Complexity Measures and Decision Tree Complexity: A Survey

Harry Buhrman [a,1] Ronald de Wolf [a,b,1]

[a] *CWI, P.O. Box 94079, Amsterdam, The Netherlands, {buhrman,rdewolf}@cwi.nl.*

[b] *University of Amsterdam*

Abstract

We discuss several complexity measures for Boolean functions: certificate complexity, sensitivity, block sensitivity, and the degree of a representing or approximating polynomial. We survey the relations and biggest gaps known between these measures, and show how they give bounds for the decision tree complexity of Boolean functions on deterministic, randomized, and quantum computers.

1 Introduction

Computational Complexity is the subfield of Theoretical Computer Science that aims to understand "how much" computation is necessary and sufficient to perform certain computational tasks. For example, given a computational problem it tries to establish tight upper and lower bounds on the length of the computation (or on other resources, like space).

Unfortunately, for many, practically relevant, computational problems no tight bounds are known. An illustrative example is the well known P versus NP problem: for all NP-complete problems the current upper and lower bounds lie exponentially far apart. That is, the best known algorithms for these computational problems need exponential time (in the size of the input) but the best lower bounds are of a linear nature.

One of the general approaches towards solving a hard problem (mathematical or otherwise) is to set the goals a little bit lower and try to tackle a simpler

[1] Partially supported by the EU fifth framework project QAIP, IST–1999–11234.

problem first. The hope is that understanding of the simpler problem will lead to a better understanding of the original, more difficult, problem.

This approach has been taken with respect to Computational Complexity: simpler and more limited models of computation have been studied. Perhaps the simplest model of computation is the *decision tree*. The goal here is to compute a Boolean function $f : \{0,1\}^n \to \{0,1\}$ using queries to the input. In the most simple form a query asks for the value of the bit x_i and the answer is this value. (The queries may be more complicated. In this survey we will only deal with this simple type of query.) The algorithm is adaptive, that is the kth query may depend on the answers of the $k-1$ previous queries. The algorithm can therefore be described by a binary tree, whence its name 'decision tree'.

For a Boolean function f we define its deterministic decision tree complexity, $D(f)$, as the minimum number of queries that an optimal deterministic algorithm for f needs to make on any input. This measure corresponds to the depth of the tree that an optimal algorithm induces. Once the computational power of decision trees is better understood, one can extend this notion to more powerful models of query algorithms. This results in *randomized* and even *quantum* decision trees.

In order to get a handle on the computational power of decision trees (whether deterministic, randomized, or quantum), other measures of the complexity of Boolean functions have been defined and studied. Some prime examples are *certificate complexity*, *sensitivity*, *block sensitivity*, the *degree of a representing polynomial*, and the *degree of an approximating polynomial*. We survey the known relations and biggest gaps between these complexity measures and show how they apply to decision tree complexity, giving proofs of some of the central results. The main results say that all of these complexity measures (with the possible exception of sensitivity) are polynomially related to each other and to the decision tree complexities in each of the classical, randomized, and quantum settings. We also identify some of the main remaining open questions. The complexity measures discussed here also have interesting relations with circuit complexity [47,4,7], parallel computing [10,41,31,47], communication complexity [33,9], and the construction of oracles in computational complexity theory [6,43,15,16], which we will not discuss here.

The paper is organized as follows. In Section 2 we introduce some notation concerning Boolean functions and multivariate polynomials. In Section 3 we define the three main variants of decision trees that we discuss: deterministic decision trees, randomized decision trees, and quantum decision trees. In Section 4 we introduce certificate complexity, sensitivity, block sensitivity, and the degree of a representing or approximating polynomial. We survey the main relations and known upper and lower bounds between these measures. In Section 5 we show how the complexity measures of Section 4 imply upper and lower bounds on deterministic, randomized, and quantum decision tree complexity. This section gives bounds that apply to all Boolean functions. Finally, in Section 6 we examine some special subclasses of Boolean functions and tighten the general bounds of Section 5 for those special cases.

2 Boolean Functions and Polynomials

2.1 Boolean functions

A Boolean function is a function $f : \{0,1\}^n \to \{0,1\}$. Note that f is *total*, i.e., defined on all n-bit inputs. For an input $x \in \{0,1\}^n$, we use x_i to denote its ith bit, so $x = x_1 \dots x_n$. We use $|x|$ to denote the Hamming weight of x (its number of 1s). If S is a set of (indices of) variables, then we use x^S to denote the input obtained by flipping the S-variables in x. We abbreviate $x^{\{i\}}$ to x^i. For example, if $x = 0011$, then $x^{\{2,3\}} = 0101$ and $x^4 = 0010$. We call f *symmetric* if $f(x)$ only depends on $|x|$. Some common symmetric functions that we will refer to are:

- $\mathrm{OR}_n(x) = 1$ iff $|x| \geq 1$
- $\mathrm{AND}_n(x) = 1$ iff $|x| = n$
- $\mathrm{PARITY}_n(x) = 1$ iff $|x|$ is odd
- $\mathrm{MAJ}_n(x) = 1$ iff $|x| > n/2$

We call f *monotone (increasing)* if $f(x)$ cannot decrease if we set more variables of x to 1. A function that we will refer to sometimes is the "address function". This is a function on $n = k + 2^k$ variables, where the first k bits of the input provide an index in the last 2^k bits. The value of the indexed variable is the output of the function. Wegener [46] gives a monotone version of the address function.

2.2 Multilinear polynomials

If S is a set of (indices of) variables, then the *monomial* X_S is the product of variables $X_S = \Pi_{i \in S} x_i$. The *degree* of this monomial is the cardinality of S. A *multilinear polynomial* on n variables is a function $p : \mathbf{R}^n \to \mathbf{C}$ that can be written as $p(x) = \sum_{S \subseteq [n]} c_S X_S$ for some complex numbers c_S. We call c_S the *coefficient* of the monomial X_S in p. The *degree* of p is the degree of its largest monomial: $deg(p) = \max\{|S| \mid c_S \neq 0\}$. Note that if we restrict attention to the Boolean domain $\{0,1\}^n$, then $x_i = x_i^k$ for all $k > 1$, so considering only multilinear polynomials is no restriction when dealing with Boolean inputs.

The next lemma implies that if multilinear polynomials p and q are equal on all Boolean inputs, then they are identical:

Lemma 1 *Let $p, q : \mathbf{R}^n \to \mathbf{R}$ be multilinear polynomials of degree at most d. If $p(x) = q(x)$ for all $x \in \{0,1\}^n$ with $|x| \leq d$, then $p = q$.*

Proof Define $r(x) = p(x) - q(x)$. Suppose r is not identically zero. Let V be a minimal-degree term in r with non-zero coefficient c, and x be the input where $x_j = 1$ iff x_j occurs in V. Then $|x| \leq d$, and hence $p(x) = q(x)$. However, since all monomials in r except for V evaluate to 0 on x, we have $r(x) = c \neq 0 = p(x) - q(x)$, which is a contradiction. It follows that r is identically zero and $p = q$. $\Box$

Below we sketch the method of *symmetrization*, due to Minsky and Papert [28] (see also [4, Section 4]). Let $p : \mathbf{R}^n \to \mathbf{R}$ be a polynomial. If π is some permutation and $x = x_1 \ldots x_n$, then $\pi(x) = (x_{\pi(1)}, \ldots, x_{\pi(n)})$. Let S_n be the set of all $n!$ permutations. The *symmetrization* p^{sym} of p averages over all permutations of the input, and is defined as:

$$p^{sym}(x) = \frac{\sum_{\pi \in S_n} p(\pi(x))}{n!}.$$

Note that p^{sym} is a polynomial of degree at most the degree of p. Symmetrizing may actually lower the degree: if $p = x_1 - x_2$, then $p^{sym} = 0$. The following lemma allows us to reduce an n-variate polynomial to a single-variate one.

