Rounds vs Queries Trade-off in Noisy Computation

Navin Goyal* Michael Saks†

September 11, 2006

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 trade-offs 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.

1 Introduction

There is a vast literature devoted to the impact of noise on computation and communication. Within the theory of computing, noise has been studied in the context of decision trees [12, 25, 10, 8, 21], formulas and circuits [23, 14, 28, 17, 9] , various kinds of communication protocols [15, 26, 18, 11, 21], sorting networks [19], cellular automata [13], quantum computation [3, 5], and other situations, e.g. [1, 7, 22, 16, 20, 29]. In this paper we will be concerned with the noisy boolean decision tree and the noisy comparison tree models.

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. When a boolean decision tree is executed in a noisy environment, each reported answer may be incorrect. Several models have been proposed to formalize the notion of noise. In the *random noise model* with noise parameter $\varepsilon \in [0, 1/2)$, the outcome of each query is incorrect independently with probability exactly ε. We say that an algorithm *computes f with error probability δ* (for some $\delta \in [0, 1/2)$) if on each input $x \in \{0,1\}^n$, the algorithm outputs $f(x)$ with probability at least $1 - \delta$.

A noise-free boolean decision tree of depth d can be simulated by a noisy decision tree of depth $O(d \log d)$ with arbitrarily small error: Each query of a variable in the noise-free setting is simulated by taking the majority of $\Omega(\log d)$ 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. [12] showed that to compute 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 [30]. In each step (or *round*), the algorithm may make many queries and the algorithm branches according to the ensemble of answers to all of the queries. 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), where each query answer is subject to noise. See Section 2 for precise definitions. PNDTs were introduced by Newman [21], who used them to construct

*Department of Computer Science, Rutgers University, Piscataway, NJ 08854, USA, email: ngoyal@cs.rutgers.edu. Supported in part by a Louis Bevier Fellowship of Rutgers University and NSF grant CCR-9988526.

†Department of Mathematics, Rutgers University, Piscataway, NJ 08854, USA, email: saks@math.rutgers.edu. Supported in part by NSF grants CCR-9988526 and CCR-0515201

protocols for the noisy broadcast model of distributed computing. We are interested in the trade-off between the the number of rounds and the total number of queries needed to compute OR_n, the OR of n input bits. The $O(n)$-query PNDT for OR of Feige et al. [12] uses $O(\log n)$ rounds. Newman reduced this to $O(\log^* n)$ rounds, 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 protocol is essentially optimal by proving a rounds-query trade-off lower bound.

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

In particular, any PNDT that solves OR_n using $O(n)$ queries needs $\log^*_{1/\varepsilon} n - O(1)$ rounds.

It has been noted (e.g., [11]) that the unrealistic assumptions of the random noise model, that each query experiences independent noise with the same probability, allow for artificial algorithms that exploit this regularity of noise. Other models of noise have been proposed [11] 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. These more robust models are discussed 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 is deduced from a lower bound on a related problem, the "Which half?" problem. Working with this problem substantially simplifies the proof. The input to the "Which half? problem 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 this problem can't be answered reliably with a linear number of queries and a constant number of rounds. The proof is by induction on the number of rounds. The induction step uses *round elimination*: we prove a lemma 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.

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 [30].

The random noise model (and the other more robust models) has a natural analog for comparison trees. Thus we can define parallel noisy comparison trees (PNCTs). Such trees were considered by Feige et al. [12]. Among other results, they showed that the function MAX, which determines the index of the maximum element, can be computed in $O(\log n)$ rounds using $O(n)$ comparaisons, even in the most robust noise model.

We are interested in the trade-offs between the total number of comparisons and the number of rounds used by a PNCT computing the maximum function MAX. We briefly review some known results for MAX in the noise-free setting. Valiant showed that for deterministic PCTs, $\Omega(\log \log n)$ rounds are essential if in each round the algorithm can make at most $O(n)$ comparisons. 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 $O(n)$ comparisons by Shiloach and Vishkin [27]. Thus for deterministic PCTs $\Theta(\log \log n)$ rounds are necessary and sufficient if the total number of somparisons is $O(n)$. For randomized PCTs, Reischuk [24] gave an algorithm with $O(1)$ rounds and $O(n)$ comparisons. which is clearly best possible.

