Quantum Complexity Theory

Jun-Ting Hsieh
TCS Toolkit Writing Project

1 Introduction

In quantum complexity theory, we are interested in the following question,

What types of problems can be solved efficiently by quantum algorithms?

In this paper, we hope to give an introduction to the complexity class BQP, the quantum analog of BPP. Currently, we know the following inclusions,

$$\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{BQP} \subseteq \mathsf{PP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}$$

We do not know whether $\mathsf{P} \neq \mathsf{BQP}$, but Shor famously showed that factoring is a problem in BQP but not known to be in P. On the other hand, whether $\mathsf{P} \neq \mathsf{PSPACE}$ is a long-standing open problem in complexity theory. Thus, a separation of P and BQP (or BPP and BQP) would be a major breakthrough in complexity theory.

Although whether these inclusions are strict or not is unknown, people have studied these relationships relative to oracles. In particular, we currently know that $\mathsf{BQP} \not\subseteq \mathsf{BPP}$ and $\mathsf{BQP} \not\subseteq \mathsf{PH}$ *relative to an oracle*. These are important and fascinating results that provide "evidence" about the power of quantum algorithms. However, they do **not** imply $\mathsf{BQP} \not\subseteq \mathsf{BPP}$ or $\mathsf{BQP} \not\subseteq \mathsf{PH}$ in the real-world. For example there are oracles A, B such that $\mathsf{P}^A = \mathsf{NP}^A$ and $\mathsf{P}^B \neq \mathsf{NP}^B$, but these do not prove or disprove $\mathsf{P} = \mathsf{NP}$.

This paper should be viewed as an introductory lecture on the BQP class. We will assume that readers are familiar with the materials from Ryan's lectures on quantum computation (e.g. quantum states, bra-ket notation, unitary matrix, quantum circuits, etc) and complexity theory (e.g. $\mathsf{BPP}, \mathsf{PSPACE}, \mathsf{PH}, \mathsf{AC}^0$, etc). This paper is structured as follows. Section 2 is an introduction of BQP and important concepts including universal gates and uncomputing. In Section 3, we will prove the inclusions $\mathsf{BPP} \subseteq \mathsf{BQP} \subseteq \mathsf{PP}$. Finally, in Section 4 we will discuss results on oracle separation of BPP, BQP, and PH.

2 BQP

Definition 2.1 (BQP)**.** *A language L is in* BQP *if there exists a (polynomial-time) uniform family of quantum circuits $\{C_n\}$ such that for input $x \in \{0,1\}^n$,*

- *If $x \in L$, then C_n accepts with probability at least $2/3$.*
- *If $x \notin L$, then C_n accepts with probability at most $1/3$.*

This closely resembles the definition of BPP. Similar to BPP, the error can be reduced by repeating the computation and taking the majority answer. Below are some technical details,

1. The input x is encoded as a state $|x\rangle$ with a polynomial number $q(n)$ of ancillary qubits initialized to $|0\rangle$, so the circuit C_n takes in the state $|x\rangle |0\rangle^{\otimes q(n)}$. The output is obtained by measurement on an output qubit.

2. Only a small set of local quantum gates is allowed in the quantum circuits, but we can choose any *universal* gate set for our algorithms and analysis. This will be further discussed in Section 2.2.

3. In a quantum algorithm, we can make multiple measurements at any time. The *principle of deferred measurement* states that measurements can always be moved to the end of the circuit, and the *principle of implicit measurement* states that we can assume that all qubits at the end of a quantum circuit are measured.

2.1 Common Gates

Some important and common quantum gates:

- *Hadamard* gate: $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.
- Pauli-X (NOT) gate: $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.
- *Phase* gate: $P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$.
- *Controlled-NOT* (CNOT) gate: $|x, y\rangle \mapsto |x, x \oplus y\rangle$.
- *Toffoli* (CCNOT) gate: $|x, y, z\rangle \mapsto |x, y, z \oplus xy\rangle$.

2.2 Universal Quantum Gates

Universal boolean gates. In classical circuits, we normally only allow the AND, OR, and NOT gates. It can be proved that the set $\{\text{AND}, \text{OR}, \text{NOT}\}$ is *universal*, meaning that any boolean function can be computable by circuits using just these gates. In fact, a single NAND gate is universal ($\text{NAND}(a, b) = \text{NOT}(\text{AND}(a, b))$), and we can easily convert a size-s circuit C with AND/OR/NOT gates into a size$-O(s)$ circuit C' with NAND gates. Moreover, since the gates can have fan-out more than 1, it is convenient to introduce the COPY gate that simply duplicates the output.