Lemma 2 (Minsky & Papert) *If $p : \mathbf{R}^n \to \mathbf{R}$ is a multilinear polynomial, then there exists a single-variate polynomial $q : \mathbf{R} \to \mathbf{R}$, of degree at most the degree of p, such that $p^{sym}(x) = q(|x|)$ for all $x \in \{0,1\}^n$.*

Proof Let d be the degree of p^{sym}, which is at most the degree of p. Let V_j denote the sum of all $\binom{n}{j}$ products of j different variables, so $V_1 = x_1 + \cdots + x_n$, $V_2 = x_1x_2 + x_1x_3 + \cdots + x_{n-1}x_n$, etc. Since p^{sym} is symmetrical, it is easily shown by induction that it can be written as

$$p^{sym}(x) = c_0 + c_1V_1 + c_2V_2 + \cdots + c_dV_d,$$

with $c_i \in \mathbf{R}$. Note that V_j assumes value $\binom{|x|}{j} = |x|(|x|-1)(|x|-2)\cdots(|x|-j+1)/j!$ on x, which is a polynomial of degree j of $|x|$. Therefore the single-variate polynomial q defined by

$$q(|x|) = c_0 + c_1\binom{|x|}{1} + c_2\binom{|x|}{2} + \cdots + c_d\binom{|x|}{d}$$

satisfies the lemma. $\Box$

3 Decision Tree Complexity on Various Machine Models

Below we define decision tree complexity for three different kinds of machine models: deterministic, randomized, and quantum.

3.1 Deterministic

A deterministic decision tree is a rooted ordered binary tree T. Each internal node of T is labeled with a variable x_i and each leaf is labeled with a value 0 or 1. Given an input $x \in \{0,1\}^n$, the tree is evaluated as follows. Start at the root. If this is a leaf then stop. Otherwise, query the variable x_i that labels the root. If $x_i = 0$, then recursively evaluate the left subtree, if $x_i = 1$ then recursively evaluate the right subtree. The output of the tree is the value (0 or 1) of the leaf that is reached eventually. Note that an input x deterministically determines the leaf, and thus the output, that the procedure ends up in.

We say a decision tree *computes* f if its output equals $f(x)$, for all $x \in \{0,1\}^n$. Clearly there are many different decision trees that compute the same f. The complexity of such a tree is its depth, i.e., the number of queries made on the worst-case input. We define $D(f)$, the decision tree complexity of f, as the depth of an optimal (= minimal-depth) decision tree that computes f.

3.2 Randomized

As in many other models of computation, we can add the power of randomization to decision trees. There are two ways to view a randomized decision tree. Firstly, we can add (possibly biased) coin flips as internal nodes to the tree. That is, the tree may contain internal nodes labeled by a bias $p \in [0,1]$, and when the evaluation procedure reaches such a node, it will flip a coin with bias p and will go to the left child on outcome 'heads' and to the right child on 'tails'. Now an input x no longer determines with certainty which leaf of the tree will be reached, but instead induces a probability distribution over the set of all leaves. Thus the tree outputs 0 or 1 with a certain probability. The complexity of the tree is the number of queries on the worst-case input and worst-case outcome of the coin flips. A second way to define a randomized decision tree is as a probability distribution μ over deterministic decision trees. The tree is evaluated by choosing a deterministic decisions tree according to μ, which is then evaluated as before. The complexity of the randomized tree in this second definition is the depth of the deepest T that has $\mu(T) > 0$. It is not hard to see that these two definitions are equivalent.

We say that a randomized decision tree *computes f with bounded-error* if its output equals $f(x)$ with probability at least $2/3$, for all $x \in \{0,1\}^n$. $R_2(f)$ denotes the complexity of the optimal randomized decision tree that computes f with bounded error. [2]

3.3 Quantum

We briefly sketch the framework of quantum computing, referring to [30] for more details. The classical unit of computation is a *bit*, which can take on the values 0 or 1. In the quantum case, the unit of computation is a quantum bit or *qubit*, which is a linear combination or *superposition* of the two classical values:

$$\alpha_0|0\rangle + \alpha_1|1\rangle.$$

More generally, an m-qubit state $|\phi\rangle$ is a superposition of all classical m-bit strings:

$$|\phi\rangle = \sum_{i\in\{0,1\}^m} \alpha_i|i\rangle.$$

Here α_i is a complex number, called the *amplitude* of *basis state* $|i\rangle$. We require $\sum_i |\alpha_i|^2 = 1$. Mathematically speaking, the set of m-qubit quantum states is the set of all unit vectors in the Hilbert spaced that has $\{|i\rangle \mid i \in \{0,1\}^m\}$ as an orthonormal basis.

There are two things we can do to such a state: measure it or apply a unitary transformation to it. One of the axioms of quantum mechanics says that if we measure the m-qubit register $|\phi\rangle$, then we will see the basis state $|i\rangle$ with probability $|\alpha_i|^2$. Since $\sum_i |\alpha_i|^2 = 1$, we thus have a valid probability distribution over the classical m-bit strings. After the measurement, $|\phi\rangle$ has "collapsed" to the specific observed basis state $|i\rangle$ and all other information in the state will be lost.

Apart from measuring $|\phi\rangle$, we can also apply a unitary transformation to it. That is, viewing the 2^m amplitudes of $|\phi\rangle$ as a vector in $\mathbf{C}^{2^m}$, we can obtain some new state $|\psi\rangle = \sum_{i\in\{0,1\}^m} \beta_i|i\rangle$ by multiplying $|\phi\rangle$ with a unitary matrix U: $|\psi\rangle = U|\phi\rangle$. A matrix U is unitary iff its inverse U^{-1} equals the conjugate transpose matrix U^*. Because unitarity is equivalent to preserving Euclidean norm, the new state $|\psi\rangle$ will still have $\sum_i |\beta_i|^2 = 1$. There is an extensive literature on how such large U can be obtained from small unitary transformations ("quantum gates") on few qubits at a time, see [30].

[2] The subscript '2' in $R_2(f)$ refers to the 2-sided error of the algorithm: it may err on 0-inputs as well as on 1-inputs. We will not discuss *zero-error* (Las Vegas) or *one-sided error* randomized decision trees here. See [38,31,22,23,20,8] for some results concerning such trees.

We formalize a query to an input $x \in \{0,1\}^n$ as a unitary transformation O that maps $|i, b, z\rangle$ to $|i, b \oplus x_i, z\rangle$. Here $|i, b, z\rangle$ is some m-qubit basis state, where i takes $\lceil \log n \rceil$ bits, b is one bit, z denotes the $(m - \lceil \log n \rceil - 1)$-bit "workspace" of the quantum computer, which is not affected by the query, and $\oplus$ denotes exclusive-or. This clearly generalizes the classical setting where a query inputs an i into a black-box, which returns the bit x_i: if we apply O to the basis state $|i, 0, z\rangle$ we get $|i, x_i, z\rangle$, from which the ith bit of the input can be read. Because O has to be unitary, we specify that it maps $|i, 1, z\rangle$ to $|i, 1-x_i, z\rangle$. Note that a quantum computer can make queries in superposition: applying O once to the state $\frac{1}{\sqrt{n}} \sum_{i=1}^n |i, 0, z\rangle$ gives $\frac{1}{\sqrt{n}} \sum_{i=1}^n |i, x_i, z\rangle$, which in some sense contains all bits of the input.

A *quantum decision tree* has the following form: we start with an m-qubit state $|\vec{0}\rangle$ where every bit is 0. Then we apply a unitary transformation U_0 to the state, then we apply a query O, then another unitary transformation U_1, etc. A T-query quantum decision tree thus corresponds to a big unitary transformation $A = U_T O U_{T-1} \cdots O U_1 O U_0$. Here the U_i are fixed unitary transformations, independent of the input x. The final state $A|\vec{0}\rangle$ depends on the input x only via the T applications of O. The output is obtained by measuring the final state and outputting the rightmost bit of the observed basis state (without loss of generality we can assume there are no intermediate measurements).

We say that a quantum decision tree *computes f exactly* if the output equals $f(x)$ with probability 1, for all $x \in \{0,1\}^n$. The tree *computes f with bounded-error* if the output equals $f(x)$ with probability at least 2/3, for all $x \in \{0,1\}^n$. $Q_E(f)$ denotes the number of queries of an optimal quantum decision tree that computes f exactly, $Q_2(f)$ is the number of queries of an optimal quantum decision tree that computes f with bounded-error. Note that we just count the number of queries, not the complexity of the U_i.

