The Power of Quantum Fourier Sampling

Thesis by

William Jason Fefferman

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2014

(Defended May 23, 2014)

Dedicated to my Mother, without whom none of this would be possible.

Acknowledgements

I am deeply grateful to all of those who provided support and assistance during the years I attended Caltech. I am particularly indebted to Chris Umans, Alexei Kitaev and John Preskill whose constant advice over my time as a graduate student greatly impacted the shape of this thesis. I am also grateful to my thesis committee consisting of Venkat Chandrasekaran, Alexei Kitaev, John Preskill and Chris Umans.

Additionally I need to thank all of my past and present colleagues at the Institute for Quantum Information including (but not at all limited to): Gorjan Alagic, Salman Beigi, Sergio Boixo, Steve Flammia, Stephen Jordan, Robert König, Yi-Kai Liu, Sprios Michalakis, Fernando Pastawski, and Norbert Schuch. I also want to thank my many collaborators outside of Caltech with whom I had many inspirational conversations, including: Scott Aaronson, Fernando Brandão, Harry Buhrman, Aram Harrow, Umesh Vazirani, Ronald de Wolf.

Of course I am most grateful to my friends and family for their endless support over the last years.

Bill Fefferman
May 2014
Pasadena

Abstract

How powerful are Quantum Computers? Despite the prevailing belief that Quantum Computers are more powerful than their classical counterparts, this remains a conjecture backed by little formal evidence. Shor's famous factoring algorithm [Sho94] gives an example of a problem that can be solved efficiently on a quantum computer with no known efficient classical algorithm. Factoring, however, is unlikely to be $\mathbf{NP}$-Hard, meaning that few unexpected formal consequences would arise, should such a classical algorithm be discovered. Could it then be the case that any quantum algorithm can be simulated efficiently classically? Likewise, could it be the case that Quantum Computers can quickly solve problems much harder than factoring? If so, where does this power come from, and what classical computational resources do we need to solve the hardest problems for which there exist efficient quantum algorithms?

We make progress toward understanding these questions through studying the relationship between classical nondeterminism and quantum computing. In particular, is there a problem that can be solved efficiently on a Quantum Computer that cannot be efficiently solved using nondeterminism? In this thesis we address this problem from the perspective of sampling problems. Namely, we give evidence that approximately sampling the Quantum Fourier Transform of an efficiently computable function, while easy quantumly, is hard for any classical machine in the Polynomial Time Hierarchy. In particular, we prove the existence of a class of distributions that can be sampled efficiently by a Quantum Computer, that likely cannot be approximately sampled in randomized polynomial time with an oracle for the Polynomial Time Hierarchy.

Our work complements and generalizes the evidence given in Aaronson and Arkhipov's work

[AA13] where a different distribution with the same computational properties was given. Our result is more general than theirs, but requires a more powerful quantum sampler.

Contents

Chapter 1

Introduction

1.1 Background

Nearly twenty years after the discovery of Shor's factoring algorithm [Sho94] that caused an explosion of interest in quantum computation, the complexity theoretic classification of quantum computation remains embarrassingly unsettled.

The foundational results of Bernstein and Vazirani [BV97], and Bennett, Bernstein, Brassard and Vazarani [BBBV97] laid the groundwork for quantum complexity theory by defining $\mathbf{BQP}$ as the class of problems solvable with a quantum computer in polynomial time, and established the upper bound, $\mathbf{BQP} \subseteq \mathbf{P}^{\#\mathbf{P}}$, which hasn't been improved since.

In particular, given that $\mathbf{BPP} \subseteq \mathbf{BQP}$, so quantum computers are surely no less powerful than their classical counterparts, it is natural to compare the power of efficient quantum computation to the power of efficient classical verification. Can every problem with an efficient quantum algorithm be verified efficiently? Likewise can every problem whose solution can be verified efficiently be solved quantumly? In complexity theoretic terms, is $\mathbf{BQP} \subseteq \mathbf{NP}$, and is $\mathbf{NP} \subseteq \mathbf{BQP}$? Factoring is contained in $\mathbf{NP} \cap \mathbf{coNP}$, and so cannot be $\mathbf{NP}$-hard unless $\mathbf{NP} = \mathbf{coNP}$ and the $\mathbf{PH}$ collapses. Thus, while being a problem of profound practical importance, Shor's algorithm does not give evidence that $\mathbf{NP} \subseteq \mathbf{BQP}$.

Even progress towards oracle separations has been agonizingly slow. These same works that defined $\mathbf{BQP}$ established an oracle for which $\mathbf{NP} \not\subset \mathbf{BQP}$ [BBBV97] and $\mathbf{BQP} \not\subset \mathbf{NP}$ [BV97]. This last result can be improved to show an oracle relative to which $\mathbf{BQP} \not\subset \mathbf{MA}$ [BV97], but even finding an oracle relative to which $\mathbf{BQP} \not\subset \mathbf{AM}$ is still wide open. This is particularly troubling given that, under widely believed complexity assumptions, $\mathbf{NP} = \mathbf{MA} = \mathbf{AM}$ [KvM02]. Thus, our failure to provide an oracle relative to which $\mathbf{BQP} \not\subset \mathbf{AM}$ indicates a massive lack of understanding of the classical power of quantum computation.

