Adaptive Sampling for Heterogeneous Rank Aggregation from Noisy Pairwise Comparisons

Yue Wu[1*], **Tao Jin**[2*], **Hao Lou**[2], **Pan Xu**[3], **Farzad Farnoud**[2†], **Quanquan Gu**[1†]

[1]University of California, Los Angeles [2]University of Virginia [3]California Institute of Technology

Abstract

In heterogeneous rank aggregation problems, users often exhibit various accuracy levels when comparing pairs of items. Thus, a uniform querying strategy over users may not be optimal. To address this issue, we propose an elimination-based active sampling strategy, which estimates the ranking of items via noisy pairwise comparisons from multiple users and improves the users' average accuracy by maintaining an active set of users. We prove that our algorithm can return the true ranking of items with high probability. We also provide a sample complexity bound for the proposed algorithm, which outperforms the non-active strategies in the literature and close to oracle under mild conditions. Experiments are provided to show the empirical advantage of the proposed methods over the state-of-the-art baselines.

1 INTRODUCTION

To rank a set of items from noisy pairwise comparisons or preferences is a widely studied topic in machine learning (Braverman and Mossel, 2008; Weng and Lin, 2011; Ren et al., 2019; Jin et al., 2020). This is also referred to as rank aggregation, which has many applications in practice such as ranking online game players (Herbrich et al., 2006), evaluating agents in games (Rowland et al., 2019), recommendation systems (Valcarce et al., 2017), etc. In the above cases, all data used in inference shares the assumption that each comparison has the same credibility. However, in a heterogeneous setting, the providers of subsets of data may have varying unknown accuracy levels. Thus, it is natural to take advantage of the more accurate ones to obtain a more accurate ranking using a smaller number of queries.

Nowadays, it is common to collect large-scale datasets in order to facilitate the process of knowledge discovery. Due to its scale, such data collection is usually carried out by crowdsourcing (Kumar and Lease, 2011; Chen et al., 2013), where different entities with diverse backgrounds generate subsets of the data. While crowdsourcing makes it possible to scale up the size, it also brings new challenges when it comes to the cost of operation and cleanness of the data. For example, the optimal ranking algorithm in the single-user setting (Ren et al., 2019) may not be straightforwardly extended to the heterogeneous setting while maintaining optimality. In particular, if we know the most accurate user among the set of users providing comparisons, the best we can do is to apply optimal single-user ranking algorithms such as Iterative-Insertion-Ranking (IIR) (Ren et al., 2019) by querying only the most accurate user. Unfortunately, in practice, the accuracies of the users are often unknown. A naive solution may be to randomly select a user to query and use the comparisons provided by this user to insert an item into the ranked list per IIR. However, as we show later, this naive method usually bears a high sample complexity. Therefore, it is of great interest to design methods that can adaptively select a subset of users at each time to query pairwise comparisons in order to insert an item correctly into the ranked list.

In this paper, we study the rank aggregation problem, where a heterogeneous set of users provide noisy pairwise comparisons for the items. We propose a novel algorithm that queries comparisons for pairs of items from a changing active user set. Specifically, we maintain a short history of user responses for a set of comparisons. When the inferred rank of these comparisons is estimated to be true with a high con-

[*]Equal Contribution

[†]Co-corresponding Authors

Proceedings of the 25[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

fidence, it is then used to calculate a reward based on the recorded responses. Then an upper confidence bound (UCB)-style elimination process is performed to remove inaccurate users from active user set. We theoretically analyze the sample complexity of the proposed algorithm, which reduces to the state-of-the-art ranking algorithm (Ren et al., 2019) for a single user. We conducted experiments on both synthetic and real-world dataset, which demonstrate that our adaptive sampling algorithm based on user elimination is much more sample efficient than baseline algorithms and can sometimes reach the performance of an oracle algorithm.

Our contributions are summarized as follows:

- We propose a novel algorithm called Ada-IIR for heterogeneous rank aggregation, which uses a successive elimination subroutine to adaptively maintain a set of active users during the ranking process.
- We prove that Ada-IIR achieves the same order of sample complexity as that of the oracle algorithm which has access to the optimal user and uses the state-of-the-art ranking approach designed for the single-user setting.
- We conducted experiments for heterogeneous rank aggregation problems on both synthetic and real-world datasets to show that the proposed algorithm costs significantly fewer samples than baseline algorithms in order to recover the exact ranking.

Notation. We use lower case letters to denote scalars, and lower and upper case bold letters to denote vectors and matrices. We use $\|\cdot\|$ to indicate the Euclidean norm. We also use the standard O and Ω notations. The notations like $\widetilde{O}$ are used to hide logarithmic factors. For a positive integer N, $[N] := \{1, 2, \ldots, N\}$.

2 RELATED WORK

In this section, we discuss two closely related topics to our work, which cover the two facets of heterogeneous rank aggregation: active ranking to infer the rank and best arm identification to select a subset of accurate information sources (e.g., users). In addition to this, we also introduce similar work bearing the idea that ranking information can be heterogeneous.

Active ranking. For passive ranking problems, a static dataset is given before hand. Inference of the ranking often relies on models of ranked data, such as the Bradley-Terry-Luce (BTL) model (Bradley and Terry, 1952) and the Thurstone model (Thurstone, 1927). In contrast to passive algorithms, active algorithms leverage assumptions embedded in the models to identify the most informative pairs to query, thus reducing the sample complexity of queries. For instance, in Maystre and Grossglauser (2017), under the assumption that the true scores for N items are generated by a Poisson process, with $O(N\text{poly}(\log(N))$ comparisons, an *approximate* ranking of n items can be found. Let the probability of making a correct comparison between item i and the most similar item to item i be $\frac{1}{2}+d_i$ and let $d_{\min} = \min_{i\in[n]} d_i$. An instance-dependent sample complexity bound of $O(n\log(n)d_{\min}^{-2}\log(n/(\delta d_{\min}))$ is provided along with a QuickSort based algorithm by Szörényi et al. (2015). In Ren et al. (2019), an analysis for a distribution agnostic active ranking scheme is provided. To achieve a δ-correct *exact* ranking, $O(\sum_{i\in[n]} d_i^{-2}(\log\log(d_i^{-1})+\log(n/\delta)))$ comparisons are required. The exact inference requirement results in repeated queries of the same pair, which costs a constant overhead compared to approximate inference.

Best Arm Identification (BAI). BAI is a pure exploration method in multi-armed bandits (Audibert et al., 2010; Chen et al., 2017). In the crowdsourcing setting, every user can be queried with the same question. Noting that some users can provide more accurate answers than the others, the goal is to identify the best user. We can regard the choice of which user to ask as an action, and the correctness of the user's response as the reward (cost) of the taken action. A long line of research has explored the identification of the best action with stochastic feedback. Recently, Resler and Mansour (2019) studied cases when the observed binary action costs can be inverted with a probability that is less than half. With a careful construction of the estimated cost despite the noise, the regret of the online algorithm suffers a constant order compared to the noiseless setting even without the knowledge of the inversion probability.

Heterogeneous Rank Aggregation. An early work from Chen et al. (2013) explored the idea of user-specific accuracy through a model that is equivalent to adding noise to the comparisons produced by the BTL model. More recently, Jin et al. (2020) proposed a natural extension of BTL and Thurstone generative models to heterogeneous population of users for pairwise comparisons and partial rankings. In addition to this line of work that assumes a global true ranking, mixture models (Zhao and Xia, 2019) were proposed for personal preference inference. These works output high accuracy approximate rankings.

3 PRELIMINARIES AND PROBLEM SETUP

3.1 Ranking from Noisy Pairwise Comparisons

Suppose there are N items that we want to rank and M users to be queried. An item is indexed by an integer $i \in [N]$. We assume there is a unique *true ranking* of the N items. A user is also indexed by an integer $u \in [M]$. For a subset of users, we use $\mathcal{U} \subseteq [M]$ to denote the index subset. In each time step, we can pick a pair of items i and j and ask a user u whether item i is better than item j. The comparison returned by the user may be noisy. We assume that for any pair of items (i, j) with true ranking $i \succ j$, the probability that the user u answers the query correctly is $p_u(i, j) = \Delta_u + 1/2$, where $\Delta_u \in (0, \frac{1}{2}]$ is referred to as the accuracy level of user u. When some of the Δ_u's are different from the others, we call the set of users *heterogeneous*. We assume comparison results for item pairs, regardless the queried user, are mutually independent. While this independence assumption may not always hold for real datasets, it is commonly adopted in the literature as it facilitates the analysis (Falahatgar et al., 2017, 2018; Jin et al., 2020).

In this paper, we aim to achieve the exact ranking for a ranking problem defined as follows.

Definition 1 (Exact Ranking with Multiple Users)**.** Given N items, M users, and $\delta \in (0, 1)$, our goal is to identify the true ranking among the N items with probability at least $1-\delta$. An algorithm $\mathcal{A}$ is δ-correct if, for any instance of the input, it will return the correct result in finite time with probability at least $1 - \delta$.

To actively eliminate the users in the user pool, we define an α-optimal user as follows.

Definition 2. Let $\mathcal{U} \subseteq [M]$ be an arbitrary subset of users. If a user $x \in \mathcal{U}$ satisfies $\Delta_x + \alpha \geq \max_{u \in \mathcal{U}} \Delta_u$, then x is called an *α-optimal user in $\mathcal{U}$*. If a user is α-optimal among all M users, then it is called an (global) α-optimal user.

3.2 Iterative Insertion Ranking with a Single User

When there is only one user u to be queried ($M = 1$), the problem defined in Section 3.1 reduces to the exact ranking problem with a single user, for which Ren et al. (2019) proposed the Iterative-Insertion-Ranking (IIR) algorithm. The sample complexity (i.e., the total number of queries) to achieve exact ranking with probability $1 - \delta$ is characterized by the following proposition:

Proposition 3 (Adapted from Theorems 2 and 12 in Ren et al. (2019))**.** Given $\delta \in (0, 1/12)$ and an instance of N items, the number of comparisons used by any δ-correct algorithm $\mathcal{A}$ on this instance is

$$\Theta\big(N\Delta_u^{-2}\big(\log\log\Delta_u^{-1} + \log(N/\delta)\big)\big). \tag{1}$$

Moreover, the IIR algorithm proposed by Ren et al. (2019) can output the exact ranking using this number of comparisons, with probability $1 - \delta$.

The complexity above can be decomposed into the complexity of inserting each item into a constructed sorting tree.

In this paper, we consider a more challenging ranking problem, where multiple users with heterogeneous levels of accuracies can be queried each time. In the multi-user setting, the optimal sample complexity in (1) can be achieved only if we know which user is the best user, i.e., $u^* = \arg\max_{u \in [M]} \Delta_u$. The optimal sample complexity can then be written as

$$\mathcal{C}_{u^*}(N) = \Theta\big(N\Delta_{u^*}^{-2}\big(\log\log\Delta_{u^*}^{-1} + \log(N/\delta)\big)\big). \tag{2}$$

However, with no prior information on the users' comparison accuracies, it is unclear whether we can achieve a sample complexity close to (2). In this scenario, the most primitive route is to perform no inference on the users' accuracy and randomly choose users to query. This leads to an equivalent accuracy of $\bar{\Delta}_0 = \frac{1}{M}\sum_{u \in [M]} \Delta_u$ and a sample complexity given as

$$\mathcal{C}_{\text{ave}}(N) = \Theta\big(N\bar{\Delta}_0^{-2}\big(\log\log\bar{\Delta}_0^{-1} + \log(N/\delta)\big)\big). \tag{3}$$

Compared with the best possible complexity (2), the sample complexity (3) is larger by a factor (ignoring logarithmic factors) up to M^2, because the ratio between Δ_{u^*} and $\bar{\Delta}_0$ could vary a lot for different set of users and can be as large as M. This is certainly undesirable, especially when there are a large number of items to be ranked. Therefore, an immediate question is: Can we design an algorithm that has a smaller multiplicative factor in its sample complexity compared with the optimal sample complexity? What we will propose in the following section is an algorithm that can achieve a sublinear regret, where the regret is defined as the difference between the sample complexity of the proposed algorithm and the optimal sample complexity.

