Parallel Algorithms for Select and Partition with Noisy Comparisons*

Mark Braverman[†]
Department of Computer
Science, Princeton University
Princeton, NJ, USA
mbraverm@cs.princeton.edu

Jieming Mao
Department of Computer
Science, Princeton University
Princeton, NJ, USA
jiemingm@cs.princeton.edu

S. Matthew Weinberg[‡]
Department of Computer
Science, Princeton University
Princeton, NJ, USA
sethwmw@cs.princeton.edu

ABSTRACT

We consider the problem of finding the k^{th} highest element in a totally ordered set of n elements (SELECT), and partitioning a totally ordered set into the top k and bottom $n - k$ elements (PARTITION) using pairwise comparisons. Motivated by settings like peer grading or crowdsourcing, where multiple rounds of interaction are costly and queried comparisons may be inconsistent with the ground truth, we evaluate algorithms based both on their total runtime and the number of interactive rounds in three comparison models: noiseless (where the comparisons are correct), erasure (where comparisons are erased with probability $1 - \gamma$), and noisy (where comparisons are correct with probability $1/2 + \gamma/2$ and incorrect otherwise). We provide numerous matching upper and lower bounds in all three models. Even our results in the noiseless model, which is quite well-studied in the TCS literature on parallel algorithms, are novel.

Categories and Subject Descriptors

F.1.2 [Computation by Abstract Devices]: Modes of Computation—*Parallelism and concurrency*

General Terms

Theory

*A full version of this paper is available at <http://arxiv.org/abs/1603.04941>

[†]Research supported in part by an NSF CAREER award (CCF-1149888), NSF CCF-1215990, NSF CCF-1525342, a Packard Fellowship in Science and Engineering, and the Simons Collaboration on Algorithms and Geometry.

[‡]Research completed in part while the author was a Microsoft Research Fellow at the Simons Institute for the Theory of Computing.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

Copyright is held by the owner/author(s). Publication rights licensed to ACM.

STOC'16, June 19–21, 2016, Cambridge, MA, USA
ACM. 978-1-4503-4132-5/16/06...\$15.00
<http://dx.doi.org/10.1145/2897518.2897642>

Keywords

Top- K , Noisy Comparisons, Parallel Algorithms, Rank Aggregation, Median Finding

1. INTRODUCTION

Rank aggregation is a fundamental problem with numerous important applications, ranging from well-studied settings such as social choice [CN91] and web search [DKNS01] to newer platforms such as crowdsourcing [CBCTH13] and peer grading [PHC⁺13]. Salient common features among these applications is that in the end, *ordinal* rather than *cardinal* information about the elements is relevant, and a precise fine-grained ordering of the elements is often unnecessary. For example, the goal of social choice is to select the best alternative, regardless of how good it is. In a curved course, the goal of peer grading is to partition assignments into quantiles corresponding to A/B/C/D, etc, regardless of their absolute quality.

Prior work has produced numerous ordinal aggregation procedures (i.e. based on comparisons of elements rather than cardinal evaluations of individual elements) in different settings, and we overview those most relevant to our work in Section 1.1. However, existing models from this literature fail to capture an important aspect of the problem with respect to some of the newer applications; that *multiple rounds of interaction are costly*. In crowdsourcing, for instance, one round of interaction is the time it takes to send out a bunch of tasks to users and wait for their responses before deciding which tasks to send out next, which is the main computational bottleneck. In peer grading, each round of interaction might take a week, and grades are expected to be determined certainly within a few weeks. In conference decisions, even one round of interaction seems to be pushing the time constraints.

Fortunately, the TCS community already provides a vast literature of algorithms with this constraint in mind, under the name of parallel algorithms. For instance, previous work resolves questions like “how many interactive rounds are necessary for a deterministic or randomized algorithm to select the k^{th} element with $O(n)$ total comparisons?” [Val75, Rei81, AKSS86, AA88a, AA88b, BB90]. This line of research, however, misses a different important aspect related to these applications (that is, in fact, captured by most works in rank aggregation), that the comparisons might be erroneous. Motivated by applications such as crowdsourcing and peer grading, we therefore study the round complexity of PARTITION, the problem of partitioning a totally ordered

set into the top k and bottom $n - k$ elements, when comparisons might be erroneous.

Our first results on this front provide matching upper and lower bounds on what is achievable for PARTITION in just one round in three different models of error: noiseless (where the comparisons are correct), erasure (where comparisons are erased with probability $1 - \gamma$), and noisy (where comparisons are correct with probability $1/2 + \gamma/2$ and incorrect otherwise). We provide one-round algorithms using dn comparisons that make $O(n/d)$, $O(n/(d\gamma))$, and $O(n/(d\gamma^2))$ mistakes (a mistake is any element placed on the wrong side of the partition) with high probability in the three models, respectively. The algorithms are randomized and different for each model, and the bounds hold both when d is an absolute constant or a function of n and γ . We provide asymptotically matching lower bounds as well: all (potentially randomized) one-round algorithms using dn comparisons necessarily make $\Omega(n/d)$, $\Omega(n/(d\gamma))$, and $\Omega(n/(d\gamma^2))$ mistakes in expectation in the three models, respectively. We further show that the same algorithms and lower bound constructions are also optimal (up to absolute constant factors) if mistakes are instead weighted by various different measures of their distance to k , the cutoff.¹

After understanding completely the tradeoff between the number of comparisons and mistakes for one-round algorithms in each of the three models, we turn our attention to multi-round algorithms. Here, the results are more complex and can't be summarized in a few sentences. We briefly overview our multi-round results in each of the three models below. Again, *all* of the upper and lower bounds discussed below extend when mistakes are weighted by their distance to the cutoff. We overview the techniques used in proving our results in Section 1.2, but just briefly note here that the level of technicality roughly increases as we go from the noiseless to erasure to noisy models. In particular, lower bounds in the noisy model are quite involved.

Multi-Round Results in the Noiseless Model.

1. We design a 2-round algorithm for PARTITION using n/ε total comparisons that makes $O(n^{1/2+\varepsilon} \text{poly}(\log n))$ mistakes with probability $1 - e^{-\Omega(n)}$, and prove a nearly matching lower bound of $\Omega(\sqrt{n} \cdot \varepsilon^{5/2})$ mistakes, for any $\varepsilon > 0$ (ε may be a constant or a function of n).
2. We design a 3-round algorithm for PARTITION making $O(n \cdot \text{poly}(\log n))$ total comparisons that makes *zero* mistakes with probability $1 - e^{-\Omega(n)}$. It is known that $\omega(n)$ total comparisons are necessary for a 3-round algorithm just to solve SELECT, the problem of *finding* the k^{th} element, with probability $1 - o(1)$ [BB90].
3. We design a 4-round algorithm for PARTITION making $O(n)$ total comparisons that makes *zero* mistakes with probability $1 - e^{-\Omega(n)}$. This matches the guarantee provided by an algorithm of Bollobás and Brightwell for SELECT, but is significantly simpler (in particular, it avoids any graph theory) [BB90].

¹Specifically, if WRONG_i denotes the random variable that is 1 if an algorithm misplaces i and 0 otherwise, we consider measures of the following form, for any choice of c : $\sum_i \text{WRONG}_i |i - k|^c$. For example, $c = 0$ counts the number of mistakes. This is further discussed in Section 2.

Multi-Round Results in the Erasure Model.

1. We design a $O(\log^*(n))$ -round algorithm for PARTITION making $O(n/\gamma)$ total comparisons that makes *zero* mistakes with probability $1 - e^{-\Omega(n)}$.
2. We show that no $o(\log^*(n))$ -round algorithm even for SELECT making $O(n/\gamma)$ total comparisons can succeed with probability $2/3$.

Multi-Round Results in the Noisy Model.

1. We design a 4-round algorithm for PARTITION making $O(n \log n/\gamma^2)$ comparisons that makes *zero* mistakes with high probability (a trivial corollary of our noiseless algorithm).
2. We show that no algorithm even for SELECT making $o(n \log n/\gamma^2)$ comparisons can succeed with probability $2/3$ (in any number of rounds).
3. We design an algorithm for FINDMIN (the special case of SELECT with $k = n$) making $O(n/\gamma^2)$ comparisons that succeeds with probability $2/3$. We also show that no algorithm making $o(n \log n/\gamma^2)$ comparisons can solve FINDMIN with probability $1 - 1/\text{poly}(n)$ (in any number of rounds).