Our first result for the noisy case gives a tight query-rounds tradeoff for randomized PNCTs.

Theorem 2 *There exists a constant $\varepsilon_0 \in (0,1)$ such that for all positive $\varepsilon \leq \varepsilon_0$, $\Theta(\log^*_{1/\varepsilon} n)$ rounds are necessary and sufficient to compute* MAX *of n variables by a randomized PNCT using $O(n)$ comparisons. This holds for each of the noise models.*

The lower bound is proved by a reduction from OR; the upper bound adapts Reischuk's [24] algorithm to the noisy case.

Restricting to deterministic PNCTs, we obtain results that differentiate between the random noise model and the more robust noise models. Indeed, in the random noise model, a deterministic PNCT can simulate access to a sequence of $O(n)$ ε-biased independent random bits by repeatedly querying a single variable. (This capability is one of the unrealistic features of the random noise model). Since the randomized PNCT that gives the upper bound in Theorem 2 uses only $O(n^{1/3} \log n)$ random bits, it can be simulated (with no significant overhead) by a deterministic PNCT.

This is not possible for the more robust models. In these models, the comparison trees are required to be correct even if all noise is eliminated, and so by Valiant's lower bound any deterministic making $O(n)$ comparisons needs $\Omega(\log \log n)$ rounds.

As noted above, for the noise-free case, Shiloach and Vishkin [27] constructed a PCT meeting Valiant's lower bound. We adapt Shiloach and Vishkin's algorithm to the noisy case with the most robust adverary to obtain:

Theorem 3 *For any of the noise models of section 2.1. there is a deterministic PNCT computing* MAX *of n variables in $O(\log_{1/\varepsilon} \log_{1/\varepsilon} n)$ rounds and $O(n)$ comparisons.*

The upper bound in the above theorem is obtained by adapting the PCT of Shiloach and Vishkin [27] mentioned above to the noisy case using techniques from [12]. The idea is that the PCT of [27] leaves enough room to adjust its parameters so that one can make additional comparisons which make the PCT robust to the noise.

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 that is different from Newman's, that meets the lower bound of Theorem 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. 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 each leaf is labeled 0 or 1. An execution of T determines a path from the root as follows: starting from the root, the algorithm asks the query labeling the current node and follows the branch indicated by the response to the next node. 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. In the *random noise model* with parameter $\varepsilon \in [0, 1/2)$, each response is answered incorrectly with probability ε, and the query responses are mutually independent. Thus the path followed is a random variable depending on the noise.

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

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 multi-set of variables corresponding to a collection of queries made in parallel. There are 2^m branches coming from v (where m is the cardinality of the multi-set), corresponding to the possible responses to these queries. Noiseless and noisy executions of PDTs are defined in the obvious way. In the noiseless setting, the collection of queries labeling a node may be taken to be a set (rather than a multi-set).

2.1 More robust noise models

As mentioned in the introduction, models of noise other than the random noise model have been proposed. Given a PNDT, each query that appears in the PNDT is subject to noise, and we represent that noise by a random variable. A *noise distribution* for the PNDT is any joint distribution over these variables. A general noise model consists of a family of allowed joint distributions over the noise variables.

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 [11] (also called the *fault tolerance* model [21]), 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* [11] 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.

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 $s \geq 1$ by:

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

We define $\log_b^*(x)$ is defined to be the least integer r such that $\mathrm{Tower}_b^{(r)}(1) \geq x$.

The base b *k-iterated log function* is defined for real numbers x that satisfy $\log_b^*(x) \geq k$ by:

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

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

1. $\log_a^*(x) - T \leq \log_b^*(x) \leq \log_a^*(x)$.
2. $\log_a^{(k)}(x) \leq (\log_a(b)) \log_b^{(k)}(x)$.
3. *For $k \leq \log_a^*(x) - T$,* $\log_a^{(k)}(x^2) \leq 2 \log_a^{(k)}(x)$.

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 $x = (x_1, \ldots, x_n)$. When the base is not specified the base of the logarithm is 2. Notation $O(.)$, $\Omega(.)$ may hide factors depending on ε. Set $[n] := \{1, \ldots, n\}$. For two sets A and B denote their symmetric difference by $A \Delta B$. $X \sim Y$ means that the random variables X and Y are identically distributed.

