Classical Measures and Quantum Query Complexity on Boolean Functions

Hongchao Zhang

August 12, 2005

INTRODUCTION

Computational Complexity Theory:

• Complexity theory is part of the theory of computation dealing with the resources required during computation to solve a given problem.

• The most common resources are time and space. The time complexity of a problem is the number of steps that it takes to solve an instance of the problem, as a function of the size of the input, (usually measured in bits) using the most efficient algorithm.

The general approach of solving a hard problem is to set the goal a little bit lower and try to sovle a simpler problem first. The hope is that by solving the simpler problem we can understand the original, more difficult problem better.

Decision Tree Complexity and Boolean Functions

Boolean Functions:
$f : \{0,1\}^n \to \{0,1\}$. Boolean function maps a bit string to a boolean value.

Eg.,
$\text{AND}_5(11111) = 1$,
$\text{MAJORITY}(10001) = 0$

Decision Tree:
Computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$ using queries to the input.

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 binary tree that an optimal algorithm can be described.

Classical measures of boolean functions:

Certificate:

Definition 1. A *certificate* of f at x is a subset $S \subseteq [n]$ of indices together with values x_i for all $i \in S$ such that for all $y \in \{0,1\}^n$ with $y_i = x_i$ for all $i \in S$, we have $f(x) = f(y)$.
Definition 2. The *certificate complexity* for f at x denoted $C_x(f)$ is the size of a smallest certificate for f at x.
Definition 3. The *certificate complexity* of f denoted $C(f)$ is $\max_x C_x(f)$

The 1-certificate complexity of f is $C^{(1)}(f) = \max_{\{x|f(x)=1\}} C_x(f)$, and similarly we can define $C^{(0)}(f)$.

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

Sensitivity and Block Sensitivity:
Measure how sensitive a value of f is to changes in the input.

Definition 4. 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 $s(f) = max_x s_x(f)$.
Definition 5. 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 $bs(f) = max_x bs_x(f)$.
Note. *Sensitivity is just block sensitivity with the size of the blocks* B_i *restricted to* 1.

Some relationships of Measures:

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

Proof. Since each sensitive bit is a size 1 sensitive block, we have $s(f) \leq bs(f)$.
Let x and $B_1, \ldots, B_b$ be the input and sensitive blocks that achieve the block sensitivity of f, $b = bs(f)$. Let $C = C_x(f)$, for each B_i, if $C \cap B_i = \emptyset$, then even if we fix C, we can still flip the function value by flipping B_i. Therefore, $C \cap B_i \neq \emptyset$ for all $1 \leq i \leq n$. $bs(f) \leq C(f)$. □

Lemma 1. *If B is a minimal sensitive block for x, then $|B| \leq s(f)$*

Proof. Let $y = x^B$, we have $f(y) \neq f(x)$ since B is a sensitive block for x. And for each bit $i \in B$, we must have that $f(y^i) \neq f(y)$ otherwise B would not be minimal. So every bit in B is sensitive for f in input y. □

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

Proof. Let $B_1, \ldots, B_b$ be the sensitive blocks for f on some input x which achieves $b = bs_x(f) \leq bs(f)$. Let $C = \bigcup_{i=1}^{b} B_i$. If C is not an $f(x)$-certificate then there exists some block B' that is disjoint to C s.t. $f(x^{B'}) \neq f(x)$ then B' is also a sensitive block for f on input x, which contradicts to $b = bs_x(f)$. So C is a certificate for f on x. And by previous lemma, we have $|B_i| \leq s(f)$ for all i. Therefore, the size of this certificate is $|\bigcup_{i=1}^{b} B_i| \leq bs(f)s(f)$ □

Known Polynomial Relationships Between Measures

From the research that have been done so far, the main results say that all of these complexity measures are polynomially related to each other and to the decision tree complexity.