4 ADAPTIVE SAMPLING AND USER ELIMINATION

The main framework of our procedure is derived based on the Iterative-Insertion-Ranking algorithm proposed in Ren et al. (2019), which, to the best of our knowledge, is the first algorithm that has matching instance-dependent upper and lower sample complexity bounds for active ranking problems in the single-user setting. We assume that the strong stochastic transitivity (SST) assumption defined in Falahatgar et al. (2017, 2018) holds in our setting. The ranking algorithm comprises the following four hierarchical parts and operates on a Preference Interval Tree (PIT) (Feige et al., 1994a; Ren et al., 2019), which stores the currently inserted and sorted items (the detailed definition is presented in Appendix A):

1. Adaptive Iterative-Insertion-Ranking (Ada-IIR): the main procedure which calls IAI to insert an item into a PIT with a high probability of correctness. It is displayed in Algorithm 1.
2. Iterative-Attempting-Insertion (IAI): the subroutine which calls ATI to insert the current item $z \in [N]$ into the ranked list with an error ϵ, and iteratively calls ATI by decreasing the error until the probability that the z-th item is inserted to the correct position is high enough. It is displayed in Algorithm 5.
3. Attempting-Insertion (ATI): the subroutine that traverses the Preference Interval Tree using binary search (Feige et al., 1994b) to find the node where the item should be inserted with error ϵ. To compare the current item and any node in the tree, it calls ATC to obtain the comparison result. It is displayed in Algorithm 6.
4. Attempting-Comparison (ATC): the subroutine that adaptively samples queries from a subset of users for a pair of items (z, j), where z is the item currently being inserted and j is any other item. ATC records the number of queries each user provides and the results of the comparisons. It is displayed in Algorithm 2.

In the heterogeneous rank aggregation problem, each user may have a different accuracy level from the others. Therefore, we adaptively sample the comparison data from a subset of users. In particular, we maintain an active set $\mathcal{U} \subseteq [M]$ of users, which contains the potentially most accurate users from the entire group. We add a user elimination phase to the main procedure (Algorithm 1) based on the elimination idea in multi-armed bandits (Slivkins et al., 2019; Lattimore and Szepesvári, 2020) to update this active set. In particular, we view each user as an arm in a multi-armed bandit, where the reward is 1 if the answer from a certain user is correct and 0 if wrong. After an item is successfully inserted by IAI, we call Algorithm 3 (EliminateUser) to eliminate users with low accuracy levels before we proceed to the next item.

To estimate the accuracy levels of users, a vector $\mathbf{s}_z \in \mathbb{R}^M$, recording the counts of responses from each user for every item, is maintained during the whole period of inserting the z-th item . We further keep track of two matrices $A_z, B_z \in \mathbb{R}^{N\times M}$. When a pair (i, j) (where i is the current item or in other words the z-th item being inserted and j is an arbitrary item) is compared by user $u \in \mathbb{R}^M$ in Algorithm 2, we increase $A[j, u]$ by 1 if user u thinks z-th item is better than j and increase $B[j, u]$ by 1 otherwise. We use w to record the total number of times that the z-th item is deemed better by any users and use the average $\widehat{p} = w/t$ to provide an estimation of the average accuracy $|\mathcal{U}|^{-1} \sum_{u\in\mathcal{U}} p^u_{ij}$. The variables A_z, B_z, and $\mathbf{s}_z$ are global variables, shared by different subroutines throughout the process. After the z-th item is successfully inserted, A_z, B_z will be discarded and the space allocated can be used for A_{z+1}, B_{z+1} (See Line 3 of Algorithm 1).

We use the 0/1 reward for each user to indicate whether the provided pairwise comparison is correct. Nevertheless, this reward is not known immediately after each arm-pull since the correctness depends on the ranking of items which is also unknown. But when IAI returns *inserted*, the item recently inserted has a high probability to be in the right place. Our method takes advantage of this fact by constructing a fairly accurate prediction of pairwise comparison for the item with all other already inserted items in the PIT. Then an estimate of the reward $\boldsymbol{n}_z$ can be obtained with the help of recorded responses A_z and B_z, which are updated in ATC as described in the preceding paragraph. At last, in Algorithm 3 a UCB-style condition is imposed on estimated accuracy levels $\boldsymbol{\mu} = \boldsymbol{n}_z/\boldsymbol{s}_z$.

Due to the space limit, we omit here the IAI and ATI routines that are proposed in Ren et al. (2019). We include them for completeness and ease of reference in Appendix A.

5 THEORETICAL ANALYSIS

In this section, we analyze the sample complexity of the proposed algorithm and compare it with other baselines mentioned in Section 3.

5.1 Sample Complexity of Algorithm 1

We first present an upper bound on the sample complexity of the proposed algorithm. Define $\bar{\Delta}_z = \frac{1}{\mathcal{U}_z} \sum_{u\in\mathcal{U}_z} \Delta_u$ to be the average accuracy of all users in the current active set. Denote

$$F(x) = x^{-2}(\log\log x^{-1} + \log(N/\delta)). \quad (4)$$

Although $F(x)$ depends on N and δ^{-1}, the dependence is only logarithmic, and it does not affect the validity of reasoning via big-O notations.

Algorithm 1 Main Procedure: ADAPTIVE ITERATIVE-INSERTION-RANKING (Ada-IIR)

Global Variables:
$z \in \mathbb{N}$: the index of the item being inserted into the ranked list.
$A_z \in \mathbb{R}^{N\times M}$: $A_z[j,u]$ is the number of times that user u thinks the z-th item is better than item j.
$B_z \in \mathbb{R}^{N\times M}$: $B_z[j,u]$ is the number of times that user u thinks the z-th item is worse than item j.
$\mathbf{s}_z \in \mathbb{R}^M$: total number of responses by each user so far.
Input parameters: A list of items S to rank and confidence δ. S is a permutation of $[N]$.
Initialize: $\boldsymbol{n}_1 = \boldsymbol{s}_1 = \mathbf{0}$

```
1: Ans ← the list containing only S[1]
2: for z ← 2 to |S| do
3:   n_z = n_{z-1}, s_z = s_{z-1}, A_z = 0, B_z = 0
4:   IAI(S[z], Ans, δ/(n-1))  ▷Algorithm 5 (global variables A_z, B_z, s_z are updated here)
5:   for j ∈ [z-1] do
6:     if S[z] > S[j] in PIT then
7:       n_z = n_z + A_z[S[j], *]
8:     else
9:       n_z = n_z + B_z[S[j], *]
10:    end if
11:  end for
12:  U_z ← ELIMINATEUSER(U_{z-1}, n_z, s_z, δ/(n-1))  ▷Algorithm 3
13: end for
14: return Ans;
```

Theorem 4. For any $\delta > 0$, with probability at least $1-\delta$, Algorithm 1 returns the exact ranking of the N items, and it makes at most $\mathcal{C}_{\mathrm{Alg}}(N)$ queries, where $\mathcal{C}_{\mathrm{Alg}}(N) = O(\sum_{z=2}^{N} \bar{\Delta}_z^{-2}(\log\log \bar{\Delta}_z^{-1} + \log(N/\delta))) = O(\sum_{z=2}^{N} F(\bar{\Delta}_z))$.

Proof. The analysis on the sample complexity follows a similar route as Ren et al. (2019) due to the similarity in algorithm design. In fact, since we randomly choose a user from $\mathcal{U}_t$ and query it for a feedback, it is equivalent to querying a single user with the averaged accuracy $\frac{1}{2} + \bar{\Delta}_z$, where $\bar{\Delta}_z := \frac{1}{|\mathcal{U}_z|}\sum_{u\in\mathcal{U}_z} \Delta_u$. This means most of the theoretical results from Ren et al. (2019) can also apply to our algorithm. In Appendix E.1, we present more detailed reasoning. □

5.2 Sample Complexity Comparison of Different Algorithms

While Theorem 4 characterizes the sample complexity of Algorithm 1 explicitly, the result therein is not directly comparable with the sample complexity of the oracle algorithm that only queries the best user $\mathcal{C}_{u^*}(N)$ or the complexity of the naive random-query algorithm $\mathcal{C}_{\mathrm{ave}}(N)$. Based on Theorem 4, we can derive the following more elaborate sample complexity for Algorithm 1.

Algorithm 2 Subroutine: ATTEMPT-TO-COMPARE (ATC) $(z, j, \mathcal{U}, \epsilon, \delta)$

Input: items (z,j) to be compared, set of users $\mathcal{U}$, confidence parameter ϵ, δ. M is the number of users originally.

```
m = |U|, p̂ = 0, w = 0, ŷ = 1. Number of rounds r = 1. r_max = ⌈(1/2) ε^{-2} log(2/δ)⌉.
while r ≤ r_max do
  Choose u uniformly at random from U
  Obtain comparison result from user u as y^u_{ij}
  Increment the counter of responses collected from this user s_z[u] ← s_z[u] + 1
  if y^u_{ij} > 0 then
    A_z[j,u] ← A_z[j,u] + 1, w ← w + 1
  else
    B_z[j,u] ← B_z[j,u] + 1
  end if
  p̂ ← w/r, r ← r + 1, c_r ← sqrt((1/(2t)) log(π^2 r^2 / (3δ)))
  if |p̂ - 1/2| ≥ c_r then
    break
  end if
end while
if p̂ ≤ 1/2 then
  ŷ = 0
end if
return: ŷ
```

Algorithm 3 Subroutine: ELIMINATEUSER

Input parameters: $(\mathcal{U}, \mathbf{n}, \mathbf{s}, \delta)$.

```
Set S = Σ_{u∈[M]} s_u, s_min = min_{u∈U} s_u, μ_u = n_u/s_u, r = sqrt(log(2|U|/δ)/(2 s_min))
Set LCB = μ - r1 and UCB = μ + r1.
if S ≥ 2M^2 log(NM/δ) then
  for u ∈ U do
    Remove user u from U if ∃u' ∈ U, UCB_u < LCB_{u'}.
  end for
end if
return U
```

Theorem 5. Suppose there are N items and M users initially. Denote $S_z = \sum_{u\in[M]}(\mathbf{s}_z)_u$ to be the number of all queries made before inserting the z-th item (Line 4 in Algorithm 1). The proposed algorithm has the following sample complexity upper bound:

$$\mathcal{C}_{\mathrm{Alg}}(N,M) = \Theta(NF(\Delta_{u^*}))+ O\left(\sum_{z=2}^{N} \mathbb{1}\left\{S_z < 2M^2\log(NM/\delta)\right\}\big(F(\bar{\Delta}_0) - F(\Delta_{u^*})\big)\right) + O\left(L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)} \sum_{z=2}^{N} \mathbb{1}\{S_z \geq 2M^2\log(NM/\delta)\}\sqrt{\frac{M}{S_z}}\right), \quad (5)$$

where $L(\mathcal{U}_0) = \frac{F(c\Delta_{u^*}^3)-F(\Delta_{u^*})}{\Delta_{u^*}-c\Delta_{u^*}^3}$ is an instance-dependent factor, with only logarithmic dependence on N and δ^{-1}(through F), and where $c = 1/25$ is a global constant.

Proof. The detailed proof can be found in Appendix E.2. □

A few discussions are necessary to show the meaning of the result. First, if the number of users $M \gg N$, then no user is eliminated because each user will be queried so few times that no meaningful inference can be made. Since the goal is to achieve the accuracy of the best user, more inaccurate users only make the task more difficult. Therefore, it is necessary to impose assumptions on M with respect to N.

This intuition can be made more precise. Suppose we loosely bound S_t as $S_t \geq t\log(t/\delta)$, which is reasonable since for a very accurate user the algorithm will spend roughly no more than $O(\log(t/\delta))$ comparisons to insert one item. This means the complexity can be bounded as (ignoring log factors)

$$\begin{aligned}\mathcal{C}_{\mathrm{Alg}}(N,M) = O\big(&NF(\Delta_{u^*})\big)\\ &+ \widetilde{O}\big(M^2\big(F(\bar{\Delta}_0) - F(\Delta_{u^*})\big)\big)\\ &+ \widetilde{O}\big(L(\mathcal{U}_0)\big(\sqrt{M}(\sqrt{N}-M)\big)\big). \quad (6)\end{aligned}$$

If $M = \Omega(\sqrt{N})$, then this is not ideal because our algorithm won't eliminate any user until $\Omega(N)$ items are inserted with accuracy $\bar{\Delta}_0$, which already leads to a gap linear in N compared with the best complexity $\mathcal{C}_{u^*}$. In this case, our algorithm roughly makes the same amount of queries as $\mathcal{C}_{\mathrm{ave}}$.

