18.405J / 6.841J Project report: Quantum computing and #P-complete problems

May 6, 2016

Abstract

As part of the more general quest of understanding how quantum complexity classes relate to classical complexity classes, it is interesting to consider how quantum computing relates to #P-complete problems. In particular, this issue goes deeper than just the BQP containment $\mathsf{BQP} \subseteq \mathsf{P}^{\#\mathsf{P}}$. This report surveys some results which connect quantum computation with #P-complete problems, for a theoretical computer science audience. Basic familiarity with quantum computing and BQP is assumed. I survey the 1993 Bernstein-Vazirani result that $\mathsf{BQP} \subseteq \mathsf{P}^{\#\mathsf{P}}$, the 1999 Fenner-Green-Homer-Pruim result that $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{NP}^{\mathsf{NQP}}$, the 2004 Aaronson result $\mathsf{PostBQP} = \mathsf{PP}$, and the 2011 Aaronson-Arkhipov result $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{BPP}^{\mathsf{NP}^{\mathsf{BosonSampling}}}$.

1 Introduction

Quantum computing is a model of computation which generalizes probabilistic computation by exploiting quantum effects. More specifically, a classical probabilistic Turing machine has an associated transition matrix P, where $P_{x,y}$ is the probability of the machine transitioning to configuration y from configuration x in a single step. Since probabilities must sum to 1, this is a row stochastic matrix: $\sum_y P_{x,y} = 1$ for all x. The probability of a particular computational path $x_0 \to x_1 \to \cdots \to x_t$ occurring is $P_{x_0,x_1} P_{x_1,x_2} \cdots P_{x_{t-1},x_t}$. The total probability that the configuration of the machine is x_t after t steps is simply the sum of the probabilities of all computational paths which go from x_0 to x_t in t steps.

Now consider the quantum case. The stochastic transition matrices are replaced by matrices corresponding to quantum gates. Whereas all entries of the classical transition matrices had to be nonnegative real numbers corresponding to probabilities, the matrices corresponding to quantum gates are complex in general. Also, the row stochastic constraint in the classical case is replaced by the constraint that the matrices in the quantum case be unitary. Say we have a quantum circuit consisting of t gates $U_1, U_2, \ldots, U_t$. The total amplitude corresponding to a particular computational path $|x_0\rangle \to \cdots \to |x_t\rangle$ is given by $\langle x_t| U_t |x_{t-1}\rangle \langle x_{t-1}| U_{t-1} |x_{t-2}\rangle \cdots U_1 |x_0\rangle \in \mathbb{C}$ and the total amplitude to go from $|x_0\rangle$ to $|x_t\rangle$ is the sum of the amplitudes of every computational path from $|x_0\rangle$ to $|x_t\rangle$ (for physicists, this is essentially just a Feynman path integral). The crucially unique aspect of the quantum case which is not present in the classical case is the fact that since amplitudes are in $\mathbb{C}$ rather than $\mathbb{R}^+$, the amplitudes corresponding to different computational paths can interfere destructively. In fact, quantum algorithm design can be viewed as an attempt to orchestrate the circuit such that computational paths going to a wrong answer interfere destructively and cancel each other other, leaving only the right answer.

Since the amplitudes of a quantum computation are sums of exponentially many complex numbers, it seems natural to investigate the relationship between quantum computing and the classical counting class #P. First we recall the definition.

Definition 1.1 (#P)**.** *A function $f : \{0,1\}^* \to \mathbb{N}$ is in* #P *if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$:*

$$f(x) = \left|\left\{y \in \{0,1\}^{p(|x|)} : M(x,y) = 1\right\}\right|$$

It is also convenient to record here the definition of BQP, the class of problems generally considered to be efficiently solvable by a universal quantum computer.

Definition 1.2 (BQP)**.** *A language L is in* BQP *if and only if there exists a uniform family of polynomial-size quantum circuits $\{C_n : n \in \mathbb{N}\}$ such that*

- *$\forall n \in \mathbb{N}$, C_n inputs n qubits and outputs 1 bit*
- *$\forall x \in L$, $\Pr\big(C_{|x|}(x) = 1\big) \geq \frac{2}{3}$*
- *$\forall x \notin L$, $\Pr\big(C_{|x|}(x) = 1\big) \leq \frac{1}{3}$*

