THEORY OF COMPUTING, Volume 6 (2010), pp. 113–134
www.theoryofcomputing.org

Rounds vs. Queries Tradeoff in Noisy Computation

Navin Goyal[*] Michael Saks[†]

Received: January 26, 2010; published: September 2, 2010.

Abstract: We show that a noisy parallel decision tree making $O(n)$ queries needs $\Omega(\log^* n)$ rounds to compute OR of n bits. This answers a question of Newman [*IEEE Conference on Computational Complexity*, 2004, 113–124]. We prove more general tradeoffs between the number of queries and rounds. We also settle a similar question for computing MAX in the noisy comparison tree model; these results bring out interesting differences among the noise models.

ACM Classification: F.2.3

AMS Classification: 68Q13

Key words and phrases: noisy computation, tradeoff lower bound, decision trees

1 Introduction

Understanding the impact of noise on computation and communication is an important problem and it has been studied in various fields. Within the theory of computing, noise has been studied in the context of various models of computation: decision trees, e. g., [5, 9, 3, 2, 7], various kinds of communication protocols, e. g., [6, 4, 7], and many others. In this paper we will be concerned with the noisy Boolean decision tree and the noisy comparison tree models.

[*]Most of this author's research was performed at Rutgers University supported in part by a Louis Bevier Fellowship of Rutgers University and NSF grant CCR-9988526.

[†]Supported in part by NSF grants CCR-9988526,CCF-0515201 and CCF-0832787.

DOI: 10.4086/toc.2010.v006a006

1.1 Noisy Boolean decision trees

A Boolean decision tree represents an algorithm for computing a Boolean function $f(x_1,\ldots,x_n)$ by adaptively querying its variables. Several models have been proposed to formalize the notion of noise in the context of decision trees. Among these models, perhaps the most popular and simplest is the *random noise model*. In this model, there is a noise parameter $\varepsilon \in [0,1/2)$ and the answer given to each query is incorrect with probability ε, independent of the answers to other queries and any randomness used by the algorithm. We call a decision tree operating under the random noise model a *noisy decision tree* (NDT). We say that an NDT *computes f with error probability δ* (for some $\delta \in [0,1/2)$) if on each input $x \in \{0,1\}^n$, the NDT outputs $f(x)$ with probability at least $1-\delta$. For brevity, we say that an NDT *computes f*, if it computes f with error probability at most $1/4$.

A noise-free Boolean decision tree of depth d can be simulated by a noisy decision tree of depth $O(d\log(d/\delta))$ with arbitrarily small error δ: Each query of a variable in the noise-free setting is simulated by taking the majority of $\Theta(\log(d/\delta))$ noisy queries of the same variable. Any n-variate function can be computed with at most n queries in the noise-free setting, and thus can be computed with at most $O(n\log n)$ queries in the random noise model. Feige et al. [5] showed that to compute $\mathsf{MAJORITY}$ and PARITY in the noisy decision tree model $\Omega(n\log n)$ queries are indeed necessary. In contrast, they showed that OR can be computed in $O(n)$ queries in the noisy model.

There is a natural notion of parallelism for decision trees [11]. In each step (or *round*), the decision tree may make many queries (as opposed to a single query in the usual decision trees) and the decision tree branches according to the ensemble of answers to all of the queries. We can think of the queries in a single round as being made in parallel. In the noise-free model, such an algorithm is called a *parallel decision tree* (PDT).

In this paper, we investigate parallel noisy decision trees (PNDTs), which is the parallel version of the noisy decision trees. See Section 2 for precise definition. PNDTs were introduced by Newman [7], who used them to construct protocols in the noisy broadcast model of distributed computing. His work raised the problem of understanding the tradeoff between the the number of rounds and the total number of queries needed to compute OR_n (the OR of n input bits) for PNDTs. The $O(n)$-query NDT for OR_n of Feige et al. [5] gives an $O(\log n)$ rounds PNDT. Newman reduced this to $O(\log^* n)$ rounds with $O(n)$ queries, and asked: Is there a noisy decision tree for OR using $O(n)$ queries and $O(1)$ rounds? In this paper we provide a negative answer to this question and show that Newman's PNDT is essentially optimal by proving a rounds-query tradeoff lower bound. The notation used in this section is defined in Section 2.

Theorem 1.1. *Let $\varepsilon \in (0,1/2)$. There is a positive integer $C = C(\varepsilon)$ such that for any positive integers n, r with $r \le \log^*_{1/\varepsilon} n - C$, any PNDT T that computes OR_n in r rounds with error probability less than 1/4 under the random noise model with noise parameter ε. Then T requires at least $\frac{1}{100} n \log^{(r)}_{1/\varepsilon} n$ queries.*

We note two special cases of the theorem: (1) $r = 1$ corresponds to the case when all queries are made in parallel in the first round; the number of queries needed here is $\Omega(n\log n)$. (2) Any PNDT that computes OR_n using $O(n)$ queries needs $\log^*_{1/\varepsilon} n - C$ rounds, where C is independent of n but depends on ε.

It has been noted (in, e. g., [4]) that the unrealistic assumption of the random noise model, that each query experiences independent noise with the same probability, allows for artificial algorithms that exploit this regularity of noise. Other models of noise have been proposed [4] that are more *robust* in that they model noise by a family of possible distributions rather than as a single distribution, and a good PNDT for a problem must succeed against any distribution in the family. For example, a minimal robustness requirement is that a good PNDT should still succeed if all noise is eliminated. Noise models of this type are conveniently defined by viewing the noise as partially controlled by an adversary; the more powerful the adversary the harder it is to design algorithms that are robust against it. In the random noise model, the adversary has no power to influence the noise, in the *static adversary* model the adversary has some power, and in the *clairvoyant adversary* model, the adversary has even more power. These models are defined in Section 2.1.

For upper bounds, one seeks to design PNDTs that tolerate noise in the most robust model possible. Since our lower bound is proved for the least robust model, they hold for the other models as well.

Our lower bound for OR_n is deduced from a lower bound on a related problem, the "Which Half Problem" denoted WHP_n. Working with this problem substantially simplifies the proof. The input to WHP_n is a $2n$-bit vector with exactly one bit set to 1. The goal is to determine whether the 1 is in the first or the second half of the bits. We prove that if the location of 1 is chosen uniformly at random then we get a query-round trade-off as in Theorem 1.1. The proof is by induction on the number of rounds. The induction step uses *round elimination*: We prove a lemma saying that after the first round of queries, with high probability we are left with a problem that is at least as hard as a smaller (but not too much smaller) version of the original problem. Round elimination as a general idea has been used in communication complexity; we are not aware of similar applications in the context of noisy computation.

1.2 Noisy comparison decision trees

In the comparison tree model, the variables take values from an abstract totally ordered set U and a query specifies two variables x_i, x_j and asks whether $x_i < x_j$ or $x_i > x_j$. (We assume for simplicity that the variables have distinct values.) The parallel version of the model, parallel comparison trees (PCTs), was considered by Valiant [11].

The random noise model and the other more robust noise models have a natural analog for comparison trees; e. g., in the random noise model the answer to each comparison query is incorrect with probability ε. Thus we can define parallel noisy comparison trees (PNCTs). Such trees were considered by Feige et al. [5]. Among other results, they showed that the function MAX_n, which determines the index of the maximum variable among n variables, can be computed in $O(\log n)$ rounds using $O(n)$ comparisons, even in the most robust noise model.

We are interested in the tradeoffs between the total number of comparisons and the number of rounds used by a PNCT computing the maximum function MAX.

For deterministic trees, in the noise-free setting, Valiant showed that deterministic PCTs that make $O(n)$ comparisons per round require $\Omega(\log\log n)$ rounds. He gave a PCT with $O(\log\log n)$ rounds and $O(n\log\log n)$ comparisons. This was improved to $O(\log\log n)$ rounds and total $O(n)$ comparisons by Shiloach and Vishkin [10]. Thus, for deterministic PCTs using $O(n)$ comparisons, $\Theta(\log\log n)$ rounds are necessary and sufficient. We adapt the Shiloach and Vishkin upper bound to deterministic PNCTs:

Theorem 1.2. *Let $\varepsilon \in [0, 1/2)$. For any of the noise models of Section 2.1 there is a deterministic PNCT computing MAX_n in $O(\log\log n)$ rounds and $O(n)$ comparisons.*

Any lower bound for deterministic PCTs in the noise-free case also applies to the static and clairvoyant adversary models, because in those two models the adversary is allowed to eliminate all noise. Therefore Valiant's lower bound for the noise-free case applies to both of these models and so the result of Theorem 1.2 is tight. As mentioned earlier, algorithms in the random noise model are potentially more powerful than in the deterministic model (which is why the random noise model is considered to be unsatisfactory when analyzing upper bounds), because the noise can be used to simulate randomized algorithms that outperform any deterministic algorithms. Our results provide a specific example of this phenomena: after presenting a randomized PNCT for MAX_n, we show in Corollary 1.4, that the randomized PNCT for MAX_n can be adapted to give a deterministic PNCT for MAX_n in the random noise model that beats Valiant's lower bound in the noise-free model.