Universal quantum gates. Similarly, we only allow the quantum circuit to have a small *universal* set of quantum gates. Recall that quantum operations are unitary transformations. Thus, a gate set $\mathcal{G}$ is universal if we can approximate any unitary U to arbitrary accuracy using gates in $\mathcal{G}$. There are several choices of universal quantum gates, for example, $\{\text{CNOT}, H, \pi/8\}$ and $\{\text{CCNOT}, H\}$[1].

A natural question is, are some universal gate sets "better" than others? Fortunately, the answer is no via the following important theorem,

Theorem 2.2 (Solovay-Kitaev)**.** *Suppose $\mathcal{G}$ is universal and closed under inverses. Then, for arbitrary $\epsilon > 0$, any unitary operator U (on a d-dimensional Hilbert space) can be approximated to within precision ϵ using $O(\text{polylog}(1/\epsilon))$ gates in $\mathcal{G}$.*[2]

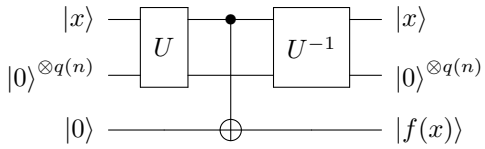
Remark 2.3. *The main takeaway from this theorem is that different universal gate sets can simulate one another efficiently with at most polynomial increase in the number of gates. Thus, for the class* BQP*, we can choose any universal gate set for our convenience.*

[1] This is only for real matrices, but this is actually enough for quantum circuits.
[2] See [DN05] for a more precise statement and proof.

2.3 Uncomputing

A typical quantum algorithm performs operations on qubits and then make a measurement on a single qubit. Normally, it is fine to simply ignore the other qubits (usually called "garbage"). However, this will be a problem if a quantum circuit is calling another quantum circuit as a subroutine, because we may input a state in superposition and hope to have interference between the results. The uncleaned garbage may prevent this.

There is a simple *uncomputing* trick, using the fact that every quantum operation is reversible: run the circuit, copy the output qubit to a separate register, and then reverse the whole thing:



Uncomputing is an important procedure in quantum computation, and there is a formal BQP *subroutine theorem* regarding this. In particular, it implies the following,

Corollary 2.4. $\mathsf{BQP}^{\mathsf{BQP}} = \mathsf{BQP}$.

3 Relationship to other complexity classes

3.1 Quantum Simulation of Classical Circuits

It is not surprising that quantum computers are at least as powerful as randomized algorithms. Indeed, we can simulate classical circuits with quantum circuits.

Theorem 3.1. $\mathsf{BPP} \subseteq \mathsf{BQP}$.

Proof. We pick $\{\text{CCNOT}, H\}$ as our universal quantum gates. We know that the CCNOT (Toffoli) gate can simulate the classical NAND gate and COPY (fan-out). Thus, since the NAND gate is universal for classical computation, we can convert any polynomial-size classical circuit into an equivalent polynomial-size quantum circuit.

In addition, we can generate random bits by Hadamard gates. A measurement on $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ w.r.t. the standard basis will give a random qubit: $|0\rangle$ or $|1\rangle$ with probability $1/2$ (this is allowed by the principle of deferred measurement). Thus, any circuit that takes random bits can be simulated by quantum circuits. □

3.2 Classical Simulation of Quantum Circuits

In this section, we show upper bounds on BQP. First, the proof of $\mathsf{BQP} \subseteq \mathsf{PSPACE}$ is adapted from [NC02]. A better bound $\mathsf{BQP} \subseteq \mathsf{PP}$ was originally given by [ADH97]. Here we present an elegant proof from Ryan's lecture notes.

Theorem 3.2. $\mathsf{BQP} \subseteq \mathsf{PSPACE}$.

Proof. The idea is that given a quantum circuit C_n and input x, rewrite the probability of outputting 1 as a huge sum of terms that can be efficiently computed, using the fact that each gate is local (at most 3-qubit gates). Then, iterating over the summation will take exponential time but only polynomial space.