In order to avoid the bad case, it is necessary to assume $M = o(\sqrt{N})$ so that the last two terms become negligible (notice that $L(\mathcal{U}_0)$ is an instance-dependent constant). Now we restate Theorem 5 with the additional assumption, and compare it with the baselines.

Proposition 6. Suppose we have M users and N items to rank exactly, with $M = o(\sqrt{N})$. We have the following complexity along with (2) and (3):

$$\begin{aligned}\mathcal{C}_{u^*}(N,M) &= \Theta(NF(\Delta_{u^*})),\\ \mathcal{C}_{\mathrm{ave}}(N,M) &= \Theta(NF(\bar{\Delta}_0)),\\ \mathcal{C}_{\mathrm{Alg}}(N,M) &= \Theta(NF(\Delta_{u^*}))+\\ &\qquad o\big(N\big(F(\bar{\Delta}_0) - F(\Delta_{u^*})\big)\big) + o\big(N\big).\end{aligned}$$

The last two terms of $\mathcal{C}_{\mathrm{Alg}}(N,M)$ are negligible when compared with the first term. Therefore, our algorithm can perform comparably efficiently as if the best user were known while enjoying an advantage over the naive algorithm with sample complexity $\mathcal{C}_{\mathrm{ave}}(N,M)$.

Remark 7. Note that if we set $\mathcal{U}_0 = \{u^*\}$ for our algorithm, it will achieve exactly the same complexity as (2) indicates. Similarly, if we construct a new user $\bar{u}$ where $\Delta_{\bar{u}} = \bar{\Delta}_0$ and set $\mathcal{U}_0 = \{\bar{u}\}$, our algorithm will recover exactly (3). By this argument and the fact that Big-O notations hide no M, the first term in each equation actually has the same absolute constant factor. Therefore, our algorithm is indeed comparable with the best user.

Remark 8. Notice that $F(x) \to +\infty$ when $x \to 0$. This means $\mathcal{C}_{\mathrm{ave}}$ is very sensitive to the initial average accuracy margin $\bar{\Delta}_0$. In the case where there is only one best user u^* and all other users have a near-zero margin $\Delta_u \to 0$, $\mathcal{C}_{\mathrm{ave}}$ can be very large compared with $\mathcal{C}_{u^*}$.

Remark 9. In the experiments, we notice that even with $N = 10$ and $M = 9$, after inserting the first item, each user has already been queried for enough times so that $S_2 \geq 2M^2\log(NM/\delta)$, which makes the second term in (5) vanish.

6 A TWO-STAGE ALGORITHM

In this section, we present an alternative simple scheme, called two-stage ranking with a heterogeneous set of users. This provides another baseline with which we can compare Ada-IIR. Additionally, it can be useful in situations with a large number of users, i.e., $M = \Omega(\sqrt{N})$, where Ada-IIR is less effective.

Two-stage ranking first performs user-selection and then item-ranking. In the user-selection stage, we search for an α-optimal user for some small α. Specifically, we first take an arbitrary pair of items (i,j) and then run the Iterative-Insertion-Ranking (IIR) algorithm (see Proposition 3) on them to determine the order, e.g., $i \succ j$, with high probability. Note that at this point, users have not been distinguished yet. So we take each query from a randomly chosen user. As discussed in Section 3.2, this is equivalent to querying the user $\bar{u}$ whose accuracy is $\bar{\Delta}_0$. Given $i \succ j$, the problem of finding an α-optimal user is reduced to pure exploration of an α-optimal arm in the context of multi-armed bandit: making queries about the pre-determined item pair from user u is the same as generating outcomes from an arm with Bernoulli($\frac{1}{2}+\Delta_u$) reward, e.g., if user u returns the answer $i \succ j$ then we get reward 1, otherwise we get reward 0. Hence, an α-optimal user is equivalent of an α-optimal arm. For determining an α-optimal arm, we can adopt the Median-Elimination (ME) algorithm from Even-Dar et al. (2002). After ME returns an α-optimal user u_α, we discard all other users and rank items by only querying u_α. Ranking with a single user can again be done by the IIR algorithm.

It is clear that two-stage ranking is composed of three procedures: IIR for determining the order of i and j, ME for obtaining an α-optimal user, and IIR again for producing the final ranking. The complexity of two-stage ranking is therefore the sum of complexities of the three procedures.

Theorem 10. For any $\alpha \in (0, \Delta_{u^*})$, two-stage ranking outputs the exact ranking with probability at least $1-\delta$ using at most $\mathcal{C}_{\text{tsr}}(N, M)$ comparisons, where

$$\mathcal{C}_{\text{tsr}}(N,M) = \Theta\biggl(\frac{1}{\bar{\Delta}_0^2}\left(\log\log\frac{1}{\bar{\Delta}_0}+\log\frac{1}{\delta}\right)+\frac{M}{\alpha^2}\log\frac{1}{\delta} + \frac{N}{(\Delta_{u^*}-\alpha)^2}\left(\log\log\frac{1}{\Delta_{u^*}-\alpha}+\log\frac{N}{\delta}\right)\biggr). \tag{7}$$

A more formal statement of two-stage ranking as well as the proof of Theorem 10 are presented in Appendix B.1.

Recall $F(x) = x^{-2}(\log\log x^{-1} + \log(N/\delta))$ defined in (4). From the preceding theorem, it is clear that for any constant α, when $M = o(N \log N)$, $\mathcal{C}_{\text{tsr}}(N, M)$ is dominated by $\Theta(NF(\Delta_{u^*}-\alpha))$. From Proposition 6, when $M = o(\sqrt{N})$, $\mathcal{C}_{\text{ave}}(N,M) = \Theta\left(NF\left(\bar{\Delta}_0\right)\right)$ and $\mathcal{C}_{\text{Alg}}(N,M) = \Theta(NF(\Delta_{u^*}))$. Therefore, as long as α is properly chosen, two-stage ranking has lower complexity than the non-adaptive ranking. However, since $\alpha > 0$, there is a linear gap between two-stage ranking to the proposed algorithm Ada-IIR. On the other hand, two-stage ranking is less constrained than Ada-IIR as it has an advantage over the non-adaptive scheme when the number of users M is in the regime $\Theta(\sqrt{N})$ while Ada-IIR does not. More detailed analysis about two-stage ranking is presented in Appendices B.2 and B.3.

7 EXPERIMENTS

In this section, we study the empirical performance of the following algorithms on both synthetic and real-world datasets:

- **IIR** (Ren et al., 2019): The original single-user algorithm adapted to the multi-user case by querying a user selected uniformly at random.
- **Ada-IIR**: The proposed method.
- **Two-stage ranking**: A simple method described in Section 6.
- **Oracle**: Query only the best user as if it is known.

Confidence parameter $\delta = 0.25$, $\alpha = 0.05$ is set if required by algorithm. The code of our implementation is available at https://github.com/ipsl/Ada-IIR.

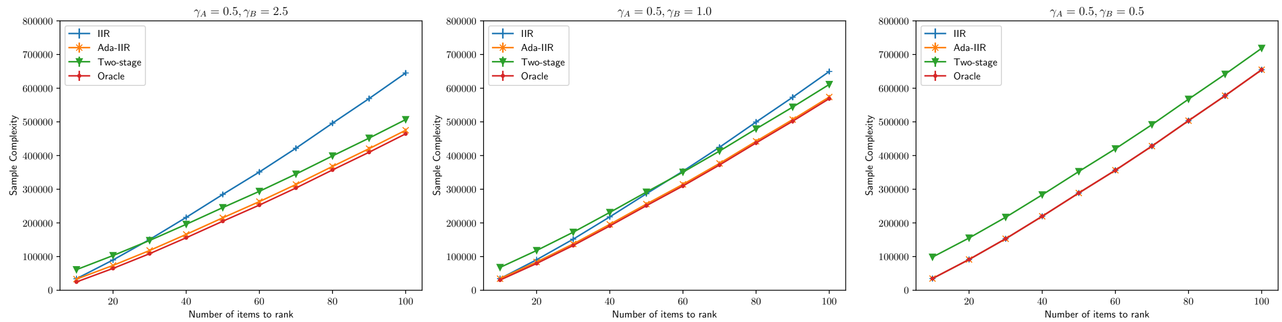
7.1 Synthetic Experiment

In our experiment, we use a similar setup as that of Jin et al. (2020), except that every pair has the same disatnace. In particular, we consider a set of users $[M]$, whose accuracies are set by $p_u(i,j) = (1+\exp(\gamma_u(s_j-s_i)))^{-1}$, for $u \in [M]$ and items $i, j \in [N]$, where parameter γ_u determines the user accuracy and s_i, s_j are the utility scores of the corresponding items in the BTL model. Larger values of γ_u lead to more accurate users. We set $s_i - s_j = 3$ if $i \prec j$ and $s_i - s_j = -3$ otherwise. Note that here we assume that the accuracy of user u is the same for all pair of items (i, j) as long as $i \prec j$. We assume that there are two distinct groups of users: the high-accuracy group in which the users have the same accuracy $\gamma_u = \gamma_B \in \{0.5, 1.0, 2.5\}$ in three different settings; and the low-accuracy group in which the users have the same accuracy $\gamma_u = \gamma_A = 0.5$ in all settings. This set of γ_u, s_i, s_j is chosen so that $p_u(i,j)$ for accurate users ranges from 0.55 to 0.99 and inaccurate users have a value close to 0.55.

The number of items to be ranked ranges from 10 to 100. Each setting is repeated 100 times with randomly generated data. To showcase the effectiveness of active user selection, we tested a relatively adverse situation where only 12 out of $M = 36$ users are highly accurate.

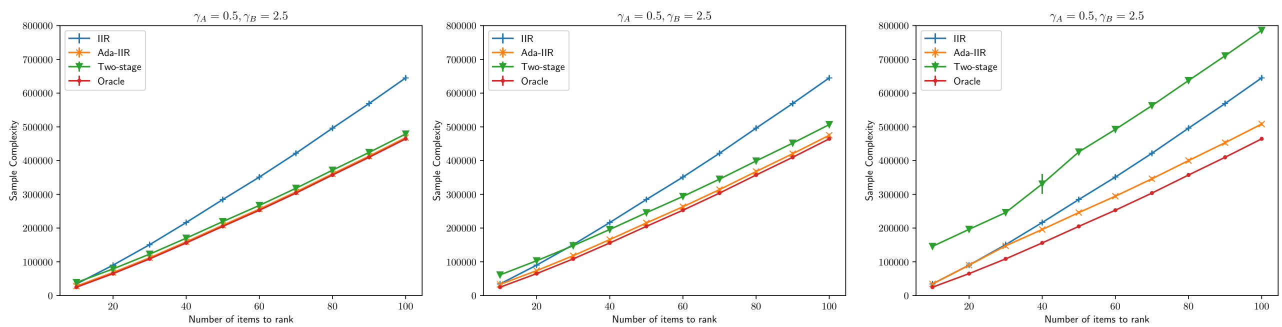
The average sample complexity and standard deviation over 100 runs are plotted in Figure 1. Note that the standard deviation is hard to see, given that it is small compared to the average. In most cases, the proposed method achieves nearly identical performance to the oracle algorithm, with only a small overhead. For two-stage ranking, we observe a constant overhead regardless the accuracy of the users. It may outperform the non-adaptive one (IIR) if there exist enough highly accurate users such as in Figure 1(a). However, the situation is less favorable for the two-stage algorithm when the cost of finding the best user overwhelms the savings of queries due to increased accuracy as shown in Figure 1(b). It may even have an adverse effect when accuracies are similar, as shown in Figure 1(c).

When we increase the total number of users and keep their accuracy the same, as shown in Figure 2, the Ada-IIR algorithm is able to tackle the increasing difficulty in finding more accurate users within a larger



(a) $\gamma_A = 0.5$, $\gamma_B = 2.5$ (b) $\gamma_A = 0.5$, $\gamma_B = 1.0$ (c) $\gamma_A = 0.5$, $\gamma_B = 0.5$

Figure 1: Sample complexities v.s. number of items for all algorithms. (a) (b) and (c) are different heterogeneous user settings where the accuracy of two group of users differs.