For randomized PCTs, in the noise-free setting, Reischuk [8] gave an algorithm with $O(1)$ rounds and $O(n)$ comparisons, which is clearly best possible up to constant factors. For the noisy case we obtain the following tight queries-rounds tradeoff for randomized PNCTs computing MAX_n.

Theorem 1.3. *There exists a constant $\varepsilon_0 \in (0,1)$ such that for noise parameter ε satisfying $0 < \varepsilon \le \varepsilon_0$, $\Theta(\log^*_{1/\varepsilon} n)$ rounds are necessary and sufficient to compute MAX_n by a randomized PNCT using $O(n)$ comparisons. This holds for each of the three noise models in Section 2.1.*

The lower bound is proved by a reduction from the PNDT lower bound for OR; the upper bound adapts Reischuk's [8] algorithm to the noisy case. The resulting algorithm can be easily modified to get a deterministic algorithm for the random noise model:

Corollary 1.4. *There exists a constant $\varepsilon_0 \in (0,1)$ such that for noise parameter ε satisfying $0 < \varepsilon \le \varepsilon_0$, $\Theta(\log^*_{1/\varepsilon} n)$ rounds are necessary and sufficient to compute MAX_n by a deterministic PNCT using $O(n)$ comparisons in the random noise model.*

Organization of the paper. The rest of this paper is organized as follows. Section 2 contains a formal description of the PNDT model and some definitions. In Section 3, we present a PNDT for OR_n, different from Newman's, that meets the lower bound of Theorem 1.1, and provides intuition for why there is no $O(n)$ query, $O(1)$ round protocol. In Section 4 we present the proof of Theorem 1.1 by reducing the problem of lower bound for OR to lower bound for the "Which Half" problem. In Section 5 we present results for the comparison tree model. Section 6 concludes with some open problems.

2 Preliminaries

A *Boolean decision tree* over Boolean variables $\{x_1,\ldots,x_n\}$ is a rooted binary tree T where each internal node of the tree is labeled by a variable and has two children, the two edges going out of a node have labels 0 and 1, and each leaf is labeled 0 or 1. An execution of T determines a path from the root as follows: Starting from the root, a query is made to the variable labeling that node, and the (possibly noisy) answer to the query determines the edge to be followed. This is repeated until a leaf is reached. The output is the label of the leaf.

In a *noise-free execution*, the response to each query is equal to the value of the queried variable and thus the path followed is completely determined by the input.

In a *noisy execution* the response to each query may disagree with the input variable. Associated to each node v is a noise random variable η_v which is 0 if the answer is correct (so that the query answer agrees with the input variable) and is 1 if the answer is incorrect (so that the query answer disagrees with the input variable). Thus the path followed is a random variable depending on the noise. In the *random noise model* with parameter $\varepsilon \in [0,1/2)$, the Z_v are each 1 with probability ε, and are mutually independent and also independent.

We say that a decision tree computes a function $f : \{0,1\}^n \to \{0,1\}$ for noise parameter ε if for all inputs x, a noisy execution on the tree outputs $f(x)$ with probability $\geq 3/4$, where the probability is taken over the random noise. Of course, $3/4$ is an arbitrary constant, and we could choose it to be any constant in $(1/2,1)$ using standard amplification, increasing the number of rounds and queries only by a constant factor.

For noiseless executions, we generally assume without loss of generality that the tree does not query the same variable more than once along any path. For noisy executions, this assumption is not made. A decision tree that is being subjected to noisy execution (and may include multiple queries to the same variable) is called a *noisy decision tree* (NDT).

A *parallel Boolean decision tree* (PDT) is similar to a Boolean decision tree, except that each internal node v is labeled by a list of variables (possibly with repetition) corresponding to a collection of queries made in parallel. There are 2^m branches coming from v (where m is the number of parallel queries made at the node), corresponding to the possible responses to these queries. A noiseless execution is defined in the obvious way. In a noisy execution, there are now noise random variables $\eta_{v,i}$ corresponding to the ith query at node v. The random noise model is defined in the obvious way. In the noiseless setting, the list of queries labeling a node may be assumed to be distinct.

2.1 More robust noise models

As mentioned in the introduction, models of noise other than the random noise model have been proposed; we will explain some relevant ones in this section. A *noise distribution* for the PNDT is any joint distribution over the random variables $\eta_{v,i}$. A general noise model consists of a family of allowed joint distributions over the noise variables. As mentioned above, in the *random noise* model, the family consists of a single distribution in which each variable is chosen independently to be 1 with probability ε.

In the *static adversary* model [4] (also called the *fault tolerance* model [7]), given the PNDT, the allowed distributions are those for which the noise variables are mutually independent but the probability of noise for each variable may be any number in $[0,\varepsilon]$.

In the *clairvoyant adversary model* [4] the noise variables are not required to be mutually independent. The allowed distributions are those that satisfy that for any noise variable and any setting of the other noise variables, the conditional probability that the noise variable is 1 given that setting is at most ε. The clairvoyant adversary model is the most robust; that is, the allowed distributions are more general than the static adversary model.

Here is another way to think about the clairvoyant adversary model with parameter ε that is especially useful. In an execution, values are generated independently for all of these noise variables as in the random noise model. The adversary then has the option of changing any noise variable from 1 to 0 (but not from 0 to 1). Effectively, the adversary may cancel any of the noise (and can do this with full knowledge of the PDT, the input values and the values of all noise variables.)

This viewpoint is particularly useful for showing that certain algorithms are robust against the clairvoyant adversary. Let B be any event that depends on the noise variables. We say that B is *monotone with respect to noise* if for any setting of the noise variables for which B occurs then B continues to occur if any variables set to 1 are changed to 0. This notion is useful for the analysis of algorithms because of the following observation:

Lemma 2.1. *Let B be an event that is monotone with respect to noise. If $\Pr[B] \le \delta$ with respect to the random noise model, then for any clairvoyant adversary, $\Pr[B] \le \delta$.*

2.2 Towers and iterated logarithms

Let $b > 1$ be a real number and k be a nonnegative integer. The *tower function* $\mathrm{Tower}_b^{(k)}(\cdot)$ is defined recursively for $s \ge 1$ by:

$$\mathrm{Tower}_b^{(k)}(s) = \begin{cases} s & \text{if } k = 0, \\ b^{\mathrm{Tower}_b^{(k-1)}(s)} & \text{if } k \ge 1. \end{cases}$$

We define $\log_b^*(x)$ to be the least integer r such that $\mathrm{Tower}_b^{(r)}(1) \ge x$. The base b *k-iterated log function* is defined for real numbers x that satisfy $\log_b^*(x) \ge k$ by:

$$\log_b^{(k)}(x) = \begin{cases} x & \text{if } k = 0, \\ \log_b(\log_b^{(k-1)}(x)) & \text{if } k \ge 1. \end{cases}$$

Observe that for each fixed k and b, $\mathrm{Tower}_b^{(k)}(\cdot)$ and $\log_b^{(k)}(\cdot)$ are inverse functions.

We state the following routine facts without proof:

Proposition 2.2. *Let $b > a > 1$ be real numbers. There is a nonnegative integer $T = T(a,b)$ with the property that for all $x \ge 1$ and $1 \le k \le \log_a^*(x) - T$,*

1. $\log_a^{(k)}(x) \le (\log_a b) \log_b^{(k)}(x)$.

2. $\log_a^{(k)}(x^2) \le 2 \log_a^{(k)}(x)$.

2.3 Other notation

For a bit vector $u \in \{0,1\}^n$, $|u| = \sum_i u_i$. For $1 \le i \le n$ let $e_i = e_i^n$ denote the vector with all except the ith bit 0. The all 0's vector is denoted by $0 = 0^n$. The input is denoted $x = (x_1, \dots, x_n)$. When the base is not specified the base of the logarithm is 2. The notation $O(.)$, $\Omega(.)$ may hide factors depending on ε. Set $[n] := \{1, \dots, n\}$. For two sets A and B denote their symmetric difference by $A \triangle B$. $X \sim Y$ means that the random variables X and Y are identically distributed.

