Lower Bounds for the Noisy Broadcast Problem

Navin Goyal *
ngoyal@cs.rutgers.edu
Dept. of Computer Science
Rutgers University

Guy Kindler†
gkindler@microsoft.com
Microsoft Research
Redmond

Michael Saks‡
saks@math.rutgers.edu
Dept. of Mathematics
Rutgers University

April 4, 2006

Abstract

We prove the first non-trivial (super linear) lower bound in the *noisy broadcast model*, defined by El Gamal in [6]. In this model there are $n+1$ processors $P_0, P_1, \ldots, P_n$, each of which is initially given a private input bit x_i. The goal is for P_0 to learn the value of $f(x_1, \ldots, x_n)$, for some specified function f, using a series of *noisy broadcasts*. At each step a designated processor broadcasts one bit to all of the other processors, and the bit received by each processor is flipped with fixed probability (independently for each recipient).

In 1988, Gallager [16] gave a noise-resistant protocol that allows P_0 to learn the entire input with constant probability in $O(n \log\log n)$ broadcasts. We prove that Gallager's protocol is optimal, up to a constant factor. Our lower bound follows by reduction from a lower bound for *generalized noisy decision trees*, a new model which may be of independent interest. For this new model we show a lower bound of $\Omega(n \log n)$ on the depth of a tree that learns the entire input.

We also show an $\Omega(n \log\log n)$ lower bound for the number of broadcasts required to compute certain explicit boolean-valued functions, when the correct output must be attained with probability at least $1 - n^{-\alpha}$ for a constant parameter $\alpha > 0$ (this bound applies to all threshold functions, as well as any other boolean-valued function with linear sensitivity). This bound also follows by reduction from a lower bound of $\Omega(n \log n)$ on the depth of generalized noisy decision trees that compute the same functions with the same error. We also show a (non-trivial) $\Omega(n)$ lower bound on the depth of generalized noisy decision trees that compute such functions with small constant error.

Finally, we show the first protocol in the noisy broadcast model that computes the Hamming weight of the input using a linear number of broadcasts.

*Supported in part by NSF grant CCR-9988526 and a Bevier fellowship of Rutgers University
†Supported by CCR grant $\mathcal{N}$CCR-0324906 and $\mathcal{N}$DMS-0111298.
‡Supported in part by NSF grants CCR-9988526 and CCR-0515201.

1 Introduction

The relationships between noise, communication and computation have been studied extensively. A recurring problem, arising both in theory and practice, is to minimize the additional resources needed to obtain reliable results in the presence of noise. This problem was studied in the context of decision trees [12, 27, 10, 8, 23], formulas and circuits [25, 15, 30, 18, 9], sorting networks [20], cellular automata [14], quantum computation [2], data structures [13, 4], various communication models [16, 29, 19, 28, 23] and other models [1, 7, 24, 17].

The *noisy broadcast model* was proposed by El Gamal [6] in 1984, and later popularized by Yao [35], as a simple model in which to study the effect of noise in a highly distributed system. This model considers n processors, $P_1, \ldots, P_n$, and a receiver P_0. Each processor P_i has a private input bit x_i, and the goal is for P_0 to evaluate a specified function $f(x_1, \ldots, x_n)$ using the smallest possible number of broadcasts. Communication is carried out in synchronous time steps: in each step a prespecified processor broadcasts a single bit to all other processors. For some fixed noise parameter $\varepsilon < 1/2$, each of the other processors independently receives the broadcast bit (with probability $1-\varepsilon$) or the complement of the bit (with probability ε). After the final broadcast, P_0 determines the output of the protocol from the bits it has heard. If for every given input $x \in \{0,1\}^n$ the protocol outputs $f(x)$ with probability at least $1-\delta$, we say that it computes f with error at most δ for noise parameter ε.

In this paper we study the case where the receiver, P_0, aims to output the entire input. Thus the function to be computed is the identity function, denoted *id*, which is clearly the hardest function to compute in this model. There is a simple protocol that computes *id* with error δ, for any fixed $\delta \in (0, 1/2)$, using $\Theta(n \log n)$ broadcasts: for each $i \in \{1, \ldots, n\}$, P_i broadcasts its bit $c \log n$ times (for some large enough constant $c = c(\varepsilon, \delta)$) and P_0 outputs the majority value of the copies of x_i it received. In 1988, Gallager [16] gave a protocol for *id* using only $O(n \log \log n)$ broadcasts, and this remains the best upper bound known. Previous to this paper, the only known lower bound was the trivial $\Omega(n)$. In this paper we prove an $\Omega(n \log \log n)$ lower bound, thereby showing that Gallager's upper bound is optimal up to a constant factor.

Related work

Kushilevitz and Mansour [19] showed that the majority function (or any other threshold function) can be computed using $O(n)$ broadcasts. This protocol takes advantage of a rather strong and unrealistic feature of the model: that the noise occurring in different receptions is independent and identically distributed. To consider protocols that are less dependent on the exact distribution of noise, Feige and Kilian [11] proposed a stronger adversarial model of noise. Roughly speaking, this model allows an omniscient adversary to cancel any of the errors introduced by the random noise, thus preventing the algorithm from taking advantage of stochastic regularities in the noise. Feige and Kilian showed that even against this stronger adversary OR can be computed in $O(n \log^* n)$ broadcasts. Newman [23] improved this to $O(n)$ broadcasts (Newman stated his result for a weaker adversarial model, but his result easily carries over to the stronger adversarial model of [11].)

Some results in closely related models are also worth mentioning: efficient error resilient protocols were given for the noisy two party communication complexity by Schulman [29], and for noisy communication networks with small degree by Rajagopalan and Schulman [26].

1.1 Our results

In this paper we prove that Gallager's $O(n \log\log n)$ protocol for *id* is optimal:

Theorem 1. *Let $\varepsilon \in (0, 1/2)$ be any noise parameter, and let $\beta \geq 1$ and n be a positive integer. Let A be a (possibly randomized) noisy broadcast protocol for n processors using at most βn broadcasts. If A is executed with noise parameter ε on a random input X distributed uniformly in $\{0, 1\}^n$, then the probability that the receiver outputs X is at most*

$$\sqrt{\frac{1}{n}} + \frac{48\beta^2 \log(1/\varepsilon)}{\varepsilon^{4\beta} \log n}.$$

This immediately gives,

Corollary 2. *Let $\varepsilon \in (0, 1/2)$ be any constant. Then any noisy broadcast protocol which computes the identity function with error at most $1/3$ for noise parameter ε, requires $\Omega(n \log\log n)$ broadcasts.*

Our lower bound is formulated for the original noise model of El Gamal; trivially, it also holds for the adversarial noise models as well. Previously, no superlinear lower bounds were known even for the adversarial noise model.

Generalized noisy decision trees. Our lower bound for the noisy broadcast model is obtained by reducing that model to a new model, which we call the *generalized noisy decision tree* (gnd*-tree) model.* This model, which may be of independent interest, considers a more centralized computational setting where a single decision tree attempts to evaluate a function of the boolean input x. The gnd-tree does not have direct access to x—instead it has access to an unlimited number of *noisy copies* of x. In each noisy copy of x each bit is complemented independently with some fixed probability $\varepsilon < 1/2$. At each step the tree selects one of the copies of x, and asks for the evaluation of any boolean-valued function at that copy. (A precise definition of the model appears in Section 2.)

The gnd-tree model can be seen as a generalization of the noisy decision tree (nd-tree) model introduced by Feige et al. [12]. The nd-tree model is obtained by restricting the gnd-tree model to *coordinate queries*, namely queries q of the form $q(y_1, \ldots, y_n) = y_i$, $i \in \{1, \ldots, n\}$. The computational power of these two models is not the same: the majority function requires $\Omega(n \log n)$ queries by an nd-tree [12], but can be computed with just $O(n)$ queries using a gnd-tree, by adapting the aforementioned noisy broadcast protocol for majority due to Kushilevitz and Mansour [19].

The proof of Theorem 1 has two parts. The first part is a reduction that takes a protocol in the noisy broadcast model and simulates it by a gnd-tree. The second (and more substantial) part is a lower-bound on the depth of gnd-trees required to compute the identity function. It seems that proving lower bounds for gnd-trees is easier than proving them directly for the noisy broadcast model, because the former model is more centralized, making it easier to analyze the progress made in intermediate steps of the computation.

Note that *id* can be easily computed by a gnd-tree (indeed, even by an nd-tree) of depth $O(n \log n)$: the tree just needs to query the value of all coordinates of $O(\log n)$ noisy copies of the input, and output the bitwise majority of the copies. Our main lower bound for gnd-trees implies that this protocol is within a constant factor of optimal.

Theorem 3. *Let $\varepsilon \in (0, 1/2)$ be a noise parameter and let n be a positive integer. If a (possibly randomized)* gnd*-tree T of depth d is run on an input X chosen uniformly from $\{0,1\}^n$, with noise parameter ε, then the probability that T outputs X is bounded above by*

$$\frac{1}{\sqrt{n}} + \frac{6\log(1/\varepsilon)}{\varepsilon^2 \log n} \cdot \left(\frac{d}{n} + \sqrt{\frac{d}{n}}\right).$$

In particular, when $d \geq n$, the probability that T outputs X is bounded above by

$$\frac{1}{\sqrt{n}} + \frac{12\log(1/\varepsilon)}{\varepsilon^2 \log n} \cdot \frac{d}{n}.$$

Decision functions A significant problem left open by Theorem 1 is whether there are *decision functions* (functions that output a boolean value) that require a superlinear number of broadcasts in the noisy broadcast model. While it seems intuitively clear that there should be such functions, proving this seems difficult. Counting arguments do not seem useful here because the number of possible protocols grows too quickly as a function of the number of broadcasts. On the other hand the linear upper bound on threshold functions of Kushilevitz and Mansour [19], as well as our linear protocol for computing Hamming weight mentioned below, indicate that it may be difficult to prove a superlinear lower bound for an explicit function.

Proving lower bounds for decision functions in the gnd-tree model is even more problematic than in the noisy broadcast model. It is trivial that in the noisy broadcast model every function that depends on all of its variables needs at least n broadcasts in worst case (since the choice of who broadcasts is fixed in advance). For gnd-trees, however, we cannot rule out the existence of trees of sublinear depth even for functions whose (noiseless) decision-tree complexity is linear (also, there are highly non-trivial functions whose decision-tree complexity is sublinear, and can also be computed in sublinear depth by gnd-trees). We don't know of any simple arguments that yield linear lower bounds for gnd-trees computing decision functions, or even bounds of the form $n^{1-\alpha}$ for small positive $\alpha > 0$.

Our proof of Theorem 3 is based on the idea that it is hard for a gnd-tree or a noisy broadcast protocol to distinguish an input from its immediate neighbors. This suggests the parity function as a candidate for proving superlinear lower bounds. It turns out, however, that any boolean function output depends only on the Hamming weight $\sum_i x_i$ of the input can be computed in the noisy broadcast model with a linear number of broadcasts:

Theorem 4. *For any $\varepsilon, \delta \in (0, 1/2)$ there is a noisy broadcast protocol that computes the Hamming weight of the input with probability of error at most δ for noise parameter ε, which uses cn broadcasts for some constant $c = c(\varepsilon, \delta)$. Therefore every symmetric boolean can be computed using a linear number of broadcasts.*

Similarly, there exists a gnd*-tree that computes $\sum_i x_i$ with error at most δ for noise parameter ε and has depth dn for some constant $d = d(\varepsilon, \delta)$.*

Our protocol for computing the Hamming weight of the input borrows some of the ideas from the majority protocol of [19]. While it is somewhat surprising that such a protocol exists, it has the same drawback as the majority protocol: it relies heavily on the unrealistic assumption of the model that every bit received experiences independent noise precisely ϵ.

Lower bounds for decision functions. We are able to get some non-trivial lower bounds for computing decision functions, both in the noisy broadcast model and for gnd-trees, by adapting the techniques from the proof of Theorem 3. Our bounds for computing a boolean function f depend on its *sensitivity*, s_f. The sensitivity of a function f is the maximum, over all $x \in \{0,1\}^n$, of the number of neighboring inputs y (inputs y that differ from x on one coordinate) for which $f(y) \neq f(x)$. In particular, $s(f) = n$ for AND,OR, and PAR, and is at least $n/2$ for any symmetric boolean function. Our results, which are more accurately stated and proved in section 5, are summarized below:

Theorem 5. *Let $\varepsilon, \delta \in (0, 1/2)$ and $\alpha > 0$ be fixed constants. Let $f : \{0,1\}^n \to \{0,1\}$ be any decision function, and let $s = s(f)$ denote the sensitivity of f. Then*

- *The depth of any* gnd*-tree that computes f with error at most δ for noise parameter ε, is at least $\Omega(s)$.*
- *The depth of any* gnd*-tree that computes f with error at most $n^{-\alpha}$ for noise parameter ε, is at least $\Omega(s \log(s))$.*
- *Any noisy broadcast protocol that computes f with error at most $n^{-\alpha}$ for noise parameter ε, requires $\Omega(s \log\log(s))$ broadcasts.*

1.2 Organization

In Section 2 we formally define the noisy broadcast and gnd-tree models. We then state a reduction theorem between noisy broadcast protocols and gnd-trees, and use it to show that Theorem 3 implies Theorem 1. Section 3 contains some technical preliminaries for the proofs of the main results. The proof of Theorem 3 is given in Section 4. Section 5 contains the precise statements and proofs for our lower bounds on decison functions. Section 6 proves the reduction theorem, showing how to simulate noisy broadcast protocols by gnd-trees. Section 7 we give our variant of Gallagher's $O(n \log\log n)$ noisy broadcast protocol for *id* and in section 8 we give our linear noisy broadcast protocol for computing the Hamming weight of the input. We conclude the paper with some open problems in section 9.

2 The noisy broadcast model and gnd-trees.

In this section we begin by defining the noisy broadcast and generalized noisy decision tree models. Next, we state a general reduction lemma from the first model to the second and use this lemma to derive Theorem 1 from Theorem 3.

2.1 Noisy copies

Let $\varepsilon \in (0, 1/2)$ be a noise parameter. An *ε-noise bit* is a $\{0,1\}$-valued random variable that takes value 1 with probability ε. An *ε-noise k-vector* N is a sequence of k independent ε-noise bits.

For a bit-vector $x \in \{0,1\}^k$, an *ε-noisy copy of x* is a random variable of the form $x \oplus N$, where $\oplus$ denotes the bitwise XOR, and N is an ε-noise k-vector. More generally, if X is any random variable taking values in $\{0,1\}^k$ an *ε-noisy copy of X* is a random variable of the form $X \oplus N$, where N is an ε-noise k-vector chosen independently from X.

2.2 Computation under noise

In standard computation models, a deterministic computation is determined by its input, and a randomized computation is determined by the input and some auxiliary independent unbiased random bits.

We will discuss several models for computing in the presence of noise. In such models, the computation also depends (in some specified way) on a boolean vector N that represents the noise that affects the computation. It is assumed that N is a ε-noise vector for some $\varepsilon \in (0, 1/2)$, and that N is independent of any random bits used by the algorithm.

For $\delta \in [0, 1]$ we say that an algorithm computes a function f with error at most δ against ε-noise if for each input x, the algorithm outputs $f(x)$ with probability at least $1-\delta$, where the probability is taken with respect to the auxiliary random bits of the algorithm and the ε-noise vector N.

2.3 The Noisy Broadcast Model

The *noisy broadcast model* considers one *receiver* P_0 and n *processors* $P_1, \ldots, P_n$. The input is a boolean vector x of length n, and each of the processors P_i initially has coordinate x_i. The goal is for P_0 to evaluate a specified function f at x. This goal is to be accomplished by a *noisy broadcast protocol*.

