size $|S|/\log|S|$ such that it contains all good elements and then use SOFT-SEQ-ELIM. Due to space constraint, we present PRUNE, the pruning algorithm, OPT-MAX-MEDIUM, and OPT-MAX-HIGH, linear complexity maxing algorithms for higher ranges of confidence in Appendix C.8.

4.4. Full Algorithm

In Theorem 8 we bound comparisons used by OPT-MAX and prove its correctness. Proof follows from Lemmas 7

Algorithm 4 OPT-MAX

```
inputs
  Set $S$, bias $\epsilon$, confidence $\delta$
if $\delta \leq \frac{1}{|S|^{1/3}}$ then
  return OPT-MAX-LOW$(S, \epsilon, \delta)$
end if
if $\delta \leq \frac{1}{\log |S|}$ then
  return OPT-MAX-MEDIUM$(S, \epsilon, \delta)$
end if
return OPT-MAX-HIGH$(S, \epsilon, \delta)$
```

and corresponding Lemmas 19 and 20 for OPT-MAX-MEDIUM and OPT-MAX-HIGH given in Appendix C.8.

Theorem 8. *For* $\delta \geq \min(1/|S|, e^{-|S|^{1/4}})$, *w.p.*$\geq 1 - \delta$, OPT-MAX$(S, \epsilon, \delta)$ *uses* $\mathcal{O}\left(\frac{|S|}{\epsilon^2} \log \frac{1}{\delta}\right)$ *comparisons and outputs an* ϵ*-maximum of* S.

5. Ranking for SST+STI

(Falahatgar et al., 2017b) provides a ranking algorithm that w.p.$\geq 1 - 1/|S|$, uses $\mathcal{O}\left(\frac{|S|}{\epsilon^2} \log |S| (\log \log |S|)^3\right)$ comparisons and outputs an ϵ-ranking of input set S.

We build on their algorithm BINARY-SEARCH-RANKING, improving two components which lead to additional $(\log \log |S|)^3$ factor, thereby proposing an optimal ϵ-ranking algorithm that uses $\mathcal{O}\left(\frac{|S|}{\epsilon^2} \log |S|\right)$ comparisons.

In Appendix 5, we outline the algorithm proposed in (Falahatgar et al., 2017b), pointing out the two components that lead to additional factor, and present ideas that improve over these components. For detailed explanation of BINARY-SEARCH-RANKING we refer readers to (Falahatgar et al., 2017b). Now we explain the high-level idea of how we improve over these components.

The two components that we improve upon share the property that each is being called for $\Omega(|S|/(\log |S|)^3)$ times and at each time finds a correct output w.p.$\geq 1 - 1/|S|^5$.

Instead of finding a correct output w.p.$\geq 1 - 1/|S|^5$ in one shot, and incurring high complexity, we propose the following. First use the component to find a correct output w.p.$\geq 1 - 1/\log |S|$, then check if the output is correct or not. If the output is deemed to be not correct, run the component again, finding a correct output w.p.$\geq 1 - 1/|S|^6$.

Thus to show the potency of this idea, it suffices to show: One, the second run is only invoked a few times and two, the complexity of checking whether an output is correct is not high. Our main contribution is RANK-CHECK algorithm that checks if an ordered set is ϵ-ranked or not 3ϵ-ranked. We present RANK-CHECK in Appendix D.3

Theorem 9. BINARY-SEARCH-RANKING(S, ϵ) *(Falahatgar et al., 2017b) with new improved components presented here, w.p.*$\geq 1 - 1/|S|$, *uses* $\mathcal{O}\left(\frac{|S| \log |S|}{\epsilon^2}\right)$ *comparisons and outputs an* ϵ*-ranking of* S.

6. Lower bound for ranking for MST+STI

In this section we show that there exists a model with both MST and STI properties under which any PAC ranking algorithm requires quadratic many comparisons. Consider the model $S = \{a_1, a_2, ..., a_n\}$ s.t. a_1 is preferable to a_2 i.e., $\tilde{p}_{a_1,a_2} = 1/2$ and comparison between any other pair is almost a fair coin flip i.e., $\tilde{p}_{a_i,a_j} = \mu \; \forall i < j$ and $\{i, j\} \neq \{1, 2\}$ for some $\mu < 1/n^{10}$. This model satisfies both MST and STI. Any permutation which has a_1 coming after a_2 is a $1/4$-ranking. But since comparison between any pair other than (a_1, a_2) is essentially a fair coin toss, any strategy that does not compare a_1 and a_2 will not have them in correct order in the output w.p.$\approx 1/2$ and hence won't be a $1/4$-ranking. Therefore this problem is similar to finding a single biased coin among $\binom{n}{2}$ coins which needs $\Omega(n^2)$ comparisons.

Theorem 10 bounds the complexity required for ϵ-ranking of models with MST and STI. Proof is in Appendix E.

Theorem 10. *There exists a model with MST and STI properties for which any algorithm requires* $\Omega(n^2)$ *comparisons to output a* $1/4$*-ranking w.p.*$\geq 7/8$.

7. Finding pairwise probabilities for SST+STI

Theorem 9 shows that for a model satisfying both SST and STI, an ϵ-ranking can be found using $\mathcal{O}\left(\frac{|S| \log |S|}{\epsilon^2}\right)$ comparisons. In this section we answer the question whether under same model we can approximate all pairwise probabilities to accuracy of ϵ using almost same complexity.

We first show a lower bound of $\Omega\left(\frac{|S| \min(|S|, 1/\epsilon)}{\epsilon^2} \log |S|\right)$ utilizing a model for which $\Omega(|S| \min(|S|, 1/\epsilon))$ pairwise probabilities need to be approximated using comparisons. Later we present APPROX-PROB that uses comparisons only for $\mathcal{O}(|S| \min(|S|, 1/\epsilon))$ pairs and hence obtain orderwise same upper bound as lower bound.

7.1. Lower Bound

We show that any algorithm requires $\Omega\left(\frac{|S| \min(|S|, 1/\epsilon) \log |S|}{\epsilon^2}\right)$ comparisons to approximate all pairwise probabilities to ϵ accuracy.

We prove the lower bound by using the model: $(4k+4)\epsilon \leq \tilde{p}_{a_{i+k},a_i} \leq (4k+8)\epsilon$ for $1 \leq k \leq \min(n - i, \lfloor \frac{1}{16\epsilon} - 2 \rfloor)$ and $\tilde{p}_{a_{i+k},a_i} = 1/4$ for $k > \min(n - i, \lfloor \frac{1}{16\epsilon} - 2 \rfloor)$.

It can be shown that this model satisfies both SST and STI.

Under this model, the only way to approximate unfixed pairwise probabilities is by comparing those pairs. Since pairwise probabilities are not fixed for $\Omega(n \min(n, 1/\epsilon))$ pairs, any algorithm needs to approximate those many probabilities to accuracy of ϵ, hence the lower bound.

Theorem 11 bounds the required complexity to approximate all pairwise probabilities. Proof is in Appendix F.1

Theorem 11. *For $\epsilon < 1/48$, there exists a model that satisfies both SST and STI for which any algorithm requires $\Omega\left(\frac{|S|\min(|S|,1/\epsilon)}{\epsilon^2}\log|S|\right)$ comparisons to approximate all pairwise probabilities to ϵ accuracy w.p. $\geq 3/4$.*

7.2. Upper Bound

Here we propose an algorithm to approximate all pairwise probabilities to an accuracy of ϵ.

The proposed algorithm, first finds an $\epsilon/8$-ranking of the input set S and then approximates pairwise probabilities. By Theorem 9, w.p.$\geq 1 - \frac{1}{|S|^2}$ we can find an $\epsilon/8$-ranking of the input set S using $\mathcal{O}\left(\frac{|S|\log|S|}{\epsilon^2}\right)$ comparisons. We present APPROX-PROB that given an $\epsilon/8$-ranked set, approximates all pairwise probabilities to an accuracy of ϵ.

APPROX-PROB APPROX-PROB takes an $\epsilon/8$-ranked ordered set S i.e., $\tilde{p}_{S(i),S(j)} \leq \epsilon/8\ \forall i < j$ and bias ϵ and approximates all pairwise probabilities to an accuracy of ϵ.

Note that it is enough to approximate $\tilde{p}_{S(j),S(i)}$ for $j \geq i$ since $\tilde{p}_{S(i),S(j)} = -\tilde{p}_{S(j),S(i)}$. For all $i > 1$, APPROX-PROB compares $S(i)$ and $S(1)$, $\frac{16\log|S|^4}{\epsilon^2}$ times and approximates $\tilde{p}_{S(i),S(1)}$ by $\hat{\tilde{p}}_{S(i),S(1)}$, the fraction of times $S(i)$ won rounded off to the nearest multiple of ϵ. Since for perfectly ranked ordered set $\tilde{p}_{S(i+1),S(1)} \geq \tilde{p}_{S(i),S(1)}$, if $\hat{\tilde{p}}_{S(i+1),S(1)} < \hat{\tilde{p}}_{S(i),S(1)}$, then APPROX-PROB corrects $\hat{\tilde{p}}_{S(i+1),S(1)}$, setting it equal to $\hat{\tilde{p}}_{S(i),S(1)}$. It can be shown that $\tilde{p}_{S(i),S(1)}$ is approximated to an accuracy of $\frac{7\epsilon}{8}$.

APPROX-PROB continues this process by approximating $\tilde{p}_{S(i),S(2)}$ for $i \geq 2$ by increasing i one at a time. For a perfectly ranked set, $\tilde{p}_{S(i-1),S(2)} \leq \tilde{p}_{S(i),S(2)} \leq \tilde{p}_{S(i),S(1)}$ and hence if $\hat{\tilde{p}}_{S(i-1),S(2)} = \hat{\tilde{p}}_{S(i),S(1)}$, APPROX-PROB does not use comparisons to approximate $\tilde{p}_{S(i),S(2)}$, instead assigns $\hat{\tilde{p}}_{S(i),S(2)} = \hat{\tilde{p}}_{S(i-1),S(2)}$. Whenever $\hat{\tilde{p}}_{S(i-1),S(2)} \neq \tilde{p}_{S(i),S(1)}$, APPROX-PROB approximates $\tilde{p}_{S(i),S(2)}$ by comparing $S(i)$ and $S(2)$. It can be shown that $\tilde{p}_{S(i),S(2)}$ is approximated to accuracy of ϵ.

APPROX-PROB continues this process for $S(3)$, then $S(4)$ and so on until $S(n)$. Notice that whenever $\hat{\tilde{p}}_{S(i-1),S(j)} = \hat{\tilde{p}}_{S(i),S(j-1)}$, APPROX-PROB does not use comparisons to approximate $\tilde{p}_{S(i),S(j)}$ but simply assigns $\hat{\tilde{p}}_{S(i),S(j)} = \hat{\tilde{p}}_{S(i-1),S(j)}$. We show this in fact happens at many places and only $\mathcal{O}(|S|\min(|S|, 1/\epsilon))$ pairwise probabilities are approximated using comparisons. This enables obtaining orderwise same upper bound as the lower bound.

Algorithm 5 APPROX-PROB
```
1:  inputs
2:      Ordered Set $S$, bias $\epsilon$
3:  $\hat{\tilde{p}}_{S(1),S(1)} = 0$
4:  for $i$ from 2 to $|S|$ do
5:      Compare $S(1)$ and $S(i)$ for $\frac{16}{\epsilon^2}\log|S|^4$ times
6:      $\hat{\tilde{p}}_{S(i),S(1)} = \left[\frac{\text{fraction of times } S(i) \text{ won}}{\epsilon} - \frac{1}{2}\right]\epsilon$
7:      if $\hat{\tilde{p}}_{S(i),S(1)} < \hat{\tilde{p}}_{S(i-1),S(1)}$ then
8:          $\hat{\tilde{p}}_{S(i),S(1)} = \hat{\tilde{p}}_{S(i-1),S(1)}$
9:      end if
10: end for
11: for $j$ from 2 to $|S|$ do
12:     $\hat{\tilde{p}}_{S(j),S(j)} = 0$
13:     for $k$ from $j+1$ to $|S|$ do
14:         if $\hat{\tilde{p}}_{S(k-1),S(j)} = \hat{\tilde{p}}_{S(k),S(j-1)}$ then
15:             $\hat{\tilde{p}}_{S(k),S(j)} = \hat{\tilde{p}}_{S(k-1),S(j)}$
16:         else
17:             Compare $S(j)$ and $S(k)$ for $\frac{16}{\epsilon^2}\log|S|^4$ times
18:             $\hat{\tilde{p}}_{S(k),S(j)} = \left[\frac{\text{fraction of times } S(k) \text{ won}}{\epsilon} - \frac{1}{2}\right]\epsilon$
19:         end if
20:     end for
21: end for
```

Theorem 12 shows the correctness of APPROX-PROB and bounds its comparisons. Proof is in Appendix F.3

Theorem 12. *Given an $\epsilon/8$-ranked ordered set S i.e., $\tilde{p}_{S(i),S(j)} \leq \epsilon/8\ \forall i < j$,* APPROX-PROB$(S, \epsilon)$ *uses $\mathcal{O}(\frac{|S|\min(|S|,1/\epsilon)}{\epsilon^2}\log|S|)$ comparisons and w.p.$\geq 1 - \frac{1}{|S|^2}$ approximates all pairwise probabilities to accuracy of ϵ.*

8. Experiments

In this section, we compare the performance of our maxing algorithms with previous work on synthetic data. All results presented here are averaged over 1000 runs.

We compare our maxing algorithms SOFT-SEQ-ELIM, NEAR-OPT-MAX, and OPT-MAX with **SEQ-ELIMINATE** (Falahatgar et al., 2017a), **KNOCK-OUT** (Falahatgar et al., 2017b), **MallowsMPI** (Busa-Fekete et al., 2014a), **AR** (Heckel et al., 2016) and **BTM-PAC** (Yue & Joachims, 2011). **KNOCKOUT** and **BTM-PAC** are PAC maxing algorithms for models with both SST and STI properties. **SEQ-ELIMINATE** is a PAC maxing algorithm for SST model. **MallowsMPI**, originally designed for Mallows model, finds a condorcet winner which exists under WST. **AR** is a maxing algorithm that finds Borda winner that is same as condorcet winner

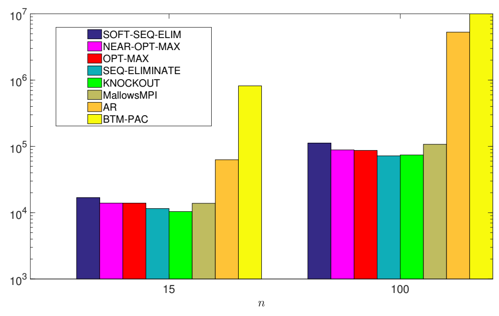

Figure 1. Maxing Algorithms for model with SST and STI

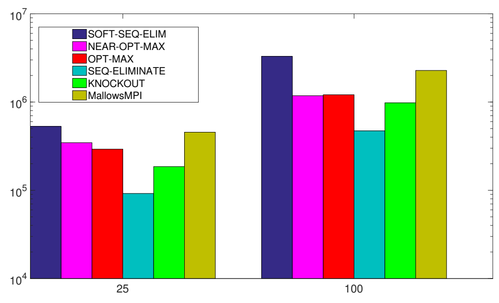

Figure 2. Maxing Algorithms for model with MST but not SST

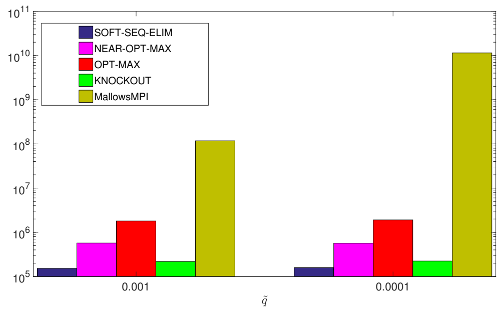

Figure 3. Maxing algorithms for model without STI

under WST. In all experiments, we use maxing algorithms to find a 0.05-maximum with $\delta = 0.1$.

We first consider the model $p_{i,j} = 0.6 \; \forall i < j$ same as in (Yue & Joachims, 2011; Falahatgar et al., 2017b;a) that satisfies both SST and STI properties. Note that $i = 1$ is the only 0.05-maximum under this model. Figure 1 presents number of comparisons used by each maxing algorithm. Observe that compared to other algorithms, **BTM-PAC** uses too many comparisons even for $n = 15$. The reason might be **BTM-PAC** is mainly intended for reducing regret in the conventional bandits setting. The bar for **BTM-PAC** complexity for $n = 100$ is not fully shown in the figure to better scale the other complexity bars. Comparison complexity of AR is high for $n = 100$ mainly because **AR** eliminates elements based on Borda scores and Borda scores are very close to each other for large n. We drop **BTM-PAC** and **AR** henceforth.

