6.896 Quantum Complexity Theory Octobor 16th, 2008

Lecture 13

Lecturer: Scott Aaronson

Last time we finished quantum query complexity of boolean functions. Recall that the quantum query complexity of recursive AND-OR tree is $\Theta(\sqrt{n})$. We then studied the collision problem:

> **Collision Problem.** Given oracle access to a function $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$, decide whether f is 1-to-1 or 2-to-1, promised that one of those is the case.

Today, we'll prove the query lower bound for this problem, and discuss two applications of the lower bound. There are two motivations to study this question.

1 Motivations

Scott's original motivation is to understand the computation power of hidden variable theory. Imagine that nature knows not only the quantum state, but also the "real state" about what outcome would be if you perform a measurement, even if you do not look at it. So at any time, there is not only a superposition of particles, but also the "real place" where the particles are. If we believe that, then the real position of particles must follow some trajectory overtime. The trajectory is necessarily guided by the quantum state, because it has to agree with the prediction of the quantum mechanics. But in addition to that, it has to be a real particle trajectory. This is the kind of things you see in Bohm mechanics, a famous reformulation of quantum mechanics in 1950s to involve hidden variables.

The question Scott asked was: suppose you can see the entire trajectory of the hidden variable, what would that give you the power to do? One thing we can do with this ability is to solve the collision problem in a single query, and $O(\log n)$ computational steps. If f is 2-to-1, it is easy to prepare a superposition of a collision pairs $(1/\sqrt{2})(|x\rangle + |y\rangle)$, where $f(x) = f(y)$ (by going to a superposition of all input x's, computing $f(x)$ in the superposition, and measuring the f register.) Now, if only we can range things so that we can see both x and y then we would find the collision and solve the collision problem. If we can see the entire trajectory of hidden variables, that is exactly the thing we can do. We can arrange so that the hidden variable starts being x, and varies to y and vice versa.

Thus, a lower bound on the quantum query complexity of collision problem proves a separation in the block box world between the ordinary quantum computing, and the quantum computing in the hidden variable theories.

However, a person on the street doesn't necessary care about this interpretation of quantum mechanics. On the other hand, people care more about graph isomorphism, and breaking cryptographic hash functions. These are the "real application" that you could do if you have a quantum algorithm for solving the collision problem. We are interested in rule out this possibility by lower bounds on the collision problem. This would imply, for example, to break the cryptographic hash functions, we have to exploit the structure of the hash functions somehow. Likewise, to solve graph isomorphism, we have to exploit the graph structure.

2 The Lower Bound

We have seen on Tuesday that there is a $O(n^{1/3})$-query quantum algorithm for solving the collision problem. It turns out to be optimal. Today, we will show a lower bound $\Omega(n^{1/4})$.

We have mentioned that the difficult for proving the lower bound is that sort of all the approaches we saw before are based on the hardness of finding a single needle in a haystack. The underlying principle is that a quantum algorithm cannot monitor the change of many disjoint places. For example, the block sensitivity argument says that we need $\sqrt{(\#\text{ of blocks})}$ queries to detect the change of any single block. However, in this problem, the block sensitivity is only 2.

We use polynomial method to prove the lower bound. For every $x, h \in \{1, \ldots, n\}$, define variables $\Delta_{x,h} = 1$ if $f(x) = h$, and $\Delta_{x,h} = 0$ otherwise. The acceptance probability of a quantum algorithm can be expressed as a polynomial over $\Delta_{x,h}$'s.

Lemma 1 *Let Q be a quantum algorithm makes T queries to f. Then Q's acceptance probability is a degree $2T$ multi-linear polynomial $p(f)$ over variables $\Delta_{x,h}$'s. Furthermore, every monomial of $p(f)$ is of the form $\Delta_{x_1,h_1} \cdot \Delta_{x_2,h_2} \cdots \Delta_{x_d,h_d}$ with distinct x_i's.*