The specification of a noisy broadcast protocol consists of:

- The number s of broadcasts used in the protocol.
- A sequence $i^1, \ldots, i^s$ of indices of processors (with repetitions allowed).
- A sequence $g^1, \ldots, g^s$ of *broadcast functions*, where $g^j : \{0,1\}^j \longrightarrow \{0,1\}$.
- An *output function* h which is defined over the domain $\{0,1\}^s$.

Running a protocol. The execution of a noisy broadcast protocol A depends on the input x, and on a noise vector N. We will think of N as a concatenation of s independent noise vectors $N^1, \ldots, N^s$, each of length $n+1$. In the j'th step of the execution of A the processor P_{i^j} broadcasts a bit b^j and each of the other processors receives an independent noisy copy of b^j. Formally, P_h receives $b^j_h = b^j \oplus N^j_h$; it will be convenient (but unimportant) to regard P_{i^j} as receiving a noisy copy $b^j_j = b^j \oplus N^j_j$ of his own message. The bit b^j broadcast by P_{i^j} at step j is g^j evaluated at the j-vector consisting of P_{i^j}'s input bit and the $j-1$ bits received by P_{i^j} during the first $j-1$ rounds. Thus $b^j = g^j(x_{i^j}, b^1_{i^j}, b^2_{i^j}, \ldots, b^{j-1}_{i^j})$. The output of the protocol is $h(b^0_1, \ldots, b^0_s)$, that is, the value of h on the s-vector of bits received by P_0.

The randomized version of the model is defined in the natural way. each processor has access to a source that generates independent random bits where the processor has the ability to specify the bias of each successive bit. The function g^j determining the bit broadcast by P_{i_j} may depend on the random bits generated by P_{i_j}.

Our version of the noisy broadcast model is similar to that of Gallager [16]. Other minor variants of this model have been proposed, e.g., there is no receiver P_0, and the goal of the computation is for all of the processors to learn the correct value of $f(x)$. These differences are not significant, as protocols in one model can easily and efficiently be simulated in another.

This model enforces certain properties typically required of communication protocols in noisy environments. First, protocols must be *oblivious*: in the sense that the sequence of processors who broadcast is fixed in advance and does not depend on the execution. Without this requirement, noise could lead to several processors speaking at the same time. Second, it rules out *communication by silence*: when it is the turn of a processor to speak, it must speak.

2.4 Generalized Noisy Decision Tree Model

The generalized noisy decision tree (gnd-tree) model is a centralized computation model in which an algorithm seeks to evaluate f on input $x \in \{0,1\}^n$ by asking queries. The algorithm has no direct access to the input x. Instead, there is a collection $(y^\lambda : \lambda \in \Lambda)$ of independent ε-noisy copies of x; here Λ is an arbitrary index set. In each step, the algorithm is allowed to make an *arbitrary boolean-valued query* about any one of the noisy copies.

Formally, a generalized noisy decision tree algorithm (gnd-tree) is represented by a rooted labeled binary tree. Each internal node v is assigned a *copy type* $\lambda_v \in \Lambda$ and a *query function* $q_v : \{0,1\}^n \longrightarrow \{0,1\}$. The two arcs out of internal node v are labeled by 0 and 1. Each leaf v is labeled by an output value out_v.

The noisy copies $y^1, y^2, \ldots$ of x determine a unique root-to-leaf path, called the *execution path* as follows: start from the root r and follow the arc labeled by the output of q_r evaluated at noisy copy y^{λ_r} to a new node. Upon arriving at internal node v, evaluate $q_v(y^{i_v})$ and follow the indicated arc. The output of the computation is equal to the output value out_v labeling the leaf.

We also consider randomized gnd-trees. For our purposes, a randomized gnd-tree is simply a probability distribution over gnd-trees.

2.5 Proof of Theorem 1 via reduction

Let us now state a reduction theorem between the noisy broadcast model and the gnd-tree model, and show how it can be used to deduce Theorem 1 from Theorem 3.

To state the reduction theorem we need some notation. For $K \subseteq [n]$, we refer to a point in $\{0,1\}^K$ as a *partial assignment to* K. If ρ is a partial assignment to K and x is a partial assignment to $[n]-K$ we write $x\rho$ for the point in $\{0,1\}^n$ that agrees with ρ on K and with x on $[n]-K$.

Theorem 6. *Let $\alpha > 1$, and $\varepsilon \in (0,1/2)$. Suppose $\mathcal{P}$ is a noisy broadcast protocol that uses k broadcasts. Then there is a subset $K \subseteq [n]$ of size at most n/α such that for any partial assignment ρ to K there is a* gnd*-tree T of depth $2k$ with input variables indexed by $[n]-K$ with the following property: for any $x \in \{0,1\}^{[n]-K}$, when T is run on x with noise parameter $\varepsilon^{\alpha k/n}$, the output distribution is exactly the same as that of the output of $\mathcal{P}$ when run on $x\rho$ with noise parameter ε.*

The proof of this theorem appears in Section 6. Now we can deduce Theorem 1 from Theorem 3.

Proof of Theorem 1 from Theorem 3. Suppose that for some $\beta \geq 1$, A is a βn-step noisy broadcast protocol that computes id_n against ε-noise with some probability of correctness . Choosing $\alpha = 2$ in Theorem 6, we have a gnd-tree of depth $2k$ that computes $id_{\lceil n/2 \rceil}$ against

$\varepsilon^{2\beta}$-noise with the same probability of correctness as A. By Theorem 3, the probability that this gnd-treeis correct is bounded above by:

$$\frac{1}{\sqrt{n}} + \frac{12\log(1/\varepsilon^{2\beta})}{\varepsilon^{4\beta}\log n}(2\beta) \leq \\ \frac{1}{\sqrt{n}} + \frac{48\beta^2\log(1/\varepsilon)}{\varepsilon^{4\beta}\log n}.$$

This completes the proof of Theorem 1.

□

3 Preliminaries to the proof of Theorem 3

In this section we present some preliminary notation, conventions, and technical facts.

Some notation. We use log to denote logarithm in base 2. When n is implicit from the context, we use $\mathbf{e_i}$ to denote the point in $\{0,1\}^n$ whose i'th cooridnate is 1 and whose other coordinates are 0. The point with all zero coordinates is denoted $\mathbf{0}$.

3.1 Entropy and relative entropy.

We use some basic notions from information theory. In that context, we consider terms of the form $0\log\frac{1}{0}$ or $0\log\frac{0}{0}$ to have value 0. For a random variable X taking values in a finite set A, the *binary entropy* of X is given by:

$$\mathbb{H}\left[X\right] := \sum_{x\in A}\Pr[X=x]\log\frac{1}{\Pr[X=x]}. \tag{1}$$

If another random variable Y, taking values in a finite set B, is defined over the same probability space as X, the *conditional entropy* of X given Y is defined by

$$\mathbb{H}\left[X|Y\right] := \sum_{y\in B}\Pr[Y=y]\mathbb{H}\left[X|Y=y\right]. \tag{2}$$

For two probability measures p and q defined over a finite set A, the *relative entropy* between p and q, denoted $\mathsf{D}(p\|q)$ is defined by

$$\mathsf{D}(p\|q) := \sum_{x\in A}p(x)\log\frac{p(x)}{q(x)}$$

(if $q(x)=0$ for an x where $p(x)\neq 0$ the relative entropy is infinite). Entropy and relative entropy satisfy the following inequalities (see, e.g., [5]):

Fact 7. *For X, p, q, and A as above, we have:*

$$\mathsf{D}(p\|q) \geq 0.$$

$$\mathbb{H}\left[X\right] \leq \log|A|.$$

3.2 Some estimates for logarithms

The Taylor series

$$\ln(1+x) = \sum_{n\geq 1} \frac{(-1)^{n+1}x^n}{n},$$

which is valid for $x \in (-1,1)$, implies that the following functions are continuous and differentiable for all $x > -1$.

$$a(x) = \begin{cases} \frac{\ln(1+x)}{x} & \text{if } x \neq 0, \\ 1 & \text{if } x = 0 \end{cases} \tag{3}$$

$$b(x) = \begin{cases} \frac{\ln(1+x)-x}{x^2} & \text{if } x \neq 0 \\ -\frac{1}{2} & \text{if } x = 0 \end{cases} \tag{4}$$

Proposition 8. *The function a is positive and decreasing on $(-1,\infty)$, and the function b is negative and increasing on $(-1,\infty)$.*

Proof. For $x \in (-1,0) \cup (0,\infty)$, $a'(x) = \frac{g(x)}{x^2}$ where $g(x) = \frac{x}{1+x} - \ln(1+x)$. To prove the statement concerning the function a it thus suffices to show that $g(x) \leq 0$ for $x > -1$. This follows from the fact that $g(0) = 0$, and that $g'(x) = \frac{-x}{(1+x)^2}$ which is positive for $x < 0$ and negative for $x > 0$.

Next we prove the statement about the function b. Since $x > \ln(1+x)$ for all $x \in (-1,0) \cup (0,\infty)$ it follows that $b(x) < 0$ for all $x \in (-1,\infty)$. To show that b is increasing, we show that b' is nonnegative. Using the Taylor expansion of $\ln(1+x)$ around 0, we have that for $x \in (-1,1)$,

$$b'(x) = \sum_{n=0}^{\infty} (-1)^n \frac{n+1}{n+3} x^n.$$

Thus $b'(0) = \frac{1}{3}$ and for $x < 0$, each term in the second sum is positive so $b'(x) > 0$. For $x > 0$, it suffices to show that the numerator of $b'(x) = (x^2 + 2x - 2(1+x)\ln(1+x))/x^3(1+x)$ is positive. The numerator is 0 at $x = 0$ and has derivative $2(x - \ln(1+x))$ which is positive for all $x > 0$. $\square$

Corollary 9. *Let $\varepsilon \in (0, \frac{1}{2})$ and $x \geq \varepsilon - 1$. Then:*

1. $0 < a(x) \leq 2\ln(1/\varepsilon)$
2. $0 > b(x) \geq -2\ln(1/\varepsilon)$.

Proof. Using Proposition 8 and $\varepsilon \leq 1/2$ we have

$$0 < a(x) \leq a(\varepsilon - 1) = \ln(1/\varepsilon)/(1-\varepsilon) \leq 2\ln(1/\varepsilon),$$

and thus the first part holds.

As for the second part, $0 > b(x)$ follows from Proposition 8. Also, since b is negative and increasing it remains to show that $|b(\varepsilon-1)| \leq 2\ln(1/\varepsilon)$. We consider two cases: if $\varepsilon \leq 1-1/\sqrt{2}$ then it is easy to see that

$$|b(\varepsilon-1)| \leq \frac{\ln(1/\varepsilon)}{(1-\varepsilon)^2} \leq 2\ln(1/\varepsilon).$$

If $\varepsilon > 1-1/\sqrt{2}$, since a is decreasing we have $a(\varepsilon-1) \leq a(-1/\sqrt{2})$ which by direct computation is less than 2. From this and $\varepsilon \leq 1/2$, we get

$$|b(\varepsilon-1)| = \frac{a(\varepsilon-1)-1}{1-\varepsilon} = \frac{2a(\varepsilon-1)-2}{2(1-\varepsilon)} \leq \frac{a(\varepsilon-1)}{2(1-\varepsilon)} = \frac{\ln(1/\varepsilon)}{2(1-\varepsilon)^2} \leq 2\ln(1/\varepsilon).$$

□

Proposition 10. *Let m be defined on $[0,1]$ by:*

$$m(x) \quad = \quad \begin{cases} x^2\ln(1/x) & \text{if } x \neq 0, \\ 0 & \text{if } x = 0 \end{cases}$$

Then $m(x) \geq m(1-x)$ for all $x \in [\frac{1}{2}, 1]$.

Proof. Let $d(x) = m(x)-m(1-x)$ for $x \in [1/2,1]$. Then d is continuous and twice differentiable on $[1/2,1]$ with second derivative given by $d''(x) = 2(\ln((1-x)/x)$. Since this is negative for all $x \in (1/2,1)$, and $m(1/2) = m(1) = 0$, we conclude that $m(x) \geq 0$ for $x \in [1/2,1]$. □

Proposition 11. *For $x, y \in [0,1]$, $x\log(x/y) + (1-x)\log((1-x)/(1-y))$ is nonnegative.*

This is a well known fact; we give a proof for completeness.

Proof. Hold x fixed; it suffices to show that as a function of y, $g(y) = x\log y + (1-x)\log(1-y)$ has a maximum at $y = x$. Taking the derivative with respect to y and simplifying yields $g'(y) = (x-y)/y(1-y)$ which is positive for $0 < y < x$ and negative for $1 > y > x$ implying g attains a maximum at $y = x$. □

3.3 Tail bounds for sums of noise bits

We will need a standard type of tail bound for sums of noise bits.

Lemma 12. *Let N be a ε-noisy n-vector. For any nonzero vector $\alpha \in \mathbb{R}^n$, and $t > 0$ we have:*

$$\Pr_N[\sum_i \alpha_i(N_i - \varepsilon) > t] \leq e^{-2t^2/\|\alpha\|^2}.$$

Since we don't know a reference for this version of the bound, we provide a proof.

Proof. For any positive λ,

$$\begin{aligned}\Pr_N[\sum_i \alpha_i(N_i-\epsilon) > t] &\leq e^{-\lambda t}\mathbb{E}[e^{\lambda\sum_i \alpha_i(N_i-\varepsilon)}] \\ &= e^{-\lambda t}\prod_i \mathbb{E}[e^{\lambda\alpha_i(N_i-\varepsilon)}] = e^{-\lambda t}\prod_i \varepsilon\left(e^{\lambda\alpha_i(1-\varepsilon)} + (1-\varepsilon)e^{-\lambda\alpha_i\varepsilon}\right) \\ &\leq e^{-\lambda t}\prod_i e^{\alpha_i^2\lambda^2/8}\end{aligned}$$

(since for $p \in [0,1]$, $pe^{\gamma(1-p)} + (1-p)e^{-\gamma p} \le e^{\gamma^2/8}$ (see Lemma A.1.6 of [3].))

$$= e^{-\lambda t + \lambda^2 \|\alpha\|^2/8}.$$

Choosing $\lambda = 4t/\|\alpha\|^2$ yields the bound of the lemma. $\square$

4 Proof of Theorem 3

In this section we prove Theorem 3. For the rest of this section, fix T to be a gnd-tree that supposedly computes *id* for inputs $x \in \{0,1\}^n$. The *success probability* of T is the probability that T gives the correct output when run on a uniformly random input. Our goal is to give an upper bound on the success probability of T in terms of its depth $\mathsf{depth}(T)$, n, and the noise parameter ε. In the case that T is randomized, i.e., a probability distribution over deterministic gnd-trees, its success probability is the average (with respect to this distribution) of the success probabilities of various deterministic trees. Thus, any upper bound for deterministic trees extends to randomized trees.

Therefore, without loss of generality, we assume that T is deterministic.

The vertices of T. Let V be the set of vertices of T. Let Λ be the set of indices of the noisy copies of the input used by T. For an index $\lambda \in \Lambda$, V^λ will denote the set of (internal) vertices of T that query the noisy copy indexed by λ.

The execution space Υ. For noise parameter $\varepsilon \in (0, 1/2)$ we define the *ε-execution space* $\Upsilon = \Upsilon(T, \varepsilon)$ of T to be the probability space corresponding to the choice of a uniformly random input $X \in \{0,1\}^n$ and an indexed family of independent ε-noise n-vectors $N^\Lambda = (N^\lambda : \lambda \in \Lambda)$. For every $\lambda \in \Lambda$, Y^λ denotes the ε-noisy copy of X defined by $Y^\lambda = X \oplus N^\lambda$. $Y^\Lambda = (Y^\lambda : \lambda \in \Lambda)$ denotes the indexed family of all noisy copies.