In Section 2 we will review a 1993 result of Bernstein and Vazirani [4] which says that the set of decision problems efficiently solvable with a #P oracle subsumes the set of decision problems efficiently solvable with a quantum computer: $\mathsf{BQP} \subseteq \mathsf{P}^{\#\mathsf{P}}$. In Section 3, based on the 1999 work of Fenner, Green, Homer, and Pruim [7], we define the language QAP (standing for Quantum Acceptance Possibility) which is complete for NQP, the quantum analog of NP. This language essentially encodes whether a quantum circuit has a nonzero acceptance probability on some input. We show that this language is also complete for the counting class $\mathsf{coC_{=}P}$ (co-Exact-Counting Polynomial-Time). Since $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{NP}^{\mathsf{coC_{=}P}}$, we find that this language is hard for $\mathsf{P}^{\#\mathsf{P}}$ under nondeterministic reductions. In Section 4, following Aaronson's 2005 paper [1], we define PostBQP – a variant of BQP in which one can postselect on a certain result occurring. This class is extremely powerful – we will see that PostBQP = PP. Since $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}}$, this implies that a quantum computer with postselection is able to solve #P-complete problems. In Section 5, which reviews parts of Aaronson and Arkhipov's 2011 paper [2], we consider an alternative and presumably quite weak (non-universal) model of computation based on the statistics of non-interacting bosons. Despite the simplicity of the model, we show that if a classical computer were able to efficiently simulate the output of a boson computer, then a $\mathsf{BPP}^{\mathsf{NP}}$ machine would be able to compute the square of the permanent of an arbitrary complex matrix (a #P-complete problem). Note that by Toda's Theorem, this would imply a collapse of PH.

2 BQP and $\mathsf{P}^{\#\mathsf{P}}$

In [4], Bernstein and Vazirani formalized many complexity-theoretic notions in the quantum setting (such as defining BQP) and also proved several foundational results, the most relevant of which for this report being that $\mathsf{BQP} \subseteq \mathsf{P}^{\#\mathsf{P}}$. We adapt their proofs here, first proving a similar but weaker result, and then moving on to the desired result.

Proposition 2.1. $\mathsf{BQP} \subseteq \mathsf{PSPACE}$

Proof. We show how to simulate a BQP circuit in PSPACE. Let the circuit consist of gates $U_1, U_2, \ldots, U_{t(n)}$ where $t(n)$ is some polynomial and n is the input length. Note that each gate acts on $O(1)$ qubits. Without loss of generality we can assume each gate acts on 3 qubits (for example, choose the Hadamard and Toffoli gates as the universal base set). Let the input be $|x\rangle$. The amplitude for state $|y\rangle$ is

$$\langle y| U_t U_{t-1} \cdots U_1 |x\rangle = \sum_{\{z_1,\ldots,z_{t-1}\}} \langle y| U_t |z_{t-1}\rangle \langle z_{t-1}| U_{t-2} |z_{t-2}\rangle \ldots \langle z_1| U_1 |x\rangle$$

Note that a classical computer can compute this sum recursively. To compute an amplitude after the U_t gate, it can compute an amplitude after the U_{t-1} gate, multiply by the relevant U_t matrix element, and add the result to a running total for the amplitude. It can reuse any space besides the space used to store the running total for the amplitude. So, if the space required to calculate an amplitude after t gates is $S(t)$, we have the recursion relation

$$S(t) = S(t-1) + O(l)$$

where $O(l)$ is the space required to store the sum of the amplitudes, which is polynomial because we only need to keep track of amplitudes to a reasonable precision. So by the recursion relation, it can compute arbitrary amplitudes in polynomial space. Hence the result follows, because to compute the total probability of the quantum circuit outputting 1 it needs only to sum over the squares of the relevant amplitudes, which it can do in polynomial space by reusing space.

□

Proposition 2.2. $\mathsf{BQP} \subseteq \mathsf{P}^{\#\mathsf{P}}$

Proof. Consider the decomposition as in the proof above:

$$\langle y| U_t U_{t-1} \cdots U_1 |x\rangle = \sum_{\{z_1,\ldots,z_{t-1}\}} \langle y| U_t |z_{t-1}\rangle \langle z_{t-1}| U_{t-2} |z_{t-2}\rangle \ldots \langle z_1| U_1 |x\rangle$$

To some approximation, the entries of the U_i are all complex rationals. Consider splitting the contributions to the amplitude of $|y\rangle$ into positive reals, negative reals, positive imaginaries, and negative imaginaries. We first consider the contributions from positive reals. Let c be some fixed constant which will control the accuracy of the simulation. Let M be a polynomial-time TM which takes as input a sequence of quantum gates, an initial state $|x\rangle$, a final state $|y\rangle$, a computational path from $|x\rangle$ to $|y\rangle$, and an integer k between 0 and 2^c. M outputs 1 if the input is valid and if $k < a \cdot 2^c$ where a is the amplitude of the path. Otherwise, it outputs 0. Note that the total contribution to the amplitude of $|y\rangle$ from the positive reals is given by

$$\frac{1}{2^c} \sum_{\text{paths from } |x\rangle \text{ to } |y\rangle, k} M(\{U_1, \ldots, U_t\}, |x\rangle, |y\rangle, \text{path}, k)$$

But this quantity can be computed in polynomial time with help from the #P oracle. Similarly, we can use this method to calculate the contributions from the negative reals, positive imaginaries, and negative imaginaries to find the total amplitude of $|y\rangle$ in $\mathsf{P}^{\#\mathsf{P}}$.