Proof: (sketch) This is essentially the same as what we have seen before. Every amplitude can be written as a polynomial over $\Delta_{x,h}$. A unitary operation does not change the degree since it is linear. A query can be expressed as

$$\sum_{x,z} \alpha_{x,z} |x, z\rangle \mapsto \sum_{x,z} \alpha_{x,z} |x, z \oplus f(x)\rangle = \sum_{x,z} \left(\sum_{h} \alpha_{x,z\oplus h} \cdot \Delta_{x,h} \right) |x, z\rangle,$$

so increase the degree by at most 1. The factor 2 comes from squaring the amplitude to get the probability. Since every variable $\Delta_{x,h}$ is either 0 or 1, $\Delta_{x,h}^2 = \Delta_{x,h}^2$, and we can assume $p(f)$ is multi-linear. Furthermore, observe that for every x, there is only one h such that $\Delta_{x,h} = 1$, and other $\Delta_{x,h}$'s are 0, we can assume every monomial does not contain two variables with the same subscript x. □

The next step is to reduce the number of variables. This time, we define $q(k) = \mathbb{E}_{f\,:\,k\text{-to-1}}[p(f)]$, where the notation means expected value over uniformly random k-to-1 function $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$. We hope $q(k)$ to be a polynomial with small degree in k. However, a technicality is that n may not divisible by k, so there is no perfect k-to-1 functions. This is the main difficulty to apply the polynomial method. For now, we first make the unrealistic assumption that n is divisible by k for every k. We will come back to this technicality later.

Lemma 2 *Let $q(k) = \mathbb{E}_{f\,:\,k\text{-to-1}}[p(f)]$, where $p(f)$ is defined as above. Assume that n is divisible by k for every k, then $q(k)$ is a polynomial in k, and* $\deg(q) \le \deg(p) \le 2T$.

Proof: Let $p(f) = \sum_I \alpha_I I(f)$, where each $I(f) = \Delta_{x_1,h_1} \cdot \Delta_{x_2,h_2} \cdots \Delta_{x_d,h_d}$ is a monomial of f with distinct x_i's. By linearity of expectation,

$$q(k) = \sum_I \alpha_I \mathop{\mathbb{E}}_{f\,:\,k\text{-to-1}} [I(f)].$$

Fix a $I(f) = \Delta_{x_1,h_1} \cdot \Delta_{x_2,h_2} \cdots \Delta_{x_d,h_d}$ with $d \le \deg(p(f)) \le 2T$, let us compute $\mathbb{E}_{f\,:\,k\text{-to-1}}[I(f)]$.

Observe that $I(f)=1$ if $f(x_1)=h_1,\ldots,f(x_d)=h_d$, and $I(f)=0$ otherwise. $\mathbb{E}_{f\,:\,k\text{-to-1}}[I(f)]$ is the fraction of k-to-1 functions that satisfy the d constraints $f(x_i)=h_i$. The number of k-to-1 functions is

$$\binom{n}{n/k}\cdot\frac{n!}{(k!)^{n/k}}.$$

(We first decide the range of a function, and then decide how to map the domain to the range.) To compute the number of functions satisfying $f(x_i)=h_i$, we need the following variables. Let s be the number of distinct h values in $I(f)$. Let $i_1,i_2,\ldots,i_s$ be the number of each h in $I(f)$ (so $i_1+\cdots+i_s=d$). The number of functions satisfying $f(x_i)=h_i$ is

$$\binom{n-s}{(n/k)-s}\cdot\frac{(n-d)!}{(k!)^{n/k-s}\cdot(k-i_1)!\cdot\cdots\cdot(k-i_s)!}.$$

(Again, the first factor is to pick the range of a function. Every h_j appears in $I(f)$ must in the range. The second factor is to decide the map from $n-d$ unfixed domain to the range.) Staring at the expression for a while, we have

$$\mathop{\mathbb{E}}_{f\,:\,k\text{-to-1}}[I(f)]=\frac{\binom{n-s}{(n/k)-s}\cdot(n-d)!}{(k!)^{n/k-s}\cdot(k-i_1)!\cdot\cdots\cdot(k-i_s)!}\cdot\frac{(k!)^{n/k}}{\binom{n}{n/k}\cdot n!}$$

$$\begin{aligned}
&= \frac{(n-s)!\cdot(n-d)!}{((n/k)-s)!(n-(n/k))!(k!)^{(n/k)-s}(k-i_1)!\cdots(k-i_s)!}\cdot\frac{(n/k)!(n-(n/k))!(k!)^{n/k}}{n!\cdot n!}\\
&= r(n,d,s)\cdot\frac{(n/k)!(k!)^s}{((n/k)-s)!(k-i_1)!\cdots(k-i_s)!}\qquad\text{for some function } r\\
&= r(n,d,s)\cdot(n/k)((n/k)-1)\cdots((n/k)-s+1)\cdot\left(\prod_{j=1}^{s}k(k-1)\cdots(k-i_j+1)\right)\\
&= r(n,d,s)\cdot(n)(n-k)\cdots(n-(s-1)k)\cdot\left(\prod_{j=1}^{s}(k-1)\cdots(k-i_j+1)\right)
\end{aligned}$$