Unlike the classical deterministic or randomized decision trees, the quantum algorithms are not really trees anymore (the names 'quantum query algorithm' or 'quantum black-box algorithm' are also in use). Nevertheless we prefer the term 'quantum decision tree', because such quantum algorithms generalize classical trees in the sense that they can simulate them, as sketched below. Consider a T-query deterministic decision tree. It first determines which variable it will query initially; then it determines the next query depending upon its history, and so on for T queries. Eventually it outputs an output-bit depending on its total history. The basis states of the corresponding quantum algorithm have the form $|i, b, h, a\rangle$, where i, b is the query-part, h ranges over all possible histories of the classical computation (this history includes all previous queries and their answers), and a is the rightmost qubit, which will eventually contain the output. Let U_0 map the initial state $|\vec{0}, 0, \vec{0}, 0\rangle$ to $|i, 0, \vec{0}, 0\rangle$, where x_i is the first variable that the classical tree would query. Now the quantum algorithm applies O, which turns the state into $|i, x_i, \vec{0}, 0\rangle$. Then the algorithm applies a transformation U_1 that maps $|i, x_i, \vec{0}, 0\rangle$ to $|j, 0, h, 0\rangle$, where h is the new history (which includes i and x_i) and x_j is the variable that the classical tree would query given the outcome of the previous query. Then the quantum tree applies O for the second time, it applies a transformation U_2 that updates the workspace and determines the next query, etc. Finally, after T queries the quantum tree sets the answer bit to 0 or 1 depending on its total history. All operations U_i performed here are injective mappings from basis states to basis states, hence they can be extended to permutations of basis states, which are unitary transformations. Thus a T-query deterministic decision tree can be simulated by an exact T-query quantum algorithm. Similarly a T-query randomized decision tree can be simulated by a T-query quantum decision tree with the same error probability (basically because a superposition can "simulate" a probability distribution). Accordingly, we have $Q_2(f) \leq R_2(f) \leq D(f) \leq n$ and $Q_2(f) \leq Q_E(f) \leq D(f) \leq n$ for all f.

4 Some Complexity Measures

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. We can associate several measures of complexity with such functions, whose definitions and relations are surveyed below.

4.1 Certificate complexity

Certificate complexity measures how many of the n variables have to be given a value in order to fix the value of f.

Definition 1 *Let C be an assignment $C : S \to \{0,1\}$ of values to some subset S of the n variables. We say that C is* consistent *with $x \in \{0,1\}^n$ if $x_i = C(i)$ for all $i \in S$.*

*For $b \in \{0,1\}$, a b-*certificate *for f is an assignment C such that $f(x) = b$ whenever x is consistent with C. The* size *of C is $|S|$, the cardinality of S.*

The certificate complexity $C_x(f)$ of f on x *is the size of a smallest $f(x)$-certificate that is consistent with x. The* certificate complexity *of f is $C(f) = \max_x C_x(f)$. The* 1-certificate complexity *of f is $C^{(1)}(f) = \max_{\{x \mid f(x)=1\}} C_x(f)$, and similarly we define $C^{(0)}(f)$.*

For example, $C^{(1)}(\mathrm{OR}_n) = 1$ since it suffices to set one variable $x_i = 1$ to force the OR-function to 1. On the other hand, $C(\mathrm{OR}_n) = C^{(0)}(\mathrm{OR}_n) = n$.

Sensitivity and block sensitivity measure how sensitive the value of f is to changes in the input. Sensitivity was introduced in [10] (under the name of *critical complexity*) and block sensitivity in [31].[3]

Definition 2 *The* sensitivity $s_x(f)$ of f on x *is the number of variables* x_i *for which* $f(x) \neq f(x^i)$. *The* sensitivity of f *is* $s(f) = \max_x s_x(f)$.

The block sensitivity $bs_x(f)$ of f on x *is the maximum number* b *such that there are disjoint sets* $B_1, \ldots, B_b$ *for which* $f(x) \neq f(x^{B_i})$. *The* block sensitivity of f *is* $bs(f) = \max_x bs_x(f)$. *(If* f *is constant, we define* $s(f) = bs(f) = 0$.*)*

Note that sensitivity is just block sensitivity with the size of the blocks B_i restricted to 1. Simon [41] gave a general lower bound on $s(f)$:

Theorem 1 (Simon) *If* f *depends on all* n *variables, then we have* $s(f) \geq \frac{1}{2}\log n - \frac{1}{2}\log\log n + \frac{1}{2}$.

Wegener [46] proved that this theorem is tight up to the $O(\log\log n)$-term for the monotone address function.

We now prove some relations between $C(f)$, $s(f)$, and $bs(f)$. Clearly, for all x we have $s_x(f) \leq bs_x(f)$ and $bs_x(f) \leq C_x(f)$ (since a certificate for x will have to contain at least one variable of each sensitive block). Hence:

Proposition 1 $s(f) \leq bs(f) \leq C(f)$.

The biggest gap known between $s(f)$ and $bs(f)$ is quadratic and was exhibited by Rubinstein [37]:

Example 1 *Let* $n = 4k^2$. *Divide the* n *variables in* $\sqrt{n}$ *disjoint blocks of* $\sqrt{n}$ *variables: the first block* B_1 *contains* $x_1, \ldots, x_{\sqrt{n}}$, *the second block* B_2 *contains* $x_{\sqrt{n}+1}, \ldots, x_{2\sqrt{n}}$, *etc. Define* f *such that* $f(x) = 1$ *iff there is at least one block* B_i *where two consecutive variables have value 1 and the other* $\sqrt{n}-2$ *variables are 0. It is easy to see that* $s(f) = \sqrt{n}$ *and* $bs(f) = n/2$, *so we have a quadratic gap between* $s(f)$ *and* $bs(f)$. *Since* $bs(f) \leq C(f)$, *this is also a quadratic gap between* $s(f)$ *and* $C(f)$ *(Wegener and Zádori give a different function with a smaller gap between* $s(f)$ *and* $C(f)$ *[48]).*

It has been open for quite a while whether $bs(f)$ can be upper bounded by a polynomial in $s(f)$. It may well be true that $bs(f) \in O(s(f)^2)$.

[3] There has also been some work on *average* (block) sensitivity [5] and its applications [7,40,2]. In particular, Shi [40] has shown that the average sensitivity of a total function f is a lower bound on its approximate degree $\widetilde{deg}(f)$.

Open problem 1 *Is $bs(f) \in O(s(f)^k)$ for some k?*

We proceed to give Nisan's proof [31] that $C(f)$ is bounded by $bs(f)^2$.

Lemma 3 *If B is a minimal sensitive block for x, then $|B| \leq s(f)$.*

Proof If we flip one of the B-variables in x^B, then the function value must flip from $f(x^B)$ to $f(x)$ (otherwise B would not be minimal), so every B-variable is sensitive for f on input x^B. Hence $|B| \leq s_{x^B}(f) \leq s(f)$. □

Theorem 2 (Nisan) $C(f) \leq s(f)bs(f)$.

Proof Consider an input $x \in \{0,1\}^n$ and let $B_1, \ldots, B_b$ be disjoint minimal sets of variables that achieve the block sensitivity $b = bs_x(f) \leq bs(f)$. We will show that the function $C : \cup_i B_i \to \{0,1\}$ that sets variables according to x is a sufficiently small certificate for $f(x)$.

If C is not an $f(x)$-certificate, then let x' be an input that is consistent with C, such that $f(x') \neq f(x)$. Define B_{b+1} by $x' = x^{B_{b+1}}$. Now f is sensitive to B_{b+1} on x and B_{b+1} is disjoint from $B_1, \ldots, B_b$, which contradicts $b = bs_x(f)$. Hence C is an $f(x)$-certificate. By the previous lemma we have $|B_i| \leq s(f)$ for all i, hence the size of this certificate is $| \cup_i B_i| \leq s(f)bs(f)$. □

No quadratic gap between $bs(f)$ and $C(f)$ seems to be known. Some subquadratic gaps may be found in [48, Section 3].