We now claim that for a $\mathsf{P}^{\#\mathsf{P}}$ machine to simulate a BQP machine, it suffices to compute a single amplitude. This is true because one can first amplify the success probability of the BQP

machine using a majority voting procedure, and then one can use uncomputing to ensure that the squared amplitude of the final state is highly concentrated on $|x\rangle|f(x)\rangle$, where $f(x)$ denotes the correct output of the circuit on input $|x\rangle$. This is a good exercise for the reader, and it is proven in [3]. Hence, the $\mathsf{P}^{\#\mathsf{P}}$ machine just needs to compute the amplitude of the state $|x\rangle|0\rangle$ after $|x\rangle$ goes through the modified BQP circuit. □

This is essentially the best bound we have on BQP.

3 QAP and $\mathsf{coC_{=}P}$

Of course, there are other questions one can ask about a quantum circuit than whether it can decide some language with bounded error. In 1999, Fenner, Green, Homer, and Pruim [7] considered the problem of determining whether a quantum circuit accepts with nonzero probability. Specifically, we are interested in the language QAP (Quantum Acceptance Possibility).

Definition 3.1 (QAP)**.**

$$\mathsf{QAP} = \{\langle C_n, x\rangle : C_n \textit{ encodes a uniform polynomial-size quantum circuit family that has non-zero probability of accepting } x\}$$

One can check that this is a natural complete language for NQP, the quantum analog of NP.

Definition 3.2 (NQP)**.** *A language L is in NQP if and only if there exists a uniform family of polynomial-size quantum circuits $\{C_n\}$ such that*

$$x \in L \iff \Pr\big(C_{|x|}(x) = 1\big) \neq 0$$

To see why this is a natural quantum analog of NP, note that a language is in NP if and only if there is a probabilistic polynomial-time TM with the property that a string is in the language if and only if the TM has a nonzero probability of accepting. We will show that NQP is equal to a certain classical counting class called $\mathsf{coC_{=}P}$, but first we need a few more definitions.

Definition 3.3 (GapP)**.** *Given any $L \in \{0,1\}^*$, let $L_x = \{y \in \{0,1\}^* : \langle x, y\rangle \in L\}$. A function $f : \{0,1\}^* \to \mathbb{Z}$ is in GapP if there is a language L in P and an integer k such that,*

$$f(x) = \frac{|\{0,1\}^{n^k} \cap L_x| - |\{0,1\}^{n^k} - L_x|}{2}$$

where $n = |x|$.

Note that if $f \in \mathsf{GapP}$, then $f(x)$ corresponds to the difference between the number of accepting and rejecting paths of some nondeterministic TM on input x.

Definition 3.4 ($\mathsf{C_{=}P}$)**.** *A language L is said to be in the class $\mathsf{C_{=}P}$ if there is a GapP function f such that for any x, $x \in L$ if and only if $f(x) = 0$. The class $\mathsf{coC_{=}P}$ is the set of all languages with complements in $\mathsf{C_{=}P}$.*

We will now show that $\mathsf{NQP} = \mathsf{coC_{=}P}$. We first show that $\mathsf{NQP} \subseteq \mathsf{coC_{=}P}$, due to Fortnow and Rogers [8].

Lemma 3.5. *For any quantum circuit family C_n of size bounded by a polynomial $s(n)$, there is a* GapP *function f and a constant c such that for all inputs x,*

$$\Pr\big(C_{|x|}(x) \text{ accepts}\big) = \frac{f(x)}{c^{2s(|x|)}}$$

Proof. Approximate the entries of the quantum gates by rationals such that for each quantum gate U, cU has all integral entries. Let the initial state be $|x\rangle$, and the final state before measurement be $|y\rangle = U_t \cdots U_1 |x\rangle$. Consider the state $c^t |y\rangle$, and note that each amplitude of this state consists of a sum of products of polynomial-time computable functions. But one can check that $\mathsf{FP} \subseteq \mathsf{GapP}$ and that GapP is closed under addition and multiplication (for proofs and more analysis of gap-definable counting classes, [6] is a good review). Hence, it follows that the probability that $C_{|x|}(x)$ accepts is $\frac{f(x)}{c^{2s(|x|)}}$ for some $f \in \mathsf{GapP}$. □

Note that it follows from this lemma that $L \in \mathsf{NQP} \Rightarrow L \in \mathsf{coC_{=}P}$. We now show the converse, due to Fenner, Green, Homer, and Pruim.

