THEORY OF COMPUTING, Volume 22 (2072), 2026, pp. 1–27
www.theoryofcomputing.org

SPECIAL ISSUE: APPROX-RANDOM 2021

On the Probabilistic Degree of an n-variate Boolean Function

Srikanth Srinivasan* S. Venkitesh†

Received July 21, 2021; Revised July 12, 2023; Published June 30, 2026

Abstract. (Nisan and Szegedy, Comput. Complex. 1994) showed that any Boolean function $f : \{0,1\}^n \to \{0,1\}$ that depends on all its input variables, when represented as a real-valued multivariate polynomial $P(x_1, \ldots, x_n)$, has degree at least $\log n - O(\log \log n)$. This was improved to a tight $(\log n - O(1))$ bound by (Chiarelli et al., Combinatorica 2020). Similar statements are also known for other Boolean function complexity measures such as Sensitivity (Simon, Inf. Control. 1982), Quantum query complexity, and Approximate degree (Ambainis and de Wolf, Comput. Complex. 2014).

In this paper, we address this question for *Probabilistic degree*. The function f has probabilistic degree at most d if there is a random real-valued polynomial of degree at most d that agrees with f at each input with high probability. Our understanding of this complexity measure is significantly weaker than those above: for instance, we do not even know the probabilistic degree of the OR function, the best-known

A conference version of this paper appeared in the Proceedings of RANDOM 2021 [28].

*On leave from Department of Mathematics, IIT Bombay. Supported by startup grant from Aarhus University.

†This work was completed while the author was at the Department of Mathematics, IIT Bombay, supported by the Senior Research Fellowship of HRDG, CSIR, Government of India.

ACM Classification: F.1.3

AMS Classification: 68Q17

Key words and phrases: truly n-variate Boolean function, probabilistic polynomial, probabilistic degree

© 2026 Srikanth Srinivasan and S. Venkitesh
Licensed under a Creative Commons Attribution License (CC-BY)
DOI: 10.4086/toc.2026.v026a2072

bounds put it between $(\log n)^{1/2-o(1)}$ and $O(\log n)$ (Beigel et al., STOC'91; Tarui, Theor. Comput. Sci. 1993; Harsha and Srinivasan, Random Struct. Algorithms 2019).

Here we can give a near-optimal understanding of the probabilistic degree of n-variate functions f, *modulo* our lack of understanding of the probabilistic degree of OR. We show that if the probabilistic degree of OR is $(\log n)^c$, then the minimum possible probabilistic degree of such an f is at least $(\log n)^{c/(c+1)-o(1)}$, and we show this is tight up to $(\log n)^{o(1)}$ factors.

1 Introduction

1.1 Background and motivation

Representing Boolean functions $f : \{0,1\}^n \to \{0,1\}$ by polynomials is a tried-and-tested technique that has found uses in many areas of Theoretical Computer Science. In particular, such representations have led to important results in Complexity theory [7, 9], Learning theory [19, 11], and Algorithm Design [31].

There are many different kinds of polynomial representations that are useful in various applications. The most straightforward way to represent a Boolean function $f : \{0,1\}^n \to \{0,1\}$ by a polynomial is by the *exact representation* – finding a $P \in \mathbb{R}[x_1, \ldots, x_n]$ such that $P(a) = f(a)$ for all $a \in \{0,1\}^n$. It is a standard fact (say by Möbius Inversion or polynomial interpolation) that any f has an exact representation has degree at most n, and the smallest degree of such an exact representation is called the *degree of* f (or sometimes the *Fourier degree of* f because of its close relation to the Fourier spectrum of f [23]), and denoted $\deg(f)$. The exact representation is in fact *unique* if we restrict P to be *multilinear*, i.e. that no variable has degree more than 1 in P, and consequently, we have $\deg(f) = \deg(P)$.

The degree of f is an important notion of *complexity* of the function f and is closely related to a slew of combinatorial measures of Boolean function complexity such as *Sensitivity, Decision Tree complexity, Quantum Query complexity*, etc. (see, e.g., the survey of Buhrman and de Wolf [10] for a nice introduction). Given a complexity measure $\mu(\cdot)$ (such as $\deg(\cdot)$) on Boolean functions, a natural question to ask is the following.

Question 1.1. How small can $\mu(f)$ be for a function f on n variables?

To make this question interesting, one must exclude trivial functions like the constant functions, and more generally, functions that depend on just a small subset of their input variables. This brings us to the following definition.

Definition 1.2 (Truly n-variate Boolean function[1])**.** We say that a Boolean function $f(x_1, \ldots, x_n)$ depends on its input variable x_i, or equivalently that x_i is *influential* for f, if there is an input a such that flipping the value of the ith variable at a changes the value of f (in this case, we also say that x_i is influential for f at a). We say that a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *truly n-variate* if it depends on all its n variables.

[1]Such functions are also called *non-degenerate* Boolean functions in the literature [26].

A number of results have addressed questions regarding how small complexity measures can be for truly n-variate Boolean functions.

1. Motivated by problems in Learning theory and PRAM lower bounds, Nisan and Szegedy [22] showed that any truly n-variate function has degree at least $\log n - O(\log\log n)$. Recently, this was improved to $\log n - O(1)$ by Chiarelli, Hatami and Saks [13]. There are standard examples of Boolean functions (see, e.g., the *Addressing function* defined below) for which this is tight.

2. Ambainis and de Wolf [4] studied the same question for the *approximate degree of* f, which is defined to be the minimum degree of a polynomial P such that $|P(a) - f(a)| < 1/3$ for all $a \in \{0,1\}^n$. This complexity measure is closely related to the *quantum query complexity of* f [10].

   Ambainis and de Wolf [4] showed that any truly n-variate function has approximate degree (and also quantum query complexity) $\Omega(\log n/\log\log n)$. They also constructed variants of the Addressing function for which this bound is tight up to constant factors.

3. Such results are also known for more combinatorial complexity measures, such as the *sensitivity* of a Boolean function f, which is defined as follows. The sensitivity of f at a point $a \in \{0,1\}^n$ is the number of input variables to f that are influential for f at a. The sensitivity of f is the maximum sensitivity of f at any input.

   Simon [26] showed that any truly n-variate f has sensitivity at least $\log n - O(\log\log n)$. This is also tight up to the $O(\log\log n)$ additive term (say, for the Addressing function).

We address Question 1.1 for another well-known polynomial-degree measure called the *Probabilistic degree.* We define this notion first. By a *random polynomial* over $\mathbb{R}[x_1,\ldots,x_n]$, we mean a discrete random variable (over some sample space) that takes polynomials in $\mathbb{R}[x_1,\ldots,x_n]$ as its values, and has a probability mass function with finite support.

Definition 1.3 (Probabilistic polynomial and Probabilistic degree[2])**.** Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and an $\varepsilon \geq 0$, an *ε-error probabilistic polynomial* for f is a random polynomial $\mathbf{P}$ over $\mathbb{R}[x_1,\ldots,x_n]$ such that for each $a \in \{0,1\}^n$,

$$\Pr_{\mathbf{P}}[\mathbf{P}(a) \neq f(a)] \leq \varepsilon.$$

(Note that $\mathbf{P}(a)$ need not be Boolean when $\mathbf{P}(a) \neq f(a)$.)

We say that the degree of $\mathbf{P}$, denoted $\deg(\mathbf{P})$, is at most d if the probability mass function of $\mathbf{P}$ is supported on polynomials of degree at most d. Finally, we define the *ε-error probabilistic degree* of f, denoted $\mathrm{pdeg}_\varepsilon(f)$, to be the least d such that f has an ε-error probabilistic polynomial of degree at most d.

In the special case that $\varepsilon = 1/3$, we omit the subscript in the notation above and simply use $\mathrm{pdeg}(f)$.

[2]These, including the notion of a random polynomial, can also be defined over other fields similarly.

The probabilistic degree is a fundamentally important and well-studied complexity measure of Boolean functions. It was implicitly introduced (in the finite field setting) in a celebrated result of Razborov [24], who showed how to use it to construct low-degree polynomial approximations to small-depth circuits, and hence prove strong circuit lower bounds. The real-valued version was first studied by Beigel, Reingold and Spielman [8] and Tarui [29] who were motivated by other circuit lower bound questions and oracle separations. This measure has since found other applications in complexity theory [5, 7], Pseudorandom generator constructions [9], Learning theory [11], and Algorithm design [31, 1]. Further, in many of these applications (e.g, [5, 9, 1]) we *need* real-valued approximations.

Despite this, however, our understanding of probabilistic degree is much less developed than the other measures above. For instance, near-optimal lower bounds of $n^{1-o(1)}$ on the probabilistic degree of an explicit Boolean function $f : \{0,1\}^n \to \{0,1\}$ were proved only recently by Viola [30], and are only known for a function in the complexity class E^{NP}; in comparison, the Parity function has degree and approximate degree n, which is the largest possible. Another example is the OR function on n variables. It is trivial to estimate the degree of OR (which is n) and well-known that its approximate degree is $\Theta(\sqrt{n})$ [22, 15]. However, its probabilistic degree (over the reals) remains unknown: the best known upper bound is $O(\log n)$ due to independent results of Beigel et al. [8] and Tarui [29], while the best lower bound is $(\log n)^{1/2-o(1)}$ due to Harsha and the first author [16]. This indicates that we need better tools to understand probabilistic degree in general and over the reals in particular. This is one of the motivations behind this paper.

Another motivation is to understand the contrast between the setting of real-valued probabilistic polynomials and polynomials over constant-sized finite fields. At a high level, this helps us understand the contrast between circuit complexity classes AC^0 and $\mathrm{AC}^0[p]$, as the former class of circuits has low-degree probabilistic polynomials over the reals [8, 29], while the latter does not [27]. It is easy to show that there are truly n-variate Boolean functions of constant degree over finite fields (e.g., the parity function is a linear polynomial over the field $\mathbb{F}_2$). It is interesting to ask to what extent such phenomena fail over the reals. For instance, one of our results in this work (Corollary 1.6) is that all truly n-variate Boolean functions over the reals have probabilistic degree at least $(\log n)^{1/3-o(1)}$.

A final motivating reason is to understand more precisely the relationships between probabilistic degree and other complexity measures such as approximate degree. A recent conjecture of Golovnev, Kulikov and Williams [14] shows that porting results for approximate degree to probabilistic degree would have interesting consequences for De Morgan formula lower bounds. While our results in this paper do not have a direct implication for this question, and in particular do not make progress on the conjecture of [14], we hope that our techniques will be useful in making progress on this line of investigation, and hopefully uncover other connections between probabilistic degree and other Boolean function complexity measures.

With these motivations in mind, we address Question 1.1 in the setting of Probabilistic degree. That is, what is the lowest possible probabilistic degree of a truly n-variate Boolean function? As far as we know, this question has not been addressed before. Putting together Simon's bound on the sensitivity of a truly n-variate function with known probabilistic degree

lower bounds [16], one can show a lower bound of $(\log\log n)^{1/2-o(1)}$. This is quite far from the best known upper bounds of $O(\log n)$, which hold for say the OR function [8, 29] and the Addressing function defined below in Section 1.3.

1.2 Results

Our aim is to prove a result characterizing the minimum possible probabilistic degree of a truly n-variate Boolean function. However, the gap even just in our understanding of the OR function (as mentioned above) tells us that this may not yet be within reach. What we are able to do is to give a near-complete characterization modulo the gap between known upper and lower bounds for $\mathrm{pdeg}(\mathrm{OR})$. *Moreover, the answer is non-trivial: it is not simply* $\mathrm{pdeg}(\mathrm{OR})$.