We say that T is *executed on* Υ if it is run with input X, and with access Y^Λ as the noisy copies of the input. Note that when T is executed on Υ, the unique root-to-leaf execution path is a random variable that is completely determined by Y^Λ. We define the following random variables and events over the execution space:

- Let Π denote the leaf that terminates the execution path.
- Let $\mathsf{success}$ be the event that T correctly outputs the input X.
- For a vertex v, let $\mathsf{vis}(v)$ denote the event that v lies on the execution path. In this case we say that T *visits* v.
- When the meaning is clear from context, the event $X = x$ (for $x \in \{0,1\}^n$) is abbreviated as x; especially when writing the probabality of events. For example, we may write $\Pr[x]$ instead of $\Pr[X = x]$ and $\Pr[\mathsf{success}|x]$, instead of $\Pr[\mathsf{success}|X = x]$.

The progress function. As with many lower bound proofs, our proof proceeds by defining a function that measures the progress of the computation towards its goal. We show that (1) for T to succeed with high probability, the expected value of the progress measure at the final leaf must be large; and (2) the aforementioned expected value is bounded by a function of $\mathsf{depth}(T)$, and thus for it to be large the depth of T must be large as well.

For a vertex v in T and an input vector $x \in \{0,1\}^n$ we define the following functions. The final function L serves as our progress measure.

$$\begin{aligned} p(v,x) &= \Pr\left[\mathsf{vis}(v) | X = x\right], \\ L_i(v,x) &= \log\left(\frac{p(v,x)}{p(v,x \oplus \mathbf{e_i})}\right) \qquad \text{for } i \in \{1,\ldots,n\}, \\ \text{and} \quad L(v,x) &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} L_i(v,x). \end{aligned} \tag{5}$$

Some intuition. Theorem 3 gives an upper bound on the success probability of T in terms of its depth, or equivalently, on the number of queries it makes. Let us consider the most interesting case, where the success probability is at least some constant, say $1/2$. In this case the theorem gives a lower bound of $\Omega(n \log n)$ on the number of queries.

To prove this lower bound using a progress measure, we want to show that in order to succeed with high probability, the expectation of the progress measure achieved by T at the completion of its computation must be higher than some threshold τ. We then want to show (roughly) that the progress measure achieved by any gnd-tree making $o(n \log n)$ queries is smaller than τ.

So how does one measure the progress made by T after the computation has reached a vertex v? One natural choice would be to look at $\mathbb{H}[X] - \mathbb{H}[X|\mathsf{vis}(v)]$, the reduction in entropy of the random input X provided by knowing $\mathsf{vis}(v)$. Initially, at the root, this is equal to 0, and it is not hard to show that for any gnd-tree that outputs X correctly with probability $1/2$ the expectation of this expression must be at least $\Omega(n)$. To prove our result, we'd like to show that $\Omega(n \log n)$ queries are needed to reduce the entropy of X by $\Omega(n)$, but this is not true: if we query all n coordinates of a single noisy copy, then the expected reduction of entropy in X is already $\Omega(n)$.

The reason for this is that querying each coordinate once already gives us, w.h.p., a guess for X which is correct on roughly $(1-\varepsilon)$-fraction of coordinates. To go from that to getting all the coordinates right takes many more queries. Indeed, it turns out that in a sense, distinguishing between X and its neighbors (namely from inputs that differ from it on a singly coordinate) is as hard as figuring out all the bits of X from scratch. But although after all coordinates have been queried once, making further queries will provide less of an entropy reduction, this behavior is difficult to capture and use, since to show that the entropy reduction is slow one must use the fact that certain information about X is already known. Furthermore, other algorithms which perform different queries may have a different pattern of entropy reduction.

The progress measure L turns out to be more useful. For example, it behaves very nicely when we restrict to coordinate queries of noisy copies of X. Indeed, the change in L due to the response to a coordinate query in a noisy copy of X is context independent, i.e., does not depend on the answers to any of the previous queries. (The proof of this is easy, but since we do not need it explicitly, it is omitted.) A single coordinate query of bit i changes in L_i by $\Theta(1)$ while L_j is unchanged, so L changes by at most $\emptyset(1/n)$. It is easy to show that the expectation of the final value of L must have expectation at least $\Omega(\log n)$, and thus using our progress measure it is relatively easy to get an $\Omega(\log n)$ lower-bound on the depth of a tree computing *id* and only making coordinate queries. In the case of more general queries we do not have context independence in progress measure gain, but its behavior is still nice enough for us to control its increase as a function of the depth of T.

By rewriting the expression for L we can get further intuition. Since x is uniformly distributed we have $\Pr[x] = \Pr[x \oplus \mathbf{e_i}]$, and thus by standard laws of conditional probability the functions L_i can be written as

$$L_i(v, x) = \log\bigl(\Pr[x|\mathsf{vis}(v)]\bigr) - \log\bigl(\Pr[x \oplus \mathbf{e_i}|\mathsf{vis}(v)]\bigr). \tag{6}$$

The term $\Pr[y|\mathsf{vis}(v)]$ measures the conditional probability of the event $[X = y]$, given that T was run on the random input X and has reached the node v in the course of computation. It is therefore the "perceived probability" that T assigns to y being the value of the input, when it reaches v during the computation. Thus $L_i(v, x)$ compares the perceived probabilities of x and its neighbor $x \oplus \mathbf{e_i}$ and thus measures how well the computation has distinguished x from $x \oplus \mathbf{e_i}$, given that it has arrived at v. $L(v, x)$ gives an aggregate measure of how well the computation has distinguished x from all of its neighbors. Note that while it is also important that the computation distinguish x from points other than its neighbors, L does not take this into account directly. Intuitively, the neighbors are the "hardest" points to distinguish x from, so it is enough for L to only consider these.

Let us now get a rough estimate for the change in L at the completion of a successful algorithm when the true input is x. At the beginning of the computation, when v is the root, $L(v, x)$ is obviously zero. At the end of the computation, when T reaches a leaf π, we expect that T should assign a high perceived likelihood to x, say $\Omega(1)$. On the other hand, since the sum of the perceived likelihoods of all points is 1, the average perceived likelihoods of the neighbors of x is at most $\frac{1}{n}$ and the concavity of the logarithm yields that $L(\pi, x) \geq \log n - O(1)$.

A coarse intuition for the proof of Theorem 3 is that we bound the expected gain to L obtained from each query by $O(\frac{1}{n})$. Since the typical value of $L(\Pi, X)$ is roughly $\log n$ (this is the value of L at the leaf reached by the computation and the actual input on which T is run), it follows that at least $\Omega(n \log n)$ queries are typically required.

As noted above, if we restrict queries to noisy copies of individual input bits, it is easy to show that each query changes L by at most $O(\frac{1}{n})$. When we turn to general queries this is no longer true, indeed one can construct situations where a single general query changes L by much more. Nevertheless, we are able to formulate and prove a weaker statement (Lemma 14) that suffices for our bounds.

The relation to relative entropy. We note that the functions L_i and L are related to relative entropies. For example, looking at (6) one observes that $\mathbb{E}\Big[L_i(\Pi, X) \bigm| x\Big]$ is the relative entropy between the distribution of the leaf reached when T is run on x, and that of the leaf reached when T is run on the i'th neighbor of x.

How we proceed. The next two lemmas formalize the above intuition, and imply Theorem 3 immediately. Lemma 13 states that for T to succeed with constant probability, the expected value of the progress measure at Π should be at least logarithmic in n; and Lemma 14 shows that each level of depth in T contributes at most $O(\frac{1}{n})$ to the expectation of the progress measure at Π (this is shown to be true not just for a random input, but even if the input is arbitrarily fixed). The proof of Theorem 3 will be completed once we prove those lemmas. The relatively simple proof of Lemma 13 appears below. The proof of Lemma 14, which is considerably more involved, spans through the rest of this section.

Lemma 13. *Let T be a* gnd*-tree with inputs in $\{0,1\}^n$. Then when T is executed over $\Upsilon(T,\varepsilon)$,*

$$\Pr[\mathsf{success}] \leq \frac{2}{\log n} \cdot \mathbb{E}\Big[|L(\Pi, X)|\Big] + \frac{1}{\sqrt{n}}.$$

Lemma 14. *Let T be a* gnd*-tree with inputs in $\{0,1\}^n$. When T is run over $\Upsilon(T,\varepsilon)$, it holds for every $x \in \{0,1\}^n$ that*

$$\mathbb{E}\Big[|L(\Pi, X)| \;\Big|\; x\Big] \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2} \cdot \frac{\mathsf{depth}(T)}{n} + \frac{5\log(1/\varepsilon)}{\varepsilon} \cdot \sqrt{\frac{\mathsf{depth}(T)}{n}}.$$

Proof of Theorem 2. Since the bound on conditional expectation in Lemma 14 holds for each x, it holds for the deconditioned expectation. Furthermore, since $\varepsilon < 1/2$, we can use $5 < 3/\varepsilon$ to get:

$$\mathbb{E}\Big[|L(\Pi, X)| \;\Big|\; x\Big] \leq \frac{\log(1/\varepsilon)}{\varepsilon^2} \left(3\frac{\mathsf{depth}T}{n} + 3\sqrt{\frac{\mathsf{depth}T}{n}}\right).$$

Substituting this into Lemma 13 yields the theorem.

Proof. (of Lemma 13). We first show that for any leaf π and input x,

$$\Pr[x \wedge \pi] \leq \Pr[\pi]\frac{2^{L(\pi,x)}}{n}. \tag{7}$$

Taking logs, it suffices to show that

$$L(\pi, x) \geq \log(\Pr[\pi \wedge x]) - \log(\frac{1}{n}\Pr[\pi]). \tag{8}$$

Using (5), the fact that $\Pr[x] = \Pr[x \oplus \mathbf{e_i}]$, and the convexity of the logarithm function, we have

$$\begin{aligned}
L(\pi, x) &= \frac{1}{n}\sum_{i=1}^{n}\Big[\log(\Pr[\pi \wedge x]) - \log(\Pr[\pi \wedge (x \oplus \mathbf{e_i})])\Big] \\
&= \log(\Pr[\pi \wedge x]) - \frac{1}{n}\sum_{i=1}^{n}\log(\Pr[\pi \wedge (x \oplus \mathbf{e_i})]) \\
&\geq \log(\Pr[\pi \wedge x]) - \log(\frac{1}{n}\sum_{i=1}^{n}\Pr[\pi \wedge (x \oplus \mathbf{e_i})]) \\
&\geq \log(\Pr[\pi \wedge x]) - \log(\frac{1}{n}\Pr[\pi]),
\end{aligned}$$

as required to prove (8) and (7).

Now let $A(\pi, x)$ denote the condition $L(\pi, x) < \frac{1}{2}\log(n)$ and let $\bar{A}(\pi, x)$ denote the complementary condition. We have

$$\Pr[\mathsf{success}] \;\; \leq \;\; \Pr[\bar{A}(\Pi, X)] + \Pr[\mathsf{success} \wedge A(\Pi, X)]. \tag{9}$$

Using the general upper bound $\mathbb{E}[|Z|] \geq B \Pr[Z \geq B]$ for any random variable Z and positive real B, the first term of (9) satisfies

$$\Pr[\bar{A}(\Pi, X)] \;\; \leq \;\; \frac{2}{\log n}\mathbb{E}[|L(\Pi, X)|].$$

To bound the second term, let $\mathsf{out}(x)$ denote the set of leaves in T that output x. Using (7) we have

$$\begin{aligned} \Pr[\mathsf{success} \wedge A(\Pi, X)] \;\; &\leq \;\; \sum_x \sum_{\{\pi \in \mathsf{out}(x):\; A(\pi,x)\}} \Pr[\pi \wedge x] \\ &\leq \;\; \sum_x \sum_{\{\pi \in \mathsf{out}(x):\; A(\pi,x)\}} \frac{1}{\sqrt{n}} \Pr[\pi]. \end{aligned}$$

Since each leaf π belongs to exactly one set $\mathsf{out}(x)$ and $\sum_\pi \Pr[\pi] = 1$ this is equal to $\frac{1}{\sqrt{n}}$. $\square$

4.1 Proof of Lemma 14: Notation

The remainder of this section is devoted to the proof of Lemma 14. For brevity we fix ε for the rest of the section, and work over the execution space Υ of T, which we also refer to as simply the execution space . Also, note that while Lemma 14 is stated for every $x \in \{0,1\}^n$, it suffices (by symmetry) to prove it in the case $x = \mathbf{0}$. We begin by defining some notation.

Restriction to $x = \mathbf{0}$. Since we prove Lemma 14 only for the case $x = \mathbf{0}$, we can use the following simplifications in our notation. For any $i \in [n]$, define

$$\begin{aligned} p_0(v) \;\; &= \;\; p(v, \mathbf{0}) = \Pr[\mathsf{vis}(v) | X = \mathbf{0}] \\ p_i(v) \;\; &= \;\; p(v, \mathbf{e_i}) = \Pr[\mathsf{vis}(v) | X = \mathbf{e_i}] \\ L_i(v) \;\; &= \;\; L_i(v, \mathbf{0}) = \log\left(\frac{p_0(v)}{p_i(v)}\right) \\ L(v) \;\; &= \;\; L(v, \mathbf{0}) = \frac{1}{n}\sum_i L_i(v). \end{aligned} \tag{10}$$

Progress measure for events. Let us now extend the above notation for any event A defined over the execution space . We denote

$$\begin{aligned} p_0[A] \;\; &= \;\; \Pr[A | X = \mathbf{0}] \\ p_i[A] \;\; &= \;\; \Pr[A | X = \mathbf{e_i}] \\ L_i(A) \;\; &= \;\; \log\left(\frac{p_0(A)}{p_i(A)}\right) \\ L(A) \;\; &= \;\; \frac{1}{n}\sum_i L_i(A). \end{aligned}$$

Progress measures for variables. Given any function F, such as L and L_i, that maps events in the execution space to real numbers, we define an operator $\tilde{F}$ that maps any random variable Z over the execution space to a real valued random variable over the execution space . Let Z be a random variable taking on values from S. Viewing Z as a function from the execution space to S, we have that for each $s \in S$, $Z^{-1}(s)$ is an event. We define $\tilde{F}(Z)$ to be the real valued random variable that gets value $F(Z^{-1}(s))$ when Z gets value s. We abuse notation by omitting the "$\sim$" and writing simply $F(Z)$.

Intuitively, the value of the random variable $L_i(Z)$ indicates how helpful the observed value of Z is for distinguishing between the zero input from $\mathbf{e_i}$. The value of $L(Z)$ indicates how well the observed value of Z is for distinguishing the zero input from a randomly chosen neighbor.

In the case that Z is the random variable Π, the notation $L(\Pi)$ as just defined coincides with the definition we get by using the progress measure for vertices defined in (10) and evaluating it at the random variable Π.

We define zero-conditioned expectation and entropy by

$$\begin{aligned} \mathbb{E}_0[Z] &= \mathbb{E}[Z|X = \mathbf{0}] \\ \mathbb{H}_0[Z] &= \mathbb{H}[Z|X = \mathbf{0}]. \end{aligned} \tag{11}$$

The goal of proving Lemma 14, formalized using the new notation, is to show that

$$\mathbb{E}_0\Big[|L(\Pi)|\Big] \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2} \cdot \frac{\mathsf{depth}(T)}{n} + \frac{5\log(1/\varepsilon)}{\varepsilon} \cdot \sqrt{\frac{\mathsf{depth}(T)}{n}}. \tag{12}$$

4.2 Analyzing a single event involving one noisy copy

To prove (12) we need to understand the behavior of $L(Z)$ for certain random variables Z (particularly, for the random variable Π). We begin by understanding its behavior on particularly simple events and variables.

