THE SENSITIVITY CONJECTURE AND THE COMPLEXITY OF BOOLEAN FUNCTIONS

DAVID RAMSEY

Abstract. We present the "Sensitivity Conjecture" of the study of boolean functions and discuss progress made on the problem. We also introduce a number of different "complexity measures" of boolean functions and their relationships. Finally, we present an interesting application of complexity measures to the computational complexity of boolean functions.

Contents

1. Introduction 1
2. The Sensitivity Conjecture 2
2.1. Progress on the Sensitivity Conjecture 3
3. Complexity measures of boolean functions 3
3.1. Decision trees and decision tree complexity 4
3.2. Representing and approximating polynomials for boolean functions 5
4. The CREW PRAM complexity of a boolean function 6
Acknowledgments 6
References 7

1. Introduction

Some boolean functions are clearly more complicated than others. We can formalize the simplicity or complexity of a given boolean function f with complexity measures. Loosely defined, a complexity measure ϕ maps a boolean function f to a nonnegative integer $\phi(f) \leq n$ such that if $\phi(f) < \phi(g)$, then f is "less complex" than g with respect to the property measured by ϕ. There is a rich variety of complexity measures whose interrelationships are still being studied.

The relationship of two in particular—sensitivity and block sensitivity—is at the heart of a long-standing open problem called the Sensitivity Conjecture, which asks whether sensitivity and block sensitivity are *polynomially related*: whether one can be bounded from above by a polynomial in the other, and vice versa. However, after years of effort, the best known upper bound of block sensitivity is exponential in sensitivity, and the best known separation is quadratic. We present these results below.

In addition, we provide an introduction to a number of complexity measures—*certificate complexity*, *deterministic decision tree complexity*, *randomized decision*

Date: DEADLINE August 26, 2011.

tree complexity, *quantum decision tree complexity*, *polynomial degree*, and *approximating polynomial degree*. All of these complexity measures are polynomially related to block sensitivity, making their study highly relevant to the Sensitivity Conjecture.

Finally, we discuss a result in the computational complexity of boolean functions published by Noam Nisan in 1989 ([5]).[1] A CREW PRAM (Concurrent Read, Exclusive Write, Parallel Random Access Memory) consists of a finite amount of memory and a finite number of processors with unbounded processing power such that at each time interval, each processor reads one location from the memory, performs any computation, and writes to one memory location such that no two processors attempt to write to the same place at the same time. The minimum number of time intervals required for a CREW PRAM to evaluate a boolean function given an input is written CREW(f). Cook, Dwork, and Reischuk discovered that $\Theta(\log\ s(f)) \leq \text{CREW}(f)$ for all f ([6]). Exploiting this fact and the polynomial relationship of block sensitivity and deterministic decision tree complexity, Nisan shows that $\Theta(\log\ bs(f)) = \text{CREW}(f)$ for all f, providing a very precise characterization of the complexity of a boolean function with respect to a CREW PRAM.

For a useful survey of these and other topics related to the Sensitivity Conjecture, see [1].

2. The Sensitivity Conjecture

We first give some notation. A *boolean function in n variables* is a function $f : \{0,1\}^n \to \{0,1\}$. For the rest of the paper, f is a boolean function in n variables. We write an input to a boolean function as a string of bits $x = x_1x_2\dots x_n$, and without context x_i refers to the ith index of x. We also write $|x|$ to denote the *Hamming weight* of x, which is the number of indices in x that are equal to 1. For example, $|100111| = 4$. Let $[n] = \{1, 2, \dots, n\}$. For $i \in [n]$, let $x^i = x_1x_2\dots(1 - x_i)\dots x_n$: that is, x^i is x with the ith bit flipped. If $f(x) \neq f(x^i)$ for some x, i, then we say that i is a *sensitive bit* for $f(x)$. More generally, for $A \subset [n]$, let x^A be x with the ith bit flipped for all $i \in A$. If $f(x) \neq f(x^A)$ for some x, A, then we say that A is a *sensitive block* for $f(x)$.

Definition 2.1. The *sensitivity* $s(f, x)$ of f at x is the number of sensitive bits i for $f(x)$. The sensitivity $s(f)$ of f is the maximum of $s(f, x)$ taken over all inputs x.

Example 2.2. Let $n = 8$, and let $f(x) = 1$ if and only if $|x| = 4, 5$. Then for most inputs x, for example $x = 00000000$, $s(f, x) = 0$. But for $x = 11100000$, $f(x^i) \neq f(x)$ for $4 \leq i \leq 8$, so $s(f, x) = 5$, and indeed $s(f) = 5$.