4.3 Degree of representing polynomial

Definition 3 *A polynomial $p : \mathbf{R}^n \to \mathbf{R}$* represents *$f$ if $p(x) = f(x)$ for all $x \in \{0,1\}^n$.*

Note that since $x^2 = x$ for $x \in \{0,1\}$, we can restrict attention to *multilinear polynomials* for representing f. It is easy to see that each f can be represented by a multilinear polynomial p. Lemma 1 implies that this polynomial is unique, which allows us to define:

Definition 4 *The* degree *$deg(f)$ of f is the degree of the multilinear polynomial that represents f.*

For example, $deg(\text{AND}_n) = n$, because the representing polynomial is the monomial $x_1 \ldots x_n$. The degree $deg(f)$ may be significantly larger than $s(f)$, $bs(f)$, and $C(f)$:

Example 2 *Let f on $n = k^2$ variables be the AND of k ORs of k variables each. Both AND_k and OR_k are represented by degree-k polynomials, so the representing polynomial of f has degree $deg(f) = k^2 = n$. On the other hand, it is not hard to see that $s(f) = bs(f) = C(f) = \sqrt{n}$. Thus $deg(f)$ is quadratically larger than $s(f)$, $bs(f)$, and $C(f)$ in this case.*[4]

On the other hand, $deg(f)$ may also be significantly smaller than $s(f)$ and $bs(f)$, as the next example from Nisan and Szegedy [32] shows.

Example 3 *Consider the function E_{12} defined by $E_{12}(x_1, x_2, x_3) = 1$ iff $|x| \in \{1, 2\}$. This function is represented by the following degree-2 polynomial:*

$$E_{12}(x_1, x_2, x_3) = x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3.$$

Define E_{12}^k as the function on $n = 3^k$ variables obtained by building a complete recursive ternary tree of depth k, where the 3^k leaves are the variables and each node is the E_{12}-function of its three children. For $k > 1$, the representing polynomial for E_{12}^k is obtained by substituting independent copies of the E_{12}^{k-1}-polynomial in the above polynomial for E_{12}. This shows that $deg(f) = 2^k = n^{1/\log 3}$. On the other hand, it is easy to see that flipping any variable in the input $\vec{0}$ flips the function value from 0 to 1, hence $s(f) = bs(f) = C(f) = n = deg(f)^{\log 3}$ (Kushilevitz has found a slightly bigger gap, based on the same technique with a slightly more complex polynomial, see [33, footnote 1 on p.560]).

Below we give Nisan and Szegedy's proof that $deg(f)$ can be no more than quadratically smaller than $bs(f)$ [32]. This shows that the gap of the last example is close to optimal. The proof uses the following theorem from [12,36]:

Theorem 3 (Ehlich & Zeller; Rivlin & Cheney) *Let $p : \mathbf{R} \to \mathbf{R}$ be a polynomial such that $b_1 \leq p(i) \leq b_2$ for every integer $0 \leq i \leq n$, and its derivative has $|p'(x)| \geq c$ for some real $0 \leq x \leq n$. Then $deg(p) \geq \sqrt{cn/(c + b_2 - b_1)}$.*

Theorem 4 (Nisan & Szegedy) *$bs(f) \leq 2\, deg(f)^2$.*

Proof Let polynomial p of degree d represent f. Let $b = bs(f)$, and a and $B_1, \ldots, B_b$ be the input and sets that achieve the block sensitivity. We assume without loss of generality that $f(a) = 0$. We transform $p(x_1, \ldots, x_N)$ into a polynomial $q(y_1, \ldots, y_b)$ by replacing every x_j in p as follows:

(1) $x_j = y_i$ if $a_j = 0$ and $j \in B_i$

[4] It will follow from Theorem 10 and Corollary 2 that $deg(f) \leq C(f)^2$, so this quadratic gap between $deg(f)$ and $C(f)$ is optimal. Theorem 10 and Corollary 1 will imply $deg(f) \leq bs(f)^3$, but the quadratic gap between $deg(f)$ and $bs(f)$ of this example is the best we know of.

(2) $x_j = 1 - y_i$ if $a_j = 1$ and $j \in B_i$
(3) $x_j = a_j$ if $j \notin B_i$ for every i

Now it is easy to see that q has the following properties:

(1) q is a multilinear polynomial of degree $\leq d$
(2) $q(y) \in \{0,1\}$ for all $y \in \{0,1\}^b$
(3) $q(\vec{0}) = p(x) = f(x) = 0$
(4) $q(e_i) = p(x^{B_i}) = f(x^{B_i}) = 1$ for all unit vectors $e_i \in \{0,1\}^b$

Let r be the single-variate polynomial of degree $\leq d$ obtained from symmetrizing q over $\{0,1\}^b$. Note that $0 \leq r(i) \leq 1$ for every integer $0 \leq i \leq b$, and for some $x \in [0,1]$ we have $r'(x) \geq 1$ because $r(0) = 0$ and $r(1) = 1$. Applying Theorem 3 we get $d \geq \sqrt{b/2}$. □

The following two theorems give, respectively, a weak bound for all functions, and a strong bound for almost all functions. We state the first without proof (see [32]).

Theorem 5 (Nisan & Szegedy) *If f depends on all n variables, then we have $deg(f) \geq \log n - O(\log\log n)$.*

The address function on $n = k + 2^k$ variables has $deg(f) = k+1$, which shows that the previous theorem is tight up to the $O(\log\log n)$-term.

For the second result, define $X_1^{even} = \{x \mid |x| \text{ is even and } f(x) = 1\}$, similarly for X_1^{odd}. Let $X_1 = X_1^{even} \cup X_1^{odd}$. Let $p = \sum_S c_S X_S$ be the unique polynomial representing f, with c_S the coefficient of the monomial $X_S = \Pi_{i \in S} x_i$. The Moebius inversion formula (see [4]) says:

$$c_S = \sum_{T \subseteq S} (-1)^{|S|-|T|} f(T),$$

where $f(T)$ is the value of f on the input where exactly the variables in T are 1. We learned about the next lemma via personal communication with Yaoyun Shi.

Lemma 4 (Shi & Yao) *$deg(f) = n$ iff $|X_1^{even}| \neq |X_1^{odd}|$.*

Proof Applying the Moebius formula with $S = \{1, \ldots, n\}$, we get

$$c_S = \sum_{T \subseteq S} (-1)^{|S|-|T|} f(T) = (-1)^n \sum_{x \in X_1} (-1)^{|x|} = (-1)^n \left(|X_1^{even}| - |X_1^{odd}|\right).$$

Since $deg(f) = n$ iff the monomial $x_1 \ldots x_n$ has non-zero coefficient, the lemma follows. □

As a consequence, we can exactly count the number of functions that have less than full degree:

Theorem 6 *There are $\binom{2^n}{2^{n-1}}$ functions $f : \{0,1\}^n \to \{0,1\}$ with $deg(f) < n$.*

Proof We will count the number E of f for which $|X_1^{even}| = |X_1^{odd}|$; by Lemma 4 these are exactly the f satisfying $deg(f) < n$. Suppose we want to assign f-value 1 to exactly i of the 2^{n-1} inputs for which $|x|$ is even. There are $\binom{2^{n-1}}{i}$ ways to do this. If we want $|X_1^{even}| = |X_1^{odd}|$, then there are only $\binom{2^{n-1}}{i}$ ways to choose the f-values of the odd x. Hence

$$E = \sum_{i=0}^{2^{n-1}} \binom{2^{n-1}}{i}\binom{2^{n-1}}{i} = \binom{2^n}{2^{n-1}}.$$

The second equality is Vandermonde's convolution [18, p.174]. □

Note that $\binom{2^n}{2^{n-1}} \in \Theta(2^{2^n}/\sqrt{2^n})$ by Stirling's formula. Since there are 2^{2^n} Boolean functions on n variables, we see that the fraction of functions with degree $< n$ is $o(1)$. Thus almost all functions have full degree.

4.4 Degree of approximating polynomial

Apart from representing a function f exactly by means of a polynomial, we may also only *approximate* it with a polynomial, which can sometimes be of a smaller degree.[5]

Definition 5 *A polynomial $p : \mathbf{R}^n \to \mathbf{R}$* approximates *$f$ if $|p(x) - f(x)| \leq 1/3$ for all $x \in \{0,1\}^n$. The* approximate degree *$\widetilde{deg}(f)$ of f is the minimum degree among all multilinear polynomials that approximate f.*

As a simple example: $\frac{2}{3}x_1 + \frac{2}{3}x_2$ approximates OR_2, so $\widetilde{deg}(\mathrm{OR}_2) = 1$. In contrast, $deg(\mathrm{OR}_2) = 2$. Note that there may be many different minimal-degree polynomials that *approximate* f, whereas there is only one polynomial that *represents* f.