3 Trade-off upper bounds for OR

In this section, we present a PNDT that essentially matches the lower bound trade-off stated in Theorem 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 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 2 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 $1/16$, 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 $q \ge 1$, $\mathsf{FAST\_OR}_{n,q}$ takes as input $x_1, \ldots, x_n$ 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 $j \ge 1$,

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

$\mathsf{FAST\_OR}_{n,q}$ 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 $j+1$.

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. 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 5 *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 $i \in A$, $\Pr[i \notin S_R] \le \frac{1}{40000}$, and (2) $|S_R \Delta A| \le |A|/100$ with probability at least $99/100$.*

Proof. The constant $C(\varepsilon)$ will be specified by various conditions that arise in the argument.

For any j, if FAST_OR has not terminated by the end of round $j-1$ then:

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

Thus 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 (1) and the definition of q_{r+1}, $qn \ge q_{r+1} = \mathrm{Tower}_\beta^{(r)}(q/2)$, and so:

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

By choosing the constant $C(\varepsilon)$ large enough we can ensure by Proposition 4(3) that for q, r, n as hypothesized, $\log_\beta^{(r)}(qn) \le 2\log_\beta^{(r)}(n)$. Using Proposition 4(2) and again choosing the constant $C(\varepsilon)$ large enough, this is at most $32\log_{1/\varepsilon}^{(r)}(n)$, which contradicts (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 this majority value disagrees with x_i (which, of course, is independent of i). Then

$$p_j \le \binom{q_j}{q_j/2} \varepsilon^{q_j/2} \le 2^{q_j} \varepsilon^{q_j/2} \le \varepsilon^{q_j/4} \le (1/\beta)^{2q_j} \le 1/q_{j+1}^2, \tag{3}$$

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

For $i \in A$, $\Pr[i \notin S_R] \le \sum_{i \ge 1} p_i \le \sum_{i \ge 1} 1/q_{i+1}^2 \le 2/q_2^2 \le 1/40000$ (with room to spare) since $\varepsilon \le 1/16$ and $q \ge 80$.

We now bound $\Pr[|A \Delta S_R| \ge |A|/100]$ from above. Let $A_j := A \cap S_j$ and $B_j := B \cap S_j$; also let $a_j := |A_j|$ and $b_j := |B_j|$. Let C be the event that $|A - A_R| \ge |A|/200$ and for $j \ge 0$ let Bad_j be the event $(j \le R) \wedge (b_j > 200(2^j) p_j b_{j-1})$.

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 $|A \Delta S_R| \le |A|/100$ 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 $|A \Delta S_R| = |A - A_R| + |B_R|$. Since C doesn't occur, $|A - A_R| \le |A|/200$. It now suffices to show $|B_R| \le |A|/200$. If $\mathsf{FAST\_OR}$ terminates with $S_R = \emptyset$ then $|B_R| \le |A|/200$ is trivially satisfied. In the other case, the termination condition of $\mathsf{FAST\_OR}$ and the assumption that Bad_R does not occur imply:

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

Clearly $q_{R+1}/(200(2^{2R+1}))$ is minimized when $R = 1$, in which case it equals $(1/\varepsilon)^{q/16}/1600$. Using $\varepsilon \le 1/16$ and $q \ge 80$ this expression is more than 201 and it follows immediately that $b_R \le a_R/200 \le |A|/200$.

To prove (ii), we have first that $\mathbb{E}[|A - A_R|] \le |A|/40000$ (this we proved above) and Markov's inequality imply $\Pr[C] \le 1/200$.

For each $j \ge 1$, 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:

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

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

4 Trade-off lower bounds for OR

It is more convenient to prove the lower bound for the following problem:

Definition 6 *The "Which Half? Problem" (WHP_n).*

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

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

PNDT lower bounds for WHP_n imply similar bounds for OR_n:

Lemma 7 *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, $\mathsf{WHP}_n(x) = \mathsf{OR}_n(x_{n+1}, \ldots, x_n)$, 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 $x_1, \ldots, x_{2n}$. We will prove:

Theorem 8 *Let $\varepsilon \in (0, 1/2)$. There is a positive integer $C = C(\varepsilon)$ such that for any positive integers n, r with $r \leq \log^*_{1/\varepsilon} n - C$, any (possibly randomized) PNDT that solves WHP_n^U in r rounds with error probability less than 1/4 requires at least $\frac{1}{100}\log^{(r)}_{1/\varepsilon}(n)$ queries.*

Theorem 1 follows immediately.