Let $|\psi_0\rangle \in (\mathcal{C}^2)^{\otimes q(n)}$ be the initial input state. A quantum circuit with $m = \text{poly}(n)$ gates is just the transform $|\psi_0\rangle \mapsto |\psi\rangle = U_m U_{m-1} \cdots U_1 |\psi_0\rangle$, where $U_1, \ldots, U_m$ are gates acting on at most 3 qubits. The output is a measurement on the output qubit, and the probability of outputting 1 is determined by the projection operator $\Pi_1 = |1\rangle\langle 1|$,

$$\mathbf{Pr}[\text{Accept}] = \langle\psi|\Pi_1|\psi\rangle = \langle\psi_0|U_1^\dagger \cdots U_m^\dagger \Pi_1 U_m \cdots U_1|\psi_0\rangle \tag{1}$$

Since $\{|y\rangle : y \in \{0,1\}^{q(n)}\}$ forms an orthonormal basis, we can insert the identity $I = \sum_y |y\rangle\langle y|$ anywhere,

$$\begin{aligned}\mathbf{Pr}[\text{Accept}] &= \langle\psi_0| \left(\sum_{y_1} |y_1\rangle\langle y_1|\right) U_1^\dagger \left(\sum_{y_2} |y_2\rangle\langle y_2|\right) U_2^\dagger \cdots \Pi_1 \cdots U_1 \left(\sum_{y_{2m+2}} |y_{2m+2}\rangle\langle y_{2m+2}|\right) |\psi_0\rangle \\ &= \sum_{y_1,\ldots,y_{2m+2} \in \{0,1\}^{q(n)}} \langle\psi_0|y_1\rangle \, \langle y_1|U_1^\dagger|y_2\rangle \, \langle y_2|U_2^\dagger|y_3\rangle \cdots \langle y_{2m+1}|U_1|y_{2m+2}\rangle \, \langle y_{2m+2}|\psi_0\rangle\end{aligned} \tag{2}$$

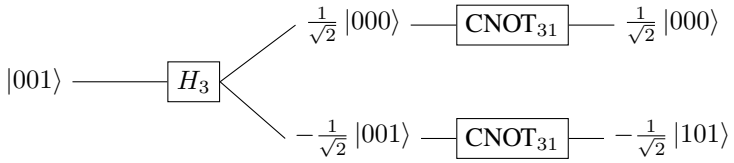
Each summand is a product of $\text{poly}(n)$ complex numbers, and since each U_i is a local transformation, we can compute each $\langle y_i|U_j|y_{i+1}\rangle$ in polynomial time/space to a small error. Thus, the entire product can be computed in $\text{poly}(n)$ time/space. By iterating over the summation, we can compute the probability of outputting 1 in polynomial space. □

The next result is a better bound on BQP. For those who understand some quantum mechanics, the computation paths we will see in the proof is basically the concept of *Feynman path integral*. Recall that PP is the class of problems solved by probabilistic TMs with error probability less than $1/2$.

Theorem 3.3. BQP $\subseteq$ PP.

Proof. Here we select the $\{H, \text{CNOT}, \text{CCNOT}\}$ gates. The main idea is that our restriction to these gates allow us to write out a "computation tree", and the coefficients of the final state can be written as sums of paths in the tree. We only need the error probability to be less than $1/2$, and it turns out that randomly picking two paths in the tree and evaluating the two leaves give us an error strictly smaller than $1/2$.

The key observation is that given a state $|x\rangle$, the CNOT and CCNOT gates only flip one qubit ($|x\rangle \mapsto |y\rangle$), whereas H splits the state into a superposition with equal magnitude ($|x\rangle \mapsto \frac{1}{\sqrt{2}}(|y_1\rangle \pm |y_2\rangle)$). Thus, starting from the initial state $|\psi_0\rangle \in (\mathbb{C}^2)^{\otimes q(n)}$, we can build a tree with depth h and 2^h leaves, where $h = \text{poly}(n)$ is the number of H gates. For example, the following is a computation tree of 3 qubits starting at $|001\rangle$,