More precisely, our results are the following. Below, OR_n denotes the OR function on n variables. We assume that we have bounds of the form $\mathrm{pdeg}(\mathrm{OR}_n) = (\log n)^{c\pm o(1)}$ for some $c > 0$.

Theorem 1.4. *Assume that* $\mathrm{pdeg}(\mathrm{OR}_n) \geq (\log n)^{c-o(1)}$ *for some* $c > 0$ *and all large enough* $n \in \mathbb{N}$. *Then, any truly* n*-variate Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *satisfies* $\mathrm{pdeg}(f) \geq (\log n)^{(c/(c+1))-o(1)}$.

Theorem 1.5. *Assume that* $\mathrm{pdeg}(\mathrm{OR}_n) \leq (\log n)^{c+o(1)}$ *for some* $c > 0$ *and all large enough* $n \in \mathbb{N}$. *Then, there exists a truly* n*-variate Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *such that* $\mathrm{pdeg}(f) \leq (\log n)^{(c/(c+1))+o(1)}$.

Thus, we get close-to-matching lower and upper bounds for truly n-variate Boolean functions assuming close-to-matching lower and upper bounds for the OR function. However, the above statements also imply *unconditional* lower and upper bounds on the probabilistic degrees of truly n-variate Boolean functions. Using known results that yield $(\log n)^{(1/2)-o(1)} \leq \mathrm{pdeg}(\mathrm{OR}_n) \leq O(\log n)$ [8, 29, 16], we get

Corollary 1.6. *Any truly* n*-variate Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *satisfies* $\mathrm{pdeg}(f) \geq (\log n)^{(1/3)-o(1)}$.

Corollary 1.7. *There exists a truly* n*-variate Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *such that* $\mathrm{pdeg}(f) \leq (\log n)^{(1/2)+o(1)}$.

Remark 1.8. The reader may wonder why we assume lower and upper bounds of the form $(\log n)^{c\pm o(1)}$ for $\mathrm{pdeg}(\mathrm{OR}_n)$. This is because the gaps between the known upper and lower bounds are $(\log n)^{\Omega(1)}$, and so it makes sense to use a characterization that shrinks this gap to something relatively insignificant. Furthermore, the best known lower bound on $\mathrm{pdeg}(\mathrm{OR}_n)$ is of the form $(\log n)^{1/2-o(1)}$ [16] (more precisely, it is $\Omega((\log n)/(\log\log n)^{3/2})$).

If we instead assume a more precise characterization $\mathrm{pdeg}(\mathrm{OR}_n) = \Theta((\log n)^c)$, then going through the proofs of the above theorems would yield a sharper lower bound of $\Omega((\log n)^{c/(c+1)}/(\log\log n)^2)$ for any truly n-variate Boolean function and a better upper bound of $O((\log n)^{c/(c+1)})$ for some truly n-variate Boolean function.

1.3 Proof Outline

Our proof is motivated by two important examples. The first of these is the OR_n function which has probabilistic degree at most $O(\log n)$ by results of [8, 29] and at least $(\log n)^{(1/2)-o(1)}$ by [16]. The second is the *Addressing function*, which we now define.

The Addressing function Addr_r has $n = r + 2^r$ variables. We think of the input variables as being divided into two parts: there are r 'addressing' variables $y_1, \ldots, y_r$ and 2^r 'addressed' variables $\{z_a \mid a \in \{0,1\}^r\}$ (the latter part of the input is thus indexed by elements of $\{0,1\}^r$). On an input $(a, A) \in \{0,1\}^r \times \{0,1\}^{2^r}$, the output of the function is defined to be A_a (i.e. the ath co-ordinate of the vector A). The Addressing function satisfies $\deg(\mathrm{Addr}_r) = r + 1 = O(\log n)$. This example is quite relevant to this line of work: in particular, it implies that the results of Nisan and Szegedy [22] and Chiarelli et al. [13] stated above are tight, and is also a tight example for Simon's theorem [26].

We now describe the upper and lower bound proofs, starting with the less technical upper bound.

The Upper Bound. Given that we have two natural families of truly n-variate functions that have degree $O(\log n)$, one may suspect that this is the best possible. Indeed this was also our initial conjecture. However, using the ideas of Ambainis and de Wolf [4], we can do better. Ambainis and de Wolf showed that there are truly n-variate Boolean functions that have *approximate degree* $O(\log n / \log\log n)$. Their construction[3] uses a modified Addressing function, where the addressing variables are present in an 'encoded' form. While this blows up the size of the first part of the input, this does not affect n much as the addressing variables take up only a small part of the input. On the other hand, the advantage is that the 'decoding' procedure can be performed approximately by a suitable low-degree polynomial: a proof of this uses two famous Quantum algorithms, the *Bernstein-Vazirani algorithm* and *Grover search*, along with the fact that efficient Quantum algorithms yield approximating low-degree polynomials [6]. Putting things together yields an improved approximate degree bound for some n-variate f.

We show how to port their construction to the probabilistic degree setting. The first observation is that Grover search, which is essentially an algorithm for computing OR_n, is much more 'efficient' in the probabilistic degree setting, as $\mathrm{pdeg}(\mathrm{OR}_n) = O(\log n)$, while its approximate degree is $\Omega(\sqrt{n})$ [22]. The second observation is that the Bernstein-Vazirani algorithm, which can be thought of as a decoding algorithm for a suitable error-correcting code, can be replaced by polynomial interpolation. This gives a good idea of why we should also be able to use a similar construction in the probabilistic degree setting. In fact, the better probabilistic degree upper bound for OR_n implies that we should be able to get a *better* bound than what is possible for approximate degree. Indeed this is true. By a similar construction, we show that we can construct a truly n-variate f with probabilistic degree $O(\sqrt{\log n})$ *unconditionally*, which is quite a bit better than previous results for any of the above degree measures. If we assume, moreover, that $\mathrm{pdeg}(\mathrm{OR}_n) \leq (\log n)^{c+o(1)}$, the same construction yields a function with probabilistic degree $(\log n)^{(c/(c+1))+o(1)}$.

[3] They actually give two, slightly different, constructions. We use the second one here.

The Lower Bound. Given that the upper bound construction uses the Addressing function as well as the OR function, it is only natural that the lower bound would use the lower bounds for these two families of functions. Our hypothesis already assumes a lower bound of $(\log n)^{c-o(1)}$ for $\mathrm{pdeg}(\mathrm{OR}_n)$. For the addressing function Addr_r described above, one can prove an $\Omega(r) = \Omega(\log n)$ lower bound in the following way. We observe that by setting the 2^r addressed variables uniformly at random, we obtain a uniformly random function on the r addressing variables. By a counting argument, one can show that a uniformly random Boolean function $\mathbf{F}$ on r variables has probabilistic degree $\Omega(r)$ with high probability. In particular, as setting some input variables to constants can only reduce probabilistic degree, this implies that $\mathrm{pdeg}(\mathrm{Addr}_r) = \Omega(r) = \Omega(\log n)$. Note that this is tight, as $\deg(\mathrm{Addr}_r) = r + 1$.

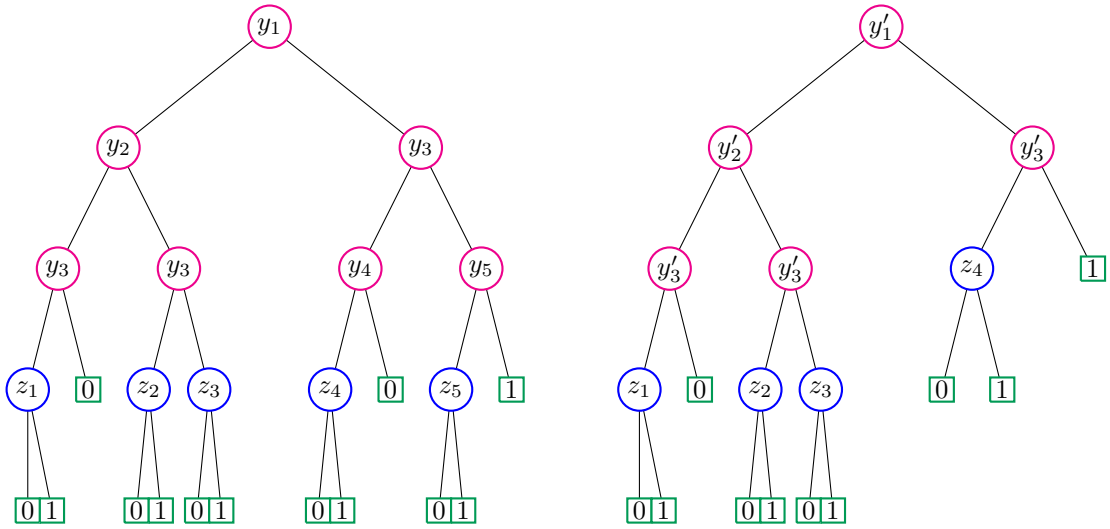
Our aim is to generalize the above lower bounds enough to prove a lower bound for any truly n-variate f. The first informal observation is that the OR_n function is the 'simplest' function on n variables to have sensitivity n. Therefore, it is intuitive that any Boolean function with sensitivity n should have probabilistic degree at least that of the OR_n function. We show that this is true, up to $(\log n)^{o(1)}$ factors. More generally, we show that any Boolean function f with sensitivity s has probabilistic degree at least that of the OR function on s variables (up to $(\log s)^{o(1)}$ factors). The proof of this is in the contrapositive: we use a probabilistic degree upper bound for f to construct a probabilistic polynomial for OR_s. The ideas behind this go back to a sampling argument used in the works of Beigel et al. and Tarui [8, 29]. Viewing this argument more abstractly, we can use this to construct a reduction from OR_s to f (for any f of sensitivity s) in the probabilistic degree setting.

The above argument implies a strong lower bound for any n-variate f with large sensitivity. In particular, it implies that if f has sensitivity at least $s = n^{\Omega(1)}$, then its probabilistic degree is almost that of the OR function. We now consider the case of functions with small sensitivity (specifically when $s = n^{o(1)}$), which is the most technical part of the proof. By a recent breakthrough result of Huang [18], we also know that f also has a *decision tree* (we refer the reader to [10] for the definition of Decision trees) of depth $d = \mathrm{poly}(s) = n^{o(1)}$.[4] The prototypical example of such an f is the Addressing function which has a decision tree of depth $r + 1 = \lfloor \log n \rfloor + 1$, which we argued a lower bound for above. The idea, in general, is to find a copy of something like an Addressing function 'inside' the function f.

We illustrate how this argument works by considering a special case of the problem, which is only a small variant of the Addressing function. Assume that a truly n-variate function f is computed by a decision tree T of depth $d = \mathrm{poly}(\log n)$. Note that as the function depends on all its variables, each of the underlying n variables appear in the tree T. To make things even simpler, assume that we have $n/2$ 'addressing' variables $y_1, \ldots, y_{n/2}$ and $n/2$ 'addressed' variables $z_1, \ldots, z_{n/2}$. The tree reads $d - 1$ addressing variables among $y_1, \ldots, y_{n/2}$ in some (possibly adaptive) fashion and then possibly queries one addressed variable, the value of which is output. (See Figure 1 (a).)

How do we argue a lower bound on $\mathrm{pdeg}(f)$? We could try to proceed as above and set

[4] Strictly speaking, we do not need to use Huang's result as we could also use the known polynomial relationship between the decision tree height and the *block sensitivity* of a function [21]. But it is notationally easier to work with sensitivity.