3 Tradeoff upper bounds for OR

In this section, we present a PNDT that essentially matches the tradeoff lower bound stated in Theorem 1.1. As mentioned earlier, Newman already gave such a PNDT. His PNDT has an additional property that on non-zero input, with high probability it returns the minimum index $i \in [n]$ such that $x[i] = 1$.

Here we give a different PNDT called FAST_OR. FAST_OR does not have the above additional property but has other advantages: (i) it is simpler, (ii) it reveals a computational bottleneck that leads to our tight lower bound, and (iii) on any input x, with high probability it outputs a subset of indices that is "close" to $\{i \in [n] : x_i = 1\}$. We need this last property in the proof of Theorem 1.3 to design an efficient PNCT for MAX. Both Newman's PNDT and FAST_OR are robust even against the most robust noise model.

For our algorithm, we assume that the noise parameter ε is at most 1/16. Given an algorithm that works with noise parameter $\varepsilon \le 1/16$, we can convert it to one that works for any noise parameter $\varepsilon < 1/2$ by replacing each query to a variable by the majority of a suitably chosen constant number of queries to that variable. This leaves the number of rounds unchanged and changes the number of queries by a constant factor.

Given n and integer $q \ge 2$, $\mathsf{FAST\_OR}_{n,q}$ takes as input $x_1, \dots, x_n$ and uses nq queries and a small number of rounds. The definition of $\mathsf{FAST\_OR}_{n,q}$ depends on two sequences $(q_j : j \ge 1)$ and $(\lambda_j : j \ge 1)$ of parameters. We define $\beta := (\frac{1}{\varepsilon})^{1/8}$ and for $j \ge 1$,

$$\begin{aligned} q_j &:= \mathrm{Tower}_{\beta}^{(j-1)}(q/2)\,, \\ \lambda_j &:= \frac{q}{q_{j+1} 2^{j+1}}\,. \end{aligned}$$

The first sequence increases rapidly like a tower function, and the second decreases rapidly like the reciprocal of the tower function. We now describe $\mathsf{FAST\_OR}_{n,q}$.

$\mathsf{FAST\_OR}_{n,q}$

$\mathsf{FAST\_OR}_{n,q}$ constructs a sequence of sets $[n] =: S_0 \supseteq S_1 \supseteq \cdots$. We write s_j for the size of S_j. During round j for $j \ge 1$, the algorithm determines S_j from S_{j-1} as follows: For each $i \in S_{j-1}$, the algorithm performs q_j (noisy) queries of x_i and places i in S_j if at least $q_j/2$ answers were 1. At the end of round j, FAST_OR terminates with output 0 if $s_j = 0$, and terminates with output 1 if $s_j > \lambda_j s_{j-1}$, and otherwise it continues to round $j+1$.

The basic idea behind $\mathsf{FAST\_OR}_{n,q}$ is that one can increase the number of queries when the number of relevant variables is small, achieving small probability of error and not making too many queries overall. This simple idea has appeared before in the context of noisy decision tree computation. The new element here is the (tight) trade-off between the number of rounds and the number of queries. This requires that we carefully choose the number of queries made in each round. Let us give some intuition for $\mathsf{FAST\_OR}_{n,q}$. $\mathsf{FAST\_OR}_{n,q}$ tries to approximate the set $S := \{i \in [n] : x_i = 1\}$ of indices whose input bit is 1. The algorithm constructs a sequence of successive approximations $S_0 \supset S_1 \supset S_2 \supset S_3 \supset \cdots$ to S. The initial approximation S_0 is the set $[n]$. In round i, the algorithm makes q_i queries to each of the variables indexed by S_{i-1} and defines S_i to be the set of indices of S_i for which the majority of queries return 1. Some indices may be misclassified, but for each individual bit the probability that it is misclassified is small. If $|S_i|$ is not much smaller than S_{i-1} (smaller by a factor of λ_i), then we will deduce that, with high probability, S is well approximated by S_i, and in particular is non-empty, so the OR is 1. Otherwise, we proceed to the $(i+1)st$ round of queries. Note that since $|S_i|/|S_{i-1}| \le \lambda_i$, we can afford to make a larger number of queries (q_i) to each bit without much increase in the overall number of queries, and so the probability that a bit is misclassified decreases (exponentially in the number of queries to it in the previous round).

For each fixed input x, the behavior of the algorithm depends on the random noise present in each query response. As we will see below, the algorithm is guaranteed to terminate, irrespective of the random noise. This is simply because if the algorithm does not stop in round j, then $|S_j|$ is much smaller than $|S_{j-1}|$, and this cannot go on for too many rounds. Let R be the random variable representing the total number of rounds.

For $x \in \{0,1\}^n$, define $A = A(x) := \{i : x_i = 1\}$ and $B = B(x) := \{i : x_i = 0\}$.

Theorem 3.1. *For each $\varepsilon \in (0, 1/16)$, there is a constant $C = C(\varepsilon)$ such that the following holds. Let n, r, q be positive integers satisfying $r \le \log^*_{1/\varepsilon} n - C$, and $q = \lceil 80 \log^{(r)}_{1/\varepsilon}(n) \rceil \le n$. For any input $x \in \{0,1\}^n$, $\mathsf{FAST\_OR}_{n,q}$ uses at most qn queries and the number R of rounds is at most r. Furthermore, the final set S_R satisfies:*

1. *for each $i \in A$, $\Pr[i \notin S_R] \le 1/40000$, and*

2. *$|S_R \triangle A| \le |A|/100$ with probability at least $99/100$.*

This holds for all three noise models.

To compute OR, $\mathsf{FAST\_OR}$ outputs 1 if it finishes with nonempty S_R and outputs 0 otherwise. If all input bits are 0, then $A = \emptyset$ and assertion (2) implies that $S_R = \emptyset$ with probability at least $99/100$. If at least one input bit is 1, then $A \ne \emptyset$ and so for any $i \in A$, assertion (1) implies that $i \in S_R$ with probability at least $1 - 1/40000$. In either case, the algorithm outputs $\mathsf{OR}_n(x)$ with very small probability of error.

Proof. We first observe that it is enough to prove assertions (1) and (2) for the random noise model. It is not hard to show that for each $i \in A$, and for each $j \in [R]$, the event $i \notin S_j$ is monotone with respect to noise (as defined in Subsection 2.1). Therefore by Lemma 2.1, if (1) holds for the random noise model it holds against any clairvoyant adversary. Similarly, for $i \in B$, and $j \in [R]$, the event $i \in S_j$ is monotone with respect to noise. It follows that the event $|S_R \triangle A| \le |A|/100$ is monotone with respect to noise and so it is enough to prove (2) for the random noise model.

Henceforth we assume the random noise model. The constant $C(\varepsilon)$ that is stipulated in the theorem will be specified by various conditions that arise in the argument.

For any j, if $\mathsf{FAST\_OR}$ has not terminated by the end of round $j-1$ then:

$$1 \le s_{j-1} \le \lambda_{j-1} s_{j-2} \le \lambda_{j-1} n = \frac{qn}{q_j 2^j}\,. \tag{3.1}$$

The number of queries in round j is at most $q_j s_{j-1} \le qn/2^j$, and so the total number of queries is at most qn.

We next show that $\mathsf{FAST\_OR}_{n,q}$ uses at most r rounds. Suppose for contradiction that $\mathsf{FAST\_OR}_{n,q}$ has not terminated after r rounds. Then by (3.1) and the definition of q_{r+1}, $qn \ge q_{r+1} = \mathrm{Tower}_\beta^{(r)}(q/2)$, and so:

$$\log_\beta^{(r)}(qn) \ge q/2 \ge 40 \log_{1/\varepsilon}^{(r)}(n)\,. \tag{3.2}$$

By choosing the constant $C(\varepsilon)$ large enough we can ensure by Proposition 2.2(2) that for q, r, n as hypothesized,

$$\log_\beta^{(r)}(qn) \le \log_\beta^{(r)}(n^2) \le 2\log_\beta^{(r)}(n)\,.$$

Using Proposition 2.2(1) and again choosing the constant $C(\varepsilon)$ large enough, this is at most $16\log_{1/\varepsilon}^{(r)}(n)$, which contradicts (3.2).