Recently, two candidate oracle problems with quantum algorithms have been proven to not be contained in the $\mathbf{PH}$, assuming plausible complexity theoretic conjectures [Aar10a, FU11].[1] These advances remain at the forefront of progress on these questions.

A line of work initiated by Bremner, Jozsa and Shepherd [BJS10], and Aaronson and Arkhipov [AA13] asks whether we can provide a theoretical basis for quantum superiority by looking at *distribution sampling problems*. In particular, Aaronson and Arkhipov show a *distribution* that can be sampled efficiently by a particular limited form of quantum computation, that assuming the validity of two feasible conjectures, cannot be approximately sampled classically (even by a randomized algorithm with a $\mathbf{PH}$ oracle), unless the $\mathbf{PH}$ collapses. The equivalent result for decision problems, establishing $\mathbf{BQP} \not\subset \mathbf{BPP}$ unless the $\mathbf{PH}$ collapses, would be a crowning achievement in quantum complexity theory. In addition, this research has been very popular not only with the theoretical community, but also with experimentalists who hope to perform this task, "Boson Sampling", in their labs.

Interestingly, it is also known that if we can find such a quantumly sampleable distribution for which no classical approximate sampler exists, there exists a "search" problem that can be solved by a quantum computer that cannot be solved classically [Aar10c]. In a search problem we are given an input $x \in \{0,1\}^n$, and our goal is to output an element in a nonempty set, $A_x \subseteq \{0,1\}^{poly(n)}$ with high probability. This would be one of the strongest pieces of evidence to date that quantum computers can outperform their classical counterparts.

[1] Although the "Generalized Linial-Nisan" conjecture proposed in [Aar10a] is now known to be false [Aar10b].

In this work we use the same general algorithmic framework used in Shor's algorithm, which we refer to as "Quantum Fourier Sampling", to demonstrate the existence of a general class of distributions that can be sampled exactly by a quantum computer. We then argue that these distributions shouldn't be able to be approximately sampled classically, unless the $\mathbf{PH}$ collapses. Perhaps surprisingly, we obtain and generalize many of the same conclusions as Aaronson and Arkhipov [AA13] with a completely different class of distributions.

1.2 Overview

We begin the thesis in Chapter 2 with a discussion of upper bounds for $\mathbf{BQP}$. In Section 2.3 we review the proof that $\mathbf{BQP} \subseteq \mathbf{P}^{\#\mathbf{P}}$, a result that hasn't been significantly improved for nearly two decades. Then, motivated by the relation between $\mathbf{BQP}$ and $\mathbf{PH}$, we give a nontrivial class of quantum circuits that can be simulated classically with an $\mathbf{NP}$ oracle. In particular, in Section 2.4 we prove that if a quantum circuit is composed of small, fixed angle rotation gates and Toffoli gates, we can classically compute the success probability using an $\mathbf{NP}$ oracle. The running time is c^r where r is the number of rotation gates in the circuit and the base of the exponent, c, gets closer to 1 as the angle of the rotation gate gets closer to 0. Thus, these circuits can be simulated with faster and faster classical time complexity.

In Chapter 3, we discuss the complexity of counting the number of satisfying assignments to a Boolean formula and review Valiant's result that computing the Permanent of a matrix with binary entries is $\#\mathbf{P}$-complete [Val79]. We then focus on demonstrating several ways in which this hardness result is robust. In Section 3.2 we show that even outputting a multiplicative estimate to the Permanent of a matrix with integer entries is $\#\mathbf{P}$-hard. We show in Section 3.3 that computing the Permanent of matrices with entries from a sufficiently large finite field on average is $\#\mathbf{P}$-hard. We then extend this result to show a class of distributions over $\mathbb{R}$ called "autocorrelatable", from which computing the Permanent on average is $\#\mathbf{P}$-hard.

In Chapter 4 we give a simple example of a distribution that can be sampled exactly on a quantum computer that cannot be sampled exactly classically unless the $\mathbf{PH}$ collapses. This chapter uses the hardness results proven in Section 3.2.

We then discuss the power of approximate quantum sampling, which is our main topic of interest. In Section 5.2 we define a general class of distributions that can be sampled exactly on a quantum computer. The probabilities in these distributions are proportional to each different $\{\pm 1\}^n$ evaluation of a particular *Efficiently Specifiable* polynomial (see Definition 40) with n variables. We then show in Section 5.3 that the existence of an approximate classical sampler for these distributions implies the existence of an *additive approximate average-case* solution to the Efficiently Specifiable polynomial. We generalize this in Section 5.4 to prove that quantum computers can sample from a class of distributions in which each probability is proportional to polynomially bounded integer evaluations of an Efficiently Specifiable polynomial.

In Section 6 we give two examples of Efficiently Specifiable polynomials. We prove in Section 6.1 that the Permanent polynomial is Efficiently Specifiable and in Section 6.2 that the Hamiltonian Cycle polynomial is Efficiently Specifiable.