(a) Before projection (b) After projection

Figure 1: The function $f(y_1, \ldots, y_5, z_1, \ldots, z_5)$ is defined by the decision tree on the left (we assume that the left child corresponds to the queried variable taking value 0). When $z_1, \ldots, z_5$ are set i.u.a.r. to $\mathbf{b}_1, \ldots, \mathbf{b}_5$, we get a random function $\mathbf{F}(y_1, \ldots, y_5)$ such that $\mathbf{F}(00000) = \mathbf{b}_1, \mathbf{F}(01000) = \mathbf{b}_2, \mathbf{F}(01100) = \mathbf{b}_3, \mathbf{F}(10000) = \mathbf{b}_4, \mathbf{F}(10100) = \mathbf{b}_5$. After a projection that maps $y_1 \mapsto y'_1; y_2 \mapsto y'_2; y_3, y_4, y_5 \mapsto y'_3$, we get the function f' computed by the tree on the right. This reduces the number of addressing variables to 3. But also note that the variable z_5 is no longer relevant as the path leading to it is inconsistent with the projection. So the number of addressed variables falls to 4.

the addressed variables $z_1, \ldots, z_{n/2}$ as random to obtain a random function $\mathbf{F}$ in $y_1, \ldots, y_{n/2}$. However, this function is not *uniformly* random, as it is sampled using only $n/2$ random bits, while the number of functions in $n/2$ variables is $2^{2^{n/2}}$. Nevertheless, we can observe that the function $\mathbf{F}$ does take independent random values at at least $n/2$ distinct inputs, those which are consistent with $n/2$ distinct paths in T leading to the various addressed variables. (See Figure 1 (a).) We could try to lower bound $\mathrm{pdeg}(\mathbf{F})$ as above.

This leads to the following general question: given a random function $\mathbf{F} : \{0,1\}^r \to \{0,1\}$ that takes independent and random values at M distinct inputs in $\{0,1\}^r$, what can we say about the probabilistic degree of $\mathbf{F}$? By a more general counting argument, we are able to show that with high probability, the probabilistic degree of $\mathbf{F}$ is at least $\Omega(\log M / \log r)$. This is easily seen to be tight in the case that X is, say, a Hamming ball of radius $R \leq r^{1-\Omega(1)}$. (In the case that $M = 2^r$, this leads to a bound of $\Omega(r/\log r)$, nearly matching the claim for random functions that we mentioned above. A tight bound can be obtained in the same way but is harder to state for general M.)

Given this bound for random functions, we can try to use it in the case of the function f above. Unfortunately, in this case, both parameters r and M are $n/2$, and hence we do not get any non-trivial bound. However, we show that we can still reduce to a case where a non-trivial

bound is possible (this is where the depth of T comes in). More precisely, we reduce the number of addressing variables by *projecting* the $n/2$ addressing variables to a smaller set of r' variables $Y' = \{y'_1, \ldots, y'_{r'}\}$. That is, we randomly set each variable Y to a uniformly random variable in Y' to get a different function in the variables $Y' \cup Z$. This has the effect of reducing the number of addressing variables to r'. But there is also a potential problem: the projection could also render some of the addressed variables irrelevant, as the paths that lead to them become inconsistent. (See Figure 1 (b).)

Nevertheless, if we choose r' large enough (something like $r' = 4d^2$ is enough by the Birthday paradox), the variables of each path are sent to *distinct* variables in Y' with high probability, which implies that each addressed variable remains relevant with high probability. In particular, there is a projection that maps f to an 'Addressing function' with only poly$(\log n)$ addressing variables and $\Omega(n)$ addressed variables. Now applying the argument for random functions, we get a probabilistic degree lower bound of $\Omega(\log n/\log\log n)$ for this function, nearly matching what we obtained for the Addressing function. As projections do not increase probabilistic degree, the same bound holds for f, concluding the proof in this special case.[5]

A similar argument can be carried out in the general case by first carefully partitioning the variables into the addressing and addressed variables. We do this by looking at the structure of the decision tree T. These details are postponed to the formal proof. In general, this argument yields a lower bound of $\Omega(\log n/\log s)$ on the probabilistic degree of a truly n-variate function f with sensitivity at most s.

Using this lower bound along with the previous lower bound for functions of sensitivity at least s, and optimizing our choice of s, yields a lower bound of $(\log n)^{c/(c+1)-o(1)}$ for any truly n-variate function f.

2 Preliminaries

Functions, Restrictions, Projections. Throughout, we work with real-valued functions $f : \{0,1\}^n \to \mathbb{R}$. Boolean functions (i.e. functions mapping $\{0,1\}^n$ to $\{0,1\}$) are also treated as real-valued. We use boldface notation to denote random variables. A *random function* $\mathbf{F}$ is a probability distribution over functions.

A *restriction* on n variables is a map $\rho : [n] \to \{0,1,*\}$. Given a function $f : \{0,1\}^n \to \mathbb{R}$ and a restriction ρ on n variables, we have a natural restricted function f_ρ defined by setting the ith input variable to f to 0, 1 or leaving it as is, depending on whether $\rho(i)$ is 0, 1 or $*$ respectively. Note that the function f_ρ now depends on $|\rho^{-1}(*)|$ many variables. However, we sometimes also treat f_ρ as a function of all the original variables that only *depends* on (a subset of) the variables indexed by $\rho^{-1}(*)$.

A *projection* from n variables to m variables is a map $\nu : [n] \to [m]$. Given a function $f : \{0,1\}^n \to \mathbb{R}$ and a projection ν from n variables to m variables, we get a function $f|_\nu : \{0,1\}^m \to \mathbb{R}$ by identifying variables of f that map to the same image under ν.

[5] Random projections of this kind have been used recently to prove important results in circuit complexity [17, 12]. However, as far as we know, they have not been used to prove probabilistic degree lower bounds.

Some Boolean functions. For any positive integer n, we use OR_n, AND_n and Maj_n to denote the OR, AND and Majority functions on n variables respectively.

Fact 2.1. *We have the following simple facts about probabilistic polynomials.*

1. *(Interpolation) Any function $f : \{0,1\}^n \to \{0,1\}$ has an exact multilinear polynomial representation of degree at most n, i.e.* $\deg(f) := \mathrm{pdeg}_0(f) \leq n$.

2. *(Shifts and Restrictions) Fix any $f : \{0,1\}^n \to \{0,1\}$ and any $\varepsilon \geq 0$. Then the function $g : \{0,1\}^n \to \{0,1\}$ defined by $g(x) = f(x \oplus y)$ for a fixed $y \in \{0,1\}^n$ has the same probabilistic degree as f, i.e.,* $\mathrm{pdeg}_\varepsilon(g) = \mathrm{pdeg}_\varepsilon(f)$.

   If $g : \{0,1\}^m \to \{0,1\}$ is a restriction or a projection of f, then $\mathrm{pdeg}_\varepsilon(g) \leq \mathrm{pdeg}_\varepsilon(f)$.

3. *(Composition) For any Boolean function f on k variables and any Boolean functions $g_1, \ldots, g_k$ on a common set of m variables, let h denote the natural composed function $f(g_1, \ldots, g_k)$ on m variables. For $\varepsilon, \delta_1, \ldots, \delta_k \geq 0$, let $\mathbf{P}, \mathbf{Q}_1, \ldots, \mathbf{Q}_k$ be probabilistic polynomials for $f, g_1, \ldots, g_k$ respectively with errors $\varepsilon, \delta_1, \ldots, \delta_k$ respectively. Then, $\mathbf{R} = \mathbf{P}(\mathbf{Q}_1, \ldots, \mathbf{Q}_k)$ is a probabilistic polynomial for h with error at most $\varepsilon + \sum_i \delta_i$.*

   In particular, for any $\varepsilon, \delta > 0$, we have $\mathrm{pdeg}_{\varepsilon+k\delta}(h) \leq \mathrm{pdeg}_\varepsilon(f) \cdot \max_{i\in[k]} \mathrm{pdeg}_\delta(g_i)$.

We also have the following theorem from the work of Harsha and the first author [16] about error reduction for probabilistic polynomials. We give a proof for completeness.

Theorem 2.2 (Error reduction for probabilistic polynomials [16, Theorem 1])**.** *For any $\delta < \varepsilon \leq 1/3$ and any Boolean function f, if $\mathbf{P}$ is an ε-error probabilistic polynomial for f, then $\mathbf{Q} = M(\mathbf{P}_1, \ldots, \mathbf{P}_\ell)$ is a δ-error probabilistic polynomial for f where $\ell = O(\log(1/\delta)/\log(1/\varepsilon))$, M is the exact multilinear polynomial for Maj_ℓ and $\mathbf{P}_1, \ldots, \mathbf{P}_\ell$ are independent copies of $\mathbf{P}$. In particular, we have* $\mathrm{pdeg}_\delta(f) \leq \mathrm{pdeg}_\varepsilon(f) \cdot O(\log(1/\delta)/\log(1/\varepsilon))$.

Proof. Consider any $\ell \geq 3$. (We may assume that ℓ is odd.) Fix any $a \in \{0,1\}^n$. So we have $\Pr_{\mathbf{P}_i}[\mathbf{P}_i(a) \neq f(a)] \leq \varepsilon$, for all $i \in [\ell]$. For simplicity, assume $\Pr_{\mathbf{P}_i}[\mathbf{P}_i(a) \neq f(a)] = \varepsilon$, for all $i \in [\ell]$. Now let $\mathbf{Y}_i$ be the indicator of $\mathbf{P}_i(a) \neq f(a)$, for all $i \in [\ell]$, and $\mathbf{Y} = \sum_{i=1}^{\ell} \mathbf{Y}_i$. So $\mathbb{E}(\mathbf{Y}) = (1-\varepsilon)\ell$.

Let $M = M(u_1, \ldots, u_\ell) \in \mathbb{R}[u_1, \ldots, u_\ell]$ be the exact multilinear polynomial for Maj_ℓ. Then we have the following, for any $S \subseteq [\ell]$, $|S| \geq \ell/2$.

- Consider the multilinear polynomial $M' := M(u_1, \ldots, u_\ell)\big|_{u_i=0,\, i\in S}$ obtained by setting all the variables $u_i = 0$ for $i \in S$. As M represents the Majority function, It is clear that the polynomial M' evaluates to 0 on any Boolean assignments to the remaining variables (i.e. the variables u_i where $i \notin S$). By the uniqueness of the exact multilinear representation, the polynomial M' must be the zero polynomial. In other words, $M(u_1, \ldots, u_\ell)\big|_{u_i=0,\, i\in S} = 0$
- Similarly, we can conclude that $M(u_1, \ldots, u_\ell)\big|_{u_i=1,\, i\in S} = 1$.

Since $\mathbf{Q} = M(\mathbf{P}_1, \ldots, \mathbf{P}_\ell)$, this implies

$$\Pr_{\mathbf{Q}}[\mathbf{Q}(a) \neq f(a)] = \Pr_{\mathbf{P}_1,\ldots,\mathbf{P}_\ell}\left[\mathbf{Y} < \frac{\ell}{2}\right] = \Pr_{\mathbf{P}_1,\ldots,\mathbf{P}_\ell}[\mathbf{Y} < (1-\theta)(1-\varepsilon)\ell], \quad \text{for } \theta = \frac{1-2\varepsilon}{2(1-\varepsilon)}.$$

Therefore by the Chernoff bound [20, Theorem 4.5], we get

$$\Pr_{\mathbf{Q}}[\mathbf{Q}(a) \neq f(a)] \leq \exp\left(-\frac{\theta^2(1-\varepsilon)\ell}{2}\right) = \exp\left(-\frac{(1-2\varepsilon)^2\ell}{8(1-\varepsilon)}\right).$$

Now one can check using elementary calculus that $\ln(1/\varepsilon) < 8(1-\varepsilon)/(1-2\varepsilon)^2$ for $0 < \varepsilon \leq 1/3$. So if we choose $\ell > \ln(1/\delta)/\ln(1/\varepsilon)$, then we get

$$\ell > \frac{\ln(1/\delta)}{8(1-\varepsilon)/(1-2\varepsilon)^2} \implies \exp\left(-\frac{(1-2\varepsilon)^2\ell}{8(1-\varepsilon)}\right) < \delta.$$

This completes the proof. □

We will need the following known upper and lower bounds on $\mathrm{pdeg}(\mathrm{OR}_n)$.