(a) $M = 18$ (b) $M = 36$ (c) $M = 72$

Figure 2: Sample complexities v.s. number of items for all algorithms. (a) (b) and (c) are different settings where the number of users differs. The accuracy of two groups of users are $\gamma_A = 0.5$, $\gamma_B = 2.5$.

pool. Although, the overhead increases, our proposed method can adapt to each case and deliver near optimal performance.

In our experiments every algorithm is able to recover the exact rank with respect to the ground truth, which is reasonable since the IIR algorithm is designed to output an exact ranking. And due to the union bounds used to guarantee a high probability correct output, the algorithms tend to request more than enough queries so we did not see a case in which a non-exact ranking was produced.

7.2 Real-world Experiment

The above synthetic experiments serve as a proof of concept. We add one more experiment based on the real data, the setting is from the "Country Population" dataset from Jin et al. (2020). In this dataset, the population of 15 countries were ranked by workers. Since the ground-truth Δ_u is not available, we first used the method described in the same work to infer the user accuracy and item parameters. During the simulation, the responses are generated according to their model with these parameters. As we have discussed in 5.2, the number of users should fall in a reasonable range. Thus, we randomly sub-sample a set of 25 users since the set of users provided by the dataset is excessive. The results, shown in Table 1, suggest that the Ada-IIR provides a moderate improvement over the non-adaptive algorithm.

METHOD	SAMPLE COMPLEXITY
IIR	59223 ± 3183
Two-stage	85027 ± 2619
Ada-IIR	52693 ± 2739
Oracle	43855 ± 2365

Table 1: Experiments on Country Population with 15 items and 25 users.

8 CONCLUSIONS

In this paper, we study the heterogeneous rank aggregation problem, where noisy pairwise comparisons are provided by a group of users with different accuracy levels. We propose a new ranking algorithm based on the idea of arm elimination from multi-armed bandits. The algorithm can identify the best user and utilize this information to efficiently perform the ranking. Under the Bernoulli setting, we provide theoretical guarantees that our algorithm is comparable with the oracle algorithm that knows the best user, and the gap between sample complexities of these two methods is only sublinear in the number of items. We conduct thorough experiments and show that the proposed algorithm can perform as good as the oracle algorithm and is significantly more sample efficient than all baseline algorithms. One immediate and interesting future direction may be to extend our adaptive sampling algorithm to more complicated models such as the heterogeneous Bradley-Terry-Luce model and the heterogeneous Thurstone Case V model (Jin et al., 2020).

9 Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. YW, PX and QG are supported in part by the NSF grants CIF-1911168 and III-1904183. TJ and FF are supported in part by the NSF grant CIF-1908544. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

Audibert, J.-Y., Bubeck, S. and Munos, R. (2010). Best arm identification in multi-armed bandits. In *COLT*.

Bradley, R. A. and Terry, M. E. (1952). Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika* **39** 324–345.

Braverman, M. and Mossel, E. (2008). Noisy sorting without resampling. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*.

Chen, J., Chen, X., Zhang, Q. and Zhou, Y. (2017). Adaptive multiple-arm identification. In *International Conference on Machine Learning*.

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

Even-Dar, E., Mannor, S. and Mansour, Y. (2002). Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*. Springer.

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

Falahatgar, M., Orlitsky, A., Pichapati, V. and Suresh, A. T. (2017). Maximum selection and ranking under noisy comparisons. *arXiv preprint arXiv:1705.05366* .

Feige, U., Raghavan, P., Peleg, D. and Upfal, E. (1994a). Computing with noisy information. *SIAM J. Comput.* **23** 1001–1018.

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

Herbrich, R., Minka, T. and Graepel, T. (2006). Trueskilltm: A bayesian skill rating system. In *NIPS*.

Jin, T., Xu, P., Gu, Q. and Farnoud, F. (2020). Rank aggregation via heterogeneous thurstone preference models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Kumar, A. and Lease, M. (2011). Learning to Rank from a Noisy Crowd. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11, ACM, New York, NY, USA.

Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press.

Maystre, L. and Grossglauser, M. (2017). Just sort it! a simple and effective approach to active preference learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org.

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

Resler, A. and Mansour, Y. (2019). Adversarial online learning with noise. In *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*. PMLR, Long Beach, California, USA.

Rowland, M., Omidshafiei, S., Tuyls, K., Pérolat, J., Valko, M., Piliouras, G. and Munos, R. (2019). Multiagent evaluation under incomplete information. In *NeurIPS*.

Slivkins, A. et al. (2019). Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning* **12** 1–286.

Szörényi, B., Busa-Fekete, R., Paul, A. and Hüllermeier, E. (2015). Online rank elicitation for plackett-luce: A dueling bandits approach. In *Advances in Neural Information Processing Systems*.

Thurstone, L. L. (1927). A law of comparative judgment. *Psychological Review* **34** 273–286.

Valcarce, D., Parapar, J. and Barreiro, Á. (2017). Combining top-n recommenders with metasearch algorithms. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* .

Weng, R. C. and Lin, C.-J. (2011). A bayesian approximation method for online ranking. *Journal of Machine Learning Research* **12**.

Zhao, Z. and Xia, L. (2019). Learning mixtures of plackett-luce models from structured partial orders. In *NeurIPS*.

A More Details About the Proposed Algorithm

We borrow the definition of Preference Interval Tree (PIT) (Feige et al., 1994a; Ren et al., 2019) based on which we can insert items to a ranked list. Specifically, given a list of ranked items S the PIT can be constructed using the following Algorithm 4.

Algorithm 4 Build PIT

Input parameters: S
Data structure: NODE $= \{left, mid, right, lchild, rchild, parent\}$, $left, mid, right$ holds index values, $lchild, rchild, parent$ points to any other NODE.
Initialize: $N = |S|$

X = CREATEEMPTYNODE returns an empty Node with above mentioned data structure
X.left $= -1$
X.right $= |S|$
X.mid $= \lfloor (X.left + X.right)/2 \rfloor$
queue = [X]
while queue.NOTEMPTY **do**
    X = queue.POPFRONT
    X.mid $= \lfloor (X.left + X.right)/2 \rfloor$
    if X.right - X.left > 1 **then**
        lnode = CREATEEMPTYNODE
        lnode.left = X.left
        lnode.right = mid
        X.lchild = lnode
        rnode = CREATEEMPTYNODE
        queue.append(lnode)
        rnode.left = X.mid
        rnode.right = X.right
        X.rchild = rnode
        queue.append(rnode)
    end if
end while
replace -1 with $-\infty$, $|S|$ with ∞ in each NODE.left and NODE.right.

For the completeness of our paper, we also present the subroutines ITERATIVE-ATTEMPTING-INSERTION (IAI) and ATTEMPTING-INSERTION (ATI) in this section, which are omitted in Section 4 due to space limit. In particular, IAI is displayed in Algorithm 5 and ATI is displayed in Algorithm 6. Both algorithms are proposed by Ren et al. (2019) for adaptive sampling in the single user setting.

Algorithm 5 Subroutine: ITERATIVE ATTEMPT TO INSERT(IAI)

Input parameters: (i, S, δ)
Initialize: For all $\tau \in \mathbb{Z}^+$, set $\epsilon_\tau = 2^{-(\tau+1)}$ and $\delta_\tau = \frac{6\delta}{\pi^2\tau^2}$; $t \leftarrow 0$; $Flag \leftarrow unsure$;

repeat
    $t \leftarrow t + 1$;
    $Flag \leftarrow$ATI$(i, S, \epsilon_\tau, \delta_\tau)$;
until $Flag = inserted$

B Two-stage Ranking

In this section, we formally state and analyze the two-stage ranking presented in Section 6.

Algorithm 6 Subroutine: ATTEMPT TO INSERT(ATI).

Input parameters: (i, S, ϵ, δ)
Initialize: Let z be a PIT constructed from S, $h \leftarrow \lceil 1 + \log_2(1 + |S|) \rceil$, the depth of z
For all leaf nodes u of z, initialize $c_u \leftarrow 0$; Set $t^{\max} \leftarrow \lceil \max\{4h, \frac{512}{25} \log \frac{2}{\delta}\} \rceil$ and $q \leftarrow \frac{15}{16}$

$X \leftarrow$ the root node of z;
for $t \leftarrow 1$ to $t^{\max}$ **do**
  if X is the root node **then**
    if $\mathrm{ATC}(i, X.\mathrm{mid}, \epsilon, 1-q) = i$ **then**
      $X \leftarrow X.\mathrm{rchild}$
    else
      $X \leftarrow X.\mathrm{lchild}$
    end if
  else if X is a leaf node **then**
    if $\mathrm{ATC}(i, X.\mathrm{left}, \epsilon, 1-\sqrt{q}) = i \wedge \mathrm{ATC}(i, X.\mathrm{right}, \epsilon, 1-\sqrt{q}) = X.\mathrm{right}$ **then**
      $c_X \leftarrow c_X + 1$
      if $c_X > b^t := \frac{1}{2}t + \sqrt{\frac{t}{2} \log \frac{\pi^2 t^2}{3\delta}} + 1$ **then**
        Insert i into the corresponding interval of X and
        return *inserted*
      end if
    else if $c_X > 0$ **then**
      $c_X \leftarrow c_X - 1$
    else
      $X \leftarrow X.\mathrm{parent}$
    end if
  else
    if $\mathrm{ATC}(i, X.\mathrm{left}, \epsilon, 1-\sqrt[3]{q}) = X.\mathrm{left} \vee \mathrm{ATC}(i, X.\mathrm{right}, \epsilon, 1-\sqrt[3]{q}) = i$ **then**
      $X \leftarrow X.\mathrm{parent}$
    else if $\mathrm{ATC}(i, X.\mathrm{mid}, \epsilon, 1-\sqrt[3]{q}) = i$ **then**
      $X \leftarrow X.\mathrm{rchild}$
    else
      $X \leftarrow X.\mathrm{lchild}$
    end if
  end if
end for
if there is a leaf node u with $c_u \geq 1 + \frac{5}{16} t^{\max}$ **then**
  Insert i into the corresponding interval of u and
  return *inserted*
else
  return *unsure*
end if

B.1 Algorithm Outline

We present two-stage ranking in Algorithm 7. As described in Section 6, an arbitrary pair of items is first fed to the IIR algorithm for determining the order using the 'average' user. Next, the Median-Elimination (ME) algorithm Even-Dar et al. (2002) is used to find an α-optimal user. After that, the total ranking can be obtained by applying the IIR algorithm on the selected user. IIR takes a set of items, the confidence level and a user as inputs and outputs a ranking of the items. ME takes a set $\mathcal{U}$ of users, real numbers α, δ and two ranked items as inputs and outputs an α-optimal user in $\mathcal{U}$ with probability at least $1 - \delta$.

Theorem 10. For any $\alpha \in (0, \Delta_{u^*})$, two-stage ranking outputs the exact ranking with probability at least $1 - \delta$

Algorithm 7 TWO-STAGE RANKING($\mathcal{N}$,$\mathcal{U}$, α, δ)

input: set of items $\mathcal{N}$, set of users $\mathcal{U}$, desired near-optimal level α, confidence level δ.
Let i, j be two arbitrary items. Let $\bar{u}$ be the 'average' user.
$[i', j'] \leftarrow$ Iterative-Insertion-Ranking($\{i, j\}$, $\frac{\delta}{3}$, $\bar{u}$).
$u_\alpha \leftarrow$ Median-Elimination($\mathcal{U}$, α, $\frac{\delta}{3}$, $[i', j']$)
output: Iterative-Insertion-Ranking($\mathcal{N}$, $\frac{\delta}{3}$, u_α)

using at most $\mathcal{C}_{\text{tsr}}(N, M)$ comparisons, where

$$\mathcal{C}_{\text{tsr}}(N, M) = \Theta\Bigg(\frac{1}{\bar{\Delta}_0^2}\left(\log\log\frac{1}{\bar{\Delta}_0} + \log\frac{1}{\delta}\right) + \frac{M}{\alpha^2}\log\frac{1}{\delta} + \frac{N}{(\Delta_{u^*} - \alpha)^2}\left(\log\log\frac{1}{\Delta_{u^*} - \alpha} + \log\frac{N}{\delta}\right)\Bigg). \tag{7}$$

Proof of Theorem 10. It is clear that two-stage ranking is composed of three procedures: IIR for determining the order of i, j, ME for obtaining an α-optimal user and IIR again for producing the final true ranking. The probability guarantee of two-stage ranking follows from applying the union bound on the three procedures.