Now we consider a model that satisfies MST but not SST, i.e., $p_{5i+l,5i+k} = 0.6 \; \forall i < n/5 - 1, \; 1 \le l < k \le 5$ and $p_{5i+l,5j+k} = 0.52 \; \forall i < j < n/5 - 1, \; 0 < l, k \le 5$. Notice that under this model elements are divided into groups of five where within each group $|\tilde{p}_{i,j}| = 0.1$ and for elements in two different groups $|\tilde{p}_{i,j}| = 0.02$, hence there is a 0.05-maximum in each group. Figure 2 demonstrates comparison complexity of algorithms under this model. **SEQ-ELIMINATE** uses fewer comparisons, but it fails to output a 0.05-maximum with probability 0.21 for $n = 25$ and 0.19 for $n = 100$. Hence **SEQ-ELIMINATE** fails once SST is not satisfied. This is because when you compare a 0.05-maximum of a group with an element in other group, 0.05-maximum can get eliminated with probability ≈ 0.5. Hence with lots of groups **SEQ-ELIMINATE** fails. Other algorithms find a 0.05-maximum in all runs. We drop **SEQ-ELIMINATE** henceforth.

Now we consider a model that does not satisfy STI but satisfies MST i.e., $n = 10$ and $p_{1,j} = 1/2 + \tilde{q} \; \forall j \le n/2$, $p_{1,j} = 1 \; \forall j > n/2$ and $p_{i,j} = 1/2 + \tilde{q} \; \forall 1 < i < j$, $\tilde{q} < 0.05$. Under this model any $i \le 5$ is a 0.05-maximum. Figure 3 shows the average comparison complexity of algorithms under this model. **KNOCKOUT** uses fewer comparisons, but fails to output a 0.05-maximum with probability 0.12 for $\tilde{q} = 0.001$ and 0.25 for $\tilde{q} = 0.0001$, hence fails to meet the confidence requirement once STI is dropped. Other algorithms find a 0.05-maximum in all runs.

It is interesting to note that **MallowsMPI** uses more comparisons as $\tilde{q}$ decreases, whereas the complexity of other algorithms remains almost same. This is because **MallowsMPI** tries to find absolute maximum which is not always practical. Further note that the performance of SOFT-SEQ-ELIM is better than NEAR-OPT-MAX, and NEAR-OPT-MAX is better than OPT-MAX. This is because the bias gap for SOFT-SEQ-ELIM, NEAR-OPT-MAX and OPT-MAX is ϵ, $\epsilon/2$ and $\epsilon/4$ respectively, resulting in higher constants for NEAR-OPT-MAX and OPT-MAX. While the theoretical order complexity is higher for SOFT-SEQ-ELIM, in practice it can find a good anchor quickly and seems to have near-linear order complexity.

9. Conclusion

We studied the problem of maxing, ranking, and estimating comparison probabilities under different stochastic transitivity constraints. We showed that under WST, maxing needs quadratic comparisons. We also presented a linear-complexity algorithm for maxing under MST. We also proposed an optimal ranking algorithm for SST models with Stochastic Triangle Inequality, closing $(\log \log n)^3$ gap. For the same model, we proposed an optimal algorithm for estimating the comparison probabilities.

ACKNOWLEDGMENTS

We thank NSF for supporting this work through grants CIF-1564355 and CIF-1619448.

References

http://www.gif.gf/.

Acharya, J., Falahatgar, M., Jafarpour, A., Orlitsky, A., and Suresh, A. T. Maximum selection and sorting with adversarial comparators and an application to density estimation. *arXiv preprint arXiv:1606.02786*, 2016.

Ajtai, M., Feldman, V., Hassidim, A., and Nelson, J. Sorting and selection with imprecise comparisons. *ACM Transactions on Algorithms (TALG)*, 12(2):19, 2015.

Baltrunas, L., Makcinskas, T., and Ricci, F. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pp. 119–126. ACM, 2010.

Busa-Fekete, R., Hüllermeier, E., and Szörényi, B. Preference-based rank elicitation using statistical models: The case of mallows. In *Proc. of the ICML*, pp. 1071–1079, 2014a.

Busa-Fekete, R., Szörényi, B., and Hüllermeier, E. Pac rank elicitation through adaptive sampling of stochastic pairwise preferences. In *AAAI*, 2014b.

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

Chatterjee, S. et al. Matrix estimation by universal singular value thresholding. *The Annals of Statistics*, 43(1):177–214, 2015.

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

Dudík, M., Hofmann, K., Schapire, R. E., Slivkins, A., and Zoghi, M. Contextual dueling bandits. *arXiv preprint arXiv:1502.06362*, 2015.

Falahatgar, M., Hao, Y., Orlitsky, A., Pichapati, V., and Ravindrakumar, V. Maxing and ranking with few assumptions. In *Advances in Neural Information Processing Systems*, pp. 7063–7073, 2017a.

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

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

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

Hüllermeier, E., Fürnkranz, J., Cheng, W., and Brinker, K. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916, 2008.

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

Lee, D. T., Goel, A., Aitamurto, T., and Landemore, H. Crowdsourcing for participatory democracies: Efficient elicitation of social choice functions. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

Luce, R. D. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2005.

Mohajer, S., Suh, C., and Elmahdy, A. Active learning for top-k rank aggregation from noisy comparisons. In *International Conference on Machine Learning*, pp. 2488–2497, 2017.

Negahban, S., Oh, S., and Shah, D. Iterative ranking from pair-wise comparisons. In *NIPS*, pp. 2474–2482, 2012.

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

Plackett, R. L. The analysis of permutations. *Applied Statistics*, pp. 193–202, 1975.

Radlinski, F. and Joachims, T. Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD*, pp. 570–579. ACM, 2007.

Radlinski, F., Kurup, M., and Joachims, T. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM conference on Information and knowledge management*, pp. 43–52. ACM, 2008.

Rajkumar, A. and Agarwal, S. A statistical convergence perspective of algorithms for rank aggregation from pairwise data. In *Proc. of the ICML*, pp. 118–126, 2014.

Shah, N., Balakrishnan, S., Guntuboyina, A., and Wainwright, M. Stochastically transitive models for pairwise comparisons: Statistical and computational issues. In *International Conference on Machine Learning*, pp. 11–20, 2016a.

Shah, N. B., Balakrishnan, S., and Wainwright, M. J. Feeling the bern: Adaptive estimators for bernoulli probabilities of pairwise comparisons. *arXiv preprint arXiv:1603.06881*, 2016b.

Skorepa, M. *Decision making: a behavioral economic approach*. Palgrave Macmillan, 2010.

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

Szörényi, B., Busa-Fekete, R., Paul, A., and Hüllermeier, E. Online rank elicitation for plackett-luce: A dueling bandits approach. In *NIPS*, pp. 604–612, 2015.

Yue, Y. and Joachims, T. Beat the mean bandit. In *Proc. of the ICML*, pp. 241–248, 2011.

A. Lower bound for WST

Outline: We first present a model that satisfies WST and relate it to a *linear jigsaw puzzle* such that lower bound on jigsaw puzzle implies lower bound for finding an $1/4$-maximum under the model.

Consider the following model with n elements $S = \{1, 2, \ldots, n\}$: $\tilde{p}_{i,i+1} = \frac{1}{2} \forall i < n$, and $\tilde{p}_{i,j} = \mu (0 < \mu < 1/n^{10}), \forall j > i + 1$. This model satisfies WST since there exists an underlying order $\succ$, $i \succ j$ if $i < j$ (because $\tilde{p}_{i,j} > 0$). Observe that 1 is the only $1/4$-maximum under this model.

We prove the Lemma by reducing the above model to the model where μ is replaced by 0.

Note that μ is so small that if we consider a model where we replace μ with 0, the comparisons behave essentially similarly. More formally, let M_μ be the model considered above and M_0 be the model when μ is replaced with 0. Let C denote a sequence of comparisons where each element of the sequence includes the elements compared and its outcome. Further, for each sequence C, let $P_\mu(C)$ and $P_0(C)$ denote the probability of sequence C under models M_μ and M_0 respectively. Now consider a sequence C of comparisons of length $\le n^2/20$. Then

$$\frac{P_0(C)}{P_\mu(C)} \ge \left(\frac{1/2}{1/2+\mu}\right)^{n^2/20} \ge e^{-n^2/(10n^{10})} \ge \frac{6}{7}$$

Thus the probability of any sequence of length $\le \frac{n^2}{20}$ is approximately same under both models. Hence if there is an algorithm that uses $\le \frac{n^2}{20}$ comparisons and w.p.$\ge 7/8$ produces the $1/4$-maximum under M_μ model then applying same algorithm over M_0 model produces the $1/4$-maximum w.p.$\ge \frac{7}{8} \cdot \frac{6}{7} = \frac{3}{4}$. Hence, lower bound of $\Omega(n^2)$ over M_0 model implies a lower bound of $\Omega(n^2)$ over M_μ model.

We now show that under M_0 model, any algorithm requires $\Omega(n^2)$ comparisons to find the $1/4$-maximum w.p.$\ge \frac{3}{4}$, thus proving the Lemma. From now, we only consider model M_0.

Notice that under M_0, whenever two non adjacent elements are compared i.e., i and j with $|i - j| > 1$, the comparison output is a fair coin toss. We make the problem simpler by revealing the extra information of whether elements that are being compared are adjacent or not. Notice that this only makes the problem easier, namely, complexity for modified problem is smaller than that of original problem.

The modified problem is similar to a linear jigsaw puzzle where if we ask a question about two pieces we will know if pieces are adjacent or not and if adjacent, which piece is on the left and the goal is to find the left most piece. To make the proof simpler, we change the question model slightly. In the new model, when we question about ordered pair (e, f), we get to know if e is left neighbor of f. Notice that we can simulate one question in previous model by asking two questions (e, f) and (f, e) in new model.

Now we present proof to show that for linear jigsaw puzzle, any algorithm requires $\Omega(n^2)$ comparisons to find the left most piece with probability $\ge 3/4$. This implies the Lemma.

A.1. Lower bound for Jigsaw Puzzle

Outline: We first describe the main idea briefly. Notice that we start with n unconnected components where each piece refers to an unconnected component. Each new connection (two pieces are connected if they are neighbours) revealed between the pieces reduces the number of unconnected components by 1. We first show that for some small constant c, even after asking $c^2n^2/2$ questions, w.h.p., $< 20cn$ connections are revealed. For this, we divide pieces into two groups based on number of questions asked about them. Group 1: pieces for which $\le cn$ questions are asked and Group 2: pieces for which $> cn$ questions are asked. Then we bound the probability that a question between two pieces from Group 1 results in finding a new connection by $8/n$. Therefore, by asking $c^2n^2/2$ questions w.h.p., we will find $\le 16cn$ such connections. Since total number of questions asked is $c^2n^2/2$, number of pieces about which we have asked $> cn$ questions is $< cn$. Hence w.h.p. total number of connections found using $c^2n^2/2$ comparisons is $< 16cn + 2 \cdot cn < 20cn$.

Since total connections found is $< 20cn$ there are $> (1 - 20c)n$ unconnected components. We show that all unconnected components' leftmost pieces for which we asked less than $\le cn$ questions are all almost equally likely to be the leftmost piece of the entire puzzle. Hence returning such a piece as the leftmost piece of puzzle will give accuracy of only $O(1/n)$. Further we show that the probability that out of $c^2n^2/2$ total questions, we have asked $> cn$ questions about the left most piece, is bounded by $2c$. This results in an upper bound of $3/4$ on probability of success in finding the left most piece.

Setup: We have n elements from 1 to n. Let $\mathcal{P}$ denotes the set of all possible permutations of elements in $[n]$. Note $|\mathcal{P}| = n!$. Adversary chooses a permutation from $\mathcal{P}$ randomly uniformly. And we want to know the top element in the permutation chosen by adversary.

Let $q_{(i,j)}$ corresponds to the question whether i is left neighbour of j and ξ be the set of all questions i.e., $\xi = \{q_{(i,j)} : i \ne j \text{ and } i, j \in [n]\}$.

Let $A(q_{(i,j)})$ denotes the answer of question $q_{(i,j)}$ which takes the value 1, if the answer is "yes" (i is left neighbor of j) and 0 if it is "no". Note that $|\xi| = n(n-1)$, and for any chosen permutation, exactly $n-1$ of these have yes answers.

At each time step we ask a question $q_t = q_{(l_t, r_t)} \in \xi$ and get answer $A_t = A(q_t)$. An algorithm is an strategy to decide which question is to be asked next based on the response of questions asked in past.

Formally any algorithm $\mathbb{A}$ consists of a sequence of functions f_t, such that $q_t = f_t(q_1^{t-1} \times A_1^{t-1}) \in \xi$, we allow f_ts to be random functions. For any reasonable algorithm we can assume $q_t \neq q_k$ for $k \in [t-1]$, i.e. it will not ask same question twice.

We denote set of all the questions asked till time t by $Q_t \triangleq \{q_k \ \forall \ k \in [t]\}$. Let $Y(t) = \{q_j : A(q_j) = 1, \ j \in [t]\}$, denotes the collection of questions asked till time t, which resulted in "yes" answers (or new connections).

Let $\mathcal{P}_t$ be the set of all valid permutations at time t. At start, $\mathcal{P}_0 = \mathcal{P}$. When we ask $q_t = q_{(l_t, r_t)}$, if $A_t = 1$, then to get $\mathcal{P}_t$ from $\mathcal{P}_{t-1}$, we remove all such permutations from $\mathcal{P}_{t-1}$ in which l_t is not the left neighbour of r_t, and if $A_t = 0$ we do the opposite i.e. remove all such permutations from $\mathcal{P}_{t-1}$ in which l_t is left neighbour of r_t to get $\mathcal{P}_t$. After asking t questions all permutations in $\mathcal{P}_t$ have equal chances of being the correct permutation (Since posterior distribution will have equal probability for all the valid permutations, because prior distribution was uniform).

We divide the elements into two groups based on number of questions asked about the element. For that, let $n_i(t) \triangleq |\{k \in [t] : i = l_k \text{ or } i = r_k\}|$, denotes the number of questions asked, which involves element i.

Let $T = c^2n^2/2$, with $c = 1/1000$. From now on we assume $t \leq T$. Let $\chi(t) \triangleq \{i : |n_i(t)| > cn\}$, which is collection of all the elements about which we asked more than cn questions till time t. Then from pigeonhole principle we have:

$$|\chi(t)| \leq cn.$$

Define indicator random variable $I(t)$ which takes values 1, if $A(q(t)) = 1$ and $l_t \notin \chi(t-1)$ and $r_t \notin \chi(t-1)$. Then, we can bound the total number of yes answers $|Y(t)|$ by the sum of $I(k)$ upto time t and size of $\chi(t)$.

$$|Y(t)| \leq \sum_{k \in [t]} I(k) + 2|\chi(t)| \leq \sum_{k \in [t]} I(k) + 2cn,$$

where first inequality follows from: each element in $\chi(t)$ can increase the count of yes answers by 2 at most.

Now we bound the total number of yes answers by $20cn$.

Lemma 13. *For any algorithm $\mathbb{A}$, $T = c^2n^2/2$, $\Pr[|Y(T)| > 20cn] < 1/4$.*

Proof. To show that $|Y(T)| \leq 20cn$, it is enough to show that $\sum_{k \in [T]} I(k) \leq 18cn$.
Let us count the time steps t for which $I(t) = 1$ as success. Then we show that probability of success in a time step is upper bounded by $8/n$ until first $18cn$ successes. Then number of steps taken for k^{th} success for $k < 18cn$ can be thought as geometric distribution with mean $> n/8$. Therefore to get $18cn$ successes we need more than $\sim 18cn \times n/8 > 2cn^2$ steps, and probability of getting as many successes is $c^2n^2/2$, with c small enough is $< 1/4$ (follows from properties of negative binomial distribution, which is sum of geometric distributions).
We complete the proof by showing that

$$\Pr[I(t) = 1 \mid \sum_{k \in [t-1]} I(k) < 18cn] < 8/n \qquad (1)$$

Lets assume $\sum\limits_{k \in [t-1]} I(k) < 18cn$.
Then we prove that for any pair $\alpha, \beta \in [n]$ such that $\alpha, \beta \notin \chi(t-1)$ and $q_{(\alpha,\beta)} \notin Q_{t-1}$, the fraction of permutations in $\mathcal{P}_{t-1}$, in which α is left neighbour of β is $\leq 8/n$.
Let B_1 be the collection of all the permutations in $\mathcal{P}_{t-1}$ in which α is left neighbour of β and α is ranked in top $n/2$ elements. Similarly, let B_2 be the collection of all the permutations in $\mathcal{P}_{t-1}$ in which α is left neighbour of β and α is ranked in bottom $n/2$ elements. We first show that $|\mathcal{P}_{t-1}| \geq |B_1|n/4$.

For each permutation in B_1, we show a set of permutations of size $> n/4$ such that all sets are disjoint and no permutation has α as left neighbour of β. We consider following permutation, which is a member of B_1.

$$b_y, b_{y-1}, \ldots . b_2, b_1, \alpha, \beta, a_1, a_2, a_3, \ldots ., a_{x-1}, a_x$$

Here $x + y + 2 = n$ and $x > n/2 - 2$.