Theorem 2.3 ([8, 29])**.** $\mathrm{pdeg}_\varepsilon(\mathrm{OR}_n) = O(\log n \cdot \log(1/\varepsilon))$.

Theorem 2.4 ([16])**.** $\mathrm{pdeg}(\mathrm{OR}_n) \geq (\log n)^{1/2-o(1)}$.

For our arguments in this work, in addition to the probabilistic degree, we will also need to consider two other Boolean function complexity measures – decision tree depth and sensitivity.

Definition 2.5 (Decision Tree for a Boolean function)**.** Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. A decision tree computing f is a binary tree such that

1. every leaf vertex is labeled either 0 or 1,
2. every non-leaf vertex is labeled one of the n input bits (say $x_1, \ldots, x_n$) with distinct vertices on any root-to-leaf path labelled by distinct variables,
3. every non-leaf vertex has exactly two children with the edge to the left child labeled 0, and the edge to the right child labeled 1,
4. for any root to leaf path $x_{i_1} \xrightarrow{a_{i_1}} x_{i_2} \xrightarrow{a_{i_2}} \cdots \xrightarrow{a_{i_{r-1}}} x_{i_r} \xrightarrow{x_{i_r}} y$, where $x_{i_1}, \ldots, x_{i_r}$ are the labels on the non-leaf vertices, $a_{i_1}, \ldots, a_{i_r} \in \{0,1\}$ are the labels on the edges, and y is the label on the leaf vertex, we get

$$f(x_1, \ldots, x_n)\big|_{x_{i_1}=a_{i_1}, \ldots, x_{i_r}=a_{i_r}} = y.$$

The depth of a decision tree T is the length of the longest root-to-leaf path in T, and the size is the number of leaves of T.

Let $f : \{0,1\}^n \to \{0,1\}$ be any Boolean function. We use $D(f)$ to denote the smallest depth of a decision Tree computing f.

Definition 2.6 (Sensitivity of a Boolean function)**.** Let $f : \{0,1\}^n \to \{0,1\}$ be any Boolean function. For $a \in \{0,1\}^n$, we use $s(f,a)$ to denote the number of $b \in \{0,1\}^n$ that can be obtained by flipping a single bit of a and satisfying $f(a) \neq f(b)$. The *Sensitivity* of f, denoted $s(f)$, is defined to be the maximum value of $s(f,a)$ as a ranges over $\{0,1\}^n$.

Huang [18] proved the following breakthrough result recently.

Theorem 2.7 (Huang's Sensitivity theorem [18])**.** *There is an absolute constant $c_0 > 0$ such that for all large enough n and all functions $f : \{0,1\}^n \to \{0,1\}$, $D(f) \leq s(f)^{c_0}$.*

Strictly speaking, we do not need to use Huang's Sensitivity theorem in what follows as we could also make do with a polynomial relationship between the decision tree height and the *block sensitivity*[6] of f, which has been known for a long time [21]. However, it is notationally simpler to work with sensitivity.

3 The Lower Bound: Proof of Theorem 1.4

The proof is made up of two lower bounds. We first prove a lower bound on $\mathrm{pdeg}(f)$ for any function f that has large sensitivity $s(f)$; this is by a suitable reduction from the case of the OR function. We then prove a lower bound on $\mathrm{pdeg}(f)$ for any function that depends on all its variables but has small sensitivity; this is by a suitable reduction from a kind of Addressing function. Optimizing over the parameters of the lower bounds will yield the lower bound of the theorem statement.

Throughout this section, we assume that $\mathrm{pdeg}(\mathrm{OR}_n) \geq (\log n)^{c-o(1)}$ for all large enough n.

3.1 The case of large sensitivity

The main result of this section is the following lower bound on the probabilistic degrees of Boolean functions with large sensitivity.

Lemma 3.1. *Let $f : \{0,1\}^n \to \{0,1\}$ be any Boolean function that has sensitivity s. Then,* $\mathrm{pdeg}(f) \geq (\log s)^{c-o(1)}$.

The above lemma is proved via a probabilistic reduction from the OR function on s variables to the function f. This is captured by the following lemma, which shows how a function that has large sensitivity can be used to obtain a probabilistic representation of a large copy of the OR function.

Recall from above that a Boolean function $h : \{0,1\}^s \to \{0,1\}$ is a restriction of a Boolean function $g : \{0,1\}^s \to \{0,1\}$ if h can be obtained by setting some inputs of g to constants. Though h no longer depends on the variables that are set to constants, here we still treat h as a function on all s variables.

Lemma 3.2. *Let $g : \{0,1\}^s \to \{0,1\}$ be any Boolean function such that $g(0^s) = 0$ and $g(x) = 1$ for every x of Hamming weight 1. Then, there exist $\ell = O(\log s)$ independent random restrictions $\mathbf{g}_1, \ldots, \mathbf{g}_\ell$ of g such that for any $a \in \{0,1\}^s$,*

$$\Pr_{\mathbf{g}_1,\ldots,\mathbf{g}_\ell}[\mathrm{OR}_\ell(\mathbf{g}_1(a), \ldots, \mathbf{g}_\ell(a)) \neq \mathrm{OR}_s(a)] \leq \frac{1}{10}.$$

[6] The Block sensitivity of a function $f : \{0,1\}^n \to \{0,1\}$ is defined as follows. Given $a \in \{0,1\}^n$, define $bs(f,a)$ to be the maximum number of pairwise disjoint sets $B_1, \ldots, B_t \subseteq [n]$ such that flipping all the bits indexed by any B_i in a results in an input $b^{(i)}$ such that $f(a) \neq f(b^{(i)})$. Then, the block sensitivity of f is defined to be the maximum value of $bs(f,a)$ over all inputs $a \in \{0,1\}^n$.

We interpret the random function $\mathrm{OR}_\ell(\mathbf{g}_1(a), \ldots, \mathbf{g}_\ell(a))$ as a probabilistic representation of the OR_s function. The reader may be confused by the fact that the probabilistic representation itself uses an OR function; however, note that this OR function is defined on $\ell \ll s$ variables and consequently is a much 'simpler' function (in particular, for us, what is relevant is that $\mathrm{pdeg}(\mathrm{OR}_\ell) = O(\log \ell)$ [8, 29] which is much smaller than $\log s$) .

The proof of Lemma 3.2 is closely related to the argument for constructing a probabilistic polynomial for the OR function from [8, 29]. The observation here is that a similar argument can be used to give a probabilistic reduction from OR_s to any function g as above.

Assuming Lemma 3.2 for now, we first finish the proof of Lemma 3.1.

Proof of Lemma 3.1. We know that f has some input of sensitivity s. Let $a \in \{0,1\}^n$ such that $s(f,a) = s$. By replacing f by $1 - f$ if required, we may assume $f(a) = 0$. (Obviously, $\mathrm{pdeg}_\varepsilon(f) = \mathrm{pdeg}_\varepsilon(1-f)$, for all $\varepsilon \geq 0$.) Now by permuting coordinates if required, we may assume that $f(\tilde{a}^{(j)}) = 1$, where $\tilde{a}^{(j)} := (a_1, \ldots, a_{j-1}, 1 - a_j, a_{j+1}, \ldots, a_n)$ for all $j \in [s]$. Further, if $a \neq 0^n$, we may replace f by f', defined as $f'(x) = f(x \oplus a)$, $x \in \{0,1\}^n$. By Fact 2.1 Item 2, $\mathrm{pdeg}_\varepsilon(f) = \mathrm{pdeg}_\varepsilon(f')$, for all $\varepsilon \geq 0$. So we may assume $f = f'$, and therefore clearly, we have $f(0^n) = 0$, and $f(0^{j-1}1^{n-j}) = 1$ for all $j \in [s]$.

Define $g : \{0,1\}^s \to \{0,1\}$ as $g(x) = f(x0^{n-s})$. Then g satisfies the hypotheses of Lemma 3.2. Hence, by Lemma 3.2, there exist $\ell = O(\log s)$ random restrictions $\mathbf{g}_1, \ldots, \mathbf{g}_\ell$ of g such that

$$\Pr_{\mathbf{g}_1,\ldots,\mathbf{g}_\ell}[\mathrm{OR}_\ell(\mathbf{g}_1(x), \ldots, \mathbf{g}_\ell(x)) \neq \mathrm{OR}_s(x)] \leq \frac{1}{10}, \quad \text{for all } x \in \{0,1\}^s. \tag{3.1}$$

We use the above representation to devise a probabilistic polynomial for OR_s.

Let $\mathbf{O}$ be any $(1/10)$-error probabilistic polynomial for OR_ℓ and $\mathbf{G}_1, \ldots, \mathbf{G}_\ell$ be any $(1/(10\ell))$-error probabilistic polynomials for $\mathbf{g}_1, \ldots, \mathbf{g}_\ell$ respectively. Then, by Fact 2.1 and (3.1), $\mathbf{O}(\mathbf{G}_1, \ldots, \mathbf{G}_\ell)$ is a $(1/3)$-error probabilistic polynomial for OR_s.

Note that by Theorem 2.3, we can choose $\mathbf{O}$ to have degree at most $O(\log \ell)$. Further, by Fact 2.1, we have $\mathrm{pdeg}(\mathbf{g}_i) \leq \mathrm{pdeg}(g) \leq \mathrm{pdeg}(f)$ for each $i \in [\ell]$. In particular, this implies that we can choose $\mathbf{G}_i$ to have degree $O(\mathrm{pdeg}(f) \cdot \log \ell)$ for each $i \in [\ell]$. This yields

$$\mathrm{pdeg}(\mathrm{OR}_s) \leq \mathrm{pdeg}(f) \cdot O(\log \ell)^2 = \mathrm{pdeg}(f) \cdot O((\log \log s)^2) = \mathrm{pdeg}(f) \cdot (\log s)^{o(1)}.$$

As $\mathrm{pdeg}(\mathrm{OR}_s) \geq (\log s)^{c - o(1)}$ by assumption, we get the desired lower bound on $\mathrm{pdeg}(f)$. □

It remains to prove Lemma 3.2, which we do now.

Proof of Lemma 3.2. We will only use restrictions g' of g obtained by setting some inputs of g to 0. The basic observation [8, 29] is the following. For any such restriction $g' = g_\rho$, the function g' always agrees with OR_s at the all-zero input. Moreover, if a is non-zero and has weight $t > 0$, then $g'(a) = \mathrm{OR}_s(a) = 1$ as long as exactly $t - 1$ of the variables that are set to 1 in a are fixed to 0 by ρ (this follows from the fact that g accepts any input of weight exactly 1). While we cannot always choose a *single* restriction that does this for all possible a, it is possible to choose a *small number of restrictions* randomly such that for each non-zero a, at least one of them is guaranteed to work with high probability. We now see the details.

