Lower Bounds for the Noisy Broadcast Problem

Extended Abstract

Navin Goyal *
Dept. of Computer Science
Rutgers University

Guy Kindler†
Institute for Advanced Study
Princeton.

Michael Saks‡
Dept. of Mathematics
Rutgers University

Abstract

We prove the first non-trivial (superlinear) lower bound in the noisy broadcast model of distributed computation. In this model, there are $n+1$ processors $P_0, P_1, \ldots, P_n$. Each P_i, for $i \geq 1$, initially has a private bit x_i and the goal is for P_0 to learn $f(x_1, \ldots, x_n)$ for some specified function f. At each time step, a designated processor broadcasts some function of its private bit and the bits it has heard so far. Each broadcast is received by the other processors but each reception may be corrupted by noise.

In this model, Gallager [16] gave a noise-resistant protocol that allows P_0 to learn the entire input in $O(n \log\log n)$ broadcasts. We prove that Gallager's protocol is optimal up to a constant factor.

Our lower bound follows from a lower bound in a new model, the generalized noisy decision tree model, which may be of independent interest.

1 Introduction

Coping with noise is a major theme in the theory and practice of information systems, and gives rise to the following question: how much additional resources are needed to obtain reliable results in the presence of noise?. This question was studied in the context of decision trees [12, 26, 10, 8, 22], formulas and circuits [24, 15, 29, 18, 9], sorting networks [20], cellular automata [14], quantum computation [2], data structures [13, 4], various communication models [16, 28, 19, 27, 22] and other models [1, 7, 23, 17],

The *noisy broadcast model* is a simple model of distributed computation defined by El Gamal [6], and popularized by Yao [34]. There are $n+1$ processors, consisting of n broadcasters, $P_1, \ldots, P_n$, and a receiver P_0. Initially, each broadcaster P_i has a private input bit x_i, and the goal is for P_0 to evaluate a given function $f(x_1, \ldots, x_n)$. Communication is carried out via noisy broadcasts in synchronous time steps; in each time step a designated broadcaster broadcasts a single bit (as a function of its private input and bits it has heard in previous time steps). Each other processor receives a noisy copy of the broadcast, which means that the received bit is complemented with some fixed probability $\varepsilon < 1/2$, independently for each processor. At the end of the protocol, P_0 computes the output from the bits it has heard. The requirement for a correct protocol is that for each $x \in \{0,1\}^n$, P_0 correctly outputs $f(x)$ with some fixed probability, say, 2/3.

In this paper we study the case where the function f to be computed is the identity function, denoted ID . Namely we require that P_0 learns the input bits of all processors (this is of course the hardest function to compute in this model). A naive protocol achieves this with $O(n \log n)$ broadcasts by having each processor broadcast its bit $c \log n$ times for a sufficiently large constant c. This was improved to $O(n \log\log n)$ broadcasts by Gallager [16] in 1988. This upper bound was not improved since then, and yet no lower-bound was previously known other than the trivial $\Omega(n)$. We show that Gallager's upper bound is best possible.

1.1 Other previous results

Kushilevitz and Mansour [19] showed that the majority function, MAJ, can be computed with $O(n)$ 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 obtain results that are less sensitive to assumptions about the 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 [22] improved this to $O(n)$ broadcasts (although Newman only proved his result for a weaker adversarial model, his result easily carries over to the stronger adversarial model of [11].)

In closely related models, efficient error resilient protocols were given for the noisy two party communication complexity by Schulman [28], and for noisy communication networks with small degree by Rajagopalan and Schulman [25].

*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.

1.2 Our results

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

Theorem 1 *For each $\varepsilon \in (0, 1/2)$ there is a number $n_0 = n_0(\varepsilon)$ such that for any $n \geq n_0$, any protocol in the noisy broadcast model that computes ID_n against noise probability ε requires at least $\frac{1}{20\ln(1/\varepsilon)} n \ln\ln n$ broadcasts.*

Our result is the first non-trivial lower bound in the noisy broadcast model. Because we prove our lower bound in the original noise model of El Gamal, our results apply to the stronger adverserial noise models as well.

This lower bound is deduced from a lower bound in a new centralized model of noisy function computation which we call the *generalized noisy decision tree model* (gnd-tree model), which seems of potential independent interest. In this model, the algorithm seeks to evaluate a given function on boolean input x having no direct access to x, but having access to a collection of independent noisy copies of x, where each bit is flipped with some fixed probability $\varepsilon < 1/2$. At each step, based on the information it has gathered so far, the algorithm selects one of the copies and chooses an arbitrary function $q : \{0,1\}^n \longrightarrow \{0,1\}$. It then is told the value of q on the selected copy. (A precise definition of the model appears in Section 2.)

The restriction of the gnd-tree model in which the queries q are required to be of the form $q(y_1, \ldots, y_n) = y_i$ for some i is equivalent to the noisy decision tree (nd-tree) model introduced by Feige et al. [12]. The 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 by a gnd-tree with $O(n)$ queries, by adapting the aforementioned protocol of Kushilevitz and Mansour [19] in the noisy broadcast model.

The function ID can be computed by a gnd-tree(indeed, by an nd-tree) with $O(n \log n)$ queries (by querying $O(\log n)$ noisy copies of each variable). Our main technical result says that no gnd-tree can do better than this:

Theorem 2 *For $\varepsilon \in (0, 1/3)$, there is a number $n_0 = n_0(\varepsilon)$ such that for any $n \geq n_0$, any (randomized) gnd-tree that computes ID for noise probability ε, must have depth at least $\frac{\varepsilon^4}{1800} n \ln(n/3)$.*

The requirement that $\varepsilon < 1/3$ is made for technical convenience. It is not hard to extend the result up to $\varepsilon < 1/2$ but we omit this detail.

1.3 Overview of the proof

Theorem 1 is derived from Theorem 2 via a reduction (Theorem 3) that shows that protocols for computing ID in the noisy broadcast model provide gnd-trees for ID . The advantage of working through the gnd-tree model is that, as a centralized model, it is easier to quantify the knowledge that the algorithm accumulates about the input. By contrast, in the noisy broadcast model, different processors have different views of the input, which makes it more difficult to define a progress measure for a protocol.

The gnd-tree lower bound. Let T be a gnd-tree. Assume that the input x is uniformly sampled from $\{0,1\}^n$. All information gained up to some point in an execution is represented by the current node v, which determines a conditional distribution over the inputs (via Bayes' rule) that can be thought of as the distribution over inputs as currently "perceived by the tree".

It is easy to verify that upon reaching a leaf π, the best strategy for minimizing the probability of error is to output the input y with the maximal perceived likelihood. Moreover, if on path π the tree outputs y, then the probability that it is correct is exactly the perceived probability of y. To obtain the desired lower bound on the depth of a gnd-tree computing ID , it therefore suffices to show that when run on a uniformly chosen input, the expected perceived probability of the input is smaller than $1/3$. In fact, we show that whatever the tree, if its depth is too small then for *every* input x, the expected perceived likelihood of x is less than $1/3$ (note that even if we fix the input x we may still talk about its likelihood as it is perceived by a party who thinks the input is uniformly chosen).