Definition 15 (λ-events and λ-variables)**.** *For an index $\lambda \in \Lambda$, (i) an event over the execution space that depends only on Y^λ is called a* λ-event*, (ii) a random variable that depends only on Y^λ is called a* λ-variable*.*

In this subsection we obtain bounds on $L(A)$ and $(L(A))^2$ for λ-events A, in terms of ε, $p_0[A]$, and the quantities $\delta_i[A]$ defined by $\delta_i[A] = p_i[A] - p_0[A]$. We then derive analogous bounds for expectations of λ-variables.

In the remainder of this subsection we use A to denote a λ-event, and for simplicity we often denote

$$\begin{aligned} p_0 &= p_0[A] \\ p_i &= p_i[A] \\ \delta_i &= p_i - p_0 \\ L_i &= \log(p_0/p_i) \end{aligned}$$

when A is implicit from the context.

The following is an easy bound on $p_i[A]/p_0[A]$ in terms of ε

Lemma 16. . *For any λ-event A, For all $i \in [n]$,*

$$\frac{\varepsilon}{1-\varepsilon} \leq \frac{p_0[A]}{p_i[A]} \leq \frac{1-\varepsilon}{\varepsilon}.$$

Proof. We have

$$\begin{aligned} p_0 &= \sum_{a \in A} p_0[Y = a] \\ p_i &= \sum_{a \in A} p_i[Y = a], \end{aligned}$$

and therefore

$$\min_{a \in A} \frac{p_0[Y=a]}{p_i[Y=a]} \leq \frac{p_0}{p_i} \leq \max_{a \in A} \frac{p_0[Y=a]}{p_i[Y=a]}.$$

Since

$$\frac{p_0[Y=a]}{p_i[Y=a]} = \begin{cases} \frac{1-\varepsilon}{\varepsilon} & \text{if } a_i = 0 \\ \frac{\varepsilon}{1-\varepsilon} & \text{if } a_i = 1 \end{cases}$$

the conclusion of the lemma follows. □

To prove the bounds on $L(A)$ and $L(A)^2$, we need the following upper bound for $\sum_i {\delta_i}^2$.

Lemma 17. *For any λ-event A,*

$$\sum_{i \in [n]} \delta_i^2 \leq \frac{2}{\varepsilon^2} p_0^2 \ln(1/p_0). \tag{13}$$

(where, as usual, $p_0^2 \ln(1/p_0)$ is defined to be 0 for $p_0 = 0$.)

Proof. Since A depends only on Y^λ, we can identify A can be identified with the subset $\{0,1\}^n$ of values for Y^λ that imply the event A. The bound of the lemma will be obtained by interpreting the quantities δ_i as *biased Fourier coefficients* (see [32]) of the characteristic function of A. For this we need some definitions and facts.

Biased Fourier transform. Let F denote the inner product space consisting of functions mapping $\{0,1\}^n$ to $\mathbb{R}$ with the inner product

$$\langle g, h \rangle = \mathbb{E}_N[g(N)h(N)].$$

For $S \subseteq [n]$, the biased Fourier character $\chi_S \in F$ is defined by

$$\chi_S(x) = \prod_{i \in S} \frac{x_i - \varepsilon}{\sqrt{\varepsilon(1-\varepsilon)}}.$$

In particular $\chi_\emptyset$ is identically 1. For $f \in F$, the ε-biased fourier transform of f is the function $\hat{f}$ mapping subsets of $[n]$ to $\mathbb{R}$ defined for $S \subseteq [n]$ by

$$\hat{f}(S) = \langle f, \chi_S \rangle.$$

For ease of notation, we use $\hat{f}(i)$ instead of $\hat{f}(\{i\})$ and χ_i for $\chi_{\{i\}}$.

It is easily verified that $\{\chi_S : S \subseteq [n]\}$ is an orthonormal basis of F. It follows that for any $f \in F$

$$f = \sum_S \hat{f}(S)\chi_S,$$

and that for any $g, h \in F$,

$$\langle g, h\rangle = \sum_S \hat{g}(S)\hat{h}(S) \tag{14}$$

For $i \in [n]$, let T_i denote the linear transformation on F defined by $T_i f(x) = f(x \oplus \mathbf{e_i})$. Define the constant $b = \frac{1-2\varepsilon}{\sqrt{\frac{1-\varepsilon}{\varepsilon}}}$. By direct computation, one obtains

$$\langle T_i\chi_S, \chi_\emptyset\rangle = \begin{cases} 1 & \text{if} S = \emptyset \\ b & \text{if} S = \{i\} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

We now return to the proof of Lemma 17. Let f be the characteristic function of the set A. Using the definitions and the facts above we have

$$\begin{aligned} p_0 &= \langle f, \chi_\emptyset\rangle, \\ p_i &= \langle T_i f, \chi_\emptyset\rangle = \sum_S \hat{f}(S)\langle T_i(\chi_S), \chi_\emptyset\rangle = \hat{f}(\emptyset) + b\hat{f}(i), \\ \delta_i &= b\hat{f}(i). \end{aligned}$$

We thus have that $\sum_i(\delta_i)^2 = b^2 \sum \hat{f}(i)^2$. The proof of Lemma 17 will therefore be completed once we prove the following lemma.

Lemma 18. *Let $f : \{0,1\}^n \longrightarrow [0,1]$ be any function. Then*

$$\sum \hat{f}(i)^2 \leq \frac{2}{\varepsilon(1-\varepsilon)}(\hat{f}(\emptyset)^2 \ln(1/\hat{f}(\emptyset))).$$

Lemma 18 is a generalization to the biased case of a lemma from [33].

Proof. Let $S(f) = \sum \hat{f}(i)^2$, and let $g \in F$ be the function $g = \sum_i \hat{f}(i)\chi_i$. By equation (14),

$$S(f) = \langle f, g\rangle.$$

Defining $\ell \in F$ by $\ell(x) = \sum_i \hat{f}_i x_i$ and $\mu = \mathbb{E}_N[\ell(N)] = \varepsilon \sum_i \hat{f}_i$ we have that for all x,

$$g(x) = \frac{\ell(x) - \mu}{\sqrt{\varepsilon(1-\varepsilon)}}.$$

Consequently, we have from Lemma 12 that

$$\Pr[g(N) \geq s] = \Pr[\ell(x) - \mu \geq s\sqrt{\varepsilon(1-\varepsilon)}] \quad \leq \quad e^{-2\varepsilon(1-\varepsilon)s^2/S(f)}.$$

Since $f(x) \in [0,1]$ for every x, it holds for any positive parameter t that

$$\begin{aligned}
S(f) &= \langle f, g \rangle \\
&\leq t\mathbb{E}_N[f(N)] + \mathbb{E}_N[(g(N)-t)\mathbf{1}_{\{g(N)>t\}}] \\
&= t\hat{f}(\emptyset) + \int_{s=0}^{\infty} \Pr[g(N) - t > s]\mathrm{d}s \\
&= t\hat{f}(\emptyset) + \int_{s=t}^{\infty} \Pr[g(N) > s]\mathrm{d}s \\
&\leq t\hat{f}(\emptyset) + \int_{s=t}^{\infty} e^{-2\varepsilon(1-\varepsilon)s^2/S(f)}\mathrm{d}s \\
&\leq t\hat{f}(\emptyset) + \frac{1}{t}\int_{s=t}^{\infty} s e^{-2\varepsilon(1-\varepsilon)s^2/S(f)}\mathrm{d}s \\
&= t\hat{f}(\emptyset) + \frac{S(f)}{4\varepsilon(1-\varepsilon)t} e^{-2\varepsilon(1-\varepsilon)t^2/S(f)}.
\end{aligned}$$

Choosing $t = \frac{\sqrt{S(f)}}{\sqrt{2\varepsilon(1-\varepsilon)}}\sqrt{\ln(1/\hat{f}(\emptyset))}$ in the last expression yields

$$S(f) \leq \frac{\hat{f}(\emptyset)\sqrt{S(f)}}{\sqrt{2\varepsilon(1-\varepsilon)}}\left[\sqrt{\ln(1/\hat{f}(\emptyset))} + \frac{1}{2\sqrt{\ln(1/\hat{f}(\emptyset))}}\right].$$

Now assume that $\hat{f}(\emptyset) \leq 1/2$. In this case the second summand is bounded above by the first, and we get

$$S(f) \leq \sqrt{\frac{2S(f)}{\varepsilon(1-\varepsilon)}}\hat{f}(\emptyset)\sqrt{\ln(1/\hat{f}(\emptyset))},$$

which implies the desired inequality

$$S(f) \leq \frac{2}{\varepsilon(1-\varepsilon)}\hat{f}(\emptyset)\ln(1/\hat{f}(\emptyset)).$$

Now if $\hat{f}(\emptyset) > 1/2$ (it always holds that $\hat{f}(\emptyset) = \mathbb{E}[f] \leq 1$), take $h = 1-f$. Then $\hat{h}(\emptyset) \leq 1/2$, and we have

$$S(f) = S(h) \leq \frac{2}{\varepsilon(1-\varepsilon)}\left(\hat{h}(\emptyset)^2\ln(1/\hat{g}(\emptyset))\right). \tag{16}$$

Since $\hat{f}(\emptyset) = 1-\hat{g}(\emptyset)$ we may, using Proposition 10, replace $\hat{g}(\emptyset)$ by $\hat{f}(\emptyset)$ in (16). This completes the proof of Lemma 18, and also the proof of Lemma 17. □

□

We are now ready to state and prove the bound for λ-events.

Lemma 19. *Let A be a λ-event. Then*

$$L(A) \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\log(1/p_0) - \frac{\log e}{p_0 n}\sum_{i=1}^{n}\delta_i \tag{17}$$

$$L^2(A) \leq \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n}\log(1/p_0). \tag{18}$$

Proof. By definition, $L(A) = -\frac{1}{n}\sum_{i=1}^{n}\log\left(1+\frac{\delta_i}{p_0}\right) = -\frac{\log e}{n}\sum_{i=1}^{n}\ln\left(1+\frac{\delta_i}{p_0}\right)$. To obtain (17) we rewrite the expression for $L(A)$ in terms of the function b defined in (4), and use the second part of Corollary 9 together with the inequality $p_0/p_i \geq \varepsilon$, and then use Lemma 17 to get

$$\begin{aligned} L(A) &= \frac{-\log e}{n}\left[\sum_{i=1}^{n}\left(\frac{\delta_i}{p_0}\right)^2 b\left(\frac{\delta_i}{p_0}\right) + \sum_{i=1}^{n}\frac{\delta_i}{p_0}\right] \\ &\leq \frac{\log e}{n}\left[\frac{2\ln(1/\varepsilon)}{p_0^2}\sum_{i=1}^{n}\delta_i^2 - \frac{1}{p_0}\sum_{i=1}^{n}\delta_i\right] \\ &\leq \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\log(1/p_0) - \frac{\log e}{p_0 n}\sum_{i=1}^{n}\delta_i \end{aligned}$$

For (18) we rewrite the expression for $L(A)^2$ in terms of the function a defined in (3), and obtain the chain of inequalities

$$\begin{aligned} L(A)^2 &= \left(\frac{\log e}{n}\sum_{i=1}^{n}\left(\frac{\delta_i}{p_0}\right) a\left(\frac{\delta_i}{p_0}\right)\right)^2 \\ &\overset{(1)}{\leq} \left(\frac{2\log(1/\varepsilon)}{p_0}\frac{1}{n}\sum_{i=1}^{n}\delta_i\right)^2 \overset{(2)}{\leq} \left(\frac{2\log(1/\varepsilon)}{p_0}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\delta_i^2}\right)^2 \\ &\leq \frac{4(\log(1/\varepsilon))^2}{p_0^2 n}\sum_{i-1}^{n}\delta_i^2 \\ &\overset{(3)}{\leq} \frac{4(\log(1/\varepsilon))^2}{p_0^2 n}\frac{2p_0^2\log(1/p_0)}{\varepsilon^2\log e} \leq \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n}\log(1/p_0). \end{aligned}$$

Here (1) comes from the first part of Corollary 9, (2) follows from the Cauchy-Schwartz inequality and (3) is obtained from Lemma 17. ☐

We now deduce an analog of Lemma 19 for expectations of λ-variables.

Corollary 20. *Let Z be a λ-random variable. Then*

$$\mathbb{E}_0[L(Z)] \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z]$$

$$\textit{and} \qquad \mathbb{E}_0[L(Z)^2] \leq \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n}\mathbb{H}_0[Z].$$

Proof. In the following expressions, the index z ranges over values of Z. Using the definition of $L(Z)$ and applying (17) we get

$$\begin{aligned} \mathbb{E}_0[L(Z)] &= \sum_z p_0[Z=z]L([Z=z]) \\ &\leq \sum_z \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}p_0[Z=z]\log(1/p_0[Z=z]) + \sum_z \frac{\log e}{n}\sum_{i=1}^{n}\delta_i[Z=z] \\ &= \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z], \end{aligned}$$

where the last transition is obtained by noting that the second double sum is 0 since for each fixed i, $\sum_z p_i[Z = z] = \sum_z p_0[Z = z] = 1$ and so $\sum_z \delta_i[Z = z] = 0$.

The bound on $\mathbb{E}_0[L(Z)^2]$ follows by a similar (actually simpler) calculation using (18). $\square$

4.3 Box events

In this subsection we define box events and box random variables, which are more general than λ-events and λ-variables. We then extend the bounds from the previous section to the box case, and apply these bounds to the random variable Π, which turns out to be a box random variable.

Box events. A *box event* is an intersection of λ-events for possibly different λ's. Any box event B can be uniquely written as $B = \bigcap_{\lambda\in\Lambda} B^\lambda$ where each B^λ is a λ-event. A box *random variable* is a random variable Z for which each of the events $[Z = s]$ is a box event.

Our interest in box events stems from the following immediate fact.

Fact 21. *For each vertex v in T the event* $\mathsf{vis}(v)$ *is a box event, and therefore* Π *is a box random variable.*

Let $B = \bigcap_{\lambda\in\Lambda} B^\lambda$ be a box event. The different λ-events $(B^\lambda : \lambda \in \Lambda)$ may well be dependent, as the variables Y^λ, each being correlated with X, are dependent. We observe, however, that once we condition on a *fixed value of* X they become mutually independent. This implies the following properties for box events.

Fact 22. *Every box event B satisfies*

$$p_0[B] \;=\; \prod_{\lambda\in\Lambda} p_0[B^\lambda].$$

Moreover, for every $i \in \{1, \ldots, n\}$,

$$\begin{aligned} p_i[B] &= \prod_{\lambda\in\Lambda} p_i[B^\lambda], \\ L_i(B) &= \sum_{\lambda\in\Lambda} L_i\left(B^\lambda\right), \\ \textit{and} \qquad L(B) &= \sum_{\lambda\in\Lambda} L\left(B^\lambda\right). \end{aligned} \tag{19}$$

Contribution of copies to progress. Looking at (19) one observes that for box events the contribution to the progress measure of each noisy copy can be easily singled out. For a box event B and for $\lambda \in \Lambda$ we define

$$\begin{aligned} L_i^\lambda(B) &= L_i\left(B^\lambda\right), \\ \text{and} \qquad L^\lambda(B) &= L\left(B^\lambda\right). \end{aligned}$$

In the same way that we extended L and L_i from events to random variables, we extend L^λ and L_i^λ from box events to box random variables.