For $i \in [\log s]$, let $\mathcal{D}_i$ be the distribution over subsets of $[s]$ where we pick each element independently to be in the set with probability 2^{-i}. For a (constant) parameter p to be chosen later, let $\mathbf{S}_1^i, \ldots, \mathbf{S}_p^i$ be independent random subsets picked from distribution $\mathcal{D}_i$. Each such set $\mathbf{S}_j^i$ is associated with the restriction ρ_j^i where each variable is set to 0 if it does not belong to $\mathbf{S}_j^i$, and left alive (i.e. set to $*$) otherwise. Note that $\mathbf{g}_j^i := g_{\rho_j^i}$ is a random restriction of g. Also observe that the total number of such restrictions is $\ell := p \log s = O(\log s)$. The final probabilistic representation is the OR of all these $\mathbf{g}_j^i$s.

We now prove correctness. Consider any $a \in \{0,1\}^s$. The case when $a = 0^s$ is easy, as each $\mathbf{g}_j^i$ is obtained by setting some inputs of g to 0 and hence $\mathbf{g}_j^i(a) = 0$ with probability 1. The same is therefore true for the OR of these functions.

Now assume that $a \neq 0$. Thus $|a| = t \in [s]$. Fix $i \in [\log s]$ such that $t \in (2^{i-1}, 2^i]$. We will show that, with probability at least 0.9, some $\mathbf{g}_j^i$ evaluates to 1. This will finish the proof.

To see this, let $S \subseteq [s]$ be the set of coordinates where a takes value 1. Note that $\mathbf{g}_j^i(a) = g(\mathbf{b}_j^i)$ where $\mathbf{b}_j^i$ denotes the indicator vector of $\mathbf{S}_j^i \cap S$. As $g(b) = 1$ for any input b of weight 1, we see that $\mathbf{g}_j^i(a) = 1$ if $|\mathbf{S}_j^i \cap S| = 1$. Hence, we have

$$\Pr_{\mathbf{g}_1^i,\ldots,\mathbf{g}_p^i}\left[\mathbf{g}_1^i(a) = \cdots = \mathbf{g}_p^i(a) = 0\right] \leq \Pr_{\mathbf{s}_1^i,\ldots,\mathbf{s}_p^i}\left[\bigwedge_{j=1}^{p} |\mathbf{S}_j^i \cap S| \neq 1\right] = \prod_{j=1}^{p} \Pr_{\mathbf{s}_j^i}\left[|\mathbf{S}_j^i \cap S| \neq 1\right], \tag{3.2}$$

where the last equality follows from the independence of the $\mathbf{S}_j^i$s.

Finally, note that for any j,

$$\Pr_{\mathbf{s}_j^i}\left[|\mathbf{S}_j^i \cap S| = 1\right] = \sum_{k \in S} \Pr_{\mathbf{s}_j^i}\left[k \in \mathbf{S}_j^i \wedge \bigwedge_{k' \in S \setminus k} k' \notin \mathbf{S}_j^i\right] = t \cdot \frac{1}{2^i} \cdot \left(1 - \frac{1}{2^i}\right)^{t-1} \geq \frac{1}{2} \cdot \left(1 - \frac{1}{2^i}\right)^{2^i - 1} \geq \frac{1}{2e},$$

where the first inequality follows from the fact that $t \in (2^{i-1}, 2^i]$ and the second from the standard fact that $(1 - 1/n)^{n-1} \geq 1/e$. Plugging the above into (3.2), we get

$$\Pr_{\mathbf{g}_1^i,\ldots,\mathbf{g}_p^i}\left[\mathbf{g}_1^i(a) = \cdots = \mathbf{g}_p^i(a) = 0\right] \leq \left(1 - \frac{1}{2e}\right)^p \leq \frac{1}{10},$$

for a large enough constant p. In particular, for this p, the probability that $\mathrm{OR}_\ell(\mathbf{g}_j^i : i \in [\ell], j \in [p])$ evaluates to 0 is at most $1/10$, completing the proof of the lemma. □

3.2 The case of small sensitivity

We prove the following lemma.

Lemma 3.3. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function of sensitivity at most s that depends on all its n variables. Then, we have* $\mathrm{pdeg}(f) = \Omega\left(\frac{\log(n/s^{O(1)})}{\log s}\right)$.

Remark 3.4. Note that the lower bound in Lemma 3.3 is trivial for large sensitivity, i.e. if $s = n^{\Omega(1)}$. However, we will use the lower bound for the case of low sensitivity, i.e. for $s = n^{o(1)}$. In this case, the lower bound simplifies to $\Omega(\log n / \log s)$.

The proof of Lemma 3.3 is in two steps. In the first step, we use a counting argument to prove a lower bound on the probabilistic degrees of random functions $\mathbf{F} : \{0,1\}^m \to \{0,1\}$ which are chosen from a distribution such that for a large subset $X \subseteq \{0,1\}^m$, the random variables $\{\mathbf{F}(x) \mid x \in X\}$ are independently and uniformly chosen random bits. In the second step, we show how any f as in the statement of Lemma 3.3 can be randomly restricted to a random function $\mathbf{F}$ where the lower bound for random functions applies.

We now state the lower bound for random functions.

Lemma 3.5 (Random function lower bound)**.** *The following holds for positive integer parameters m, M and d such that $M > m^{10d}$. Let $\mathbf{F} : \{0,1\}^m \to \{0,1\}$ be a random function such that for some $X \subseteq \{0,1\}^m$ with $|X| = M$, the random variables $(\mathbf{F}(x))_{x \in X}$ are independent and uniformly distributed random bits. Then, we have*

$$\Pr_{\mathbf{F}}\left[\mathrm{pdeg}_{1/10}(\mathbf{F}) \leq d\right] < \frac{1}{10}.$$

The proof of Lemma 3.5 uses fairly standard ideas and is deferred to Subsection 3.4. Let us now see how to use Lemma 3.5 to prove Lemma 3.3. This proof again breaks into two smaller steps.

Step 1: Show that, after a projection, f turns into something similar to an addressing function, that we will call a *Pseudoaddressing function.*

Step 2: Show that any pseudoaddressing function has large probabilistic degree.

As any projection g of f satisfies $\mathrm{pdeg}(g) \leq \mathrm{pdeg}(f)$ (Fact 2.1), the above implies a lower bound on $\mathrm{pdeg}(f)$, hence proving Lemma 3.3.

To make the above precise, we need the following definition. We say that a function $g : \{0,1\}^{r+t} \to \{0,1\}$ is an (r,t)-*Pseudoaddressing function* if the input variables to g can be partitioned into two sets $Y = \{y_1, \ldots, y_r\}$ and $Z = \{z_1, \ldots, z_t\}$ and g can be computed by a decision tree T with the following properties.

(P1) For each $z_j \in Z$, there are two root-to-leaf paths π_j^0 and π_j^1 in T that diverge at a node labeled z_j and lead to outputs 0 and 1 respectively.

(P2) All the other nodes on these paths are labeled by variables in Y, and further these variables take the same values on both paths. In particular, π_j^0 and π_j^1 differ only on the value of z_j.

Example 3.6. Consider the standard Addressing function Addr_r on $n = r + 2^r$ variables as defined in Section 1.3. This function is an $(r, 2^r)$-pseudoaddressing function as it can be computed by a decision tree of depth $r+1$, which first queries all the addressing variables to determine $a \in \{0,1\}^r$ and then queries and outputs the value of z_a (the two computational paths querying z_a give the desired root-to-leaf paths required in the definition above).

In analogy with the Addressing function, given an (r, t)-pseudoaddressing function as above, we refer to the variables in Y as the addressing variables and the variables in Z as the addressed variables.

The two steps of the proof as outlined above can now be formalized as follows.

Claim 3.7. *Let f be as in the statement of Lemma 3.3. Then, there exist $r \leq s^{O(1)}$ and $t \geq n/s^{O(1)}$ and a projection $\nu : [n] \to [r+t]$ such that $g = f|_\nu$ is an (r, t)-pseudoaddressing function.*

Claim 3.8. *Let g be any (r, t)-pseudoaddressing function. Then,* $\mathrm{pdeg}(g) = \Omega(\log t / \log r)$.

As noted above, the above claims immediately imply Lemma 3.3. We now prove these claims.

Proof of Claim 3.7. We will first outline how to isolate a set of $n/\mathrm{poly}(s)$ variables that will (almost) be the set of addressed variables. A projection will then be applied to the remaining variables to create the pseudoaddressing function. Let us now see the details.

By Theorem 2.7, we know that f has a decision tree T_f of depth $d \leq \mathrm{poly}(s)$. Fix such a tree T_f of *minimum size*, i.e. with the smallest possible number of leaves. Let $V = \{x_1, \ldots, x_n\}$ denote the input variables of f.

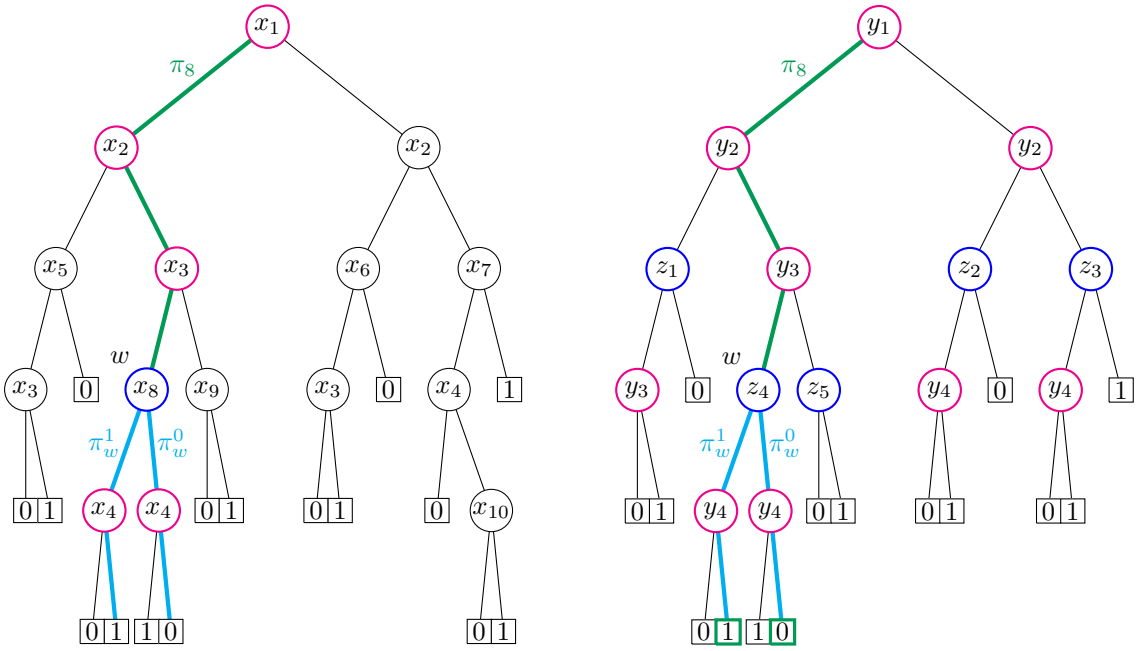
Any variable $x_i \in V$ must be queried somewhere in the tree T_f, as f depends on all its input variables by assumption. Fix any occurrence of this variable in the decision tree T_f, and let w denote the node of T_f corresponding to this query. (Refer to Figure 2 (a) for an illustration.) Let π_i denote the path from the root of T_f to w and let T_0 and T_1 be the subtrees rooted at the left and right children of w. The decision trees T_0 and T_1 both compute functions of the $n' < n$ Boolean variables not queried in π_i. Note that these decision trees compute *distinct* functions since otherwise the query made at the vertex w is unnecessary, and a smaller decision tree than T_f can be obtained by replacing the subtree rooted at w by T_0 or by T_1. This contradicts the minimality of the size of T_f.