Definition 2.3. The *block sensitivity* $bs(f, x)$ of f at x is the maximum size k of a set of pairwise disjoint sensitive blocks $A_1, A_2, \dots, A_k$ for $f(x)$. The block sensitivity of f is the maximum of $bs(f, x)$ taken over all inputs x.

[1] Nisan's paper is particularly relevant because it contains the first published appearance of block sensitivity.

Example 2.4. Let f be as in Example 2.2. If $x = 11110000$, then $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5,6\}$, and $\{7,8\}$ are 6 disjoint sensitive blocks for $f(x)$, and indeed $bs(f) = 6$.

The natural question is, how are sensitivity and block sensitivity related? Years of effort have convinced researchers that there is no more than a polynomial separation between the two. More precisely, let a, b be complexity measures. Then a and b are *polynomially related* if we have polynomials p and q such that $a(f) \leq p(b(f))$ and $b(f) \leq q(a(f))$ for all boolean functions f. Then the Sensitivity Conjecture claims that sensitivity and block sensitivity are polynomially related. It is clear from the definitions that the block sensitivity of a boolean function is always greater than or equal to the sensitivity, but it is not precisely known how much greater. Thus, the heart of the Sensitivity Conjecture is the claim that

Conjecture 2.5. *There exists a polynomial p such that $bs(f) \leq p(s(f))$ for all boolean functions f.*

2.1. **Progress on the Sensitivity Conjecture.** Currently, the best known upper bound on block sensitivity is exponential in sensitivity. Claire Kenyon and Samuel Kutin introduced the *ℓ-block sensitivity* bs_ℓ of a boolean function, which is defined identically to block sensitivity except that only sensitive blocks of size less than or equal to ℓ are considered ([2]). Kenyon and Kutin proved that for all f,

$$s(f) \geq (bs_\ell(f)((\ell-1)!/e)^{1/\ell}. \tag{2.6}$$

Suppose we have a sensitive block A for input x such that $|A|$ is minimal, so A contains no sensitive proper subblocks. Then for all $i \in A$,

$$f((x^A)^i) = f(x^{A-\{i\}}) = f(x) \neq f(x^A). \tag{2.7}$$

Thus, $f(x^A)$ is sensitive on all bits $i \in A$, so $|A| \leq s(f)$. As a general consequence, $bs_{s(f)}(f) = bs(f)$ for all f. Thus, by applying Stirling's approximation to (2.6), we get

$$bs(f) \leq (\frac{e}{\sqrt{2\pi}})e^{s(f)}\sqrt{s(f)}, \tag{2.8}$$

which is currently the best known upper bound on block sensitivity.

The best known separation between sensitivity and block sensitivity is quadratic, and the most well-known illustration of such a separation is known as Rubinstein's function. Let $n = k^2$ for some even k. We divide the n variables into k consecutive blocks of k consecutive variables. Then let $f(x) = 1$ if and only if there is at least one block $x_{ak+1}x_{ak+2}\cdots x_{ak+k}$ such that exactly two variables $x_{ak+(2i-1)}$, x_{ak+2i} are 1 and the rest are 0. Let x be such that $x_1 = x_2 = 1$ and all other bits equal 0. Then $f(x) = 1$, but $f(x)$ is sensitive at all k bits in the first block, so $s(f,x) = k$, and indeed we can check that $s(f) = k = \sqrt{n}$. On the other hand, we can partition $000\cdots0$ into $n/2$ sensitive blocks of size 2, so $bs(f, 000\cdots0) = n/2$, and indeed $bs(f) = n/2$. Thus, f exhibits a quadratic separation between sensitivity and block sensitivity. However, Kenyon and Kutin have proved that no simple adaptation of Rubinstein's function will exhibit a greater than quadratic separation.

3. Complexity measures of boolean functions

We now present several different complexity measures of boolean functions. The property of these functions that is most relevant to the Sensitivity Conjecture is that all of them are polynomially related to block sensitivity (see [4] for a proof).

Theorem 3.1. *The complexity measures block sensitivity (bs), deterministic decision tree complexity (D), two-sided bounded error randomized decision tree complexity (R_2), two-sided bounded error quantum decision tree complexity (Q_2), polynomial degree (deg), and approximating polynomial degree ($\widetilde{deg}$) are all polynomially related.*

Thus, proving a polynomial upper bound in sensitivity on any of these complexity measures is equivalent to proving the Sensitivity Conjecture.

Some complexity measures are defined in terms of the definition of a boolean function, while others arise naturally from an alternate representation of a boolean function. We first present an example of the former.