Lemma 3.6. *For any $f \in \mathsf{GapP}$, there exists a polynomial-time uniform family of quantum circuits C_n and a polynomial p such that, for all x of length n,*

$$\Pr\big(C_{|x|}(x) \text{ accepts}\big) = \frac{f(x)^2}{2^{p(n)}}$$

Proof. Let L be the language corresponding to f as in Definition 3.3, and let M be the polynomial time machine recognizing L (i.e., $M(x,y) = 1$ if and only if $\langle x, y\rangle \in L$). Let $m = n^k$, and define a circuit family as follows. On input $|x\rangle$, prepare the state $|x\rangle |0\rangle^{\otimes m} |0\rangle$. Now perform a Hadamard transform on the middle m qubits:

$$|x\rangle |0\rangle^{\otimes m} |0\rangle \mapsto \frac{1}{2^{m/2}} \sum_y |x\rangle |y\rangle |0\rangle$$

Next, using classical reversible computation, compute $M(x,y) \equiv b_y$ and write the answer into the last qubit, yielding

$$\frac{1}{2^{m/2}} \sum_y |x\rangle |y\rangle |b_y\rangle$$

Next, perform a Hadamard transform on the last $m+1$ qubits, yielding

$$|\psi\rangle = \frac{1}{\sqrt{2}} \frac{1}{2^m} \sum_y \sum_{y',b'} (-1)^{y\cdot y' + b_y b'} |x\rangle \left|y'\right\rangle \left|b'\right\rangle$$

Now consider the amplitude of $|x\rangle |0\rangle^{\otimes m} |1\rangle$:

$$\langle x| \langle 0|^{\otimes m} \langle 1| \otimes |\psi\rangle = \frac{1}{\sqrt{2}} \frac{1}{2^m} \sum_y (-1)^{y\cdot 0 + b_y 1} = \frac{1}{\sqrt{2}} \frac{1}{2^m} \sum_y (-1)^{b_y} = -\frac{1}{\sqrt{2}} \frac{1}{2^{m-1}} f(x)$$

Hence, defining the circuit to accept on measurement outcome $|x\rangle |0\rangle^{\otimes m} |1\rangle$, we have the desired result. □

The above lemma implies that if $L \in \mathsf{coC_{=}P}$, then $L \in \mathsf{NQP}$. Hence, putting the results together, we see that $\mathsf{NQP} = \mathsf{coC_{=}P}$. Since $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{NP}^{\mathsf{coC_{=}P}}$, we see that the problem of determining whether a quantum circuit accepts with non-zero probability is hard for $\mathsf{P}^{\#\mathsf{P}}$ under nondeterministic reductions.

4 PostBQP and PP

We now consider postselected BQP, PostBQP, the set of languages decidable by a fantasy version of quantum computation defined by Aaronson in 2004 [1]. This class is defined similarly to BQP, but now we have the ability to *postselect* on certain events happening. That is, we have the ability to instantly throw out all runs of a circuit where a certain event does not happen. This section closely follows his original paper.

Definition 4.1 (PostBQP)**.** *A language L is in* PostBQP *if and only if there exists a uniform family of polynomial-size quantum circuits $\{C_n\}$ such that for all inputs x,*

- *The first qubit of $C_n(|0\cdots 0\rangle\,|x\rangle)$ has nonzero probability of being measured to be $|1\rangle$.*
- *If $x \in L$, then the conditional probability of the second qubit being measured to be $|1\rangle$ assuming the first qubit is measured to be $|1\rangle$ is at least $2/3$.*
- *If $x \notin L$, then the conditional probability of the second qubit being measured to be $|1\rangle$ assuming the first qubit is measured to be $|1\rangle$ is at most $1/3$.*

One can check that PostBQP has strong closure properties. In particular, it is closed under union, intersection, and complement. Furthermore, being able to postselect on multiple qubits does not change the class, and $\mathsf{PostBQP} = \mathsf{BQP}^{\mathsf{PostBQP}}$ (assuming the queries are classical and nonadaptive).

This is a very powerful class. For example, it is not hard to show that $\mathsf{NP} \subseteq \mathsf{PostBQP}$. We can simply randomly guess a string, check if the string is a witness for the input, and if it is, set the first and second qubits to $|1\rangle$. If it's not a witness, set the second qubit to $|0\rangle$, and set the first qubit to $|1\rangle$ with an extremely small probability. Then with arbitrarily high probability, conditioned on the first qubit being measured $|1\rangle$, measurement of the second qubit will give the correct answer. In fact, we will prove that $\mathsf{PostBQP} = \mathsf{PP}$. As a consequence, since $\mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\#\mathsf{P}}$, postselection gives quantum computers the ability to solve #P-hard problems!