Together, these results tell an interesting story. In one round, one can obtain the same guarantee in the noiseless versus erasure model with an additional factor of $1/\gamma$ comparisons. And one can obtain the same guarantee in the erasure versus noisy model with an additional factor of $1/\gamma$ comparisons. In some sense, this should be expected, because this exactly captures the degradation in information provided by a single comparison in each of the three models (a noiseless comparison provides one bit of information, an erasure comparison provides γ bits of information, and a noisy comparison provides $\Theta(\gamma^2)$ bits of information). But in multiple rounds, everything changes. In four rounds, one can perfectly partition with high probability and $O(n)$ total comparisons in the noiseless model. In the erasure model, one can indeed partition perfectly with high probability and $O(n/\gamma)$ comparisons, but now it requires $\Theta(\log^*(n))$ rounds instead of just 4. Moreover, in the noisy model, any algorithm even solving SELECT with probability $2/3$ requires an $\Omega(\log n/\gamma)$ blow-up in the number of comparisons, in any number of rounds! Note that neither of these additional factors come from the desire to succeed with high probability (as the lower bounds hold against even a $2/3$ success) *nor* the desire to partition every element correctly (as the lower bounds hold even for just SELECT), but just from the way in which interaction helps in the three different models.

While we believe that the story told by our work as a whole provides the “main result,” it is also worth emphasizing independently our results in the noisy model. Our one-round algorithm, for instance, is more involved than its counterparts in the noiseless and erasure models and our analysis uses the theory of biased random walks. Our multi-round lower bounds against SELECT and FINDMIN in the noisy model are the most technical results of the paper, and tell their own interesting story about the difference between FINDMIN and SELECT in the noisy model. To our knowledge, most tight lower bounds known for SELECT come directly from lower bounding FINDMIN. It's surprising that FINDMIN requires $\Theta(\log n)$ fewer comparisons than SELECT to solve with probability $2/3$ in the noisy model.

We proceed now by discussing some related works below, and briefly overviewing our techniques in Section 1.2. We provide some conclusions and future directions in Section 1.3. Our single-round results are discussed in Section 3 and our multi-round results are discussed in Section 4. However, due to space constraints, most proofs are deferred to the appendix in the full version.

1.1 Related Work

Rank aggregation is an enormous field that we can't possibly summarize in its entirety here. Some of the works most related to ours also study PARTITION (sometimes called TOP-K). Almost all of these works also consider the possibility of erroneous comparisons, although sometimes under different models where the likelihood of an erroneous comparison scales with the distance between the two compared elements [CS15, BSC⁺13, Eri13]. More importantly, to our knowledge this line of work either considers settings where the comparisons are exogenous (the designer has no control over which comparisons are queried, she can just analyze the results), or only analyze the query complexity and not the round complexity of designed algorithms. Our results contribute to this line of work by providing algorithms designed for settings like crowdsourcing or peer grading where the designer does have design freedom, but may be constrained by the number of interactive rounds.

There is a vast literature from the parallel algorithms community studying various sorting and selection problems in the noiseless model. For instance, tight bounds are known on the round complexity of SELECT for deterministic algorithms using $O(n)$ total comparisons (it is $\Theta(\log \log n)$) [Val75, AKSS86], and randomized algorithms using $O(n)$ total comparisons (it is 4) [AA88b, AA88a, Rei81, BB90]. Similar results are known for sorting and approximate sorting as well [Col88, AAV86, AKS83, HH81, BT83, BH85, Lei84]. Many of the designed deterministic algorithms provide *sorting networks*. A sorting network on n elements is a circuit whose gates are binary comparators. The depth of a sorting network is the number of required rounds, and the number of gates is the total number of comparisons. Randomized algorithms are known to require fewer rounds than deterministic ones with the same number of total comparisons for both sorting and selecting [AA88a, BB90].

In the noisy model, one can of course take any noiseless algorithm and repeat every comparison $O(\log n/\delta^2)$ times in parallel. To our knowledge, positive results that avoid this simple repetition are virtually non-existent. This is likely because a lower bound of Leighton and Ma [LM00] proves that in fact no sorting network can provide an asymptotic improvement (for complete sorting), and our lower bound (Theorem 13) shows that no randomized algorithm can provide an asymptotic improvement for SELECT. To our knowledge, no prior work studies parallel sorting algorithms in the erasure model. On this front, our work contributes by addressing some open problems in the parallel algorithms literature, but more importantly by providing the first parallel algorithms and lower bounds for SELECT in the erasure and noisy models.

There is also an active study of sorting in the noisy model [BM08, BM09, MMV13] within the TCS community without concern for parallelization, but with concern for *resampling*. An algorithm is said to resample if it makes the same comparison multiple times. Clearly, an algorithm that doesn't

resample can't possibly find the median exactly in the noisy model (what if the comparison between $n/2$ and $n/2 + 1$ is corrupted?). The focus of these works is designing poly-time algorithms to find the maximum-likelihood ordering from a set of $\binom{n}{2}$ noisy comparisons. Our work is fundamentally different from these, as we have asymptotically fewer than $\binom{n}{2}$ comparisons to work with, and at no point do we try to find a maximum-likelihood ordering (because we only want to solve PARTITION).

1.2 Tools and Techniques

Single Round Algorithms and Lower Bounds.

Our single round results are guided by the following surprisingly useful observation: in order for an algorithm to possibly know that i exceeds the k^{th} highest element, i must at least be compared to some element between itself and k (as otherwise, the comparison results would be identical if we replaced i with an element just below k). Unsurprisingly, it is difficult to guarantee that many elements within n/d of k are compared to elements between themselves and k using only dn total comparisons in a single round, and this forms the basis for our lower bounds. Our upper bounds make use of this observation as well, and basically are able to guarantee that an element is correctly placed with high probability whenever it is compared to an element between itself and k . It's interesting that the same intuition is key to both the upper and lower bounds. We provide a description of the algorithms and proofs in Section 3.

In the erasure model, the same intuition extends, except that in order to have a non-erased comparison between i and an element between i and k , we need to make roughly $1/\gamma$ such comparisons. This causes our lower bounds to improve by a factor of $1/\gamma$. In the noisy model, the same intuition again extends, although this time the right language is that we need to learn $\Omega(1)$ bits of information from comparisons of i to elements between i and k , which requires $\Omega(1/\gamma^2)$ such comparisons, and causes the improved factor of $1/\gamma^2$ in our lower bounds. Our algorithms in these two models are similar to the noiseless algorithm, but the analysis becomes necessarily more involved. For instance, our analysis in the noisy model appeals to facts about biased random walks on the line.

Multi-Round Algorithms and Lower Bounds.

Our constant-round algorithms in the noiseless model are based on the following intuition: once we reach the point that we are only uncertain about $o(n)$ elements, we are basically looking at a fresh instance of PARTITION on a significantly smaller input size, except we're still allowed $\Theta(n)$ comparisons per round. Once we're only uncertain about only $O(\sqrt{n})$ elements, one additional round suffices to finish up (by comparing each element to every other one). The challenge in obtaining a four-round algorithm (as opposed to just an $O(1)$ -round algorithm) is ensuring that we make significant enough gains in the first three rounds.

Interestingly, these ideas for constant-round algorithms in the noiseless model don't prove useful in the erasure or noisy models. Essentially the issue is that even after a constant number of rounds, we are unlikely to be confident that many elements are above or below k , so we can't simply recurse on a smaller instance. Still, it is quite difficult to discover a formal barrier, so our multi-round lower bounds for the

erasure and noisy models are quite involved. We refer the reader to Section 4 for further details.

1.3 Conclusions

We study the problems of PARTITION and SELECT in settings where interaction is costly in the noiseless, erasure, and noisy comparison models. We provide matching (up to absolute constant factors) upper and lower bounds for one round algorithms in all three models, which also show that the number of comparisons required for the same guarantee degrade proportional to the information provided by a single comparison. We also provide matching upper and lower bounds for multi-round algorithms in all three models, which also show that the round and query complexity required for the same guarantee in these settings degrades worse than just by the loss in information when moving between the three comparison models. Finally, we show a separation between FINDMIN and SELECT in the noisy model.

We believe our work motivates two important directions for future work. First, our work considers some of the more important constraints imposed on rank aggregation algorithms in applications like crowdsourcing or peer grading, but not all. For instance, some settings might require that every submission receives the same amount of attention (i.e. is a member of the same number of comparisons), or might motivate a different model of error (perhaps where mistakes aren't independent or identical across comparisons). It would be interesting to design algorithms and prove lower bounds under additional restrictions motivated by applications.

Finally, it is important to consider incentives in these applications. In peer grading, for instance, the students themselves are the ones providing the comparisons. An improperly designed algorithm might provide “mechanism design-type” incentives for the students to actively misreport if they think it will boost their own grade. Additionally, there are also “scoring rule-type” incentives that come into play: grading assignments takes effort! Without proper incentives, students may choose to put zero or little effort into their grading and just provide random information. We believe that using ordinal instead of cardinal information will be especially helpful on this front, as it is much easier to design mechanisms when players just make binary decisions, and it's much easier to understand how the noisy information provided by students scale with effort (in our models, it is simply that γ will increase with effort). It is therefore important to design mechanisms for applications like peer grading by building off of our algorithms.