We then attempt to extend this result to quantumly sample from a distribution with probabilities proportional to exponentially bounded integer evaluations of Efficiently Specifiable polynomials. To do this, in Section 7.1, we introduce a variant of the Quantum Fourier Transform which we call the "Squashed QFT". We explicitly construct this unitary operator, and show how to use it in our quantum sampling framework. We leave as an open question whether this unitary can be realized by an efficient quantum circuit. We then prove in Section 7.4, using a similar argument to Section 5.3, that if we had a classical approximate sampler for this distribution we'd have an *additive approximate average-case* solution to the Efficiently Specifiable polynomial with respect to the binomial distribution over exponentially bounded integers.

In Section 8 we conclude with conjectures needed to establish the intractability of approximate classical sampling from any of our quantumly sampleable distributions. As shown in Sections 5.3 and 5.4 it suffices to prove that an *additive approximate average-case solution* to any Efficiently Specifiable polynomial is $\#\mathbf{P}$-hard, and we conjecture that this is possible. We also propose an

"Anti-concentration conjecture" relative to an Efficiently Specifiable polynomial over the binomial distribution, which allows us to reduce the hardness of an *additive approximate average-case* solution to a *multiplicative approximate average-case solution.* Assuming this second conjecture, we can then base our first conjecture around the hardness of *multiplicative*, rather than *additive approximate average-case solutions* to an Efficiently Specifiable polynomial.

These two conjectures generalize conjectures in Aaronson and Arkhipov's results [AA13]. They conjecture that an *additive approximate average-case solution* to the Permanent with respect to the Gaussian distribution with mean 0 and variance 1 is $\#\mathbf{P}$-hard. They further propose an "Anti-concentration" conjecture which allows them to reduce the hardness of *additive approximate average-case solutions* to the Permanent over the Gaussian distribution to the hardness of *multiplicative average case solutions* to the Permanent over the Gaussian distribution. The parameters of our conjectures match the parameters of theirs, but our conjecture is broader, applying to any Efficiently Specifiable polynomial, a class which includes the Permanent, and a wider class of distributions, and thus is formally easier to prove.

Chapter 2

Preliminaries and Basic Definitions

2.1 Computational Complexity Basics

In this section we briefly review some basic topics from Computational Complexity Theory. We assume the familiarity with basic models of universal computation such as Turing Machines, see e.g., [AB09].

Recall a "Decision Problem" is a subset of binary strings, denoted $\mathcal{L} \subseteq \{0,1\}^*$. We say a Decision Problem $\mathcal{L} \in \mathbf{P}$ if membership in $\mathcal{L}$ can be decided by a Deterministic Turing machine in time polynomial in the length of the input. Likewise, we define the class $\mathbf{NP}$ to be the set of Decision Problems $\mathcal{L}$ whose membership can be verified in $\mathbf{P}$, or more formally:

Definition 1 (Nondeterministic Polynomial Time)**.** *We say a Decision Problem $\mathcal{L} \in \mathbf{NP}$ if there exists a polynomial $p(n)$ and a polynomial time Deterministic Turing Machine V, so that for all $x \in \{0,1\}^*$:*

$$x \in \mathcal{L} \iff \exists y \in \{0,1\}^{p(|x|)}\, V(x,y) = 1$$

Next, we define a few natural Decision Problems that are of particular importance in complexity theory.

Definition 2 (Satisfiability)**.** SAT *is the Decision Problem consisting of binary encodings of sat-*

isfiable Boolean formulas.

It is known that $\mathbf{SAT}$ is $\mathbf{NP}$-complete, by which we mean that $\mathbf{SAT} \in \mathbf{NP}$ and $\mathbf{SAT}$ is $\mathbf{NP}$-hard, meaning we can efficiently decide any other Decision Problem in $\mathbf{NP}$ using the ability to solve $\mathbf{SAT}$. This was first established in the classic work of Cook and Levin (see e.g., [AB09]).

In this thesis we are primarily concerned with the Polynomial Time Hierarchy, or $\mathbf{PH}$, a class that generalizes $\mathbf{NP}$. We first define the class $\Sigma_1^{\mathbf{P}} = \mathbf{NP}$. Then define $\Sigma_k^{\mathbf{P}}$ recursively, so that, for $k > 0$, $\Sigma_{k+1}^{\mathbf{P}} = \mathbf{NP}^{\Sigma_k^{\mathbf{P}}}$, where this notation refers to the class of Decision Problems that can be decided in $\mathbf{NP}$ with the ability to query an oracle that decides any problem in $\Sigma_k^{\mathbf{P}}$. Then:

Definition 3 (PH)**.**

$$\mathbf{PH} = \bigcup_{\mathbf{k}>\mathbf{0}} \mathbf{\Sigma_k^P}$$

Interestingly this class can also be characterized in terms of a variant of Satisfiability. A natural complete problem for each level of the $\mathbf{PH}$, $\Sigma_k^{\mathbf{P}}$, is $\mathbf{QSAT_k}$, or quantified $\mathbf{SAT}$ with k alternations [AB09]:

Definition 4 (Quantified Satisfiability)**.** $\mathbf{QSAT_k}$ *is the language consisting of all formulas ψ, with variables partitioned into k subsets $S_1, S_2, ..., S_k$ so that:*

$\psi \in \mathbf{QSAT_k} \Leftrightarrow$

$$\exists S_1 \forall S_2 ... Q_k S_k \; \psi(x_{S_1}, x_{S_2}, ..., x_{S_k}) = 1$$

Where $\exists S_i$ is notation meaning "there exists an assignment to the variables in S_1", $\forall S_j$ is the notation meaning "for all assignments to the variables in S_j", and Q_k is the k-th quantifier.