It suffices to prove Theorem 8 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 [32].)

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 9 *(Round elimination) Let $\varepsilon \in (0, 1/2)$ and $\theta \geq 8$. Let n, Q, r be positive integers such that $Q \geq n > (\frac{1}{\varepsilon})^{4\theta Q/n}$, and let n' be a positive integer satisfying $n' \leq n\,\varepsilon^{2\theta Q/n}$. Then:*

$$\mathsf{err}_n(r, Q) \geq \mathsf{err}_{n'}(r-1, Q)(1 - \frac{4}{\theta}).$$

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 $\mathsf{err}_{n'}(r-1, Q)(1-\frac{4}{\theta})$. 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. $\Pr[T] \leq \frac{4}{\theta}$.
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 $\mathsf{err}_{n'}(r-1, Q)(1 - \frac{4}{\theta})$.

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 $m = \lfloor \frac{Q\theta}{n} \rfloor$; 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 $1 \leq i \leq 5$, 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

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

It remains to give an upper bound on $\Pr[E_3] + \Pr[E_4]$. For any event F containing E_3, $\Pr[E_3] + \Pr[E_4] \leq \Pr[F] + \Pr[E_4 \wedge \overline{F}] \leq \Pr[F] + \Pr[E_4|\overline{F}]$. 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 $t \in [0,m]$, $p(t) \geq \varepsilon^m$. Define $w(t) := \mathbb{E}[B(n - \lfloor \frac{n}{\theta} \rfloor, p(t))] = (n - \lfloor \frac{n}{\theta} \rfloor)p(t)$. 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:

Step 1 $E_3 \subseteq F$.

Step 2 $\Pr[F_j] \leq \frac{1}{\theta}$ for $j \in \{1,2\}$.

Step 3 $\Pr[E_4|\overline{F}] \leq \frac{1}{\theta}$.

If E_3 holds, then for some $j \in \{1,2\}$,

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

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

Next, we bound $\Pr[F_j]$ for $j \in \{1,2\}$. Fix $k \in \{1,\ldots,2n\}$ and $t \in [0,m]$ arbitrarily. We show that $\Pr[F_j|(K=k) \wedge (v_K = t)] \leq \frac{1}{\theta}$ from which $\Pr[F_j] \leq \frac{1}{\theta}$ follows immediately. 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 $s_j \sim B(n - \lfloor \frac{n}{\theta} \rfloor, p(t))$, and otherwise $s_j \sim B(n - \lfloor \frac{n}{\theta} \rfloor - 1, p(t)) + 1$; in either case we can write $s_j = Z + Y$, where $Z \sim B(n - \lfloor \frac{n}{\theta} \rfloor, p(t))$ and Y is a random variable satisfying $0 \leq Y \leq 1$. Noting that Z has variance $\mathbf{Var}[Z] = (1-p(t))\mathbb{E}[Z] \leq \mathbb{E}[Z]$ and using Chebyshev's inequality we get:

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

The second inequality follows from $w(t) \geq 2\theta$, and this follows from $w(t) \geq (n - \lfloor \frac{n}{\theta} \rfloor)\varepsilon^{\lfloor \theta Q/n \rfloor}$ together with the hypotheses $n \geq (1/\varepsilon)^{4\theta Q/n}$, $\theta \geq 8$, $\varepsilon < 1/2$ and $Q \geq n$. The final inequality also follows easily from these hypotheses.

The event E_4 occurs if x_K is among the $|s_1 - s_2|$ variables of $S_1 \cup S_2$ that the oracle reveals. This happens with probability at most $|s_1 - s_2|/s_1 + s_2$. Given $\overline{F}$, $W(1-1/\theta) \leq s_1, s_2 \leq W(1+1/\theta)$ which implies $|s_1 - s_2|/(s_1 + s_2) \leq 1/\theta$.

Summing the above three upper bounds, we conclude that $\Pr[T] \leq 4/\theta$, as required to verify Condition 1. □

4.2 Proof of the lower bound part of Theorem 1

In this section we prove Theorem 1 using Lemma 9. To do this we extend Lemma 9.