It should be noted that PostBQP has proven surprisingly useful. For example, as explained in Aaronson's original paper, after proving that $\mathsf{PostBQP} = \mathsf{PP}$ it trivially follows that PP is closed under intersection since PostBQP is trivially closed under intersection. But the question of whether PP is closed under intersection was open for eighteen years before being proven with some heavy machinery. This is one of several examples of quantum computing yielding insight into purely classical questions. In contrast, the proof that $\mathsf{PostBQP} = \mathsf{PP}$ is quite short. Another setting in which PostBQP is useful is in proving that certain quantum systems are not simulable by a classical computer unless the polynomial hierarchy collapses. As just one example, this method can be used to prove that quantum computation in which one uses only commuting gates cannot be simulated by a classical computer unless PH collapses to the third level [5]. The fact that $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PostBQP}}$ is used as an intermediate step in these proofs.

Proposition 4.2. $\mathsf{PostBQP} \subseteq \mathsf{PP}$

Proof. Since they form a universal basis set, we can assume without loss of generality that the circuit consists only of Hadamard and Toffoli gates. As in the earlier proofs, we know that the final amplitude of some basis state $|y\rangle$ is a sum of exponentially many contributions, each contribution being a product of matrix elements. Note that due to our choice of gates, each nonzero contribution

has the same magnitude; only the sign differs. If we denote these contributions, $a_{y,1}, \ldots, a_{y,N}$, then the probability of getting $|y\rangle$ upon measurement is $(a_{y,1} + \cdots + a_{y,N})^2 = \sum_{ij} a_{y,i} a_{y,j}$. We want to know which is greater: the probability of measuring $|11\ldots\rangle$ or the probability of measuring $|10\ldots\rangle$. We can do this in PP by selecting a computational path ending on $|1\ldots\rangle$ uniformly at random which has nonzero amplitude, and outputting 1 if the path has positive sign and ends on $|11\ldots\rangle$ or has negative sign and ends on $|10\ldots\rangle$. Otherwise, output 0. □

Proposition 4.3. PP ⊆ PostBQP

Proof. Consider $L \in \mathsf{PP}$. Then there exists a polynomial r and an efficiently computable function $g : \{0,1\}^{|x|} \times \{0,1\}^{r(|x|)} \to \{0,1\}$ such that $x \in L \iff |\{y : g(x,y) = 1\}| \geq 2^{r(|x|)-1}$. Hence it suffices to show that if $f : \{0,1\}^n \to \{0,1\}$ is efficiently computable and $s = |x : f(x) = 1|$, there exists a PostBQP circuit which decides whether $s < 2^{n-1}$ or $s \geq 2^{n-1}$.

To do so, first apply a Hadamard transform and use reversible classical computation to prepare the state

$$\frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$$

Now, apply a Hadamard transform to the first n qubits, and then postselect on those qubits being in $|0\rangle^{\otimes n}$. This puts the final qubit in the state

$$|\psi\rangle = \frac{(2^n - s)|0\rangle + s|1\rangle}{\sqrt{(2^n - s)^2 + s^2}}$$

Now, prepare another qubit in the state $\alpha|0\rangle + \beta|1\rangle$, and do a controlled-Hadamard operation to prepare the state $\alpha|0\rangle|\psi\rangle + \beta|1\rangle H|\psi\rangle$. Next, postselect on the second qubit being $|1\rangle$. This prepares the state

$$|\varphi_{\beta/\alpha}\rangle = \frac{\alpha s|0\rangle + \beta\sqrt{1/2}(2^n - 2s)|1\rangle}{\sqrt{\alpha^2 s^2 + (\beta^2/2)(2^n - 2s)^2}}$$

in the first qubit. Notice that if $s < 2^{n-1}$, the coefficients of both $|0\rangle$ and $|1\rangle$ are positive. Now, through elementary algebra, one can show that in this case, for some $i \in [-n, n]$,

$$|\langle +|\varphi_{2^i}\rangle| \geq \frac{1+\sqrt{2}}{\sqrt{6}}$$

where $|+\rangle \equiv \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The intuition is that changing the value of i rotates the direction of $|\varphi\rangle$ in the first quadrant in the plane spanned by $|0\rangle$ and $|1\rangle$, and some value of i will get it close to $|+\rangle$. On the other hand, if $s \geq 2^{n-1}$, then for no value of i will $|\varphi_{2^i}\rangle$ be in the first or third quadrants, and so one can show that $|\langle +|\varphi_{2^i}\rangle| \leq 1/\sqrt{2}$ in this case. Hence, a PostBQP circuit of polynomial size can distinguish between these two scenarios with high probability by repeating the algorithm with varying values of i. □