Bounding perceived likelihood gain. Most of the technical difficulty in our result lies in bounding the expected perceived likelihood of the input. An obvious approach would be to consider the gain in the likelihood of the input obtained at each step of the computation, to show that it is small in expectation, and also that the probability of large deviations is small. This does not work with the perceived likelihood itself, however, since its behavior may be erratic at times, growing very slowly for several steps and then making a large leap that builds on the information gathered in earlier steps.

We therefore introduce a new function, denoted L, that can be used to bound the perceived likelihood of the input, and which is easier to handle. The intuition for bounding the expected gain in L at each node, is that one cannot learn a lot of information on the input by making any boolean query to a noisy version of it. For example, no query to a noisy version of an input x can distinguish well between x and any of its neighbors (inputs obtained from x by flipping one coordinate). Once this is formulated precisely, the proof boils down to a bound on linear-degree coefficients in the Fourier reperentation of bounded functions. This bound is in fact a biased-case version of Proposition 2.2 in [32].

Bounding deviations in L. Bounding the expectation of L is not sufficient, as we also need a bound on large deviations. This turns out not to be trivial: the gains in L at each node are not necessarily positive, ruling out first moment methods. Also, since the gains in L in different height-levels of the tree can be correlated, second moment methods (or higher) cannot be directly applied either. We do manage to break L into a sum of random variables L_λ, each measuring the contribution made to L by queries to the noisy copy λ, and to bound the second moment of each of the L_λ variables.

To get a handle on dependencies between the L_λ variables we further break the gain in L at each node into, roughly speaking, the sum of the "expected gain", and the "remainder gain" (which

is the difference between the actual gain in L and the expected gain). It turns out that the remainder gains at different nodes have zero covariance, and that the expected gains are always non-negative. Using these facts together with some specific properties of L we obtain the desired large-deviation bound for it.

1.4 Organization

In Section 2, we give the details of the noisy broadcast and gnd-tree models. We then give a reduction of the latter to the former and, using this reduction, show that Theorem 2 implies Theorem 1. Section 3 contains some preliminaries for the proof of Theorem 2 which is given in Section 4. Some details are omitted due to space considerations and will appear in the final paper.

2 Computation models

In this section we define the noisy broadcast and generalized noisy decision tree models, state a general reduction lemma from the first model to the second and use this lemma to derive Theorem 1 from Theorem 2.

2.1 Noisy bits

Let $\varepsilon \in [0, 1/2]$. An *ε-noisy bit* is a random variable N that is 1 with probability ε and 0 otherwise. If $b \in \{0, 1\}$ an *ε-noisy copy of b* is $b \oplus N$, where N is an ε-noisy bit and $\oplus$ denotes sum modulo 2.

For $k \in \mathbb{N}$, an *ε noisy k-vector* $N(k, \varepsilon)$ is a sequence of k independent ε-noisy bits. The distribution function for $N(k, \varepsilon)$ is denoted $\mu_{k,\varepsilon}$. Thus $\mu_{k,\varepsilon}(\{x\}) = \varepsilon^{|x|}(1-\varepsilon)^{k-|x|}$, where $|x|$ denotes the number of 1's in x. For a k-vector x, an *ε-noisy copy of x* is a random variable of the form $x \oplus N(k, \varepsilon)$; if x is itself randomly generated $N(k, \varepsilon)$ is selected independently of x.

2.2 Noisy computation

We will consider various models for computing a function f on domain $\{0, 1\}^n$, in presence of noise. Here we fix some terminology common to these models. In each model, an algorithm works in discrete steps, and its cost is the number of steps. There is a specified noise parameter $\varepsilon \in [0, 1/2]$ and the execution and output of an algorithm depend on the input x, and an auxiliary ε-noisy vector N whose length depends on the algorithm and the model.

An algorithm is said to compute f against noise ε if for all inputs $x \in \{0, 1\}^n$ and for auxiliary noise vector chosen with parameter ε, the algorithm outputs $f(x)$ with probability at least $2/3$.

2.3 The Noisy Broadcast Model

In this model there are $n + 1$ processors, consisting of one *receiver* P_0 and n *broadcasters* $P_1, \ldots, P_n$. Initially, each broadcaster P_i has a private input bit x_i. The goal is for P_0 to evaluate a specified function f at x.

An s-step protocol A consists of three parts: a sequence $i^1, \ldots, i^s \in [n]^s$ of (not necessarily distinct) broadcaster indices, a sequence $g^1, \ldots, g^s$ of *broadcast functions* where $g^j : \{0, 1\}^j \longrightarrow \{0, 1\}$, and an *output function* h on domain $\{0, 1\}^s$.

The auxiliary ε-noisy vector has length ns and can be thought of as a sequence $N^1, \ldots, N^s$ of independent ε-noisy n-vectors, one for each time step. In an execution of A, at step j broadcaster 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 $b^j_h = b^j \oplus N^j_h$; P_0 receives $b^j_0 = b^j \oplus N^j_{i^j}$ and P_{i^j} "receives" $b^j_j = b^j$. The bit b^j broadcast by P_{i^j} at step j is g^j evaluated at the j-vector consisting 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.

Our version of the noisy broadcast model is similar to that of Gallager [16]. Other variants of this model have been proposed. For example, there is no receiver P_0, and the goal of the computation is for all of the broadcasters 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*: 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 General Noisy Decision Tree Model

This is a centralized model in which the 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 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 Reduction between the noisy broadcast and gnd-tree models

To state the relation between the noisy broadcast model and the gnd-tree model we need a definition. For a function f over $\{0,1\}^n$ and a set $K \subseteq [n]$, let f/K be the function on $\{0,1\}^{n-|K|}$ obtained by fixing the values of x_i's to 0, where $i \in K$.

Theorem 3 *Let $\alpha > 1$ be a constant. If there is a noisy broadcast protocol $\mathcal{P}$ that $(k, 1-\delta)$ computes f against noise ε, then there is a set $K = K(\mathcal{P}) \subseteq [n]$ with $|K| \leq n/\alpha$ and gnd-tree T with input variables indexed by $[n] - K$ that $(3k, 1-\delta)$-computes f/K against noise $\varepsilon^{\alpha k/n}$.*

This reduction is obtained as follows (the details of the proof will be in the final paper). We show how any noisy broadcast protocol can be simulated reliably in a semi-noisy broadcast model in which processors can make two kinds of broadcasts: either a noisy broadcast of its own bit or a noise-free broadcast of a bit that depends only on the bits it has heard previously but not on its own bit. We then show that in the semi-noisy broadcast model, a protocol can be modified so that it has two distinct phases: in the first phase the only broadcasts are noisy broadcasts of processors input bits and in the second phase the only broadcasts are noise-free broadcasts depending on what the processor has previously received. Choosing K to be the set of processors that broadcast more than $\alpha k/n$ times in the first phase and fixing the bits of those processors to 0, we show that the situation after the first phase can be simulated by giving each processor an $\varepsilon^{\alpha k/n}$-noisy copy of the remaining input bits, and the second (non-noisy) phase can be simulated by a gnd-tree that queries these noisy copies.