Thus, T_0 and T_1 compute distinct functions. In particular, there is an input $a \in \{0,1\}^{n'}$ on which T_0 and T_1 have different outputs; w.l.o.g., assume T_0 and T_1 output 0 and 1 respectively on a. Let π_w^0 and π_w^1 be the root-to-leaf paths followed on the input a in T_0 and T_1 respectively. Note that any variable queried on both π_w^0 and π_w^1 takes the same value on both paths, as both paths are consistent with the input a. (Again, see Figure 2 (a) for an example.)

Concatenating each of π_w^0 and π_w^1 with the path π_i gives us two root-to-leaf paths π_i^0 and π_i^1 in T such that

- P1′ The paths π_i^0 and π_i^1 diverge at the node w (labelled by variable x_i) and lead to outputs 0 and 1 respectively.
- P2′ The two paths agree on all variables other than x_i, i.e., any other variable that is queried on π_i^0 and π_i^1 takes the same value on both.

We have such a pair of paths π_i^0 and π_i^1 for each $x_i \in V$. Let P_i denote the set of all $j \neq i$ such that x_j is queried on π_i^0 or on π_i^1. Note that $|P_i| \leq 2d$.



(a) The tree T_f (b) The tree T (after projection)

Figure 2: The decision tree on the left computes a truly 10-variate function $f(x_1, \ldots, x_{10})$. The paths obtained by concatenating π_8 with π_w^0 and π_w^1 are consistent with each other except for the value of x_8, the variable queried at node w. After a projection $\nu : [10] \to [9]$ defined by $\nu(i) = i$ for $i \leq 9$ and $\nu(10) = 4$, we get a tree T, which computes a $(4, 5)$-pseudoaddressing function $g(y_1, \ldots, y_4, z_1, \ldots, z_5)$. Note that each path in T corresponds to a path in T_f but not every path in T_f survives in T (e.g. the path leading to 0 through the node querying x_{10} is pruned away, as it is inconsistent with ν).

We claim that we can choose a large subset $Z' \subseteq [n]$ such that for all $i \in Z'$, the set P_i does not contain any j where $j \in Z'$. To see this, define a graph G with vertex $[n]$ and edges between vertices distinct $i, j \in [n]$ if and only if P_i contains j or vice-versa. Since each $|P_i| \leq 2d$, it is clear that this graph has average degree at most $2d$. By Turán's theorem (see e.g. [3]), this implies that G has an independent set Z' of size at least $n/4d$. This set Z' has the required property.

We are now ready to show that the required projection ν exists. Let $r = 10d^2$ and let $\nu' : [n] \setminus Z' \to [r]$ be a random map (i.e. the image of each element of the domain is independently and uniformly chosen from $[r]$). We say that an $i \in Z'$ is *good* if ν' is 1-1 on the set P_i. Let $\mathcal{G}$ be the set of all good i, with $\mathbf{t} := |\mathcal{G}|$. Assume $\mathcal{G} = \{i_1, \ldots, i_{\mathbf{t}}\}$. We use this to define a random projection $\nu : [n] \to [r + \mathbf{t}]$ by

$$\nu(i) = \begin{cases} \nu'(i) & \text{if } i \notin Z', \\ 1 & \text{if } i \in Z' \setminus \mathcal{G}, \text{ (here, any } k \in [r] \text{ will do)} \\ r + j & \text{if } i \in \mathcal{G} \text{ and } i = i_j. \end{cases}$$

The random projection defines a random Boolean function $\mathbf{g}$ on $r + \mathbf{t}$ variables. We now show

that, with positive probability, $\mathbf{g}$ is an $(r, n/\mathrm{poly}(s))$-pseudoaddressing function, where the first r variables are the addressing variables. This will finish the proof. Note that the projection ν applied to the tree T_f also defines a random decision tree $\mathbf{T}$ computing $\mathbf{g}$. We will in fact show that $\mathbf{T}$ serves as a witness for the fact that $\mathbf{g}$ is an $(r, n/\mathrm{poly}(s))$-pseudoaddressing function (with positive probability).

In fact, this happens whenever $\mathbf{t} = |\mathcal{G}|$ is large enough. More precisely, note that

$$\mathbb{E}\nu'|Z'| - \mathbf{t} = \sum_{i \in Z'} \Pr_{\nu'}[\nu' \text{ is not 1-1 on } P_i] \leq \sum_{i \in Z'} \sum_{j \neq k \in P_i} \Pr_{\nu'}[\nu'(j) = \nu'(k)] \leq \sum_{i \in Z'} |P_i|^2 \cdot \frac{1}{r} \leq |Z'| \cdot \frac{(2d)^2}{r} \leq \frac{|Z'|}{2}.$$

In particular, there is a setting ν' of ν' such that the corresponding set of good variables $|\mathcal{G}|$ has size at least $|Z'|/2$. Fix this ν' and let $\mathcal{G}, t, \nu, g, T$ be the corresponding fixings of $\mathcal{G}, \mathbf{t}, \nu, \mathbf{g}, \mathbf{T}$ respectively.

We have $g = g(y_1, \ldots, y_r, z_1, \ldots, z_t)$. Observe that each root-to-leaf path of T can be identified with a root-to-leaf path of T_f. Further, a path π of T_f survives in T exactly when it is *consistent w.r.t.* ν, i.e., if two variables that are set to opposite values in π are not mapped to the same variable by ν (see Figure 2 (b) for an example). In particular, if a path π has the property that the variables queried along π are mapped injectively by ν, then the path π survives in T.

This implies that for any good $i_j \in \mathcal{G}$, the corresponding paths $\pi^0_{i_j}$ and $\pi^1_{i_j}$ survive in T. Moreover, as the projection ν is injective on the entire set P_{i_j}, these paths continue to agree with each other on all variables except the variable z_j queried at the point of their divergence. This gives both properties P1 and P2 stated above. As this holds for each $i_j \in \mathcal{G}$, we see that g is indeed an (r, t)-pseudoaddressing function. Note that $r = 10d^2 \leq \mathrm{poly}(s)$ and $t \geq n/4d \geq n/\mathrm{poly}(s)$. Hence, we have proved the claim. □

Proof of Claim 3.8. The proof is via a reduction to Lemma 3.5.

Let $g(y_1, \ldots, y_r, z_1, \ldots, z_t)$ be an (r, t)-pseudoaddressing function. Consider the random function $\mathbf{F}$ on $\{0,1\}^r$ obtained by setting the addressed variables $z_1, \ldots, z_t$ to $\mathbf{b}_1, \ldots, \mathbf{b}_t \in \{0,1\}$ chosen i.u.a.r.. We show that there is an $X \subseteq \{0,1\}^r$ of size t such that the random variables $(\mathbf{F}(a) : a \in X)$ are independent and uniformly distributed bits. Then, Lemma 3.5 implies the statement of the claim.

Let us see how X is defined. Let T be the decision tree guaranteed for g by virtue of the fact that it is an (r, t)-pseudoaddressing function. Further, for any z_j, let π^0_j and π^1_j be the paths satisfying P1 and P2 above. By P2, we can fix a setting $a^{(j)} \in \{0,1\}^r$ to the y-variables that is consistent with both paths. We set $X = \{a^{(j)} \mid j \in [t]\}$.

To analyze $\mathbf{F}(a^{(j)})$, note that setting the variables $z_1, \ldots, z_t$ to $\mathbf{b}_1, \ldots, \mathbf{b}_t$ in T gives us a (random) decision tree $\mathbf{T}'$ that computes $\mathbf{F}$. In particular, the path followed by $\mathbf{T}'$ on input $a^{(j)}$ is uniformly chosen among π^0_j and π^1_j depending on the value of z_j, and hence $\mathbf{F}(a^{(j)})$ is either $\mathbf{b}_j$ or $1 - \mathbf{b}_j$ (exactly which depends on the value of z_j that is consistent with π^0_j and π^1_j). In either case, however, $\mathbf{F}(a^{(j)})$ is a uniformly chosen random bit depending only on $\mathbf{b}_j$. Hence, the random variables $(\mathbf{F}(a^{(j)}) : j \in [t])$ are independent and uniformly distributed.

Thus, Lemma 3.5 implies that with positive probability, $\mathrm{pdeg}_{1/10}(\mathbf{F}) = \Omega(\log t/\log r)$. However, we know by Fact 2.1 that, as $\mathbf{F}$ is a restriction of g, $\mathrm{pdeg}_{1/10}(\mathbf{F}) \le \mathrm{pdeg}_{1/10}(g)$. Hence, we obtain the same lower bound for $\mathrm{pdeg}_{1/10}(g)$. Finally, by error reduction (Fact 2.1), the same lower bound (up to constant factors) holds for $\mathrm{pdeg}_{1/3}(g) = \mathrm{pdeg}(g)$. □

3.3 Finishing the proof of Theorem 1.4

Lemma 3.3 and Lemma 3.1 imply that

$$\mathrm{pdeg}(f) = \Omega\left(\max\left\{(\log s)^{c-o(1)}, \frac{\log(n/s^{O(1)})}{\log s}\right\}\right)$$

where s denotes the sensitivity of f. The above is minimized for s so that $(\log s)^{c+1} = \Theta(\log n)$ (note that this implies that $s = n^{o(1)}$). For this s, we get

$$\mathrm{pdeg}(f) = \Omega((\log n)^{c/(c+1)-o(1)}) \ge (\log n)^{c/(c+1)-o(1)},$$

proving the theorem.

3.4 Proof of the Random function lower bound (Lemma 3.5)

Let us conclude this section by proving Lemma 3.5. The proof is via a counting argument. Recall that we have positive integer parameters m, M and d such that $M > m^{10d}$, and a random function $\mathbf{F} : \{0,1\}^m \to \{0,1\}$ such that for some $X \subseteq \{0,1\}^m$ with $|X| = M$, the random variables $(\mathbf{F}(x))_{x\in X}$ are independent and uniformly distributed random bits.

We start with a standard observation, which follows from a simple averaging argument. If $G : \{0,1\}^m \to \{0,1\}$ has $(1/10)$-error probabilistic degree d, then for any probability distribution μ over $\{0,1\}^m$, there is a polynomial P of degree at most d such that

$$\Pr_{a\sim\mu}[P(a) = G(a)] \ge \frac{9}{10}. \tag{3.3}$$

Conversely, by Yao's min-max theorem [32] if there is a probability distribution μ (which we call the *hard distribution*) such that (3.3) does not hold for any polynomial of degree at most d, then $\mathrm{pdeg}(G) > d$. For our proof, we will take the hard distribution to be the uniform distribution over X.

More precisely, call a function $g : X \to \{0,1\}$ *bad* if there is a polynomial P of degree at most d that agrees with g on at least $9|X|/10 = 9M/10$ points of X. Let $\mathcal{B}$ be the set of bad functions. The reasoning above tells us that

$$\Pr_{\mathbf{F}}\left[\mathrm{pdeg}_{1/10}(\mathbf{F}) \le d\right] \le \Pr_{\mathbf{F}}[\mathbf{F}|_X \in \mathcal{B}] = \frac{|\mathcal{B}|}{2^M}. \tag{3.4}$$

where for the latter inequality we have used the fact that the random variables $(\mathbf{F}(x) : x \in X)$ are independently and uniformly distributed. Hence, it will suffice to bound $|\mathcal{B}|$ to prove the lemma.