5 BosonSampling and the Permanent

In the last section we analyzed a modified, more powerful model of quantum computation and found that it is hard for #P-complete problems. In this section, we will consider a very simple model

of computing due to Aaronson and Arkhipov in 2011 based on the quantum statistics of bosons that appears to not even be universal for classical computation, let alone quantum computation. Nonetheless, one can prove that if it can be simulated efficiently by a classical computer, then one could approximate the squared permanent of an arbitrary complex matrix with a $\mathsf{BPP}^{\mathsf{NP}}$ machine. But since approximating the squared permanent of an arbitrary complex matrix is #P-hard, this would imply that $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{BPP}^{\mathsf{NP}}$. Applying Toda's Theorem [11], this implies that $\mathsf{P}^{\#\mathsf{P}} = \mathsf{PH} = \Sigma_3^{\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$ and the polynomial hierarchy collapses to the third level. Hence, even though it is believed that a boson computer (or any quantum computer) could not compute the permanent of an arbitrary matrix efficiently, an efficient classical simulator for a boson computer could be used to compute a permanent efficiently! This subtly is more easily understood after seeing the proof.

5.1 Computing with noninteracting bosons

If we have a quantum system consisting of two distinguishable particles and we find particle 1 at location a and particle 2 at location b, the state of the system is just $|ab\rangle$. But what if the particles are indistinguishable? In this case, if we catch one particle at a and one particle at b, we can't say that we caught particle 1 at a and particle 2 at b. All we can say is that there is one at a and one at b. What this means for our mathematics is that $|ab\rangle$ must describe the same physical state as $|ba\rangle$. Recalling that states correspond to rays in the hilbert space, we see $|ab\rangle = e^{i\phi}|ba\rangle$ for some ϕ, where we are assuming $|ab\rangle$ and $|ba\rangle$ are normalized. This in turn implies $|ab\rangle = e^{i2\phi}|ab\rangle$, from which we find $e^{i\phi} = \pm 1$. The $+$ sign corresponds to bosons and $-$ to fermions, so for bosons we have $|ab\rangle = |ba\rangle$. So then if we catch one boson at a and one at b, how is the final state described in terms of $|ab\rangle$ and $|ba\rangle$? The answer is that the state must be symmetrized: $\frac{1}{\sqrt{2}}(|ab\rangle + |ba\rangle)$. Note that this state treats $|ab\rangle$ and $|ba\rangle$ on a completely equal footing, and if the two particles are interchanged $a \leftrightarrow b$, the phase is simply multiplied by a $+1$ factor as desired. For the purposes of BosonSampling, a very interesting feature is that there appears to be some form of effective entanglement between the two particles just because they are identical: no interactions are required! (In other words, it is not possible to write the symmetrized state as a product state.) The presence of this effective entanglement suggests that perhaps we can do classically intractable feats with a computer just using noninteracting photons.

We now define the model more specifically following Aaronson and Arkhipov. The computer will be built out of linear optical elements. The particles will be photons (which are bosons). There will be n particles in m possible modes (where a mode can simply be thought of as a place that a photon can be). We assume that $n \leq m \leq poly(n)$. Photons are never created or destroyed, and any mode can have any nonnegative integer number of photons. The basis states of the computer can be written as $|S\rangle = |s_1, \ldots, s_m\rangle$ where s_i is the number of photons in mode i. Further, S must satisfy $S \in \Phi_{m,n}$ where $\Phi_{m,n}$ is the set of tuples $S = (s_1, \ldots, s_m)$ such that each $s_i \geq 0$ and $s_1 + \cdots + s_m = n$.

The boson computer works as follows. It starts in the computational basis state $|1_n\rangle \equiv |1, \ldots, 1, 0, \ldots, 0\rangle$ where there is exactly one photon in modes 1 through n, and modes $n+1$ through m have no photons. It then applies a unitary transformation by applying some sequence of phaseshifters and beamsplitters. Finally, a projective measurement in the computational basis is performed.