Using this reduction we deduce Theorem 1 from Theorem 2.

Proof. (of Theorem 1 from Theorem 2)

Suppose there is a k-step noisy broadcast protocol that computes ID_n against noise ε. By Theorem 3 we have

a gnd-tree of depth $3k$ that computes $\mathsf{ID}_{\lceil(1-1/\alpha)n\rceil}$ against noise $\varepsilon^{\alpha k/n}$. Let $r = k/n$ By Theorem 2 for sufficiently large n,

$$\begin{aligned} 3k &\geq \frac{1}{900}(\varepsilon^{\alpha r})^4(1-1/\alpha)n\ln((1-1/\alpha)n/3), \\ 3r &\geq \frac{1}{1200}\varepsilon^{16r}\ln(n/4) \quad \text{(choosing } \alpha = 4\text{)}, \\ 3r(\frac{1}{\varepsilon})^{16r} &\geq \frac{1}{1200}\ln(n/4), \end{aligned}$$

Taking logs of both sides it can be seen that for n sufficiently large (depending on ε),

$$r \geq \frac{1}{20\ln(1/\varepsilon)}\ln\ln n.$$

This completes the proof of Theorem 1. □

3 Preliminaries to the proof of Theorem 2

Some notation. We use $\log$ to mean logarithm in base 2. The input with i'th coordinate 1 and other coordinates 0 is denoted $\mathbf{e_i}$ and the input with all coordinates 0 is denoted $\mathbf{0}$.

We will need the following technical bounds on $\ln(1+x)$, whose routine proof we omit from this version.

Lemma 4 *Let $d \in (0,1)$. For $x \geq d-1$:*

1. $|\ln(1+x)| \leq \frac{\ln(1/d)}{1-d}|x|$.
2. $\ln(1+x) = x + \zeta(x)x^2$, *where* $|\zeta(x)| \leq 2/d$.

We will use some basic notions from information theory. In the expressions below, terms of the form $0\log\frac{1}{0}$ or $0\log\frac{0}{0}$ are assumed to have value 0.

Let X be a random variable taking values in a finite set A. The *binary entropy* of X is given by:

$$\mathsf{H}(X) := \sum_{x\in A}\Pr[X=x]\log\frac{1}{\Pr[X=x]}. \tag{1}$$

For another random variable Y, taking values in a finite set B, with a joint distribution with X, the *conditional entropy* of X given Y, is given by:

$$\mathsf{H}(X|Y) := \sum_{y\in B}\Pr[Y=y]\,\mathsf{H}(X|Y=y). \tag{2}$$

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

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

We will use the following well-known facts (see, e.g., [5]).

Fact 5 *For X, p, q, and A as above, we have:*

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

$$\mathsf{H}(X) \leq \log|A|.$$

Bounding the power of a single noisy query. We present a result that says intuitively that no single query made to a ε-noisy copy of the input vector can be very effective in distinguishing any fixed input x from its neighbors. For simplicity, the lemma is stated for the case $x = \mathbf{0}$, but the obvious generalization holds by symmetry.

For a boolean function $f : \{0,1\}^n \to \{0,1\}$, we define the following associated probabilities.

$$p(0) = p(0)(f) := \Pr_{N\sim\mu_\varepsilon}[f(\mathbf{0}+N)=1],$$

and for $i \in [n]$,

$$p(i) = p(i)(f) := \Pr_{N \sim \mu_\varepsilon}[f(\mathbf{e_i} + N) = 1].$$

The quantity $(p(i) - p(0))^2$ is a measure of how well a noisy query of f distinguishes the input $\mathbf{0}$ from $\mathbf{e_i}$.

Theorem 6 *For all $f : \{0,1\}^n \to \{0,1\}$,*

$$\sum_{i \in [n]} (p(i) - p(0))^2 \leq \frac{8}{\varepsilon^2} p(0)^2 \ln(1/p(0)). \tag{3}$$

(we define $p(0)^2 \ln(1/p(0))$ to be 0 for $p(0) = 0$ in the above inequality.)

The proof of Theorem 6 goes by relating the sum on the left-hand-side of (3) to the some of weights of f on its linear biased-Fourier coefficients. A bound on this sum can then be obtained using a technique from [32]. The details of the proof will appear in the final paper.

4 Proof of Theorem 2

In this section we begin the proof of Theorem 2. We are given a (possibly randomized) gnd-tree T of depth d that is supposed to compute ID_n. We want to prove that if d is too small then there is an $x \in \{0,1\}^n$ such that on input x the probability that T outputs x is small. We make use of a common technique for lower bound proofs for nonuniform models, namely rather than reason about an arbitrary input x, we prove the stronger result that for input x chosen uniformly at random, T fails to output x high probability. It is well known that it suffices to prove such a result for deterministic trees, so for the entire section, we assume that T is a deterministic gnd-tree.

Notation concerning T. The vertex set of T is denoted V and the leaf set of T is denoted leaves.

- $v^\uparrow$ is the set of ancestors of v, i.e. those nodes other than v that belong to the path from the root to v.
- $v^\downarrow$ is the set of descendents of v, i.e., those nodes other than v that belong to the subtree rooted at v.
- $[u, v)$, for $v \in u^\downarrow$, is the set of nodes on the path in T from node u to node v, including u but not v. Similarly, (u, v) is the set of vertices on the path from u to v excluding both u and v.
- $\lambda(v)$ is the noisy copy that labels node v.
- For any subset W of V we write W_λ for the subset of vertices of W whose noisy copy is λ. For example $v_\lambda^\uparrow$ is the set of nodes on the path to v that query copy λ.

The probability space and some notation. The probability space over which we are working consists of a uniformly chosen input (denoted X) and independent noise vectors $(N^\lambda : \lambda \in \Lambda)$ where Λ is the index set of the noisy copies and each $N^\lambda \in \{0,1\}^n$. We denote this distribution by R and for $x \in \{0,1\}^n$ R_x denotes the distribution R conditioned on $X = x$. Note that in a considerable protion of this paper we are working over the distribution $R_{\mathbf{0}}$.

The random variables defined above completely determine the execution of the tree and specify a root-to-leaf path. The leaf arrived at is a random variable denoted Π. We define the following events:

- success is the event that the output of the algorithm (determined by Π) is X.
- $\mathsf{vis}(v)$ is the event that the node v is visited.