To bound the size of $\mathcal{B}$, it will suffice to give a short encoding of each element of $\mathcal{B}$. Fix any $g \in \mathcal{B}$ and a polynomial P that agrees with g on a set $X' \subseteq X$ such that $|X'| \geq 9M/10$. Note that g can be specified by

1. The set X'.

2. The set of values of g on $X \setminus X'$ (in some pre-determined order).

3. A polynomial Q of degree at most d that agrees with g on X' (specified as a list of coefficients of monomials).

Note that the number of choices for X' is at most $\binom{M}{\leq M/10}$, which is bounded by $2^{H(1/10)M}$, where $H(\cdot)$ denotes the binary entropy function. Further, the number of possibilities for g on $X \setminus X'$ is at most $2^{|X \setminus X'|} \leq 2^{M/10}$.

It remains to bound the number of possibilities for Q. A priori, it is not completely clear how to bound the number of Q as the coefficients of Q could be arbitrary real numbers. However, we note that if there is a polynomial P that agrees with g on X', then there is also a Q that satisfies this property, and furthermore, the coefficients of Q are rational numbers of small bit complexity.

Formally, we will use the following lemma, which is an easy consequence of [25, Corollary 3.2d].

Lemma 3.9. *Consider a system of linear equations $Ax = b$ over the rational numbers, where A is an $p \times q$* Boolean *matrix, and $b \in \{0,1\}^p$. Then, if the system has a real solution, it has a rational solution that can be specified (as a list of numerator-denominator pairs in binary) by at most $10q^3$ bits.*

To use the above lemma, consider the problem of finding a polynomial Q of degree at most d that agrees with g at all points in X'. The coefficients of such a polynomial Q solve a linear system of $p := |X'|$ many linear equations in $q := \binom{m}{\leq d}$ variables. By the existence of the polynomial P, this system has a solution. Thus by Lemma 3.9, we know that there is a solution of bit-complexity at most $10q^3 \leq m^{4d} < M/10$. Therefore, we may always choose Q from the set $\mathcal{Q}$ of polynomials of bit-complexity (as specified above) at most $M/10$. Note that $|\mathcal{Q}| \leq 2^{M/10}$ by definition.

Overall, this gives a complete specification of any given $g \in \mathcal{B}$. More precisely, we have given a 1-1 map $\tau : \mathcal{B} \to \mathcal{X} \times S \times \mathcal{Q}$, where $\mathcal{X}$ is the collection of subsets of X of size at least $9M/10$, S is the set of Boolean tuples of length $M/10$, and $\mathcal{Q}$ is the set of polynomials of degree at most d of bit-complexity at most $M/10$. Hence, $|\mathcal{B}| \leq |\mathcal{X}| \cdot |S| \cdot |\mathcal{Q}| \leq 2^{M \cdot (H(1/10)+1/10+1/10)} \leq 2^{9M/10}$. Plugging this into (3.4), we get

$$\Pr_{\mathbf{F}}\left[\mathrm{pdeg}_{1/10}(\mathbf{F}) \leq d\right] \leq \frac{2^{9M/10}}{2^M} < \frac{1}{10}.$$

This finishes the proof of the lemma.

4 The Upper Bound: Proof of Theorem 1.5

The construction is motivated by and closely follows a construction of Ambainis and de Wolf [2], who used it to prove the existence of a truly n-variate Boolean function f whose approximate degree is $O(\log n/\log\log n)$. The construction of [2] uses the fact that the approximate degree of the OR_n function is $O(\sqrt{n})$ [15]. Using our assumption that the probabilistic degree of the OR_n function is $(\log n)^{c+o(1)}$ we are able to prove a stronger degree upper bound for probabilistic degree. In particular, Theorem 2.3 allows us to prove an unconditional upper bound of $(\log n)^{(1/2)+o(1)}$ on the probabilistic degree of some n-variable function.

The construction is a variant of the Addressing function, where the addressing bits are replaced by elements of a larger alphabet $[s]$, which are themselves presented in an encoded form that allows them to be easily 'decoded' by low-degree polynomials. More precisely, we construct the function as follows.

Construction. Let s be a power of 2 and let $H \subseteq \{0,1\}^s$ be the set of codewords of the Hadamard code. That is, assume $s = 2^t$ and identify elements of $\{0,1\}^s$ with functions $h : \{0,1\}^t \to \{0,1\}$. Then H consists of precisely those elements $h \in \{0,1\}^s$ such that h is a linear function when considered as a mapping from $\mathbb{F}_2^s$ to $\mathbb{F}_2$ in the natural way. The set H contains precisely s elements, say $\{h_1, \ldots, h_s\}$.

We define a Boolean function f on $n = sr + s^r + 1$ bits as follows. Any input a is parsed as

$$a = (g_1, \ldots, g_r, T, b)$$

where $g_1, \ldots, g_r : \{0,1\}^t \to \{0,1\}$, $T : [s]^r \to \{0,1\}$ and b is a single bit. We define f by

$$f(a) = \begin{cases} T(i_1, \ldots, i_r) & \text{if } g_1, \ldots, g_r \in H \text{ and } g_1 = h_{i_1}, \ldots, g_r = h_{i_r}, \\ b & \text{otherwise.} \end{cases}$$

Analysis. We have

$$f(g_1, \ldots, g_r, T, b) = \sum_{i_1, \ldots, i_r \in [s]} 1(g_1 = h_{i_1}, \ldots, g_r = h_{i_r}) \cdot T(i_1, \ldots, i_r) + (1 - 1(g_1, \ldots, g_r \in H)) \cdot b. \tag{4.1}$$

Here $1(\mathcal{E})$ for a Boolean predicate $\mathcal{E}$ takes the value 1 when the Boolean predicate is satisfied and 0 otherwise.

The above implies, in particular, that the function f is truly n-variate. To see this, say the variables of f are

- $x_{j,\alpha}$ ($j \in [r]$, $\alpha \in \{0,1\}^t$) encoding the entries of the truth tables of $g_1, \ldots, g_r$, More formally, the variable $x_{j,\alpha}$ is set to $g_j(\alpha)$.
- $y_{i_1,\ldots,i_r}$ encoding the entries of T, and
- y_0 which gives the value of b.

Any variable $x_{j,\alpha}$ is influential at an input $(g_1, \ldots, g_r, T, b)$ where $g_1, \ldots, g_r$ are $h_{i_1}, \ldots, h_{i_r} \in H$ respectively, and $b \neq T(i_1, \ldots, i_r)$, which implies that flipping the value of $x_{j,\alpha}$ at this point changes the output from $T(i_1, \ldots, i_r)$ to b. The variable $y_{i_1,\ldots,i_r}$ is also influential at the same point. The variable y_0 is influential at any input where not all the g_i are in H. Thus, we see that f is indeed n-variate.

Now, we will show an upper bound on $\mathrm{pdeg}(f)$. This will be done by constructing two polynomials.

- A 1/3-error probabilistic polynomial $\mathbf{Q}(x_{j,\alpha} : j \in [r], \alpha \in \{0,1\}^t)$ for the Boolean function $1(g_1, \ldots, g_r \in H)$.
- For each $i_1, \ldots, i_r \in [s]$, a polynomial $R_{i_1,\ldots,i_r}(x_{j,\alpha} : j \in [r], \alpha \in \{0,1\}^t)$ such that at input $(g_1, \ldots, g_r) \in H^r$, $R_{i_1,\ldots,i_r}(g_1, \ldots, g_r) = 1$ if $g_1 = h_{i_1}, \ldots, g_r = h_{i_r}$, and 0 otherwise. (In other words, $R_{i_1,\ldots,i_r}$ computes a δ-function on inputs from H^r. Note that we do not claim anything if $(g_1, \ldots, g_r) \notin H^r$.)

Given the above constructions, the following yields a probabilistic polynomial $\mathbf{P}$ for f.

$$\mathbf{P} = \mathbf{Q} \cdot \left(\sum_{i_1,\ldots,i_r \in [s]} R_{i_1,\ldots,i_r} \cdot y_{i_1,\ldots,i_r} \right) + (1 - \mathbf{Q}) \cdot y_0 \tag{4.2}$$

(The two copies of $\mathbf{Q}$ are chosen with the same randomness and are *not* independent of each other.) To see that this works, fix any input $a = (g_1, \ldots, g_r, T, b)$. If $(g_1, \ldots, g_r) \in H^r$, the term in the parenthesis evaluates to $T(i_1, \ldots, i_r)$ with probability 1. Further, $\mathbf{Q}(g_1, \ldots, g_r)$ evaluates to 1 with probability 2/3. Hence, $\mathbf{P}(a) = T(i_1, \ldots, i_r) = f(a)$ with probability at least 2/3. On the other hand, if $(g_1, \ldots, g_r) \notin H^r$, then $\mathbf{Q}(g_1, \ldots, g_r)$ evaluates to 0 with probability 2/3. When this event occurs, the first summand evaluates to 0 and the second summand evaluates to b. Hence, $\mathbf{P}(a) = b = f(a)$ with probability at least 2/3.

It remains to construct the polynomials $\mathbf{Q}$ and $R_{i_1,\ldots,i_r}$. We start with $\mathbf{Q}$. Recall that a function $g : \{0,1\}^t \to \{0,1\}$ lies in H when it is linear over $\mathbb{F}_2$, or equivalently if $g(\alpha \oplus \beta) \oplus g(\alpha) \oplus g(\beta) = 0$ for every $\alpha, \beta \in \{0,1\}^t$. Thus, the condition that $g_1, \ldots, g_r \in H$ can be rewritten as

$$\bigwedge_{j=1}^{r} \bigwedge_{\alpha,\beta \in \{0,1\}^t} (1 \oplus g_j(\alpha \oplus \beta) \oplus g_j(\alpha) \oplus g_j(\beta)).$$

Let $q(z_1, z_2, z_3)$ be a constant-degree polynomial of 3 Boolean variables that evaluates to $1 \oplus z_1 \oplus z_2 \oplus z_3$. Then, the above can be rewritten as $\bigwedge_{j=1}^{r} \bigwedge_{\alpha,\beta \in \{0,1\}^t} q(g_j(\alpha), g_j(\beta), g_j(\alpha \oplus \beta))$. Thus, we can define the probabilistic polynomial to be

$$\mathbf{Q}(x_{j,\alpha} : j \in [r], \alpha \in \{0,1\}^t) = \mathbf{Q}_1(q(x_{j,\alpha}, x_{j,\beta}, x_{j,\alpha\oplus\beta}) : j \in [r], \alpha, \beta \in \{0,1\}^t)$$

where Q_1 is any probabilistic polynomial for the $\mathrm{AND}_{r2^{2t}} = \mathrm{AND}_{rs^2}$ function. By assumption, $\mathrm{pdeg}(\mathrm{OR}_{rs^2})$ and hence, by DeMorgan's laws, $\mathrm{pdeg}(\mathrm{AND}_{rs^2})$ is at most $\log(rs^2)^{c+o(1)} = (\log r + \log s)^{c+o(1)}$.