From Proposition 3, Iterative-Insertion-Ranking$\big(\{i, j\}, \frac{\delta}{3}, \bar{u}\big)$ takes number of queries at most

$$\Theta\left(\frac{2}{\bar{\Delta}_0^2}\left(\log\log\frac{1}{\bar{\Delta}_0} + \log\left(\frac{6}{\delta}\right)\right)\right), \tag{8}$$

Iterative-Insertion-Ranking$\big(\mathcal{N}, \frac{\delta}{3}, u_\alpha\big)$ takes number of queries at most

$$\Theta\left(\frac{N}{(\Delta_{u^*} - \alpha)^2}\left(\log\log\frac{1}{\Delta_{u^*} - \alpha} + \log\left(\frac{3N}{\delta}\right)\right)\right) \tag{9}$$

by noting that the accuracy of u_α is at least $\Delta_{u^*} - \alpha$. Moreover, it is shown in Even-Dar et al. (2002) that ME outputs an α-optimal user using at most

$$\Theta\left(\frac{M}{\alpha^2}\log\frac{1}{\delta}\right) \tag{10}$$

comparisons.

The desired complexity bound thus follows from summing up (8), (9) and (10). □

B.2 Complexity Analysis

In this subsection, we provide a more detailed discussion on the complexity of the two-stage algorithm described in Algorithm 7. Recall that we define

$$F(x) = x^{-2}\left(\log\log x^{-1} + \log\left(N/\delta\right)\right).$$

When the average user accuracy $\bar{\Delta}_0$ and the maximum accuracy Δ_{u^*} are constants reflecting population statistics[1], the first term in (7) becomes negligible and we can write

$$\mathcal{C}_{\text{tsr}} = \Theta\left(\frac{M}{\alpha^2}\log\frac{1}{\delta} + NF\left(\Delta_{u^*} - \alpha\right)\right).$$

Noting that $NF\left(\Delta_{u^*} - \alpha\right)$ is of order $\frac{N\log N}{(\Delta_{u^*} - \alpha)^2}$, the following propositions can be made as M and α take different values.

[1]For instance, when user accuracies follow from a probability distribution, it is reasonable to let $\bar{\Delta}_0$ and Δ_{u^*} remain constants as N, M grow.

Proposition 11. When $M = \omega(N \log N)$ or $\alpha = o\left(\sqrt{\frac{M}{N \log N}}\right)$,

$$\mathcal{C}_{\text{tsr}}(N, M) = \omega(N \log N) + \Theta\left(NF(\Delta_{u^*} - \alpha)\right).$$

When $M = \omega(N \log N)$ or $\alpha = o\left(\sqrt{\frac{M}{N \log N}}\right)$, the dominating term in $\mathcal{C}_{\text{tsr}}$ is $\frac{M}{\alpha^2} \log \frac{1}{\delta} = \omega\left(N \log N\right)$, i.e., the number of comparisons it takes in the user-selection stage can be more costly than ranking items. If particular, when the number of users M is too large, even asking each user one question becomes unaffordable. When α is chosen too small, although the selected user is closer to optimal, the saving on ranking complexity is not obvious. Both cases are undesirable.

Proposition 12. When $M = O(N)$ and $\alpha = \omega\left(\sqrt{\frac{M}{N \log N}}\right) \cap o(1)$, then

$$\mathcal{C}_{\text{tsr}}(N, M) = \Theta\left(NF(\Delta_{u^*})\right) + o(N \log N) + O(1).$$

When $M = O(N)$ and $\alpha = \omega\left(\sqrt{\frac{M}{N \log N}}\right) \cap o(1)$, $\mathcal{C}_{\text{tsr}}$ is dominated by (9) and equals $\Theta\left(NF\left(\Delta_{u^*}\right)\right)$ since

$$\frac{N}{\left(\Delta_{u^*} - \alpha\right)^2}\left(\log \log \frac{1}{\Delta_{u^*} - \alpha} + \log \frac{3N}{\delta}\right) = NF\left(\Delta_{u^*}\right)\left(1 + o(1)\right).$$

Therefore, when the number of users M is not much larger than the number of items N, two-stage ranking can achieve order optimal by choosing α sufficiently small. In particular, if there exists a universal constant D such that the complexity of IIR with user accuracy Δ is $D \cdot NF\left(\Delta\right)\left(1 + o(1)\right)$, then (9) equals $D \cdot NF\left(\Delta_{u^*}\right)\left(1 + o(1)\right)$, which implies that $\mathcal{C}_{\text{tsr}} = \mathcal{C}_{u^*}\left(1 + o(1)\right)$, i.e., two-stage ranking is asymptotically optimal.

Proposition 13. When α is a constant,

$$\mathcal{C}_{\text{tsr}}(\alpha) = \Theta\left(NF(\Delta_{u^*} - \alpha)\right) + O(M).$$

When α is a constant, (10) and (9) are the dominating terms of $\mathcal{C}_{\text{tsr}}$. Moreover, if $M = o(N \log N)$, then $\mathcal{C}_{\text{tsr}}(N, M)$ equals $\Theta\left(NF\left(\Delta_{u^*} - \alpha\right)\right)$. Two-stage ranking in this case is equivalent of the complexity of IIR using a single user with accuracy $\Delta_{u^*} - \alpha$. Therefore, as long as $\alpha < \Delta_{u^*} - \bar{\Delta}_0$, two-stage ranking will be more efficient than the non-adaptive baseline.

More generally, if we remove the assumption that $\bar{\Delta}_0$ and Δ_{u^*} are constants, $\mathcal{C}_{\text{tsr}}(N, M)$ can be in the worst case as large as

$$\Theta\left(\frac{M^2}{\Delta_{u^*}^2}\left(\log \log \frac{M}{\Delta_{u^*}} + \log \frac{1}{\delta}\right) + \frac{M}{\alpha^2} \log \frac{1}{\delta} + \frac{N}{\left(\Delta_{u^*} - \alpha\right)^2}\left(\log \log \frac{1}{\Delta_{u^*} - \alpha} + \log \frac{N}{\delta}\right)\right), \tag{11}$$

by noting that $\bar{\Delta}_0 \geq \frac{\Delta_{u^*}}{M}$. Again, the desired situation is when the first two terms are negligible so that $\mathcal{C}_{\text{tsr}}(N, M)$ is equivalent to the complexity of ranking using an α-optimal user, as formulated in the following proposition.

Proposition 14. By (11), when $M = O\left(\sqrt{N}\right)$ and $\frac{M}{\alpha^2} = o\left(N \log N\right)$,

$$\mathcal{C}_{\text{tsr}}(N, M) = \Theta\left(NF\left(\Delta_{u^*} - \alpha\right)\right).$$

Recall from (2), (3) and Proposition 6 that

$$\begin{aligned}
&\mathcal{C}_{u^*}(N, M) = \Theta(NF(\Delta_{u^*})), \quad \mathcal{C}_{\text{ave}}(N, M) = \Theta(NF(\bar{\Delta}_0)), \\
&\mathcal{C}_{\text{Alg}}(N, M) = \Theta(NF(\Delta_{u^*})) + o\big(N\big(F(\bar{\Delta}_0) - F(\Delta_{u^*})\big)\big) + o\big(N\big) \text{ when } M = o(\sqrt{N}).
\end{aligned}$$

Therefore, when $M = O\left(\sqrt{N}\right)$, if there exists α such that $\frac{M}{\alpha^2} = o\left(N \log N\right)$ and $\alpha < \Delta_{u^*} - \bar{\Delta}_0$, two-stage ranking can be more efficient than the non-adaptive baseline with this choice of α. Note that this choice of α always exists as long as $\Delta_{u^*} - \bar{\Delta}_0 \geq \omega\left(\sqrt{\frac{M}{N \log N}}\right)$. The case when $\Delta_{u^*} - \bar{\Delta}_0 \geq O\left(\sqrt{\frac{M}{N \log N}}\right)$ is not interesting in the context of heterogeneous ranking since the difference between users essentially does not exist. On the other hand, as discussed in Section 5.2 that when $M = o\left(\sqrt{N}\right)$, the proposed Ada-IIR algorithm performs ranking in an adaptive way and is comparably efficient as if the best user were known. While for two-stage ranking to achieve the same complexity level, the value of α needs to be properly chosen. If α is set as a pre-determined constant then $\mathcal{C}_{\text{tsr}}(N, M)$ can be larger than $\mathcal{C}_{\text{Alg}}(N, M)$ by a constant multiplicative factor. Moreover, since we have no access of Δ_{u^*}, if α is chosen to be larger than Δ_{u^*} then the user-selection process can not provide any benefit any more. However, two-stage ranking has a slightly milder constraint on M. While Ada-IIR would make roughly the same amount of queries as the non-adaptive baseline $\mathcal{C}_{\text{ave}}$ if $M = \Omega\left(\sqrt{N}\right)$, two-stage ranking can still be more efficient than the non-adaptive baseline when $M = \Theta\left(\sqrt{N}\right)$.

B.3 User Selection in a Subset

As shown in Proposition 11, when M is much larger than $N \log N$, even querying each user once costs time linear in M which could be higher than the ranking complexity. Therefore, instead of selecting a global α-optimal user, we devise a subroutine Subset-User-Selection (SUS) that randomly picks without replacement L ($L \leq M$) users and only search for an α-optimal user among them (see Algorithm 8). We use $\mathcal{L}$ to denote this L-subset of users.

Algorithm 8 Subroutine: SUBSET-USER-SELECTION($\mathcal{U}, L, \alpha, \delta_i, \delta_m, i, j$)

input: set of users $\mathcal{U}$, user subset size L, desired near-optimal level α, confidence level δ_i of initial ranking, confidence level δ_m of user selection, two items $i, j \in \mathcal{N}$.
$[i', j'] \leftarrow$ Iterative-Insertion-Ranking($\{i, j\}, \delta_i, \bar{u}$).
Randomly choose a subset $\mathcal{L}$ of L users from $\mathcal{U}$.
output: Median-Elimination($\mathcal{L}, \alpha, \delta_m, [i', j']$)

The main procedure of the two-stage algorithm is also modified, shown in Algorithm 9.

Algorithm 9 MODIFIED-TWO-STAGE-RANKING($\mathcal{N}, \mathcal{U}, L, \alpha, \delta_i, \delta_m, \delta_r$)

input: set of items $\mathcal{N}$, set of users $\mathcal{U}$, user subset size L, desired near-optimal level α, confidence level δ_i of initial ranking, confidence level δ_m of user selection, confidence level δ_r of final ranking.
Let i, j be two arbitrary items.
$[i', j'] \leftarrow$ Iterative-Insertion-Ranking($\{i, j\}, \delta_i, \bar{u}$).
Randomly choose a subset $\mathcal{L}$ of L users from $\mathcal{U}$.
$u_\alpha \leftarrow$ Median-Elimination($\mathcal{L}, \alpha, \delta_m, [i', j']$)
output: Iterative-Insertion-Ranking($\mathcal{N}, \delta_r, u_\alpha$)

Generally, no guarantee can be made on how close is a subset α-optimal user to the global optimal user. So analysis on the two-stage algorithm will be done under the assumption that the M user accuracies are iid samples drawn from a probability distribution $F(x)$ over the interval $(0, \frac{1}{2}]$ ($F(x)$ is independent of any other quantities). Let $b = \inf_x\{x : F(x) = 1\}$.

Since $\Delta_1, \Delta_2, \ldots, \Delta_M$ are iid samples from $F(x)$ and $\mathcal{L}$ is drawn randomly, we assume without loss of generality that $\mathcal{L}$ contains the first L users, i.e., $\mathcal{L} = \{1, 2, \ldots, L\}$. Let $\Delta^\circ = \max_{u \in \mathcal{L}} \Delta_u$. Recall that $\Delta_{u^*} = \max_{u \in \mathcal{U}} \Delta_u$. We first show in the following lemma that $\Delta_{u^*} - \Delta^\circ$ is independent of M.