Definition 3.2. Given $x \in \{0,1\}^n$, a set $C \subset [n]$ is a *certificate* for $f(x)$ if for all $y \in \{0,1\}^n$ such that x and y agree on all bits $i \in C$, $f(x) = f(y)$. The *certificate complexity* $C(f,x)$ of f at x is the minimum size of a certificate for $f(x)$, and the certificate complexity $C(f)$ of f is the maximum of $C(f,x)$ taken over all inputs x.

Example 3.3. Let f be as in Example 2.2. If $x = 11110000$, we can see that given any subset C of $[n]$ with cardinality less than or equal to 6, we can find a y such that x and y agree on all bits in C, but $f(x) \neq f(y)$. Indeed, $C(f,x) = C(f) = 7$.

3.1. Decision trees and decision tree complexity.

One of the most basic problems in the theory of boolean functions is the problem of computing them; that is, given a boolean function f, give an algorithm that, when given any n-bit string input x, outputs $f(x)$. Such an algorithm will in almost all cases require one or more steps to read the input x. A step in which the algorithm reads the ith bit of the input x is called a *query* for x_i. Such an algorithm is conveniently modeled with a decision tree, a diagram of the algorithm in terms of which indices are queried and how the next steps of the algorithm depend on the values of these indices.

A *deterministic decision tree* is a finite rooted binary tree in which each of the nodes is labeled with an index x_i, each node has a left and a right child, and each leaf is labeled either 0 or 1. To compute using a deterministic decision tree given an input x, first start at the root. If the current vertex is a leaf, output the label of the leaf. Otherwise, query the bit labeled on the vertex. If the value is 0, go to the left child and continue recursively, and if the value is 1, go the the right child and continue recursively, until a leaf is reached. A deterministic decision tree *computes* f if it agrees with f on all inputs x. Note that for any boolean function there are infinitely many decision trees that compute it.

Definition 3.4. The *deterministic decision tree complexity* $D(f)$ of f is the depth of the minimum-depth deterministic decision tree that computes f.

We can generalize this definition by allowing a decision tree to incorporate randomness in its computation. One way to do so is to allow the decision tree to contain internal nodes such that whenever the computation of a decision tree reaches such a node, we proceed to either the left or right child according to a random variable. Then the complexity of the tree is the number of queries the tree makes on the worst-case input. (Note that this number is not the depth of the tree in general.) Equivalently, we can model the decision tree as a probability distribution over a set of deterministic decision trees that compute f. To compute with such a tree, randomly choose a deterministic decision tree according to the distribution, then proceed as with a deterministic decision tree. The complexity of the tree is the depth of the maximum-depth deterministic decision tree with nonzero probability of being chosen.

Given our new model, we can consider zero-error (Las Vegas) computation, one-sided bounded error computation, or two-sided bounded error computation. For the purposes of this paper, it is sufficient to consider the latter. We say that a randomized decision tree computes f with bounded error if its output is equal to $f(x)$ with probability at least $\frac{2}{3}$ for all inputs $x \in \{0,1\}^n$.

Definition 3.5. The *two-sided bounded error randomized decision tree complexity* $R_2(f)$ of f is the complexity of the minimum-complexity randomized decision tree that computes f with bounded error.

Finally, we can extend our definition to quantum decision trees. An introduction to quantum computation of boolean functions is beyond the scope of this paper; see [3] for a survey.

Definition 3.6. The *two-sided bounded error quantum decision tree complexity* $Q_2(f)$ of f is the complexity of the minimum-complexity quantum decision tree that computes f with bounded error.

3.2. **Representing and approximating polynomials for boolean functions.** Any boolean function in n variables can be written as a polynomial with integer coefficients in n variables restricted to the domain $\{0,1\}^n$. To see this, note that the product

$$[x_{a_1} \cdots x_{a_i}][(1 - x_{b_1}) \cdots (1 - x_{b_j})] \tag{3.7}$$

is equal to 1 on a boolean input if and only if $x_{a_1} = \cdots = x_{a_i} = 1$ and $x_{b_1} = \cdots = x_{b_j} = 0$. Thus, for all n-bit vectors x, we can write a polynomial that evaluates to 1 on x and to 0 on all other boolean inputs. Then a general boolean function f is the sum of all such polynomials for all x such that $f(x) = 1$.

Furthermore, this polynomial has the property that each of its monomials is a product of distinct variables, so no variable is taken to a higher power than 1. Such a polynomial is called *multilinear*. For each $A \subset \{0,1\}^n$, the monomial of A is $X_A = \prod_{i \in A} x_i$. Then any boolean function f can be written as a sum $\sum_{A \subset \{0,1\}^n} c_A X_A$ for some integer coefficients c_A. In fact, this multilinear polynomial is unique. We present the proof from [4] below.