Next we consider the properties of the final set S_R. For $j \le R$ and for each $i \in S_{j-1}$, $\mathsf{FAST\_OR}$ computes the majority of q_j queries to x_i. Let p_j denote the probability that the majority value disagrees with x_i. Then

$$p_j \le \binom{q_j}{q_j/2} \varepsilon^{q_j/2} \le 2^{q_j} \varepsilon^{q_j/2} \le \varepsilon^{q_j/4} \le \left(\frac{1}{\beta}\right)^{2q_j} = \frac{1}{q_{j+1}^2}\,, \tag{3.3}$$

where the third inequality uses $\varepsilon \le 1/16$.

For $i \in A$,

$$\Pr[i \notin S_R] \le \sum_{i \ge 1} p_i \le \sum_{i \ge 1} \frac{1}{q_{i+1}^2} \le \frac{2}{q_2^2} \le \frac{1}{40000}$$

(with room to spare) since $\varepsilon \le 1/16$ and $q \ge 80$. This completes the proof of the first assertion in the theorem.

To prove the second assertion in the theorem, we now bound

$$\Pr[|A \triangle S_R| \ge |A|/100]$$

from above. Let $A_j := A \cap S_j$ and $B_j := B \cap S_j$; also let $a_j := |A_j|$ and $b_j := |B_j|$. For the event $|A \triangle S_R| \ge |A|/100$ to occur, either $|A_R|$ is too small compared to $|A|$ ($\mathsf{FAST\_OR}$ has lost too many indices i with $x_i = 1$) or B_R is too large. We will show that neither of these is likely.

Let C be the event that $|A - A_R| \ge |A|/200$ and for $j \ge 0$ let Bad_j be the event

$$(j \le R) \wedge (b_j > 200(2^j) p_j b_{j-1})\,.$$

This is the event that in round j of $\mathsf{FAST\_OR}$ the set B_j does not shrink fast enough.

It suffices to prove the following two claims: (i) If none of the events in the set $\{C\} \cup \{\mathrm{Bad}_j : j \ge 1\}$ occur, then $|A \triangle S_R| \le |A|/100$ and (ii) $\Pr[\cup_{j \ge 1} \mathrm{Bad}_j] \le 1/200$ and $\Pr[C] \le 1/200$.

To prove (i), assume that none of the given events occur. Write $|A \triangle S_R| = |A - A_R| + |B_R|$. Since C doesn't occur, $|A - A_R| \leq |A|/200$. It now suffices to show $|B_R| \leq |A|/200$. If $\mathsf{FAST\_OR}$ terminates with $S_R = \emptyset$ then $|B_R| \leq |A|/200$ is trivially satisfied. In the other case, the termination condition of $\mathsf{FAST\_OR}$ and the assumption that Bad_R does not occur imply:

$$a_R + b_R \geq \lambda_R(a_{R-1} + b_{R-1}) \geq \lambda_R b_{R-1} \geq \frac{\lambda_R}{200(2^R)p_R} b_R > \frac{q_{R+1}}{200(2^{2R+1})} b_R\,.$$

Here the last inequality uses the definition of λ_R, upper bound (3.3) on p_R, and the fact that $q > 1$. Using $q \geq 80$ it is easy to see that $q_{R+1}/(200(2^{2R+1}))$ is minimized when $R = 1$, in which case it equals $(1/\varepsilon)^{q/16}/1600$. Using $\varepsilon \leq 1/16$ and $q \geq 80$ this expression is more than 201 and it follows immediately that $b_R \leq a_R/200 \leq |A|/200$.

To prove (ii), we notice that $\mathbb{E}[|A - A_R|] \leq |A|/40000$ (this in the consequence of the first assertion in the theorem proved above) and Markov's inequality imply $\Pr[C] \leq 1/200$.

For each $j \geq 1$, conditioned on the event that $\mathsf{FAST\_OR}$ executes at least j rounds and also conditioned on the value of b_{j-1}, the expectation of b_j is $p_j b_{j-1}$. By Markov's inequality:

$$\Pr[\mathrm{Bad}_j] \leq \frac{1}{2^j 200}$$

and therefore $\Pr[\cup_{j\geq 1}\mathrm{Bad}_j] \leq 1/200$. □

4 Tradeoff lower bounds for OR

In round j, $\mathsf{FAST\_OR}$ keeps an approximation S_j of the set of 1s. The problem for the next round becomes the OR problem for the bits in S_j. The bits in S_j are treated similarly by $\mathsf{FAST\_OR}$ at the end of round j because the answers to the queries to these bits have more 1s than 0s. We can think of S_j as representing the uncertainty $\mathsf{FAST\_OR}$ has about the input. In our lower bound proof, we basically show that the size of S_j is not too small if $\mathsf{FAST\_OR}$ has not made too many queries. More generally, we will show that for any PNDT, in a round there is a set of indices which look the same, and this set does not shrink too fast with the number of rounds.

Given this strategy, a most direct way of working it out would be to (1) use an input distribution over the all 0 vector and vectors with a single 1, its index uniformly randomly distributed over $[n]$, (2) show that when the input x is such that $\mathsf{OR}(x) = 1$, sets S_j are not too small by showing that there are many indices which look the same as the index whose bit is 1, (3) when the input is the all 0 vector, then the distribution of the answers is not too far from when there is one bit with value 1. The last part in this strategy requires rather technical computations, which we can avoid by working with the following problem instead:

Definition 4.1. The "Which Half Problem" (WHP_n).

Input: A $2n$-bit vector with a single 1.

Output: 0, if the 1 appears among the first n bits, and 1 otherwise.

The advantage of working with WHP is that we only need to work with one distribution on the inputs and PNDT lower bounds for WHP_n immediately imply similar bounds for OR_n:

Lemma 4.2. *If there is a qn-query, r-round randomized PNDT for OR_n with error probability at most δ then there is a qn-query, r-round randomized PNDT for WHP_n with error probability at most δ.*

Proof. For a $2n$-bit vector x with a single 1, $\mathsf{WHP}_n(x) = \mathsf{OR}_n(x_{n+1}, \ldots, x_{2n})$, so $\mathsf{WHP}_n(x)$ can be solved using a PNDT for OR_n. □

Let WHP_n^U denote WHP_n with input chosen uniformly at random from $e^1, \ldots, e^{2n}$, where $e^j \in \{0,1\}^{2n}$ has a unique 1 in position j.

Theorem 4.3. *Let $\varepsilon \in (0, 1/2)$. There is a positive integer $C = C(\varepsilon)$ such that for any positive integers n, r with $r \le \log^*_{1/\varepsilon} n - C$, any (possibly randomized) PNDT that solves WHP_n^U in r rounds with error probability less than 1/4 requires at least $\frac{1}{100} n \log^{(r)}_{1/\varepsilon} n$ queries.*

Theorem 1.1 follows immediately using Lemma 4.2.

It suffices to prove Theorem 4.3 for deterministic PNDTs, since a randomized PNDT can be viewed as a probability distribution over deterministic PNDTs and the probability of error of a randomized PNDT running on a fixed input distribution is a suitable average over deterministic trees run over the same input distribution. (This is the easy direction of Yao's lemma [13].)

4.1 The round elimination lemma

Let $\mathsf{err}_n(r,t)$ be the minimum possible error probability of an r-round, deterministic PNDT for WHP_n^U that uses at most t queries. The round elimination lemma relates $\mathsf{err}_n(r,t)$ to $\mathsf{err}_{n'}(r-1,t)$ for some n' that is not much smaller than n:

Lemma 4.4. *(Round elimination) Let $\varepsilon \in (0, 1/2)$ and $\theta \ge 10$. Let n, Q, r be positive integers such that $Q \ge n > (\frac{1}{\varepsilon})^{4\theta Q/n}$, and let n' be a positive integer satisfying $n' \le n\,\varepsilon^{2\theta Q/n}$. Then:*

$$\mathsf{err}_n(r,Q) \ge \mathsf{err}_{n'}(r-1,Q)\left(1 - \frac{5}{\theta}\right).$$

We remark that in the above inequality the total number of queries available for r rounds is equal to the total number of queries available for $r-1$ rounds. In our application of this inequality, after the first round of queries fewer queries will be available for the remaining $r-1$ rounds, however the proof goes through without taking this into account.

Proof. Let X be the random input to WHP_n^U and let K be the (random) index such that $X_K = 1$.

We want to show that any r-round, Q-query deterministic PNDT for WHP_n^U has error probability at least $\mathsf{err}_{n'}(r-1,Q)(1-5/\theta)$. The analysis is simplified by introducing an *advisor* who provides some additional information to the PNDT. We show the desired lower bound holds for PNDTs with this advisor; since the advisor can only reduce the probability of error, the lower bound applies to PNDT without advice.