2.2 Quantum Preliminaries

In this next section we cover the basic priciples of quantum computing needed to understand the content in the thesis. For a much more complete overview there are many references available, e.g., [KSV02, NC00].

The state of an n-*qudit quantum system* is described by a unit vector in $\mathcal{H} = (\mathbb{C}^d)^{\otimes n}$, a d^n-dimensional complex Hilbert space, endowed with the standard Hilbert-Schmidt inner product. When d=2, we say the system is composed of n qubits. As per the literature we will denote the standard orthogonal basis vectors of $\mathcal{H}$ by $\{|v\rangle\}$ for $v \in [d]^n$.

In accordance with the laws of quantum mechanics, transformations of states are described by unitary transformations acting on $\mathcal{H}$, where a *unitary transformation* over $\mathcal{H}$ is a linear transformation specified by a $d^n \times d^n$ square complex matrix U, such that $UU^* = I$, where U^* is the conjugate transpose. Equivalently, the rows (and columns) of U form an orthonormal basis. A *local* unitary is a unitary that operates only on $b = O(1)$ qudits; i.e. after a suitable renaming of the standard basis by reordering qudits, it is the matrix $U \otimes I_{d^{n-b}}$, where U is a $d^b \times d^b$ unitary U. A local unitary can be applied in a single step of a Quantum Computer. A *local decomposition* of a unitary is a factorization into local unitaries. We say a $d^n \times d^n$ unitary is *efficiently quantumly computable* if this factorization has at most $poly(\log(d^n))$ factors.

We will also need the concept of *projective measurement*, which given an orthonormal basis O for $\mathcal{H}$ associates a value designated by a real number r_i for each basis vector $|v_i\rangle \in O$. Suppose our quantum system is in the state $|\phi\rangle \in \mathcal{H}$. We define $\{\Pi_{r_j}\}$ to be a collection of projection operators that project into the subspace spanned by the designated $|v_j\rangle$ for all v_j associated to the same output value r_j. When we measure our system, we obtain the respective outcome r_j with probability $|\Pi_{r_j}|\phi\rangle|^2$ and the resulting state of the system becomes $\frac{\Pi_{r_j}|\phi\rangle}{|\Pi_{r_j}|\phi\rangle|}$.

As an example, suppose our Hilbert space $\mathcal{H}$ can be decomposed into orthogonal subspaces $\mathcal{H} = S_1 \oplus S_2$. When we measure $\{\Pi_1, \Pi_2\}$ which project into the orthogonal subspaces S_1 and S_2, it causes the system to collapse to $\Pi_1|\phi\rangle/|\Pi_1|\phi\rangle|$ or $\Pi_2|\phi\rangle/|\Pi_2|\phi\rangle|$ with probability $|\Pi_1|\phi\rangle|^2$ and $|\Pi_2|\phi\rangle|^2$ respectively.

An efficient *quantum circuit* consists of at most $poly(n)$ local unitaries, followed by a measurement.

There are universal finite gate sets for which any efficiently quantumly computable unitary can be realized (up to exponentially small error) by a $poly(n)$-size quantum circuit [KSV02]. In this thesis, we will use the Hadamard and Toffoli gate set. The Hadamard is a one qubit gate:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

And the Toffoli is the three qubit gate implementing a Controlled-Controlled-Not, which simply flips the state of the last qubit iff the first two qubits are 1. Together these are known to be a universal gate set [Shi03].

2.3 Quantum Complexity and $\mathbf{BQP}$

Definition 5 (Uniform family of quantum circuits)**.** *A uniform family of quantum circuits is a set of efficient quantum circuits $\{Q_x\}$, so that there exists a polynomial time Deterministic Turing Machine that, on input x outputs a classical description of the circuit Q_x.*

Definition 6 (BQP). *A Decision Problem $\mathcal{L} \in \mathbf{BQP}$ iff there exists a uniform family of quantum circuits $\{Q_x\}$ so that for all $x \in \{0,1\}^*$:*

$$x \in \mathcal{L} \Rightarrow \Pr\left[Q_x|\mathbf{0}\rangle = \mathbf{1}\right] \geq 2/3$$

And

$$x \notin \mathcal{L} \Rightarrow \Pr\left[Q_x|\mathbf{0}\rangle = \mathbf{1}\right] \leq 1/3$$

Where implicitly in this definition, the quantum circuit makes a projective measurement on a designated qubit and accepts (outputs 1) iff it obtains a desired measurement outcome.