Some more notation. The following notation is used for the bounds for box random variables below. Throughout this subsection we use the letter y to denote points in $(\{0,1\}^n)^\Lambda$ representing an assignment to Y^Λ (recall that Y^Λ is the set of all noisy copies of the input). We use y^λ to denote a value of Y^λ. Also, writing $\hat{\lambda}$ for the set $\Lambda - \{\lambda\}$, we use $y^{\hat{\lambda}}$ to denote a value of $Y^{\hat{\lambda}}$, namely an assignment to all noisy copies but Y^λ. In a context where a point $y \in (\{0,1\}^n)^\Lambda$ is specified, we use $y^{\hat{\lambda}}$ and y^λ to denote the restrictions of y to the λ coordinate or the set $\hat{\lambda}$ of coordinates respectively. We also write $y = (y^\lambda, y^{\hat{\lambda}})$.

Lemma 23. *Let Z be a box random variable. Then*

$$\begin{aligned} \mathbb{E}_0[L^\lambda(Z)] &\leq \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z|Y^{\hat{\lambda}}] \\ \mathbb{E}_0[(L^\lambda(Z))^2] &\leq \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n}\mathbb{H}_0[Z|Y^{\hat{\lambda}}]. \end{aligned}$$

Proof. Since Z is a box random variable, its value is determined by the value of Y^λ. We can therefore write $Z = Z(y) = Z(y^\lambda, y^{\hat{\lambda}})$. Lemma 23 is obtained by applying Corollary 20 to restrictions of Z of the form $Z(Y^\lambda, y^{\hat{\lambda}})$, which are λ random variables. Let us therefore denote the random variable $Z(Y^\lambda, y^{\hat{\lambda}})$ by $Z_{y^{\hat{\lambda}}}$. Letting y range over all values of Y^Λ, we have

$$\begin{aligned} \mathbb{E}_0[L^\lambda(Z)] &= \sum_y p_0[Y^\Lambda = y]L^\lambda([Z = Z(y)]) \\ &= \sum_y p_0[Y^\Lambda = y]L([Z = Z(y)]^\lambda) \\ &= \sum_y p_0[Y^\Lambda = y]L([Z(Y^\lambda, y^{\hat{\lambda}}) = Z(y)]) \\ &= \sum_{y^{\hat{\lambda}}} p_0[Y^{\hat{\lambda}} = y^{\hat{\lambda}}]\mathbb{E}_0\left[L(Z_{y^{\hat{\lambda}}})\right] \\ &\leq \sum_{y^{\hat{\lambda}}} p_0[Y^{\hat{\lambda}} = y^{\hat{\lambda}}]\frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z|Y^{\hat{\lambda}} = y^{\hat{\lambda}}] \qquad \text{(by Corollary 20)} \\ &= \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z|Y^{\hat{\lambda}}]. \end{aligned}$$

The bound on $\mathbb{E}_0[(L^\lambda[Z])^2]$ is obtained similarly. □

Lemma 24. *For any box random variable Z,*

$$\mathbb{H}_0[Z] = \sum_\lambda \mathbb{H}_0[Z|Y^{\hat{\lambda}}].$$

Proof. As in the proof of Lemma 23 we can write $Z = Z(y)$, since Z only depends on the value of Y^Λ. We denote the box events $[Z = Z(y)]$ by $B(y)$, and let $\mathcal{B} = \{B(y)\}_y$. In the sums below,

y ranges over the values of Y^Λ and $y^{\hat{\lambda}}$ ranges over the values of $Y^{\hat{\lambda}}$.

$$
\begin{aligned}
\mathbb{H}_0[Z] &= \sum_{B\in\mathcal{B}} p_0[B]\log p_0[B] \\
&= \sum_{B\in\mathcal{B}} p_0[B]\sum_{\lambda}\log p_0[B^\lambda] \\
&= \sum_{\lambda}\sum_{B\in\mathcal{B}} p_0[B]\log p_0[B^\lambda] \\
&= \sum_{\lambda}\sum_{y} p_0[Y^\Lambda = y]\log p_0[B(y)^\lambda] \\
&= \sum_{\lambda}\sum_{y^{\hat{\lambda}}}\Big(p_0[Y^{\hat{\lambda}} = y^{\hat{\lambda}}]\cdot\sum_{y^\lambda} p_0[Y^\lambda = y^\lambda]\log p_0[B(y)^\lambda]\Big) \\
&= \sum_{\lambda}\sum_{y^{\hat{\lambda}}}\Big(p_0[Y^{\hat{\lambda}} = y^{\hat{\lambda}}]\cdot\sum_{y^\lambda} p_0[Y^\lambda = y^\lambda|Y^{\hat{\lambda}} = y^{\hat{\lambda}}]\log p_0[B(y)|Y^{\hat{\lambda}} = y^{\hat{\lambda}}]\Big) \\
&= \sum_{\lambda}\sum_{y^{\hat{\lambda}}}\Big(p_0[Y^{\hat{\lambda}} = y^{\hat{\lambda}}]\cdot\sum_{y^\lambda}\mathbb{H}_0[Z|Y^{\hat{\lambda}} = y^{\hat{\lambda}}]\Big) \\
&= \sum_{\lambda}\mathbb{H}_0[Z|Y^{\hat{\lambda}}].
\end{aligned}
$$

□

Since Π is a box random variable, we obtain the following as an immediate consequence of Lemmas 23 and 24.

Corollary 25. *Let T be a* gnd*-tree. Then the random variable Π defined over the execution space of T satisfies*

$$\mathbb{E}_0[L(\Pi)] \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[\Pi], \tag{20}$$

and

$$\mathbb{E}_0\Big[\sum_{\lambda}(L^\lambda(\Pi))^2\Big] \leq \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n}\mathbb{H}_0[\Pi]. \tag{21}$$

4.4 Dealing with correlations

Since $\mathbb{H}_0[\Pi]$ is obviously bounded from above by $\mathsf{depth}(T)$, we would be happy to prove a bound on $\mathbb{E}_0[|L(\Pi)|]$ of the form $\mathbb{H}_0[\Pi]\cdot\mathrm{polylog}(1/\varepsilon)/n$. Corollary 25 comes close: noting that $L(\Pi) = \sum_\lambda L^\lambda(\Pi)$ we see that if only the variables $L^\lambda(\Pi)$ were uncorrelated, (21) would have given a bound on the second moment of $L(\Pi)$, and together with (20) we would have easily obtained the desired bound (since $L(\Pi)$ can be negative, (20) by itself is not sufficient to get the bound).

However in general the variables $L^\lambda(\Pi)$ can be correlated. For example, T can decide to query one noisy copy only if queries to a different copy have yielded very little gain to the progress measure. This would introduce correlation between the contributions to the progress measure of those two copies. To handle the correlations we define two random variable, $Q = Q(\Pi)$ and $R = R(\Pi)$. Random variable Q is the sum over nodes on the path to Π, of the expected progress due to the query at that node (a precise definition appears below), and $R = L(\Pi) - Q$.

We show that Q has the same expectation as $L(\Pi)$ but is nonnegative, and thus $\mathbb{E}_0[|Q|] = \mathbb{E}_0[Q] = \mathbb{E}_0[L(\Pi)]$, which we already know is suitably small from (20). It is then left to show a bound on $\mathbb{E}_0[|R|]$. We show that R can be written as a sum $R = \sum_{\lambda\in\Lambda} R^\lambda$ of contributions from the individual noisy copies, but unlike the variables $L^\lambda(\Pi)$, the R^λ's are uncorrelated. The bound on $\mathbb{E}_0[|R|]$ is then obtained by proving a bound on $\mathbb{E}_0[(R^\lambda)^2]$ for all $\lambda \in \Lambda$.

The variables L, Q, and R

In this subsection we write L to denote the random variable $L(\Pi)$, and L^λ for $L^\lambda(\Pi)$ (we still use the notation $L(v)$ and $L^\lambda(v)$ for vertices other than Π). Recall that V is the set of vertices of T, and define a partial order on V by writing $v \le w$ if v lies on the path from the root to w. We say that v and w are *incomparable* if neither $v \le w$ nor $w \le v$. For an internal vertex v, let $v0$ and $v1$ denote its children.

In order to define Q, we first decompose L as a sum of random variables $\{L_v\}_{v\in V}$, where L_v represents the contribution of the query at vertex v to L. For vertices $v, w \in T$ we define

$$L_v(w) = \begin{cases} L(v0) - L(v) & \text{if } v0 \le w \\ L(v1) - L(v) & \text{if } v1 \le w \\ 0 & \text{otherwise.} \end{cases}$$

$L_v(w)$ measures the contribution of the query made at v to $L(w)$. It easily follows from the definition that $L(w) = \sum_{v\in V} L_v(w)$. Moreover, one can verify that

$$L^\lambda(w) = \sum_{v\in V^\lambda} L_v(w)$$

(where V^λ is the set of vertices in T that query the noisy copy indexed by λ). For every v we define the random variable $L_v = L_v(\Pi)$. We further define the following random variables:

$$\begin{aligned} Q_v &= \begin{cases} \mathbb{E}_0[L_v|\mathsf{vis}(v)] & \text{if } v \le \Pi \\ 0 & \text{otherwise ,} \end{cases} \\ R_v &= L_v - Q_v, \\ Q &= \sum_{v\in V} Q_v \ , \qquad \text{and} \quad Q^\lambda = \sum_{v\in V^\lambda} Q_v \quad \text{for every } \lambda\in\Lambda, \\ R &= \sum_{v\in V} R_v \ , \qquad \text{and} \quad R^\lambda = \sum_{v\in V^\lambda} R_v \quad \text{for every } \lambda\in\Lambda. \end{aligned}$$

Q_v can be thought of as an a priori approximation to L_v, and R_v can be thought of as the a posteriori correction to Q_v. Following are some properties of Q_v and R_v .

Proposition 26. *For any internal vertices v and w,*

1. $\mathbb{E}_0[Q_v] = \mathbb{E}_0[L_v]$.

2. *If $v \geq w$, $\mathbb{E}_0[Q_v|\mathsf{vis}(w)] = \mathbb{E}_0[L_v|\mathsf{vis}(w)]$.*

3. *If $v \geq w$, $\mathbb{E}_0[R_v|\mathsf{vis}(w)] = 0$.*

4. *$Q_v \geq 0$.*

Proof. Part 1 is immediate from the definitions of Q_v and L_v. For part 2, since $Q_v = L_v = 0$ when w is not visited we have $\mathbb{E}_0[Q_v|\mathsf{vis}(w)] = \frac{1}{\Pr[\mathsf{vis}(w)]}\mathbb{E}_0[Q_v] = \frac{1}{\Pr[\mathsf{vis}(w)]}\mathbb{E}_0[L_v] = \mathbb{E}_0[L_v|\mathsf{vis}(w)]$. The third part follows immediately from the second part. For part 4, fix v and for $j \in \{0,1\}$ and $i \in \{0,\dots,n\}$ let $a_i(j) = p_i(vj)/p_i(v)$. Observe that $a_i(0) + a_i(1) = 1$ for each i. Now

$$Q_v = \frac{1}{n}\sum_{i=1}^{n}\left(a_0(0)\log\frac{a_0(0)}{a_i(0)} + a_0(1)\log\frac{a_0(1)}{a_i(1)}\right),$$

and each term of the sum is nonnegative by Proposition 11. □

Proposition 27. *For any vertices $v, w \in V$,*

1. *If v and w are incomparable then $\mathbb{E}_0[Q_vQ_w] = 0$.*

2. *If $v \leq w$ then $\mathbb{E}_0[Q_vQ_w] = \mathbb{E}_0[Q_vL_w]$.*

3. *If $v \neq w$ then $\mathbb{E}_0[R_vR_w] = 0$*

Proof. If v and w are incomparable then on any execution of T either v or w are not visited, so $Q_vQ_w = R_vR_w = 0$ and we have part 1. Part 2 follows since both Q_w and L_w are 0 if w is not visited, and conditioned on w being visited Q_v is constant. Formally,

$$\begin{aligned}
\mathbb{E}_0[Q_vQ_w] &= \Pr[\mathsf{vis}(w)]\mathbb{E}_0[Q_vQ_w|\mathsf{vis}(w)] \\
&= \Pr[\mathsf{vis}(w)]\mathbb{E}_0[Q_v|\mathsf{vis}(w)]\mathbb{E}_0[Q_w|\mathsf{vis}(w)] \\
&= \Pr[\mathsf{vis}(w)]\mathbb{E}_0[Q_v|\mathsf{vis}(w)]\mathbb{E}_0[L_w|\mathsf{vis}(w)] \\
&= \Pr[\mathsf{vis}(w)]\mathbb{E}_0[Q_vL_w|\mathsf{vis}(w)] = \mathbb{E}_0[Q_vL_w].
\end{aligned}$$

For the third part, if v and w are incomparable then R_vR_w is 0. If $v < w$ then the product is 0 unless w is visited. But conditioned on w being visited R_v is a constant, and $\mathbb{E}_0[R_w|\mathsf{vis}(w)] = 0$. □

As an immediate corollary of part 3 of Proposition 27 we have the following.

Corollary 28. *For $\lambda \neq \kappa \in \Lambda$, $\mathbb{E}_0[R^\lambda R^\kappa] = 0$.*

Bounds for L, Q, and R

Having defined L, Q, R and related variables, and proven their basic properties, we now prove some bounds for these variables that ultimately yield (12). The first bound is on the contribution that the subtree of T below a given vertex can make to L^λ.

Proposition 29. *Let $v \in V^\lambda$. Then $|\sum_{w\in V^\lambda : w \geq v} L_w| \leq 2\log(1/\varepsilon)$.*

Proof. The sum inside the absolute value equals 0 unless $\Pi \geq v$ does not hold. If $\Pi \geq v$ then the sum is equal to $|L^\lambda(\Pi) - L^\lambda(v)| \leq |L^\lambda(\Pi)| + |L^\lambda(v)| \leq 2\log(1/\varepsilon)$ by Lemma 16. □

Next we show that the second moment of R^λ cannot be much higher than the second moment of L^λ.

Lemma 30. *For each $\lambda \in \Lambda$, $\mathbb{E}_0[(R^\lambda)^2] \leq \mathbb{E}_0[(L^\lambda)^2] + 6\log(1/\varepsilon)\mathbb{E}_0[L^\lambda]$.*

Proof.

$$\begin{aligned}
\mathbb{E}_0[(R^\lambda)^2] &= \mathbb{E}_0[(L^\lambda - Q^\lambda)^2] \\
&\leq \mathbb{E}_0[(L^\lambda)^2] + 2|\mathbb{E}_0[L^\lambda Q^\lambda]| + \mathbb{E}_0[(Q^\lambda)^2]. \qquad (22)
\end{aligned}$$

Since by Lemma 16 we have $|L^\lambda| \leq \log(1/\varepsilon)$, we can bound the second term in (22) by

$$2|\mathbb{E}_0[L^\lambda Q^\lambda]| \leq 2\log(1/\varepsilon)\mathbb{E}_0[|Q^\lambda|] = 2\log(1/\varepsilon)\mathbb{E}_0[L^\lambda].$$

For the third term in (22), we have

$$\begin{aligned}
\mathbb{E}_0[(Q^\lambda)^2] &= \sum_{v,w \in V^\lambda} \mathbb{E}_0[Q_v Q_w] \\
&\overset{(1)}{\leq} 2 \sum_{v,w \in V^\lambda : v \leq w} \mathbb{E}_0[Q_v Q_w] \\
&\overset{(2)}{=} 2 \sum_{v,w \in V^\lambda : v \leq w} \mathbb{E}_0[Q_v L_w] \\
&\leq 2 \sum_{v \in V^\lambda} \mathbb{E}_0\Big[Q_v \cdot \sum_{w \in V^\lambda : w \geq v} L_w\Big] \\
&\overset{(3)}{=} 2\mathbb{E}_0\Big[\sum_{v \in V^\lambda} Q_v\Big] \cdot 2\log(1/\varepsilon) \\
&= 4\log(1/\varepsilon)\mathbb{E}_0[Q^\lambda] \overset{(4)}{=} 4\log(1/\varepsilon)\mathbb{E}_0[L^\lambda],
\end{aligned}$$

where (1) follows from part 1 of Proposition 27, (2) follows from part 2 of Proposition 27, (3) follows from Proposition 29, and (4) follows from the first part of 26. Combining the bounds for the second and third terms in (22), the lemma is obtained. □

Completing the proof

We are now ready to prove (12), which implies Lemma 14, which in turn completes the proof of Theorem 3.