Lemma 15. For any $\delta' \in (0, \frac{1}{2}), \alpha \in (0, b)$, if $L \geq \log(\delta')/\log(F(b - \alpha))$, then with probability at least $1 - \delta'$,

$$\Delta^\circ \geq \Delta_{u^*} - \alpha.$$

Proof. Note that the claim becomes trivial when $M \leq \frac{\log \delta'}{\log(F(b-\alpha))}$. We consider the case when $\frac{\log \delta'}{\log(F(b-\alpha))} \leq L \leq M$.

Since $\Delta_1, \Delta_2, \ldots, \Delta_L$ are iid samples from $F(x)$, with probability $(F(b - \alpha))^L$,

$$\Delta_i \leq b - \alpha \text{ for all } 1 \leq i \leq L.$$

Hence, $(F(b-\alpha))^L \leq \delta'$ gives

$$\Delta^\circ = \max_{1\leq i\leq L} \Delta_i \geq b-\alpha \geq \Delta_{u^*} - \alpha$$

with probability at least $1-\delta'$, where the last inequality follows from $\Delta_{u^*} \leq b$ with probability 1. $\square$

The preceding lemma states that when user accuracies follow a fixed distribution, at least one of the L users we select randomly will be close to the global best user as long as L is large enough (but still independent of M). Thus, even when the number of users M is huge, we do not need to collect information from every one of them. A randomly chosen subset is able to accurately reflect the statistics of the larger group.

Next, we compute the number of comparisons needed for user selection. Our goal is to show that the complexity of user selection becomes negligible compared with item ranking. In the following analysis, for simplification, we assign the confidence levels $\delta_i, \delta_m, \delta_r$ in Two-Stage-Ranking as well as the confidence level δ' for the existence of an α-optimal user equal values. Specifically, we let $\delta' = \delta_i = \delta_m = \delta_r = \frac{\delta}{4}$ for some $\delta \in (0,1)$.

Theorem 16. For any $\delta \in (0, \frac{1}{2}), \alpha \in (0, b), L = \min\left(\lceil \frac{\log(\delta/4)}{\log(F(b-\alpha/2))} \rceil, M\right)$, with probability at least $1-\frac{3\delta}{4}$, subroutine Subset-User-Selection$(\mathcal{U}, L, \frac{\alpha}{2}, \frac{\delta}{4}, \frac{\delta}{4}, i, j)$ outputs a global α-optimal user after

$$\Theta\left(\bar{\Delta}^{-2}\left(\log\log\bar{\Delta}^{-1} + \log\frac{4}{\delta}\right) + \frac{4L}{\alpha^2}\log\frac{4}{\delta}\right)$$

comparisons.

Proof. By Lemma 15, letting $L = \min\left(\lceil \frac{\log(\delta/4)}{\log(F(b-\alpha/2))} \rceil, M\right)$ gives $\Delta^\circ \geq \Delta_{u^*} - \frac{\alpha}{2}$ with probability at least $1-\frac{\delta}{4}$. Moreover, IIR finds the correct order of items i, j with probability at least $1-\frac{\delta}{4}$ and given that Median-Elimination outputs an $\frac{\alpha}{2}$-optimal user in the L-subset with probability at least $1-\frac{\delta}{4}$. Therefore, by the union bound, with probability $1-\frac{3\delta}{4}$, the $\frac{\alpha}{2}$-optimal user found is a global α-optimal user.

The complexity is a sum of two terms: the complexity of IIR ranking two items and the complexity of Median-Elimination outputting an $\alpha/2$-optimal user among L users. $\square$

Theorem 17. For any $\delta \in (0, \frac{1}{2}), \alpha \in (0, b), L = \min\left(\lceil \frac{\log(\delta/4)}{\log(F(b-\alpha/2))} \rceil, M\right)$, with probability at least $1-\delta$, Modified-Two-Stage-Ranking$(\mathcal{N}, \mathcal{U}, L, \alpha, \frac{\delta}{4}, \frac{\delta}{4}, \frac{\delta}{4})$ outputs the exact ranking of $\mathcal{N}$, and consumes

$$\mathcal{C}_{\mathrm{mtsr}}(\alpha) = \Theta\left(\bar{\Delta}^{-2}\left(\log\log\bar{\Delta}^{-1} + \log\frac{4}{\delta}\right) + \frac{4L}{\alpha^2}\log\frac{4}{\delta} + NF\left(\Delta_{u^*} - \alpha\right)\right)$$

comparisons.

Proof. Modified-Two-Stage-Ranking being able to output the exact ranking of $\mathcal{N}$ is guaranteed by the algorithm IIR.

It remains to compute the complexity. By Theorem 16, with probability at least $1-\frac{3}{4}\delta$, Subset-User-Selection outputs a global α-optimal user. With a global α-optimal user, IIR outputs the exact ranking of $\mathcal{N}$ after

$$\Theta\left(NF\left(\Delta_{u^*} - \alpha\right)\right),$$

comparisons with probability at least $1-\frac{\delta}{4}$. Therefore, the desired complexity follows from applying the union bound and summing up the complexities of SUS and IIR. $\square$

From the preceding theorem, the modified-two-stage ranking can achieve complexity $\Theta\left(NF\left(\Delta_{u^*} - \alpha\right)\right)$ even when M is much larger than $N \log N$. Specifically, with α chosen as a constant, $L = \min\left(\lceil \frac{\log(\delta/4)}{\log(F(b-\alpha/2))} \rceil, M\right)$ is $\Theta(1)$. Plus that for M sufficiently large, $\bar{\Delta}$ equals the mean of $F(x)$ with probability 1 and thus $\bar{\Delta}^{-2}\left(\log\log\bar{\Delta}^{-1} + \log\frac{4}{\delta}\right) = O(1)$, we have the following proposition.

Proposition 18. When $\alpha = \Theta(1)$,

$$\mathcal{C}_{\mathrm{mtsr}}(\alpha) = \Theta\left(NF(\Delta_{u^*} - \alpha)\right),$$

regardless of how large M is.

Table 2: Sample complexities on HistoryEvent dataset.

Method	IIR	**Ada-IIR**	Two-stage	Oracle
Full dataset	8826439	**6543195**	9736980	2910488
Diverse subset	9548635	**4050986**	9964645	3157220

C Additional Experiments on Synthetic Data

In this section, we provide additional numerical experiments to demonstrate the advantage of our method. First we extend the accuracy of users to be generated in Section 7 to a wider range of parameters: $\gamma_A \in [0.25, 0.5, 1.0], \gamma_B \in [0.5, 1.0, 2.5]$. We also tested the performance of the algorithm when there are larger amount of users as $M = [9, 18, 36]$ and the result with different γ_u configurations are shown in Fig. 3, 4, 5. In each case, a portion of $\frac{1}{3}$ of the users have $\gamma_u = \gamma_B$, and the rest have $\gamma_u = \gamma_A$. Though the original 'Medium Elimination' order optimal, its constant factor penalty is too large to be practical. We turn to use the successive elimination algorithm (Even-Dar et al., 2002) as we did in Algorithm 3 with $\epsilon = 0.15$ to identify the best user in the given result.

When comparing the same γ_u setting with different users such as in Fig 3(c), 4(c), 5(c). The adaptive algorithm has similar performance with the two-stage one but without the overhead when there are smaller amount of items. It also shows when M is increasing, the advantage of adaptive sampling is diminishing compared to the non-adaptive one due to the fact that the queries are spread over more users thus it takes longer to find the better set of more accurate users.

D Additional Experiments on Real-world Data

In this section, we provide additional experiments when the method is applied to crowdsourcing data. We applied Ada-IIR to a crowdsourcing dataset[2], with the results given in Table 2. To clean up the data, we first selected users who provided more than 200 responses, resulting in the "Full dataset", where it can be observed that Ada-IIR reduces the sample complexity of IIR by 26%. To further highlight the capabilities of the algorithm, we created a more diverse dataset by selecting the top 25% and the bottom 25% of the workers from the“Full dataset", resulting in the “Diverse subset". In this case, the sample complexity is almost 58% lower.

E Proof of Main Results

E.1 Query Complexity of the Proposed Algorithm

The following lemmas characterize the performance of each subroutine:

Lemma 19 (Lemma 9 in Ren et al. (2019))**.** For any input pair (i, j) and a set of users $\mathcal{U}$, Algorithm 2 terminates in $\lceil r_{\max} \rceil = \lceil \epsilon^{-2} \log(2/\delta) \rceil$ queries. If $\epsilon \le \bar{\Delta}$, then the returned $\widehat{y}$ indicates the preferable item with probability at least $1 - \delta$.

Lemma 20 (Lemma 10 in Ren et al. (2019))**.** Algorithm 6 returns after $O(\epsilon^2 \log(|S|/\delta)$ queries and, with probability $1 - \delta$, correctly insert or return unsure. Additionally, if $\epsilon \le \bar{\Delta}$, Algorithm 6 will insert correctly with probability $1 - \delta$.

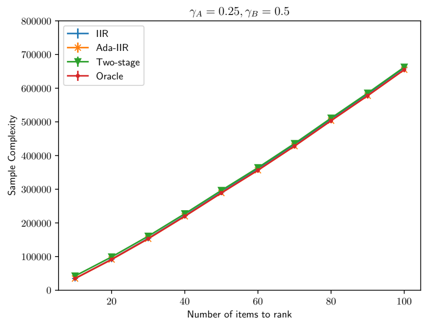
Lemma 21 (Lemma 11 in Ren et al. (2019))**.** With probability $1 - \delta$, Algorithm 5 correctly insert the item and makes $O(\bar{\Delta}^{-2}(\log\log \bar{\Delta}^{-1} + \log(N/\delta)))$ queries at most.

Proof of Theorem 4. When inserting the z-th item, we makes at most $\bar{\Delta}_z^{-2}(\log\log \bar{\Delta}_z^{-1} + \log(N/\delta))$ queries, for $z = 2, 3, \ldots, N$.

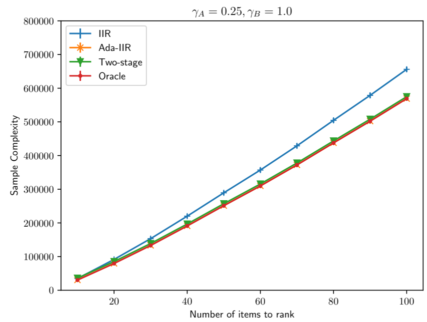
The number of total queries can be obtained by summing up the term above, which is

$$\mathcal{C}_{\text{Alg}}(N) = O\bigg(\sum_{z=2}^{N} \bar{\Delta}_z^{-2}(\log\log \bar{\Delta}_z^{-1} + \log(N/\delta))\bigg).$$

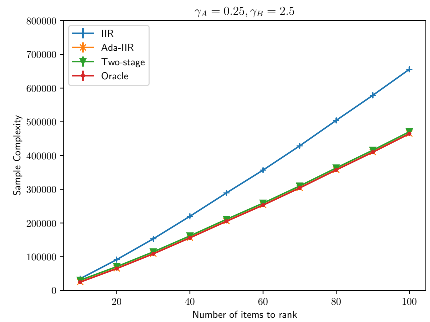
[2] https://doi.org/10.14778/2921558.2921559