Theorem 7 (Bernstein & Vazirani [BV97])**.** $\mathbf{BQP} \subseteq \mathbf{P}^{\#\mathbf{P}}$

Proof. We first prove a lemma that shows the acceptance probability of any uniform family of quantum circuits can be expressed as the difference of two $\#\mathbf{P}$ functions. In the proceeding discussion, we utilize a standard fact of quantum computation (see e.g., [NC00, BBBV97]), which is we can assume without loss of generality that our quantum circuit has only a single accepting basis state, which we denoted here by $|\mathbf{0}\rangle$. This means that we can obtain the acceptance probability of the quantum algorithm by looking at a single entry of the unitary matrix realized by the circuit, $\langle\mathbf{0}|Q|\mathbf{0}\rangle$ (the idea of the proof is to use a Controlled-Not gate to "copy" the value of the output qubit to an ancillary register, and uncompute all work qubits, which we assume were initialized to $|\mathbf{0}\rangle$).

Lemma 8 (Adaptation of Fortnow & Rogers [FR99][DHM+05])**.** *Suppose $\mathcal{L} \in \mathbf{BQP}$. Then $\mathcal{L}$ can be decided by a uniform family of quantum circuits $\{C_x\}$. Without loss of generality, we can assume each C_x is composed of at most polynomial number of Toffoli and Hadamard gates, since this is a universal gate set. We let the number of Hadamard gates in the circuit C_x be $h(n)$*[1]*. Then there exists $f, g \in \#\mathbf{P}$ so that for every $x \in \{0,1\}^n$:*

$$\langle\mathbf{0}|C_x|\mathbf{0}\rangle = \frac{f(x) - g(x)}{2^{h(n)/2}}$$

Proof. Fix an $x \in \{0,1\}^n$. Suppose $C_x = U_m U_{m-1} ... U_1$ with $m \in poly(n)$ where each U_i is either a Hadamard gate acting on one qubit or a Toffoli gate which acts on three qubits. Clearly,

$$\langle\mathbf{0}|C_x|\mathbf{0}\rangle = \sum_{y_2, y_3, \ldots, y_m \in \{0,1\}^n} \langle\mathbf{0}|U_m|y_m\rangle\langle y_m|U_{m-1}|y_{m-1}\rangle ... \langle y_2|U_1|\mathbf{0}\rangle \tag{2.1}$$

Now consider the value of some term in the product $a = \langle y_i|U_{i-1}|y_{i-1}\rangle$.

- Suppose U_{i-1} is a Toffoli gate acting on qubits k_1, k_2, k_3, then $a = 1$ if $y_i(k_1) = y_{i-1}(k_1)$, $y_i(k_2) = y_{i-1}(k_2)$, $y_i = y_{i-1}(k_3) \oplus y_{i-1}(k_1)y_{i-1}(k_2)$, and $y_i(k) = y_{i-1}(k)$, for all $k \neq k_1, k_2, k_3$, and $a = 0$ otherwise.

[1]Note that this should technically be a function depending on x. We will use $h(n)$ here because for all practical purposes the number of Hadamard gates in circuit C_x should be independent of the input, x.

- Suppose U_{i-1} is a Hadamard gate acting on qubit k then $a = \frac{-1}{\sqrt{2}}$ if $y_i(k) = y_{i-1}(k) = 1$ and else $\frac{1}{\sqrt{2}}$, if the bits outside k agree (i.e., if $y_i(j) = y_{i-1}(j)$ for all $j \neq k$), or $a = 0$ if the bits outside k don't agree.

We refer to any term in the sum of Equation 2.1, corresponding to a setting of $y_2, ..., y_m \in \{0, 1\}^n$, as a path of the quantum circuit. We define the value of that path to be its contribution to the sum, and an admissible path as one whose value is non-zero. Note that the absolute value of the value of each nonzero path is $\frac{1}{\sqrt{2^{h(n)}}}$, and there are $2^{h(n)}$ different admissible paths in our circuit. Let A be the set of admissible paths.

Additionally, we can breakup the set of admissible paths A, into the set of positive paths A_+, in which the sign of the value of each admissible path $y \in A_+$ is positive, and A_-, in which the sign of the value of each admissible path $y \in A_-$ is negative.

The theorem follows, letting $f(x)$ be the number of admissible paths y so that $y \in A_+$ and $g(x)$ the number of admissible paths y so that $y \in A_-$. Since, given any path y, we can determine efficiently if it belongs to A_+ or A_-, this tells us that both $f, g \in \#\mathbf{P}$.

□

Note that this immediately implies $\mathbf{BQP} \subseteq \mathbf{PSPACE}$, because we can simply compute the value of each path in the sum of 2.1 using only a $poly(n)$ amount of space. Lemma 8 proves that $\mathbf{BQP} \subseteq \mathbf{P}^{\mathbf{gapP}}$, where $\mathbf{gapP}$ is the difference of two $\#\mathbf{P}$ functions. We can appeal to the bounds (and the characterization of $\mathbf{P}^{\#\mathbf{P}}$) proven in [FFK91] to show that this suffices to prove $\mathbf{BQP} \subseteq \mathbf{P}^{\#\mathbf{P}}$.