We now see how to construct $R_{i_1,\dots,i_r}$ for any fixed $i_1,\dots,i_r \in [s]$. Recall the standard fact (see, e.g. [23]) that for $h_{i_1} \neq h_{i_2} \in H$, the functions $\hat{h}_{i_1}, \hat{h}_{i_2} : \{0,1\}^t \to \{-1,1\}$ defined by

$$\hat{h}_{i_b}(\alpha) = 1 - 2h_{i_b}(\alpha), \qquad \text{for all } \alpha \in \{0,1\}^t \text{ and } b \in \{1,2\},$$

are orthogonal to one another, i.e., $\sum_\alpha \hat{h}_{i_1}(\alpha)\hat{h}_{i_2}(\alpha) = 0$. Based on this observation, we define the polynomial as follows.

$$R_{i_1,\dots,i_r}(x_{j,\alpha} : j \in [r], \alpha \in \{0,1\}^t) = \frac{1}{s^r}\prod_{j=1}^{r}\left(\sum_{\alpha\in\{0,1\}^t} \hat{h}_{i_j}(\alpha)(1-2x_{j,\alpha})\right).$$

Let us see that this polynomial has the desired properties. Consider input $(g_1,\dots,g_r) \in H^r$. Assume $g_j = h_{i'_j}$ for each $j \in [r]$. Then, we have

$$\begin{aligned} R_{i_1,\dots,i_r}(g_1,\dots,g_r) &= \frac{1}{s^r}\prod_{j=1}^{r}\left(\sum_{\alpha\in\{0,1\}^t} \hat{h}_{i_j}(\alpha)(1-2h_{i'_j}(\alpha))\right) \\ &= \frac{1}{s^r}\prod_{j=1}^{r}\left(\sum_{\alpha\in\{0,1\}^t} \hat{h}_{i_j}(\alpha)\hat{h}_{i'_j}(\alpha)\right) \end{aligned}$$

and the latter quantity can be seen to be 1 if $i'_j = i_j$ for all $j \in [r]$ and 0 otherwise. Thus, $R_{i_1,\dots,i_r}$ behaves as stipulated. Note that $\deg(R_{i_1,\dots,i_r}) = r$.

This concludes the construction of the probabilistic polynomial for f. The degree of the polynomial thus constructed is at most $\deg(\mathbf{Q}) + \max_{i_1,\dots,i_r} \deg(R_{i_1,\dots,i_r}) = O((\log r + \log s)^{c+o(1)} + r) = O((\log s)^{c+o(1)} + r)$.

Parameters. We set $r = (\log s)^c = t^c$. This gives a truly n-variate Boolean function on $n = O(s^r) = O(2^{t^{1+c}})$ variables with probabilistic degree $t^{c+o(1)} = (\log n)^{(c/(c+1))+o(1)}$.

Acknowledgements

The authors thank the anonymous reviewers of the journal, as well as the anonymous reviewers for the conference RANDOM 2021, for their critical feedback.

References

[1] JOSH ALMAN, TIMOTHY M. CHAN, AND R. RYAN WILLIAMS: Polynomial representations of threshold functions and algorithmic applications. In *Proc. 57th FOCS*, pp. 467–476. IEEE Comp. Soc., 2016. [doi:10.1109/FOCS.2016.57] 4

[2] JOSH ALMAN AND R. RYAN WILLIAMS: Probabilistic polynomials and Hamming nearest neighbors. In *Proc. 56th FOCS*, pp. 136–150. IEEE Comp. Soc., 2015. [doi:10.1109/FOCS.2015.18] 21

[3] NOGA ALON AND JOEL H. SPENCER: *The Probabilistic Method*. Wiley, 3rd edition, 2008. [doi:10.1002/9780470277331] 17

[4] ANDRIS AMBAINIS AND RONALD DE WOLF: How low can approximate degree and quantum query complexity be for total Boolean functions? *Comput. Complexity*, 23(2):305–322, 2014. Preliminary version in CCC'13. [doi:10.1007/s00037-014-0083-2] 3, 6

[5] JAMES ASPNES, RICHARD BEIGEL, MERRICK L. FURST, AND STEVEN RUDICH: The expressive power of voting polynomials. *Combinatorica*, 14(2):135–148, 1994. [doi:10.1007/BF01215346] 4

[6] ROBERT BEALS, HARRY BUHRMAN, RICHARD CLEVE, MICHELE MOSCA, AND RONALD DE WOLF: Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. [doi:10.1145/502090.502097] 6

[7] RICHARD BEIGEL: The polynomial method in circuit complexity. In *Proc. 8th IEEE Conf. Structure in Complexity Theory (SCT'93)*, pp. 82–95. IEEE Comp. Soc., 1993. [doi:10.1109/SCT.1993.336538] 2, 4

[8] RICHARD BEIGEL, NICK REINGOLD, AND DANIEL SPIELMAN: The perceptron strikes back. In *Proc. 6th IEEE Conf. Structure in Complexity Theory (SCT'91)*, pp. 286–291. IEEE Comp. Soc., 1991. [doi:10.1109/SCT.1991.160270] 4, 5, 6, 7, 11, 13

[9] MARK BRAVERMAN: Polylogarithmic independence fools AC^0 circuits. *J. ACM*, 57(5):28:1–10, 2010. [doi:10.1145/1754399.1754401] 2, 4

[10] HARRY BUHRMAN AND RONALD DE WOLF: Complexity measures and decision tree complexity: a survey. *Theoret. Comput. Sci.*, 288(1):21–43, 2002. [doi:10.1016/S0304-3975(01)00144-X] 2, 3, 7

[11] MARCO L. CARMOSINO, RUSSELL IMPAGLIAZZO, VALENTINE KABANETS, AND ANTONINA KOLOKOLOVA: Learning algorithms from natural proofs. In *Proc. 31st Comput. Complexity Conf. (CCC'16)*, pp. 10:1–24. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. [doi:10.4230/LIPIcs.CCC.2016.10] 2, 4

[12] XI CHEN, IGOR CARBONI OLIVEIRA, ROCCO A. SERVEDIO, AND LI-YANG TAN: Near-optimal small-depth lower bounds for small distance connectivity. In *Proc. 48th STOC*, pp. 612–625. ACM Press, 2016. [doi:10.1145/2897518.2897534] 9

[13] JOHN CHIARELLI, POOYA HATAMI, AND MICHAEL E. SAKS: An asymptotically tight bound on the number of relevant variables in a bounded degree Boolean function. *Combinatorica*, 40(2):237–244, 2020. [doi:10.1007/s00493-019-4136-7] 3, 6

[14] ALEXANDER GOLOVNEV, ALEXANDER S. KULIKOV, AND R. RYAN WILLIAMS: Circuit depth reductions. In *Proc. 12th Innovations in Theoret. Comp. Sci. Conf. (ITCS'21)*, pp. 24:1–20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.ITCS.2021.24] 4

[15] LOV K. GROVER: A fast quantum mechanical algorithm for database search. In *Proc. 28th STOC*, pp. 212–219. ACM Press, 1996. [doi:10.1145/237814.237866] 4, 21

[16] PRAHLADH HARSHA AND SRIKANTH SRINIVASAN: On polynomial approximations to AC^0. *Random Struct. Algor.*, 54(2):289–303, 2019. [doi:10.1002/rsa.20786] 4, 5, 6, 10, 11

[17] JOHAN HÅSTAD, BENJAMIN ROSSMAN, ROCCO A. SERVEDIO, AND LI-YANG TAN: An average-case depth hierarchy theorem for Boolean circuits. *J. ACM*, 64(5):35:1–27, 2017. [doi:10.1145/3095799] 9

[18] HAO HUANG: Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Ann. Math.*, 190(3):949–955, 2019. [doi:10.4007/annals.2019.190.3.6] 7, 12

[19] ADAM R. KLIVANS AND ROCCO A. SERVEDIO: Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *J. Comput. System Sci.*, 68(2):303–318, 2004. [doi:10.1016/j.jcss.2003.07.007] 2

[20] MICHAEL MITZENMACHER AND ELI UPFAL: *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge Univ. Press, USA, 2nd edition, 2017. [doi:10.1017/CBO9780511813603] 11

[21] NOAM NISAN: CREW PRAMs and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991. Preliminary version in STOC'89. [doi:10.1137/0220062] 7, 12

[22] NOAM NISAN AND MARIO SZEGEDY: On the degree of Boolean functions as real polynomials. *Comput. Complexity*, 4(4):301–313, 1994. [doi:10.1007/BF01263419] 3, 4, 6

[23] RYAN O'DONNELL: *Analysis of Boolean Functions*. Cambridge Univ. Press, USA, 2014. [doi:10.1017/CBO9781139814782] 2, 23

[24] ALEXANDER A. RAZBOROV: Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mat. Zametki*, 41(4):598–607, 1987. An English translation in Math. Notes, 41(4):333–338, 1987. [doi:10.1007/BF01137685] 4

[25] ALEXANDER SCHRIJVER: *Theory of Linear and Integer Programming*. Wiley, 1998. Available at Wiley. 20

[26] HANS-ULRICH SIMON: A tight $\omega(\log\log n)$-bound on the time for parallel RAM's to compute nondegenerated Boolean functions. *Inform. Control*, 55(1–3):102–106, 1982. Preliminary version in FCT'93. [doi:10.1016/S0019-9958(82)90477-6] 2, 3, 6

[27] ROMAN SMOLENSKY: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In ALFRED V. AHO, editor, *Proc. 19th STOC*, pp. 77–82. ACM Press, 1987. [doi:10.1145/28395.28404] 4

[28] SRIKANTH SRINIVASAN AND S. VENKITESH: On the probabilistic degree of an n-variate Boolean function. In *Proc. 25th Internat. Conf. on Randomization and Computation (RANDOM'21)*, pp. 42:1–20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.APPROX/RANDOM.2021.42] 1

[29] JUN TARUI: Probabilistic polynomials, AC^0 functions and the polynomial-time hierarchy. *Theoret. Comput. Sci.*, 113(1):167–183, 1993. [doi:10.1016/0304-3975(93)90214-E] 4, 5, 6, 7, 11, 13

[30] EMANUELE VIOLA: New lower bounds for probabilistic degree and AC^0 with parity gates. *Electron. Colloq. Comput. Complexity*, TR20-015, 2020. [ECCC] 4

[31] R. RYAN WILLIAMS: Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. [doi:10.1137/15M1024524] 2, 4

[32] ANDREW CHI-CHIH YAO: Probabilistic computations: Toward a unified measure of complexity. In *Proc. 18th FOCS*, pp. 222–227. IEEE Comp. Soc., 1977. [doi:10.1109/SFCS.1977.24] 19

AUTHORS

Srikanth Srinivasan
Associate Professor
Department of Computer Science
Aarhus University
Aarhus, Denmark
srikanth@cs.au.dk
https://cs.au.dk/~srikanth

S. Venkitesh
Postdoctoral Fellow
Department of Computer Science
University of Haifa
Haifa, Israel
venkitesh.mail@gmail.com
https://sites.google.com/view/venkitesh

ABOUT THE AUTHORS

SRIKANTH SRINIVASAN got his undergraduate degree from the Indian Institute of Technology Madras, where his interest in the theory side of CS was piqued under the tutelage of N. S. Narayanswamy. Subsequently, he obtained his Ph. D. from The Institute of Mathematical Sciences in 2011, where his advisor was V. Arvind. His research interests span all of TCS (in theory), but in practice are limited to circuit complexity, derandomization, and related areas of mathematics. He looks back fondly at the days when he enjoyed running and pretending to play badminton.

S. VENKITESH obtained his Bachelors' degree in Mathematics at the University of Mumbai, and his Masters' degree in Mathematics at the Indian Institute of Technology Bombay. His first mentor in mathematics from the Bachelors' times, Shriprasad Tambe, continues to be a constant source of encouragement for him. His affinity with Mumbai continued for many more years, and he completed his Ph.D. in Mathematics at IIT Bombay, advised by the terrific Srikanth Srinivasan. He is forever grateful to Srikanth for initiating him into theory CS in general, and computational complexity in particular. His current research interests are Boolean function complexity, error-correcting codes, and algebraic methods in combinatorics. In his dreams, he is a gifted singer, photographer, filmmaker, and tennis player. In reality, well, he manages to do math just about fine.