2. PRELIMINARIES AND NOTATION

In this work, we study two problems, SELECT and PARTITION. Both problems take as input a randomly sorted, totally ordered set and an integer k . For simplicity of notation, we denote the i^{th} smallest element of the set as i . So if the input set is of size n , the input is exactly $[n]$. In SELECT, the goal is to output the (location of the) element k . In PARTITION, the goal is to partition the elements into the top k , which we'll call A for Accept and the bottom $n - k$, which we'll call R for Reject. Also for ease of notation, we'll state all of our results for $k = n/2$, the median, w.l.o.g.²

²We show formally in the full version that this is indeed w.l.o.g.

We say an algorithm solves SELECT if it outputs the median, and solves PARTITION if it places correctly all elements above and below the median. For SELECT, we will say that an algorithm is a t -approximation with probability p if it outputs an element in $[n/2 - t, n/2 + t]$ with probability at least p . For PARTITION, we will consider a class of success measures, parameterized by a constant c , and say the c -weighted error associated with a specific partitioning into $A \sqcup R$ is equal to $\sum_{i > n/2} I(i \in R) i^c + \sum_{i < n/2} I(i \in A) i^c$.³ Interestingly, in all cases we study, the same algorithm is asymptotically optimal for all c .

Query and Round Complexity.

Our algorithms will be comparison-based. We study both the number of queries, and the number of adaptive rounds necessary to achieve a certain guarantee.⁴ We may not always emphasize the runtime of our algorithms, but they all run in time $\text{poly}(n)$.

Notation.

We always consider settings where the input elements are a priori indistinguishable, or alternatively, that our algorithms randomly permute the input before making comparisons. When we write $x < y$, we mean literally that $x < y$ in the ground truth. In the noisy model, the results of comparisons may disagree with the underlying ordering, so we say that x beats y if a noisy comparison of x and y returned x as larger than y (regardless of whether or not $x > y$).

Models of Noise.

We consider three comparison models, which return the following when $a > b$.

- Noiseless: Returns a beats b .
- Erasure: Returns a beats b with probability γ , and $\perp$ with probability $1 - \gamma$.
- Noisy: Returns a beats b with probability $1/2 + \gamma/2$, and b beats a with probability $1/2 - \gamma/2$.

PARTITION versus SELECT.

We design all of our algorithms for PARTITION, and prove all of our lower bounds against SELECT. We do this because SELECT is in some sense a strictly easier problem than PARTITION. We discuss how one can get algorithms for SELECT via algorithms for PARTITION and vice versa formally in the full version.

Resampling.

Finally, note that in the erasure and noisy models, it may be desirable to query the same comparison multiple times. This is called *resampling*. It is easy to see that without resampling, it is impossible to guarantee that the exact median is found with high probability, even when all $\binom{n}{2}$ com-

³For instance, $c = 0$ counts the number of mistakes. $c = 1$ counts the number of mistakes, weighted by the distance of the mistaken element from the median. $c = 2$ is similar to mean-squared-error, etc.

⁴For example, an algorithm that makes Q queries one at a time, waiting for the result of previous queries before deciding which queries to make next has round complexity Q . An algorithm that makes all queries up front, without knowing any results has round complexity 1. We call protocols with round complexity 1 *non-adaptive*.

parisons are made (what if the comparison between $n/2$ and $n/2 + 1$ is corrupted?). Resampling is not necessarily undesirable in the applications that motivate this work, so we consider our main results to be in the model where resampling is allowed. Still, it turns out that all of our algorithms can be easily modified to avoid resampling at the (necessary) cost of a small additional error, and it is easy to see the required modifications.⁵ All of our lower bounds hold even against algorithms that resample.

3. RESULTS FOR NON-ADAPTIVE ALGORITHMS

In this section, we provide our results on non-adaptive (round complexity = 1) algorithms. We begin with the upper bounds below, followed by our matching (up to constant factors) lower bounds.

3.1 Upper Bounds

We provide asymptotically optimal algorithms in each of the three comparison models. Our three algorithms actually choose the same comparisons to make, but determine whether or not to accept or reject an element based on the resulting comparisons differently. The algorithms pick a *skeleton set* S of size $\sqrt{n}$ and compare every element in S to every other element. Each element not in S is compared to $d - 1$ random elements of S .

From here, the remaining task in all three models is similar: the algorithm must first estimate the rank of each element in the skeleton set. Then, for each i , it must use this information combined with the results of $d - 1$ comparisons to guess whether i should be accepted or rejected. The correct approach differs in the three models, which we discuss next.

Noiseless Model.

Algorithm 1 Non-adaptive procedure for querying dn comparisons

- 1: Select a *skeleton set*, S , of size $\sqrt{n}$ (without replacement).
 - 2: Compare each element of S to each other element of S .
 - 3: Compare each element not in S to $d - 1$ random elements of S .
-

Algorithm 2 Non-adaptive algorithm for the noiseless model

- 1: Run Algorithm 1. Let S denote the skeleton set selected, and x denote the median of S .
 - 2: Denote by A_S the subset of S that beat x . Denote by $R_S = S - A_S$.
 - 3: For all $i \in S$, accept i iff $i \in A_S$. Otherwise, reject.
 - 4: For all $i \notin S$, if i beats an element in A_S , accept. If an element in R_S beats i , reject. Otherwise, make a random decision for i .
-

Pseudocode for our algorithm in the noiseless model is provided as Algorithm 2. First, we estimate that the median of the skeleton set, x , is close to the actual median.

⁵Essentially, replace all resampled comparisons with comparisons to “nearby” elements instead.

Then, we hope that each $i \notin S$ is compared to some element in S between itself and x . If this happens, we can pretty confidently accept or reject i . If it doesn’t, then all we learn is that i is beaten by some elements above x and it beats some elements below x , which provides no helpful information about whether i is above or below the median, so we just make a random decision.

THEOREM 1. *Algorithm 2 has query complexity dn , round complexity 1, does not resample, and outputs a partition that, for all c , has:*

- *expected c -weighted error $O((n/d)^{c+1})$, for any $d = o(n^{1/4})$*
- *c -weighted error $O((n/d)^{c+1})$ with probability $1 - e^{-\Omega(n^3/d^{2c+2})}$, for any $d = o(n^{1/4})$.*

We first provide some intuitions before we go to the actual proof. There are two sources of potential error in Algorithm 2. First, maybe the skeleton set is poorly chosen and not representative of the ground set. But this is extremely unlikely with such a large skeleton set. Second, note that if i is compared to any element in S between itself and x , and x is very close to $n/2$, then i will be correctly placed. If $|i - n/2| > n/d$, then we’re unlikely to miss this window on $d - 1$ independent tries, and i will be correctly placed.

PROOF OF THEOREM 1. We consider the error contributed by $n/2 + i$ (which is the same as $n/2 - i$). There are two events that might cause $n/2 + i$ to be misplaced. First, maybe $n/2 + i$ loses to y for some $y \in S, y < x$. This is unlikely because this can only happen in the event that x is a poor approximation to the median. Second, maybe $n/2 + i$ is never compared to an element $y \in [x, n/2 + i]$. This is also unlikely because the fraction of such elements should be about i/n . We bound the probability of the former event first.

LEMMA 1. *Let X_i denote the fraction of elements in S greater than $n/2 + i$. Then $X_i \leq 1/2 - i/n + \varepsilon$ with probability at least $1 - 2e^{-\varepsilon^2 \sqrt{n}/2}$.*

PROOF. X_i is the average of $\sqrt{n}$ independent random variables, each denoting whether $n/2 + i < \text{some } y \in S$. $E[X_i] = 1/2 - i/n$. Applying Lemma 4 in full version yields the lemma. $\square$

We call a skeleton set S is good if for $i = 1, \dots, n/2$, $X_i \leq 1/2 - i/n + \varepsilon$. Now let’s fix the skeleton set S and assume S is good. From the above lemma, we know S is good with probability at least $1 - n \cdot e^{-\varepsilon^2 \sqrt{n}/2}$.