□

We will need the concept of quantum evaluation of an efficiently classically computable function

$f : \{0,1\}^n \to \{0,1\}^m$, which in one quantum query to f maps:

$$\sum_{x\in\{0,1\}^n} |x\rangle|z\rangle \to \sum_{x\in\{0,1\}^n} |x\rangle|z \oplus f(x)\rangle$$

Note that this is a unitary map, as applying it again inverts the procedure, and can be done efficiently as long as f is efficiently computable.

Assuming f is $\{0,1\}$-valued, we can use this state together with a simple phase flip unitary gate to prepare:

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

And one more quantum query to f, which "uncomputes" it, allows us to obtain the state $\sum\limits_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle$. Equivalently, if the efficiently computable function is $f : \{0,1\} \to \{\pm 1\}$ we can think of this as a procedure to prepare:

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

With two quantum queries to the function f.

2.4 Better Classical Algorithms for Simulating $\mathbf{BQP}$ using Approximate Counting?

Note that Theorem 30 tells us that we can approximately count any $\#\mathbf{P}$ function on n variables to within multiplicative error ϵ in time $poly(n, 1/\epsilon)$ with an $\mathbf{NP}$ oracle. We have shown in Lemma 8 that for any quantum circuit C_x, the acceptance probability p_x can be expressed as the difference of two $\#\mathbf{P}$ functions f and g, over a common denominator. A naive strategy towards deciding any language in $\mathbf{BQP}$ is to approximate count f, approximate count g, and subtract the two estimates.

We need to ascertain how small we need to set our error tolerance ϵ to approximate count f and g to determine if $p_x \geq 2/3$ or $p_x \leq 1/3$.

We will show that this tolerance depends heavily on the choice of gate set. In particular for $k > 2$ and $\theta = \frac{\pi}{2^k}$ we define R_θ to be the one qubit gate:

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Note that the gate set $\{R_\theta, T\}$ is a universal gate set for quantum computation since we can always "build" Hadamard gates out of k R_θ gates.

Theorem 9. *Let $\mathcal{L} \in \mathbf{BQP}$, and for any fixed $k > 2$, let $\theta = \pi/2^k$ and let $\{C_x\}$ be a uniform family of quantum circuits deciding $\mathcal{L}$, composed of $\{R_\theta, T\}$ gates. Let $|x| = n$ and let $r(n)$ be the number of R_θ gates in circuit C_x. We can decide whether $p_x \geq 2/3$ or $p_x \leq 1/3$ in poly $\left(n, (\cos\theta + \sin\theta)^{r(n)}\right)$ time with an $\mathbf{NP}$ oracle.*

Proof. As before, any language $\mathcal{L} \in \mathbf{BQP}$ can be decided by a uniform family of circuits $\{C_x\}$, with each $C_x = U_m U_{m-1} ... U_1$, where U_j is either R_θ or T, for some fixed polynomial $m \in poly(n)$. The acceptance probability on input $x \in \{0,1\}^n$ is $p_x = |\langle \mathbf{0}|C_x|\mathbf{0}\rangle|^2$. We can express this probability as the square of the sum of efficiently computable terms and as before, we can write:

$$p_x = \left(\sum_{y_2,\ldots,y_m \in \{0,1\}^n} \langle \mathbf{0}|U_m|y_m\rangle \langle y_m|U_{m-1}|y_{m-1}\rangle ... \langle y_2|U_1|\mathbf{0}\rangle \right)^2 = \left(\sum_{y_2 \cdots y_m} v\left(y_2, ..., y_m\right) \right)^2 \quad (2.2)$$

Define $f_x(y_2, ..., y_m) = v(y_2, ..., y_m)$ iff $v(y_2, ..., y_m) > 0$, and $g_x(y_2, ..., y_m) = -v(y_2, ..., y_m)$ iff $v(y_2, ..., y_m) < 0$.

Note that:

$$\sum_{y_2,\ldots,y_m} v(y_2, ..., y_m) = \sum_{y_2,\ldots,y_m} f_x(y_2, ..., y_m) - \sum_{y_2,\ldots,y_m} g_x(y_2, ..., y_m)$$

Let D be an integer representing the "precision", whose value will be set later. Since the value of each path is efficiently computable, we can define two binary valued circuits F_x and G_x in the following natural way:

$$F_x(y_2, ..., y_m, z \in [D]) = 1 \iff z \le f_x(y_2, ..., y_m)D$$

and

$$G_x(y_2, ..., y_m, z \in [D]) = 1 \iff z \le g_x(y_2, ..., y_m)D$$

Now note that, for all $y_2, ..., y_m \in \{0, 1\}^n$:

$$f_x(y_2, ..., y_m)D - 1 \le \sum_{z \in [D]} F_x(y_2, ..., y_m, z) \le f_x(y_2, ..., y_m)D \tag{2.3}$$

and

$$g_x(y_2, ..., y_m)D - 1 \le \sum_{z \in [D]} G_x(y_2, ..., y_m, z) \le g_x(y_2, ..., y_m)D \tag{2.4}$$

Note also:

$$\sum_{y_2, ..., y_m} f_x(y_2, ..., y_m)D - 2^{r(n)} \le \sum_{y_2, ..., y_m, z} F_x(y_2, ..., y_m, z) \le \sum_{y_2, ..., y_m} f_x(y_2, ..., y_m)D \tag{2.5}$$

and

$$\sum_{y_2, ..., y_m} g_x(y_2, ..., y_m)D - 2^{r(n)} \le \sum_{y_2, ..., y_m, z} G_x(y_2, ..., y_m, z) \le \sum_{y_2, ..., y_m} g_x(y_2, ..., y_m)D \tag{2.6}$$

Lines 2.5 and 2.6 follow from lines 2.3 and 2.4, since there are exactly $2^{r(n)}$ admissible paths in the sum of line 2.2.