Let $\mathcal{U}$ be the unitary matrix corresponding to this sequence of phaseshifters and beamsplitters for the case of 1 photon. Note that in this case, there are m computational basis states and so

$\mathcal{U}$ is $m \times m$. Define the matrix A to be the $m \times n$ matrix obtained by keeping only the first n columns of $\mathcal{U}$. Now, let $S \in \Phi_{m,n}$, and define the matrix A_S as follows. If $S = (s_1, \ldots, s_m)$, take s_i copies of the i'th row of $\mathcal{U}$ for all $i \in [m]$. Hence, A_S is an $n \times n$ matrix. Now, if $\mathcal{D}_A$ is the probability distribution corresponding to the outputs of the boson computer upon measuring in the computational basis, we have

$$\Pr_{\mathcal{D}_A}[S] = \frac{|\mathrm{Per}(A_S)|^2}{s_1! \cdots s_m!} \tag{1}$$

The derivation of this result is beyond the scope of this overview, but is essentially a consequence of the statistics of identical bosons. See [2] for details.

The BosonSampling problem is to sample from $\mathcal{D}_A$, given A as input (note that A fully specifies the distribution of the boson computer). We now give an outline of the proof of how a BosonSampling oracle $\mathcal{O}$ would allow one to compute the square of an arbitrary $\mathbb{C}^n \times \mathbb{C}^n$ permanent in $\mathsf{BPP}^{\mathsf{NP}}$, but first we record a few ingredients which are necessary for the proof. By BosonSampling oracle, we mean an oracle which takes a string $r \in \{0,1\}^{poly(n)}$ and an $m \times n$ matrix A specifying the boson computer whose distribution over r chosen uniformly at random is equal to $\mathcal{D}_A$. Note that the oracle is a deterministic function of r and A – repeatedly querying the oracle with the same values of r and A will result in the same output.

Theorem 5.1 (Stockmeyer [10]). *Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, let*

$$p = \Pr_{x \in \{0,1\}^n}[f(x) = 1] = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x).$$

Then for all $g \geq 1 + \frac{1}{poly(n)}$, there exists an $\mathsf{FBPP}^{\mathsf{NP}^f}$ machine that approximates p to within a multiplicative factor of g.

Intuitively, this result says that a $\mathsf{BPP}^{\mathsf{NP}}$ machine can approximate the acceptance probability of a BPP machine to within a polynomially small factor, even if that acceptance probability is exponentially small.

Famously, Valiant [12] proved that computing the permanent is #P-complete:

Theorem 5.2 (Valiant [12]). *The following problem is #P-complete: given a matrix $X \in \{0,1\}^{n \times n}$, compute* $\mathrm{Per}(X)$.

Since the output distribution of the boson computer involves the square of permanent, a somewhat different result for the hardness of the permanent is needed for BosonSampling applications:

Theorem 5.3 (Aaronson-Arkhipov [2]). *The following problem is #P-hard, for any $g \in [1, poly(n)]$: given a matrix $X \in \mathbb{R}^{n \times n}$, approximate* $\mathrm{Per}(X)^2$ *to within a multiplicative factor of g.*

We also need the following lemma, also proved in the original BosonSampling paper:

Lemma 5.4 (Aaronson-Arkhipov [2]). *Let $X \in \mathbb{C}^{n \times n}$. Then for all $m \geq 2n$ and $\varepsilon \leq 1/\|X\|$, there exists an $m \times m$ unitary matrix U which can be computed in polynomial time that contains εX as a submatrix.*

Given all of the requisite ingredients, the desired result follows straightforwardly:

Theorem 5.5. *For any* BosonSampling *oracle* $\mathcal{O}$, $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{BPP}^{\mathsf{NP}^{\mathcal{O}}}$.

Proof. Given a matrix $X \in \mathbb{R}^{n \times n}$, we show how to approximate the squared permanent of X in $\mathsf{BPP}^{\mathsf{NP}^{\mathcal{O}}}$. By Theorem 5.3 this is a #P-hard task, and so the result follows. The strategy is to embed X into a unitary matrix corresponding to a boson computer, so that the squared permanent of X corresponds to the probability of the boson computer outputting $|1_n\rangle$. Then by Stockmeyer's result, if one has an oracle for BosonSampling, then one can approximate this probability in $\mathsf{BPP}^{\mathsf{NP}}$ and hence approximate the squared permanent of X.