At the leaf of the tree, each leaf is $\pm\frac{1}{2^{h/2}}|y\rangle$ for some $y \in \{0,1\}^{q(n)}$. Next, let P denote a path from the root to a leaf, and define $\text{label}(P), \text{sign}(P)$ to be the state y and the sign at that leaf. Then, for the final state $|\psi\rangle = \sum_y \alpha_y |y\rangle$, we can calculate the coefficient α_y by

$$\begin{aligned}|\psi\rangle = \frac{1}{2^{h/2}} \sum_P \text{sign}(P) \, |\text{label}(P)\rangle &= \frac{1}{2^{h/2}} \sum_{y \in \{0,1\}^{q(n)}} \left(\sum_{P:\text{label}(P)=y} \text{sign}(P)\right) |y\rangle \\ \implies \alpha_y^2 = \frac{1}{2^h} \sum_{\substack{P,P': \\ \text{label}(P)=\text{label}(P')=y}} &\text{sign}(P)\,\text{sign}(P')\end{aligned} \tag{3}$$

With the coefficients α_y, we can compute the probability of outputting 1 by the projection operator $\Pi_1 = |1\rangle\langle 1|$, $\mathbf{Pr}[C_n(x) = 1] = \langle\psi|\Pi_1|\psi\rangle = \sum_{y:y_1=1} \alpha_y^2$. Thus,

$$\begin{aligned}\mathbf{Pr}[C_n(x) = 1] - \mathbf{Pr}[C_n(x) = 0] &= \sum_y \alpha_y^2 \cdot (-1)^{\mathbf{1}(y_1=0)} \\ &= \frac{1}{2^h} \sum_{P,P':\text{label}(P)=\text{label}(P')} \text{sign}(P)\,\text{sign}(P')(-1)^{\mathbf{1}(\text{label}(P)_1=0)}\end{aligned} \tag{4}$$

Given input x, we randomly simulate two paths P, P' in the computation tree, and compute $\text{sign}(P)$, $\text{sign}(P')$ and $\text{label}(P)$, $\text{label}(P')$. This only takes $O(h) = \text{poly}(n)$ time. Then,

- If $\text{label}(P) \neq \text{label}(P')$, then accept or reject with probability $1/2$.
- If $\text{label}(P) = \text{label}(P')$, then accept iff $\text{sign}(P)\,\text{sign}(P')(-1)^{\mathbf{1}(\text{label}(P)_1=0)} > 0$.

Observe that this quantity is a summand in Equation 4. Let $q = \mathbf{Pr}[\text{label}(P) = \text{label}(P')] > 0$.

$$\begin{aligned}\mathbf{Pr}[\text{Accept}] - \mathbf{Pr}[\text{Reject}] &= q \cdot \mathbb{E}[\text{sign}(P)\,\text{sign}(P')(-1)^{\mathbf{1}(\text{label}(P)_1=0)} | \text{label}(P) = \text{label}(P')] \\ &\propto \mathbf{Pr}[C_n(x) = 1] - \mathbf{Pr}[C_n(x) = 0]\end{aligned} \tag{5}$$

Thus, by the definition of BQP, $\mathbf{Pr}[\text{Accept}] - \mathbf{Pr}[\text{Reject}]$ is strictly positive (resp. negative) if $x \in L$ (resp. $x \notin L$). Thus, we have classically achieved error less than $1/2$ in polynomial time. This proves that $\mathsf{BQP} \subseteq \mathsf{PP}$. □

4 Oracle Separation

In complexity theory, the study of complexity classes relative to an oracle can provide insights to the complexity classes. Recall that an oracle A is a language $L \subseteq \{0,1\}^*$, and given access to the oracle, the algorithm can query any input x and know whether $x \in L$ immediately. For two complexity classes $C_1 \subseteq C_2$, it is interesting to see whether there exists an oracle A such that $C_1^A = C_2^A$ or $C_1^A \neq C_2^A$. Note again that $C_1^A \neq C_2^A$ does **not** prove that $C_1 \neq C_2$.

Generally, people study quantum algorithms in the "black-box model". In this setting, the algorithm is given a black-box function f and is able to query on any x and obtain $f(x)$ in one step. The task is to solve a problem with as few queries as possible. It is well-known that a separation in *query complexity* in the black-box model implies an oracle separation. Thus, the common strategy to separate BQP and a complexity class $\mathcal{C}$ is as follows,

1. Construct a function f promised to have either property 1 or property 2. The task is to determine which property f satisfies by queries to f, viewing f as a black-box.
2. Find a quantum algorithm that achieves this using a small number of queries.
3. Prove a lower-bound on the number of queries required for algorithms in $\mathcal{C}$.