The letter x is used exclusively to denote a generic element of $\{0,1\}^n$ and the letter π is used exclusively to denote a generic leaf of the tree. With this convention, we will often abbreviate the event $X = x$ as simply "x" and the event $\Pi = \pi$ as simply "π". Thus, for example, $\Pr[x]$ means $\Pr[X = x]$ and $\Pr[x|\pi]$ means $\Pr[X = x | \Pi = \pi]$.

The progress function. As with many lower bound proofs, we will define a function on computation states (in this case, ordered pairs (v, x) where x is the input and v is a vertex) that measures the progress of the computation towards its goal. We then show that (1) for T to succeed it must be the case that the value of the progress measure at the final leaf is large, and (2) If T is too shallow then for most leaves the value of the progress measure is small. For vertex v and input x we define:

$$\begin{aligned} L^i(v,x) &= \log \frac{\Pr[\mathsf{vis}(v)|x]}{\Pr[\mathsf{vis}(v)|x \oplus \mathbf{e_i}]} \\ &= \log \frac{\Pr[\mathsf{vis}(v) \wedge x]}{\Pr[\mathsf{vis}(v) \wedge (x \oplus \mathbf{e_i})]}, \\ L(v,x) &= \frac{1}{n} \sum_{i=1}^n L^i(v,x). \end{aligned}$$

A large value of $L^i(v,x)$ indicates that the algorithm is more likely to reach v on input x than on input $x \oplus \mathbf{e_i}$. Hence arriving at v means that it is more likely that the input is x than $x \oplus \mathbf{e_i}$. A large value of $L(v,x)$ indicates that arriving at v makes it more likely that the input is x rather than one of x's neighbors. Intuitively, a successful algorithm on input x should tend to end at a leaf π for which $L(\pi, x)$ is large. This intuition is made precise by:

Lemma 7 *Let $\theta > 1$ and $\delta \in (0,1)$. Suppose that $\Pr[L(\Pi, x) > \ln(n/\theta) | X = x] < \delta$ for all $x \in \{0,1\}^n$. Then $\Pr[\mathsf{success}] \leq \delta + \frac{1}{\theta}$.*

Proof. We first show that for any leaf π and input x,

$$\Pr[\pi \wedge x] \leq \Pr[\pi] \frac{e^{L(\pi,x)}}{n}. \tag{4}$$

Rearranging and taking logs, it suffices to show:

$$L(\pi, x) \geq \log(\Pr[\pi \wedge x]) - \log(\frac{1}{n}\Pr[\pi]). \quad (5)$$

Using the convexity of the function $\log(z)$,

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

as required to prove (5) and (4).

Now let $A(\pi, x)$ denote the condition that $L(\pi, x) < \log(n/\theta)$ and $\bar{A}(\pi, x)$ denote the complementary condition. Trivially,

$$\begin{aligned} \Pr[\mathsf{success}] &\leq \Pr[\bar{A}(\Pi, X)] + \Pr[\mathsf{success} \wedge A(\Pi, X)] \\ &\leq \delta + \Pr[\mathsf{success} \wedge A(\Pi, X)]. \end{aligned}$$

We now bound the second term by $1/\theta$. Let $\mathsf{out}(x)$ denote the set of leaves that output x. Then using (4) we have:

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

This sum is at most $1/\theta$ since each leaf π belongs to one set $\mathsf{out}(x)$ and $\sum_\pi \Pr[\pi] = 1$. □

The main lemma of this section is:

Lemma 8 *For any $x \in \{0,1\}^n$,*

$$\Pr[L(\Pi, x) > \ln(n/\theta) | X = x] < \frac{600}{\varepsilon^4} \cdot \frac{\mathsf{depth}(T)}{n \ln(n/\theta)}.$$

The rest of the paper is dedicated to the proof of Lemma 8. But first let us show that it indeed implies Theorem 2.

Proof of Theorem 2 from Lemma 8. Assume T correctly outputs X with probability at least $2/3$. Then by Lemma 7, there is an input x such that $\Pr[L(\Pi, x) > \ln(n/\theta) | X = x] \geq 1/3$. By Lemma 8, T must have depth at least $\frac{\varepsilon^4}{1800} n \ln(n/\theta)$.

By symmetry it suffices to prove the tail bound for $x = \mathbf{0}$, so we restrict to this case. For simplicity we write $L(\pi)$ for $L(\pi, \mathbf{0})$ and $L^i(\pi)$ for $L^i(\pi, \mathbf{0})$. Also we write L for the random variable $L(\Pi)$. Since we are interested in the behavior of L conditioned on $X = \mathbf{0}$, through the rest of the paper the probabilities we evaluate are with respect to the distribution $R_{\mathbf{0}}$ unless otherwise stated .

4.1 The contribution to knowledge by specific noisy copies

In this section we show how to decompose $L^i(\pi)$ as a sum of variables $L^i_\lambda(\pi)$, for $\lambda \in \Lambda$. Intuitively, $L^i_\lambda(\pi)$ is the contribution of nodes in V_λ to $L^i(\pi)$. We then define $L_\lambda(\pi) = \frac{1}{n}\sum_{i\in[n]} L^i_\lambda(\pi)$; thus L is the sum of the L_λ's. Intuitively, L_λ represents the contribution of vertices in V_λ to the progress of the computation.

Let us first remark that the choice of all of the noisy vectors determines the answers to all of the queries at all of the nodes, even though only those on Π are actually asked. For $x \in \{0,1\}^n$ and $S \subseteq V$ we define:

- $\mathsf{A}(x, S)$ is the tuple of answers to the queries at nodes in S when the input is x. Note that $\mathsf{A}(x, S)$ is a random variable depending on the noise variables. For $v \in V$, we write $\mathsf{A}(x, v)$ for $\mathsf{A}(x, \{v\})$.
- $\mathsf{A}(x, S) \rightarrow \pi$ is the event that answers to the queries at nodes in S are consistent with the leaf π. More precisely, for all vertices $v \in S \cap \pi^\uparrow$ $\mathsf{A}(x, v)$ is 0 if π is a left descendent of v and 1 if π is the right descendent. By definition $\Pr[\mathsf{A}(x, S) \rightarrow \Pi] = 1$.