By the same technique as Theorem 4, Nisan and Szegedy [32] showed

Theorem 7 (Nisan & Szegedy) $bs(f) \leq 6\ \widetilde{deg}(f)^2$.

[5] Also *non-deterministic* polynomials for f have been studied [49], but we will not cover that notion in this survey.

The approximate degree of f can sometimes be significantly smaller than the degree of f. Nisan and Szegedy [32] constructed a degree-$O(\sqrt{n})$ polynomial that approximates OR_n. Since $bs(\mathrm{OR}_n) = n$, the previous theorem implies that this degree is optimal. Since $deg(\mathrm{OR}_n) = n$ we have a quadratic gap between $deg(f)$ and $\widetilde{deg}(f)$. This is the biggest gap known.

Ambainis [1] showed that almost all functions have high approximate degree:

Theorem 8 (Ambainis) *Almost all f have $\widetilde{deg}(f) \geq n/2 - O(\sqrt{n}\log n)$.*

5 Application to Decision Tree Complexity

The complexity measures discussed above are intimately related to the decision tree complexity of f in various models. In fact, $D(f)$, $R_2(f)$, $Q_E(f)$, $Q_2(f)$, $bs(f)$, $C(f)$, $deg(f)$, and $\widetilde{deg}(f)$ are all polynomially related.

5.1 Deterministic

We start with two simple lower bounds on $D(f)$.

Theorem 9 $bs(f) \leq D(f)$.

Proof On input x with disjoint sensitive blocks $B_1, \ldots, B_{bs(f)}$, a deterministic decision tree must query at least one variable in each block B_i, for otherwise we could flip that block (and hence the correct output) without the tree noticing it. Thus the tree must make at least $bs(f)$ queries on input x. $\Box$

Theorem 10 $deg(f) \leq D(f)$.

Proof Consider a decision tree for f of depth $D(f)$. Let L be a 1-leaf (i.e., a leaf with output 1) and $x_1, \ldots, x_r$ be the queries on the path to L, with values $b_1, \ldots, b_r$. Define the polynomial $p_L(x) = \Pi_{i:b_i=1} x_i \Pi_{i:b_i=0}(1 - x_i)$. Then p_L has degree $r \leq D(f)$. Furthermore, $p_L(x) = 1$ if leaf L is reached on input x, and $p_L(x) = 0$ otherwise. Let $p = \sum_L p_L$ be the sum of all p_L over all 1-leaves. Then p has degree $\leq D(f)$, and $p(x) = 1$ iff a 1-leaf is reached on input x, so p represents f. $\Box$

Below we give some upper bounds on $D(f)$ in terms of $bs(f)$, $C(f)$, $deg(f)$, and $\widetilde{deg}(f)$. Beals et al. [3] prove

Theorem 11 $D(f) \leq C^{(1)}(f)bs(f)$.

Proof The following describes an algorithm to compute $f(x)$, querying at most $C^{(1)}(f)bs(f)$ variables of x (in the algorithm, by a "consistent" certificate C or input y at some point we mean a C or y that agrees with the values of all variables queried up to that point).

```
(1) Repeat the following at most $bs(f)$ times:
        Pick a consistent 1-certificate $C$ and query those of its variables whose
        $x$-values are still unknown (if there is no such $C$, then return 0 and
        stop); if the queried values agree with $C$ then return 1 and stop.
(2) Pick a consistent $y \in \{0,1\}^n$ and return $f(y)$.
```

The nondeterministic "pick a C" and "pick a y" can easily be made deterministic by choosing the first C resp. y in some fixed order. Call this algorithm A. Since A runs for at most $bs(f)$ stages and each stage queries at most $C^{(1)}(f)$ variables, A queries at most $C^{(1)}(f)bs(f)$ variables.

It remains to show that A always returns the right answer. If it returns an answer in step (1), this is either because there are no consistent 1-certificates left (and hence $f(x)$ must be 0) or because x is found to agree with a particular 1-certificate C. In both cases A gives the right answer.

Now consider the case where A returns an answer in step (2). We will show that all consistent y must have the same f-value. Suppose not. Then there are consistent y, y' with $f(y) = 0$ and $f(y') = 1$. A has queried $b = bs(f)$ 1-certificates $C_1, C_2, \ldots, C_b$. Furthermore, y' contains a consistent 1-certificate C_{b+1}. We will derive from these C_i disjoint sets B_i such that f is sensitive to each B_i on y. For every $1 \leq i \leq b+1$, define B_i as the set of variables on which y and C_i disagree. Clearly, each B_i is non-empty, for otherwise the procedure would have returned 1 in step (1). Note that y^{B_i} agrees with C_i, so $f(y^{B_i}) = 1$, which shows that f is sensitive to each B_i on y. Suppose variable k occurs in some B_i $(1 \leq i \leq b)$, then $x_k = y_k \neq C_i(k)$. If $j > i$, then C_j has been chosen consistent with all variables queried up to that point (including x_k), so we cannot have $x_k = y_k \neq C_j(k)$. This shows that $k \notin B_j$, hence all B_i and B_j are disjoint. But then f is sensitive to $bs(f) + 1$ disjoint sets on y, which is a contradiction. Accordingly, all consistent y in step 2 must have the same f-value, and A returns the right value $f(y) = f(x)$ in step 2, because x is one of those consistent y. $\Box$

Combining with $C^{(1)} \leq C(f) \leq s(f)bs(f)$ (Theorem 2) we obtain:

Corollary 1 $D(f) \leq s(f)bs(f)^2 \leq bs(f)^3$.

It might be possible to improve this to $D(f) \leq bs(f)^2$. This would be optimal, since the function f of Example 2 has $bs(f) = \sqrt{n}$ and $D(f) = n$.

Open problem 2 *Is $D(f) \in O(bs(f)^2)$?*

Of course, Theorem 11 also holds with $C^{(0)}$ instead of $C^{(1)}$. Since $bs(f) \leq \max\{C^{(0)}(f), C^{(1)}(f)\}$, we also obtain the following result, due to [6,21,43].

Corollary 2 $D(f) \leq C^{(0)}(f)C^{(1)}(f)$.

Now we will show that $D(f)$ is upper bounded by $deg(f)^4$ and $\widetilde{deg}(f)^6$. The first result is due to Nisan and Smolensky, below we give their (previously unpublished) proof. It improves the earlier result $D(f) \in O(deg(f)^8)$ of Nisan and Szegedy [32]. Here a *maxonomial* of f is a monomial with maximal degree in f's representing polynomial p.

Lemma 5 (Nisan & Smolensky) *For every maxonomial M of f, there is a set B of variables in M such that $f(\vec{0}^B) \neq f(\vec{0})$.*

Proof Obtain a restricted function g from f by setting all variables outside of M to 0. This g cannot be constant 0 or 1, because its unique polynomial representation (as obtained from p) contains M. Thus there is some subset B of the variables in M that makes $g(\vec{0}^B) \neq g(\vec{0})$ and hence $f(\vec{0}^B) \neq f(\vec{0})$. $\square$

Lemma 6 (Nisan & Smolensky) *There exists a set of $deg(f)bs(f)$ variables that intersects each maxonomial of f.*

Proof Greedily take all variables in maxonomials of f, as long as there is a maxonomial that is still disjoint from those taken so far. Since each such maxonomial will contain a sensitive block for $\vec{0}$, and there can be at most $bs(f)$ disjoint sensitive blocks, this procedure can go on for at most $bs(f)$ maxonomials. Since each maxonomial contains $deg(f)$ variables, the lemma follows. $\square$

Theorem 12 (Nisan & Smolensky) $D(f) \leq deg(f)^2 bs(f) \leq 2deg(f)^4$.

Proof By the previous lemma, there is a set of $deg(f)bs(f)$ variables that intersects each maxonomial of f. Query all these variables. This induces a restriction g of f on the remaining variables, such that $deg(g) < deg(f)$ (because the degree of each maxonomial in the representation of f drops at least one) and $bs(g) \leq bs(f)$. Repeating this inductively for at most $deg(f)$ times, we reach a constant function and learn the value of f. This algorithm uses at most $deg(f)^2 bs(f)$ queries, hence $D(f) \leq deg(f)^2 bs(f)$. Theorem 4 gives the second inequality of the theorem. $\square$

Combining Corollary 1 and Theorem 7 we obtain the following result from [3] (improving the earlier $D(f) \in O(\widetilde{deg}(f)^8)$ result of Nisan and Szegedy [32]):

Theorem 13 $D(f) \in O(\widetilde{deg}(f)^6)$.