Since $L = Q + R$, we have

$$\mathbb{E}_0[|L|] \leq \mathbb{E}_0[|Q|] + \mathbb{E}_0[|R|]. \qquad (23)$$

Using the parts 1 and 4 of Proposition 26, we have that $\mathbb{E}_0[|Q|] = \mathbb{E}_0[Q] = \mathbb{E}_0[L]$. Hence using Corollary 20 and the immediate fact that $\mathbb{H}_0[\Pi] \leq \mathsf{depth}(T)$, we can bound the first term in the r.h.s. of (23) by

$$\mathbb{E}_0[|Q|] = \mathbb{E}_0[L] \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2 n} \cdot \mathbb{H}_0[\Pi] \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2} \cdot \frac{\mathsf{depth}(T)}{n}. \qquad (24)$$

For the second term of the r.h.s. of (23), we have using the Cauchy–Schwartz inequality,

$$\begin{aligned}
\mathbb{E}_0[|R|]^2 &\le \mathbb{E}_0[R^2] \\
&= \sum_\lambda \mathbb{E}_0[(R^\lambda)^2] \qquad \text{(by part 3 of Proposition 27)} \\
&\le \sum_\lambda \left(\mathbb{E}_0[(L^\lambda)^2] + 6\log(1/\varepsilon)\mathbb{E}_0[L^\lambda]\right) \qquad \text{(by Lemma 30)} \\
&= \left(\sum_\lambda \mathbb{E}_0[(L^\lambda)^2]\right) + 6\log(1/\varepsilon)\mathbb{E}_0[L] \\
&\le \frac{24(\log(1/\varepsilon))^2}{\varepsilon^2 n}\mathbb{H}_0[\Pi] \qquad \text{(by Corollary 25)} \\
&\le \frac{24(\log(1/\varepsilon))^2}{\varepsilon^2}\frac{\mathsf{depth}(T)}{n}.
\end{aligned}$$

Combining the bounds for Q and for R we get

$$\mathbb{E}_0[|L|] \le \frac{3\log(1/\varepsilon)}{\varepsilon^2} \cdot \frac{\mathsf{depth}(T)}{n} + \frac{5\log(1/\varepsilon)}{\varepsilon} \cdot \sqrt{\frac{\mathsf{depth}(T)}{n}},$$

which completes the proof.

5 Lower Bound for Decision Functions

In this section we prove lower bounds for computing *decision functions* (function that output a single boolean value), as stated in Thorem 5. Out lower bounds are stated in terms of the *sensitivity* of the function to be computed, which is defined as follows.

Definition 31. *Let $f : \{0,1\}^n \to \{0,1\}$ be any function. The sensitivity of f at input $x \in \{0,1\}^n$, denoted $s_x(f)$ is the number of indices $i \in [n]$ such that $f(x) \ne f(x \oplus \mathbf{e_i})$. The sensitivity of f is the maximum of $s_x(f)$ over all x.*

For example, AND, OR and PAR all have sensitivity n, and every other symmetric boolean function has sensitivity at least $n/2$.

We have the following lower-bound:

Theorem 32. *Let $\varepsilon \in (0,1/2)$ and $\delta \in (0,1/16)$, and let f be an n-variate boolean function. Any randomized gnd-tree that for every input x, outputs $f(x)$ with probability $1-\delta$ when run with noise parameter ε satisfies:*

$$\mathsf{depth}(T) \ge \frac{\varepsilon^2 \log(1/4\delta)}{50\log^2(1/\varepsilon)} \cdot s(f).$$

Thus gnd-trees that compute symmetric boolean functions with small constant probability of error require linear depth. While this fact is trivial for noisy broadcast protocols (since the protocol must consider all bits of the input), it is not so for gnd-trees. In fact, we do not know how to achieve linear lower bounds for gnd-trees without applying the full power of the techniques developed in Section 4.

Theorem 32 also implies that gnd-trees for symmetric functions that are correct with probability $1 - n^{\Omega(1)}$ must have depth $\Omega(n \log n)$. Before proving Theorem 32, we derive a corollary for noisy broadcast protocols which shows that they require $\Omega(n \log \log n)$ broadcasts to compute symmetric functions with polynomially small error probability.

Corollary 33. *Let $\varepsilon \in (0, 1/2)$ and $\beta > 0$. Let f be an n-variate boolean function. Any randomized noisy broadcast protocol for f that, for every input x, outputs $f(x)$ with probability $1 - n^{-\beta}$ when run with noise parameter ε, uses at least $\Omega(s(f) \cdot \frac{\log \log(n^\beta)}{\log(1/\varepsilon)})$ broadcasts.*

Proof. Suppose that A is a noisy broadcast protocol for f that on any input x has error probability $1 - n^{-\beta}$ and suppose A uses $t \cdot s(f)$ broadcasts. Applying Theorem 6 with $\alpha = \frac{s(f)}{2n}$ yields a subset K of $s(f)/2$ variables given by the theorem. Let z be an input with sensitivity $s(f)$. If we fix the variables of K according to z, the resulting function f' with variables indexed by $[n] - K$ has sensitivity at least $s(f)/2$. Furthermore, by the conclusion of Theorem 6, there is a gnd-tree T that uses $2t \cdot s(f)$ queries and for all inputs $y \in \{0, 1\}^{[n]-K}$ outputs $f'(y)$ with probability at least $1 - n^{-\beta}$ when run with noise parameter ε^{2t}. By Theorem 32, we have:

$$2t \cdot s(f) \geq \frac{\varepsilon^{4t} \log(n^\beta/4)}{50 \log^2(1/\varepsilon^{2t})} \cdot s(f),$$

from which the claimed lower bound easily follows. □

The decision-function execution space Υ^z. Let T be a gnd-tree. To prove Theorem 32, it suffices to prove a suitable upper bound on the probability that T is correct, when T is executed on an input selected at random from some distribution. Rather than use the uniform distribution as we did for Theorem 3, we use a distribution tailored to f.

Let $z \in \{0, 1\}^n$ be chosen such that $s_z(f) = s(f)$, and let $I \subseteq [n]$ be such that $f(z) \neq f(z \oplus \mathbf{e_i})$ for $i \in I$. The ε-execution space for f, denoted $\Upsilon^z = \Upsilon^z(T, \varepsilon)$ is similar to Υ defined in Section 4, except that X is set to z with probability $1/2$, and to $z \oplus \mathbf{e_i}$ with probability $1/2s$ for each $i \in I$.

Without loss of generality, we may assume that $z = \mathbf{0}$ and that $f(\mathbf{0}) = 0$ (we can replace f by the function mapping x to $f(x \oplus z) \oplus f(z)$ and make the analogous change to T). Furthermore, we may assume that $s(f) = n$ by fixing all variables outside I to 0 in both f and T. We therefore have that the execution space for f is $\Upsilon^{\mathbf{0}}$, and it satisfies that the input X is equal to $\mathbf{0}$ with probability $1/2$ (in which case the $f(X) = 0$), and to any of the vectors $\mathbf{e_i}$ with probability $1/2n$ (in which case $f(X) = 1$. As in the proof of Theorem 3, since we are proving a lower bound with respect to an input distribution, we may now assume that T is deterministic.

The progress measure. We use the same progress measure L as was used in Section 4. Note that once X is fixed to a value in $\{\mathbf{0}, \mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_n}\}$ the spaces $\Upsilon(T, \varepsilon)$ and $\Upsilon^{\mathbf{0}}(T, \varepsilon)$ become identical, and therefore the values of $p_0(v), p_i(v), L_i(v)$ and $L(v, \mathbf{0})$, as defined at the beginning of Section 4.1, remain unchanged when defined over $\Upsilon^{\mathbf{0}}(T, \varepsilon)$. For the same reason, Lemma 14 holds for $x = \mathbf{0}$ with Υ replaced by $\Upsilon^{\mathbf{0}}$. Using the notation for zero-conditioned expectation as defined in (11) (again, the definition is the same whether it is done over Υ or over $\Upsilon^{\mathbf{0}}$ since it fixes X), we have that (12) holds over $\Upsilon^{\mathbf{0}}(T, \varepsilon)$.

Adaptation of notation. As in Section 4, we denote $L(v) = L(v, \mathbf{0})$. We also define the following random variables and events over $\Upsilon^{\mathbf{0}}(T, \varepsilon)$:

- Let Π denote the leaf that terminates the execution of T over $\Upsilon^{\mathbf{0}}(T, \varepsilon)$.
- Let err denote the event that T does not output $f(X)$ (when this occurs we say that T errs).

5.1 The analogue of Lemma 13

Having observed that Lemma 14 applies in the decision-function case, we now need an analogue of Lemma 13. First we derive a lower bound on the probability that T errs, in terms of the probability that on input $\mathbf{0}$, T arrives at a leaf π where $L(\pi)$ is small. Define, for every $t \geq 1$,

$$A_t = \{\pi \text{ leaf of } T \mid L(\pi) \leq \log t\} \tag{25}$$

Lemma 34. *For every $t \geq 1$,*

$$\Pr[\mathsf{err}] \geq \frac{\Pr[A_t | X = 0]}{2t}.$$

Proof. For every leaf π we have

$$\begin{aligned} L(\pi) &= \log\left(\Pr[\Pi = \pi | X = 0]\right) - \frac{1}{n}\sum_i \log\left(\Pr[\Pi = \pi | X = e_i]\right) && \text{(by definition)} \\ &\geq \log\left(\Pr[\Pi = \pi | X = 0]\right) - \log\left(\frac{1}{n}\sum_i \Pr[\Pi = \pi | X = e_i]\right) && \text{(concavity of log)} \\ &= \log\left(\Pr[\Pi = \pi | X = 0]\right) - \log\left(\Pr[\Pi = \pi | X \neq 0]\right). && \text{(by definition of } \Upsilon^{\mathbf{0}}) \end{aligned}$$

Exponentiating both sides of the inequality with base 2, we get

$$\Pr\left[\Pi = \pi | X \neq 0\right] \geq \frac{\Pr\left[\Pi = \pi | X = 0\right]}{2^{L(\pi)}}, \tag{26}$$

Now let $A_t^0 = \{\pi \in A_t | out_\pi = 0\}$ denote the set of leaves in A_t where T outputs zero, and similarly, let $A_t^1 = \{\pi \in A_t | out_\pi = 1\}$. Since $f(\mathbf{0}) = 0$ and $f(\mathbf{e_i}) = 1$ for all $i \in [n]$, we have

$$\begin{aligned} \Pr[\mathsf{err}] &= \tfrac{1}{2}\Pr[\mathsf{err} | X = 0] + \tfrac{1}{2}\Pr[\mathsf{err} | X \neq 0] \\ &\geq \tfrac{1}{2}\Pr\left[A_t^1 | X = 0\right] + \tfrac{1}{2}\Pr\left[A_t^0 | X \neq 0\right] \\ &\geq \tfrac{1}{2}\Pr\left[A_t^1 | X = 0\right] + \tfrac{1}{2}\Pr\left[A_t^0 | X = 0\right] \cdot \tfrac{1}{t} && \text{(by (26) and definition of } A_t) \\ &\geq \tfrac{1}{2t} \cdot \left(\Pr\left[A_t^1 | X = 0\right] + \Pr\left[A_t^0 | X = 0\right]\right) \\ &= \frac{\Pr\left[A_t | X = 0\right]}{2t}, \end{aligned}$$

concluding the proof. ∎

The following lemma is the analogue of Lemma 13 for the decision-function case.

Lemma 35. *Denote $\delta = \Pr[\mathsf{err}]$. Then if $\delta \leq 1/4$,*

$$\mathbb{E}\left[|L(\Pi)| | X = 0\right] \geq \tfrac{1}{2}\log(1/4\delta).$$

Proof. Take $t = 1/4\delta$. Then by Lemma 34 we have

$$\delta = \Pr[\mathsf{err}] \geq \frac{\Pr[A_t|X=0]}{2t} = 2\delta \cdot \Pr[A_t|X=0],$$

and thus

$$\Pr[A_t|X=0] \leq 1/2.$$

Now on the complement of A_t with respect to the set of leaves in T, the values of L are greater than $\log t$. We therefore have

$$\mathbb{E}\left[|L(\Pi)||X=0\right] \geq (1 - \Pr[A_t|X=0]) \cdot \log(t) \geq \tfrac{1}{2}\log(1/4\delta)$$

as desired. □

5.2 Proof of Theorem 32

We are now ready to complete the proof of Theorem 32.

Proof of Theorem 32. Let us consider the execution of T over $\Upsilon^{\mathbf{0}}(T, \varepsilon)$. It is given that $\Pr[\mathsf{err}] = \delta < \frac{1}{16}$. Applying Lemma 35 and (12), we thus have that

$$\frac{3\log(1/\varepsilon)}{\varepsilon^2} \cdot \frac{\mathsf{depth}(T)}{n} + \frac{5\log(1/\varepsilon)}{\varepsilon} \cdot \sqrt{\frac{\mathsf{depth}(T)}{n}} \geq t, \tag{27}$$

where $t = \frac{1}{2}\log\left(\frac{1}{4\delta}\right) \geq 1$. Therefore either the first or the second summand in (27) is greater than $\frac{t}{2}$, and by isolating $\mathsf{depth}(T)$ in the resulting inequalities we obtain

$$\mathsf{depth}(T) \geq \min\left\{\frac{\varepsilon^2 tn}{3\log(1/\varepsilon)}, \frac{\varepsilon^2 t^2 n}{25\log^2(1/\varepsilon)}\right\} \geq \frac{\varepsilon^2 tn}{25\log^2(1/\varepsilon)},$$

as desired. □

6 Proof of reduction theorem (Theorem 6)

Let us now prove Theorem 6. We are given a noisy broadcast protocol $\mathcal{P}$ that uses k broadcasts. We are also given a noise-parameter ε and another parameter $\alpha > 1$. Our goal is to select a subset K of size n/α and show that for each partial assignment ρ that fixes K there is a (possibly randomized) gnd-tree T that uses $2k$ queries and, for any input $x \in \{0,1\}^{[n]-K}$, when T is run on x with noise parameter $\varepsilon^{\alpha k}$ it gives the same output that $\mathcal{P}$ gives when run on the input $x\rho \in \{0,1\}^n$ with noise parameter ε.

We select K to be the index set of processors that broadcast more than $\alpha k/n$ times in $\mathcal{P}$. Since there are k broadcasts in $\mathcal{P}$, $|K| < \alpha n$.

Now fix a partial assignment ρ to K. The gnd-tree T is obtained by a sequence of reductions. These reductions involve two intermediate models of noisy computation which we now define.

The *semi-noisy broadcast model* $SNB(\varepsilon)$ is similar to the noisy broadcast model. In this model there are n *input processors* $Q_1, \ldots, Q_n$, a receiver P_0, and a collection $\{P_\lambda : \lambda \in \Lambda\}$ of *auxiliary processors* (where Λ is an arbitrary index set). Q_i initially has input x_i and is restricted to making ε-noisy broadcasts of x_i. An auxiliary processor P_λ may broadcast any boolean function of the bits it heard previously, and this broadcast is noise-free, i.e., is received by every other processor with no error.