Since the noise for different copies is independent, for leaf π in T we have

$$\Pr[\pi | x] = \prod_\lambda \Pr[\mathsf{A}(x, \pi^\uparrow_\lambda) \rightarrow \pi].$$

Hence

$$L^i(\pi) = \sum_\lambda \ln \frac{\Pr[\mathsf{A}(\mathbf{0}, \pi^\uparrow_\lambda) \rightarrow \pi]}{\Pr[\mathsf{A}(\mathbf{e}_i, \pi^\uparrow_\lambda) \rightarrow \pi]}. \quad (6)$$

We can now separate the contributions of different processors to $L^i(\pi)$ and $L(\pi)$ in a natural manner. Let $L^i_\lambda(\pi)$ denote the summand in the above sum, and define $L_\lambda(\pi) := \frac{1}{n}\sum_{i=1}^n L^i_\lambda(\pi)$. Define random variables $L^i_\lambda = L^i_\lambda(\Pi)$ and $L_\lambda = L_\lambda(\Pi)$. Clearly

$$L = \sum_\lambda L_\lambda,$$

and for each i, $L^i = \sum_\lambda L^i_\lambda$.

We now prove bounds on the random variables L_λ's. First we have:

Lemma 9 *For every noisy copy λ, $i \in [n]$, and $v \in V$ we have*

$$|L^i_\lambda(v)| \leq \ln\left(\frac{1-\varepsilon}{\varepsilon}\right).$$

Consequently, the same bound holds for $|L_\lambda(v)|$ and, in particular, for $|L_\lambda| = |L_\lambda(\Pi)|$.

We omit the simple proof of Lemma 9. The next lemma gives an upper bound on $|\mathbb{E}[L_\lambda]|$: Denote by $N_{\hat{\lambda}}$ the set of random variables $\{N_\kappa : \kappa \in \Lambda - \{\lambda\}\}$.

Lemma 10 *For each $\lambda \in \Lambda$,*

$$|\mathbb{E}[L_\lambda]| \leq \frac{16}{\varepsilon^3} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n}. \tag{7}$$

The quantity $\mathsf{H}(\Pi|N_{\hat{\lambda}})$ can be thought of as the contribution of N_λ to the entropy of Π.

Proof. Let $\eta_{\hat{\lambda}}$ denote an assignment to $N_{\hat{\lambda}}$. We will prove:

$$|\mathbb{E}[L_\lambda|N_{\hat{\lambda}} = \eta_{\hat{\lambda}}]| \leq \frac{16}{\varepsilon^3 n} \mathsf{H}(\Pi|N_{\hat{\lambda}} = \eta_{\hat{\lambda}}); \tag{8}$$

the lemma is then obtained by averaging over $\eta_{\hat{\lambda}}$.

Since we have fixed the noisy copies for all copies but λ, the tree T reduces to a gnd-tree $T_{\eta_{\hat{\lambda}}}$ whose leaf set is a subset of the leaf set of T and whose computation is completely determined by the noisy inputs $\eta_{\hat{\lambda}}$. For leaf π in $T_{\eta_{\hat{\lambda}}}$, define function $f_\pi : \{0,1\}^n \to \{0,1\}$ by: $f(y) = 1$ if when $\eta_\lambda = y$ the result of the computation by $T_{\eta_{\hat{\lambda}}}$ is π, and $f(y) = 0$ otherwise. Define $p_\pi(x) := \Pr[f_\pi(x + N_\lambda) = 1]$, and $\delta_\pi(\mathbf{e_i}) := p_\pi(\mathbf{e_i}) - p_\pi(\mathbf{0})$. In this new notation we can rewrite

$$L_\lambda(\pi) = \frac{1}{n} \sum_{i \in [n]} \ln\left(1 + \frac{\delta_\pi(\mathbf{e_i})}{p_\pi(\mathbf{0})}\right).$$

In the sums below the range of π is over all leaves in $T_{\eta_{\hat{\lambda}}}$. We have, $|\mathbb{E}[L_\lambda|N_{\hat{\lambda}} = \eta_{\hat{\lambda}}]|$is equal to

$$\left|\sum_\pi p_\pi(\mathbf{0}) L_\lambda(\pi)\right| = \frac{1}{n}\left|\sum_{\pi, i \in [n]} p_\pi(\mathbf{0}) \ln\left(1 + \frac{\delta_\pi(\mathbf{e_i})}{p_\pi(\mathbf{0})}\right)\right|$$

$$\leq \frac{1}{n}\left|\sum_{\pi, i \in [n]} \delta_\pi(\mathbf{e_i})\right| + \frac{2(1-\varepsilon)}{n\varepsilon}\left|\sum_{\pi, i \in [n]} p_\pi(\mathbf{0}) \left(\frac{\delta_\pi(\mathbf{e_i})}{p_\pi(\mathbf{0})}\right)^2\right|$$

(using Lemma 4 and the fact that $1 + \frac{\delta_\pi(\mathbf{e_i})}{p_\pi(\mathbf{0})} \geq \frac{\varepsilon}{1-\varepsilon}$)

$$\leq \frac{1}{n}\left|\sum_{i \in [n]} \left(\sum_\pi \delta_\pi(\mathbf{e_i})\right)\right| + \frac{16}{n\varepsilon^3} \sum_\pi p_\pi(\mathbf{0}) \ln \frac{1}{p_\pi(\mathbf{0})},$$

using Theorem 6.

Each inner sum in the first part is 0 as $\sum_\pi p_\pi(\mathbf{0}) = \sum_\pi p_\pi(\mathbf{e_i})$ for all $i \in [n]$; the sum in the second part is the entropy (except that the logarithm is natural not binary) of the density function given by $p_\pi(\mathbf{0})$. Changing the logarithms to binary we get

$$\frac{16}{n\varepsilon^3} \sum_\pi p_\pi(\mathbf{0}) \ln \frac{1}{p_\pi(\mathbf{0})} \leq \frac{16 \ln 2}{\varepsilon^3} \cdot \frac{\mathsf{H}(\Pi_{\eta_{\hat{\lambda}}})}{n} \leq \frac{16}{\varepsilon^3 n} \cdot \mathsf{H}(\Pi_{\eta_{\hat{\lambda}}}).$$

This completes the proof of (8), and hence of Lemma 10. □

The next lemma states a similar bound on $\mathbb{E}[L_\lambda^2]$.

Lemma 11 *Under the same notations as in Lemma 10, it holds for each $\lambda \in \Lambda$ that*

$$\mathbb{E}[L_\lambda^2] \leq \frac{32}{\varepsilon^4} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n}. \tag{9}$$

The proof of Lemma 11 is similar to that of Lemma 10, and will appear in the final paper.

4.2 Overview of the rest of the proof

Our goal at this point is to bound from above the probability that the random variable L is large. The upper bound on $\mathbb{E}[L_\lambda]$ obtained above gives an upper bound on $\mathbb{E}[L] = \sum_\lambda \mathbb{E}[L_\lambda]$. This upper bound is small enough so that if L were nonnegative valued then Markov's inequality, which says that for nonnegative random variable V $\Pr[V \geq T] \leq \mathbb{E}[V]/T$, would give a tail bound on L that is strong enough for our purposes. Unfortunately, L is not always nonnegative.