Finally, since $deg(f)$ may be polynomially larger or smaller than $bs(f)$, the following theorem may be weaker or stronger than Theorem 11. The proof uses an idea similar to the above Nisan-Smolensky proof.

Theorem 14 $D(f) \leq C^{(1)}(f)deg(f)$.

Proof Let p be the representing polynomial for f. Choose some certificate $C : S \to \{0,1\}$ of size $\leq C^{(1)}(f)$. If we fill in the S-variables according to C, then p must reduce to a constant function (constant 0 if C is a 0-certificate, constant 1 if C is a 1-certificate). Hence the certificate has to intersect each maxonomial of p. Accordingly, querying all variables in S reduces the polynomial degree of the function by at least 1. Repeating this $deg(f)$ times, we end up with a constant function and hence know $f(x)$. In all, this algorithm takes at most $C^{(1)}(f)deg(f)$ queries. $\Box$

5.2 Randomized

Here we show that $D(f)$, $R_2(f)$, $bs(f)$, and $\widetilde{deg}(f)$ are all polynomially related. We first give the bounded-error analogues of Theorems 10 and 9:

Theorem 15 $\widetilde{deg}(f) \leq R_2(f)$.

Proof Consider a randomized decision tree for f of depth $R_2(f)$, viewed as a probability distribution μ over different deterministic decision trees T, each of depth at most $R_2(f)$. Using the technique of Theorem 10, we can write each of those T as a 0/1-valued polynomial p_T of degree at most $R_2(f)$. Define $p = \sum_T \mu(T)p_T$, which has degree at most $R_2(f)$. Then it is easy to see that p gives the acceptance probability of R, so p approximates f. $\Box$

Nisan [31] proved

Theorem 16 (Nisan) $bs(f) \leq 3\ R_2(f)$.

Proof Consider an algorithm with $R_2(f)$ queries, and an input x that achieves the block sensitivity. For every set S such that $f(x) \neq f(x^S)$, the probability that the algorithm queries a variable in S must be $\geq 1/3$, otherwise the algorithm could not "see" the difference between x and x^S with sufficient probability. Hence on input x the algorithm has to make an expected number of at least $1/3$ queries in each of the $bs(f)$ sensitive blocks, so the total expected number of queries on input x must be at least $bs(f)/3$. Since the worst-case number of queries on input x is at the least the expected number of queries on x, the theorem follows. $\Box$

Combined with Corollary 1 we see that the gap between $D(f)$ and $R_2(f)$ can be at most cubic [31]:

Corollary 3 (Nisan) $D(f) \leq 27\ R_2(f)^3$.

There may be some room for improvement here, because the biggest gap known between $D(f)$ and $R_2(f)$ is much less than cubic:

Example 4 *Let f on $n = 2^k$ variables be the complete binary AND-OR-tree of depth k. For instance, for $k = 2$ we have $f(x) = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$. It is easy to see that $deg(f) = n$ and hence $D(f) = n$. There is a simple randomized algorithm for f [42,38]: randomly choose one of the two subtrees of the root and recursively compute the value of that subtree; if its value is 0 then output 0, otherwise compute the other subtree and output its value. It can be shown that this algorithm always gives the correct answer with expected number of queries $O(n^\alpha)$, where $\alpha = \log((1+\sqrt{33})/4) \approx 0.7537\ldots$. Saks and Wigderson [38] showed that this is asymptotically optimal for zero-error algorithms for this function, and Santha [39] proved the same for bounded-error algorithms. Thus we have $D(f) = n = \Theta(R_2(f)^{1.3\ldots})$.*

Open problem 3 *What is the biggest gap between $D(f)$ and $R_2(f)$?*

5.3 Quantum

As in the classical case, $deg(f)$ and $\widetilde{deg}(f)$ give lower bounds on quantum query complexity. The next lemma from [3] is also implicit in the combination of some proofs in [15,16].

Lemma 7 *Let A be a quantum decision tree that makes T queries. Then there exist complex-valued n-variate multilinear polynomials α_i of degree at most T, such that the final state of A is*

$$\sum_{i \in \{0,1\}^m} \alpha_i(x)|i\rangle,$$

for every input $x \in \{0,1\}^n$.

Proof Let $|\phi_k\rangle$ be the state of quantum decision tree (on input x) just before the kth query. Note that $|\phi_{k+1}\rangle = U_k O|\phi_k\rangle$. The amplitudes in $|\phi_0\rangle$ depend on the initial state and on U_0 but not on x, so they are polynomials of x of degree 0. A query maps basis state $|i, b, z\rangle$ to $|i, b \oplus x_i, z\rangle$, so if the amplitude of $|i, 0, z\rangle$ in $|\phi_0\rangle$ is α and the amplitude of $|i, 1, z\rangle$ is β, then the amplitude of $|i, 0, z\rangle$ *after* the query becomes $(1 - x_i)\alpha + x_i\beta$ and the amplitude of $|i, 1, z\rangle$ becomes $x_i\alpha + (1 - x_i)\beta$, which are polynomials of degree 1. (In general, if the amplitudes before a query are polynomials of degree $\leq j$, then the amplitudes after the query will be polynomials of degree $\leq j + 1$.) Between the first and the second query lies the unitary transformation U_1. However, the amplitudes after applying U_1 are just linear combinations of the amplitudes before applying U_1, so the amplitudes in $|\phi_1\rangle$ are polynomials of degree at most 1. Continuing inductively, the amplitudes of the final state are found to be polynomials of degree at most T. We can make these polynomials multilinear without affecting their values on $x \in \{0, 1\}^n$, by replacing all x_i^m by x_i. $\square$

Theorem 17 *$deg(f) \leq 2\ Q_E(f)$.*

Proof Consider an exact quantum algorithm for f with $Q_E(f)$ queries. Let S be the set of basis states corresponding to a 1-output. Then the acceptance probability is $P(x) = \sum_{k \in S} |\alpha_k(x)|^2$. By the previous lemma, the α_k are polynomials of degree $\leq Q_E(f)$, so $P(x)$ is a polynomial of degree $\leq 2Q_E(f)$. But P represents f, so it has degree $deg(f)$ and hence $deg(f) \leq 2Q_E(f)$. $\square$

By a similar proof:

Theorem 18 *$\widetilde{deg}(f) \leq 2\ Q_2(f)$.*

Both theorems are tight for $f = \text{PARITY}_n$: here we have $deg(f) = \widetilde{deg}(f) = n$ [28] and $Q_E(f) = Q_2(f) = \lceil n/2 \rceil$ [3,13]. No f is known with $Q_E(f) > deg(f)$ or $Q_2(f) > \widetilde{deg}(f)$, so the following question presents itself:

Open problem 4 *Are $Q_E(f) \in O(deg(f))$ and $Q_2(f) \in O(\widetilde{deg}(f))$?*

Note that the degree lower bounds of Theorems 6 and 8 now imply strong lower bounds on the quantum decision tree complexities of almost all f. In particular, Theorem 8 implies that $Q_2(f) \geq n/4 - O(\sqrt{n} \log n)$ for almost all f. In contrast, Van Dam [45] has shown that $Q_2(f) \leq n/2 + \sqrt{n}$ for *all* f.

Combining Theorems 17 and 18 with Theorems 12 and 13 we obtain the polynomial relations between classical and quantum complexities of [3]:

Corollary 4 *$D(f) \in O(Q_E(f)^4)$ and $D(f) \in O(Q_2(f)^6)$.*

Some other quantum lower bounds via degree lower bounds may be found in [3,1,29,14,8].

The biggest gap that is known between $D(f)$ and $Q_E(f)$ is only a factor of 2: $D(\text{PARITY}_n) = n$ and $Q_E(\text{PARITY}_n) = \lceil n/2 \rceil$. The biggest gap we know between $D(f)$ and $Q_2(f)$ is quadratic: $D(\text{OR}_n) = n$ and $Q_2(\text{OR}_n) \in \Theta(\sqrt{n})$ by Grover's quantum search algorithm [19]. Also, $R_2(\text{OR}_n) \in \Theta(n)$, $deg(\text{OR}_n) = n$, $\widetilde{deg}(\text{OR}_n) \in \Theta(\sqrt{n})$.