After the first round, the advisor reveals the values of X at all but n' locations of each half of the input, and also provides some additional information about the unrevealed values. Let T be the event that the unique 1 is revealed, and let $\overline{T}$ be the complementary event. The behavior of the advisor will ensure the following conditions:

1. $\Pr[T] \leq \frac{5}{\theta}$.

2. The conditional distribution on K given $\overline{T}$ is uniform over the set of unrevealed locations.

These two conditions imply the theorem since by Condition 2, the conditional probability that the PNDT makes an error given $\overline{T}$ is at least $\mathsf{err}_{n'}(r-1,Q)$, and thus by Condition 1, the probability of error is at least $\mathsf{err}_{n'}(r-1,Q)(1-5/\theta)$.

We now describe a strategy for the advisor that guarantees these two conditions. The first round of queries of a PNDT for WHP_n is specified by a vector $(m_1,\ldots,m_{2n})$, where m_i is the number of times X_i is queried. By hypothesis, $m_1+\ldots+m_{2n} \leq Q$. Let B_1 (resp. B_2) consist of the $\lfloor n/\theta \rfloor$ indices among the first half (resp. second half) of the variables that are queried most often, breaking ties arbitrarily. Let $A_1 := [n]-B_1$ and $A_2 := [n+1,2n]-B_2$, and $A := A_1 \cup A_2$. Note that for $j \in \{1,2\}$ and $i \in A_j$ since $m_i \leq m_h$ for all $h \in B_j$, we have

$$m_i \leq \frac{1}{|B_j|+1} \sum_{h \in B_j \cup \{i\}} m_h \leq \frac{Q\theta}{n} .$$

Define $m = \lfloor Q\theta/n \rfloor$; since the m_i are integers, we have $m_i \leq m$ for all $i \in A_j$.

We now describe the behavior of the advisor.

Step 1. The advisor reveals the values of $(X_i : i \in B := B_1 \cup B_2)$. If $K \in B$, the computation ends.

Step 2. For each $i \in A$, the advisor provides answers to $m - m_i$ additional (noisy) queries to X_i. Thus for each $i \in A$, the PNDT has m noisy values for X_i.

Step 3. For $i \in A$, let v_i be the number of queries to X_i that reported 1. For $j \in \{1,2\}$, let $S_j := \{i \in A_j : v_i = v_K\}$ and let $s_j := |S_j|$. (S_j is the set of indices that "look the same" as K to the PNDT.) If $s_1 < n'$ or $s_2 < n'$ the advisor reveals the value at location K and the computation ends. Otherwise, the advisor reveals the values of the variables outside $S := S_1 \cup S_2$.

Step 4. The advisor chooses $|s_1 - s_2|$ indices at random from the larger of S_1, S_2 and reveals those values. Let R_1, R_2 be the subsets of S_1, S_2 that remain unrevealed.

Step 5. If K is unrevealed then $|R_1| = |R_2| \geq n'$ and the conditional distribution of K is uniform over $R_1 \cup R_2$. Finally, the advisor chooses a pair $R'_1 \subseteq R_1$ and $R'_2 \subseteq R_2$ of subsets uniformly at random from among the pairs of subsets of size exactly n' such that $K \in R'_1 \cup R'_2$, and reveals all the values of locations in $(R_1 - R'_1) \cup (R_2 - R'_2)$. This maintains the property that K is unrevealed and the location of K is uniformly distributed over $R'_1 \cup R'_2$.

It is clear from this description that Condition 2 is satisfied and it remains to verify Condition 1. For $1 \le i \le 5$, let E_i be the event that the advisor reveals the location of K during step i. The events E_i are disjoint and E_2 and E_5 are empty, so $\Pr[T] = \Pr[E_1] + \Pr[E_3] + \Pr[E_4]$. Since K is independent of the queries performed in the first round

$$\Pr[E_1] = \Pr[K \in B] = \frac{|B|}{2n} \le \frac{1}{\theta}\,.$$

It remains to give an upper bound on $\Pr[E_3] + \Pr[E_4]$. For any event F containing E_3,

$$\Pr[E_3] + \Pr[E_4] \le \Pr[F] + \Pr[E_4 \wedge \overline{F}] \le \Pr[F] + \Pr[E_4 | \overline{F}]\,.$$

We will choose such an event F and bound the two terms of this final sum.

For a positive integer s and $\gamma \in [0,1]$, let $B(s,\gamma)$ denote the sum of s independent 0-1 random variables, each equal to 1 with probability γ. Let

$$p(t) := \Pr[B(m,\varepsilon) = t] = \binom{m}{t} \varepsilon^t (1-\varepsilon)^{m-t}\,.$$

Observe that for any $t \in [0,m]$, $p(t) \ge \varepsilon^m$, using the fact that $\varepsilon \le 1/2$. Define

$$w(t) := \mathbb{E}\big[B(n - \lfloor n/\theta \rfloor, p(t))\big] = (n - \lfloor n/\theta \rfloor) p(t)\,.$$

Let $W = w(v_K)$; thus W is a random variable depending on v_K. For $j \in \{1,2\}$ we define F_j to be the event that $|s_j - W| > W/\theta$ and F to be the event $F_1 \vee F_2$. To show that Condition 1 holds it now suffices to show:

(A1) $E_3 \subseteq F$.

(A2) $\Pr[F_j] \le 1/\theta$ for $j \in \{1,2\}$.

(A3) $\Pr[E_4 | \overline{F}] \le 2/\theta$.

For (A1), if E_3 holds, then for some $j \in \{1,2\}$,

$$s_j < n' \le n\varepsilon^{2m} \le n\varepsilon^m p(v_K) \le \left(1 - \frac{1}{\theta}\right) w(v_K)\,,$$

which implies F. (The last inequality is very loose.)

Next, we bound $\Pr[F_j]$ for $j \in \{1,2\}$. Fix $k \in \{1,\dots,2n\}$ and $t \in [0,m]$ arbitrarily. We show that

$$\Pr[F_j | (K = k) \wedge (v_K = t)] \le \frac{1}{\theta}\,,$$

from which $\Pr[F_j] \le 1/\theta$ follows immediately. Condition on the event $(K = k) \wedge (v_K = t)$. Then for $i \in [n]$, let Y_i be the indicator of the event $i \in S_j$. Thus $s_j = \sum_{i \in A_j} Y_i$. We have $Y_k = 1$ and, for $i \in A_j - \{k\}$, $Y_i \sim B(1, p(t))$. If $k \notin A_j$ then $s_j \sim B(n - \lfloor n/\theta \rfloor, p(t))$, and otherwise $s_j \sim B(n - \lfloor n/\theta \rfloor - 1, p(t)) + 1$; in either case we can write $s_j = Z + Y$, where $Z \sim B(n - \lfloor n/\theta \rfloor, p(t))$ and Y is a random variable satisfying

$0 \le Y \le 1$. Noting that Z has variance $\mathbf{Var}[Z] = (1-p(t))\mathbb{E}[Z] \le \mathbb{E}[Z] = w(t)$ and using Chebyshev's inequality we get:

$$\begin{aligned}\Pr[F_j|(K=k)\wedge(v_k=t)] &= \Pr\left[|s_j - w(t)| > \frac{w(t)}{\theta}\right] \le \Pr\left[|Z-w(t)| \ge \frac{w(t)}{\theta} - 1\right] \\ &\le \Pr\left[|Z-w(t)| \ge \frac{w(t)}{2\theta}\right] \le \frac{4\mathbf{Var}[Z]\theta^2}{\mathbb{E}[Z]^2} \le \frac{4\theta^2}{\mathbb{E}[Z]} \le \frac{4\theta^2}{\varepsilon^m(n-\lfloor\frac{n}{\theta}\rfloor)} \le \frac{1}{\theta}.\end{aligned}$$

The second inequality follows from $w(t) \ge 2\theta$, which in turn follows from $w(t) \ge (n - \lfloor n/\theta \rfloor)\varepsilon^{\lfloor \theta Q/n \rfloor}$ together with the hypotheses $n \ge (1/\varepsilon)^{4\theta Q/n}$, $\theta \ge 10$, $\varepsilon < 1/2$ and $Q \ge n$. The final inequality also follows easily from these hypotheses. Thus we have shown that $\Pr[F_1] = \Pr[F_2] \le 1/\theta$.

It remains to upper bound $\Pr[E_4]$. The event E_4 occurs if x_K is among the $|s_1 - s_2|$ variables of $S_1 \cup S_2$ that the advisor reveals. This happens with probability at most $|s_1 - s_2|/(s_1+s_2)$. Given $\overline{F}$, we have $W(1-1/\theta) \le s_1, s_2 \le W(1+1/\theta)$, which implies $|s_1-s_2|/(s_1+s_2) \le 2/\theta$, using $\theta \ge 10$.