Proposition 3.8. *Let $p, q : \mathbb{R}^n \to \mathbb{R}$ be multilinear polynomials of degree at most $d \leq n$. 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 0. Let X_A be a monomial in r of minimal degree such that c_A is nonzero. Define y such that $y_i = 1$

if and only if $i \in A$. Then $|y| = |A| \leq d$, but all monomials in r other than X_A evaluate to 0 on y, so $f(y) = c_A$ is nonzero, a contradiction. Thus, $p = q$. □

We call this polynomial the *representing polynomial* for f. We can now give a new complexity measure.

Definition 3.9. The *polynomial degree* $deg(f)$ of a boolean function f is the polynomial degree of the representing polynomial for f.

We can also consider polynomials that approximate the behavior of f, but are not necessarily the representing polynomial. An *approximating polynomial* for f is a multilinear polynomial p such that for all $x \in \{0,1\}^n$, $|p(x) - f(x)| \leq \frac{1}{3}$.

Definition 3.10. The *approximating polynomial degree* $\widetilde{deg}(f)$ of a boolean function f is the minimum polynomial degree of all approximating polynomials for f.

4. The CREW PRAM complexity of a boolean function

The following results are by Nisan, and demonstrate a remarkably precise characterization of the CREW PRAM complexity of a boolean function ([5]). Recall that CREW(f) is the minimum number of steps required for a CREW PRAM to evaluate $f(x)$ given an input x.

Proposition 4.1. *For any boolean function f, CREW(f) $\geq \Theta(log\ bs(f))$.*

Proof. Let x be an input such that $bs(f,x) = bs(f)$, and let $A_1, \cdots, A_{bs(f)}$ be disjoint sensitive blocks for $f(x)$. We define a new boolean function f' in $bs(f)$ variables. For any $S \subset [bs(f)]$, let $A_S = \bigcup_{i \in S} A_i$. Then f' is defined by letting $f'(000...0^S) = f(x^{A_S})$. Thus, $f'(000...0) = f(x)$, and for all $i \in [bs(f)]$, $f'(000...0^i) = f(x^{A_i}) \neq f(x)$, so $s(f') = s(f', 000...0) = bs(f)$.

From [6], we have that CREW(f') $\geq \Theta(\log s(f')) = \Theta(\log bs(f))$. But f' clearly reduces to f on a CREW PRAM, so CREW(f) $\geq$ CREW(f'). Thus, CREW(f) $\geq \Theta(\log bs(f))$.

□

Proposition 4.2. *For any boolean function f, $\Theta(log\ D(f)) \geq CREW(f)$.*

Proof. We simulate a deterministic decision tree of depth d with a CREW PRAM in $\log d$ time. Define a CREW PRAM with a processor at each node of the decision tree. On the first step, each processor reads the input variable belonging to its node and, according to the value read, writes a pointer pointing to the next node in the computation. For all subsequent steps, each processor looks at the node to which it is pointing and copies the pointer at that node over its own pointer (the "pointer doubling" method). After $\log d$ steps, the root will point to the final value of the computation.

□

Theorem 4.3. *For any boolean function f, CREW(f) $= \Theta(log\ bs(f))$.*

Proof. We have $\Theta(\log D(f)) \geq$ CREW(f) $\geq \Theta(\log bs(f))$. But $bs(f)$ and $D(f)$ are polynomially related by Theorem 3.1, so $\log bs(f)$ and $\log D(f)$ differ by a constant factor.

□

Acknowledgments. The author would like to thank Lazslo Babai for introducing him to the sensitivity conjecture, Katie Mann and William Lopes for helping direct his research, his mentor Daniel Le for his advice and for helping revise this paper, and Peter May for his advice on this paper.

References

[1] Pooya Hatami, Raghav Kulkarni, and Denis Pankratov. Variations on the Sensitivity Conjecture. Theory of Computing Library Graduate Surveys, 4:1-27, 2011

[2] Claire Kenyon and Samuel Kutin. Sensitivity, block sensitivity, and ℓ-block sensitivity of boolean functions. Information and Computation, 189:43-53, 2004.

[3] Ronald de Wolf. Quantum communication and complexity. Theoretical Computer Science, 287:337353, 2002.

[4] Harry Burman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. Theoretical Comptuer Science, 288:21-43, 2002

[5] Noam Nisan. CREW PRAMs and decision trees. Proceedings of the twenty-first annual ACM symposium on Theory of computing, 327-335, 1989

[6] Stephen Cook, Cynthia Dwork, and Rdiger Reischuk. Upper and lower bounds on parallel random access machines without simultaneous writes. Siam Journal of Computing, 15,1:87-97, 1986