LEMMA 2. *If S is good, then the probability that $n/2 + i$ is rejected is at most $(1 - i/n + \varepsilon)^{d-1}$ (and at most 1 if $i/n \leq \varepsilon$).*

PROOF. The elements that $n/2 + i$ are compared to are chosen uniformly at random from S . At least a $1/2 + i/n - \varepsilon$ fraction of them are less than $n/2 + i$, so an $i/n - \varepsilon$ fraction of elements in S lie in $[x, n/2 + i]$. So each time we choose a random element of S to compare to $n/2 + i$, we have at least a $\max\{i/n - \varepsilon, 0\}$ chance of comparing to an element in $[x, n/2 + i]$. In the event that this happens, we are guaranteed to accept $n/2 + i$. The probability that we miss on each of $d - 1$ independent trials is exactly $(1 - i/n + \varepsilon)^{d-1}$. $\square$

Therefore, conditioning that S is good, the expected c -weighted error contributed by elements $n/2 + 1, \dots, n$ is at most :

$$\begin{aligned}
& \sum_{i=1}^{n/2} \min\{1, (1 - i/n + \varepsilon)\}^{d-1} i^c \\
& \leq \sum_{i=1}^{\varepsilon n} i^c + \sum_{i=\varepsilon n}^{n/2} (1 - i/n + \varepsilon)^{d-1} i^c \\
& \sum_{i=\varepsilon n}^{n/2} (1 - i/n + \varepsilon)^{d-1} i^c \leq \sum_{i=\varepsilon n}^{n/2} e^{(d-1)(-i/n + \varepsilon)} i^c \\
& \leq e^{d\varepsilon} \sum_{i=1}^{n/2} e^{-(d-1)i/n} i^c \\
& \leq 2e^{d\varepsilon} c!(n/(d-1))^{c+1}
\end{aligned}$$

The last inequality is proved in the full version (the proof is just several lines of algebra). Taking $\varepsilon = 1/d$ shows that the expected c -weighted error contributed by elements $n/2 + 1, \dots, n$ is $\frac{(n\varepsilon)^{c+1}}{c+1} + \frac{2n^{c+1}e^{d\varepsilon}c!}{(d-1)^{c+1}}$ conditioned on S is good, which is $O((n/d)^{c+1})$. Notice that when S is fixed, the c -weighted error contributed by each element is independent and bounded by n^c . By the Hoeffding bound, the probability that the c -weighted error exceeds its expectation by more than $(n/d)^{c+1}$ is at most $e^{-2n^3/d^{2c+2}}$ conditioned on S is good. To sum up, taking elements that are smaller than the median into account, we know that with probability at least $1 - 2n \cdot e^{-\sqrt{n}/(2d^2)} - 2e^{-2n^3/d^{2c+2}}$, the c -weighted error is $O((n/d)^{c+1})$.

Here we also show that our algorithm is better than a simpler solution. The simpler solution would be to just compare every element to a random d elements, and accept if it is larger than at least half, and reject otherwise. We show that, unlike the algorithm above, this doesn't obtain asymptotically optimal error as d grows.

THEOREM 2. *The simple solution has expected error $\Omega(n^{c+1}/d^{(c+1)/2})$.*

PROOF. Let X_{ij} be an indicator variable for the event that $n/2 + i$ is smaller than the j^{th} element it is compared to. Then $n/2 + i$ is accepted iff $\sum_j X_{ij} > d/2$. As each X_{ij} is a bernoulli random variable that is 1 with probability $1/2 - i/n$, the probability that $n/2 + i$ is mistakenly rejected is exactly the probability that a $B(d, 1/2 - i/n)$ random variable exceeds its expectation by at least di/n . This happens with probability on the order of $e^{-(i/n)^2 d}$. So for $i = O(n/\sqrt{d})$, this is at least $1/e$, meaning that the error contribution from all $n/2 + i$, $i \leq kn/\sqrt{d}$ for some absolute constant k is at least i^c/e . Summing over all i means that the total error is at least $\sum_{i=1}^{kn/\sqrt{d}} i^c/e = \Omega(n^{c+1}/d^{(c+1)/2})$. $\square$

Erasure Model.

In the erasure model, pseudocode for the complete algorithm we use is Algorithm 3. At a high level, the algorithm is similar to Algorithm 2 for the noiseless model.

THEOREM 3. *Algorithm 3 has query complexity dn , round complexity 1, does not resample, and outputs a partition that, for all c , has:*

- *expected c -weighted error $O((n/(d\gamma))^{c+1})$, for any d, γ such that $d/\gamma = o(n^{1/4})$*
- *c -weighted error $O((n/(d\gamma))^{c+1})$ with probability $1 - e^{-\Omega(n^3/d^{2c+2})}$, whenever $d/\gamma = o(\sqrt{n})$ and $d\gamma = o(n^{1/4})$.*

The additional ingredient beyond the noiseless case is a proof that with high probability, not too many of the comparisons within S are erased and therefore while we can't learn the median of S exactly, we can learn a set of almost $|S|/2$ elements that are certainly above the median, and almost $|S|/2$ elements that are certainly below. If $i \notin S$ beats an element that is certainly above the median of S , we can confidently accept it, just like in the noiseless case.

Algorithm 3 Non-adaptive algorithm for the erasure model

- 1: Run Algorithm 1. Let S denote the skeleton set selected.
- 2: Say that element $a \in S$ is *known to beat* element $b \in S$ if there exists some $a = s_0 > s_1 > \dots > s_\ell = b$ with all $s_j \in S$, and s_j beats s_{j+1} for all $j \in \{0, \dots, \ell-1\}$ (i.e. all of these comparisons were not erased).
- 3: Denote by A_S the elements of S that are known to beat at least $|S|/2$ elements of S . Denote by R_S the elements of S that are known to be beaten by at least $|S|/2$ elements of S (note that S may not equal $A_S \cup R_S$).
- 4: For all $i \in S$, accept i if $i \in A_S$. Reject i if $i \in R_S$. Otherwise, make a random decision for i .
- 5: For all $i \notin S$, if i beats an element in A_S , accept. If an element in R_S beats i , reject. Otherwise, make a random decision for i .

PROOF OF THEOREM 3. We again consider the error contributed by $n/2 + i$ (which is the same as $n/2 - i$). There are again two events that might cause $n/2 + i$ to be misplaced. First, maybe it is beaten an element in R_S . This is unlikely because this can only happen in the event that some element below the median makes it into A_S . Second, maybe i is never compared to an element that beats it in A_S . This is also unlikely because the fraction of such elements should be at least $1/2 - i/n$. We bound the probability of the former event first, making use of Lemma 1.

Again let X_i denote the fraction of elements in S that are smaller than $n/2 + i$, and define S to be good if $X_i \leq 1/2 - i/n + \varepsilon$ for all $i \in [1, n/2]$. Then Lemma 1 guarantees that S is good with probability at least $1 - ne^{-\varepsilon^2 \sqrt{n}/2}$. This time, in addition to S being good, we also need to make sure that A_S is large (which will happen as long as not too many comparisons are erased).

LEMMA 3. *Let $x, y \in S$ such that $x > y$ and $|S \cap (x, y)| = k$. Then with probability at least $1 - e^{-k\gamma^2}$, x is known to beat y .*

PROOF. x is known to beat y if there is some $z \in S \cap (x, y)$ such that the comparisons between x and z and y and z are both not erased. There are k such possible z , and all comparisons are erased independently. So the probability that for all z , at least one of the comparisons to $\{x, y\}$ were erased is $(1 - \gamma^2)^k \leq e^{-k\gamma^2}$. $\square$

COROLLARY 1. *For all $\varepsilon \in (0, 1/2)$, with probability at least $1 - n^2 e^{-\varepsilon^2 \sqrt{n}}$, both A_S and R_S have at least $(1/2 - \varepsilon)|S|$ elements.*

PROOF. By Lemma 3 and a union bound, with probability at least $1 - n^2 e^{-\varepsilon \gamma^2 \sqrt{n}}$, it is the case that for all $x, y \in S$ that have at least $\varepsilon|S|$ elements between them, it is known whether x beats y or vice versa. In the event that this happens, any element that is at least $\varepsilon|S|$ elements away from the median will be in A_S or R_S . $\square$

We'll call a skeleton set S really good if it is good, and $|A_S| \geq (1/2 - \varepsilon)|S|$. Now let's fix the skeleton set S and assume S is really good. From the above arguments, we know S is really good with probability at least $1 - n \cdot e^{-\varepsilon^2 \sqrt{n}/2} - n^2 e^{-\varepsilon \gamma^2 \sqrt{n}}$.

Next, observe that if $X_i \leq 1/2 - i/n + \varepsilon$, and $|A_S| \geq (1/2 - \varepsilon)|S|$, then there are at least $(i/n - 2\varepsilon)|S|$ elements in A_S less than $n/2 + i$. Therefore the probability that $n/2 + i$ never beats an element in A_S is at most $(1 - (i/n - 2\varepsilon)\gamma)^{d-1}$ (and at most 1 if $i/n \leq 2\varepsilon$). So the total c -weighted error that comes from these cases is at most $\min\{1, (1 - (i/n - 2\varepsilon)\gamma)^{d-1}\} i^c$.