We need to determine the tolerance, ϵ, required to decide if $p_x \ge 2/3$ or $p_x \le 1/3$.

In the discussion that follows, we'll use F and G as shorthand for $|F_x^{-1}(1)|, |G_x^{-1}(1)|$ and we'll use f and g as shorthand for $\sum\limits_{y_2,\dots,y_m} f_x(y_2,...,y_m)$ and $\sum\limits_{y_2,\dots,y_m} g_x(y_2,...,y_m)$.

Formally, our goal is to find an α_1, α_2 with the property:

$$(1-\epsilon)F \leq \alpha_1 \leq (1+\epsilon)F$$

and

$$(1-\epsilon)G \leq \alpha_2 \leq (1+\epsilon)G$$

So that the quantity $\frac{\alpha_1-\alpha_2}{D}$ allows us to distinguish between $p_x \geq 2/3$ and $p_x \leq 1/3$.

Note that:

$$\begin{aligned}
(1-\epsilon)\frac{F}{D} - (1+\epsilon)\frac{G}{D} \quad &\leq \quad \frac{\alpha_1-\alpha_2}{D} \leq (1+\epsilon)\frac{F}{D} - (1-\epsilon)\frac{G}{D} \\
&\Rightarrow \quad \frac{F-G}{D} - \frac{\epsilon(F+G)}{D} \leq \frac{\alpha_1-\alpha_2}{D} \leq \frac{F-G}{D} + \frac{\epsilon\,(F+G)}{D} \\
&\Rightarrow \quad f-g-\frac{2^{r(n)}}{D} - \epsilon(f+g) \leq \frac{\alpha_1-\alpha_2}{D} \leq f-g+\frac{2^{r(n)}}{D} + \epsilon(f+g)
\end{aligned}$$

Where the last implication follows from lines 2.5 and 2.6.

So, we set D so that $\frac{2^{r(n)}}{D} \leq \epsilon(f+g)$ and it follows that:

$$f-g-2\epsilon(f+g) \leq \frac{\alpha_1-\alpha_2}{D} \leq f-g+2\epsilon(f+g)$$

Now setting $\epsilon = \frac{1}{18(f+g)}$ with the inequalities from above, and recalling that $\langle \mathbf{0}|C_x|\mathbf{0}\rangle = f-g$ we have:

$$\begin{aligned}\langle \mathbf{0}|C_x|\mathbf{0}\rangle - 2\epsilon(f+g) &\leq \frac{\alpha_1-\alpha_2}{D} \leq \langle \mathbf{0}|C_x|\mathbf{0}\rangle + 2\epsilon(f+g)\\ &\Rightarrow \langle \mathbf{0}|C_x|\mathbf{0}\rangle - \frac{1}{9} \leq \frac{\alpha_1-\alpha_2}{D} \leq \langle \mathbf{0}|C_x|\mathbf{0}\rangle + \frac{1}{9}\end{aligned}$$

Then,

1. If $p_x \geq 2/3$, then $\langle \mathbf{0}|C_x|\mathbf{0}\rangle \geq \sqrt{2/3}$ or $\langle \mathbf{0}|C_x|\mathbf{0}\rangle \leq -\sqrt{2/3}$ and so:

   $0.7 \leq \frac{\alpha_1-\alpha_2}{D} \leq 0.93$, or $-0.93 \leq \frac{\alpha_1-\alpha_2}{D} \leq -0.7$

2. If $p_x \leq 1/3$, then $0 \leq \langle \mathbf{0}|C_x|\mathbf{0}\rangle \leq \sqrt{1/3}$ or $-\sqrt{1/3} \leq \langle \mathbf{0}|C_x|\mathbf{0}\rangle \leq 0$ and so:

   $0.47 \leq \frac{\alpha_1-\alpha_2}{D} \leq 0.67$, or $-0.67 \leq \frac{\alpha_1-\alpha_2}{D} \leq -0.47$

These cases are distinguishable.

Now, by definition, $f+g = \sum_{y_2,\cdots y_m} |v(y_2,...,y_m)|$. We note that $\sum_{y_2,\cdots y_m} |v(y_2,...,y_m)| = (\cos\theta + \sin\theta)^{r(n)}$ since we can think of this sum as a binomial expansion in $\cos\theta$ and $\sin\theta$.