Let u be the minimum index such that algorithm has asked less than cn questions about a_{u-1} and questions (α, a_u), (a_{u-1}, a_u) are not asked before time t i.e.,
$u = \min\{i > 1 : a_{i-1} \notin \chi(t-1) \text{ and } q_{(\alpha,a_i)} \notin Q_{t-1} \ \& \ q_{(a_{i-1},a_i)} \notin Q_{t-1}\}$.

Note that $u \leq 1 + |\{i : a_{i-1} \in \chi(t-1)\}| +$
$|\{i : q_{(\alpha,a_i)} \in Q_{t-1}\}| + |\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}|$.

Observe that:

$$|\{i : a_{i-1} \in \chi(t-1)\}| \leq |\chi(t-1)| \leq cn \qquad (2)$$

and

$$|\{i : q_{(\alpha,a_i)} \in Q_{t-1}\}| \leq n_\alpha(t-1) \leq cn. \qquad (3)$$

If a_{i-1} is right neighbour of a_i in a valid permutation in $\mathcal{P}_{t-1}$, then if $q_{(a_{i-1},a_i)}$ has been asked then it would have resulted in answer "yes" because if answer was "no" then they can't be neighbours in a valid permutation. Hence $|\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}|$ is upper bounded by $|Y(t-1)|$. Combining these and with our assumption above that $\sum_{k\in[t-1]} I(k) \leq 18cn$, we get,

$$u \leq 1+cn+cn+2cn+\sum_{k\in[t-1]} I(k) \leq 1+22cn \leq n/32. \tag{4}$$

Let Λ be the set of indices $i > u+1$ such that questions $q_{(\alpha,a_i)}, q_{(a_{u-1},a_i)}, q_{(a_{i-1},a_i)}$ and $q_{(a_{i-1},\beta)}$ are not asked before time $t-1$ i.e.,
$\Lambda = \{i > u+1 : q_{(\alpha,a_i)} \notin Q_{t-1} \ \& \ q_{(a_{(u-1)},a_i)} \notin Q_{t-1} \ \& \ q_{(a_{i-1},a_i)} \notin Q_{t-1} \ \& \ q_{(a_{i-1},\beta)} \notin Q_{t-1}\}$.

Notice that

$$\begin{aligned}|\Lambda| \geq x-u-1-|\{i : q_{(\alpha,a_i)} \in Q_{t-1}\}| \\ -|\{i : q_{(a_{(u-1)},a_i)} \in Q_{t-1}\}|-|\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}| \\ -|\{i : q_{(a_{i-1},\beta)} \in Q_{t-1}\}|.\end{aligned}$$

Observing $x > n/2-2$, then following the steps similar to the proof of equation (4) we get:

$$\begin{aligned}|\Lambda| \geq\ & n/2-2-n/32-1-n_\alpha(t-1) \\ & -n_{a_{(u-1)}}(t-1)-|Y(t-1)|-n_\beta(t-1) \\ \geq\ & n/2-2-n/32-1-cn-cn-20cn-cn \\ \geq\ & 15n/32-3-23cn \\ \geq\ & n/4.\end{aligned}$$

For any $v \in \Lambda$, we claim that following permutation will also be part of $\mathcal{P}_{t-1}$:

$$\begin{aligned}&\{b_y, b_{y-1}, ..., b_2, b_1, \alpha\}, \{a_u, a_{u+1}, ..., a_{v-1}\}, \\ &\{\beta, a_1, a_2, ..., a_{u-1}\}, \{a_v, a_{v+1}, ...a_x\}.\end{aligned}$$

In above permutation we have put curly parentheses to highlight the changes made from original permutation. To show that the permutation above is valid, i.e., lies in $\mathcal{P}_{t-1}$, we need to show that (α, a_u), (a_{v-1}, β) and (a_{u-1}, a_v) are valid connections,which is easy to do from the definition of u and Λ. We skip the details here.

The original permutation can be uniquely recovered by finding the link (a_{u-1}, a_v), which can be found using the fact that (a_{u-1}, a_v) is the first set of consecutive elements (i, j) which are ranked after β such that $q_{(i,j)} \notin Q_{t-1}$, $q_{(\alpha,j)} \notin Q_{t-1}$ and $i \notin \chi(t-1)$ (this again can be verified from the definition of u and Λ). Therefore, to each permutation in B_1, we can map a disjoint set of $n/4$ permutations, which are part of $\mathcal{P}_{t-1}$ and in which α is not the right neighbour of β.
Hence $|\mathcal{P}_{t-1}| \geq |\Lambda| \times |B_1| \geq |B_1|n/4$. Similarly, it can be shown that $|\mathcal{P}_{t-1}| \geq |B_2|n/4$. Therefore,

$$\begin{aligned}&\Pr[A(q_{(\alpha,\beta)}) = 1| \sum_{k\in[t-1]} I(k) < 18cn] \\ &= \frac{|B_1|+|B_2|}{|\mathcal{P}_{t-1}|} \leq 8/n.\end{aligned}$$

Thus we get:

$$\Pr[I(t) = 1| \sum_{k\in[t-1]} I(k) < 18cn] \leq 8/n.$$

□

Next, we prove that if $\beta \notin \chi(t-1)$ and $|Y(t-1)| \leq 20cn$, then $\Pr[\beta$ is top element in permutation$] \leq 2/n \ \forall\ t \leq c^2n^2/2$. We use the same idea as earlier in the proof, to each permutation in $\mathcal{P}_{t-1}$ with β as top element, we can map a disjoint set of $n/2$ permutations which are also part of $\mathcal{P}_{t-1}$, and don't have β at the top.
Consider the following permutation of $\mathcal{P}_{t-1}$, in which β is a top element.

$$\beta, a_1, a_2, a_3, ..., a_{n-2}, a_{n-1}$$

Let $\widehat{u}$ be the minimum index such that question $q_{(a_{\widehat{u}-1},a_{\widehat{u}})}$ is not asked before and algorithm didn't ask more than cn questions about $a_{\widehat{u}-1}$ i.e.,
$\widehat{u} = \min\{i > 1 : a_{i-1} \notin \chi(t-1)$ and $q_{(a_{i-1},a_i)} \notin Q_{t-1}\}$.
Note that
$\widehat{u} \leq 1+|\{i : a_{i-1} \in \chi(t-1)\}|+|\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}|$.

As before

$$\widehat{u} \leq 1+cn+|Y(t-1)| < 1+cn+20cn < n/16.$$

Let $\widehat{\Lambda}$ be the set of all indices $i > \widehat{u}+1$ such that questions $q_{(a_{(\widehat{u}-1)},a_i)}$, $q_{(a_{i-1},a_i)}$, $q_{(a_{i-1},\beta)}$ are not asked before i.e., $\widehat{\Lambda} = \{i > \widehat{u}+1 : q_{(a_{(\widehat{u}-1)},a_i)} \notin Q_{t-1} \ \& \ q_{(a_{i-1},a_i)} \notin Q_{t-1} \ \& \ q_{(a_{i-1},\beta)} \notin Q_{t-1}\}$. Then observe that

$$\begin{aligned}|\widehat{\Lambda}| \geq n-2-\widehat{u}-|\{i : q_{(a_{u-1},a_i)} \in Q_{t-1}\}| \\ -|\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}|-|\{i : q_{(a_{i-1},\beta)} \in Q_{t-1}\}| \\ \geq n-2-n/16-cn-|Y(t-1)|-cn > n/2\end{aligned}$$

As before, for any $\widehat{v} \in \widehat{\Lambda}$, we claim that following permutation will also be part of $\mathcal{P}_{t-1}$:

$$\begin{aligned}&\{a_{\widehat{u}}, a_{\widehat{u}+1}, ..., a_{\widehat{v}-1}\}, \{\beta, a_1, a_2, ..., a_{\widehat{u}-2}, a_{\widehat{u}-1}\}, \\ &\{a_{\widehat{v}}, a_{\widehat{v}+1}, ...a_{n-1}\}\end{aligned}$$

And it is easy to verify that from this we can get the original permutation back uniquely. Hence, we have showed that if $|Y(t-1)| \le 20cn$, then for any $\beta \notin \chi(t-1)$, less than $2/n$ fraction of all the permutation $\mathcal{P}_{t-1}$ contains β as the top element. Therefore, at the end if algorithm returns an element $\beta \notin \chi(T)$ as top element, probability of it being correct is upper bounded by $2/n$. And if we predict the top element β such that $\beta \in \chi(T)$, then success probability is upper bounded by the probability that $\chi(T)$ contains the top element. At $t = 0$, $\chi(0)$ is empty. New elements are added in $\chi(.)$ as we ask new questions. When we asked $cn + 1$'th question about that element, from the previous discussion at that point probability of that element being top element is upper bounded by $2/n$. And we have at most cn elements in $\chi(T)$, therefore:

$$\Pr\left[\chi(T) \text{ contains top element}\middle| |Y(t-1)| \le 20cn\right] \\ \le cn \times 2/n = 2c.$$

Therefore, probability that our prediction of top element is correct:

$$\begin{aligned} &\Pr[\text{Algorithm returns correct top element}] \\ \le &\Pr[|Y(T)| > 20cn] \\ &+ \Pr[\chi(T) \text{ contains top element} || Y(t-1) \le 20cn] \\ &+ \Pr[\text{Algorithm returns } \beta \notin \chi(T) \\ &\ \text{and is correct}|Y(t-1)| \le 20cn] \\ &\le 1/4 + 2c + 2/n < 3/4. \end{aligned}$$

B. Estimating pairwise probabilities for WST

B.1. BRUTE-FORCE

For WST model, (Falahatgar et al., 2017a) presented a PAC-ranking algorithm that uses $\mathcal{O}\left(\frac{n^2}{\epsilon^2}\log\frac{n}{\delta}\right)$ comparisons. In the process, they present a trivial algorithm to estimate all pairwise probabilities to accuracy of ϵ using $\mathcal{O}\left(\frac{n^2 \log n}{\epsilon^2}\right)$ comparisons. We present their algorithm here for completeness.

B.1.1. EST-PROB

EST-PROB(i, j, ϵ, δ) compares i and j for $\frac{1}{2\epsilon^2}\log\frac{2}{\delta}$ times and returns the fraction of times i won. With probability $\ge 1-\delta$, this fraction approximates $p_{i,j}$ to an accuracy of ϵ.

Algorithm 6 EST-PROB

```
1: inputs
2:     element $i$, element $j$, bias $\epsilon$, confidence $\delta$.
3: Compare $i$ and $j$ for $\frac{1}{2\epsilon^2}\log\frac{2}{\delta}$ times.
4: return Fraction of times $i$ won
```

Lemma 14. *EST-PROB(i, j, ϵ, δ) uses $\frac{1}{2\epsilon^2}\log\frac{2}{\delta}$ comparisons and w.p.$\ge 1-\delta$ approximates $p_{i,j}$ to an additive error of ϵ.*

Proof. Proof follows from Hoeffding's inequality. □

B.1.2. BRUTE-FORCE

BRUTE-FORCE(S, ϵ, δ) approximates all pairwise probabilities $p_{i,j}$ using EST-PROB$(i, j, \epsilon, \frac{2\delta}{|S|(|S|-1)})$. Observe that w.p.$\ge 1 - \frac{2\delta}{|S|(|S|-1)}$, EST-PROB$(i, j, \epsilon, \frac{2\delta}{|S|(|S|-1)})$ approximates $\tilde{p}_{i,j}$ to an accuracy of ϵ. Hence by union bound, w.p.$\ge 1-\delta$, BRUTE-FORCE(S, ϵ, δ) approximates all pairwise probabilities to an accuracy of ϵ.

Algorithm 7 BRUTE-FORCE

```
1: inputs
2:     Set S, bias $\epsilon$, confidence $\delta$
3: for every pair $\{i, j\}$ such that $i, j \in S$ do
4:     $\hat{p}(i, j) \leftarrow$ EST-PROB$(i, j, \frac{\epsilon}{2}, \frac{2\delta}{n(n-1)})$
5:     $\hat{p}(j, i) \leftarrow 1 - p(i, j)$
6: end for
```

In the below Lemma, we bound complexity of BRUTE-FORCE and prove its correctness.

Lemma 15. *BRUTE-FORCE(S, ϵ, δ) uses $\mathcal{O}(\frac{n^2}{\epsilon^2}\log\frac{n}{\delta})$ comparisons and w.p.$\ge 1-\delta$ approximates all pairwise probabilities to accuracy of ϵ.*

Notice that under WST, once we all pairwise probabilities are approximated to accuracy of $\epsilon/2$, one can find ϵ-maximum and ϵ-ranking.

C. PAC maxing for MST

C.1. Property of MST

We first prove a property of MST that helps us in bounding comparisons for MST.

Lemma 16. *Under MST, if $\epsilon > 0$, $\tilde{p}_{i,j} \le \epsilon$, $\tilde{p}_{k,j} > \epsilon$ then $\tilde{p}_{i,k} \le \epsilon$.*

Proof. We assume that $\tilde{p}_{i,k} > \epsilon$ and prove Lemma by contradiction.

Since $\tilde{p}_{i,k} > \epsilon$ and $\tilde{p}_{k,j} > \epsilon$, then $i \succ k \succ j$ and hence by MST,

$$\tilde{p}_{i,j} \ge \min(\tilde{p}_{i,k}, \tilde{p}_{k,j}) > \epsilon$$

which contradicts the Lemma statement.

Hence $\tilde{p}_{i,k} \le \epsilon$. □

C.2. COMPARE

For better performance in practice COMPARE stops earlier if $\tilde{p}_{i,j} \ll \epsilon_l$ or $\tilde{p}_{i,j} \gg \epsilon_u$. This step does not affect our bounds.

Algorithm 8 COMPARE

```
1: inputs
2:     element $i$, element $j$, lower bias $\epsilon_l \ge 0$, upper bias $\epsilon_u > \epsilon_l$, confidence $\delta$
3: initialize
4:     $\epsilon_m = \frac{\epsilon_l+\epsilon_u}{2}$, $\hat{\tilde{p}}_{i,j} \leftarrow 0$, $\hat{c} \leftarrow \frac{1}{2}$, $t \leftarrow 0$, $w \leftarrow 0$
5: while $|\hat{\tilde{p}}_{i,j} - \epsilon_m| \le \hat{c} + (\epsilon_u - \epsilon_l)/4$ and $t \le \frac{8}{(\epsilon_u-\epsilon_l)^2}\log\frac{2}{\delta}$ do
6:     Compare $i$ and $j$
7:     if $i$ wins then
8:         $w \leftarrow w+1$
9:     end if
10:    $t \leftarrow t+1$
11:    $\hat{\tilde{p}}_{i,j} \leftarrow \frac{w}{t} - \frac{1}{2}$, $\hat{c} \leftarrow \sqrt{\frac{1}{2t}\log\frac{4t^2}{\delta}}$
12: end while
13: if $\hat{\tilde{p}}_{i,j} < (\epsilon_l + \epsilon_m)/2$ then
14:     return 1
15: end if
16: if $\hat{\tilde{p}}_{i,j} > (\epsilon_m + \epsilon_u)/2$ then
17:     return 3
18: end if
19: return 2
```

C.2.1. PROOF FOR LEMMA 2

Proof. We first bound the number of comparisons.

Notice that COMPARE$(i, j, \epsilon_l, \epsilon_u, \delta)$ compares elements i and j for at most $m = \frac{8}{(\epsilon_u-\epsilon_l)^2}\log\frac{2}{\delta}$ times and hence bound on comparisons follows.

We first show correctness for case of $\tilde{p}_{i,j} \le \epsilon_l$.

Let $\hat{p}^t_{i,j}$ and $\hat{c}^t$ denote $\hat{p}_{i,j}$ and $\hat{c}$ respectively after t comparisons between i and j during COMPARE$(i, j, \epsilon_l, \epsilon_u, \delta)$. COMPARE$(i, j, \epsilon_l, \epsilon_u, \delta)$ does not output 1 only if $\hat{p}^t_{i,j} \ge \frac{1}{2} + \frac{\epsilon_l+3\epsilon_u}{4} + \hat{c}^t$ for any $t < m = \frac{8}{(\epsilon_l-\epsilon_u)^2}\log\frac{2}{\delta}$ or if $\hat{p}^m_{i,j} > \frac{1}{2} + \frac{3\epsilon_l+\epsilon_u}{4}$. We bound the probability of either of these events by $\frac{\delta}{2}$ and the result follows from the union bound.

By Hoeffding's inequality,

$$\begin{aligned} Pr\left(\hat{p}^t_{i,j} > \frac{1}{2} + \frac{\epsilon_l+3\epsilon_u}{4} + \hat{c}^t\right) &\le Pr\left(\hat{p}^t_{i,j} > \frac{1}{2} + \epsilon_l + \hat{c}^t\right) \\ &\le e^{-2t(\hat{c}^t)^2} \\ &= e^{-\log\frac{4t^2}{\delta}} \\ &= \frac{\delta}{4t^2}. \end{aligned}$$

By the union bound, $Pr\left(\exists t \text{ s.t. } \hat{p}^t_{i,j} > \frac{1}{2} + \frac{\epsilon_l+3\epsilon_u}{4} + \hat{c}^t\right) \le \sum_t \frac{\delta}{4t^2} \le \frac{\delta}{2}$.