Conditioning on S is really good, the expected c -weighted error is

$$\begin{aligned} & \sum_{i=1}^{n/2} \min\{1, (1 - (i/n - 2\varepsilon)\gamma)^{d-1}\} i^c \\ & \leq \sum_{i=1}^{2\varepsilon n} i^c + \sum_{i=2\varepsilon n}^{n/2} (1 - (i/n - 2\varepsilon)\gamma)^{d-1} i^c \\ & \sum_{i=2\varepsilon n}^{n/2} (1 - (i/n - 2\varepsilon)\gamma)^{d-1} i^c \leq \sum_{i=2\varepsilon n}^{n/2} e^{(d-1)\gamma(-i/n+2\varepsilon)} i^c \\ & \leq e^{2d\varepsilon\gamma} \sum_{i=1}^{n/2} e^{-(d-1)\gamma i/n} i^c \\ & \leq \frac{e^{2d\varepsilon\gamma} c! n^{c+1}}{(\gamma(d-1))^{c+1}} \end{aligned}$$

Taking $\varepsilon = 1/(\gamma d)$ shows that the expected c -weighted error contributed by elements $n/2 + 1, \dots, n$ is $n^{c+1} \left(\frac{(2\varepsilon)^{c+1}}{c+1} + \frac{e^{2d\varepsilon\gamma} c!}{(\gamma(d-1))^{c+1}} \right) = O((n/(\gamma d))^{c+1})$ conditioned on S is really good. Notice that when S is fixed, the c -weighted error contributed by each element is independent and bounded by n^c . By the Hoeffding bound, the probability that the c -weighted error exceeds its expectation by more than $(n/(\gamma d))^{c+1}$ is at most $e^{-2n^3/(\gamma d)^{2c+2}}$, conditioned on S is really good. To sum up, taking elements that are smaller than the median into account, we know that with probability at least $1 - 2n \cdot e^{-\sqrt{n}/(2\gamma^2 d^2)} - 2n^2 e^{-\gamma \sqrt{n}/d} - 2e^{-2n^3/(\gamma d)^{2c+2}}$, the c -weighted error is $O((n/(\gamma d))^{c+1})$.

Noisy Model.

Pseudocode for our algorithm in the noisy model is provided as Algorithm 4. Algorithm 4 is necessarily more involved than the previous two. We can still recover a good ranking of the elements in the skeleton set using the Braverman-Mossel algorithm [BM08], so this isn't the issue. The big difference between the noisy model and the previous two is that no single comparison can guarantee that $i \notin S$ should be accepted or

rejected. Instead, every time we have a set of elements all above the median of S , x , of which i beats at least half, this provides some evidence that i should be accepted. Every time we have a set of elements all below x of which i is beaten by at least half, this provides some evidence that i should be rejected. The trick is now just deciding which evidence is stronger. Due to space constraints, we refer the reader to Algorithm 4 to see our algorithm, which we analyze using theory from biased random walks on the line.

THEOREM 4. *Algorithm 4 has query complexity dn , round complexity 1, does not resample, and outputs a partition that, for all c , has:*

- *expected c -weighted error $O((n/(d\gamma^2))^{c+1})$, for any $d = o(n^{1/4})$, $\gamma = \omega(n^{1/8})$.*
- *c -weighted error $O((n/(d\gamma^2))^{c+1})$ with probability $1 - e^{\Omega(n^3/d^{2c+2})}$, for any $d = o(n^{1/4})$, $\gamma = \omega(n^{1/8})$.*

Algorithm 4 Non-adaptive algorithm for the noisy model

- 1: Run Algorithm 1. Let S denote the skeleton set selected.
 - 2: Run the Braverman-Mossel algorithm to recover the maximum likelihood ordering of elements in S in time $\text{poly}(n, 1/\gamma)$ [BM08]. Let x denote the median under this ordering, and denote by $y <_S z$ that y comes before z in this ordering.
 - 3: For all $i \in S$, accept i iff $i >_S x$, otherwise reject.
 - 4: For all $i \notin S$, let $b_{i1} <_S \dots <_S b_{ic}$ denote the elements $>_S x$ that i is compared to (note that maybe $c = 0$), and $\ell_{i1} >_S \dots >_S \ell_{if}$ denote the elements $<_S x$ that i is compared to.
 - 5: Let $X_{ij} = 1$ iff i beats b_{ij} , and -1 otherwise. Let $Y_{ij} = 1$ iff i beats ℓ_{ij} and -1 otherwise. Let $B_i = \max\{J | \sum_{j=1}^J X_{ij} \geq 0\}$ and $L_i = \max\{J | \sum_{j=1}^J Y_{ij} \leq 0\}$. If $B_i > L_i$, accept i . If $B_i < L_i$, reject. If $B_i = L_i$, make a random decision.
-

PROOF OF THEOREM 4. Again we consider the probability of mistakenly accepting the element $n/2 + i$. This can happen for the same reasons as in the previous sections: perhaps the elements selected for S are not representative of the ground set, or $n/2 + i$ is not compared to elements that separate it from the median. Additionally, mistakes may now happen due to erroneous comparisons, even when $n/2 + i$ is compared to "the right" elements. We proceed now to bound the probability of each bad event, beginning with the event that S is poorly sorted due to comparison errors.

THEOREM 5. ([BM08]) *For each element $e \in S$, let $\pi(e)$ denote the true rank of e when S is properly sorted (i.e. the best element in S has rank 1), and let $\sigma(e)$ denote the rank of e in the maximum likelihood ordering after taking all $\binom{|S|}{2}$ pairwise comparisons. Then there is an absolute constant K such that for all ε and for all e simultaneously, with probability at least $1 - 2\varepsilon$, $|\pi(e) - \sigma(e)| \leq K\gamma^{-4}(\log |S| - \log \varepsilon)$. In particular, taking $\varepsilon = 1/\text{poly}(n)$ and recalling that $|S| = \sqrt{n}$ implies that with probability at least $1 - 2/\text{poly}(n)$, $|\pi(e) - \sigma(e)| = O(\log(n)/\gamma^4) = o(|S|)$.*

Now, we want to combine this with Lemma 1 to claim that there are not only many elements in S smaller than $n/2 + i$, but that these elements are sorted well enough in S for this

to be useful. Recall that Lemma 1 states that for all ε , with probability at least $1 - e^{-\varepsilon^2 \sqrt{n}/2}$, the fraction of elements in S larger than $n/2 + i$ is at most $1/2 - i/n + \varepsilon$. Combining Lemma 1 with Theorem 5 yields the following:

COROLLARY 2. *For all ε , with probability at least $1 - 2/\text{poly}(n) - e^{-\varepsilon^2 \sqrt{n}/2}$, there is a set $T \subset S$ such that:*

- $|T| \geq (i/n - \varepsilon - o(1))|S|$.
- Every element in T is $>_S x$.
- Every element in T is smaller than $n/2 + i$.
- Every element in T is $<_S y$, for all $y \in S \cap (n/2 + i, n]$.

PROOF. By Lemma 1, there are at least $(1/2 + i/n - \varepsilon)|S|$ elements in S smaller than $n/2 + i$ (except with probability $e^{-\varepsilon^2 \sqrt{n}/2}$). Therefore, by Theorem 5, the maximum likelihood ordering cannot place any element $> n/2 + i$ in any rank below $(1/2 + i/n - \varepsilon - o(1))|S|$ (except with probability $1/n^2$). So simply define T to be the set of elements ranked between $|S|/2$ and $(1/2 + i/n - \varepsilon - o(1))|S|$, and T necessarily satisfies the desired properties (except with the stated probability). $\square$

From here, the idea is that $B_{n/2+i}$ is likely to be large, as the elements in T form a large set immediately above x such that $n/2 + i$ should beat more than half the elements in whatever subset of T it is compared to. Similarly, we expect $L_{n/2+i}$ to be small, because $n/2 + i$ shouldn't lose to a large set of elements below x . We first show that $n/2 + i$ is likely to be compared to many elements of T .

LEMMA 4. *Let T be any set of size $\delta|S|$ elements. Then with probability at least $1 - e^{-\delta(d-1)/8}$, $n/2 + i$ is compared to at least $\delta(d-1)/2$ elements of T .*

PROOF. $n/2 + i$ is compared to a total of $d-1$ elements of S , chosen independently and uniformly at random. So each comparison is to an element in T with probability δ , and the expected number of elements in T that $n/2 + i$ is compared to is therefore $\delta(d-1)$. Let A_j be the indicator random variable that is 1 if the j^{th} comparison is to an element in T and 0 otherwise. Then the number of elements in T that $n/2 + i$ is compared to is $\sum_j A_j$ and is the sum of independent $\{0, 1\}$ random variables. Therefore, the Chernoff bound states that $\Pr[\sum_j A_j \leq \delta(d-1)/2] \leq e^{-\delta(d-1)/8}$. $\square$

LEMMA 5. *Let $T \subseteq S$ be any set of elements such that:*

- Every element in T is $>_S x$.
- Every element in T is smaller than $n/2 + i$.
- Every element in T is $<_S y$, for all $y \in S \cap (n/2 + i, n]$.

Then if $n/2 + i$ is compared to at least t elements of T , $\Pr[B_{n/2+i} \geq t] \geq 1 - e^{-t\gamma^2/8}$.