In the *Noisy-copy broadcast model* $NCB(\varepsilon)$, there is a receiver P_0 and an indexed collection $\{P_\lambda : \lambda \in \Lambda\}$ of processors. Each of the P_i initially gets an (independent) ε-noisy copy of the *entire* input. All broadcasts are noise-free.

Starting from $\mathcal{P} =: \mathcal{P}_1$, we construct a sequence of protocols:

- $\mathcal{P}_2$ in the model $SNB(\varepsilon)$ takes input in $\{0,1\}^n$,
- $\mathcal{P}_3$ in the model $SNB(\varepsilon)$ takes input in $\{0,1\}^{[n]-K}$.
- $\mathcal{P}_4$ runs in the model $NCB(\varepsilon^{\alpha k})$ and takes input in $\{0,1\}^{[n]-K}$.
- T is a gnd-tree that takes input in $\{0,1\}^{[n]-K}$.

As stated, there are k broadcasts in $\mathcal{P}$. Using the notation from Section 2.3, we have for $1 \le j \le k$,

- i^j is the index of the processor broadcasting at step j
- g^j is the boolean function (depending on x_j and the bits received by P_j prior to step j) that determines broadcast j.
- b^j is the bit broadcast at step j (obtained by evaluating g^j)
- For $h \in [n]$, b^j_h is the noisy copy of b^j received by P_h.

We also introduce the notation g^j_0 and g^j_1 for the functions obtained from g^j by setting x_j to 0 and to 1, respectively.

From $\mathcal{P}_1$ to $\mathcal{P}_2$. Given $\mathcal{P}_1$, the protocol $\mathcal{P}_2$ in the model $SNB(\varepsilon)$ will consist of k stages. Stage j simulates broadcast j of $\mathcal{P}_1$ and consists of three broadcasts.

The index set Λ for auxiliary processors is $[n]$. In $\mathcal{P}_2$, we will maintain the invariant that after each stage j:

> For $0 \le i \le n$, P_i has constructed a sequence $b^i_1, \ldots, b^i_j$ of bits and the collection $(b^i_h : 0 \le i \le n, 1 \le h \le j\}$ has the same joint probability distribution as it does in $\mathcal{P}_1$.

P_0 will compute its output exactly as in $\mathcal{P}_1$ and the invariant insures that the outputs of the two protocols have the same distribution.

Assume that the invariant holds inductively after stage $j-1$. We define stage j of $\mathcal{P}_2$ so as to maintain the invariant. In stage j, Q_{i^j} broadcasts x_{i^j} (which is all it can do in this model) and P_{i^j} evaluates both g^j_0 and g^j_1 at $b^{i^j}_1, \ldots, b^{i^j}_{j-1}$ and broadcasts the two values.

Each P_i must generate b^j_i. P_i receives 3 bits (x, a_0, a_1), where x is a noisy copy of x_{i^j} broadcast by Q_{i^j}, and a_0 and a_1 are the exact bits sent by P_{i^j}. If $a_0 \ne a_1$, P_i sets b^i_j to a_x, and if $a_0 = a_1$, P_i uses its private randomness to generate an ε-noise bit N, and sets b^i_j to $a_0 \oplus N$. It is easy to check that this choice maintains the invariant.

The total number of broadcasts in $\mathcal{P}_2$ is $3k$, of which k are by input processors and $2k$ are by auxiliary processors.

From $\mathcal{P}_2$ to $\mathcal{P}_3$. We now construct a protocol $\mathcal{P}_3$ for the restricted function $f\lceil_\rho$ in the model $SNB(\varepsilon)$. For this function, the input (and the Q_i) are indexed by $[n] - K$. The index set Λ of auxiliary processors is chosen to be $[n]$.

$\mathcal{P}_3$ simulates $\mathcal{P}_2$ in a step by step fashion. The only difficulty is that the processors Q_i for $i \in K$ don't exist when running $\mathcal{P}_3$. In $\mathcal{P}_2$, processor Q_i would send its input bit, which is now fixed to ρ_i. In $\mathcal{P}_3$, the value ρ_i can be "hardwired" and each processor simply simulates the reception of ρ_i as $\rho_i \oplus N$ where N is an ε-noise bit that it generates locally. Every other step is done exactly as in $\mathcal{P}_2$.

The total number of broadcasts in $\mathcal{P}_3$ is $3k$, of which at most k are by input processors and the remaining are by other processors. Furthermore no input processor sends its bit more than αk times.

From $\mathcal{P}_3$ to $\mathcal{P}_4$. We now construct a protocol $\mathcal{P}_4$ in the $NCB(\gamma)$ model for $\gamma = \varepsilon^{\alpha k}$, that simulates $\mathcal{P}_3$.

In $\mathcal{P}_3$ each input bit x_i is broadcast at most αk times, so each P_j receives at most αk noisy copies of x_i. In $\mathcal{P}_4$, P_j starts with a γ-noisy copy of x_i. It suffices to show αk independent ϵ-noisy copies of x_i can be generated from a single γ-noisy copy of x_i.

Lemma 36. *Let t be an arbitrary integer, $\varepsilon \in (0, 1/2)$ and $\gamma = \varepsilon^t$. There is a randomized algorithm that takes as input a single bit b and outputs a sequence of t bits and has the property that if the input is a γ-noisy copy of 0 (resp. of 1) then the output is a sequence of independent ε-noisy copies of* 0 *(resp. of 1).*

Proof. The algorithm is specified by two probability distributions q_0 and q_1 over $\{0,1\}^t$. On input b, the algorithm outputs a string according to the distribution q_b.

If the input b to the algorithm is a γ-noisy copy of 0, resp. 1, then the output will be generated according to the distribution function $r_0 = (1-\gamma)q_0 + \gamma q_1$, resp. $r_1 = (1-\gamma)q_1 + \gamma q_0$.

Let p_0 (resp. p_1) be the distribution on $\{0,1\}^t$ obtained by taking t independent ε-noisy copies of 0 (resp. 1). We want to choose q_0 and q_1 so that $r_0 = p_0$ and $r_1 = p_1$. For each $s \in \{0,1\}^t$, we need:

$$\begin{aligned}(1-\gamma)q_0(s) + \gamma q_1(s) &= p_0(s),\\ (1-\gamma)q_1(s) + \gamma q_0(s) &= p_1(s).\end{aligned}$$

Solving for $q_0(s)$ and $q_1(s)$ we get:

$$\begin{aligned}q_0(s) &= \frac{(1-\gamma)p_0(s) - \gamma p_1(s)}{1-2\gamma},\\ q_1(s) &= \frac{(1-\gamma)p_1(s) - \gamma p_0(s)}{1-2\gamma}.\end{aligned}$$

It suffices to show that q_0 and q_1 are probability distributions, i.e., they are nonnegative and sum to 1. That they sum to 1 follows from the fact that p_0 and p_1 each sum to 1. Nonnegativity follows immediately from the easy fact that for any $s \in \{0,1\}^t$, $p_0(s)/p_1(s) \geq p_0(1^t)/p_1(1^t) \geq \gamma/(1-\gamma)$ and $p_1(s)/p_0(s) \geq p_1(0^t)/p_0(0^t) \geq \gamma/(1-\gamma)$. ∎

From $\mathcal{P}_4$ to T. $\mathcal{P}_4$ is randomized, i.e., each processor uses its own internal source of random bits in the protocol. We construct a random gnd-tree that first simulates the random choices of all the processors in $\mathcal{P}_4$. Once these are fixed, $\mathcal{P}_4$ reduces to a deterministic protocol. So it is enough to show how to simulate a deterministic protocol $\mathcal{Q}$ in the $NCB(\gamma)$ model by a gnd-tree.

Let $\lambda^1, \ldots, \lambda^{2k}$ be the sequence of indices of processors that broadcast in $\mathcal{Q}$ and let $b^1, \ldots, b^{2k}$ be the bits broadcast. The gnd-tree makes $2k$ queries and will be chosen so that the answer to query j is b^j. Query j is made to copy λ^j. Given that the sequence of answers to the first $j-1$ is $b^1, \ldots, b^{j-1}$, the question asked of copy i^j is: "what would processor P_{i^j} broadcast in $\mathcal{Q}$ given that the string of bits received during the first $j-1$ rounds is $b^1, \ldots, b^{j-1}$?" The answer to this is a boolean function depending only on copy i^j. Since all broadcasts in $\mathcal{Q}$ are noise-free, the sequence of answers in the gnd-tree has exactly the same distribution as the sequence of bits broadcast by $\mathcal{Q}$.

This completes the proof of Theorem 6.

7 A protocol for Identity

In this section we give a protocol for computing id with $O(n \log\log n)$ broadcasts. Our protocol is similar to, but simpler than, the protocol of Gallagher mentioned earlier. Gallagher's protocol and ours work in all three models of noise discussed in the introduction.

Gallagher gave an $O(n \log\log n)$ broadcast protocol for PAR, and used that protocol to construct one for id. Our protocol uses constant rate error correcting codes, which are well-known to exist (random codes are good) but non-trivial to construct explicitly. Gallagher's protocol has the advantage of being self-contained and explicit.

Error correcting codes are an efficient way to communicate large blocks of data on a noisy channel. In the noisy broadcast model, each processor only has a single bit to start, so error correcting codes are not directly applicable.

For our protocol, we divide the processors into teams of size $t = \log n$. The processors in each team work together to transmit their bits to the receiver. Processors ignore transmissions by processors in other teams.

For the analysis, we will need to assume $\varepsilon < 1/12$. This is without loss of generality since given a protocol $\mathcal{P}$ that works for some constant noise parameter ε' we can get a protocol that works for larger noise parameter $\varepsilon < 1/2$ by a routine amplification: every broadcast in $\mathcal{P}$ is repeated C times some appropriate constant $C = C(\varepsilon, \varepsilon')$, and each receiver decodes the broadcast by majority vote.

We will describe a protocol for a single team, denoted $T = \{Q_1, \ldots, Q_t\}$. The protocol uses $O(\log t)$ broadcasts. We show that the probability that the receiver fails to recover the team's entire input, is at most $2(4^{-t}) = 2/n^2$. Repeating the same protocol for each team yields a protocol for id that fails with probability less than $2/(n \log n)$.

Let $y = (y_1, \ldots, y_t)$ denote the team's input. The protocol works in two phases.

Phase 1. For each $i \in [t]$, Q_i broadcasts y_i $s = c_1 \log t$ times, for some constant c_1 to be chosen according to Lemma 37. Every other processor tries to recover y_i by taking a majority vote of the s copies it received.

Let y^i be the copy of y recovered by Q_i. Say that Q_i is *successful* if $y^i = y$. Let U denote the set of unsuccessful processors.

Lemma 37. *There exists a constant $c_1 = c_1(\varepsilon)$ such that:*

$$\Pr[|U| \geq \varepsilon t] \leq 4^{-t}].$$

Proof. Processor Q_i decodes y_j inccorrectly if at least $1/2$ of the copies of y_j that Q_i received were corrupted by noise. By Lemma 12 with each $\alpha_i = 1$ and t (in the lemma) equal to $(1/2-\varepsilon)s$, the probability that Q_i decodes y_j incorrectly is at most $e^{-2(1/2-\varepsilon)^2 s}$. Summing over the t bits of y, $\Pr[Q_i \in U]$ is at most $q = te^{-2(1/2-\varepsilon)^2 c_1 \log t}$.

For $S \subseteq T$, $\Pr[U = S] \leq q^{|S|}$. Summing over subsets S of size at least εt, we obtain:

$$\Pr[|U| \geq \varepsilon t] \leq q^{\varepsilon t} 2^t = (2q^{\varepsilon})^t.$$

We want that this is at most 4^{-t}, so it suffices that $q < (1/8)^{1/\varepsilon}$. It is now easy to choose c_1 sufficiently large (depending on ε) so that this last condition holds. □

Phase 2. In this phase the processors of T work together to transmit an encoding of y based on an error correcting code. For boolean strings v, w of the same length, let $d(v, w)$ denote the number of coordinates where they differ. The following well known result states what we need about the existence of good codes:

Lemma 38. *For $\gamma \in (0, 1/2)$ there is an integer $K_1 = K_1(\gamma)$ such that for each positive integer t and each $K \geq K_1$, there is a subset C_t of $\{0,1\}^{Kt}$ having size 2^t, such that for all $v, w \in C_t$ with $v \neq w$, $d(v, w) \geq \gamma Kt$.*

This is easily proved by the probabilistic method [3]. For γ sufficiently small, there are explicit constructions, e.g. Justesen codes (see, e. g., [21]).

Set γ to be 6ε (here is where we need $\varepsilon < 1/12$. Let $K = \max\{K_1(\gamma), 1/\varepsilon^2)$ and let C_t be given by the lemma. Fix a bijection σ from $\{0,1\}^t$ to C_t. Divide $\sigma(y)$ into t blocks of size K and denote the jth block by $\sigma_j(y)$.

In phase 2, Q_i broadcasts $\sigma_i(y_i)$. Let r_i be the noisy version of this heard by the receiver and let r be the concatenation of $r_1, \ldots, r_{\log n}$. The receiver chooses $w \in C_t$ to minimize $d(r, w)$ and outputs $\sigma^{-1}(w)$.

If $d(r, \sigma(y)) < \gamma Kt/2 = 3\varepsilon t$, then by the choice of C_t, $w = \sigma(y)$ and the receiver correctly outputs y.

Since every processor that was successful in phase 1 broadcasts $\sigma(y)$, r differs from $\sigma(y)$ only on coordinates sent by unsuccessful processors and coordinates that were flipped due to noise. Let $I \subseteq [Kt]$ be the index set of the coordinates that were flipped by noise. Thus $d(r, \sigma(y) \leq K|U| + |I|$.

By Lemma 37, $\Pr[k|U| \geq K\varepsilon t] \leq 4^{-t}$. By Lemma 12, $\Pr[|I| \geq 2K\varepsilon t] \leq e^{-2\varepsilon^2 Kt}$. which is at most 4^{-t} for $K \geq 1/\varepsilon^2$. Thus the probability that the receiver does not output the input of the team is at most $2 \times 4^{-t} \leq 2/n^2$. Summing over all $n/\log n$ teams, the probability that the receiver fails is at most $2/(n \log n)$.

8 A linear noisy broadcast protocol computing weight

In this section we prove Theorem 4 by giving a protocol for computing $\mathsf{weight}(x)$ which uses a linear number of broadcasts in the noisy broadcast model, and showing a gnd-tree of linear depth that computes it. This immediately implies linear protocols or gnd-trees for PAR, or any other boolean function whose value at x depends only on $\mathsf{weight}(x)$. Our protocol does not work under any of the adversarial noise models.

We begin by giving a gnd-tree T of linear depth for $\mathsf{weight}(x)$, and then convert it to a protocol in the noisy broadcast model.

8.1 A gnd-tree for computing weight

We need some additional notation. For $\theta \in [0, n]$, the threshold function $f_\theta : \{0,1\}^n \rightarrow \{0,1\}$ is defined to be 1 if $\mathsf{weight}(x) \geq \theta$. All queries made by T are threshold functions. We define the real function $\rho : [0, n] \longrightarrow [0, n]$ by $\rho(a) = \varepsilon n + (1 - 2\varepsilon)a$. Observe that an ε-noisy copy of 0 has expectation ε and a ε-noisy copy of 1 has expecation $1 - \varepsilon$. Therefore, for fixed x, the expected weight of a noisy copy of x satisfies:

$$\mathbb{E}[\mathsf{weight}(x \oplus N)] = \rho(\mathsf{weight}(x)).$$

The gnd-tree T works in two phases. During the first phase, T does a modified binary search to identify an interval $[a, b]$ of length at most $O(\sqrt{n}$ that contains $\mathsf{weight}(x)$ with high probability. During the second phase, T repeats the threshold query $f_{(a+b)/2}$ on $O(n)$ different copies and determines $\mathsf{weight}(x)$ from this with high probability.