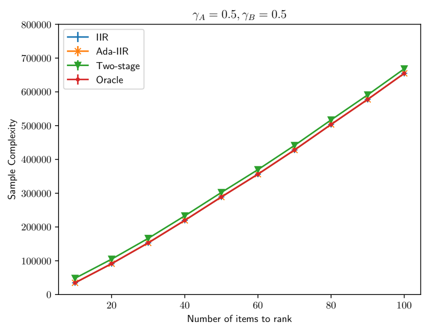
(a) $\gamma_A = 0.25$, $\gamma_B = 0.5$



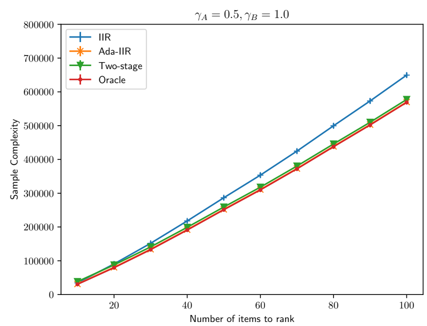
(b) $\gamma_A = 0.25$, $\gamma_B = 1.0$



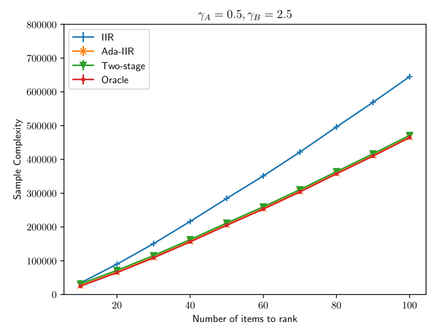
(c) $\gamma_A = 0.25$, $\gamma_B = 2.5$



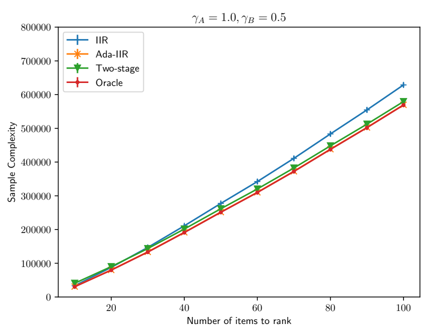
(d) $\gamma_A = 0.5$, $\gamma_B = 0.5$



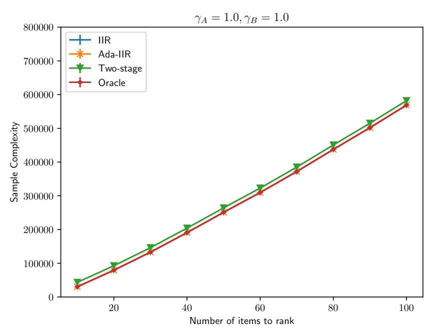
(e) $\gamma_A = 0.5$, $\gamma_B = 1.0$



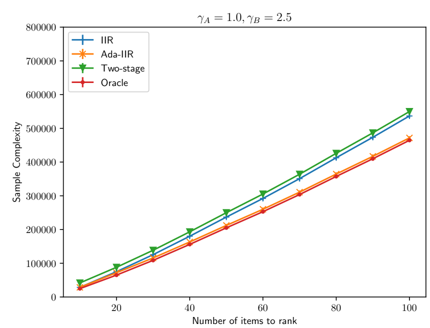
(f) $\gamma_A = 0.5$, $\gamma_B = 2.5$



(g) $\gamma_A = 1.0$, $\gamma_B = 0.5$



(h) $\gamma_A = 1.0$, $\gamma_B = 1.0$



(i) $\gamma_A = 1.0$, $\gamma_B = 2.5$

Figure 3: When $M = 9$. Sample complexities v.s. number of items for all algorithms. The 3-by-3 grid shows different heterogeneous user settings where the accuracy of two group of users differs.

$\square$

E.2 Complexity Gap Analysis

The first lemma we will introduce is about the confidence interval:

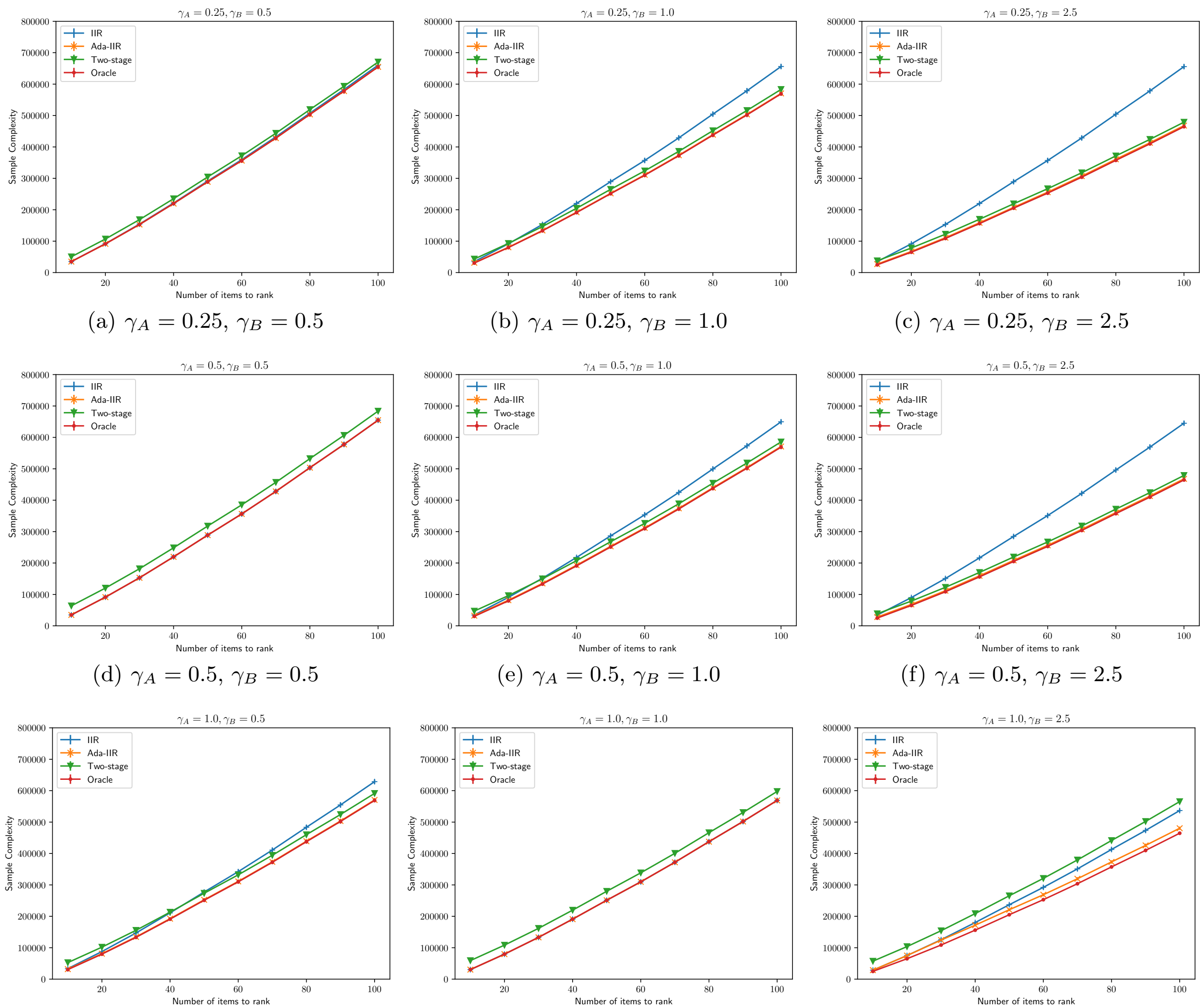
Lemma 22. With probability $1 - \delta$, it holds for any $z \in [N]\backslash\{1\}$ and $u \in \mathcal{U}_z$,

$$\frac{1}{2} + \Delta_u \in \Big[(\mathbf{LCB}_z)_u, (\mathbf{UCB}_z)_u\Big].$$

This also indicates that when inserting the z-th item, for any $u \in \mathcal{U}_z$,

$$\Delta_{u^*} - \Delta_u \leq 4r_z.$$

Proof of Lemma 22. Recall that $(\boldsymbol{\mu}_z)_u$ is the empirical mean of the Bernoulli variable with parameter $\frac{1}{2} + \Delta_u$.



(a) $\gamma_A = 0.25,\ \gamma_B = 0.5$ (b) $\gamma_A = 0.25,\ \gamma_B = 1.0$ (c) $\gamma_A = 0.25,\ \gamma_B = 2.5$

(d) $\gamma_A = 0.5,\ \gamma_B = 0.5$ (e) $\gamma_A = 0.5,\ \gamma_B = 1.0$ (f) $\gamma_A = 0.5,\ \gamma_B = 2.5$

(g) $\gamma_A = 1.0,\ \gamma_B = 0.5$ (h) $\gamma_A = 1.0,\ \gamma_B = 1.0$ (i) $\gamma_A = 1.0,\ \gamma_B = 2.5$

Figure 4: When $M = 18$. Sample complexities v.s. number of items for all algorithms. The 3-by-3 grid shows different heterogeneous user settings where the accuracy of two group of users differs.

For a given z and u, by Hoeffding's inequality we have

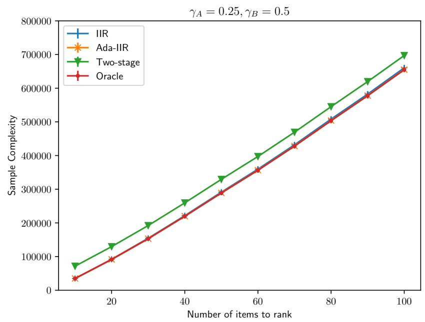
$$\mathbb{P}\left(\left|(\boldsymbol{\mu}_z)_u - \left(\frac{1}{2} + \Delta_u\right)\right| > r_z\right) \le 2e^{-2(\mathbf{s}_z)_u r_u^2} \le 2e^{-2(\mathbf{s}_z)_{\min} r_u^2} \le \frac{\delta}{|\mathcal{U}_z|N},$$

and applying union bound over $z = 2, 3, \ldots, N$ and $u \in \mathcal{U}_z$ gives the claim.

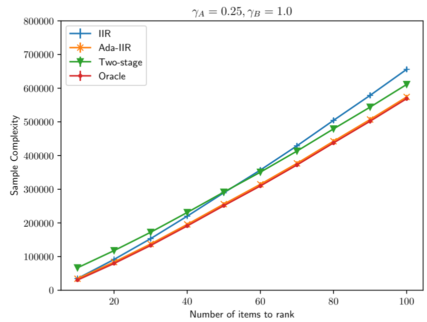
Under this event, we have

$$\begin{aligned}
\Delta_{u^*} - \Delta_u &= \left(\frac{1}{2} + \Delta_{u^*}\right) - \left(\frac{1}{2} + \Delta_u\right) \\
&\le (\mathbf{UCB}_z)_{u^*} - (\mathbf{LCB}_z)_u \\
&\le (\mathbf{UCB}_z)_{u^*} - (\mathbf{LCB}_z)_{u^*} + (\mathbf{UCB}_z)_u - (\mathbf{LCB}_z)_u \\
&= 4r_z,
\end{aligned}$$

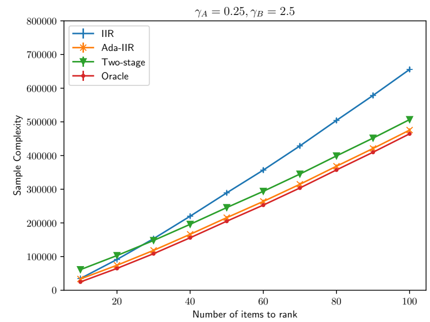
where the first inequality is clearly from the confidence interval, and the second inequality holds because the two confidence intervals should intersect. $\square$