PROOF. Let T' denote the subset of T that $n/2 + i$ is compared to. Define A_j to be a random variable that is 1 if $n/2 + i$ beats the j^{th} element of T' , and 0 otherwise. Then $B_{n/2+i} \geq |T'|$ if $\sum_{j=1}^{|T'|} A_j \geq |T'|/2$. As the A_j s are independent $\{0, 1\}$ random variables that are 1 with probability $1/2 + \gamma/2$, the Chernoff bound states that this probability is at least $1 - e^{-|T'|\gamma^2/8}$. $\square$

Finally, we show that $L_{n/2+i}$ is likely to be small, and put everything together. Recall that $\ell_1, \dots, \ell_c$ denote the elements that $n/2 + i$ are compared to that are $<_S x$ in decreasing order, and that $Y_{ij} = 1$ iff $n/2 + i$ beats ℓ_j , and -1 otherwise.

LEMMA 6. *Assume that every element $<_S x$ is also $< n/2 + i$. Then for all t , $\Pr[\sum_{j \leq t} Y_{ij} \geq t\gamma/2] \geq 1 - e^{-t\gamma^2/32}$.*

PROOF. The random variables $(1 + Y_{ij})/2$ are independent and in $\{0, 1\}$. $\mathbb{E}[\sum_{j \leq t} Y_{ij}] = (1/2 + \gamma/2)t$. So the Chernoff bound states that $\Pr[\sum_{j \leq t} (1 + Y_{ij})/2 \leq (1/2 + \gamma/4)t] \leq e^{-t\gamma^2/32}$. Note that $\sum_{j \leq t} (1 + Y_{ij})/2 \leq (1/2 + \gamma/4)t \Leftrightarrow \sum_{j \leq t} Y_{ij} \leq t\gamma/2$. $\square$

PROPOSITION 1. *Assume that every element $<_S x$ is also $< n/2 + i$. Then for all t , $\Pr[L_{n/2+i} \geq t] \leq e^{-t\gamma^2/32} + e^{-t\gamma^2/2}$.*

PROOF. Define a random walk starting at 0 that moves left at time j if $Y_{ij} = -1$, and right at time j if $Y_{ij} = 1$. Then the event that $L_{n/2+i} \geq t$ is exactly the event that this random walk returns to the origin (or left of the origin) at some step $\geq t$. If $\sum_{j \leq t} Y_{ij} \geq t\gamma/2$, then this is exactly the event that the random walk starting at time t winds up $t\gamma/2$ steps to the left. Note that this random walk moves right with probability $1/2 + \gamma/2$ and left with probability $1/2 - \gamma/2$, so the probability that this event occurs is exactly $(\frac{1/2 - \gamma/2}{1/2 + \gamma/2})^{t\gamma/2}$ by Proposition 3 in the full version.

Finally, we observe:

$$\left(\frac{1/2 - \gamma/2}{1/2 + \gamma/2}\right)^{t\gamma/2} \leq (1 - \gamma)^{t\gamma/2} \leq e^{-t\gamma^2/2}$$

$\square$

Now, we are ready to put everything together to prove the theorem. By Corollary 2, with probability at least $1 - 2/\text{poly}(n) - e^{-\varepsilon^2 \sqrt{n}/2}$, there is a set of size $(i/n - \varepsilon - o(1))|S|$ of elements immediately $>_S x$ that are all $< n/2 + i$. By Lemmas 4 and 5, in the event that this happens, $B_{n/2+i} \geq (d-1)(i/n - \varepsilon - o(1))/2$ with probability at least

$$1 - e^{-(i/n - \varepsilon - o(1))(d-1)/8} - e^{-(i/n - \varepsilon - o(1))(d-1)\gamma^2/2}.$$

Furthermore, by Proposition 1, in the event that this happens, $L_{n/2+i} < (d-1)(i/n - \varepsilon - o(1))/2$ with probability at least

$$1 - e^{-(i/n - \varepsilon - o(1))(d-1)\gamma^2/64} - e^{-(i/n - \varepsilon - o(1))(d-1)\gamma^2/4}.$$

Combining all of this with a union bound, we see that $n/2 + i$ is mistakenly accepted with probability at most $2/\text{poly}(n) + e^{-\varepsilon^2 \sqrt{n}/2} + 4e^{-(i/n - \varepsilon)(d-1)\gamma^2/64}$. Taking $\varepsilon = 1/d$ and summing this over all i , we get:

$$\sum_{i=1}^{n/2} (2/\text{poly}(n) + e^{-\sqrt{n}/2d^2}) i^c = o(n^{c+1}).$$

$$\begin{aligned} & \sum_{i=1}^{n/2} \left(4e^{-(i/n - \varepsilon)(d-1)\gamma^2/64} \right) i^c \\ & \leq 4e^{\gamma^2/64} \sum_{i=1}^{n/2} e^{-((d-1)\gamma^2/64n)i} i^c \\ & \leq 8c! (64nd^{-1}\gamma^{-2})^{c+1}. \end{aligned}$$

The last inequality is due to Lemma 2 in the full version. Summing both terms together yields the theorem.

3.2 Lower Bounds

In this section, we show that the algorithms designed in the previous section are optimal up to constant factors. All of the algorithms in the previous section are “tight,” in the sense that we expect element i to be correctly placed whenever it is compared to enough elements between itself and the median. In the noiseless model, one element is enough. In the erasure model, we instead need $\Omega(1/\gamma)$ (to make sure at least one isn’t erased). In the noisy model, we need $\Omega(1/\gamma^2)$ (to make sure we get $\Omega(1)$ bits of information about the difference between i and the median). If we don’t have enough comparisons between i and elements between itself and the median, we shouldn’t hope to be able to classify i correctly, as the comparisons involving i would look nearly identical if we replaced i with an element just on the other side of the median. Due to space constraint, the proof of the noiseless model is provided. The proofs of the erasure model and the noisy model can be found in the full version.

THEOREM 6. *For all $c, d > 0$, any non-adaptive algorithm with query complexity dn necessarily has expected c -weighted error $\Omega((n/d)^{c+1})$ in the noiseless model, $\Omega((n/(d\gamma))^{c+1})$ in the erasure model, and $\Omega((n/(d\gamma^2))^{c+1})$ in the noisy model.*

3.2.1 Noiseless Model

The main idea of the proof is that if two elements $i < n/2$ and $j > n/2$ are never compared to any elements in $[i, j]$, then the results of these comparisons are independent of whether or not i and j are swapped. So however the algorithm decides to place i and j , a coupling argument (by swapping the role of i and j in all comparisons) shows that the error must be large if i and j are unlikely to be compared to elements in $[i, j]$. We proceed by analyzing pairs of elements each a distance i away from $n/2$ separately.

Let D be any distribution over set of comparisons chosen non-adaptively for the algorithm. As the elements are a priori indistinguishable, it’s clear that D is invariant under permutations. So let S denote a sample from D , and S_i denote the same set after swapping the roles of $n/2 - i$ and $n/2 + i$ in all comparisons. Let also $X_i(S)$ denote the random variable that equal to the number of $\{n/2 + i, n/2 - i\}$ that are incorrectly classified by the algorithm with comparisons S . Then clearly the expected error is equal to $\sum_S \sum_{i=1}^{n/2} \Pr[S] E[X_i(S)] i^c$. And because D is invariant under permutations, we can rewrite this as:

$$\sum_S \sum_{i=1}^{n/2} \Pr[S] \frac{E[X_i(S)] + E[X_i(S_i)]}{2} i^c$$

Now we proceed to bound $E[X_i(S)] + E[X_i(S_i)]$.

LEMMA 7. *Let p_i denote the probability that neither of $\{n/2 + i, n/2 - i\}$ are compared to an element in $[n/2 - i, n/2 + i]$ when sampling a set of comparisons from D . Then $\sum_S \frac{E[X_i(S)] + E[X_i(S_i)]}{2} \geq p_i$.*

PROOF. Because neither element is compared to an element between them, the result of all comparisons will be the same using S or S_i . So coupling the randomness used by any algorithm with S versus S_i , we see that swapping the elements $n/2 + i, n/2 - i$ simply swaps the set that each element winds up in. If both are placed in R using S , then

both are placed in R using S_i , so each produces one mistake. If both are placed in A using S , then both are placed in A using S_i , so each produces one mistake. If one is placed in A and the other in R using S , then the same holds using S_i , but swapped. So either S produces zero mistakes and S_i produces two, or vice versa. So no matter the algorithm, it is clear that when S does not contain comparisons between $\{n/2 + i, n/2 - i\}$ and any elements between them, that $E[X_i(S)] + E[X_i(S_i)] = 2$. $\square$

COROLLARY 3. *Let p_i denote the probability that neither of $\{n/2 + i, n/2 - i\}$ are compared to an element in $[n/2 - i, n/2 + i]$ during algorithm $\mathcal{A}$. Then the expected error of $\mathcal{A}$ is at least $\sum_{i=1}^{n/2} p_i i^c$.*

The remaining work is to lower bound p_i and evaluate the sum.