Similarly, by Hoeffding's inequality,

$$\begin{aligned} Pr\left(\hat{p}^m_{i,j} > \frac{1}{2} + \frac{3\epsilon_l+\epsilon_u}{4}\right) &\le e^{-2m((\epsilon_u-\epsilon_l)/4)^2} \\ &= e^{-\log\frac{2}{\delta}} \\ &= \frac{\delta}{2}. \end{aligned}$$

Hence if $\tilde{p}_{i,j} < \epsilon_l$, w.p.$\ge 1-\delta$, COMPARE outputs 1. We can prove similarly for cases $\tilde{p}_{i,j} > \epsilon_u$, $\epsilon_l \le \tilde{p}_{i,j} \le (\epsilon_l+\epsilon_u)/2$ and $(\epsilon_u+\epsilon_l)/2 < \tilde{p}_{i,j} \le \epsilon_u$. □

C.3. Proof for Lemma 3

Proof. Observe that COMPARE is called for at most $|S|(|S|-1)/2$ times. Since when calling COMPARE, we use confidence parameter of $\frac{2\delta}{|S|^2}$, the probability that COMPARE gives expected answer always is $\ge 1-\delta$. From now, we assume that COMPARE$(i, j, \epsilon_l, \epsilon_u, 2\delta/|S|^2)$ always returns 1 if $\tilde{p}_{i,j} \le \epsilon_l$, 1 or 2 if $\epsilon_l < \tilde{p}_{i,j} \le (\epsilon_l+\epsilon_u)/2$, 2 or 3 if $(\epsilon_l+\epsilon_u)/2 < \tilde{p}_{i,j} \le \epsilon_u$ and 3 if $\tilde{p}_{i,j} > \epsilon_u$.

Let r^1 be the value of anchor r in the beginning and r^t be the value of r after $t-1$ changes of r.

We first show that $\tilde{p}_{r^t,r^l} > (\epsilon_l+\epsilon_u)/2$ for all $l < t$. Since $r^{t+1} = e$ only if COMPARE$(e, r^t, \epsilon_l, \epsilon_u, \frac{2\delta}{|S|^2})$ returns 3, $\tilde{p}_{r^{t+1},r^t} > (\epsilon_l+\epsilon_u)/2$. Hence by MST, $\tilde{p}_{r^t,r^l} > (\epsilon_l+\epsilon_u)/2$ for all $l < t$.

We now bound the number of comparisons. We first bound the number of COMPARE calls during SOFT-SEQ-ELIM. Let T be the set of elements for which r^1 is ϵ_l-preferable i.e., $T \overset{\text{def}}{=} \{e : \tilde{p}_{e,r^1} \le \epsilon_l\}$. Since r^1 is an (ϵ_l, m)-*good anchor* element, $|T| \ge |S| - m$. Notice that for $e \in T$, $\tilde{p}_{e,r^1} \le \epsilon_l$ and since $\tilde{p}_{r^t,r^1} > (\epsilon_u+\epsilon_l)/2 > \epsilon_l$ by MST and Lemma 16, $\tilde{p}_{e,r^t} \le \epsilon_l$ $\forall t$. Hence COMPARE$(e, r^t, \epsilon_l, \epsilon_u, 2\delta/|S|^2)$ returns 1 for all t and therefore, number of COMPARE calls spent on e is 1. Thus number of COMPARE calls spent on set T is $|T|$. Since elements in T will not become anchors, there are atmost $|S|-|T| \le m$ rounds and hence number of COMPARE calls spent on an element in $S \setminus T$ is $\le m$. Therefore total number of COMPARE calls is

$$\le |T| + m|S \setminus T| \le |S| + m^2.$$

Since each call of COMPARE$(i, j, \epsilon_l, \epsilon_u, 2\delta/|S|^2)$ uses $\mathcal{O}\left(\frac{1}{(\epsilon_u-\epsilon_l)^2}\log\frac{|S|}{\delta}\right)$ comparisons, bound on comparisons follows.

We now show that output is an ϵ_u-maximum. Let r^o be the output. We show that if $\tilde{p}_{e,r^o} > (\epsilon_l+\epsilon_u)/2$, then e is not eliminated before last round. Since $\tilde{p}_{e,r^o} > (\epsilon_l+\epsilon_u)/2$, by MST $\tilde{p}_{e,r^t} > (\epsilon_l+\epsilon_u)/2$ and hence not omitted by r^t (since COMPARE$(e, r^t, \epsilon_l, \epsilon_u, 2\delta/|S|^2)$ does not return 1). Hence all elements for which r^o is not $(\epsilon_l+\epsilon_u)/2$-preferable are present in the last round. Since anchor element is not updated in last round, COMPARE$(e, r^o, \epsilon_l, \epsilon_u, 2\delta/|S|^2)$ didn't return 3 for any remaining element and hence $\tilde{p}_{e,r^0} \le \epsilon_u$ for all elements in the last round and r^o is an ϵ_u-maximum of S.

Further notice that either $r^o = r$ or $\tilde{p}_{r^o,r} > (\epsilon_l+\epsilon_u)/2$. □

C.4. Proof for Lemma 5

Proof. Let $m' = \frac{|S|}{m}\log\frac{|S|}{\delta}$. We now show that w.p.$\ge 1-\delta$, r is an (ϵ, m')-good anchor element. We prove this by showing that for every element e in S which is not an (ϵ, m')-*good anchor* element, Q will contain an element for which e is not ϵ-preferable. Let $e \in S$ be not an (ϵ, m')-*good anchor* element, then there are more than m' elements for which e is not ϵ-preferable. The probability that Q does not contain any element for which e is not ϵ-preferable is

$$\le \left(1 - \frac{m'}{|S|}\right)^m = \left(1 - \frac{\log(|S|/\delta)}{m}\right)^m \le \frac{\delta}{|S|}.$$

Let T be the set of all elements in S which are not (ϵ, m')-*good anchor* elements. Hence by union bound, w.p.$\ge 1-\delta$, for every element $e \in T$, Q has an element for which e is not ϵ-preferable.

If $r \in T$, then Q has an element for which r is not ϵ-preferable. But this contradicts our assumption that r is an ϵ-maximum of Q hence $r \notin T$. Therefore r is an (ϵ, m')-*good anchor* element. □

C.5. Proof for Lemma 6

Proof. Since $|Q| = \sqrt{|S|\log(4|S|/\delta)}$, by Corollary 4, SOFT-SEQ-ELIM$(Q, a, 0, \epsilon/2, \delta/4)$ uses

$$\mathcal{O}\left(\frac{|S|\log(4|S|/\delta)}{\epsilon^2}\log\frac{|S|\log(4|S|/\delta)}{\delta}\right)$$
$$= \mathcal{O}\left(\frac{|S|}{\epsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right)$$

comparisons and w.p. $\ge 1-\delta/4$, outputs r, an $\epsilon/2$-maximum of Q.

By Lemma 5, w.p.$\ge 1-\delta/4$, r, an $\epsilon/2$- maximum of Q is an $(\epsilon/2, \sqrt{|S|\log(4|S|/\delta)})$-*good anchor* element of S.

Since r is an $(\epsilon/2, \sqrt{|S|\log(4|S|/\delta)})$-*good anchor* element, by Lemma 3, w.p.$\ge 1-\delta/2$, SOFT-SEQ-ELIM$(S, r, \epsilon/2, \epsilon, \delta/2)$ uses

$$\mathcal{O}\left(\frac{|S| + (\sqrt{|S|\log(4|S|/\delta)})^2}{\epsilon^2}\log\frac{|S|}{\delta}\right)$$
$$= \mathcal{O}\left(\frac{|S|}{\epsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right)$$

comparisons and outputs an ϵ-maximum of S.

Proof follows from union bound. □

C.6. Proof of Lemma 7

Proof. Since Q contains $|S|^{3/4}$ elements, by Lemma 6, w.p.$\ge 1-\delta/3$, NEAR-OPT-MAX$(Q, \frac{\epsilon}{2}, \frac{\delta}{3})$ outputs an $\frac{\epsilon}{2}$-maximum of Q and uses

$$\mathcal{O}\left(\frac{|S|^{3/4}}{\epsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right) \overset{(a)}{=} \mathcal{O}\left(\frac{|S|^{3/4}}{\epsilon^2}\log\frac{|S|}{\delta}|S|^{1/4}\right)$$
$$\overset{(b)}{=} \mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$$

comparisons where (a) is because $\delta \ge \min(1/|S|, e^{-|S|^{1/4}})$ and (b) is because $\delta \le \frac{1}{|S|^{1/3}}$.

By Lemma 5, w.p.$\ge 1-\delta/3$, an $\epsilon/2$-maximum of Q is an $\left(\frac{\epsilon}{2}, \frac{|S|}{|S|^{3/4}}\log\frac{3|S|}{\delta}\right)$-*good anchor* element and hence r is an $\left(\epsilon/2, |S|^{1/4}\log\frac{3|S|}{\delta}\right)$-*good anchor* element of S.

If r is an $\left(\epsilon/2, |S|^{1/4}\log\frac{3|S|}{\delta}\right)$-*good anchor* element of S, by Lemma 3, w.p.$\ge 1-\delta/3$, SOFT-SEQ-ELIM$(S, r, \frac{\epsilon}{2}, \epsilon, \frac{\delta}{4})$ outputs an ϵ-maximum of S and uses

$$\mathcal{O}\left(\frac{|S| + (|S|^{1/4}\log(3|S|/\delta))^2}{\epsilon^2}\log\frac{|S|}{\delta}\right)$$
$$\overset{(a)}{=} \mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{|S|}{\delta}\right)$$
$$\overset{(b)}{=} \mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$$

comparisons where (a) is because $\delta \ge \min(1/|S|, e^{-|S|^{1/4}})$ and (b) is because $\delta \le \frac{1}{n^{1/3}}$.

Result follows by union bound. □

Algorithm 9 COMPARE2

```
1: inputs
2:     element $i$, element $j$, lower bias $\epsilon_l \geq 0$, upper bias $\epsilon_u > \epsilon_l$, confidence $\delta$
3: $\epsilon_m \leftarrow (\epsilon_l + \epsilon_u)/2$, $\hat{\tilde{p}}_{i,j} \leftarrow 0$, $\hat{c} \leftarrow \frac{1}{2}$, $t \leftarrow 0$, $w \leftarrow 0$
4: while $|\hat{\tilde{p}}_{i,j} - \epsilon_m| \leq \hat{c}$ and $t \leq \frac{2}{(\epsilon_u-\epsilon_l)^2} \log \frac{2}{\delta}$ do
5:     Compare $i$ and $j$
6:     if $i$ wins then
7:         $w \leftarrow w + 1$
8:     end if
9:     $t \leftarrow t + 1$
10:    $\hat{\tilde{p}}_{i,j} \leftarrow \frac{w}{t} - \frac{1}{2}$, $\hat{c} \leftarrow \sqrt{\frac{1}{2t} \log \frac{4t^2}{\delta}}$
11: end while
12: if $\hat{\tilde{p}}_{i,j} \leq \epsilon_m$ then
13:     return 1
14: end if
15: return 2
```

C.7. COMPARE2

Lemma 17. *For $\epsilon_u > \epsilon_l$, COMPARE2$(i, j, \epsilon_l, \epsilon_u, \delta)$ uses $\leq \frac{2}{(\epsilon_u-\epsilon_l)^2} \log \frac{2}{\delta}$ comparisons and if $\tilde{p}_{i,j} \leq \epsilon_l$, then w.p.$\geq 1-\delta$, it returns 1, else if $\tilde{p}_{i,j} \geq \epsilon_u$, w.p.$\geq 1-\delta$, it returns 2.*

C.8. High Ranges of Confidence

Notice that number of comparisons used by NEAR-OPT-MAX on a set of size $|S|/(\log|S|)^2$ is

$$\mathcal{O}\left(\frac{|S|/(\log|S|)^2}{\epsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right) = \mathcal{O}\left(\frac{|S|}{\epsilon^2}\right)$$

where last equality is because of $\delta \leq \frac{1}{|S|^{1/3}}$. So we can find an $\epsilon/2$-maximum over a random set of size $|S|/(\log|S|)^2$ and use this element to prune the original set S and use SOFT-SEQ-ELIM over the pruned set. But it turns out we need multiple rounds of pruning and SOFT-SEQ-ELIM before we increase set size from $|S|/(\log|S|)^2$ to $|S|$. In each round we increase set size by a factor of $1/\delta$ until we reach the set size of $|S|$. We first present PRUNE that we use for pruning a set with an anchor element.

C.8.1. PRUNE

We do pruning similar to that in (Falahatgar et al., 2017a) but here we want to ensure that all better elements are still surviving not just the absolute maximum element. Hence we use similar pruning technique with small tweaks and derive different guarantees.

PRUNE takes five parameters: input set S, anchor element a, lower bias ϵ_l, upper bias ϵ_u and confidence δ. Goal of PRUNE is to output a set of size $\leq \frac{4\log(2|S|/\delta)}{\delta}$ (we later show that for our purpose this quantity is $\mathcal{O}(|S|/\log|S|)$) such that it contains all elements for which a is not ϵ_u-preferable. If a is an (ϵ_l, n_1)-*good anchor* element, with $n_1 \leq \frac{2\log(2|S|/\delta)}{\delta}$, w.p.$\geq 1 - n_1\delta$, PRUNE$(S, a, \epsilon_l, \epsilon_u, \delta)$ achieves this goal using $\mathcal{O}\left(\frac{|S|}{(\epsilon_u-\epsilon_l)^2}\log\frac{1}{\delta}\right)$ comparisons.

PRUNE prunes input set in rounds. At each round, it compares the remaining elements with the anchor element (sufficient number of times) and if it deems the element to be bad it eliminates the element. This ensures that number of bad elements decrease approximately by a factor of δ after each round. Notice that since number of elements decrease after each round, PRUNE can afford to compare elements for more times in latter rounds for better accuracy. This process is continued until number of remaining elements is less than required target $\frac{4\log(2|S|/\delta)}{\delta}$.

For an anchor a, we call an element e as bad if $\tilde{p}_{e,a} \leq \epsilon_l$ and good if $\tilde{p}_{e,a} \geq \epsilon_u$. We want to ensure that number of bad elements decrease after each round and good elements never get eliminated. We can use COMPARE for this comparison. But we would like to mention that since requirement of comparison subroutine is less stringent than that required in SOFT-SEQ-ELIM, comparison subroutine COMPARE2 in (Falahatgar et al., 2017a) suffices here which in some cases can save a factor of 4 but has same orderwise complexity.

For completeness, we represent subroutine COMPARE2 in Appendix C.7.

Algorithm 10 PRUNE

```
1: inputs
2:     Set $S$, element $a$, lower bias $\epsilon_l$, upper bias $\epsilon_u$, confidence $\delta$.
3: $t \leftarrow 1$
4: $S_1 \leftarrow S$
5: while $|S_t| > 4\frac{\log\frac{2|S|}{\delta}}{\delta}$ and $t < \log^2 n$ do
6:     Initialize: $Q_t \leftarrow \emptyset$
7:     for $e$ in $S_t$ do
8:         if COMPARE2$(e, a, \epsilon_l, \epsilon_u, \delta/2^{t+1}) = 1$ then
9:             $Q_t \leftarrow Q_t \bigcup \{e\}$
10:        end if
11:    end for
12:    $S_{t+1} \leftarrow S_t \setminus Q_t$
13:    $t \leftarrow t + 1$
14: end while
15: return $S_t$.
```

We now bound the number of comparisons used by PRUNE and prove its correctness.

Lemma 18. *If a is an (ϵ_l, n_1)-good anchor element with $n_1 \leq \frac{2\log(2|S|/\delta)}{\delta}$ then w.p.$\geq 1 - \frac{\delta}{2}$,* PRUNE$(S, a, \epsilon_l, \epsilon_u, \delta)$ *uses $\mathcal{O}\left(\frac{|S|}{(\epsilon_u-\epsilon_l)^2}\log\frac{1}{\delta}\right)$ comparisons and outputs a set of* *size less than $\frac{4\log(2|S|/\delta)}{\delta}$. Further if a is not an ϵ_u-maximum of S then w.p.$\geq 1-\frac{n_1\delta}{2}$, the output set contains all elements for which a is not ϵ_u- preferable.*

Proof. Proof is similar to proof of Lemma 5 in (Falahatgar et al., 2017a). We present the proof for reader's convenience.