Oracles in quantum circuits can be defined as "oracle gates". In Ryan's lecture, oracle gates are defined as transforming $|x\rangle \to (-1)^{f(x)}|x\rangle$ for boolean functions f (phase oracle). Here, we use an equivalent and more generalized definition. For a black-box function $f : \{0,1\}^m \to \{0,1\}^n$, the oracle gate is the following transformation,

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle \tag{6}$$

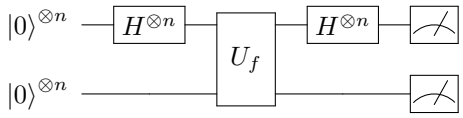
4.1 Simon's Problem: BQP and BPP

We know that there is an oracle A such that $\mathsf{BPP}^A = \mathsf{BQP}^A$ (any PSPACE-complete oracle A works because $\mathsf{P}^A = \mathsf{PSPACE}^A$). Thus, the next question is, is there an oracle B such that $\mathsf{BPP}^B \neq \mathsf{BQP}^B$? The answer is yes due to Simon's algorithm [Sim97]. Simon's problem is an example such that quantum algorithms provably require exponentially fewer queries than classical algorithms.

Simon's problem. We are given a black-box $f : \{0,1\}^n \to \{0,1\}^n$ promised that either

- f is 1-to-1, or
- there exists an unknown string $s \neq 0$ such that for any $x \neq y$, $f(x) = f(y)$ if and only if $x \oplus s = y$. (a 2-to-1 function).

For classical randomized algorithms, it is not hard to prove that $\Theta(2^{n/2})$ queries are necessary to solve the problem with constant probability. For quantum algorithms, Simon's algorithm shows that $O(n)$ queries are sufficient. The following is the quantum circuit for Simon's algorithm.



Suppose that f satisfies the second condition. Using 2 registers, each with n qubits initialized to 0,

1. Apply the H gates to the first n qubits,
$$|\psi\rangle = \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} |x\rangle\,|0\rangle^{\otimes n}$$
2. Query f and store the result in the second register,
$$|\psi\rangle = \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} |x\rangle\,|f(x)\rangle$$
3. Measure the second register and discard it. Since $f(x) = f(x+s)$, the resulting state will be,
$$|\psi\rangle = \frac{1}{\sqrt{2}}(|x\rangle + |x+s\rangle) \quad \text{for some } x$$
4. Apply the H gates again,
$$|\psi\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{y\in\{0,1\}^n} ((-1)^{x\cdot y} + (-1)^{(x+s)\cdot y})\,|y\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}(1 + (-1)^{s\cdot y})\,|y\rangle$$
5. Make a measurement. We will get a y that satisfies $s \cdot y = 0 \pmod 2$.

   From linear algebra, we can uniquely determine s if we have n linearly independent equations $s \cdot y_i = 0$ by solving the system of linear equations. Thus, by repeating the procedure $O(n)$ times, with constant probability, we can get linearly independent equations that uniquely determine s in $\mathrm{poly}(n)$ time.

This quantum algorithm determines s with $O(n)$ queries and $\mathrm{poly}(n)$ time, whereas classical algorithms require $\Theta(2^{n/2})$ queries. Thus, we have the following,

Theorem 4.1. *There is an oracle relative to which* $\mathsf{BPP} \neq \mathsf{BQP}$.

Remark 4.2. *Recall that the Hadamard transform is a quantum Fourier transform on $\mathbb{F}_2^n$. Simon's algorithm is an example of using Fourier transform to extract "hidden periodicity information". This turns out to be a crucial ingredient in the famous Shor's factoring algorithm [Sho99].*

4.2 Forrelation: BQP and PH

The relationship between BQP and the polynomial hierarchy is a big open problem (we know that BQP $\subseteq$ PSPACE and PH $\subseteq$ PSPACE). We do not even know the relationship between BQP and NP. Nevertheless, some results are known relative to an oracle. For example, [BV97] constructed a "Recursive Fourier Sampling" problem which yields an oracle A relative to which $\mathsf{BQP}^A \not\subseteq \mathsf{MA}^A$, a significant result even though MA is low in the hierarchy. In 2018, the oracle separation of BQP and PH was finally proved by [RT19], building on works from [Aar10, AA18].

Many famous results on the relativized polynomial hierarchy (PH with an oracle) utilize a fundamental connection between PH and AC^0, which reduces the problem to proving a lower bound on the size of constant-depth circuits[3]. Thus, the goal is to find a problem solved by quantum algorithms with a few queries, but cannot be solved by quasi-polynomial sized circuits.