Summing the above three upper bounds, we conclude that $\Pr[T] \le 5/\theta$, as required to verify Condition 1. □

4.2 Proof of Theorem 4.3

In this section we prove Theorem 4.3 using Lemma 4.4. To do this we extend Lemma 4.4. Recall that we already noted that Theorem 1.1 follows immediately from Theorem 4.3 and Lemma 4.2.

Lemma 4.5 (Multiple-Round elimination). *Let $\varepsilon \in (0,1/2)$ and $\theta \ge 10$. Let n, Q, r be positive integers with $Q \ge n$. Define the sequence $(n_j : j \ge 0)$ by the recurrence $n_0 = n$ and for $j \ge 1$,*

$$n_j = \lfloor n_{j-1}\varepsilon^{2^j\theta Q/n_{j-1}} \rfloor .$$

For a natural number i, let $I(i)$ denote the inequality:

$$I(i) : n_i \varepsilon^{2^{i+2}\theta Q/n_i} \ge 1,$$

If j is a natural number such that $I(j-1)$ holds then:

$$\mathsf{err}_n(r,Q) \ge \mathsf{err}_{n_j}(r-j,Q)\left(1 - \frac{10}{\theta}(1-2^{-j})\right).$$

Proof. We proceed by induction on j; the case $j=1$ follows immediately from Lemma 4.4. So assume $j \ge 2$ and $I(j-1)$ holds. Since n_j is a decreasing function of j, $I(j-2)$ holds and so by the induction hypothesis:

$$\mathsf{err}_n(r,Q) \ge \mathsf{err}_{n_{j-1}}(r-j+1,Q)\left(1 - \frac{10}{\theta}(1-2^{-(j-1)})\right).$$

Now by Lemma 4.4 with parameters n, Q, θ, n' in that lemma replaced by $n_{j-1}, Q, 2^{j-1}\theta, n_j$, we have

$$\mathsf{err}_{n_{j-1}}(r-j+1,Q) \ge \mathsf{err}_{n_j}(r-j,Q)\left(1 - \frac{10}{\theta 2^j}\right).$$

Substituting this in to the right-hand side of the previous inequality we get:

$$\begin{aligned}\mathsf{err}_n(r,Q) &\geq \mathsf{err}_{n_j}(r-j,Q)\left(1-\frac{10}{\theta 2^j}\right)\left(1-\frac{10}{\theta}(1-2^{-(j-1)})\right)\\ &\geq \mathsf{err}_{n_j}(r-j,Q)\left(1-\frac{10}{\theta}(1-2^{-j})\right).\end{aligned}$$

□

Setting $j=r$ and $\theta=20$ in the lemma, and noting that $\mathsf{err}_n(0,Q)=1/2$ (since a 0-round algorithm has probability at most 1/2 to guess the correct half) we obtain:

Corollary 4.6. *Let $\varepsilon \in (0,1/2)$. Let n,Q,r be positive integers with $Q \geq n$ and let $\theta = 20$. Define the sequence $(n_j : j \geq 0)$ and the condition $I(j)$ as in Lemma 4.5. If r is a natural number such that $I(r-1)$ holds then:*

$$\mathsf{err}_n(r,Q) \geq \frac{1}{2}\left(1-\frac{1}{2}(1-2^{-r})\right) \geq \frac{1}{4}.$$

We are now ready to prove Theorem 4.3.

Proof of Theorem 4.3. Set $\theta=20$ as in Corollary 4.6 and define Q to be:

$$Q = \frac{1}{160} n \log^{(r)}_{1/\varepsilon}(n). \tag{4.1}$$

If r is an integer satisfying

$$r \leq \log^*_{1/\varepsilon}(n) - \log^*_{1/\varepsilon}(160), \tag{4.2}$$

then $Q \geq n$. We will show further that condition $I(r-1)$ holds and then Corollary 4.6 implies

$$\mathsf{err}_n(r,Q) \geq \frac{1}{4},$$

as needed to prove the theorem.

From the recurrence for n_j we have:

$$n_j = \lfloor n_{j-1}\varepsilon^{2^j\theta Q/n_{j-1}}\rfloor \geq n_{j-1}\varepsilon^{2^{j+1}\theta Q/n_{j-1}}. \tag{4.3}$$

Define $q_j = 2^{j+3}\theta Q/n_j$. Note that $q_0 = 8\theta Q/n \geq 160$ since $\theta = 20$, and also note that (q_j) is an increasing sequence since (n_j) is a decreasing sequence. Condition $I(r-1)$ can be rewritten as:

$$2^{r+2}\theta Q \geq q_{r-1}(1/\varepsilon)^{q_{r-1}/2}.$$

Since the left hand side is larger than n and the right hand side is smaller than $(1/\varepsilon)^{q_{r-1}}$, $I(r-1)$ follows from:

$$\log_{1/\varepsilon} n \geq q_{r-1}. \tag{4.4}$$

Substituting the definition of q_j into (4.3), we get that for $j \geq 1$:

$$2^{j+3}\theta Q/q_j \geq 2^{j+2}\theta Q/q_{j-1}\varepsilon^{q_{j-1}/2},$$

which can be rewritten as:

$$q_j \le 2q_{j-1}(1/\varepsilon)^{q_{j-1}/2} \le (1/\varepsilon)^{q_{j-1}},$$

from which we conclude that $q_{r-1} \le \mathrm{Tower}_{1/\varepsilon}^{(r-1)}(160Q/n)$. Substituting definition (4.1) for Q gives (4.4) as required. □

5 Comparison trees

In this section we prove Theorems 1.2 and 1.3.

5.1 Deterministic comparison trees

Proof sketch for Theorem 1.2. Our goal is to give an algorithm that finds the maximum of n numbers in $O(\log\log n)$ rounds and a linear number of comparison queries, even against a clairvoyant adversary. The algorithm we use is a fault tolerant version of the Shiloach-Vishkin algorithm [10] mentioned earlier.

First we describe a simple one round algorithm, TRIVMAX, which takes as input a subset S of $[n]$ and an odd integer parameter k and attempts to output the index $i \in S$ for which y_i is maximum. For each pair i, j of distinct indices in S, TRIVMAX performs the comparison x_i, x_j k times. and declares the “winner” to be the one that is larger in a majority of comparisons. If there is some index that is the winner against every other index then it is output as the maximum, otherwise TRIVMAX outputs an arbitrary index of S.

TRIVMAX performs $k\binom{|S|}{2}$ comparisons. For each pair i, j, the probability that the winner is selected incorrectly can be bounded above by $C(\varepsilon)^k$ where $C(\varepsilon) < 1$ is a constant depending only on ε (in fact, it is easy to show that $C(\varepsilon) \le 2\sqrt{\varepsilon(1-\varepsilon)}$). Thus the probability that the maximum is not correctly identified is at most $|S|C(\varepsilon)^k$.

We now define a parametrized multiround algorithm based on repeated use of TRIVMAX. The algorithm maintains a subset of $[n]$ whose members are candidates for the index at which the maximum occurs. In each round, the algorithm reduces the set of candidates. During the round, the algorithm splits the candidates into groups of (nearly) equal size and performs TRIVMAX on each group. The winners for each group advance to the next round. For simplicity, let us assume that n is a power of 2. The algorithm takes as parameters r (the number of rounds) and integers $g_1, \dots, g_r, k_1, \dots, k_r$ where each g_i is a power of 2 which specifies the size of the groups in round i (and $g_1 g_2 \cdots g_r = n$) and k_i is a positive odd integer which specifies the number of times each comparison is performed in round i. For $0 \le j \le r$, $n_j := n/\prod_{i=1}^{j} g_i$ is then the number of candidates that remain after j rounds; note that $n_0 = n$ and $n_r = 1$.

Since each candidate participates in $k_i(g_i - 1)$ pairwise comparisons, the number of comparisons in round i is $k_i n_{i-1}(g_i - 1)/2$. The algorithm fails if the maximum is eliminated in some round; the probability that the maximum is eliminated in round i is bounded by $g_i C(\varepsilon)^{k_i}$, where $C(\varepsilon)$ is the constant arising in the analysis of TRIVMAX (defined above).

It now remains to choose the parameters g_i, k_i so that the total number of comparisons is linear and the probability that the maximum is eliminated is bounded below 1/4, while minimizing the number of rounds. By Valiant’s lower bound the number of rounds must be $\Omega(\log\log n)$. Here is one way (of many possible ways) to match this bound. All logarithms in this description are to the base 2. Let $h = \lceil \log\log n \rceil$.