We first show that any element $e \in S$ for which a is not ϵ_u-preferable is part of output set w.p.$\geq 1-\delta/2$. e gets eliminated in round t if COMPARE2$(e, a, \epsilon_l, \epsilon_u, \delta/2^{t+1})$ returns 1 but since $\tilde{p}_{e,a} > \epsilon_u$, w.p.$\geq 1-\delta/2^{t+1}$, COMPARE2$(e, a, \epsilon_l, \epsilon_u, \delta/2^{t+1})$ returns 2. Hence e gets eliminated in round t w.p.$\leq \delta/2^{t+1}$. Therefore by union bound, probability that e gets eliminated in any one of the rounds is $\leq \sum_t \delta/2^{t+1} \leq \delta/2$.

Since a is an (ϵ_l, n_1)-*good anchor* element number of elements for which a is not ϵ_u-preferable is $\leq n_1$. Hence invoking union bound once again, probability that any such element gets eliminated is $\leq n_1\delta/2$.

Now we bound output set size and number of comparisons used.

Notice that any element e for which a is ϵ_l-preferable if present in round t then gets eliminated in that round w.p.$\geq 1-\delta/2^{t+1}$, since COMPARE2$(e, a, \epsilon_l, \epsilon_u, \delta/2^{t+1})$ returns 1 with that probability. Hence if at the beginning of a round, the number of such elements (for which a is ϵ_l-preferable) is more than $\frac{2\log(2|S|/\delta)}{\delta}$, the probability that number of these elements surviving after round does not reduce by at least a factor of δ is

$$\begin{aligned}\leq e^{-\frac{2\log(2|S|/\delta)}{\delta}D(\delta||\delta/2^{t+1})} &\leq e^{-\frac{2\log(2|S|/\delta)}{\delta}D(\delta||\delta/4)}\\ &\leq e^{-\frac{2\log(2|S|/\delta)}{\delta}\delta/2}\\ &= e^{-\log(2|S|/\delta)}\\ &= \frac{\delta}{2|S|}.\end{aligned}$$

If number of elements for which a is ϵ_l-preferable decrease by a factor of δ after each round, then number of such elements fall below $\frac{2\log(2|S|/\delta)}{\delta}$ in $\leq \log_{1/\delta}\frac{|S|}{n_1}$ (since $n_1 \leq \frac{2\log(2|S|/\delta)}{\delta}$). Hence by union bound, w.p.$\geq 1-\delta/2$, number of such elements fall below $\frac{2\log(2|S|/\delta)}{\delta}$ in $\log_{1/\delta}\frac{|S|}{n_1}$ rounds. Henceforth we assume this and bound the number of comparisons.

Notice that number of elements for which a is not ϵ_l-preferable in round t is $\leq |S|\delta^{t-1}$. Since COMPARE2$(e, a, \epsilon_l, \epsilon_u, \delta')$ uses $\frac{2}{(\epsilon_u-\epsilon_l)^2}\log\frac{1}{\delta'}$ comparisons, total number of comparisons used on elements for which a is not ϵ_l-preferable is

$$\begin{aligned}&\leq \sum_{t=1}^{\log_{1/\delta}\frac{|S|}{n_1}} \frac{2|S|\delta^{t-1}}{(\epsilon_u-\epsilon_l)^2}\log\frac{2^{t+1}}{\delta}\\ &\leq \frac{2|S|}{(\epsilon_u-\epsilon_l)^2}\sum_{t=1}^{\infty}\left(\delta^{t-1}\log\frac{1}{\delta}+(t+1)\delta^{t-1}\log 2\right)\\ &= \mathcal{O}\left(\frac{|S|}{(\epsilon_u-\epsilon_l)^2}\log\frac{1}{\delta}\right)\end{aligned}$$

where last equality follows since $\sum_{i=1}^{\infty} x^i$ and $\sum_i^{\infty}(i+1)x^{i-1}$ are bounded for $x \leq 1/2$.

We now bound the comparisons on elements for which a is not ϵ_l-preferable using number of rounds for which PRUNE runs. Number of comparisons used on elements for which a is not ϵ_l-preferable is

$$\begin{aligned}&\leq \sum_{t=1}^{\log_{1/\delta}\frac{|S|}{n_1}} \frac{2n_1}{(\epsilon_u-\epsilon_l)^2}\log\frac{2^{t+1}}{\delta}\\ &\leq \frac{2n_1}{(\epsilon_u-\epsilon_l)^2}\sum_{t=1}^{\log_{1/\delta}\frac{|S|}{n_1}}\left(\log\frac{1}{\delta}+(t+1)\log 2\right)\\ &\leq \frac{2n_1}{(\epsilon_u-\epsilon_l)^2}\left(\left(\log_{1/\delta}\frac{|S|}{n_1}\right)\log\frac{1}{\delta}+\left(2\log_{1/\delta}\frac{|S|}{n_1}\right)^2\right)\\ &= \mathcal{O}\left(\frac{|S|}{(\epsilon_u-\epsilon_l)^2}\log\frac{1}{\delta}\right).\end{aligned}$$

Hence the Lemma follows. □

For better understanding, we further divide high confidence values into two ranges: 1) medium ($\frac{1}{\log|S|} \geq \delta \geq 1/|S|^{1/3}$) where one sequence of NEAR-OPT-MAX, PRUNE, SOFT-SEQ-ELIM is enough to produce output and 2) high ($\delta \geq \frac{1}{\log|S|}$) where NEAR-OPT-MAX and multiple sequences of PRUNE, SOFT-SEQ-ELIM are used to produce the final output.

C.8.2. OPT-MAX-MEDIUM

OPT-MAX-MEDIUM takes 3 parameters: input set S, bias ϵ and confidence δ such that $\frac{1}{\log|S|} \geq \delta \geq 1/|S|^{1/3}$. W.p.$\geq 1-\delta$, OPT-MAX-MEDIUM$(S, \epsilon, \delta)$ uses $\mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$ comparisons and outputs an ϵ-maximum of S..

OPT-MAX-MEDIUM picks a good anchor element and prunes the input set using this anchor element and use SOFT-SEQ-ELIM with the anchor on pruned set to output ϵ-maximum.

To pick the good anchor element it calls NEAR-OPT-MAX to find an $\epsilon/3$-maximum of a random set of size $|S|/(\log|S|)^2$. We later show that this anchor is $\left(\epsilon/3, (\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element. Notice that elements for which anchor is not $\epsilon/3$-preferable is $\mathcal{O}(1/\delta^3)$. Hence using PRUNE with this anchor, we manage to prune set S to much smaller set Q'. Hence SOFT-SEQ-ELIM with this anchor on pruned set takes very few comparisons.

Algorithm 11 OPT-MAX-MEDIUM

```
1: inputs
2:     Set $S$, bias $\epsilon$, confidence $\delta$
3: Form a set $Q$ by selecting $|S|/(\log|S|)^2$ random elements in $S$
4: $a \leftarrow$ NEAR-OPT-MAX$(Q, \epsilon/3, \delta/4)$
5: $Q' \leftarrow$ PRUNE$\left(Q, a, \frac{\epsilon}{3}, \frac{2\epsilon}{3}, \frac{\delta}{4(\log|S|)^2\log(4|S|/\delta)}\right)$
6: return SOFT-SEQ-ELIM$(Q', a, \epsilon/3, \epsilon, \delta/4)$
```

In the below Lemma, we bound comparisons used by OPT-MAX-MEDIUM and prove its correctness.

Lemma 19. *For $\frac{1}{\log|S|} \ge \delta \ge \frac{1}{|S|^{1/3}}$, w.p.$\ge 1-\delta$, OPT-MAX-MEDIUM$(S,\epsilon,\delta)$ uses $\mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$ comparisons and outputs an ϵ-maximum of S.*

Proof. Since Q has $|S|/(\log|S|)^2$ elements, by Lemma 6, NEAR-OPT-MAX$(Q,\epsilon/3,\delta/4)$ uses

$$\mathcal{O}\left(\frac{|Q|}{\epsilon^2}\left(\log\frac{|Q|}{\delta}\right)^2\right) = \mathcal{O}\left(\frac{|S|}{\epsilon^2(\log|S|)^2}\left(\log\frac{|S|}{\delta}\right)^2\right) = \mathcal{O}\left(\frac{|S|}{\epsilon^2}\right)$$

comparisons (where last equality is because $\delta \ge \frac{1}{|S|^{1/3}}$) and w.p.$\ge 1-\delta/3$ outputs a, an $\epsilon/3$-maximum of Q.

By Lemma 5, w.p.$\ge 1-\delta/4$, an $\epsilon/3$-maximum of Q is an $\left(\frac{\epsilon}{3}, \frac{|S|}{|Q|}\log\frac{4|S|}{\delta}\right)$-*good anchor* element and hence a is an $\left(\frac{\epsilon}{3}, (\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element.

Let $\delta' = \frac{\delta}{4(\log|S|)^2\log(4|S|/\delta)}$. Notice that $\delta' \ge \frac{\delta}{16(\log|S|)^3}$. Since $(\log|S|)^2\log\frac{4|S|}{\delta} \le \frac{2}{\delta'} < \frac{2\log(2|S|/\delta')}{\delta'}$, by Lemma 18, w.p.$\ge 1-\delta/4$, PRUNE$(S,a,\epsilon/3,2\epsilon/3,\delta'/4)$ uses

$$\mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta'}\right) = \mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{\log|S|}{\delta}\right) = \mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$$

comparisons (where last equality is because $\delta \le \frac{1}{\log|S|}$) and outputs a set Q' of size

$$\le 4\frac{\log\frac{2|S|}{\delta'}}{\delta'} = \mathcal{O}\left(\frac{(\log|S|)^4}{\delta}\right)$$

(where equality is because $\delta' \ge \frac{\delta}{16(\log|S|)^3}$ and $\delta \ge \frac{1}{|S|^{1/3}}$) s.t. Q' contains all elements in S for which a is not $2\epsilon/3$-preferable i.e., $\tilde{p}_{e,a} \le 2\epsilon/3\ \forall e \in S\setminus Q'$.

W.p.$\ge 1-\delta/4$, SOFT-SEQ-ELIM$(Q',a,2\epsilon/3,\epsilon,\delta/4)$ uses

$$\mathcal{O}\left(\frac{|Q'|^2}{\epsilon^2}\log\frac{|Q'|}{\delta}\right) = \mathcal{O}\left(\frac{(\log|S|)^8}{\delta^2\epsilon^2}\log\frac{\log|S|}{\delta}\right) = \mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$$

comparisons (where last equality is because $\frac{1}{\log|S|} \ge \delta \ge \frac{1}{|S|^{1/3}}$) and outputs r^o, an ϵ-maximum of Q' s.t., either $r^o = a$ or $\tilde{p}_{r^o,a} \ge \frac{2\epsilon}{3}$. Since $\tilde{p}_{e,a} \le \frac{2\epsilon}{3}\ \forall e \in S\setminus Q'$, by MST and Lemma 16, $\tilde{p}_{e,r^o} \le \frac{2\epsilon}{3}\ \forall e \in S\setminus Q'$. Hence r^o is an ϵ-maximum of S.

Lemma follows by union bound and noting that comparisons used for each step is $\mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$. □

As mentioned before for high ranges of confidence $\delta \ge 1/\log|S|$, our algorithm OPT-MAX-HIGH first uses NEAR-OPT-MAX over a set of size $|S|/(\log|S|)^2$ and then uses multiple rounds of PRUNE and SOFT-SEQ-ELIM before increasing size to $|S|$. After each round of PRUNE and SOFT-SEQ-ELIM, OPT-MAX-HIGH increases set size by a fraction of $1/\delta$.

C.8.3. OPT-MAX-HIGH

OPT-MAX-HIGH takes 3 parameters: input set S, bias ϵ and confidence δ such that $\delta \ge \frac{1}{\log|S|}$. W.p.$\ge 1-\delta$, OPT-MAX-HIGH$(S,\epsilon,\delta)$ uses $\mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$ comparisons and outputs an ϵ-maximum of S.

Similar to OPT-MAX-MEDIUM, to pick the initial anchor OPT-MAX-HIGH finds an $\epsilon/3$-maximum of a random set of $|S|/(\log|S|)^2$ elements. We later show that this anchor is an $(\epsilon/3, (\log|S|)^2\log(4|S|/\delta))$- *good anchor* element.

Notice that number of elements for which anchor is not $\epsilon/3$-preferable could be much higher than $1/\delta^i$ for constant δ and any constant i. Hence a single round of pruning might not ensure that all such elements will survive. Hence we prune in stages improving anchor element in each stage using SOFT-SEQ-ELIM over pruned set in previous stage. After each stage, we improve anchor element and make sure that number of elements for which current anchor is not ϵ-preferable decreases.

In each stage we increase set size by a fraction of $1/\delta$ and prune the set using current anchor and use SOFT-SEQ-ELIM over pruned set with current anchor to find next anchor. Notice that initially set size is small and hence we can afford to repeat comparisons more times and thereby incur less bias error and confidence error in initial rounds. This makes sure that total bias and confidence error are less than required.

Algorithm 12 OPT-MAX-HIGH

```
1: inputs
2:     Set $S$, bias $\epsilon$, confidence $\delta$
3: Form a set $Q$ by selecting $|S|/(\log|S|)^2$ random elements in $S$ without replacement
4: $a_1 \leftarrow$ NEAR-OPT-MAX$(Q, \epsilon/3, \delta/4)$
5: $m = \lceil 2\log_{1/\delta}\log|S|\rceil$
6: for $i$ from 1 to $m$ do
7:     $n'_i \leftarrow \max(|S|/(\log|S|)^2, |S|\delta^{m-i})$
8:     $\epsilon'_i \leftarrow \epsilon/3 + \frac{2\epsilon/3}{2^{(m-i)/3}}$
9:     $\epsilon''_i \leftarrow \epsilon/3 + \frac{2\epsilon/3}{2^{(m-i+1)/3}}$
10:    $\delta'_i \leftarrow \delta^{m-i+4}$
11:    Form a set $Q_i$ by selecting $n'_i$ random elements in $S$ without replacement
12:    $Q'_i \leftarrow$ PRUNE$(Q_i, a_i, \epsilon''_i, (\epsilon''_i+\epsilon'_i)/2, \delta'^5_i/3)$
13:    $a_{i+1} \leftarrow$ SOFT-SEQ-ELIM$(Q'_i, a_i, \epsilon''_i, \epsilon'_i, \delta'_i/3)$
14: end for
15: return $a_{m+1}$
```

In the below Lemma, we bound comparisons used by OPT-MAX-HIGH and prove its correctness.

Lemma 20. *For $\delta \ge \frac{1}{\log|S|}$, w.p.$\ge 1-\delta$, OPT-MAX-HIGH$(S,\epsilon,\delta)$ uses $\mathcal{O}\left(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\right)$ comparisons and outputs an ϵ-maximum of S.*

Proof. We first bound the comparisons used and confidence error accrued in base step 4 NEAR-OPT-MAX$(Q,\epsilon/3,\delta/4)$. By Lemma 6, NEAR-OPT-MAX$(Q,\epsilon/3,\delta/4)$ uses

$$\mathcal{O}\left(\frac{|Q|}{\epsilon^2}\left(\log\frac{|Q|}{\delta}\right)^2\right) = \mathcal{O}\left(\frac{|S|}{\epsilon^2(\log|S|)^2}\left(\log\frac{|S|}{\delta}\right)^2\right) = \mathcal{O}\left(\frac{|S|}{\epsilon^2}\right)$$

comparisons (where last equality is because $\delta \ge \frac{1}{|S|^{1/3}}$) and w.p.$\ge 1-\delta/4$ outputs a_1, an $\epsilon/3$-maximum of Q. Further by Lemma 5, w.p.$\ge 1-\delta/4$, , a_1 is an $\left(\epsilon/3, (\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element of S.

Hence by union bound, step 4 uses $\mathcal{O}\left(\frac{|S|}{\epsilon^2}\right)$ comparisons and w.p.$\ge 1-\delta/2$, outputs an $\left(\epsilon/3, (\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element of S.

From now we bound the comparisons used and confidence error incurred by steps 12, 13 in m recursions. We bound these quantities for a single recursion step i assuming that it got a correct answer from recursion step $i-1$.

Notice that at recursion step i, $n'_i = \max(|S|/(\log|S|)^2, |S|\delta^{m-i})$, $\epsilon'_i = \epsilon/3 + \frac{2\epsilon/3}{2^{(m-i)/3}}$, $\epsilon''_i = \epsilon/3 + \frac{2\epsilon/3}{2^{(m-i+1)/3}}$ and $\delta'_i = \delta^{m-i+4}$.

Assume that anchor element a_i at the start of recursion i is an $(\epsilon''_i, 1/\delta'^4_i)$-*good anchor* element of S. Notice that assumption for recursion step 1 is true with probability $1-\delta/2$, since with same probability, a_1 is an $\left(\epsilon/3, (\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element of S. It is easy to check that $\epsilon''_1 > \epsilon/3$ and $1/\delta'^4_1 \ge (\log|S|)^2\log\frac{4|S|}{\delta}$. Hence a_1 is also an $(\epsilon_1, 1/\delta'^4_1)$-*good anchor* element.