Lemma 10 *(Multiple-Round elimination) Let $\varepsilon \in (0, 1/2)$ and $\theta \geq 8$. Let n, q, r be positive integers. Let $b = (1/\varepsilon)^{3\theta}$. For $i \geq 0$ define $q_i = \mathrm{Tower}_b^{(i)}(q)$, and $n_i = \lfloor \frac{qn}{q_i} \rfloor$. Let j be a nonnegative integer satisfying $j \leq r$ and $n \geq q_j^{2^j}$. Then*

$$\mathsf{err}_n(r, qn) \geq \mathsf{err}_{n_j}(r - j, qn)(1 - \frac{8}{\theta}(1 - 2^{-j})).$$

Proof. We proceed by induction on j; the case $j = 0$ is trivial. So assume $1 \leq j \leq r$ and $n \geq q_j^{2^j}$, and assume by induction:

$$\mathsf{err}_n(r, qn) \geq \mathsf{err}_{n_{j-1}}(r - j + 1, qn)(1 - \frac{8}{\theta}(1 - 2^{-(j-1)})).$$

We bound the left hand side by applying Lemma 9 with n, Q, θ, n' in the lemma replaced by $n_{j-1}, qn, 2^{j-1}\theta, n_j$. The hypothesis of Lemma 9 requires $n_{j-1} \geq (1/\varepsilon)^{4\theta qn/n_{j-1}}$ and this follows easily from $n_{j-1} = \lfloor qn/q_{j-1} \rfloor$ and $n \geq q_j^{2^j}$. Therefore:

$$\mathsf{err}_{n_{j-1}}(r - j + 1, qn) \geq \mathsf{err}_{n_j}(r - j, qn)(1 - \frac{8}{2^j\theta}).$$

Combining the previous two inequalities and observing that $(1 - \frac{8}{\theta}(1 - 2^{-(j-1)}))(1 - \frac{8}{2^j\theta}) \geq 1 - \frac{8}{\theta}(1 - 2^{-j})$ yields the desired result.

□

Corollary 11 *Let $\varepsilon \in (0, 1/2)$ and let $b = (1/\varepsilon)^{48}$. Let q, r, n be positive integers such that $n \geq (\mathrm{Tower}_b^{(r)}(q))^{2^r}$, then*

$$\mathsf{err}_n(r, qn) \geq 1/4.$$

Proof. We apply Lemma 10, with $\theta = 1/16$ and $j = r$ to get:

$$\mathsf{err}_n(r, qn) \geq \mathsf{err}_{n_j}(0, qn)(1 - \frac{1}{2}(1 - 2^{-j})) \geq \frac{\mathsf{err}_{n_j}(0, qn)}{2} = \frac{1}{4}.$$

The last equality holds because $n_j \geq 1$ (by the hypothesis on n), and so $\mathsf{err}_{n_j}(0, qn) = 1/2$ (since a 0-round algorithm has only a $1/2$ chance to guess the correct half). □

Theorem 8 follows readily from Corollary 11. Assume

$$r \leq \log^*_{1/\varepsilon} n - C \tag{4}$$

for some C to be chosen below depending only on ε. Suppose we have a deterministic algorithm that solves WHP_n^U with qn queries and r rounds with error at most $1/4$. We want to show that $q \geq \frac{1}{100} \log^{(r)}_{1/\varepsilon}(n)$ (provided that C is suitably chosen). By Corollary 11, we must have $n < (\mathrm{Tower}_b^{(r)}(q))^{2^r}$, which implies $q \geq \log_b^{(r)}(n^{1/2^r})$. By Proposition 4 (2), with $x = n^{1/2^r}$ and $a = 1/\varepsilon$, we have $q \geq \log_b^{(r)}(n^{1/2^r}) \geq \frac{1}{48} \log^{(r)}_{1/\varepsilon}(n^{1/2^r})$ (Here we need hypothesis of Prop. 4 that $\log^*_{1/\varepsilon}(x) \geq r + T(a, b)$ and this follows from (4) by choosing $C(\varepsilon)$ large enough.