The rounds are divided into two stages. The first stage consists of $2h$ rounds. For $1 \le i \le 2h$, we take $g_i = 2$ and $k_i = 2W_1 i + 1$ for some suitably large constant integer W_1. This reduces the size of the candidate set to $n_{2h} = n/4^h \le n/(\log n)^2$. The number of comparisons during round i is

$$\frac{(2W_1 i + 1)n_{i-1}}{2} = \frac{(2W_1 i + 1)n}{2^i},$$

so the total number of comparisons in the first $2h$ rounds is $O(n)$. Also, the probability that the maximum is eliminated in round i is bounded above by $\lambda(W_1)^i$ where $\lambda(W_1)$ can be made arbitrarily small by choosing W_1 large enough. So the probability that the maximum is eliminated in the first T rounds can be made arbitrarily small.

Now let q be the least integer such that $2^{2^q - 1} \ge n_{2h}$; note $q \le \lceil \log\log n \rceil$. The second phase consists of q rounds. For rounds $2h+i$, $1 \le i \le q-1$, set $g_{2h+i} := 2^{2^{i-1}}$; this implies that $n_{2h+i} = n_{2h}/2^{2^i - 1}$. In particular, $n_{2h+q-1} = n_{2h}/2^{2^{q-1}-1} \le 2^{2^{q-1}}$ by the assumption on q. Set $g_{2h+q} := n/n_{2h+q-1}$; this ensures that $n = \prod_{i=1}^{2w+q} g_i$ and so after $2h+q = O(\log\log n)$ rounds we have reduced to a single candidate.

Set $k_{2h+i} = 2\lceil W_2 \log n \rceil + 1$ for each $1 \le i \le q$, where W_2 is a suitably large constant. The number of comparisons in round $2h+i$ is $O(n_{2h+i-1} g_{2h+i} W_2 \log n) = O(n/\log n)$, so the total number of comparisons in the second phase is $o(n)$.

The probability that the maximum is eliminated in round $2h+i$ is at most $g_{2w+i} C(\varepsilon)^{k_i}$ which is $o(1/n)$ for an appropriate choice of W_2. Thus the chance that the maximum is eliminated in the second stage can be made arbitrarily small. □

5.2 Randomized comparison trees

Proof sketch for Theorem 1.3. **Lower bound.** As mentioned before, we prove the tradeoff lower bound for MAX_n by reducing OR_n to it. Suppose we have a PNCT that computes MAX_n of n distinct inputs in the random noise model; we will show how to use it to compute OR_n on a given Boolean input $x = (x_1, \dots, x_n)$. Define $y_i := x_i + i/2n$ so $y_1, \dots, y_n$, so the y_i are distinct. If we determine the index j such that y_j is maximum, then $\mathsf{OR}_n(x) = x_j$. We then finish the algorithm by reading x_j in parallel $2s+1$ times (for some suitably large constant s) and taking the majority value to be the output.

To find $\mathsf{MAX}_n(y)$ we simulate the comparison queries of the PNCT for MAX_n using the Boolean queries: To compare y_i and y_j with $i < j$ we query x_i and x_j. Call the returned values x'_i and x'_j, and answer the simulated query with $y_i > y_j$ if $x'_i = 1$ and $x'_j = 0$ and with $y_j > y_i$ otherwise.

In the above simulation, the probability of error for the simulated comparison query depends on the values of x_i and x_j. For example, if $n = 2$ and $(x_1, x_2) = (0,0)$, then $(y_1, y_2) = (1/4, 1/2)$, and so $y_1 < y_2$. The probability that the simulated query incorrectly answers $y_2 < y_1$ is $\varepsilon(1-\varepsilon)$, which corresponds to the event $(x'_1, x'_2) = (1,0)$. On the other hand, if $(x_1, x_2) = (0,1)$, then $(y_1, y_2) = (1/4, 3/2)$, and so $y_1 < y_2$. But now the probability that the simulated query incorrectly answers $y_2 < y_1$, is ε^2, again corresponding to the event $(x'_1, x'_2) = (1,0)$.

Since our PNCT for MAX_n is only guaranteed to work for the random noise model in which the probability of error of each comparison is the same fixed value, we need to modify the above simulation so that every comparison query has the same probability of error. To this end, we randomly *perturb* the answers of the above simulation with small probabilities so that the errors for all comparison queries are the same.

The modified simulation requires four *perturbation probabilities* $q_{00}, q_{01}, q_{10}, q_{11}$ which we will specify below. To simulate the comparison x_i, x_j for $i < j$ we again perform noisy queries to x_i and x_j to get x'_i and x'_j. The comparison query answers $x_i > x_j$ with probability $q_{x'_i, x'_j}$ and $x_j > x_i$ with probability $1 - q_{x'_i, x'_j}$.

We want to choose the perturbation probabilities so that the probability of error α of comparison queries is fixed and not too large. We have the following equations:

$$\begin{aligned}
&\text{Equation } (0,0) \qquad \alpha = q_{00}(1-\varepsilon)^2 + q_{01}(1-\varepsilon)\varepsilon + q_{10}(1-\varepsilon)\varepsilon + q_{11}\varepsilon^2\,,\\
&\text{Equation } (0,1) \qquad \alpha = q_{00}(1-\varepsilon)\varepsilon + q_{01}(1-\varepsilon)^2 + q_{10}\varepsilon^2 + q_{11}(1-\varepsilon)\varepsilon\,,\\
&\text{Equation } (1,0) \qquad \alpha = (1-q_{00})(1-\varepsilon)\varepsilon + (1-q_{01})\varepsilon^2 + (1-q_{10})(1-\varepsilon)^2 + (1-q_{11})(1-\varepsilon)\varepsilon\,,\\
&\text{Equation } (1,1) \qquad \alpha = q_{00}\varepsilon^2 + q_{01}(1-\varepsilon)\varepsilon + q_{10}(1-\varepsilon)\varepsilon + q_{11}(1-\varepsilon)^2\,.
\end{aligned}$$

Equation (u,v) corresponds to the case $(x_i, x_j) = (u,v)$ and expresses the probability that the simulated comparison gives the incorrect answer. In each equation, there are four terms corresponding to the possible query results (x'_i, x'_j). For example, in equation $(0,0)$, the first term comes from the fact that $(x'_i, x'_j) = (0,0)$ with probability $(1-\varepsilon)^2$ and then the probability that the response is $x_i > x_j$ q_{00}.

Solving these equations for the qs gives:

$$\begin{aligned}
q_{00} = q_{11} &= (\alpha - \varepsilon + \varepsilon^2 - 2\alpha\varepsilon + 2\alpha\varepsilon^2)/(1-2\varepsilon)^2\,,\\
q_{01} &= (\alpha + \varepsilon^2 - 4\alpha\varepsilon + 2\alpha\varepsilon^2)/(1-2\varepsilon)^2\,,\\
q_{10} &= (1 - \alpha - 2\varepsilon + \varepsilon^2 + 2\alpha\varepsilon^2)/(1-2\varepsilon)^2\,.
\end{aligned}$$

We want to choose α to be small, so that these are all in the range $[0,1]$. For example, $\alpha = \sqrt{\varepsilon}$ gives:

$$\begin{aligned}
q_{00} = q_{11} &= \sqrt{\varepsilon} + O(\varepsilon)\,,\\
q_{01} &= 1 - \sqrt{\varepsilon} + O(\varepsilon)\,,\\
q_{10} &= \sqrt{\varepsilon} + O(\varepsilon)\,.
\end{aligned}$$

These are all in $[0,1]$ provided that ε is small enough. We have shown that there is a constant $\varepsilon_1 \in (0,1)$ such that for all positive $\varepsilon \le \varepsilon_1$ we can simulate a comparison query with a fixed probability of error $\sqrt{\varepsilon}$ using two Boolean queries each with probability of error ε.

Using Theorem 1.1 this shows that any $\sqrt{\varepsilon}$-noisy comparison decision tree for MAX_n with $O(n)$ queries must use $\Omega(\log^*_{1/\sqrt{\varepsilon}} n)$ rounds. Since $\log^*_{1/\varepsilon} n = \Theta(\log^*_{1/\sqrt{\varepsilon}} n)$, an ε-noisy decision tree with $O(n)$ queries needs $\Omega(\log^*_{1/\varepsilon} n)$ rounds, completing the proof of the lower bound part of Theorem 1.3.