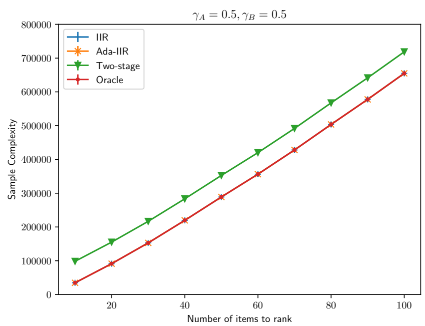
(a) $\gamma_A = 0.25$, $\gamma_B = 0.5$



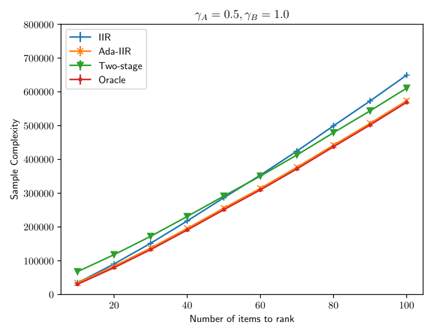
(b) $\gamma_A = 0.25$, $\gamma_B = 1.0$



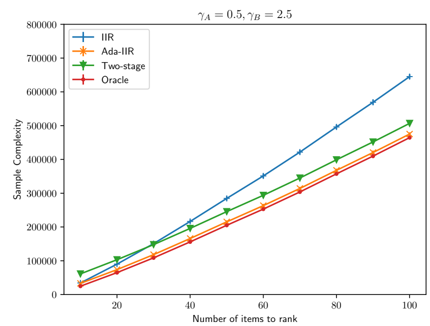
(c) $\gamma_A = 0.25$, $\gamma_B = 2.5$



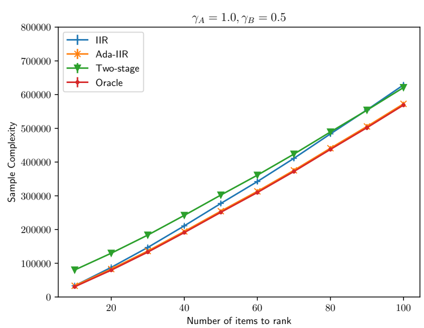
(d) $\gamma_A = 0.5$, $\gamma_B = 0.5$



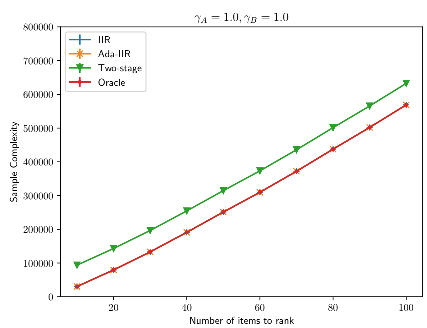
(e) $\gamma_A = 0.5$, $\gamma_B = 1.0$



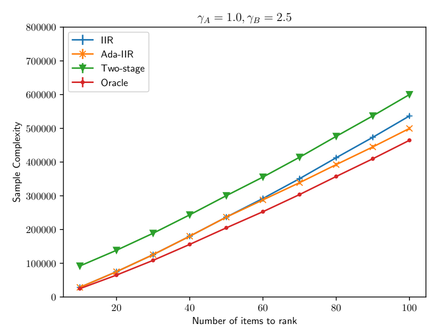
(f) $\gamma_A = 0.5$, $\gamma_B = 2.5$



(g) $\gamma_A = 1.0$, $\gamma_B = 0.5$



(h) $\gamma_A = 1.0$, $\gamma_B = 1.0$



(i) $\gamma_A = 1.0$, $\gamma_B = 2.5$

Figure 5: When $M = 36$. Sample complexities v.s. number of items for all algorithms. The 3-by-3 grid shows different heterogeneous user settings where the accuracy of two group of users differs.

Next, we will introduce another lemma concerning the growth of $(\mathbf{s}_z)_u$ for each $u \in \mathcal{U}_z$.

Lemma 23. Denote S_z as all queries made till inserting the z-th item and $M = |\mathcal{U}_0|$. Suppose $S_z \geq 2M^2 \log(NM/\delta)$. With probability $1-\delta$, we have for any $z \in \{2, 3, \ldots, N\}$,

$$(\mathbf{s}_z)_{\min} \geq \frac{S_z}{2M}.$$

Proof of Lemma 23. For fixed z and $u \in \mathcal{U}_z$, by Hoeffding's inequality we have

$$\begin{aligned}\mathbb{P}\left(\frac{(\mathbf{s}_z)_u}{S_z} - \frac{1}{M} < -\frac{1}{2M}\right) &\leq \mathbb{P}\left(\frac{(\mathbf{s}_z)_u}{S_z} - \mathbb{E}\left[\frac{(\mathbf{s}_z)_u}{S_z}\right] < -\frac{1}{2M}\right)\\ &\leq \exp\left(-\frac{S_z}{2M^2}\right) \leq \frac{\delta}{NM}.\end{aligned}$$

Applying union bound we know that with probability $1-\delta$,

$$(\mathbf{s}_z)_u \geq \frac{S_z}{2M}, \forall z \in \{2, 3, \ldots, N\}, \forall u \in \mathcal{U}_z.$$

Since $(\mathbf{s}_z)_{\min} := \min_{u\in\mathcal{U}_z}(\mathbf{s}_z)_u$, we have

$$(\mathbf{s}_z)_{\min} \geq \frac{S_z}{2M}, \forall z \in \{2,3,\dots,N\}.$$

□

With the two lemmas above, we can control the accuracy gap as follows:

Lemma 24. Denote $\bar{\Delta}_z = \frac{1}{|\mathcal{U}_z|}\sum_{u\in\mathcal{U}_z}\Delta_u$. Suppose $S_z \geq 2|M|^2\log(NM/\delta)$. With probability $1-2\delta$, we have for any $t\in[N]$,

$$\Delta_{u^*} - \bar{\Delta}_z \leq \mathrm{polylog}(N, M, \delta^{-1}) \cdot \sqrt{\frac{M}{S_z}}.$$

Proof of Lemma 24. The proof has two steps:

From Lemma 23 we know that with probability $1-\delta$,

$$(\mathbf{s}_z)_{\min} \geq \frac{S_z}{2M}, \forall t \in [N], \forall u \in \mathcal{U}_z.$$

From Lemma 22, we know with probability $1-\delta$(recall that $(\mathbf{r}_z)_u = \sqrt{\frac{\log(2|\mathcal{U}_z|N/\delta)}{2(\mathbf{s}_z)_{\min}}}$),

$$\begin{aligned}
\Delta_{u^*} - \Delta_u &\leq 4r_z \\
&\leq 4\sqrt{\frac{M\log(2MN/\delta)}{S_z}} \\
&= 4\sqrt{\log(2MN/\delta)} \cdot \sqrt{\frac{M}{S_z}}.
\end{aligned}$$

□

Define function $F(x) = x^{-2}(\log\log(x^{-1}) + \log(N/\delta))$ with $x \in (0, 1/2]$. We care about the following term GAP which characterize the query complexity gap between our algorithm and the optimal user.

$$\mathrm{GAP}(N, M, \delta) = \sum_{z=2}^{N} F(\bar{\Delta}_z) - F(\Delta_{u^*}).$$

The following lemma provide a way to linear bound the gap between function values:

Lemma 25. $F(x) = x^{-2}(\log\log(x^{-1}) + \log(N/\delta))$ with $x \in (0, 1/2]$ is a convex function over $(0, 1/2]$, and for any $\Delta \in [a, b]$, we have

$$F(\Delta) - F(b) \leq \frac{F(a) - F(b)}{b - a} \cdot (b - \Delta) = L(a, b) \cdot (b - \Delta).$$

Furthermore, under the event of Lemma 24, for any $z \in [N]$ such that $S_z > 2M^2\log(NM/\delta)$, we have $\bar{\Delta}_z \in [c\Delta_{u^*}^3, \Delta_{u^*}]$ and therefore

$$F(\bar{\Delta}_z) - F(\Delta_{u^*}) \leq \frac{F(c\Delta_{u^*}^3) - F(\Delta_{u^*})}{\Delta_{u^*} - c\Delta_{u^*}^3} \cdot (\Delta_{u^*} - \bar{\Delta}_z) = L(\mathcal{U}_0) \cdot (\Delta_{u^*} - \bar{\Delta}_z).$$

Here we use $L(\mathcal{U}_0) = \frac{F(c\Delta_{u^*}^3) - F(\Delta_{u^*})}{\Delta_{u^*} - c\Delta_{u^*}^3}$ is indeed a instance-dependent factor, with only logarithmic dependent in N and δ^{-1}(in F). c is a global constant and in fact $c = 1/25$.

Proof. Differentiate $F(x)$ twice and it can be verified that $F''(x) > 0$. For any $\Delta \in [a, b]$, the inequality above is easy to prove via convexity.

The rest is to prove that $\forall t \in [N]$, we have $\bar{\Delta}_z \in [\Delta_{u^*}/M, \Delta_{u^*}]$. It is clear that the upper bound holds because $\Delta_{u^*} := \max_{u \in \mathcal{U}_0} \Delta_u$.

The lower bound is proved as follows: We still have $\bar{\Delta}_z > \Delta_{u^*}/M$ because at any time u^* always remains in the user set and by the assumption $\Delta_u > 0$.

Also, since $S_z > 2M^2 \log(NM/\delta)$, by Lemma 24, we have

$$\begin{aligned}
\Delta_{u^*} - \bar{\Delta}_z &\leq 4\sqrt{\frac{M \log(2MN/\delta)}{S_z}} \\
&\leq 4\sqrt{\frac{M \log(2MN/\delta)}{2M^2 \log(NM/\delta)}} \\
&\leq \frac{4}{\sqrt{M}}.
\end{aligned}$$

Now we will prove that

$$\max\left\{\frac{\Delta_{u^*}}{M}, \Delta_{u^*} - \frac{4}{\sqrt{M}}\right\} \geq c\Delta_{u^*}^3.$$

Suppose $\frac{\Delta_{u^*}}{M} < c\Delta_{u^*}^3$, then we have $M > c^{-1}\Delta_{u^*}^{-2}$, this means

$$\Delta_{u^*} - \frac{4}{\sqrt{M}} \geq \Delta_{u^*} - 4\sqrt{c}\Delta_{u^*} \geq c\Delta_{u^*}^3.$$

The last inequality is due to $\Delta_{u^*} \leq 1/2$ and $c = 1/25$. $\square$

Now we are ready to prove the main result:

Proof of Theorem 5. Based on our algorithmic design, we will not eliminate any user until the cumulative number of queries S_z reach the threshold $S_z \geq 2M^2 \log(NM/\delta)$. We have

$$\begin{aligned}
\mathrm{GAP}(N, M, \delta) &= \sum_{z=2}^{N} F(\bar{\Delta}_z) - F(\Delta_{u^*}) \\
&= \underbrace{\sum_{z=2}^{N} \mathbb{1}\{S_z < 2M^2 \log(NM/\delta)\}\big(F(\bar{\Delta}_z) - F(\Delta_{u^*})\big)}_{I_1} \\
&\quad + \underbrace{\sum_{z=2}^{N} \mathbb{1}\{S_z \geq 2M^2 \log(NM/\delta)\}\big(F(\bar{\Delta}_z) - F(\Delta_{u^*})\big)}_{I_2}.
\end{aligned}$$

For I_1, no elimination is performed, so $\mathcal{U}_z = \mathcal{U}_0$, and we have

$$I_1 = \sum_{z=2}^{N} \mathbb{1}\{S_z < 2M^2 \log(NM/\delta)\}\big(F(\bar{\Delta}_0) - F(\Delta_{u^*})\big).$$

For each term in I_2, we have $F(\bar{\Delta}_z) - F(\Delta_{u^*}) \leq L(\mathcal{U}_0) \cdot 4\sqrt{\log(2MN/\delta)} \cdot \sqrt{\frac{M}{S_z}}$ due to Lemma 25 and Lemma 24. Therefore,

$$I_2 \leq L(\mathcal{U}_0) 4\sqrt{\log(2MN/\delta)} \sum_{z=2}^{N} \mathbb{1}\{S_z \geq 2M^2 \log(NM/\delta)\}\sqrt{\frac{M}{S_z}}.$$

$\square$

E.3 Proof and Discussions of Proposition 6

Suppose $M = o(N^{1/2})$, since $S_z \geq z\log(z/\delta) \geq z$(at least one comparison for an item), from (5) we have

$$\sum_{z=2}^{N} \mathbb{1}\left\{S_z < 2M^2\log(NM/\delta)\right\} \leq \sum_{z=2}^{N} \mathbb{1}\left\{z < 2M^2\log(NM/\delta)\right\} = o(N).$$

The third term can be bounded with the fact $\mathbb{1}\left\{z < 2M^2\log(NM/\delta)\right\} \leq 1$,

$$\begin{aligned}
&L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)}\sum_{z=2}^{N}\mathbb{1}\{S_z \geq 2M^2\log(NM/\delta)\}\sqrt{\frac{M}{S_z}} \\
&\leq L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)}\sum_{z=2}^{N}\sqrt{\frac{M}{S_z}} \\
&\leq L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)}\sum_{z=2}^{N}\sqrt{\frac{M}{z}} \\
&\leq 2L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)}\sqrt{MN} \\
&= O(L(\mathcal{U}_0)\sqrt{\log(MN/\delta)}\sqrt{MN}).
\end{aligned}$$

$L(\mathcal{U}_0)$ is actually dominated by the minimal mean accuracy $\min_z \bar{\Delta}_z$ throughout the algorithm. In practice, $L(\mathcal{U}_0)$ is usually a constant, related to all users' accuracy. In the worst theoretical case, $L(\mathcal{U}_0)$ will be dominated by $F(\Delta_{u^*}/M) = \widetilde{O}(M^2)$, which further turns the last term into $\widetilde{O}(M^{5/2}N^{1/2})$, and requires $M = o(N^{1/5})$ so that this term becomes negligible.