Open problem 5 *What are the biggest gaps between the classical $D(f)$, $R_2(f)$ and their quantum analogues $Q_E(f)$, $Q_2(f)$?*

The previous two open problems are connected via the function $f = E_{12}^k$ on $n = 3^k$ variables (Example 3): this has $D(f) = s(f) = n$ but $deg(f) = n^{1/\log 3}$. The complexity $Q_E(f)$ is unknown; it must lie between $n^{1/\log 3}/2$ and n. However, it must either show a gap between $D(f)$ and $Q_E(f)$ (partly answering the last question) or between $deg(f)$ and $Q_E(f)$ (answering the penultimate question).

6 Some Special Classes of Functions

Here we look more closely at several special classes of Boolean functions.

6.1 Symmetric functions

Recall that a function is *symmetric* if $f(x)$ only depends on the Hamming weight $|x|$ of its input, so permuting the input does not change the value of the function. A symmetric f is fully described by giving a vector $(f_0, f_1, \ldots, f_n) \in \{0,1\}^{n+1}$, where f_k is the value of $f(x)$ for $|x| = k$. Because of this and Lemma 2, there is a close relationship between polynomials that represent symmetric functions, and single-variate polynomials that assume values 0 or 1 on $\{0, 1, \ldots, n\}$. Using this relationship, von zur Gathen and Roche [17] prove $deg(f) = (1 - o(1))n$ for all symmetric f:

Theorem 19 (von zur Gathen & Roche) *If f is non-constant and symmetric, then $deg(f) = n - O(n^{0.548})$. If, furthermore, $n + 1$ is prime, then $deg(f) = n$.*

In fact, von zur Gathen and Roche conjecture that $deg(f) = n - O(1)$ for all symmetric f. The biggest gap they found is $deg(f) = n - 3$ for some specific f and n. Via Theorems 10 and 17, the above degree lower bounds give strong lower bounds on $D(f)$ and $Q_E(f)$.

For the case of *approximate* degrees of symmetric f, Paturi [34] gave the following tight characterization. Define $\Gamma(f) = \min\{|2k - n + 1| : f_k \neq f_{k+1}\}$. Informally, this quantity measures the length of the interval around Hamming weight $n/2$ where f_k is constant.

Theorem 20 (Paturi) *If f is non-constant and symmetric, then $\widetilde{deg}(f) = \Theta(\sqrt{n(n - \Gamma(f))})$.*

Paturi's result implies lower bounds on $R_2(f)$ and $Q_2(f)$. For $Q_2(f)$ these bounds are in fact tight (a matching upper bound was shown in [3]), but for $R_2(f)$ a stronger bound can be obtained from Theorem 16 and the following result [44]:

Proposition 2 (Turán) *If f is non-constant and symmetric, then $s(f) \geq \lceil \frac{n+1}{2} \rceil$.*

Proof Let k be such that $f_k \neq f_{k+1}$, and $|x| = k$. Without loss of generality assume $k \leq \lfloor (n-1)/2 \rfloor$ (otherwise give the same argument with 0s and 1s reversed). Note that flipping any of the $n-k$ 0-variables in x flips the function value. Hence $s(f) \geq s_x(f) \geq n - k \geq \lceil (n+1)/2 \rceil$. $\Box$

This lemma is tight, since $s(\mathrm{MAJ}_n) = \lceil (n+1)/2 \rceil$.

Collecting the previous results, we have tight characterizations of the various decision tree complexities of all symmetric f:

Theorem 21 *If f is non-constant and symmetric, then*

- $D(f) = (1 - o(1))n$
- $R_2(f) = \Theta(n)$
- $Q_E(f) = \Theta(n)$
- $Q_2(f) = \Theta(\sqrt{n(n - \Gamma(f))})$

6.2 Monotone functions

One nice property of monotone functions was shown in [31]:

Proposition 3 (Nisan) *If f is monotone, then $C(f) = s(f) = bs(f)$.*

Proof Since $s(f) \leq bs(f) \leq C(f)$ for all f, we only have to prove $C(f) \leq s(f)$. Let $C : S \to \{0, 1\}$ be a minimal certificate for some x with $|S| = C(f)$. Without loss of generality we assume $f(x) = 0$. For each $i \in S$ it must hold that $x_i = 0$ and $f(x^i) = 1$, for otherwise i could be dropped from the certificate, contradicting minimality. Thus each variable in S is sensitive in x, hence $C(f) \leq s_x(f) \leq s(f)$. $\Box$

Theorem 11 now implies:

Corollary 5 *If f is monotone, then $D(f) \leq s(f)^2$.*

This corollary is exactly tight, since the function f of Example 2 has $D(f) = n$ and $s(f) = \sqrt{n}$ and is monotone.

Also, the lower bound of Theorem 4 can be improved to

Proposition 4 *If f is monotone, then $s(f) \leq deg(f)$.*

Proof Let x be an input on which the sensitivity of f equals $s(f)$. Assume without loss of generality that $f(x) = 0$. All sensitive variables must be 0 in x, and setting one or more of them to 1 changes the value of f from 0 to 1. Hence by fixing all variables in x except for the $s(f)$ sensitive variables, we obtain the OR function on $s(f)$ variables, which has degree $s(f)$. Therefore $deg(f)$ must be at least $s(f)$. $\Box$

The above two results strengthen some of the previous bounds for monotone functions:

Corollary 6 *If f is monotone, then $D(f) \in O(R_2(f)^2)$, $D(f) \in O(Q_E(f)^2)$, and $D(f) \in O(Q_2(f)^4)$.*

For the special case where f is both monotone and symmetric, we have:

Proposition 5 *If f is non-constant, symmetric, and monotone, then $deg(f) = n$.*

Proof Note that f is simply a threshold function: $f(x) = 1$ iff $|x| \geq t$ for some t. Let $p : \mathbf{R} \to \mathbf{R}$ be the non-constant single-variate polynomial obtained from symmetrizing f. This has degree $\leq deg(f) \leq n$ and $p(i) = 0$ for $i \in \{0, \ldots, t-1\}$, $p(i) = 1$ for $i \in \{t, \ldots, n\}$. Then the derivative p' must have zeroes in each of the $n-1$ intervals $(0,1), (1,2), \ldots, (t-2, t-1), (t, t+1), \ldots, (n-1, n)$. Hence p' has degree at least $n-1$, which implies that p has degree n and $deg(f) = n$. $\Box$

6.3 Monotone graph properties

An interesting and well studied subclass of the monotone functions are the *monotone graph properties.* Consider an undirected graph on n vertices. There are $N = \binom{n}{2}$ possible edges, each of which may be present or absent, so we can pair up the set of all graphs with the set of all N-bit strings. A *graph property* P is a set of graphs that is closed under permutation of the vertices (so isomorphic graphs have the same properties). The property is *monotone* if it is closed under the addition of an edge. We are now interested in the question: At how many edges must we look in order to determine if a graph has the property P? This is just the decision-tree complexity of P if we view P as a total Boolean function on N bits.

A property P is called *evasive* if $D(P) = N$, so if we have to look at all edges in the worst case. The *evasiveness conjecture* (also sometimes called Aanderaa-Karp-Rosenberg conjecture) says that all non-constant monotone graph properties P are evasive. This conjecture is still open; see [27] for an overview. The conjecture has been proved for graphs where the number of vertices is a prime power [25], but the best known general bound is $D(P) \in \Omega(N)$ [35,25,26]. This bound also follows from a degree-bound by Dodis and Khanna [11, Theorem 2]:

Theorem 22 (Dodis & Khanna) *If P is a non-constant monotone graph property, then $deg(P) \in \Omega(N)$.*

Corollary 7 *If P is a non-constant monotone graph property, then $D(P) \in \Omega(N)$ and $Q_E(P) \in \Omega(N)$.*

Thus the evasiveness conjecture holds up to a constant factor for both deterministic classical and exact quantum algorithms. $D(P) = N$ may actually hold for all monotone graph properties P, but [8] exhibit a monotone P with $Q_E(P) < N$. Only much weaker lower bounds are known for the bounded-error complexity of such properties [26,20,8].