Forrelation problem [Aar10]. Given $(x, y) \in \{\pm 1\}^{2N}$ sampled from the *Forrelation* distribution D, the task is to detect the correlation between x and y, i.e. distinguish D from the uniform distribution U. Here $N = 2^n$, so we view x, y as truth tables of black-box boolean functions $\{0,1\}^n \to \{\pm 1\}$, and the quantum algorithm can obtain x_i, y_j with queries $i, j \in \{0,1\}^n$.

[Aar10] gave a quantum algorithm that solves this with just 1 query in $O(\log N) = O(n)$ time. Next, it suffices to show that D *fools* constant-depth circuits with $\mathrm{quasipoly}(N)$ (or $2^{\mathrm{poly}(n)}$) gates, i.e. circuits cannot distinguish between D and the uniform distribution U. Aaronson "almost" did it, in particular he proved an oracle separation for the relational version of BQP and PH (the version where we allow problems with many valid outputs). [RT19] completed the puzzle by slightly modifying the Forrelation distribution and proving a lower bound for boolean circuits, finally resolving this huge open problem in quantum complexity theory. Here is their main theorem,

Theorem 4.3. *There exists an explicit distribution D over $\{\pm 1\}^{2N}$ such that:*

1. *There exists an $O(\log N)$ time quantum algorithm that makes one query, and distinguishes D and U with advantage $\Omega(1/\log N)$.*
2. *No boolean circuit of size $\mathrm{quasipoly}(N)$ and constant depth that distinguishes D and U with with advantage better than $\mathrm{polylog}(N)/\sqrt{N}$.*

We can amplify the advantage by $\mathrm{polylog}(N)$ repetitions, achieving $1 - 1/\mathrm{poly}(N)$ for quantum but still $\widetilde{O}(1/\sqrt{N})$ for circuits. Then, using the connection between AC^0 and PH, we finally get the main result,

Corollary 4.4. *There is an oracle relative to which* $\mathsf{BQP} \not\subseteq \mathsf{PH}$.

We will present the paper of [RT19] in this section. We will define the Forrelation distribution D (Section 4.2.1), and the "advantage" of distinguishing between D and U (Section 4.2.2). The quantum algorithm is shown in Section 4.2.3, and finally the lower bound for circuits is briefly discussed in Section 4.2.4. We will only present the high-level idea and leave out many details, since the full paper is too complicated to include in this paper[4].

4.2.1 The Forrelation Distribution

Let $N = 2^n$, $\epsilon = \Omega(1/\log N)$. Let $H \in \mathbb{R}^{N \times N}$ be the Hadamard matrix where $H_{ij} = \frac{1}{\sqrt{N}}(-1)^{\langle i,j \rangle}$ indexed by $i, j \in \mathbb{F}_2^n$. The Forrelation distribution is defined as follows,

1. $x' \sim \mathcal{N}(0, \epsilon I_N)$. $y' = Hx'$. $z' = (x', y') \in \mathbb{R}^{2N}$.
2. Truncate each z'_i to $[-1, 1]$.

[3] The concept was originally used to prove an oracle separation for PH and PSPACE. It is a bit complicated, so we will just take it for granted.
[4] See https://www.youtube.com/watch?v=QfVgiNNPVpo for a nice presentation of the result.

3. For each i, sample $z_i \in \{\pm 1\}$ such that $\mathbb{E}[z_i] = z'_i$. Output $z = (x, y) \in \{\pm 1\}^{2N}$.

Remark 4.5. *Recall that H is a Fourier transform on $\mathbb{Z}_2^n$, so y is close to the Fourier transform of x. The task is to detect the correlation between x and y, hence the name "Forrelation".*

The first step is equivalent to drawing z' from a multi-variate Gaussian distribution G on $\mathbb{R}^{2N}$ with covariance matrix $\epsilon \begin{bmatrix} I_N & H \\ H & I_N \end{bmatrix}$. Note that x and y themselves are uniform and independent, but they are correlated.

A technical part of the paper is proving that for the functions f we are concerned with, $\mathbb{E}_{z\sim D}[f(z)] \approx \mathbb{E}_{z'\sim G}[f(z')]$ up to constant factors. Thus, all we need to do is to calculate expectations over the multi-variate Gaussian G, allowing us to use some nice properties of Gaussian distributions.