- $s(f) \leq bs(f) \leq C(f)$
- $C(f) \leq s(f)bs(f)$
- $bs(f) \leq D(f)$
- $D(f) \leq s(f)bs(f)^2 \leq bs(f)^3$

Quantum query algorithms

Suppose we want to compute function f. For input $x \in \{0,1\}^N$, a query gives us access to the input bits:

$$O_x : |i, b, z\rangle \rightarrow |i, b \oplus x_i, z\rangle. \qquad (1)$$

where $i \in [N] = \{1, \ldots, N\}$ and $b \in \{0, 1\}$; the z-part corresponts to the workspace.

A T-query quantum algorithm is in the form $A = U_T O_x U_{T-1} \cdots O_x U_1 O_x U_0$, where the U_k are fixed unitary transformations, independent of x. The algorithm starts in initial state $|0\rangle$. And the output of A is obtained by observing the leftmost qubit of the final state $A|0\rangle$

Quantum lower-bound method for Quantum query algorithms

Weighted Scheme method:

Definition 6. Let $f : \{0,1\}^N \to \{0,1\}$, $A \subseteq f^{-1}(0)$, $B \subseteq f^{-1}(1)$ and $R \subseteq A \times B$. A weight scheme for A, B, R consists of numbers $w(x,y) > 0$, $w'(x,y,i) > 0$, $w'(y,x,i) > 0$ for all $(x,y) \in R$ and $i \in [N]$ satisfying $x_i \neq y_i$, we have $w'(x,y,i)w'(y,x,i) \geq w^2(x,y)$.

Definition 7. The weight of x is

$wt(x) = \sum_{y:(x,y)\in R} w(x,y)$, if $x \in A$ and $wt(x) = \sum_{y:(y,x)\in R} w(x,y)$ if $x \in B$.

Definition 8. Let $i \in [N]$. The load of variable x_i in assignment x is

$$v(x,i) = \sum_{y:(x,y)\in R, x_i \neq y_i} w'(x,y,i)$$

if $x \in A$ and

$$v(x,i) = \sum_{y:(y,x)\in R, x_i \neq y_i} w'(x,y,i)$$

if $x \in B$.

Let the maximum A-load be $v_A = \max_{x\in A, i\in[N]} \frac{v(x,i)}{wt(x)}$.
Let the maximum B-load be $v_B = \max_{x\in B, i\in[N]} \frac{v(x,i)}{wt(x)}$.
The maximum load of a weight scheme is $v_{max} = \sqrt{v_A v_B}$.

Theorem 3. *If a function f has a weight scheme with maximum load v_{max}, then $Q_2(f) = \Omega(\frac{1}{v_{max}})$.*

For example, Grover's algorithm. If we set all the weight w, w' to be 1, where the pair x and y are different in only one bit, and anything else to be 0. We can have that the algorithm is lower bounded by $\sqrt{N}$, which is tight.

Another example: Two-level And-Or tree. The method gives a tight lower bound $\Omega(\sqrt{N})$

Relation to classical measures

Theorem 4. *Let f be any Boolean function. Then any quantum query algorithm computing f uses $\Omega(\sqrt{bs(f)})$ queries.*

We can just choose the pairs of x, y where x is one of the inputs that archieves the block sensitivity, and all the y's be the inputs we optain by flipping a sensitive block on x. Then we make the weights of all these pairs to be 1.

Theorem 5. $\sqrt{\frac{1}{v_{max}}} \leq \sqrt{N \cdot C_{-}(f)}$ *where* $C_{-}(f) = \min\{C_0(f), C_1(f)\}$. $C_0(f)$ *and* $C_1(f)$ *are the* 0*-certificate and* 1*-certificate for any boolean function* f.

Theorem 6. *If* f *is a total function,* $\sqrt{\frac{1}{v_{max}}} \leq \sqrt{C_0(f)C_1(f)}$.

Example: general binary And-Or tree, best known to be $O(N^{0.753})$, and the lower-bound we get from the method is limited by $\sqrt{C_0(f)C_1(f)} = \sqrt{N}$