Upper bound. Our PNCT for MAX_n borrows the sampling idea used by Reischuk [8] to design a noise-free PCT for MAX_n with $O(1)$ rounds and $O(n)$ queries. Let $y_1, \ldots, y_n$ denote the inputs, which are distinct elements of the totally ordered set U.

The PNCT uses two subroutines TRIVMAX (which was defined in the previous subsection) and APXM.

APXM takes as input a subset S of $\{1,\dots,n\}$ and $u \in \{1,\dots,n\}$ and returns a subset R of S such that with probability at least $1-1/90$, $\max(S) \in R$, and $|R \triangle S_{\geq u}| \leq |S_{\geq u}|/100$ where $S_{\geq u}$ is the set $\{i \in S : y_i \geq y_u\}$.

APXM(S,u) is derived from the PNDT for Theorem 3.1: We create a Boolean input $z = (z_1,\dots,z_n)$ for the PNDT by setting $z_i = 1$ if $y_i > y_u$, and $z_i = 0$ otherwise. We then run FAST_OR on z. Theorem 3.1 implies that APXM(S,u) has the properties claimed with probability at least $1-1/100-1/40000 > 1-1/90$.

APXM uses $O(|S|)$ queries and $O(\log^* |S|)$ rounds.

Now we present our PNCT for MAX_n. The parameter k for TRIVMAX is $K \log n$ for a suitably large constant K. Let $T_0 := [n]$.

```
PNCT for $\mathrm{MAX}_n$

1 Uniformly randomly choose $T_1 \subset \{1,\dots,n\}$ of size $n^{1/3}$.
      $m_1 = \mathrm{TRIVMAX}(T_1)$.
      $T_2 = \mathrm{APXM}(T_0, m_1)$.

2 Uniformly randomly choose $T_3 \subset T_2$ of size $n^{1/3}$.
      $m_2 = \mathrm{TRIVMAX}(T_3)$.
      $T_4 = \mathrm{APXM}(T_2, m_2)$.
      If $|T_4| > 100\,n^{1/3}$, HALT with ERROR.

3 Output $\mathrm{TRIVMAX}(T_4)$.
```

We sketch the routine analysis. Since the elements of T_1 are chosen uniformly randomly, with very high probability, $S_{>m_1}$ has at most $50n^{2/3}$ elements, and the properties of APXM imply that with high probability T_2 contains the index of the overall maximum and has size at most $60n^{2/3}$. Similarly, with high probability T_4 contains the index of the overall maximum and has size at most $1000n^{1/3}$.

The total number of comparisons is $O(n)$ since TRIVMAX is run only on sets of size $O(n^{1/3})$ and APXM takes at most $O(n)$ comparisons. □

Finally we prove Corollary 1.4 which shows how to simulate a randomized PNCT for MAX_n in the random noise model by a deterministic PNCT.

Proof of Corollary 1.4. We adapt the above randomized PNCT for MAX_n. The above PNCT uses randomness for choosing the sets T_1 and T_3. For each of these it needs $O(\log \binom{n}{n^{1/3}}) = O(n^{1/3} \log n)$ random bits. It can be simulated by a $O(n)$-query, $\log^*_{1/\varepsilon} n$-round deterministic PNCT because it has access to $O(n)$ biased random bits (it can just query a single variable repeatedly to generate biased random bits). This simulation can be done, for example, using von Neumann's procedure [12] for generating perfectly random bits from independent random bits with common but unknown bias: take noisy samples of x_1 in pairs. Ignore pairs that return the same value. If the pair returns 01 interpret this as a 0 and if the pair returns 10 interpret this as a 1. □

6 Future work

There are several existing results in the literature about the NDT query complexity (without restriction on rounds) for some specific Boolean functions, e. g., MAJORITY and PARITY. This paper gives tradeoff results between rounds and queries for OR function. It would be nice to obtain analogous results for other functions. More generally, one might hope to characterize the query/rounds tradeoff for any Boolean function in terms of combinatorial properties of the function.

For the noisy comparison tree model, one can ask questions about for SELECTION similar to the ones we studied above for MAX, Note that answers to these questions are known for the noise-free comparison tree model by the work of Ajtai et al. [1] and Reischuk [8] for deterministic and randomized algorithms respectively.

Acknowledgements

We would like to thank anonymous referees of previous versions for their helpful comments.

References

[1] MIKLÓS AJTAI, JÁNOS KOMLÓS, W. L. STEIGER, AND ENDRE SZEMERÉDI: Optimal parallel selection has complexity $O(\log\log n)$. *J. Comput. System Sci.*, 38(1):125–133, 1989. [doi:10.1016/0022-0000(89)90035-4]. 132

[2] WILLIAM EVANS AND NICHOLAS PIPPENGER: Average-case lower bounds for noisy Boolean decision trees. *SIAM J. Comput.*, 28(2):433–446, 1998. [doi:10.1137/S0097539796310102]. 113

[3] URIEL FEIGE: On the complexity of finite random functions. *Inform. Process. Lett.*, 44(6):295–296, 1992. [doi:10.1016/0020-0190(92)90102-2]. 113

[4] URIEL FEIGE AND JOE KILIAN: Finding OR in a noisy broadcast network. *Inform. Process. Lett.*, 73(1-2):69–75, 2000. [doi:10.1016/S0020-0190(00)00002-8]. 113, 114, 115, 117

[5] URIEL FEIGE, PRABHAKAR RAGHAVAN, DAVID PELEG, AND ELI UPFAL: Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, 1994. [doi:10.1137/S0097539791195877]. 113, 114, 115

[6] EYAL KUSHILEVITZ AND YISHAY MANSOUR: Computation in noisy radio networks. *SIAM J. Discrete Math.*, 19(1):96–108, 2005. [doi:10.1137/S0895480103434063]. 113

[7] ILAN NEWMAN: Computing in fault tolerance broadcast networks. In *Proc. IEEE Conf. on Comput. Complex.*, pp. 113–122. IEEE Comp. Soc. Press, 2004. [doi:10.1109/CCC.2004.1313813]. 113, 114, 117

[8] RÜDIGER REISCHUK: Probabilistic parallel algorithms for sorting and selection. *SIAM J. Comput.*, 14(2):396–409, 1985. [doi:10.1137/0214030]. 116, 130, 132

[9] RÜDIGER REISCHUK AND BERND SCHMELTZ: Reliable computation with noisy circuits and decision trees—A general $n \log n$ lower bound. In *Proc. 32nd FOCS*, pp. 602–611. IEEE Comp. Soc. Press, 1991. [doi:10.1109/SFCS.1991.185425]. 113

[10] YOSSI SHILOACH AND UZI VISHKIN: Finding the maximum, merging, and sorting in a parallel computation model. *J. Algorithms*, 2(1):88–102, 1981. 115, 128

[11] LESLIE G. VALIANT: Parallelism in comparison problems. *SIAM J. Comput.*, 4(3):348–355, 1975. [doi:10.1137/0204030]. 114, 115

[12] JOHN VON NEUMANN: Various techniques used in connection with random digits. In A. S. HOUSEHOLDER, G. E. FORSYTHE, AND H. H. GERMOND, editors, *Monte Carlo Method*, volume 12 of *National Bureau of Standards Applied Mathematics Series*, pp. 36–38. 1951. 131

[13] ANDREW CHI-CHIH YAO: Probabilistic computations: Toward a unified measure of complexity. In *Proc. 18th FOCS*, pp. 222–227. IEEE Comp. Soc. Press, 1977. [doi:10.1109/SFCS.1977.24]. 123

AUTHORS

Navin Goyal
Researcher
Microsoft Research India, Bangalore
navingo@microsoft.com
`http://research.microsoft.com/en-us/people/navingo/`

Michael Saks
Professor
Rutgers University
saks@math.rutgers.edu
`http://www.math.rutgers.edu/~saks/`

ABOUT THE AUTHORS

NAVIN GOYAL graduated from Rutgers University in 2005; his advisor was Mike Saks. His thesis focused on information theoretic limitations on computation in models such as communication protocols. He moved to his native India in 2009 after postdocs at McGill University and Georgia Tech. He gets easily excited about many research topics and has worked in several areas after his thesis, but continues to be interested in the interplay between computation and information.

MICHAEL SAKS is a Professor of Mathematics at Rutgers University. He has worked in a wide range of areas within theoretical computer science including circuit lower bounds, decision tree complexity, communication complexity, derandomization, online algorithms, and distributed computing. Most recently he has been thinking about communication complexity lower bounds, sublinear time approximation algorithms, polynomial identity testing, and algorithms for stable instances of NP-hard problems.