LEMMA 8. *For any algorithm $\mathcal{A}$ that makes at most dn non-adaptive queries, $p_i \geq \max\{1 - 8di/n, 0\}$.*

PROOF. Because D is invariant under permutations, we can sample S by first building a graph with n nodes and edges between two nodes that are compared, then randomly labeling the nodes with $1, \dots, n$. For any pair (x, y) , the probability that they are compared is then the probability that two randomly selected nodes have an edge between them. If there are dn total edges in the graph, this probability is (almost) exactly $2d/n$. Taking a union bound over the $4i$ possible comparisons means that the probability that either $n/2 - i$ or $n/2 + i$ is compared to something in between is at most $8di/n$. $\square$

Now, we may complete the proof by evaluating the sum.

$$\begin{aligned} \sum_{i=1}^{n/2} p_i i^c &\geq \sum_{i=1}^{n/8d} p_i i^c \\ &\geq \sum_{i=1}^{n/8d} (1 - 8di/n) i^c \\ &\geq \sum_{i=1}^{n/8d} i^c - \sum_{i=1}^{n/8d} 8di^{c+1}/n \\ &\geq (n/8d)^{c+1}/(c+1) - 8d(n/8d)^{c+2}/n(c+2) \\ &= (n/8d)^{c+1}/(c+1)(c+2) = \Omega((n/d)^{c+1}) \end{aligned}$$

4. RESULTS FOR MULTI-ROUND ALGORITHMS

4.1 Noiseless Model

We first present our algorithm and nearly matching lower bound for 2-round algorithms. The first round of our algorithm tries to get as good of an approximation to the median as possible, and then compares it to every element in round two. Getting the best possible approximation is actually a bit tricky. For instance, simply finding the median of a skeleton set of size $\sqrt{n}$ only guarantees an element

within $\Theta(n^{3/4})$ of the median.⁶ We instead take several “iterations” of nested skeleton sets to get a better and better approximation to the median. In reality, all iterations happen simultaneously in the first round, but it is helpful to think of them as sequential refinements.

For any $r \geq 1$, our algorithm starts with a huge skeleton set S_1 of $n^{2r/(2r+1)}$ random samples from $[n]$. This is too large to compare every element in S_1 with itself, so we choose a set $T_1 \subseteq S_1$ of $n^{1/(2r+1)}$ random pivots. Then we compare every element in S_1 to every element in T_1 , and we will certainly learn two pivots, a_1 and b_1 such that the median of S_1 lies in $[a_1, b_1]$, and a p_1 such that the median of S_1 is exactly the $(p_1|A_1|)^{th}$ element of $A_1 = S_1 \cap [a_1, b_1]$. Now, we recurse within A_1 and try to find the $(p_1|A_1|)^{th}$ element. Of course, because all of these comparisons happen in one round, we don’t know ahead of time in which subinterval of S_1 we’ll want to recurse, so we have to waste a bunch of comparisons. These continual refinements still make some progress, and allow us to find a smaller and smaller window containing the median of S_1 , which is a very good approximation to the true median because S_1 was so large. Pseudocode for our algorithm can be found in the full version, which “recursively” tries to find the $(p_i|A_i|)^{th}$ element of A_i .

THEOREM 7. *For all c, r and $\varepsilon > 0$, we have an algorithm which has round complexity 2, query complexity $(r+1)n$, and outputs a partition that:*

- *has expected c -weighted error at most $(8rn^{(r+1)/(2r+1)+\varepsilon})^{c+1}$*
- *has c -weighted error at most $(8rn^{(r+1)/(2r+1)+\varepsilon})^{c+1}$ with probability at least $1 - re^{-\Omega(n^\varepsilon)}$.*

Note that setting $r = \log n$, and ε such that $n^\varepsilon = 8 \log^3 n$, we get an algorithm with round complexity 2, query complexity $n \log n + n$ that outputs a partition with c -weighted error $O((\sqrt{n} \log^4 n)^{c+1})$ with probability $1 - O(\log n/n^2)$.

We also prove a nearly matching lower bound on two-round algorithms in the noiseless model. At a very high level, our lower bound repeats the argument of our one round lower bound twice. Specifically, we show that after one round, there are many elements within a window of size $\Theta(n/d)$ of the median such that a constant fraction of these elements have not been compared to any other elements in this window. We then show that after the second round, conditioned on this, there is necessarily a window of size $\approx \sqrt{n}$ such that a constant fraction of these elements have not been compared to any other elements in this window. Finally we show that this implies that we must err on a constant fraction of these elements. The actual proof is technical, but follows this high level outline. Proofs of Theorems 7 and 8 can be found in the full version.

THEOREM 8. *For all c , and any $d = o(n^{1/5})$, any algorithm with query complexity dn and round complexity 2 necessarily has expected c -weighted error $\Omega((\sqrt{n}/d^{5/2})^{c+1})$.*

From here we show how to make use of our two-round algorithm to design a three-round algorithm that makes zero

⁶This is exactly what Bollobás and Brightwell do in the first round of their 4-round algorithm, which is why sophisticated graph theory follows to fit into four rounds. Our improved first round simplifies the remaining rounds.

mistakes with high probability. After our two-round algorithm with appropriate parameters, we can be pretty sure that the median lies somewhere in a range of $O(\sqrt{n} \log^4 n)$, so we can just compare all of these elements to each other in one additional round. Pseudocode can be found in the full version.

THEOREM 9. *For all c , we have an algorithm that has query complexity $O(n \log^8 n)$, round complexity 3, and outputs a partition with zero c -weighted error with probability $1 - O(\log n/n^2)$.*

Again, recall that $\omega(n)$ queries are necessary for any three-round algorithm just to solve SELECT with probability $1 - o(1)$ [BB90]. Finally, we further make use of ideas from our two-round algorithm to design a simple four round algorithm that has query complexity $O(n)$ and makes zero mistakes with high probability. More specifically, we appropriately tune the parameters for our two-round algorithm (i.e. set $r = 1$) to find a window of size $\approx n^{2/3}$ that contains the median (and already correctly partition all other elements). We then use similar ideas in round three to further find a window of size $\approx \sqrt{n}$ that contains the median (and again correctly partition all other elements). We use the final round to compare all remaining uncertain elements to each other and correctly partition them.

THEOREM 10. *For all c , and any $\varepsilon \in (0, 1/18)$, we have an algorithm which has query complexity $O(n)$, round complexity 4, and outputs a partition with zero c -weighted error with probability at least $1 - e^{-\Omega(n^\varepsilon)}$.*

4.2 Erasure and Noisy Models

Here we briefly overview our results on multi-round algorithms in the erasure and noisy models. We begin with an easy reduction from these models to the noiseless model, at the cost of a blow-up in the round or query complexity. Essentially, we are just observing that one can adaptively resample any comparison in the erasure model until it isn’t erased (which will take $1/\gamma$ resamples in expectation), and also that one can resample in parallel any comparison in either the erasure or noisy model the appropriate number of times and have it effectively be a noiseless comparison.

PROPOSITION 2. *If there is an algorithm solving PARTITION, SELECT or FINDMIN in the noiseless model with probability p that has query complexity Q and round complexity r , then there are also algorithms that resample that:*

- *solve PARTITION, SELECT or FINDMIN in the erasure model with probability p that have expected query complexity Q/γ , but perhaps with expected round complexity Q/γ as well.*
- *solve PARTITION, SELECT or FINDMIN in the erasure model with probability $p - 1/\text{poly}(n)$ that have query complexity $O(Q(\log Q + \log n)/\gamma)$, and round complexity r .*
- *solve PARTITION, SELECT or FINDMIN in the noisy model with probability $p - 1/\text{poly}(n)$ that have query complexity $O(Q(\log Q + \log n)/\gamma^2)$, and round complexity r .*

COROLLARY 4. *There are algorithms that resample that:*

- solve PARTITION or SELECT in the erasure model with probability 1 with expected query complexity $O(n/\gamma)$ (based on the QuickSelect or Median-of-Medians algorithm [Hoa61, BFP⁺73]).
- solve PARTITION or SELECT in the erasure model with probability $1 - 1/\text{poly}(n)$ with query complexity $O(n \log n/\gamma)$ and round complexity 4.
- solve PARTITION or SELECT in the noisy model with probability $1 - 1/\text{poly}(n)$ with query complexity $O(n \log n/\gamma^2)$ and round complexity 4.