4.2.2 Pseudorandom Distributions

We would like D to *fool* constant-depth circuits. Recall from Ryan's lectures on pseudorandom distributions: let U be the uniform distribution, and D is a distribution that fools a function class $\mathcal{C}$ if for any $f \in \mathcal{C}$,

$$\mathbb{E}_{x\sim D}[f(x)] \approx \mathbb{E}_{x\sim U}[f(x)]$$

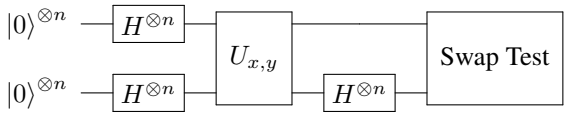
Define the "advantage" of an algorithm A over D as,

$$\alpha = \left| \mathop{\mathbf{Pr}}_{x\sim D}[A \text{ accepts } x] - \mathop{\mathbf{Pr}}_{x\sim U}[A \text{ accepts } x] \right| \tag{7}$$

Thus, our goal is to prove that the Forrelation distribution is pseudorandom for AC^0, i.e. any $\mathrm{quasipoly}(N)$-size constant-depth circuits have small advantage.

4.2.3 Quantum Algorithm

The quantum algorithm for Forrelation is proposed by [AA18]. Our input is $(x, y) \in \{\pm 1\}^{2N}$, which can be viewed as truth tables. We assume that the quantum algorithm can query them as phase oracles such that with input $\sum_i \alpha_i |i\rangle$, the oracle outputs $\sum_i \alpha_i x_i |i\rangle$ (or $\sum_i \alpha_i y_i |i\rangle$).



The swap test is a simple procedure that checks how much two quantum states differ. For two states $|\psi\rangle$ and $|\phi\rangle$, the swap test outputs 1 with probability $\frac{1}{2} + \frac{1}{2} + |\langle\psi|\phi\rangle|^2$.

1. Apply H on both registers, and query x and y respectively,

$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{i\in\{0,1\}^n} x_i |i\rangle, \quad |\psi_2\rangle = \frac{1}{\sqrt{N}} \sum_{j\in\{0,1\}^n} y_j |j\rangle$$

2. Apply H on the second state,

$$|\psi_2\rangle = \frac{1}{N} \sum_{j,k\in\{0,1\}^n} (-1)^{j\cdot k} y_j |k\rangle$$

3. The swap test,

$$\mathbf{Pr}[\text{Accept}] = \frac{1}{2} + \frac{1}{2}|\langle\psi_1|\psi_2\rangle|^2 = \frac{1}{2} + \frac{1}{2}\frac{1}{N^{3/2}}\sum_{i,j}(-1)^{i\cdot j}x_i y_j \equiv \frac{1+\varphi(x,y)}{2},$$
$$\text{where } \varphi(x,y) = \frac{1}{N^{3/2}}\sum_{i,j\in\{0,1\}^n}(-1)^{i\cdot j}x_i y_j = \frac{1}{N}x^\top H y \tag{8}$$

The next step is to analyze $\varphi(x,y)$ under the distributions U and D. The following claims are straightforward, using the fact that $y' = Hx'$ for $(x',y') \sim G$.

Claim 4.6. $\mathbb{E}_{(x,y)\sim U}[\varphi(x,y)] = 0$.

Claim 4.7. $\mathbb{E}_{(x',y')\sim G}[\varphi(x',y')] = \epsilon = \Omega(1/\log N)$.

It can be proved (with a bit of calculation) that $\mathbb{E}_{(x,y)\sim D}[\varphi(x,y)] \approx \mathbb{E}_{(x',y')\sim G}[\varphi(x',y')]$ up to a factor of 2. Thus, this proves the first point in Theorem 4.3 since we choose $\epsilon = \Omega(1/\log N)$.

4.2.4 Lower Bound for Circuits

The lower bound for circuits is the main contribution of the paper. The authors proved that all circuits with quasi-polynomial size and constant depth have small advantage. We will only give a high-level explanation of the proof.