is a polynomial over k of degree $(s-1)+(i_1-1)+\cdots+(i_s-1)=i_1+\cdots+i_s-1=d-1$. Therefore,

$$q(k)=\sum_I\alpha_I\mathop{\mathbb{E}}_{f\,:\,k\text{-to-1}}[I(f)]$$

is a polynomial over k of degree at most $2T$. □

Now, suppose $p(f)$ is the acceptance probability of a quantum algorithm solving the collision problem, then

$$0\le q(1)=\mathop{\mathbb{E}}_{f\,:\,1\text{-to-1}}[p(f)]\le 1/3,\ \text{and } 2/3\le q(2)=\mathop{\mathbb{E}}_{f\,:\,2\text{-to-1}}[p(f)]\le 1.$$

For $k\ge 2$, since $p(f)$ represents a probability, we have

$$0\le q(k)=\mathop{\mathbb{E}}_{f\,:\,k\text{-to-1}}[p(f)]\le 1.$$

We can then apply Markov inequality as in Lecture 10 to lower bound the degree of $q(k)$, which gives query lower bound on Q. However, recall the technicality that n may not divisible by k. The price to resolve this technicality is that the above argument is only valid for small k. (Intuitively, the divisibility problem produces some error, which will hurt us when k is too large.) Originally, Scott's paper got $\Omega(n^{1/5})$. This was subsequently improved to $\Omega(n^{1/4})$. Yaoyun Shi gave a sophisticated way to resolve it, and achieved tight lower bound $\Omega(n^{1/3})$. We do not go through the detail in this lecture, but only intuitively argue that the above is valid up to $k = \Omega(n^{1/2})$, and so we can get a lower bound $\Omega(n^{1/4})$. The following argument should be able to turn into a rigorous proof.

When n is not divisible by k, there is no perfect k-to-1 functions, but there are almost k-to-1 functions. Intuitively, the difference is only on the left-over ($n \bmod k$) $< k$ inputs. Recall the intuition from the Grover lower bound that a quantum algorithm needs $\sqrt{n/s}$ queries to notice a change of f on s inputs. Therefore, when $k = O(n^{1/2})$, a quantum algorithm with $O(n^{1/4})$ query can not notice this difference.

Theorem 3 *Any quantum algorithm for the collision problem needs to make $\Omega(n^{1/4})$ queries. I.e., $Q(\text{COLLISION}) = \Omega(n^{1/4})$.*

Proof: Let Q be a quantum algorithm for the collision problem making T queries to f, and the corresponding $p(f)$ and $q(k)$ be defined as above. From the above argument, we have $\deg(q) \leq 2T$, $0 \leq q(1) \leq 1/3$, $2/3 \leq q(2) \leq 1$, and $0 \leq q(k) \leq 1$ for $k = 3, 4, \ldots, \Omega(n^{1/2})$. Now, we are in the same situation as Lecture 10. We have $q'(k) \geq 1/3$ for some $k \in [1, 2]$, and the Markov inequality says

$$\deg(q) \geq \sqrt{\frac{\mathit{Length} \cdot \mathit{MaxDeriv}}{\mathit{Height}}} \geq \sqrt{\frac{\Omega(n^{1/2}) \cdot 1/3}{O(1)}} = \Omega(n^{1/4}).$$

Therefore, the number of query $T = \Omega(n^{1/4})$. $\square$

We next discuss some applications of this lower bound of collision problem.

3 Implication to Element Distinctness Problem

The first application is to the query lower bound of element distinctness problem. We start by the problem definition.

> **Element Distinctness Problem.** Given oracle access to $x_1, x_2, \ldots, x_n$, decide whether there exists $i \neq j$ such that $x_i = x_j$.

Compare to the collision problem, this problem has much less structure to exploit. There might be only one collision pair, instead of many pairs. What is the query complexity of the element distinctness problem? Let us consider the upper bound first. A good thing to try is to apply Grover's algorithm, and see what we get. If we apply Grover in a naive way, there are n^2 pairs, and we need $\sqrt{n^2} = n$ queries to find collision pairs. Can we use Grover in a smarter way to do better?