It now suffices to show that (4) implies $\log^{(r)}_{1/\varepsilon}(n^{1/2^r}) \geq \frac{1}{2} \log^{(r)}_{1/\varepsilon}(n)$. If $r = 1$, this is trivial so assume $r \geq 2$. Assumption (4) with $C(\varepsilon)$ suitably chosen guarantees $\log_{1/\varepsilon} n \geq 2^{2r}$ and so $\log^{(r)}_{1/\varepsilon}(n^{1/2^r}) = \log^{(r-2)}_{1/\varepsilon}(\log^{(2)}_{1/\varepsilon}(n) - r \log_{1/\varepsilon}(2)) \geq \log^{(r-2)}_{1/\varepsilon}(\frac{1}{2} \log^{(2)}_{1/\varepsilon}(n)) \geq \frac{1}{2} \log^{(r)}_{1/\varepsilon} n$.

This last inequality follows by using assumption (4) and repeatedly applying $\log_{1/\varepsilon}(x/2) \geq 1/2 \log_{1/\varepsilon} x$ which holds for all $x \geq 4$.

5 Comparison Trees

In this section we prove Theorems 2 and 3.

5.1 Randomized Comparison Trees

Proof. (Sketch for Theorem 2)
Lower bound. As mentioned before, we prove the trade-off lower bound for MAX by reducing OR to it. Suppose we have a PNCT for MAX. Given the input $x = (x_1, \ldots, x_n)$ for OR, we define the rational valued inputs $y_i := x_i + i/2n$; note the y_i are distinct. To compute $\mathsf{OR}(x)$, it is sufficient to determine $\mathsf{MAX}(y)$, since once the index i for which y_i is known, we can recover x_i reliably by querying x_i $O(\log n)$ times.

To find $\mathsf{MAX}(y)$ we simulate the comparison queries of the PNCT for MAX using the boolean queries: To compare y_i and y_j we query x_i and x_j, call the returned values x'_i and x'_j, and return y_i if $x'_i + i/2n > x'_j + j/2n$, else return y_j.

In the above simulation, the probability of error for the comparison query is not a fixed constant and may depend on the values of x_i and x_j. E.g., 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 y_1 is the answer of comparison between x_1 and x_2 is $\varepsilon(1 - \varepsilon)$ corresponding 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 y_1 is the answer of comparison between x_1 and x_2 is ε^2, again corresponding to the event $(x'_1, x'_2) = (1, 0)$.

Since our PNCT for MAX is only guaranteed to work for the case when the probability of error of each comparison is fixed, we need to modify the above simulation so that a comparison query is simulated by a fixed

probability of error by boolean queries. 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 i with probability $q_{x'_i,x'_j}$ and j 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:

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

Equation (u, v) corresponds to the case $(x_i, x_j) = (u, v)$ and expresses the probability that the simulated comparison incorrectly answers j. 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 i is 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 this shows that any $\sqrt{\varepsilon}$-noisy comparison decision tree for MAX 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 2.

Upper bound. Our PNCT for MAX borrows the sampling idea of Reischuk [24], which he used to design a noise-free PCT for MAX 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 and APXM.

TRIVMAX takes as input a subset S of $[n]$, and attempts to output the index $i \in S$ for which y_i is maximum. For some constant C, for each pair of indices in S, it compares the pair $C \log n$ times and declares the "winner" between the two to be the index that wins the majority of comparisons. For C large enough, the index of the maximum of S will be the winner against all other indices with probability arbitrarily close to 1. This PNCT uses one round and $O(|S|^2 \log n)$ queries.

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

$\mathsf{APXM}(S, u)$ is derived from the PNDT for Theorem 5: We create a boolean input $z_1, \ldots, z_n$ for the PNDT by setting $z_i = 1$ if $y_i > x$, and $z_i = 0$ otherwise. Theorem 5 implies that $\mathsf{APXM}(S, x)$ 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.

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

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

3 Output $\mathsf{TRIVMAX}(T_4)$.

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 $\mathsf{TRIVMAX}$ is run only on sets of size $O(N^{1/3})$ and APXM takes at most $O(n)$ comparisons.

Finally, we remark on how the above algorithm can be implemented by a deterministic PNCT in the random noise model. The above PNCT uses randomness for choosing 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, in the style of von Neumann's procedure [31] for generating perfectly random bits from biased random bits. □

5.2 Deterministic Comparison Trees

Proof. (sketch for Theorem 3) We describe a deterministic PNCT computing MAX in $O(n)$ queries and $O(\log \log n)$ rounds in the fault-tolerance and clairvoyant adversary models. Our PNCT is obtained by modifying the PCT of [27] for the same problem in the noise-free model.