Now we will show that w.p.$\ge 1-\delta'_i$, recursion i uses $\mathcal{O}\left(\frac{n'_i}{(\epsilon'_i-\epsilon''_i)^2}\log\frac{1}{\delta'_i}\right)$ comparisons and outputs an $(\epsilon''_{i+1}, 1/\delta'^4_{i+1})$-*good anchor* element of S which will serve as an assumption for next recursion and that output is an ϵ'_i-maximum of set Q_i picked in step 11 which will show that at end of mth recursion we have an ϵ-maximum of S.

Let T_i be the set of all elements in S for which a_i is not ϵ''_i-preferable. Further add anchor element a_i to T_i i.e., $T_i = T_i\bigcup\{a_i\}$.

We now show that for any element e in T_i which is not $(\epsilon'_i, 1/\delta'^4_{i+1})$-*good anchor* element, an element which is not ϵ'_i-preferable is present in Q_i and thereby later show that such an element won't be an output of ith recursion.

Notice that by assumption, $|T_i| \le 1/\delta'^4_i + 1$. Further let T'_i be the set of all elements in T_i which are not $(\epsilon'_i, 1/\delta'^4_{i+1})$-*good anchor* elements. Observe that $|T'_i| \le |T_i| \le 1/\delta^4_i + 1$.

Since in recursion i we pick set Q_i by picking n'_i elements randomly from S, each element in S is part of Q_i with probability $n'_i/|S|$.

For an element e in T'_i, probability that Q_i does not contain an element for which e is not ϵ'_i-preferable is

$$\le \left(1 - \frac{1/\delta'^4_{i+1}}{|S|}\right)^{n'_i} \le \left(1 - \frac{(\delta/\delta'_i)^4}{|S|}\right)^{|S|\delta'_i/\delta^4} \le e^{-\frac{1}{\delta'^3_i}}$$

Hence by union bound, probability that for any element e in T'_i, Q_i does not contain an element for which e is not ϵ'_i-preferable is

$$\le |T'_i|e^{-\frac{1}{\delta'^3_i}} \le \left(\frac{1}{\delta'^4_i}+1\right)e^{-\frac{1}{\delta'^3_i}} \le \delta'_i/3.$$

Now we bound comparisons used and confidence error incurred in step 12 during ith recursion.

Since a_i is an $(\epsilon_i'', 1/\delta_i'^4)$-*good anchor* element and $1/\delta_i'^4 < 3/\delta_i'^5 < 2^{\frac{2\log(6|Q_i|/\delta_i'^5)}{\delta_i^5/3}}$, by Lemma 18, PRUNE$(Q_i, a_i, \epsilon_i'', (\epsilon_i'' + \epsilon_i')/2, \delta_i'^5/3)$ uses

$$\mathcal{O}\bigg(\frac{|Q|}{(\epsilon_i' - \epsilon_i'')^2}\log\frac{1}{\delta_i'}\bigg) = \mathcal{O}\bigg(\frac{n_i'}{(\epsilon_i' - \epsilon_i'')^2}\log\frac{1}{\delta_i'}\bigg)$$

comparisons and outputs a set Q_i' of size $\frac{12\log(6|Q_i|/\delta_i'^5)}{\delta_i'^5} = \mathcal{O}\bigg(\frac{\sqrt{|Q_i|}}{\log|Q_i|}\bigg)$ (since $1/\delta_i' = \mathcal{O}((\log|Q_i|)^5)$) and contains all elements in Q_i for which a_i is not an $(\epsilon_i' + \epsilon_i'')/2$-preferable i.e., $\tilde{p}_{e,a_i} \le (\epsilon_i'' + \epsilon_i')/2 \;\forall e \in Q_i \setminus Q_i'$.

Now we bound comparisons used and confidence error incurred during step 13.

By Corollary 4, w.p.$\ge 1 - \delta_i'/3$, SOFT-SEQ-ELIM$(Q_i', a_i, \epsilon_i'', \epsilon_i', \delta_i'/3)$ uses

$$\mathcal{O}\bigg(\frac{|Q_i'|^2}{(\epsilon_i'' - \epsilon_i')^2}\log\frac{|Q_i'|}{\delta_i'}\bigg) = \mathcal{O}\bigg(\frac{|Q_i|/\log|Q_i|}{(\epsilon_i' - \epsilon_i'')^2}\log\frac{|Q_i|}{\delta_i'}\bigg) = \mathcal{O}\bigg(\frac{n_i'}{(\epsilon_i' - \epsilon_i'')^2}\log\frac{1}{\delta_i'}\bigg)$$

comparisons and outputs a_{i+1}, an ϵ_i'-maximum of Q_i' s.t., either $a_{i+1} = a_i$ or $\tilde{p}_{a_{i+1},a_i} > \frac{\epsilon_i'' + \epsilon_i'}{2}$. Since $\tilde{p}_{e,a_i} \le (\epsilon_i'' + \epsilon_i')/2 \;\forall e \in Q_i \setminus Q_i'$, by MST and Lemma 16,$\tilde{p}_{e,a_{i+1}} \le (\epsilon_i'' + \epsilon_i')/2 \;\forall e \in Q_i \setminus Q_i'$. Hence a_{i+1} is an ϵ_i'-maximum of Q_i.

Further notice that since either $a_{i+1} = a_i$ or $\tilde{p}_{a_{i+1},a_i} > (\epsilon_i' + \epsilon_i'')/2 > \epsilon_i''$, $a_{i+1} \in T_i$. Since for every $e \in T_i'$, Q_i contains an element for which e is not ϵ_i'-preferable, if $a_{i+1} \in T_i'$, it violates that a_{i+1} is an ϵ_i'-maximum of Q_i. Hence $a_{i+1} \notin T_i'$ and hence a_{i+1} is also an $(\epsilon_i', (\delta/\delta_i')^4)$-*good anchor* element of S which is assumption for next recursion $i+1$.

Hence by union bound, w.p.$\ge 1 - \delta_i$, recursion i uses $\mathcal{O}\Big(\frac{n_i'}{(\epsilon_i' - \epsilon_i'')^2}\log\frac{1}{\delta'}\Big)$ comparisons and outputs an $(\epsilon_i', 1/\delta_{i+1}'^4)$-*good anchor* element of S, which is also an ϵ_i'-maximum of Q_i.

Hence by union bound, total error incurred over all recursion steps and base case is

$$\le \frac{\delta}{4} + \sum_{i=1}^{m}\delta_i' \le \frac{\delta}{4} + \sum_{i=1}^{m}\delta^{m-i+4} \le \frac{\delta}{4} + \frac{\delta}{2} < \delta.$$

Hence w.p.$\ge 1 - \delta$, after m recursions, a_{m+1} is an ϵ-maximum of S.

Total number of comparisons used over all recursion steps and base case is

$$\begin{aligned}
&= \mathcal{O}\bigg(\frac{|S|}{\epsilon^2}\bigg) + \sum_{i=1}^{m}\mathcal{O}\bigg(\frac{n_i'}{(\epsilon_i' - \epsilon_i'')^2}\log\frac{1}{\delta_i'}\bigg)\\
&= \sum_{i=1}^{m}\mathcal{O}\bigg(\frac{n_i' 2^{2(m-i)/3}}{\epsilon^2}\log\frac{1}{\delta_i'}\bigg)\\
&= \sum_{i=1}^{m}\mathcal{O}\bigg(\frac{|S|\delta^{m-i}2^{2(m-i)/3}}{\epsilon^2}\log\frac{1}{\delta_i'}\bigg)\\
&\overset{(a)}{=} \sum_{i=1}^{m}\mathcal{O}\bigg(\frac{|S|}{2^{(m-i)/3}\epsilon^2}\log\frac{1}{\delta_i'}\bigg)\\
&= \sum_{i=1}^{m}\mathcal{O}\bigg(\frac{|S|}{2^{(m-i)/3}\epsilon^2}\log\frac{1}{\delta^{m-i+4}}\bigg)\\
&= \mathcal{O}\bigg(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\bigg)\sum_{i=1}^{m}\frac{m-i+4}{2^{(m-i)/3}}\\
&= \mathcal{O}\bigg(\frac{|S|}{\epsilon^2}\log\frac{1}{\delta}\bigg)
\end{aligned}$$

where (a) is because $\delta \le 1/2$ and last equality because $x^i(i+4)$ series is convergent for $x \le 1/2^{1/3}$. Hence proved. □

D. Ranking

D.1. Outline of BINARY-SEARCH-RANKING and how to improve

(Falahatgar et al., 2017b) first proposes a ranking algorithm MERGE-RANK that uses $\mathcal{O}\Big(\frac{|S|(\log|S|)^3}{\epsilon^2}\log\frac{|S|}{\delta}\Big)$ comparisons and w.p.$\ge 1 - \delta$ outputs an ϵ-ranking of S. They use this algorithm as a building block for the final algorithm BINARY-SEARCH-RANKING. We outline BINARY-SEARCH-RANKING algorithm and mention components that lead to additional $(\log\log n)^3$ factor. We also propose modified components that remove additional $(\log\log n)^3$ factor.

BINARY-SEARCH-RANKING first selects $\frac{|S|}{(\log|S|)^3}$ random elements in S as anchors and ranks them using MERGE-RANK. Notice that this step uses $\mathcal{O}\Big(\frac{|S|\log|S|}{\epsilon^2}\Big)$ comparisons.

Then BINARY-SEARCH-RANKING forms bins between two consecutively ranked anchors. For each element e BINARY-SEARCH-RANKING finds which bin it belongs to using INTERVAL-BINARY-SEARCH. INTERVAL-BINARY-SEARCH does a noisy random walk over the bins to determine which bin element e belongs to. Noisy random walk is run for $\mathcal{O}(\log|S|)$ steps where in each step $\mathcal{O}(\frac{1}{\epsilon^2})$ comparisons are used. Notice that this random walk step uses $\mathcal{O}(\frac{\log|S|}{\epsilon^2})$ comparisons for each element e and hence $\mathcal{O}(\frac{|S|\log|S|}{\epsilon^2})$ comparisons over all elements. This noisy random walk can sometimes fail if element e visits a close anchor a, i.e., $|\tilde{p}_{e,a}| \le k\epsilon$ for some constant $k < 1$.

Then INTERVAL-BINARY-SEARCH uses BINARY-SEARCH over visited anchors to find closeby anchor. Notice that number of visited anchors during noisy random walk is $\mathcal{O}(\log |S|)$. BINARY-SEARCH does a binary search over ranked set of visited anchors and hence runs for $\mathcal{O}(\log\log |S|)$ steps. It uses $\mathcal{O}\left(\frac{\log |S|}{\epsilon^2}\right)$ comparisons over each step to ensure that error probability during is $1/|S|^9$ for each element. Hence BINARY-SEARCH uses $\mathcal{O}(\frac{\log |S| (\log\log |S|)}{\epsilon^2})$ comparisons for each element and therefore uses $\mathcal{O}(\frac{|S| \log |S| (\log\log |S|)}{\epsilon^2})$ comparisons over all elements incurring an overhead factor of $\log\log |S|$.

We fix this step BINARY-SEARCH by not using $\mathcal{O}(\log |S|)$ comparisons over each step but instead $\mathcal{O}(\log\log |S|)$ comparisons over each steps. Notice that this increases error probability to $1/\log |S|$ for each element in place of $1/|S|^9$. Hence we check if in fact BINARY-SEARCH found a closeby anchor by comparing with the output anchor element for $\mathcal{O}(\log |S|)$ times which will give a wrong decision only w.p. $\le 1/|S|^{10}$. If we find that output anchor element is indeed closeby anchor then we output it or else this time we do one more round of binary search repeating each comparison $\mathcal{O}(\log |S|)$ times thereby ensuring that final output is correct w.p.$\ge 1 - \frac{1}{|S|^9}$.

Notice that even though worst case complexity for each element is same as before, most of the elements use much fewer comparisons. Since first round of binary search is correct w.p.$\ge 1 - 1/\log |S|$, much fewer than $10/\log |S|$ fraction of elements fail to find closeby anchor during first round. Hence less than $10/\log |S|$ fraction of total elements use second round and hence total comparisons are $\mathcal{O}(\frac{|S| \log |S|}{\epsilon^2})$ comparisons.

Later for each bin, BINARY-SEARCH-RANKING finds if elements are close to bin's boundary anchor elements by repeating comparisons $\mathcal{O}(\log |S|)$ times and if an element is close to boundary anchor it ranks that element close to boundary anchor. Notice that this step uses $\mathcal{O}(\frac{\log |S|}{\epsilon^2})$ comparisons for each element and hence $\mathcal{O}(\frac{|S| \log |S|}{\epsilon^2})$ comparisons over all elements. Therefore this step is not a problem.

It can be shown that after removing elements which are close to bin boundaries each bin has $\mathcal{O}((\log |S|)^4)$ elements.

Then BINARY-SEARCH-RANKING uses MARGE-RANK to rank each bin with error probability $1/|S|^3$ for each bin. Hence it uses $\mathcal{O}(\frac{|B_i| (\log |B_i|)^3}{\epsilon^2} \log \frac{|B_i|}{1/|S|^3}) = \mathcal{O}(\frac{|B_i| (\log |B_i|)^3}{\epsilon^2} \log |S|) = \mathcal{O}(\frac{|B_i| (\log\log |S|)^3}{\epsilon^2} \log |S|)$ comparisons (since $|B_i| = \mathcal{O}((\log n)^4)$) for each bin B_i. When we sum over all bins B_i, total comparisons is $\mathcal{O}(\frac{|S| \log |S| (\log\log |S|)^3}{\epsilon^2})$ which has an overhead factor of $(\log\log |S|)^3$.

We improve this step by not using error probability of $1/|S|^3$ but using $1/\log |S|$ for error probability. This ensures that comparisons are bounded but overall error probability is high. So we check if each bin is ranked correctly. Our main contribution is an algorithm to find an ordered set is ϵ-ranked. Using this algorithm we find if each bin is correctly ranked by making sure that we know right answer w.p.$\ge 1 - \frac{1}{|S|^5}$. Checking over all bins with this algorithm uses only $\mathcal{O}(\frac{|S| \log |S|}{\epsilon^2})$ comparisons. If a bin is not correctly ranked, then we rank that bin again this time with error probability of $1/|S|^4$.

Observe that in worst case, ranking a bin uses same comparisons as previously but only few bins need these many comparisons since most of the bins are ranked correctly in first round. Notice that since first round of ranking bin is correct with probability $\ge 1 - \frac{1}{\log |S|}$, $< 10/\log |S|$ fraction of bins need second round of ranking which helps us in bounding comparisons.

We first present new BINARY-SEARCH that has low overall sample complexity than the one in (Falahatgar et al., 2017b). We prove Lemmas similar to those in (Falahatgar et al., 2017b) with lower complexities.

D.2. BINARY-SEARCH

BINARY-SEARCH uses a subroutine BUD-BINARY-SEARCH that takes a confidence parameter δ and outputs desired result w.p.$\ge 1 - \delta$.

D.2.1. BUD-BINARY-SEARCH

As outlined previously, BUD-BINARY-SEARCH does a simple binary search using $\mathcal{O}\left(\frac{1}{\epsilon^2} \log \frac{\log |Q|}{\delta}\right)$ comparisons (where Q is a set over which binary search is performed) for each step.

Algorithm 13 BUD-BINARY-SEARCH

```
1: inputs
2:    Ordered array $S$, ordered array $Q$, search item $e$, bias $\epsilon$, confidence $\delta$
3: $l \leftarrow 1, h \leftarrow |Q|$.
4: while $h - l > 0$ do
5:    Compare $e$ and $S(Q(\lceil \frac{l+h}{2} \rceil))$ for $\frac{4}{\epsilon^2} \log \frac{10 \log |Q|}{\delta}$ times.
6:    $t \leftarrow$ fraction of times $e$ won.
7:    if $t \in \left[\frac{1}{2} - 3\epsilon, \frac{1}{2} + 3\epsilon\right]$ then
8:       return $Q(\lceil \frac{l+h}{2} \rceil)$.
9:    else if $t < \frac{1}{2} - 3\epsilon$ then
10:      $h = \lceil \frac{l+h}{2} \rceil$
11:   else
12:      $l = \lceil \frac{l+h}{2} \rceil$
13:   end if
14: end while
15: return $Q(h)$.
```

Lemma 21. *For $\epsilon'' > \epsilon'$, consider ordered sets S, Q s.t. $\tilde{p}_{S(Q(i)),S(Q(j))} \le \epsilon' \forall i < j$. For an element e s.t., $\exists g : |\tilde{p}_{S(Q(g)),a}| < 2\epsilon''$, BINARY-SEARCH$(S, Q, a, \epsilon'', \delta)$ uses $\mathcal{O}(\frac{\log |Q|}{\epsilon''^2} \log \frac{\log |Q|}{\delta})$ comparisons and w.p.$\ge 1 - \delta$ returns y s.t. $|\tilde{p}_{S(Q(y)),a}| < 4\epsilon''$.*

Proof. This proof is similar to Lemma 23 in (Falahatgar et al., 2017b)

Notice that this is a binary search over ordered set Q using $\mathcal{O}(\frac{1}{\epsilon''^2} \log \frac{\log |Q|}{\delta})$ comparisons for each step and hence bound on comparisons follow.