So we can try instead to get an upper bound on $\mathbb{E}[L^2]$, and use the tail bound $\Pr[L \geq T] \leq \mathbb{E}[L^2]/T^2$. Now, $L^2 = \sum_\lambda L_\lambda^2 + 2\sum_{\lambda \neq \kappa} L_\lambda L_\kappa$. Above, we found a good upper bound for $\mathbb{E}[L_\lambda^2]$. If we could show that $\mathbb{E}[L_\lambda L_\kappa]$ is not much bigger than $\mathbb{E}[L_\lambda]\mathbb{E}[L_\kappa]$ (i.e., the covariance of L_λ and L_κ is small) this again would be good enough to give the bound we want. But we don't know whether this is true.

Since we can't make either of these approaches work, we find a way to combine them. More precisely we show how to decompose each L_λ as a sum of two random variables Y_λ and Z_λ such that: (1) Z_λ is nonnegative valued, (2) $\mathbb{E}[Y_\lambda] = 0$, and therefore $\mathbb{E}[Z_\lambda] = \mathbb{E}[L_\lambda]$, (3) $\mathbb{E}[Y_\lambda^2]$ has an upper bound similar to that on $\mathbb{E}[L_\lambda^2]$, and (4) $\mathrm{Cov}[Y_\lambda, Y_\kappa] = 0$ for all $\lambda \neq \kappa$.

Defining $Z = \sum_\lambda Z_\lambda$ and $Y = \sum_\lambda Y_\lambda$ we have $L = Z + Y$, and we get a tail bound for Z using Markov's inequality and tail bound for Y using Markov's inequality for Y^2; together these give the desired tail bound for L.

The decomposition of L_λ into $Y_\lambda + Z_\lambda$ is accomplished as follows. The random variable L_λ represents the amount of evidence that copy λ provides towards showing that the input is $\mathbf{0}$ conditioned on the input being $\mathbf{0}$. This evidence accumulates the results of all queries to λ. We first decompose L_λ as a sum of random variables L_v. Here v comes from the set of nodes which are on the path taken by the computation and are labeled C, and L_v represents the evidence provided by the query to λ at node v. (The definition of L_v appears in Section 4.4)

Next we decompose L_v into $Y_v + Z_v$. The event that the node v is on the computation path conditions the distribution. Conditioned on this event we consider the expectation of L_v. This conditional expectation represents how much evidence one expects the query at v to provide given the previous answers. We can view this conditional expectation as a random variable which depends only on the outcomes of the queries at the ancestors of v. This random variable is Z_v. We then define $Y_v = L_v - Z_v$ and Y_λ and Z_λ in the obvious way.

The details of this argument are given in Sections 4.3, 4.4 and 4.5.

4.3 The variables L_v

Let us now continue with the proof of Lemma 8.

$$\Pr[\pi|x] = \prod_{v \in \pi^\uparrow} \Pr[\mathsf{A}(x, v) \to \pi \mid \mathsf{A}(x, v^\uparrow) \to \pi]. \tag{10}$$

In the above equation we adopt the convention that the term in the product corresponding to the root is $\Pr[\mathsf{A}(x,v) \to \pi]$. Since $\mathsf{A}(x,v)$ only depends on the answers to the queries to copy $\lambda(v)$ that were posed at the ancestors of v, and not to the other copies, we get

$$(10) = \prod_{v \in \pi^\uparrow} \Pr[\mathsf{A}(x,v) \to \pi \mid \mathsf{A}(x, v^\uparrow \cap V_{\lambda(v)}) \to \pi].$$

We are going to assign fixed weights to the edges in the tree T. To the outgoing edge from vertex v in direction $a \in \{0,1\}$ we will assign $L_v(a)$. This can be thought of as the measure of information the tree gains when it traverses that edge. Using $L_v(a)$ we will then define random variables L_v. Note that

$$\begin{aligned}&\Pr[\mathsf{A}(x, \pi^\uparrow \cap V_\lambda) \to \pi] \\ &\quad = \prod_{v \in \pi^\uparrow \cap V_\lambda} \Pr[\mathsf{A}(x,v) \to \pi \mid \mathsf{A}(x, v^\uparrow \cap V_\lambda) \to \pi].\end{aligned}$$

We can therefore rewrite $L(\pi)$ using (6) as

$$\begin{aligned}L(\pi) &= \sum_{v \in \pi^\uparrow} \frac{1}{n} \sum_{i \in [n]} \ln \frac{\Pr[\mathsf{A}(\mathbf{0},v) \to \pi \mid \mathsf{A}(\mathbf{0}, v^\uparrow) \to \pi]}{\Pr[\mathsf{A}(\mathbf{e}_i,v) \to \pi \mid \mathsf{A}(\mathbf{e}_i, v^\uparrow) \to \pi]} \\ &= \sum_{v \in \pi^\uparrow} \frac{1}{n} \sum_{i \in [n]} \ln \frac{\Pr[\mathsf{A}(\mathbf{0},v) \to \pi \mid \mathsf{A}(\mathbf{0}, v^\uparrow \cap V_{\lambda(v)}) \to \pi]}{\Pr[\mathsf{A}(\mathbf{e}_i,v) \to \pi \mid \mathsf{A}(\mathbf{e}_i, v^\uparrow \cap V_{\lambda(v)}) \to \pi]}\end{aligned} \tag{11}$$

For $a \in \{0,1\}$ and node $v \in V(T)$, define the constants $L_v(a)$ mentioned above

$$L_v(a) := \frac{1}{n} \sum_{i \in [n]} \ln \frac{\Pr[\mathsf{A}(\mathbf{0},v) = a \mid \mathsf{A}(\mathbf{0}, v^\uparrow \cap V_{\lambda(v)}) \to v]}{\Pr[\mathsf{A}(\mathbf{e}_i,v) = a \mid \mathsf{A}(\mathbf{e}_i, v^\uparrow \cap V_{\lambda(v)}) \to v]}. \tag{12}$$

Define random variable $L_v = L_v(\mathsf{A}(\mathbf{0},v))$. Thus L_v is just $L_v(a)$ chosen according to the probability that the answer of the query at v is a. For leaf π and $v \in \pi^\uparrow$, let $L_v(\pi) = L_v(a)$, where $a = 0$ if π is a left descendent of v and 1 otherwise. confusion. Note that with the above notation $L_v(\Pi)$ coincides with $L_v(\mathsf{A}(\mathbf{0},v))$. We have

$$L(\pi) = \sum_{v \in \pi^\uparrow} L_v(\pi), \tag{13}$$

$$L_\lambda(\pi) = \sum_{v \in \pi^\uparrow \cap V_\lambda} L_v(\pi), \tag{14}$$

$$L = \sum_{v \in \Pi^\uparrow} L_v(\Pi). \tag{15}$$

4.4 Breaking the variable L_v into Y_v and Z_v

We break the variables L_v into a sum of new random variables $L_v = Y_v + Z_v$, and then use these variables for the definition of Y_λ and Z_λ. For $v \in V_T$, define

$$\begin{aligned}Z_v &:= \mathbb{E}[L_v \mid v \in \Pi^\uparrow] \\ Z_\lambda &:= \sum_{v \in \Pi^\uparrow \cap V_\lambda} Z_v.\end{aligned}$$