The PNCT works in two phases. The first phase has $2 \log \log n$ rounds, and is similar to the NBA tournament type PNCT of Feige et al. [12]. Divide the input variables into groups of size 2. For each group ask 3 times which variable is greater. For each group choose the variable which is greater in the majority of comparisons. These variables participate in the second round, where we again divide these variables into groups of size 2, and for each group ask 5 queries, and so on. In general, in the ith round, each comparison is made $2i + 1$ times. So after $2 \log \log n$ rounds we have $n' = n/(\log n)^2$ variables.

For the second phase, consider a leveled rooted tree T (not to be confused with a decision tree) with n' leaves. In T an internal node with ℓ leaves as descendants has $\sqrt{\ell}$ children. It is easy to see that such a tree has height $\log \log n'$. In the second phase the PNCT assigns the n' variables from the first phase to the n' leaves of T. Now the PNCT proceeds level by level starting from the leaves. Leaves have height 0. We inductively describe round i of the second phase. From the previous rounds the PNCT has assigned a variable to each node in T at height $i - 1$. For each node v at level i we have a group of variables which correspond to the children of v. Now the PNCT will find the maximum of each group and assign it to the node at level i associated with the group. It remains to specify how this maximum is found. This is done in the most simple way: For each pair make $c_1 \log n$

comparisons for some constant c_1, and choose the variable which comes out greater in the majority of comparisons as greater. If it does not get a total order on the elements in a group then it halts with error. Else, if it gets a total order on the elements of the group then choose the maximum variable in this total order. The variable labeling the root is the answer.

Clearly the number of rounds taken by this PNCT is $2\log\log n + \log\log n' < 3\log\log n$. The number of queries in the first phase is $O(n)$, because in round i the number of queries is $(2i+1)n/2^i$. For each round in the second phase, the number of queries is $n'\log n$. The number of rounds in the second phase is $< \log\log n$. So the total number of queries in the second phase is $< n'\log n\log\log n = \frac{n}{(\log n)^2}\log n\log\log n < n$.

By the Chernoff bound the probability that the maximum gets eliminated in the ith round of the first phase is $\approx c_2^i$ for some small positive constant $c_2 < 1$ which can be chosen to be small by choosing ε to be small. Hence the probability of the maximum getting eliminated in the first round is $\leq \sum_{i\geq 1} c_2^i < 2c_2 \ll 1$. In the second phase, by choosing c_1 large enough, again by the Chernoff bound we get that the probability of error for any pair is $\leq 1/n^5$. Since the total number of pairs being compared is $O(n^2)$, with probability $1 - O(1)/n^3$ answer for every pair is correct, and for each group the right total order is found, and the maximum is computed correctly. Our PNCT is robust to adversarial noise because it makes decisions based on majority voting. □

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 trade-off 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 trade-off 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. [2] and Reischuk [24] for deterministic and randomized algorithms respectively.

References

[1] M. Ajtai. The invasiveness of off-line memory checking. *Proc. of ACM Symp. on Theory of Computing* 2002, 504–513.

[2] M. Ajtai, J. Komlos, W .L Steiger, E. Szemeredi. Optimal parallel selection has complexity $O(\log\log n)$. *J. Comput. Syst. Sci.* 38(1): 125–133, (1989).

[3] D. Aharonov, M. Ben-Or. Fault Tolerant Quantum Computation with Constant Error. *Proc. of ACM Symp. on Theory of Computing* 1997, 176–188.

[4] N. Alon, J. Spencer. The Probabilistic Method. Second Edition. *John Wiley & Sons*, 2000.

[5] H. Buhrman, I. Newman, H. Röhrig, R. de Wolf. Robust quantum algorithms and polynomials. *Quantum Physics abstracts*, number quant-ph/0309220.

[6] A. El Gamal. Open problems presented at the 1984 workshop on Specific Problems in Communication and Computation sponsored by Bell Communication Research.

[7] W. Evans, C. Kenyon, Y. Peres, L. J. Schulman. Broadcasting on trees and the Ising model. Annals of Applied Probability, 10(2), pp. 410–433, (2000).