More explicitly, let $m \equiv 2n$ and $\varepsilon \equiv 1/\|X\|$. Let U be a $m \times m$ unitary matrix whose $n \times n$ upper-left submatrix $U_{n,n}$ is equal to εX. Such a U exists by Lemma 5.4. Let A be the $m \times n$ submatrix of U corresponding to selecting the first n columns. Note that A can be interpreted as a description of a boson computer. Let p_A be the probability that the boson computer outputs $|1_n\rangle$. Now we have

$$p_A = \Pr_r[\mathcal{O}(A, r) = 1_n] = |\mathrm{Per}(U_{n,n})|^2 = \varepsilon^{2n}|\mathrm{Per}(X)|^2$$

The first equality is by definition of the BosonSampling oracle, the second follows from Equation 1, and the third follows from the embedding of X into U. But by Theorem 5.3, we can approximate p_A in $\mathsf{BPP}^{\mathsf{NP}^{\mathcal{O}}}$, and hence we can approximate $|\mathrm{Per}(X)|^2$ in the same. But by Theorem 5.3 approximating this quantity is #P-complete, and so the desired result follows. □

This result does *not* imply that a boson computer can solve a #P-complete problem. In particular, the permanent that one is trying to compute corresponds to a probability that is in general exponentially small. Hence, one would need an exponential number of samples to get a good estimate of it. This is why the deterministic nature of the BosonSampling oracle is crucial for the above proof.

It should be noted that a possible objection to the above proof is that the oracle is assumed to be perfect. That is, it is assumed that the distribution of outputs of the oracle is exactly the same as the distribution of outputs of the boson computer. However, in reality, a boson computer will have at least a small amount of noise, and so it can be argued that even a boson computer can't sample perfectly from $\mathcal{D}_A$! Unfortunately, the above proof completely breaks down when even a small amount of noise in the oracle is permitted, since the proof relies on the accurate estimation of exponentially small quantities. However, Aaronson and Arkhipov prove that assuming two highly plausible conjectures involving random matrices are true, then even an efficient approximate sampling algorithm would imply that a #P-complete problem could be solved relatively efficiently by a classical computer, and PH would collapse. But this proof is beyond the scope of the report.

6 Conclusion

One task for quantum computing researchers is to better understand the relationship between quantum complexity classes and classical complexity classes. In this report, I have surveyed some work which relates quantum computing to complete problems for the classical counting class #P. That such formal relationships exist is not too surprising, considering the fact that amplitudes of a quantum computation are sums of an exponentially large number of complex numbers. In fact, this consideration leads naturally to the proof that $\mathsf{BQP} \subseteq \mathsf{P}^{\#\mathsf{P}}$. What seems a bit more surprising is the fact that #P-completeness appears in many varied quantum contexts. We have seen that it appears when one asks about whether a quantum circuit accepts with nonzero probability, when

one considers quantum computation with the extra resource of postselection, and even when one considers the weak, non-universal, and extremely simple model of BosonSampling. Given this, it seems likely that further connections between #P-complete problems and other aspects of quantum computing will appear in the future, and an open task is to discover such connections. For example, are there other non-universal models of quantum computation like BosonSampling in which one can embed a #P-complete problem as the permanent problem was embedded into BosonSampling? On the other hand, the hardness for BosonSampling is based on the fact that the embedded matrix could have both positive and negative entries – it is known due to an algorithm of Jerrum, Sinclair, and Vigoda [9] that there exists a fully polynomial randomized approximation scheme (FPRAS) for approximating the permanent of a matrix with nonnegative real entries. Is there some model of quantum computation that *can* be efficiently simulated classically due to a connection to the permanent of a nonnegative matrix? For example, some classes of Hamiltonians (so-called "stoquastic" Hamiltonians) correspond to classical stochastic transition matrices. So, perhaps some model of computation based on stoquastic Hamiltonians could be classically simulable due to the efficient approximation of the permanent of nonnegative matrices.

References

[1] S. Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proc. Roy. Soc. London*, A461(2063):3473–3482, 2005. quant-ph/0412187.

[2] S. Aaronson and A. Arkhipov. The computational complexity of linear optics. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 333–342, 2011.

[3] C. Bennett, E. Bernstein, G. Brassard, U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Computing*, 26:1510-1523, 1997. quant-ph/9701001

[4] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. First appeared in ACM STOC 1993.

[5] M. Bremner, R. Jozsa, and D. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proc. Roy. Soc. London*, 2010. arXiv:1005.1407.

[6] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[7] S. Fenner, F. Green, S. Homer, and R. Pruim. Determining acceptance possibility for a quantum computation is hard for the polynomial hierarchy. *Proc. Roy. Soc. London*, A455:3953–3966, 1999. quant-ph/9812056.

[8] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *J. Comput. System Sci.*, 59 (1999), no. 2, pp. 240?252.

[9] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. *J. ACM*, 51(4):671–697, 2004. Earlier version in STOC?2001.

[10] L. J. Stockmeyer. The complexity of approximate counting. In *Proc. ACM STOC*, pages 118–126, 1983.

[11] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[12] L. G. Valiant. The complexity of computing the permanent. *Theoretical Comput. Sci.*, 8(2):189–201, 1979.

18.405J / 6.841J Advanced Complexity Theory

Spring 2016