Note that the Z_v's are constants. In words, Z_v is the expectation of L_v conditioned on the event that the execution visits node v. Also, define

$$\begin{aligned}Y_v &:= L_v - Z_v, \\ Y_\lambda &:= \sum_{v \in \Pi^\uparrow \cap V_\lambda} Y_v = L_\lambda - Z_\lambda.\end{aligned}$$

It follows imediately from the definitions that

$$\mathbb{E}[Y_v \mid v \in \Pi^\uparrow] = 0. \tag{16}$$

We can bound the large deviation of L_λ in terms of those of Y_λ and Z_λ:

$$\begin{aligned}&\Pr[\sum_\lambda L_\lambda \geq \ln(n/\theta)] \\ &\leq \Pr[\sum_\lambda Y_\lambda \geq \ln(n/\theta)/2] + \Pr[\sum_\lambda Z_\lambda \geq \ln(n/\theta)/2].\end{aligned} \tag{17}$$

In the next section we complete the proof of Lemma 8, by bounding each of the two terms above in the RHS separately.

Before we can complete the proof of Lemma 8, we need to establish some properties of Z_v's and Y_v's.

Lemma 12 *For all noisy copies λ we have $\mathbb{E}[L_\lambda] = \mathbb{E}[Z_\lambda]$, which implies that $\mathbb{E}[Y_\lambda] = 0$.*

Proof. We have

$$\begin{aligned}\mathbb{E}[L_\lambda] &= \mathbb{E}\Big[\sum_{v \in \Pi^\uparrow \cap V_\lambda} L_v\Big] = \sum_{v \in V_\lambda} \Pr[\mathsf{vis}(v)]\, \mathbb{E}[L_v \mid v \in \Pi^\uparrow] \\ &= \sum_{v \in V_\lambda} \Pr[\mathsf{vis}(v)]\, Z_v = \mathbb{E}\Big[\sum_{v \in \Pi^\uparrow \cap V_\lambda} Z_v\Big] = \mathbb{E}[Z_\lambda].\end{aligned}$$

The second statement is now immediate from the definitions of L_λ, Z_λ, and Y_λ. □

Lemma 13 *For all distinct copies λ and κ we have*

$$\mathbb{E}[Y_\lambda Y_\kappa] = 0. \tag{18}$$

Proof. We have

$$\begin{aligned}\mathbb{E}[Y_\lambda Y_\kappa] &= \mathbb{E}\Big[\Big(\sum_{u \in \Pi^\uparrow \cap V_\lambda} Y_u\Big)\Big(\sum_{v \in \Pi^\uparrow \cap V_\kappa} Y_v\Big)\Big] \\ &= \sum_{u,v : u \in V_\lambda, v \in V_\kappa \cap u^\downarrow} \Pr[\mathsf{vis}(v)] \mathbb{E}[Y_u Y_v \mid \mathsf{vis}(v)] \\ &\quad + \sum_{u,v : v \in V_\lambda, u \in V_\kappa \cap v^\downarrow} \mathsf{vis}(u) \mathbb{E}[Y_u Y_v \mid \mathsf{vis}(u)].\end{aligned} \tag{19}$$

Consider the summand in the first sum in the RHS corresponding to u, v. Conditioned on v being visited, Y_u is fixed to a constant denoted $y_{u,v}$ since v is a descendent of u. Now (16) gives

$$\mathbb{E}[Y_u Y_v | u, v \in \Pi^{\uparrow}] = y_{u,v}\mathbb{E}[Y_v | \mathsf{vis}(v)] = y_{u,v} \cdot 0 = 0.$$

Similarly all the terms in the RHS of (19) are 0 implying $\mathbb{E}[Y_\lambda Y_\kappa] = 0$. □

The following lemma shows what one would intuitively expect: on average, the knowledge of the tree that the input is indeed 0 does not decrease.

Lemma 14 *For nodes $v \in V_T$ we have $Z_v \geq 0$.*

The simple proof goes by writing Z_v as a sum of relative entropies. We omit the details in this version.

Now using the above lemmas and the properties of the L_λ's and L_v's proven in the previous section we can bound from above the second moments of Z_λ and Y_λ, which will in turn yield the desired bound on the RHS of (17).

Lemma 15 *For every noisy copy λ, $\mathbb{E}[Z_\lambda^2] \leq \frac{96}{\varepsilon^4} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n}$.*

Proof. We express Z_λ as a sum of Z_v's and then use the lemmas proved above to obtain the desired bound.

$$\mathbb{E}[Z_\lambda^2] = \mathbb{E}\left[\left(\sum_{v \in \Pi^{\uparrow} \cap V_\lambda} Z_v\right)^2\right]$$
$$= \sum_{v \in V_\lambda} \Pr[\mathsf{vis}(v)] Z_v^2 + \sum_{u,v \in V_\lambda : v \in u^{\downarrow}} 2\ \Pr[\mathsf{vis}(v)] Z_u Z_v. \quad (20)$$

We have, $Z_v^2 \leq \max_{w \in V_T}[|Z_w|]|Z_v|$. By Lemma 9 we know that $\max_v[|L_v|] \leq \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) \leq \ln(1/\varepsilon)$, and since $Z_v = \mathbb{E}[L_v \mid v \in \Pi^{\uparrow}]$, the same bound holds for $\max_v[|Z_v|]$. Also, by Lemma 14 we know that Z_v is nonegative, hence we have $|Z_v| = Z_v$. From these considerations, and rearrangement of the second sum in the RHS in (20) we have,

$$(20) \leq 2\ln(1/\varepsilon) \sum_{v \in V_\lambda} \mathsf{vis}(v)\ Z_v$$
$$+ 2 \sum_{u \in V_\lambda} \left(\Pr[\mathsf{vis}(u)] Z_u \sum_{v \in u^{\downarrow} \cap V_\lambda} \Pr[\mathsf{vis}(v) | \mathsf{vis}(u)] Z_v \right)$$
$$= 2\ln(1/\varepsilon)\mathbb{E}[Z_\lambda]$$
$$+ 2 \sum_{u \in V_\lambda} \left(\Pr[\mathsf{vis}(u)] Z_u\, \mathbb{E}\Big[\sum_{v \in (u,\Pi) \cap V_\lambda} L_v \Big| \mathsf{vis}(u) \Big] \right).$$