[8] W. S. Evans, N. Pippenger. Average-Case Lower Bounds for Noisy Boolean Decision Trees. *SIAM J. Comput.*, 28(2): 433–446, (1998).

[9] W. S. Evans., L. J. Schulman. Signal propagation and noisy circuits. IEEE Transactions on Information Theory, 44(3), 1299–1305, (1998).

[10] U. Feige. On the Complexity of Finite Random Functions. *Inf. Process. Lett.*, 44(6): 295–296 (1992).

[11] U. Feige, J. Kilian. Finding OR in a noisy broadcast network. *Inf. Process. Lett.* 73(1-2): 69–75 (2000).

[12] U. Feige, P. Raghavan, D. Peleg, E. Upfal. Computing with Noisy Information. *SIAM J. Comput.*, 23(5): 1001–1018 (1994).

[13] P. Gács. Reliable Cellular Automata with Self-Organization. *Proc. 38th IEEE Symp. Foundations of Computer Science* 1997, 90–99.

[14] P. Gács, A. Gál. Lower bounds for the complexity of reliable Boolean circuits with noisy gates. *IEEE Transactions on Information Theory*, Vol. 40, No. 2, 579–583 (1994). roc

[15] R. G. Gallager. Finding parity in simple broadcast networks. *IEEE Transactions on Information Theory*, Vol. 34, 176–180 (1988).

[16] A. Kalai, R. Servedio. Boosting in the Presence of Noise. *P. 35th ACM Symposium on Theory of Computing*, 2003, pp. 196–205.

[17] D. J. Kleitman, F. T. Leighton, Y. Ma. On the Design of Reliable Boolean Circuits That Contain Partially Unreliable Gates. *J. Comput. Syst. Sci.* 55(3), 385–401 (1997).

[18] E. Kushilevitz, Y. Mansour. Computation in Noisy Radio Networks. *Proc. ACM-SIAM Symp. on Discrete Algorithms* 1998, 236–243.

[19] F. T. Leighton, Y. Ma. Tight Bounds on the Size of Fault-Tolerant Merging and Sorting Networks with Destructive Faults. *SIAM J. Comput.*, 29(1): 258–273 (1999).

[20] E. Mossel. Survey: Information flow on trees. *In Graphs, Morphisms and Statistical Physics. DIMACS series in discrete mathematics and theoretical computer science J. Nesetril and P. Winkler editors.* (2004).

[21] I. Newman. Computing in fault tolerance broadcast networks. 19th IEEE Annual Conf. on Computational Complexity, 2004, 113–122.

[22] A. Pelc. Searching games with errors—fifty years of coping with liars. *Theoret. Comput. Sci.*, 270, no. 1-2, 71–109 (2002).

[23] N. Pippenger. On the Networks of Noisy Gates. *Proc. 26h IEEE Symp. on Foundations of Computer Science* 1985, 30–36.

[24] R. Reischuk. Probabilistic Parallel Algorithms for Sorting and Selection. *SIAM J. Computing*, 14(2): 396–409, 1985.

[25] R. Reischuk, B. Schmeltz. Reliable Computation with Noisy Circuits and Decision Trees—A General $n \log n$ Lower Bound. *Proc. 32nd IEEE Symp. on Foundations of Computer Science* 1991, 602–611.

[26] L. Schulman. Coding for Interactive Communication. *IEEE Trans. Information Theory*, 42(6) Part I, 1745–1756, (1996).

[27] Y. Shiloach, U. Vishkin. Finding the maximum, merging and sorting in a parallel computation model. *Journal of Algorithms*, 2(1), 88–102, (1981).

[28] D. A. Spielman. Highly Fault-Tolerant Parallel Computation. *Proc. 37th IEEE Symp. on Foundations of Computer Science* 1996, 154–163.

[29] M. Szegedy, X. Chen. Computing Boolean Functions from Multiple Faulty Copies of Input Bits. *LATIN* 2002, 539–553.

[30] L. G. Valiant. Parallelism in comparison problems. *SIAM J. Comp*, 4(3), 348–355 (1975).

[31] J. von Neumann. Various techniques used in connection with random digits. *Applied Math. Series*, 12, 36–38 (1951).

[32] A. Yao, Probabilistic computations: Towards a unified measure of complexity, *Proc. 17th IEEE Symp. on Foundations of Computer Science*, 1977, 222–227.