Open problem 6 *Are $D(P) = N$ and $R_2(P) \in \Omega(N)$ for all non-constant monotone graph properties P?*

There is no P known with $R_2(P) \in o(N)$, but the OR-problem can trivially be turned into a monotone graph property P with $Q_2(P) \in o(N)$, in fact $Q_2(P) \in \Theta(n)$ [8].

Finally we mention a result about sensitivity from [46]:

Theorem 23 (Wegener) *$s(P) \geq n-1$ for all non-constant monotone graph properties P.*

This theorem is tight, as witnessed by the property "No vertex is isolated" [44].

Acknowledgments

We thank Noam Nisan for permitting us to include his and Roman Smolensky's proof of Theorem 12, Louis Wingers for finding a bug in an earlier version of Theorem 6, and an anonymous referee for some useful comments that improved the presentation of the paper.

References

[1] A. Ambainis. A note on quantum black-box complexity of almost all Boolean functions. *Information Processing Letters*, 71(1):5–7, 1999. Also at http://arxiv.org/abs/quant-ph/9811080.

[2] A. Ambainis and R. de Wolf. Average-case quantum query complexity. In *Proceedings of 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS'2000)*, volume 1770 of *Lecture Notes in Computer Science*, pages 133–144. Springer, 2000. quant-ph/9904079.

[3] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. In *Proceedings of 39th FOCS*, pages 352–361, 1998. quant-ph/9802049.

[4] R. Beigel. The polynomial method in circuit complexity. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 82–95, 1993.

[5] A. Bernasconi. Sensitivity vs. block sensitivity (an average-case study). *Information Processing Letters*, 59(3):151–157, 1996.

[6] M. Blum and R. Impagliazzo. Generic oracles and oracle classes (extended abstract). In *Proceedings of 28th FOCS*, pages 118–126, 1987.

[7] R. B. Boppana. The average sensitivity of bounded-depth circuits. *Information Processing Letters*, 63(5):257–261, 1997.

[8] H. Buhrman, R. Cleve, R. de Wolf, and Ch. Zalka. Bounds for small-error and zero-error quantum algorithms. In *Proceedings of 40th FOCS*, pages 358–368, 1999. cs.CC/9904019.

[9] H. Buhrman and R. de Wolf. Communication complexity lower bounds by polynomials. cs.CC/9910010, 1999.

[10] S. Cook, C. Dwork, and R. Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM Journal on Computing*, 15:87–97, 1986.

[11] Y. Dodis and S. Khanna. Space-time tradeoffs for graph properties. In *Proceedings of 26th ICALP*, pages 291–300, 1999.

[12] H. Ehlich and K. Zeller. Schwankung von Polynomen zwischen Gitterpunkten. *Mathematische Zeitschrift*, 86:41–44, 1964.

[13] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. A limit on the speed of quantum computation in determining parity. *Physical Review Letters*, 81:5442–5444, 1998. quant-ph/9802045.

[14] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. How many functions can be distinguished with k quantum queries? quant-ph/9901012, 7 Jan 1999.

[15] S. Fenner, L. Fortnow, S. Kurtz, and L. Li. An oracle builder's toolkit. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 120–131, 1993.

[16] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *Journal of Computer and Systems Sciences*, 59(2):240–252, 1999. Earlier version in Complexity'98. cs.CC/9811023.

[17] J. von zur Gathen and J. R. Roche. Polynomials with two values. *Combinatorica*, 17(3):345–362, 1997.

[18] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1989.

[19] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th STOC*, pages 212–219, 1996. quant-ph/9605043.

[20] P. Hajnal. An $n^{4/3}$ lower bound on the randomized complexity of graph properties. *Combinatorica*, 11:131–143, 1991. Earlier version in Structures'90.

[21] J. Hartmanis and L.A. Hemachandra. One-way functions, robustness and the non-isomorphism of NP-complete sets. In *Proceedings of the 2nd IEEE Structure in Complexity Theory Conference*, pages 160–174, 1987.

[22] R. Heiman, I. Newman, and A. Wigderson. On read-once threshold formulae and their randomized decision tree complexity. *Theoretical Computer Science*, 107(1):63–76, 1993. Earlier version in Structures'90.

[23] R. Heiman and A. Wigderson. Randomized vs. deterministic decision tree complexity for read-once Boolean functions. *Computational Complexity*, 1:311–329, 1991. Earlier version in Structures'91.

[24] J. Kahn, G. Kalai, and N. Linial. The influence of variables on Boolean functions. In *Proceedings of 29th FOCS*, pages 68–80, 1988.

[25] J. Kahn, M. Saks, and D. Sturtevant. A topological approach to evasiveness. *Combinatorica*, 4:297–306, 1984. Earlier version in FOCS'83.

[26] V. King. Lower bounds on the complexity of graph properties. In *Proceedings of 20th STOC*, pages 468–476, 1988.

[27] L. Lovász and N. Young. Lecture notes on evasiveness of graph properties. Technical report, Princeton University, 1994. Available at `http://www.uni-paderborn.de/fachbereich/AG/agmadh/WWW/english/scripts.html`.

[28] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, MA, 1968. Second, expanded edition 1988.

[29] A. Nayak and F. Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of 31st STOC*, pages 384–393, 1999. quant-ph/9804066.

[30] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[31] N. Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. Earlier version in STOC'89.

[32] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994. Earlier version in STOC'92.

[33] N. Nisan and A. Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995. Earlier version in FOCS'94.

[34] R. Paturi. On the degree of polynomials that approximate symmetric Boolean functions (preliminary version). In *Proceedings of 24th STOC*, pages 468–474, 1992.

[35] R. Rivest and S. Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3:371–384, 1976.

[36] T. J. Rivlin and E. W. Cheney. A comparison of uniform approximations on an interval and a finite subset thereof. *SIAM Journal on Numerical Analysis*, 3(2):311–320, 1966.

[37] D. Rubinstein. Sensitivity vs. block sensitivity of Boolean functions. *Combinatorica*, 15(2):297–299, 1995.

[38] M. Saks and A. Wigderson. Probabilistic Boolean decision trees and the complexity of evaluating game trees. In *Proceedings of 27th FOCS*, pages 29–38, 1986.

[39] M. Santha. On the Monte Carlo decision tree complexity of read-once formulae. In *Proceedings of the 6th IEEE Structure in Complexity Theory Conference*, pages 180–187, 1991.

[40] Y. Shi. Lower bounds of quantum black-box complexity and degree of approximating polynomials by influence of Boolean variables. *Information Processing Letters*, 75(1–2):79–83, 2000. quant-ph/9904107.

[41] H. U. Simon. A tight $\Omega(\log\log n)$-bound on the time for parallel RAM's to compute non-degenerate Boolean functions. In *Symposium on Foundations of Computation Theory*, volume 158 of *Lecture Notes in Computer Science*, pages 439–444. Springer, 1983.

[42] M. Snir. Lower bounds for probabilistic linear decision trees. *Theoretical Computer Science*, 38:69–82, 1985.

[43] G. Tardos. Query complexity, or why is it difficult to separate $NP^A \cap coNP^A$ from P^A by random oracles A? *Combinatorica*, 9(4):385–392, 1989.

[44] G. Turán. The critical complexity of graph properties. *Information Processing Letters*, 18:151–153, 1984.

[45] W. van Dam. Quantum oracle interrogation: Getting all information for almost half the price. In *Proceedings of 39th FOCS*, pages 362–367, 1998. quant-ph/9805006.

[46] I. Wegener. The critical complexity of all (monotone) Boolean functions and monotone graph properties. *Information and Control*, 67:212–222, 1985.

[47] I. Wegener. *The Complexity of Boolean Functions.* Wiley-Teubner Series in Computer Science, 1987.

[48] I. Wegener and L. Zádori. A note on the relations between critical and sensitive complexity. *Journal of Information Processing and Cybernetics (EIK)*, 25(8/9):417–421, 1989.

[49] R. de Wolf. Characterization of non-deterministic quantum query and quantum communication complexity. In *Proceedings of 15th IEEE Conference on Computational Complexity*, pages 271–278, 2000. cs.CC/0001014.