Fourier analysis. Recall from Ryan's lecture on analysis of boolean functions, we can write a function $f : \{\pm1\}^{2N} \to \{\pm1\}$ as its Fourier expansion (a polynomial),

$$f(x) = \sum_{S\subseteq[2N]}\widehat{f}(S)x_S, \quad \text{where } x_S = \prod_{i\in S}x_i \tag{9}$$

It is well-known that AC^0 is *approximable* by low-degree polynomials. The key observation is that AC^0 has *sparse* low-degree approximations, i.e. bounded L_1 norm on the Fourier coefficients, specifically $\sum_{|S|=k}|\widehat{f}(S)| \le \text{polylog}(N)^k$. In contrast, Equation 8 is a degree-2 polynomial with *dense* coefficients.

Then, we can write the advantage (Equation 7) as

$$\begin{aligned}\mathbb{E}_{z\sim G}[f(z)] - \mathbb{E}_{z\sim U}[f(z)] &= \sum_{S\subseteq[2N]}\widehat{f}(S)\left(\mathbb{E}_G[z_S] - \mathbb{E}_U[z_S]\right) = \sum_{\ell=1}^{N}\sum_{|S|=2\ell}\widehat{f}(S)\mathbb{E}_G[z_S] \\ &\le \sum_{\ell=1}^{N}\sum_{|S|=2\ell}|\widehat{f}(S)|\cdot\epsilon^\ell\frac{\ell!}{\sqrt{N}^\ell} \le \sum_{\ell=1}^{N}\text{polylog}(N)^{2\ell}\cdot\epsilon^\ell\frac{\ell!}{\sqrt{N}^\ell}\end{aligned} \tag{10}$$

where the inequalities are due to nice properties of Gaussian distributions and the bound on the Fourier coefficients. This is almost what we would like if we ignore the large-degree terms (i.e. $\ell \ge \sqrt{N}$). To get around this, the authors use brilliant ideas from random walks (which we will not describe here) to obtain a better upper bound similar to Equation 10, but with the crucial difference that all high-degree terms can be safely ignored. This proves a $\text{polylog}(N)/\sqrt{N}$ upper bound on the advantage, which completes the proof of Theorem 4.3.

5 Conclusion

In this paper, we defined the BQP class and presented some known inclusions and oracle separations. Unsurprisingly, there are still many open problems, for example the relationship between BQP and NP, or whether there is an oracle A such that $\mathsf{NP}^A \subseteq \mathsf{BQP}^A$ but $\mathsf{PH}^A \not\subseteq \mathsf{BQP}^A$. Of course, BQP is just one of the many quantum complexity classes (e.g. QMA and QIP), and they are all exciting research fields with fascinating results and open problems.

6 Resources

The definition of BQP and some important properties of quantum computing can be found in Chapter 4 of the famous book *Quantum Computation and Quantum Information* [NC02], and two sources by Scott Aaronson: *Quantum Computing Since Democritus* and Lecture 2 of his notes. I synthesize some important topics from these sources and present them without going too deep into unnecessary details.

The proof of $\mathsf{BQP} \subseteq \mathsf{PSPACE}$ is based on Chapter 4.5.5 of [NC02], and the proof of $\mathsf{BQP} \subseteq \mathsf{PP}$ is based on Ryan's lecture notes. There are several different proofs of $\mathsf{BQP} \subseteq \mathsf{PP}$ out there, but I find Ryan's the easiest to understand.

Next, quantum oracle gates and Simon's problem are described in the two sources by Scott Aaronson. For the oracle separation of BQP and PH, the full paper [RT19] has all the details, but this blog post by Scott Aaronson and this video by Avishay Tal provide very good explanations of the paper without too many technical details. I would also like to thank Ryan for explaining the connection between PH and AC^0 during office hours. Overall, I tried to present high-level intuitions, hoping that my entire Section 4.2 is not too hard to understand.

References

[AA18] Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. *SIAM Journal on Computing*, 47(3):982–1038, 2018.

[Aar10] Scott Aaronson. Bqp and the polynomial hierarchy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 141–150, 2010.

[ADH97] Leonard M Adleman, Jonathan DeMarrais, and Ming-Deh A Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on computing*, 26(5):1411–1473, 1997.

[DN05] Christopher M Dawson and Michael A Nielsen. The solovay-kitaev algorithm. *arXiv preprint quant-ph/0505030*, 2005.

[NC02] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[RT19] Ran Raz and Avishay Tal. Oracle separation of bqp and ph. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 13–23, 2019.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[Sim97] Daniel R Simon. On the power of quantum computation. *SIAM journal on computing*, 26(5):1474–1483, 1997.