The idea is a bit tricky when you see it first time. Let $U = \{x_1, \ldots, x_n\}$. Consider the following algorithm.

1. Randomly pick $\sqrt{n}$ elements $S \subset U$, and query those elements classically.

2. If we find collision in S, output the collision.
3. Use Grover's algorithm to find collisions between S and $U - S$.

What is the query complexity of the algorithm? The first step makes $\sqrt{n}$ queries. The third step uses Grover to search $x_j \in U - S$ such that $x_j = x_i$ for some $x_i \in S$. This uses $O(\sqrt{n}$ queries. In total, the algorithm uses $O(\sqrt{n})$ queries.

What is the success probability of this algorithm? Suppose for the worst case, there is only one collision pair $x_i = x_j$. For the algorithm to be able to find the collision, either x_i or x_j needs to be picked in S in the first step. This happens with probability $\Omega(1/\sqrt{n})$. When this happens, the algorithm can find the collision with constant probability in Step 2 or 3. Therefore, the overall success probability is $\Omega(1/\sqrt{n})$.

Now, suppose we invoke the above algorithm $O(\sqrt{n})$ times, we can see a collision with high probability. Doing so naively requires $O(\sqrt{n}) \cdot O(\sqrt{n}) = O(n)$ queries. However, we can use another Grover on top of the $O(\sqrt{n})$ invocation of the algorithm to find a success invocation. Searching over $O(\sqrt{n})$ items only requires $O(\sqrt{\sqrt{n}})$ queries. Overall, the two layers Grover only requires $O(n^{1/4}) \cdot O(\sqrt{n}) = O(n^{3/4})$.

On the other hand, what is the lower bound? $\Omega(\sqrt{n})$ lower bound is easy to observe. Suppose there is only one collision $x_i = x_j$, and we know x_i at beginning, then the task reduce to find x_j. This is exactly the Grover's problem. Therefore, Grover's lower bound gives $\Omega(\sqrt{n})$ lower bound for the element distinctness problem. Can we do better?

By applying the collision lower bound, we can improve the lower bound to $\Omega(n^{2/3})$. Consider the contrapositive, if we can solve the element distinctness problem in $o(n^{2/3})$ queries, can we solve the collision problem in $o(n^{1/3})$ queries? The answer is yes: to solve the collision problem, we randomly pick $O(\sqrt{n})$ elements, and run the element distinctness algorithm on those $O(\sqrt{n})$ elements. If f is 2-to-1, by birthday paradox, we can see a collision with constant probability, and so it reduces to the element distinctness problem on $O(\sqrt{n})$ elements. Thus, the above reduction solves the collision problem in $o(n^{1/3})$ queries, a contradiction.

Now, we have $O(n^{3/4})$ upper bound, and $\Omega(n^{2/3})$ lower bound. People are always interested in closing the gap. It turns out that the lower bound is tight. Ambainis gave a sophisticated $O(n^{2/3})$-query algorithm for the element distinctness problem based on quantum walks.

4 Oracle Separation to SZK $\not\subseteq$ BQP

Another application of the collision lower bound is an oracle relative to which **SZK** $\not\subseteq$ **BQP**. What is **SZK**? It stands for statistical zero knowledge, which means a protocol where a verifier interact with a prover, and the result of this interaction is that the verifier become convinced that some statement is true, but he does not learn anything else. In particular, the verifier does not gain the ability to convince anyone else that the statement is true. It sounds paradoxical at beginning, but there is a canonical example to illustrate the point. Let us consider the graph non-isomorphism problem.

Graph Non-isomorphism Problem. Given two graphs G and H, decide whether G and H are not isomorphic. (The answer is yes if G and H are not isomorphic.)

Let us introduce two characters. The verifier Arthur is a skeptical polynomial time king, and the prover Merlin is an omniscient wizard, but not trustworthy. Merlin wants to convince Arthur that G and H are not isomorphic, but does not want Arthur to learn anything else. Arthur has to test Merlin to discover whether it is the case. How should we design the protocol?

The idea is what called the Coke-Pepsi test. Someone claims that Coke and Pepsi are different but does not able to pin point the difference. How can he convince you? One way to do it is to perform a blind test to see if he can tell them apart or not. Apply this idea, we can do the following.