Now $|\sum_{v : v \in (u,\Pi) \cap V_\lambda} L_v(\pi)| = |L_\lambda(\Pi) - L_\lambda(u)| \leq |L_\lambda(\Pi)| + |L_\lambda(v)|$, which is at most $2\ln(1/\varepsilon)$ by Lemma 9, and thus

$$\leq 2\ln(1/\varepsilon)\mathbb{E}[Z_\lambda] + 4\ln(1/\varepsilon) \sum_{u \in \Pi^{\uparrow} \cap V_\lambda} \mathsf{vis}(u) Z_u$$
$$\leq 2\ln(1/\varepsilon)\mathbb{E}[Z_\lambda] + 4\ln(1/\varepsilon)\mathbb{E}[Z_\lambda]$$
$$\leq 6\ln(1/\varepsilon)\mathbb{E}[L_\lambda] \quad \text{(Lemma 12)}$$
$$\leq 6\ln(1/\varepsilon)\frac{16}{\varepsilon^3}\frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n} \quad \text{(Lemma 10)}$$
$$\leq \frac{96}{\varepsilon^4} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n} \quad (\text{using } \ln\frac{1}{\varepsilon} \leq \frac{1}{\varepsilon}).$$

□

Lemma 16 *For every noisy copy λ,*

$$\mathbb{E}[Y_\lambda^2] \leq \frac{136}{\varepsilon^4} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n}.$$

Proof. We have

$$\mathbb{E}[Y_\lambda^2] = \mathbb{E}[(L_\lambda - Z_\lambda)^2] = \mathbb{E}[L_\lambda^2] - 2\mathbb{E}[L_\lambda Z_\lambda] + \mathbb{E}[Z_\lambda^2].$$

We know by Lemma 9 that $|L_\lambda| \leq \ln(1/\varepsilon)$, hence

$$|\mathbb{E}[L_\lambda Z_\lambda]| \leq \ln(1/\varepsilon)\mathbb{E}[|Z_\lambda|],$$

now by Lemmas 14, 12, and 10 we get

$$\mathbb{E}[|Z_\lambda|] = \mathbb{E}[Z_\lambda] = \mathbb{E}[L_\lambda] \leq \frac{4}{\varepsilon^3} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n},$$

hence

$$|\mathbb{E}[L_\lambda Z_\lambda]| \leq \ln(1/\varepsilon)\frac{4}{\varepsilon^3} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n} \leq \frac{4}{\varepsilon^4} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n}.$$

We know by Lemma 11 that $\mathbb{E}[L_\lambda^2] \leq \frac{32}{\varepsilon^4} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n}$, and by Lemma 15 that $\mathbb{E}[Z_\lambda^2] \leq \frac{96}{\varepsilon^4} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n}$. These observations together give

$$\mathbb{E}[Y_\lambda^2] \leq \left(\frac{32}{\varepsilon^4} + \frac{8}{\varepsilon^4} + \frac{96}{\varepsilon^4}\right)\frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n} \leq \frac{136}{\varepsilon^4} \cdot \frac{\mathsf{H}(\Pi|N_{\hat{\lambda}})}{n}.$$

□

We need one more observation before putting things together to complete the proof of Lemma 8.

Lemma 17 *Let T be a* gnd*-tree, and let Π denote the random variable denoting the leaf reached by executing T on input* **0***. We have,*

$$\sum_\lambda \mathsf{H}(\Pi|N_{\hat{\lambda}}) = \mathsf{H}(\Pi). \quad (21)$$

The proof follows by induction on the number of internal nodes in T, where the inductive step is obtained by removing two sibling leaves from the tree. The details will appear in the full paper.

4.5 Completing the proof of Lemma 8

We separately upper bound the two probabilities in the RHS of (17).

$$\Pr\Big[\sum_\lambda Y_\lambda \geq \frac{\ln(\frac{n}{\theta})}{2}\Big] \leq \Pr[4(\sum_\lambda Y_\lambda)^2 \geq \ln^2(n/\theta)]$$
$$\overset{(1)}{\leq} \frac{\mathbb{E}\left[4(\sum_\lambda Y_\lambda)^2\right]}{\ln^2(n/\theta)} \overset{(2)}{\leq} \frac{4\sum_\lambda \mathbb{E}[Y_\lambda^2]}{\ln^2(n/\theta)}$$
$$\overset{(3)}{\leq} \frac{544\sum_\lambda \mathsf{H}(\Pi|N_{\hat{\lambda}})}{\varepsilon^4 n \ln^2(n/\theta)} \overset{(4)}{=} \frac{544\ \mathsf{H}(\Pi)}{\varepsilon^4 n(\ln(n/\theta)^2)}$$
$$\overset{(5)}{\leq} \frac{544\,\mathsf{depth}(T)}{\varepsilon^4 n(\ln(n/\theta)^2)}, \quad (22)$$

where (1) uses Markov's inequality, (2) follows from Lemma 13, (3) follows from Lemma 16, (4) from Lemma 17 and (5) from the fact that $H(\Pi) \leq \log|\mathsf{leaves}| \leq \mathsf{depth}(T)$.

For the second upper bound, we can apply Markov's inequality, since we know by Lemma 14 that the Z_λ's are positive.

$$\Pr\left[\sum_\lambda Z_\lambda \geq \ln(n/\theta)/2\right] \leq \frac{2\,\mathbb{E}\left[\sum_\lambda Z_\lambda\right]}{\ln(n/\theta)} = \frac{2\,\sum_\lambda \mathbb{E}[Z_\lambda]}{\ln(n/\theta)} \tag{23}$$

Lemmas 12 and 11 together now give

$$\begin{aligned}\frac{2\,\sum_\lambda \mathbb{E}[Z_\lambda]}{\ln(n/\theta)} = \frac{2\sum_\lambda \mathbb{E}[L_\lambda]}{\ln(n/\theta)} &\leq \frac{2\sum_\lambda \mathsf{H}(\Pi|N_{\hat{\lambda}})}{n\ln(n/\theta)} \\ &\leq \frac{8\;\mathsf{H}(\Pi)}{\varepsilon^3 n\ln(n/\theta)} \leq \frac{8\,\mathsf{depth}(T)}{\varepsilon^3 n\ln(n/\theta)}\end{aligned} \tag{24}$$

Using (22), (23), and (24) in (17), we get

$$\begin{aligned}\Pr[\sum_\lambda L_\lambda \geq \ln(n/\theta)] &\leq \frac{544\,\mathsf{depth}(T)}{\varepsilon^4 n(\ln(n/\theta)^2)} + \frac{8\,\mathsf{depth}(T)}{\varepsilon^3 n\ln(n/\theta)} \\ &\leq \frac{600\,\mathsf{depth}(T)}{\varepsilon^4 n\ln(n/\theta)}.\end{aligned}$$

This completes the proof of Lemma 8 and thus also of Theorem 2 and of Theorem 1.

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.

[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).

[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] 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).

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

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

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

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

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

[27] 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.

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

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

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

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

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

[33] 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.

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