At any stage of BUD-BINARY-SEARCH, there are three possibilities that can happen . Consider the case when we are comparing e with $S(Q(i))$.

1. $|\tilde{p}_{S(Q(i)),e}| < 2\epsilon''$. Probability that the fraction of wins for e is not between $\frac{1}{2} - 3\epsilon''$ and $\frac{1}{2} + 3\epsilon''$ is less than $e^{-\frac{\log(4|Q|/\delta)}{\epsilon''^2}\epsilon''^2} \le \frac{\delta}{4 \log |Q|}$. Hence BUD-BINARY-SEARCH outputs $Q(i)$.

2. $\tilde{p}_{S(Q(i)),e} > 2\epsilon''$. Probability that the fraction of wins for e is more than $\frac{1}{2}$ is less than $e^{-\frac{\log(4|Q|/\delta)}{\epsilon''^2}\epsilon''^2} \le \frac{\delta}{4 \log |Q|}$. So BUD-BINARY-SEARCH will not move right. Also notice that $\tilde{p}_{S(Q(j)),e} > 2\epsilon'' - \epsilon' > \epsilon'' \; \forall j > i$.

3. $\tilde{p}_{S(Q(i)),e} > 4\epsilon''$. Probability that the fraction of wins for e is more than $\frac{1}{2} - 3\epsilon''$ is less than $e^{-\frac{\log(4|Q|/\delta)}{\epsilon''^2}\epsilon''^2} \le \frac{\delta}{4 \log |Q|}$. Hence BUD-BINARY-SEARCH will move left. Also notice that $\tilde{p}_{S(Q(j)),e} > 4\epsilon'' - \epsilon' > \epsilon'' \; \forall j > i$.

We can show similar results for $\tilde{p}_{S(Q(i)),e} < -2\epsilon''$ and $\tilde{p}_{S(Q(i)),e} < -4\epsilon''$. Hence if $|\tilde{p}_{S(Q(i)),e})| < 2\epsilon''$ then BUD-BINARY-SEARCH outputs $Q(i)$, and if $2\epsilon'' < |\tilde{p}_{S(Q(i)),e}| < 4\epsilon''$ then either BUD-BINARY-SEARCH outputs $Q(i)$ or moves in the correct direction and if $|\tilde{p}_{S(Q(i)),e}| > 4\epsilon''$, then BUD-BINARY-SEARCH moves in the correct direction.

Proof follows by noting that that binary search visits $\log |Q|$ indices and using union bound. □

D.2.2. BINARY-SEARCH

BINARY-SEARCH initially calls BUD-BINARY-SEARCH with confidence parameter of $1/(4 \log |S|)$ and then it checks if BUD-BINARY-SEARCH gave a required result by comparing with output for $\mathcal{O}(\frac{\log |S|}{\epsilon^2})$ times. If it deems that output is not good then it calls BUD-BINARY-SEARCH this time with confidence parameter $1/|S|^{10}$. Notice that if BINARY-SEARCH is called $|S|$ independent times then less than $10/\log |S|$ fraction of times second BUD-BINARY-SEARCH is called since first BUD-BINARY-SEARCH gives required answer w.p.$\ge 1 - 1/(4 \log |S|)$.

Algorithm 14 BINARY-SEARCH

```
1: inputs
2:    Ordered Set $S$, ordered array $Q$, search item $e$, bias $\epsilon$
3: $l \leftarrow$ BUD-BINARY-SEARCH$(S, Q, e, \epsilon, 1/4 \log |S|)$
4: Compare $e$ and $S(l)$ for $\frac{20 \log |S|}{\epsilon^2}$ times. $t \leftarrow$ fraction of times $e$ won
5: if $t \in [1/2 - 5\epsilon, 1/2 + 5\epsilon]$ then
6:    return $l$
7: else
8:    return BUD-BINARY-SEARCH$(S, Q, e, \epsilon, 1/|S|^{10})$
9: end if
```

Lemma 22. *For $\epsilon'' > \epsilon'$, consider ordered sets S, Q s.t. $\tilde{p}_{S(Q(i)),S(Q(j))} \le \epsilon' \forall i < j$ and $|Q| \le 90 \log |S|$. For an element e s.t., $\exists g : |\tilde{p}_{S(Q(g)),a}| < 2\epsilon''$, BINARY-SEARCH$(S, Q, a, \epsilon'', \delta)$ w.p. $\ge 1 - 1/|S|^9$ returns y s.t. $|\tilde{p}_{S(Q(y)),a}| < 6\epsilon''$ and w.p. $\ge 1 - 1/(3 \log |S|)$ uses $\mathcal{O}\left(\frac{\log |S|}{\epsilon^2}\right)$ comparisons and always uses $\mathcal{O}\left(\frac{(\log |S|)(\log \log |S|)}{\epsilon^2}\right)$ comparisons. If repeated for $|S|$ independent copies of Q and e, w.p.$\ge 1 - 1/|S|^9$, BINARY-SEARCH uses $\mathcal{O}\left(\frac{|S| \log |S|}{\epsilon^2}\right)$ comparisons.*

Proof. BUD-BINARY-SEARCH$(S, Q, e, \epsilon, 1/(4 \log |S|))$ uses $\mathcal{O}(\frac{\log \log |S|}{\epsilon^2} \log \log |S|)$ comparisons and w.p.$\ge 1 - 1/(4 \log |S|)$ outputs y s.t. $|\tilde{p}_{S(Q(y)),a}| < 4\epsilon''$.

If $|\tilde{p}_{S(Q(y)),e}| < 4\epsilon''$, probability that the fraction of wins for e is not between $1/2 - 5\epsilon''$ and $1/2 + 5\epsilon''$ is less than $e^{-\frac{20 \log |S|}{\epsilon''^2}\epsilon''^2} \le \frac{1}{|S|^{20}}$.

If first BUD-BINARY-SEARCH fails that is if $|\tilde{p}_{S(Q(y)),e}| > 6\epsilon''$, probability that the fraction of wins for e is between $1/2 - 5\epsilon''$ and $1/2 + 5\epsilon''$ is less than $e^{-\frac{20\log|S|}{\epsilon''^2}\epsilon''^2} \le \frac{1}{|S|^{20}}$ and hence second BUD-BINARY-SEARCH is invoked.

BUD-BINARY-SEARCH$(S, Q, e, \epsilon, 1/|S|^{10})$ uses $\mathcal{O}(\frac{\log\log|S|}{\epsilon^2}\log|S|)$ comparisons and w.p.$\ge 1 - \frac{1}{|S|^{10}}$ outputs y s.t. $|\tilde{p}_{S(Q(y)),a}| < 4\epsilon''$.

So by union bound, w.p.$\ge 1 - 1/|S|^9$, BINARY-SEARCH outputs required answer and w.p.$\ge 1 - \frac{1}{3\log|S|}$ uses $\mathcal{O}(\frac{|S|\log|S|}{\epsilon^2})$ comparisons and always uses $\mathcal{O}(\frac{(\log|S|)(\log\log|S|)}{\epsilon''^2})$ comparisons.

If we repeat this for $|S|$ independent copies, then probability that second BUD-BINARY-SEARCH used for more than $20/|\log|S|$ is less than $e^{-|S|D(\frac{10}{\log|S|}||\frac{1}{3\log|S|})} \le e^{-10|S|/\log|S|} \le 1/|S|^9$. Hence the bound on comparisons follows. □

We now present RANK-CHECK that checks if an ordered set is ϵ-ranked or not 3ϵ-ranked.

D.3. RANK-CHECK

RANK-CHECK takes three parameters: ordered set S, bias ϵ and confidence δ. RANK-CHECK deems if S is ϵ-ranked i.e., $\tilde{p}_{S(i),S(j)} \le \epsilon\ \forall i < j$ or if S is not 3ϵ-ranked i.e., there exists $i < j$ s.t. $\tilde{p}_{S(i),S(j)} > 3\epsilon$. RANK-CHECK$(S, \epsilon, \delta)$ uses $\mathcal{O}\Big(\frac{|S|}{\epsilon^2}\log\frac{|S|}{\delta}\Big)$ comparisons and w.p.$\ge 1-\delta$, outputs a correct decision.

RANK-CHECK first sets anchor a as $S(1)$ and iterates over the set S checking if $\tilde{p}_{S(i),a} > \epsilon$ vs $\tilde{p}_{S(i),a} < 0$ using COMPARE2. If COMPARE2 deems $\tilde{p}_{S(i),a} > \epsilon$, then RANK-CHECK updates current anchor to $S(i)$. If COMPARE2 deems $\tilde{p}_{S(i),a} < 0$, then RANK-CHECK checks if $\tilde{p}_{a,S(i)} < \epsilon$ vs $\tilde{p}_{a,S(i)} > 2\epsilon$ again using COMPARE2. If COMPARE2 deems $\tilde{p}_{a,S(i)} > 2\epsilon$, then $S(i)$ is much worse than a which is to the left of $S(i)$ in S and hence S is not ϵ-ranked and RANK-CHECK returns FALSE.

Due to probabilistic nature if S is neither ϵ-ranked nor not 3ϵ-ranked then it can either of the two outputs.

Algorithm 15 RANK-CHECK
```
1: inputs
2:     Ordered Set $S$, bias $\epsilon$, confidence $\delta$
3: $a \leftarrow S(1)$
4: for $i$ from 2 to $|S|$ do
5:     if COMPARE2$(S(i), a, 0, \epsilon, \frac{\delta}{2|S|)}) = 2$ then
6:         $a \leftarrow S(i)$
7:     else if COMPARE2$(a, S(i), \epsilon, 2\epsilon, \frac{\delta}{2|S|}) = 2$ then
8:         return FALSE
9:     end if
10: end for
11: return TRUE
```

In the below Lemma, we bound comparisons used by RANK-CHECK and prove its correctness.

Lemma 23. *For an ordered set S,* RANK-CHECK(S, ϵ, δ) *uses* $\mathcal{O}\Big(\frac{|S|}{\epsilon^2}\log\frac{|S|}{\delta}\Big)$ *comparisons. If S is an ϵ-ranked set i.e., $\tilde{p}_{S(i),S(j)} \le \epsilon\ \forall i < j$, w.p.$\ge 1-\delta$,* RANK-CHECK$(S, \epsilon, \delta)$ *outputs TRUE. If S is not a 3ϵ-ranked set i.e., $\exists i < j$ s.t. $\tilde{p}_{S(i),S(j)} > 3\epsilon$ then w.p.$\ge 1-\delta$,* RANK-CHECK(S, ϵ, δ) *outputs FALSE.*

Proof. We first bound the comparisons.

Since both COMPARE2$(e, f, 0, \epsilon, \delta/(2|S|))$ and COMPARE2$(e, f, \epsilon, 2\epsilon, \delta/(2|S|))$ use $\mathcal{O}(\frac{1}{\epsilon^2}\log\frac{|S|}{\delta})$ comparisons and each one is called for at most $|S|$ times, total number of comparisons used is $\mathcal{O}\Big(\frac{|S|}{\epsilon^2}\log\frac{|S|}{\delta}\Big)$.

Since each COMPARE2 is called with confidence parameter of $\delta/(2|S|)$ and total number of COMPARE2 calls are less than $2|S|$, by union bound, w.p.$\ge 1-\delta$, all COMPARE2 calls give desired result i.e., COMPARE2$(e, f, \epsilon_l, \epsilon_u, \delta/(2|S|))$ outputs 1 if $\tilde{p}_{e,f} \le \epsilon_l$ and 2 if $\tilde{p}_{e,f} \ge \epsilon_u$.

Let a_j be the anchor element a that is compared with $S(j)$. Notice that $a_2 = S(1)$.

We first consider the case of ϵ-ranked set S. Notice that since anchor element is always left to competing element in the set S i.e., $a_j = S(i)$ for some $i < j$. Since $\tilde{p}_{S(i),S(j)} \le \epsilon$, $\forall i < j$, $\tilde{p}_{a_j,S(j)} \le \epsilon$ and hence COMPARE2$(a_j, S(j), \epsilon, 2\epsilon, \delta/(2|S|))$ outputs 1 for all j. Therefore FALSE is never returned.

Now we consider the second case: S is not an 3ϵ-ranked set. Notice that there exists $i < j$ s.t. $\tilde{p}_{S(i),S(j)} > 3\epsilon$. If RANK-CHECK reaches $S(j)$, then we show that $\tilde{p}_{a_j,S(j)} > 2\epsilon$ thereby proving the required result.

We first show that anchor element only keeps getting better i.e., $\tilde{p}_{a_l,a_k} \ge 0\ \forall l \ge k$. Since COMPARE2$(S(k), a_k, 0, \epsilon, \delta/(2|S|))$ does not output 2 if

$\tilde{p}_{S(k),a_k} < 0$, $a_{k+1} = S(k)$ only if $\tilde{p}_{S(k),a_k} \geq 0$. Hence $\tilde{p}_{a_{k+1},a_k} \geq 0$. Therefore by SST, $\tilde{p}_{a_l,a_k} \geq 0\,\forall l \geq k$.

We now show that $\tilde{p}_{S(k),a_{k+1}} < \epsilon$. Notice that $a_{k+1} = a_k$ if COMPARE2$(S(k), a_k, 0, \epsilon, \delta/(2|S|))$ outputs 1 and hence $\tilde{p}_{S(k),a_k} < \epsilon$. Hence either $a_{k+1} = S(k)$ or $a_{k+1} = a_k$ and $\tilde{p}_{S(k),a_{k+1}} < \epsilon$. Therefore $\tilde{p}_{S(k),a_{k+1}} < \epsilon$.

Since $\tilde{p}_{S(i),a_{i+1}} < \epsilon$ and $\tilde{p}_{S(i),S(j)} > 3\epsilon$, by SST and STI, $\tilde{p}_{a_{i+1},S(j)} > 2\epsilon$. Because of $\tilde{p}_{a_j,a_{i+1}} \geq 0$, by SST, $\tilde{p}_{a_j,S(j)} > 2\epsilon$. Hence COMPARE2$(S(j), a_j, 0, \epsilon, \delta/(2|S|))$ outputs 1 and COMPARE2$(a, S(i), \epsilon, 2\epsilon, \delta/2|S|)$ outputs 2 thereby resulting in FALSE output. □

We now briefly talk about how to use RANK-CHECK to improve sample complexity of ranking bins

D.4. Ranking Bins

Similar to modified BINARY-SEARCH, we first rank each bin using MERGE-RANK with bias parameter of $\epsilon/3$ and confidence parameter of $1/(4\log|S|)$ and check if bin is correctly ranked using RANK-CHECK with confidence parameter of $1/|S|^{10}$. Notice that RANK-CHECK uses only $\mathcal{O}(\frac{|B_i|}{\epsilon^2}\log|S|)$ comparisons for each bin B_i. If RANK-CHECK deems that bin is not correctly ranked then we use a second round of MERGE-RANK with bias parameter of ϵ and confidence parameter of $1/|S|^{10}$.

Notice that since we rank $|S|/(\log|S|)^3$ bins, less than $20/\log|S|$ fraction of them use second round of MERGE-RANK and hence total number of comparisons is $\sum_i \mathcal{O}(\frac{|B_i|}{\epsilon^2}\log|S|) = \mathcal{O}(\frac{|S|\log|S|}{\epsilon^2})$. Proof is similar to that of Lemma 22.

D.5. Final Ranking Algorithm

Use BINARY-SEARCH-RANKING presented in (Falahatgar et al., 2017b) with new BINARY-SEARCH subroutine in place of the one presented there. Also replace 5(b) step there which ranks each bin in one go with a round of ranking followed by checking with RANK-CHECK if ranking is done correctly and repeating ranking if not done correctly.

Hence we improved both components that were incurring additional $\log\log|S|$ factors and thereby removed $(\log\log|S|)^3$ factor.

E. Proof for Theorem 10

Proof. This proof is similar to proof of Theorem 7 in (Falahatgar et al., 2017a).

Consider the model where $\tilde{p}_{a_1,a_2} = 1/2$, $\tilde{p}_{a_i,a_j} = (0 <)\mu(\ll 1/n^{10})$, when $i < j$ and $(i,j) \neq (1,2)$. This model has an order: $a_1 \succ a_2 \succ \cdots \succ a_{n-1} \succ a_n$ i.e., $\tilde{p}_{a_i,a_j} > 0$ $\forall i < j$. Further this model satisfies MST since $\tilde{p}_{a_i,a_k} \geq \min(\tilde{p}_{a_i,a_j}, \tilde{p}_{a_j,a_k})\ \forall i < j < k$. This model also satisfies STI since $\tilde{p}_{a_i,a_k} \leq \tilde{p}_{a_i,a_j} + \tilde{p}_{a_j,a_k}\ \forall i < j < k$.

We prove the Lemma by reducing the above model to the model where μ is replaced by 0.