- Arthur randomly picks one graph, permutes the vertices, send the result to Merlin, and ask, "which graph did I start with?"
- Merlin answers Arthur's challenge.

If G and H are not isomorphic, then any permutations of G are different to any permutations of H. Since Merlin is an omniscient wizard, he can always answer the challenge. On the other hand, suppose G and H are isomorphic, then the set of permutation of G are exactly the same as the set of permutation of H. Thus, Merlin has no way of knowing the answer, and can only guess correctly with probability $1/2$. We can run this protocol several times to make the probability as lower as we want.

Can Arthur learn anything from the protocol? Let us argue this intuitively. First of all, Arthur has the graphs at beginning, so he does not learn anything new from the graphs. Does he learn anything from Merlin? Merlin tells him which graph he started with. But Arthur already knew that. So, intuitively, Arthur has not learnt anything new, and yet, he is convinced that G and H are not isomorphic if it is the case. This is what we mean by zero knowledge proof.

The way to formalize the concept of not learning anything is that, Arthur should be able to simulate the entire interaction with Merlin, without even bringing Merlin into the picture at all. Note that this is the case for our example, Arthur just need to answer the challenge he produced by himself, who apparently knows the answer.

We will not give the formal definition of **SZK**. Basically, **SZK** is the class of problem for which yes answer can be proved by a protocol like this one. That is, there is a protocol for which a polynomial time Arthur interacts with computationally unbounded Merlin that satisfies three properties:

- Completeness: Arthur should accept when the answer is yes.
- Soundness: Arthur should reject with high probability whenever the answer is no.
- Zero Knowledge: Arthur should not learn anything beyond the answer.

We saw before that Grover lower bound gives an oracle relative to which $\mathbf{NP} \not\subseteq \mathbf{BQP}$. We claim that the collision lower bound gives an oracle relative to which $\mathbf{SZK} \not\subseteq \mathbf{BQP}$. The oracle encode the collision problem. The collision lower bound says that there is no efficient **BQP** algorithm to distinguish 1-to-1 functions from 2-to-1 functions. We need to show that this problem has a **SZK** protocol. The question is, can Merlin convince Arthur f is an 1-to-1 function without giving any other information? The idea is as follows.

- Arthur randomly picks an x, computes $f(x)$, send it to Merlin, and ask, "what is the inverse x?"
- Merlin answers Arthur's challenge.

If f is 1-to-1, then Merlin can always answer the challenge. If f is 2-to-1, then Merlin can only succeed with probability 1/2. Can Arthur learn anything? Merlin tells Arthur the inverse x of $f(x)$, which Arthur already knew it if he follows the protocol. In this case, Arthur learns nothing. However, Arthur may not be honest, and sends Merlin some y that he is interested in. He can then learn the inverse of y from Merlin. Intuitively, this seems not zero knowledge. Indeed, the protocol is zero knowledge only when Arthur is honest. However, there is a powerful theorem says that there is a way to modify the protocol so that it is zero knowledge even if Arthur is dishonest. Therefore, the collision problem has statistical zero knowledge protocol. By standard diagonalization trick, we can get an oracle relative to which $\mathbf{SZK} \not\subseteq \mathbf{BQP}$.

Like **BQP** vs. **NP**, we do not know the relation between **SZK** and **NP**. But under some hypothesis that most people believed in, **SZK** is contained in $\mathbf{NP} \cap co\mathbf{NP}$. Thus, we do not believe **NP**-complete problems have statistical zero knowledge proof protocols. On the other hand, they have computational zero knowledge proof protocols, assuming cryptographic assumption. If Merlin could encode his proof to Arthur cryptographically, and then selectively decode parts of the answer, then there is a computational zero knowledge protocols for **NP**-complete problems, due to Goldreich, Micali, Widgerson.

As an interesting side note, we mentioned that there is a whole industry of taking things in classical computation, putting a letter Q in from of them. This is no exception. There is a whole literature now about quantum statistical zero knowledge. Recall that honest verifier and dishonest verifier are equivalent for statistical zero knowledge (and also for computational zero knowledge.) Interestingly, in the quantum world, **NP**-complete problems have honest verifier quantum statistical zero knowledge protocol, but apparently not with dishonest verifier.

MIT OpenCourseWare
http://ocw.mit.edu

6.845 Quantum Complexity Theory
Fall 2010

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.