Phase I. During this phase T does a modified binary search to identify an interval of length $O(\sqrt{n})$ that contains $\mathsf{weight}(x)$ with high probability. Initially the interval is $[a_0, b_0] = [0, n]$. The phase consists of stages. During stage i, the interval is reduced to $2/3$ of its previous length. The interval after stage i is denoted $[a_i, b_i]$. Phase 1 ends when $b_i - a_i \leq c(\varepsilon)\sqrt{n}$, for $c(\varepsilon) = 6/(1 - 2\varepsilon)$. This uses $O(\log n)$ stages.

The algorithm for a stage depends on the interval $[a, b]$ at the beginning of the stage, and is denoted $S_{a,b}$. Let $m = (a + b)/2$ be the midpoint. $S_{a,b}$ consists of $k = 16 \ln n$ threshold queries $f_{\theta(m)}$ to distinct copies of the input. If the number of 0 answers is less than $k/2$ then the interval $[a', b']$ output by $S_{a,b}$ is $[a, \frac{a+2b}{3}]$ and otherwise it is $[\frac{2a+b}{3}, b]$.

We say that stage $S_{a,b}$ *fails* if $\mathsf{weight}(x)$ belongs to $[a, b]$ but not to $[a', b']$.

Lemma 39. *let $c(\varepsilon) = 6/(1 - 2\varepsilon)$. Let $x \in \{0,1\}^n$ and suppose $a, b \in [0, n]$ satisfy $b - a > \sqrt{n}$ and $\mathsf{weight}(x) \in [a, b]$. The probability that $S_{a,b}$ fails is at most $1/n^2$.*

Proof. We divide the analysis according to which of the intervals $[a, \frac{2a+b}{3})$, $[\frac{2a+b}{3}, \frac{a+2b}{3}]$ and $(\frac{a+2b}{3}, b]$ contains $\mathsf{weight}(x)$.

If $\mathsf{weight}(x) \in [\frac{2a+b}{3}, \frac{a+2b}{3}]$ then $\mathsf{weight}(x) \in [a', b']$ for either of the two possible outputs.

We next consider the case $\mathsf{weight}(x) \in [a, \frac{2a+b}{3})$; the analysis in the other case is very similar and is omitted. For such an x, $f_{\rho(m)}(x) = 0$.

Let q denote the probability a single noisy query of $f_{\rho(m)}$ is incorrect. For $i \in [k]$, let Z_i be the random variable that is 1 if the answer to the ith noisy query is incorrectly 1. The stage fails if $\sum_i Z_i \geq k/2$. Using Lemma 12 with $\varepsilon = q$ and all of the $a_i = 1$, the probability that the stage fails is at most:

$$\Pr[\sum(Z_i - q) \geq k(\frac{1}{2} - q)] \leq e^{-2(1/2-q)^2 k} = e^{-32(1/2-q)^2 \ln n}.$$

If $q \leq 1/4$, then this is at most $1/n^2$. So we complete the proof by showing that $q \leq 1/4$.

Let X be a noisy copy of x and N the associated noise vector. For $i \in [n]$, let $a_i = 1 - 2x_i$. $X_i = x_i + a_i N_i = \rho(x_i) + a_i(N_i - \varepsilon)$ and $\sum_i X_i = \rho(\mathsf{weight}(x)) + \sum a_i(N_i - \varepsilon)$. Noting that under the case assumption we have $\rho(m) - \rho(\mathsf{weight}(x)) = (m - \mathsf{weight}(x))(1-2\varepsilon) \geq (b-a)(1-2\varepsilon)/6 \geq c(\varepsilon)\sqrt{n}(1-2\varepsilon)/6 \geq \sqrt{n}$. Lemma 12 implies:

$$\begin{aligned} q = \Pr[\sum_i X_i \geq \rho(m)] &= \Pr[\sum_i a_i(N_i - \varepsilon) \geq \rho(m - \mathsf{weight}(x))] \\ &\leq \Pr[\sum_i a_i(N_i - \varepsilon) \geq \sqrt{n}] \leq e^{-2}. \end{aligned}$$

□

Observe that the total number of queries in phase 1 is $O((\log n)^2)$.

Bias Differences. After successfully completing the first phase T "knows" that the weight of x is in a strip $[a_r, b_r]$ of length roughly $\sqrt{n}$. In the second phase T simply makes a linear number of queries of the form $f_{\rho(\theta)}$ to distinct noisy copies of x, where $\theta = \frac{a_r + b_r}{2}$. The weight of x is estimated based on the number of '1' answers. Intuitively, the larger the $\mathsf{weight}(x)$ is the more 1's we expect to see (note that the distribution on the number of '1' answers depends only on the weight of x). We make this observation more formal in the next definition and lemma.

Definition 40. *Let $\varepsilon \in (0, 1/2)$ be a noise parameter, and let $\theta \in [0, n]$. For any $x \in \{0,1\}^n$ we define*

$$\beta_{n,\varepsilon}(\omega, \theta) = \Pr\left[f_{\rho(\theta)}(x \oplus N) = 1\right],$$

where $\omega = \mathsf{weight}(x)$, and N is an ε-noise vector.

Lemma 41. *There exists a global constant $C > 0$ which satisfies the following. Let $\varepsilon \in (0, 1/2)$ be a noise parameter, and let $c(\varepsilon)$ be as in Lemma 39. Then for n large enough it holds for every $\theta \in [0, n]$ and every $\omega \in [\theta - c(\varepsilon)\sqrt{n}, \theta + c(\varepsilon)\sqrt{n}]$ that*

$$\beta_{n,\varepsilon}(\omega + 1, \theta) - \beta_{n,\varepsilon}(\omega, \theta) \geq \frac{\exp\left(-\frac{C}{\varepsilon(1-2\varepsilon)^4}\right)}{C\sqrt{n}} \tag{28}$$

Lemma 41 follows in a straightforward way by writing $\beta_{n,\varepsilon}(\omega + 1, \theta) - \beta_{n,\varepsilon}(\omega, \theta)$ as an expression involving binomial coefficients and powers of ε and of $(1 - \varepsilon)$, and then applying simple estimations using Stirling's formula. We omit the full derivation.

Phase II. For brevity, let us denote the expression in the right hand side of (28) by $\Delta(n, \epsilon)$. In the second phase, T applies the threshold tree $T_{k,\theta}$ to x, where $\theta = \frac{a_r+b_r}{2}$ and $k = 17/\Delta^2(n, \epsilon)$. That is, T queries $f_{\rho(\theta)}$ on k distinct noisy copies of x (note that k is linear in n, and thus this only adds a linear depth to T).

To determine the output of T, let Z denote the number of queries of the threshold tree for which the result was 1. Also, for every $\omega \in \{0, 1, \ldots, n\}$ define a segment I_ω by

$$I_\omega = \left[k \cdot \beta_{n,\epsilon}(\omega, \theta) - 2\sqrt{k}\,,\; k \cdot \beta_{n,\epsilon}(\omega, \theta) + 2\sqrt{k}\right].$$

As we show below, the segments I_ω are all disjoint for $\omega \in [a_r, b_r]$. If $Z \in I_\omega$ for such an ω, T outputs ω. Otherwise T declares failure in computing $\mathsf{weight}(x)$.

Analysis of phase II. As noted above, the depth that the second phase contributes to T is linear in n (the total depth of T is therefore linear, as required). To show that the output is really $\mathsf{weight}(x)$ with high probability, we have to first note that the output is well defined, namely that the segments I_ω are disjoint for all $\omega \in [a_r, b_r]$. Indeed, since each I_ω is a segment of length $4\sqrt{k}$ centered around $k \cdot \beta_{n,\varepsilon}(\omega, \theta)$, the choice of k and Lemma 41 directly imply that these segments are disjoint.

Now suppose $\mathsf{weight}(x) = \omega$ for some $\omega \in [a_r, b_r]$. By definition, each query of the threshold tree applied in the second phase returns the value 1 with probability $\beta_{n,\varepsilon}(\omega, \theta)$, and thus

$$\mathbb{E}[Z] = k \cdot \beta_{n,\varepsilon}(\omega, \theta).$$

Also, since Z is the sum of k independent random variables (the indicators of events of the from "the i'th query being 1") each with variance at most 1, it follows that the variance of Z is at most $\sqrt{k}$. Therefore by Chebyshev's inequality,

$$\Pr[Z \in I_\omega] = \Pr\left[|Z - \mathbb{E}[Z]| \leq 2\sqrt{k}\right] \geq \Pr\left[|Z - \mathbb{E}[Z]| \leq 2\sqrt{\mathbb{V}[Z]}\right] \geq 3/4.$$

It follows that, assuming the first phase succeeds, T outputs $\mathsf{weight}(x)$ with probability at least $3/4$. The overall probability that T computed $\mathsf{weight}(x)$ correctly is therefore at least $2/3$.

8.2 Translating to a noisy broadcast protocol

We have constructed a gnd-tree T of linear depth in n, which computes $\mathsf{weight}(x)$. Let us sketh how T can be transfomed into a protocol in the noisy broadcast network model. As each query that T makes is applied to a distinct ε-noisy copy of the input, we first need to get hold of that many noisy copies of x.

Obtaining noisy copies of x. Let $\varepsilon \in (0, 1/2)$ be the noisy parameter in a noisy broadcast network, and let $x \in \{0, 1\}^n$ be an input. The first step in simulating T is to obtain noisy copies of the input. To achieve $d \cdot n$ noisy copies of x, say, we can make each processor broadcast its own bit d times. After this step (which takes $d \cdot n$ broadcasts) is completed, each processor has d noisy copies of each of the other player's bits. Using its own private randomness, each processor can also obtain d copies of its own input bit, thus obtaining d ϵ-noisy copies of x. In the protocol described below we only need a linear number of noisy copies, and therefore this step only takes a linear number of broadcasts.

Phase I simulation. Once sufficient number of noisy copies of x have been obtained, we can start by simulating the first phase of T. We can pre-assign $\Theta(\log^2 n)$ processors to perform the queries originally applied by T in the first phase. To avoid errors, we can have each processor repeat the result of the query it makes, broadcasting it $\Theta(\log n)$ times, so that with very high probability all the other processors will get the result of the query correctly. At the end of the first phase simulation we have that with probability at least $1 - 1/\mathrm{poly}(n)$ all processors agree on the same segment $[a_r, b_r]$, the same as would have been acheived by T using the same noisy copies.

Since T makes $\Theta(\log^2 n)$ queries in the first phase, and since we simulate each query by $\Theta(\log n)$ broadcasts, the simulation of the first phase requires $\Theta(\log^3 n)$ broadcasts.

Phase II simulation. The second phase in T consists of counting the number of 1s obtained in a series of threshold queries. In the simulation of this phase the receiver, P_0, will do the counting and output the result accordingly. The actual threshold queries will be applied by the other processors: each of them, in its turn, will apply $f_{\rho(\theta)}$ queries to the noisy copies under its possession, and broadcast the results.

A slight problem arises from the fact that P_0 may receive the wrong result for some of the queries due to noise. If $\mathsf{weight}(x) = \omega$, the probability that P_0 will receive the answer 1 for a query of type $f_{\rho(\theta)}$ is not $\beta_{n,\varepsilon}(\omega, \theta)$ as before, but rather it is

$$\beta'_{n,\varepsilon}(\omega, \theta) = \varepsilon + (1 - 2\varepsilon)\beta_{n,\varepsilon}(\omega, \theta).$$

But it is obvious that Lemma 41 still holds for $\beta'_{n,\epsilon}(\omega, \theta)$, perhaps with a different constant C, and thus the simulation of the second phase can still be carried out.

9 Open Problems

The main questions left open by this paper concern lower bounds for decision functions.

1. Can every decision function be computed by a linear noisy broadcast protocol if constant error is allowed? The same question can be asked for gnd-trees, and we believe these two questions should have the same answer, and that the answer should be negative. A random function would seem to be a natural candidate for a hard function.

2. What other interesting classes of decision functions besides symmetric functions have linear noisy broadcast protocols and/or gnd-trees?

3. Are there any functions that can be computed by a gnd-tree whose depth is significantly smaller than their randomized decision-tree complexity, namely the depth required to compute them by a (noiseless) randomized decision tree?

4. The same questions as above can be asked about the adverserial noise model of [11]. For this model, the only nontrivial upper bound known for both the noisy broadcast and gnd-tree models is the protocol for OR due to [23], which is linear in both models. What other non-trivial functions have linear protocols? Does the parity function have linear cost protocols/gnd-trees? We conjecture that the answer to the latter question is negative.

5. In the adversarial noise model, we don't know of any examples of decision functions where gnd-trees do better than ordinary noisy decision trees. Are these models equivalent for adversarial noise?

References

[1] M. Ajtai. The invasiveness of off-line memory checking. *STOC* 2002, 504–513.

[2] D. Aharonov, M. Ben-Or. Fault Tolerant Quantum Computation with Constant Error. *STOC* 1997, 176–188.

[3] N. Alon, J. Spencer. The Probabilistic Method. Second Edition. *Wiley*, 2000.

[4] Y. Aumann and M. A. Bender. Fault Tolerant Data Structures. *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*, 580–589, 1996.

[5] T. Cover, J. Thomas. Elements of Information Theory. *Wiley*, 1991

[6] A. El Gamal. Open problems presented at the 1984 workshop on Specific Problems in Communication and Computation sponsored by Bell Communication Research. Appeared in Thomas M. Cover and B. Gopinath, editors. *Open Problems in Communication and Computation*, 1987. Springer-Verlag.

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

[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), May 1998, 1299–1305.

[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).

[Fel72] W. Feller. *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons, 1972.

[13] I. Finocchi, G. F. Italiano. Sorting and searching in the presence of memory faults (without redundancy). *STOC* 2004.

[14] P. Gács. Reliable Cellular Automata with Self-Organization. *FOCS* 1997, 90–99.

[15] 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, 1994, pp. 579–583.

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

[17] A. Kalai, R. Servedio. Boosting in the Presence of Noise. *35th Annual Symposium on Theory of Computing (STOC)*, 2003, pp. 196–205.

[18] 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)

[19] E. Kushilevitz, Y. Mansour. Computation in Noisy Radio Networks. *SODA* 1998: 236–243.

[20] 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).

[21] J. H. van Lint. Introduction to Coding Theory. Third Edition. *Springer-Verlag* 1999.

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

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

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

[25] N. Pippenger. On the Networks of Noisy Gates. *FOCS* 1985, 30–36.

[26] S. Rajagopalan, L. J. Schulman. A coding theorem for distributed computing. *STOC* 1994, 790–799.

[27] R. Reischuk, B. Schmeltz. Reliable Computation with Noisy Circuits and Decision Trees—A General $n \log n$ Lower Bound. FOCS 1991: 602–611.

[28] A. Russell, M. Saks, D. Zuckerman. Lower Bounds for Leader Election and Collective Coin-Flipping in the Perfect Information Model. *SIAM Journal on Computing*, 31(6):1645–1662, 2003.

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

[30] D. A. Spielman. Highly Fault-Tolerant Parallel Computation. *FOCS* 1996, 154–163.

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

[32] M. Talagrand. On Russo's approximate zero-one law. *Ann. Probab.* 22 (1994), no. 3, 1576–1587.

[33] M. Talagrand. How much are increasing sets positively correlated? *Combinatorica 16* (1996), no. 2, 243–258.

[34] A. Yao. Probabilistic computations: Towards a unified measure of complexity, *In Proceedings of the Seventeenth IEEE Conference on Foundations of Computer Science*, 1977, pages 222–227.

[35] A. Yao. "On the Complexity of Communication under Noise". Invited talk in the *5th ISTCS conference*, 1997.