Note that μ is so small that if we consider a model where we replace μ with 0, the comparisons behave essentially similarly. More formally, let model M_μ be the model we consider and M_0 be the model when μ is replaced with 0. Let C denote a sequence of comparisons where each element of the sequence includes the elements compared and its outcome. Further, for each sequence C, let $P_\mu(C)$ and $P_0(C)$ denote the probability of sequence C under models M_μ and M_0 respectively. Now consider a sequence C of comparisons of length $\leq n^2/20$ (chosen to make proof simpler. One can also prove for constants higher than $1/20$). Then

$$\frac{P_0(C)}{P_\mu(C)} \geq \left(\frac{1/2}{1/2+\mu}\right)^{n^2/20} \geq e^{-n^2/(10n^{10})} \geq \frac{6}{7}$$

Thus the probability of any sequence of length $\leq \frac{n^2}{20}$ is approximately same under both models. Hence if there is an algorithm that uses $\leq \frac{n^2}{20}$ comparisons and w.p.$\geq 7/8$ produces an $1/4$-ranking under M_μ model then applying same algorithm over M_0 model produces an $1/4$-ranking w.p.$\geq \frac{7}{8}\cdot\frac{6}{7} = \frac{3}{4}$.

We now show that there exists no algorithm that uses $\leq \frac{n^2}{20}$ comparisons and w.p.$\geq \frac{3}{4}$ generates a $1/4$-ranking under M_0, thus proving the Lemma. It is easy to see that any ordering outputted without querying the comparison between a_1 and a_2 is a $1/4$-ranking w.p. exactly 1/2 since no order between a_1 and a_2 can be deduced. Since the pair (a_1, a_2) is one random pair among $\binom{n}{2}$ pairs, the probability that the algorithm asks a comparison between this pair with $n^2/20$ comparisons is $< \frac{1}{2}$. So the probability that the output order contains a_1 and a_2 in the right order is $< \frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} = \frac{3}{4}$. □

F. Approximating Pairwise Probabilties

F.1. Proof for Theorem 11

Proof. Consider the model $\{a_1, a_2, ..., a_n\}$ where $(4+4k)\epsilon \leq \tilde{p}_{a_{i+k},a_i} \leq (8+4k)\epsilon$ for $1 \leq k \leq \min(n-i, \lfloor\frac{1}{16\epsilon} - 2\rfloor)$ and $\tilde{p}_{a_{i+k},a_i} = 1/4$ for $k > \min(n-i, \lfloor\frac{1}{16\epsilon} - 2\rfloor)$.

Notice that this model satisfies SST since there exists an ordering $\succ$ i.e., $a_j \succ a_i$ if $j > i$ (since $\tilde{p}_{a_j,a_i} > 0$) and $\tilde{p}_{a_k,a_i} \geq \min(\tilde{p}_{a_k,a_j}, \tilde{p}_{a_j,a_i})\ \forall k > j > i$.

Further observe that this model also satisfies STI since $\tilde{p}_{a_k,a_i} \leq \tilde{p}_{a_k,a_j} + \tilde{p}_{a_j,a_i}\ \forall k > j > i$.

Notice that even after knowing every other pairwise probability other than for pair $\{a_k, a_i\}$ exactly, SST and STI do not impose any additional constraints on $\tilde{p}_{a_k,a_i}$ than those already given by the model definition. Hence we can only infer $\tilde{p}_{a_k,a_i}$ from model definition and comparisons between a_k and a_i.

Model does not fix pairwise probability for

$$\geq \sum_{i=1}^{n} \min\left(n-i, \frac{1}{16\epsilon} - 2\right) = \Omega(n\min(n, 1/\epsilon))$$

pairs and for each such pair $\{a_i, a_j\}$, $\tilde{p}_{a_i,a_j}$ is known only upto an accuracy of 2ϵ.

So any algorithm has to approximate $\Omega(n\min(n,1/\epsilon))$ probabilities between $1/2$ and $3/4$ to an accuracy of ϵ. Using Information Theoretic arguments, it follows that any algorithm needs $\Omega\left(\frac{n\min(n,1/\epsilon)}{\epsilon^2}\log n\right)$ comparisons for error accuracy of $\leq 1/4$. $\square$

F.2. Properties of ϵ-ranked ordered Set

We first prove some properties of ϵ-ranked ordered set that we use in proving correctness of APPROX-PROB.

Lemma 24. *If S is an ϵ-ranked ordered set i.e., $\tilde{p}_{S(i),S(j)} \leq \epsilon\ \forall i < j$, then $\tilde{p}_{S(k),S(j)} \leq \tilde{p}_{S(k),S(i)} + \epsilon$ $\forall k \geq j \geq i$ and $\tilde{p}_{S(k),S(i)} \geq \tilde{p}_{S(j),S(i)} - \epsilon\ \forall k \geq j \geq i$.*

Proof. Notice that if $S(j) \succ S(i)$, then by SST $\tilde{p}_{S(k),S(j)} \leq \tilde{p}_{S(k),S(i)}$. If $S(i) \succ S(j)$, then since S is ϵ-ranked, $\tilde{p}_{S(i),S(j)} \leq \epsilon$, and hence by SST and STI, $\tilde{p}_{S(k),S(j)} \leq \tilde{p}_{S(k),S(i)} + \tilde{p}_{S(i),S(j)} \leq \tilde{p}_{S(k),S(i)} + \epsilon$.

Similarly it can be shown that $\tilde{p}_{S(k),S(i)} \geq \tilde{p}_{S(j),S(i)} - \epsilon$ $\forall k \geq j \geq i$. $\square$

F.3. Proof for Theorem 12

Proof. We first prove the correctness. Let $\epsilon' = \epsilon/8$. Notice that S is ϵ'-ranked. Let $\hat{\tilde{p}}_{S(i),S(j)}$ be the approximated output of $\tilde{p}_{S(i),S(j)}$.

Observe that whenever $\tilde{p}_{S(i),S(j)}$ is approximated using comparisons between $S(i)$ and $S(j)$, by Hoeffding's inequality, w.p.$\geq 1 - 1/|S|^4$, $|\hat{\tilde{p}}_{S(i),S(j)} - \tilde{p}_{S(i),S(j)}| < \frac{3\epsilon}{4}$, (since $S(i)$ and $S(j)$ are compared for $\frac{16}{\epsilon^2}\log|S|^4$ times and $\tilde{p}_{S(i),S(j)}$ is approximated to the nearest multiple of ϵ). Since at most $|S|^2$ pairs are compared, by union bound w.p.$\geq 1 - \frac{1}{|S|^2}$, all pair probabilities approximated using comparisons are correct to an accuracy of $3\epsilon/4$.

Now we show that even when $\hat{\tilde{p}}_{S(k),S(i)}$ is given the same value as $\hat{\tilde{p}}_{S(k-1),S(i)}$ without using comparisons, $\tilde{p}_{S(k),S(i)}$ is approximated to an accuracy of $3\epsilon/4 + 2\epsilon'$.

Notice that for $S(1)$, when approximating pairwise probability $\tilde{p}_{S(k),S(1)}$ using comparisons if $\hat{\tilde{p}}_{S(k),S(1)} < \hat{\tilde{p}}_{S(k-1),S(1)}$, then $\hat{\tilde{p}}_{S(k),S(1)}$ is given same value as $\hat{\tilde{p}}_{S(k-1),S(1)}$. We first show that this process approximates $\tilde{p}_{S(k),S(1)}$ to an accuracy of $3\epsilon/4 + \epsilon'$. Consider largest $l < k$ s.t. $\hat{\tilde{p}}_{S(l),S(1)} = \hat{\tilde{p}}_{S(k),S(1)}$. Notice that $\tilde{p}_{S(l),S(1)}$ is approximated using comparisons (because $\hat{\tilde{p}}_{S(l),S(1)} \neq \hat{\tilde{p}}_{S(l-1),S(1)}$) and hence $|\hat{\tilde{p}}_{S(l),S(1)} - \tilde{p}_{S(l),S(1)}| \leq 3\epsilon/4$. Since by Lemma 24, $\tilde{p}_{S(k),S(1)} \geq \tilde{p}_{S(l),S(1)} - \epsilon'\ \forall l < k$, it follows that

$$\begin{aligned}\hat{\tilde{p}}_{S(k),S(1)} &= \hat{\tilde{p}}_{S(l),S(1)} \\ &\leq \tilde{p}_{S(l),S(1)} + 3\epsilon/4 \\ &\leq \tilde{p}_{S(k),S(1)} + 3\epsilon/4 + \epsilon'.\end{aligned}$$

Notice that since initially $\hat{\tilde{p}}_{S(k),S(1)}$ was assigned a value smaller than $\hat{\tilde{p}}_{S(k-1),S(1)}$ using comparisons, $\tilde{p}_{S(k),S(1)} < \hat{\tilde{p}}_{S(k-1),S(1)} + 3\epsilon/4$ (since approximation using comparisons is correct to $3\epsilon/4$). Therefore $\tilde{p}_{S(k),S(1)} < \hat{\tilde{p}}_{S(k),S(1)} + 3\epsilon/4$. Hence $|\hat{\tilde{p}}_{S(k),S(1)} - \tilde{p}_{S(k),S(1)}| \leq 3\epsilon/4 + \epsilon'$.

Therefore if comparisons are invoked (even when approximated probability is not due to comparisons), pairwise probability is approximated to an accuracy of $3\epsilon/4 + \epsilon'$.

Now we consider the pairs for which no comparisons are invoked and show that even those pairwise probabilities are approximated to an accuracy of $3\epsilon/4 + 2\epsilon'$.

Suppose $\hat{\tilde{p}}_{S(k),S(i)}$ is copied same as $\hat{\tilde{p}}_{S(k-1),S(i)}$. Let j be the largest number $< k$ such that $\hat{\tilde{p}}_{S(j),S(i)} \neq \hat{\tilde{p}}_{S(j-1),S(i)}$. Notice that $\hat{\tilde{p}}_{S(j),S(i)} = \hat{\tilde{p}}_{S(k),S(i)}$ and comparisons are used between $S(j)$ and $S(i)$ (since $\hat{\tilde{p}}_{S(j),S(i)} \neq \hat{\tilde{p}}_{S(j-1),S(i)}$). Therefore $|\tilde{p}_{S(j),S(i)} - \hat{\tilde{p}}_{S(j),S(i)}| \leq 3\epsilon/4 + \epsilon'$. Further by Lemma 24, $\tilde{p}_{S(j),S(i)} \leq \tilde{p}_{S(k),S(i)} + \epsilon'$ and hence it follows that

$$\begin{aligned}\hat{\tilde{p}}_{S(k),S(i)} &= \hat{\tilde{p}}_{S(j),S(i)} \\ &\leq \tilde{p}_{S(j),S(i)} + 3\epsilon/4 + \epsilon' \\ &\leq \tilde{p}_{S(k),S(j)} + 3\epsilon/4 + 2\epsilon'.\end{aligned}$$

Similarly, it can be shown that $\hat{\tilde{p}}_{S(k),S(i)} \geq \tilde{p}_{S(k),S(j)} - 3\epsilon/4 - 2\epsilon'$. Therefore $|\hat{\tilde{p}}_{S(k),S(i)} - \tilde{p}_{S(k),S(i)}| \leq 3\epsilon/4 + 2\epsilon' = \epsilon$.

Hence w.p.$\geq 1 - \frac{1}{|S|^2}$, all pairwise probabilities are approximated to an accuracy of ϵ.

We now bound the number of comparisons.

We first show that $0 \leq \hat{\tilde{p}}_{S(k),S(l)} \leq \hat{\tilde{p}}_{S(k+1),S(l)} \leq \hat{\tilde{p}}_{S(k+1),S(l-1)}\ \forall k \geq l$. We prove this statement using induction. Notice that it is true for $l = 1$. We assume that it is true for l and prove correctness for $l + 1$.

Notice that $0 = \hat{\tilde{p}}_{S(l+1),S(l+1)} \leq \hat{\tilde{p}}_{S(l+2),S(l)}$. Now we further assume that $0 \leq \hat{\tilde{p}}_{S(k),S(l+1)} \leq \hat{\tilde{p}}_{S(k+1),S(l)}$

and show that $0 \le \hat{\tilde{p}}_{S(k+1),S(l+1)} \le \hat{\tilde{p}}_{S(k+2),S(l)}$. Notice that if $\hat{\tilde{p}}_{S(k),S(l+1)} = \hat{\tilde{p}}_{S(k+1),S(l)}$, $\hat{\tilde{p}}_{S(k+1),S(l+1)} = \hat{\tilde{p}}_{S(k+1),S(l)} \le \hat{\tilde{p}}_{S(k+2),S(l)}$. And if $\hat{\tilde{p}}_{S(k),S(l+1)} < \hat{\tilde{p}}_{S(k+1),S(l)}$, then $\tilde{p}_{S(k+1),S(l+1)}$ is approximated using comparisons and hence approximation accurate to $3\epsilon/4$. Notice that $\hat{\tilde{p}}_{S(k+1),S(l+1)}$ can't be more than $\hat{\tilde{p}}_{S(k+1),S(l)}$ since if it happens, $\hat{\tilde{p}}_{S(k+1),S(l+1)} \ge \hat{\tilde{p}}_{S(k+1),S(l)} + \epsilon \ge \hat{\tilde{p}}_{S(k),S(l+1)} + 2\epsilon$ but $\hat{\tilde{p}}_{S(k+1),S(l+1)} \le \tilde{p}_{S(k+1),S(l+1)} + 3\epsilon/4 \le \tilde{p}_{S(k),S(l+1)} + 3\epsilon/4 + \epsilon' \le \hat{\tilde{p}}_{S(k),S(l+1)} + 3\epsilon/2 + 3\epsilon' < \hat{\tilde{p}}_{S(k),S(l+1)} + 2\epsilon$ (contradiction). So $\hat{\tilde{p}}_{S(k+1),S(l+1)} \le \hat{\tilde{p}}_{S(k+1),S(l)} \le \hat{\tilde{p}}_{S(k+2),S(l)}$. Similarly it can be shown that $\hat{\tilde{p}}_{S(k+1),S(l+1)} \le \hat{\tilde{p}}_{S(k),S(l+1)}$.

Consider the pairs $(\{S(1),S(i)\},\{S(2),S(i-1)\},\{(S(3),S(i-2)\},...,\{S(i/2),S(i/2+1)\})$. Since $1/2 \ge \hat{\tilde{p}}_{S(i),S(1)} \ge \hat{\tilde{p}}_{S(i-1),S(2)} \ge \hat{\tilde{p}}_{S(i-2),S(3)} \cdots \ge \hat{\tilde{p}}_{S(i/2+1),S(i/2)} \ge 0$ and all values are multiples of ϵ, $\hat{\tilde{p}}_{S(k),S(j)} \ne \hat{\tilde{p}}_{S(k-1),S(j+1)}$ only for $\mathcal{O}(min(|S|,1/\epsilon))$ pairs $\{k,j\}$ such that $k+j=i+1$. Hence only for $\mathcal{O}(\min(|S|,1/\epsilon))$ pairs $\{S(l),S(m)\}$ s.t. $l+m=i+2$, comparisons are used to approximate pairwise probabilities. This statement is true for all sum of indices and hence only for $\mathcal{O}(|S|min(|S|,1/\epsilon))$ pairs, comparisons are used and hence total comparisons used is $\mathcal{O}\Big(\frac{|S|\min(|S|,1/\epsilon)}{\epsilon^2}\log|S|\Big)$. $\square$

G. Other Relevant Models

Researchers also considered models other than WST. For these models, one can still define maximum and ranking based on Borda scores, Copeland scores, or Von Neumann winner.

One such interesting model is considered in (Hüllermeier et al., 2008): there is a probability distribution P over set $\mathcal{R}$ of all possible rankings and pairwise preferences can be defined based on this probability distribution,

$$p_{i,j} = \sum_{r\in\mathcal{R}} P(r)1_{i\succ j \text{ in } r}.$$

This model does not necessarily satisfy WST, however (Hüllermeier et al., 2008) shows that this model still satisfies some relation among pairwise probabilities

$$p_{i,j} \ge p_{i,k} + p_{k,j} - 1. \tag{5}$$

It is natural to ask if a combination of WST and (5) guarantee a sub-quadratic complexity maxing algorithm. We give a negative answer to this question.

Recall that the model used in Section 3 to show lower bound of $\Omega(n^2)$ for WST is $p_{i,i+1} = 1$ and $p_{i,j} \approx 0.5$ for $|i-j| \ne 1$. This model satisfies WST but not Equation (5). Yet we can slightly change the model to satisfy this new relation and derive the lower bound of $\Omega(n^2)$. The new model that results in lower bound of $\Omega(n^2)$ under WST and Equation (5) is $p_{i,i+1} = 0.75$ and $p_{i,j} = 0.5$ for $|i-j| \ne 1$. It is easy to check that this new model requires more comparisons than the model in Section 3 to find $\frac{1}{8}$-maximum. Hence lower bound of $\Omega(n^2)$ also holds under the combination of WST and Equation 5.