In the erasure model, the algorithms provided by this reduction do not have the optimal round/query complexity. We show that $\Theta(n/\gamma)$ queries are necessary and sufficient, as well as $\Theta(\log^*(n))$ rounds. For the algorithm, we begin by finding the median of a random set of size $n/\log n$ elements. This can be done in 4 rounds and $O(n/\delta)$ total comparisons by Corollary 4. Doing this twice in parallel, we find two elements that are guaranteed to be above/below the median, but very close. Then, we spend $\log^*(n)$ rounds comparing every element to both of these. It's not obvious that this can be done in $\log^*(n)$ rounds. Essentially what happens is that after each round, a fraction of elements are successfully compared, and we don't need to use any future comparisons on them. This lets us do even more comparisons involving the remaining elements in future rounds, so the fraction of successes actually increases with successive rounds. Analysis shows that the number of required rounds is therefore $\log^*(n)$ (instead of $\log(n)$ if the fraction was constant throughout all rounds). After this, we learn for sure that the median lies within a sublinear window, and we can again invoke the 4-round algorithm of Corollary 4 to finish up. Our lower bound essentially shows that it takes $\log^*(n)$ rounds just to have a non-erased comparison involving all n elements even with $O(n/\delta)$ per round, and that this implies a lower bound. Pseudocode for the algorithm and proofs of both theorems can be found in the full version.

THEOREM 11. *With probability at least $1 - 1/\text{poly}(n)$, we have an algorithm that has query complexity $O(n/\gamma)$, round complexity $\log^*(n) + 8$, and solves PARTITION.*

THEOREM 12. *Assume $\gamma \leq 1/2$. In the erasure model, any algorithm solving SELECT with probability $2/3$ even with $O(n/\gamma)$ comparisons per round necessarily has round complexity $\Omega(\log^*(n))$.*

We now introduce a related problem that is strictly easier than PARTITION or SELECT, which we call RANK, and prove lower bounds on the round/query complexity of RANK noisy models, which will imply lower bounds on PARTITION and SELECT. In RANK, we are given as input a set S of n elements, and a special element b and asked to determine b 's rank in S (i.e. how many elements in S are less than b).⁷ We say that a solution is a t -approximation if the guess is within t of the element's actual rank. We show formally that RANK is strictly easier than SELECT in the full version. From here, we prove lower bounds against RANK in the noisy model.

At a high level, we show (in the proof of Theorem 13) that with only $O(n \log n/\gamma^2)$ queries, it's very likely that there are a constant fraction of a_i 's such that the algorithm is can't be very sure about the relation between a_i and b .

This might happen, for instance, if not many comparisons were done between a_i and b and they were split close to 50-50. From here, we use an anti-concentration inequality (the Berry-Essen inequality) to show that the rank of b does not concentrate within some range of size $\Theta(n^{3/8})$ conditioned on the available information. In other words, the information available simply cannot narrow down the rank of b to within a small window with decent probability, no matter how that information is used. We then conclude that no algorithms with $o(n \log n/\gamma^2)$ comparisons can approximate the rank well with probability $2/3$.

THEOREM 13. *In the noisy model, any algorithm obtaining an $(n^{3/8}/40)$ -approximation for RANK with probability $2/3$ necessarily has query complexity $\Omega(n \log n/\gamma^2)$.*

Finally, we conclude with an algorithm for FINDMIN in the noisy model showing that FINDMIN is strictly easier than SELECT. This is surprising, as most existing lower bounds against SELECT are obtained by bounding FINDMIN. Our algorithm again begins by finding the minimum, x , of a random set of size $n/\log n$ using $O(n/\gamma^2)$ total comparisons by Corollary 4. Then, we iteratively compare each element to x a fixed number of times, throwing out elements that beat it too many times. Again, as we throw out elements, we get to compare the remaining elements to x more and more. We're able to show that after only an appropriate number of iterations (so that only $O(n/\delta^2)$ total comparisons have been made), it's very likely that only $n/\log n$ elements remain, and that with constant probability the true minimum was not eliminated. From here, we can again invoke the algorithm of Corollary 4 to find the true minimum (assuming it wasn't eliminated).

THEOREM 14. *Assume n is large enough and $10 \leq c \leq \log n$. we have an algorithm that has query complexity $\frac{3cn}{\gamma^2}$ and solves FINDMIN in the noisy model with probability at least $1 - e^{-\Omega(c)}$.*

THEOREM 15. *Assume $c \geq 1$, n is large enough and $\gamma \leq 1/4$. Any algorithm in the noisy model with query complexity $\frac{cn}{\gamma^2}$ solves FINDMIN with probability at most $1 - e^{-O(c)}$.*

Theorem 14 shows that FINDMIN is strictly easier than SELECT (as it can be solved with constant probability with asymptotically fewer comparisons). Theorem 15 is included for completeness, and shows that it is not possible to get a better success probability without a blow-up in the query complexity. The proof of Theorem 15 is similar to that of Theorem 13.

5. REFERENCES

- [AA88a] Noga Alon and Yossi Azar. The average complexity of deterministic and randomized parallel comparison-sorting algorithms. *SIAM J. Comput.*, 17(6):1178–1192, 1988.
- [AA88b] Noga Alon and Yossi Azar. Sorting, approximate sorting, and searching in rounds. *SIAM J. Discrete Math.*, 1(3):269–280, 1988.
- [AAV86] Noga Alon, Yossi Azar, and Uzi Vishkin. Tight complexity bounds for parallel comparison sorting. In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 502–510, 1986.

⁷A more formal statement of RANK is in the full version.

- [AKS83] Miklós Ajtai, János Komlós, and Endre Szemerédi. An $o(n \log n)$ sorting network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, 25-27 April, 1983, Boston, Massachusetts, USA, pages 1–9, 1983.
- [AKSS86] M Ajtai, J Komlos, W L Steiger, and E Szemerédi. Deterministic selection in $o(\log \log n)$ parallel time. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 188–195, New York, NY, USA, 1986. ACM.
- [BB90] Béla Bollobás and Graham Brightwell. Parallel selection with high probability. *SIAM J. Discrete Math.*, 3(1):21–31, 1990.
- [BFP⁺73] Manuel Blum, Robert W. Floyd, Vaughan Ronald Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.
- [BH85] Béla Bollobás and Pavol Hell. Sorting and graphs. In Ivan Rival, editor, *Graphs and Order*, volume 147 of *NATO ASI Series*, pages 169–184. Springer Netherlands, 1985.
- [BM08] Mark Braverman and Elchanan Mossel. Noisy sorting without resampling. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 268–276, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.
- [BM09] Mark Braverman and Elchanan Mossel. Sorting from noisy information. *CoRR*, abs/0910.1191, 2009.
- [BSC⁺13] Róbert Busa-Fekete, Balázs Szörényi, Weiwei Cheng, Paul Weng, and Eyke Hüllermeier. Top-k selection based on adaptive sampling of noisy preferences. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1094–1102, 2013.
- [BT83] Béla Bollobás and Andrew Thomason. Parallel sorting. *Discrete Applied Mathematics*, 6(1):1 – 11, 1983.
- [CBCTH13] Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 193–202, New York, NY, USA, 2013. ACM.
- [CN91] Andrew Caplin and Barry Nalebuff. Aggregation and social choice: A mean voter theorem. *Econometrica*, 59(1):1–23, 1991.
- [Col88] Richard Cole. Parallel merge sort. *SIAM J. Comput.*, 17(4):770–785, 1988.
- [CS15] Yuxin Chen and Changho Suh. Spectral MLE: top-k rank aggregation from pairwise comparisons. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 371–380, 2015.
- [DKNS01] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM.
- [Eri13] Brian Eriksson. Learning to top-k search using pairwise comparisons. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, pages 265–273, 2013.
- [HH81] Roland Häggkvist and Pavol Hell. Parallel sorting with constant time for comparisons. *SIAM J. Comput.*, 10(3):465–472, 1981.
- [Hoa61] Charles Antony Richard Hoare. Algorithm 65: Find. *Communications of the ACM*, 4(7):321–322, 1961.
- [Lei84] Frank Thomson Leighton. Tight bounds on the complexity of parallel sorting. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, April 30 - May 2, 1984, Washington, DC, USA, pages 71–80, 1984.
- [LM00] Frank Thomson Leighton and Yuan Ma. Tight bounds on the size of fault-tolerant merging and sorting networks with destructive faults. *SIAM Journal on Computing*, 29(1):258–273, 2000.
- [MMV13] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Sorting noisy data with partial information. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 515–528, New York, NY, USA, 2013. ACM.
- [PHC⁺13] Chris Piech, Jonathan Huang, Zhenghao Chen, Chuong Do, Andrew Ng, and Daphne Koller. Tuned models of peer assessment in moocs. In *Proceedings of the Sixth International Conference on Educational Data Mining*, EDM '13, 2013.
- [Rei81] Rüdiger Reischuk. A fast probabilistic parallel sorting algorithm. In *22nd Annual Symposium on Foundations of Computer Science*, Nashville, Tennessee, USA, 28-30 October 1981, pages 212–219, 1981.
- [Val75] Leslie G. Valiant. Parallelism in comparison problems. *SIAM J. Comput.*, 4(3):348–355, 1975.