Now our theorem is proven, setting $\epsilon = \frac{1}{9(f+g)} = \frac{1}{18(\cos\theta+\sin\theta)^{r(n)}}$. Then, for fixed $\theta > 0$ we can simulate a generic quantum circuit with $poly(n)$ Toffoli gates and $r(n)$ R_θ gate gates, in time $poly(n, 1/\epsilon) = poly\left(n, 18\left(\cos\theta + \sin\theta\right)^{r(n)}\right)$ using Theorem 30.

□

Chapter 3

The Complexity of Counting and the Permanent function

3.1 Basic Counting Definitions and Results

In this section we consider the complexity of counting the number of solutions to $\mathbf{NP}$-complete problems.

Definition 10 ($\#\mathbf{P}$). *A function $f : \{0,1\}^* \to \mathbb{Z}_+$ is in $\#\mathbf{P}$ iff there exists a polynomial $p(n)$ and a polynomial time Deterministic Turing Machine M so that for all $x \in \{0,1\}^*$:*

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

In particular, we define the following $\#\mathbf{P}$ function, which corresponds to counting the number of satisfying assignments to a Boolean formula:

Definition 11 ($\#\mathbf{SAT}$). *We define a function $\#\mathbf{SAT} : \{\mathbf{0},\mathbf{1}\}^* \to \mathbb{Z}_+$ which takes as input a binary encoding of a formula ψ and outputs the number of satisfying assignments to ψ.*

It is well known that $\#\mathbf{SAT}$ is $\#\mathbf{P}$-complete, and can be proven as an easy extension of the Cook-Levin theorem establishing the $\mathbf{NP}$-completeness of $\mathbf{SAT}$ (see e.g., [AB09]). Due to a theorem of Toda, we know that $\mathbf{PH}$ is no harder than $\mathbf{P}^{\#\mathbf{P}}$:

Theorem 12 (Toda [Tod91]). $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$

We also know that computing the Permanent of an $n \times n$ matrix with entries in $\{0,1\}$ is $\#\mathbf{P}$-complete.

Theorem 13 (Valiant [Val79]). *The function* $\mathbf{Permanent} : \{0,1\}^{n\times n} \to \mathbb{Z}$ *defined by* $\mathbf{Permanent}[X] = \sum\limits_{\sigma \in S_n} \prod\limits_{i=1}^{n} x_{i,\sigma(i)}$ *is* $\#\mathbf{P}$*-complete.*

The fact that $\mathbf{Permanent}$ is in $\#\mathbf{P}$ can be shown by the known equivalence between the $\mathbf{Permanent}$ of a $\{0,1\}$ matrix and counting the number of perfect matchings in a bipartite graph. Deciding if there is perfect matching in a bipartite graph is in $\mathbf{NP}$ (and in fact, in $\mathbf{P}$ by the Hopcroft-Karp algorithm [HK73]) and so counting the number of perfect matchings is a $\#\mathbf{P}$ problem. Valiant showed that counting the number of perfect matchings in a bipartite graph is also $\#\mathbf{P}$ hard, and so it is as hard as $\#\mathbf{SAT}$.

3.2 The Hardness of Multiplicative Estimation of the Permanent

In this work we will be interested in the robustness of this hardness result. First we state a famous result of Jerrum, Sinclair and Vigoda which tells us that we can achieve a multiplicative estimate to the Permanent of an $n \times n$ matrix with positive entries.

Theorem 14 (Jerrum, Sinclair, Vigoda [JSV04]). *Given as input a matrix* $X \in \mathbb{Z}_+^{n\times n}$*, we can approximate* $\mathbf{Permanent}[X]$ *to within multiplicative error* $\epsilon = 1/poly(n)$*, so that our output* α *is:*

$$(1-\epsilon)\mathbf{Permanent}[X] \leq \alpha \leq (1+\epsilon)\mathbf{Permanent}[X]$$

In randomized $poly(n)$ *time.*

However, we now show that the hardness result of Valiant is robust to multiplicative polynomial error, if we allow for matrices with positive and negative integer entries.

Theorem 15 (Aaronson [Aar11])**.** *Given as input a matrix $X \in \mathbb{Z}^{n \times n}$, where integer entries are described by binary values of length $poly(n)$, it is $\#\mathbf{P}$-hard to approximate $\mathbf{Permanent}[X]$ to within multiplicative error $\epsilon = 1/poly(n)$, so that our output α is:*

$$(1-\epsilon)\mathbf{Permanent}[X] \leq \alpha \leq (1+\epsilon)\mathbf{Permanent}[X]$$

(Proof Sketch following [Aar11]). We give a sketch of the proof. The full proof uses facts about linear-optics circuits that are beyond the scope of this thesis, but can be found in [Aar11].

Claim 16. *Given as input a description of a classical circuit C_f that computes a function $f : \{0,1\}^n \to \{0,1\}$, computing $\sum\limits_{x \in \{0,1\}^n} C_f(x)$ is $\#\mathbf{P}$-hard.*

This claim is a simple consequence of the $\#\mathbf{P}$-hardness of $\#\mathbf{SAT}$, for if we can solve this problem, we can compute the number of satisfying assignments to an arbitrary n-variate Boolean formula ψ with at most $poly(n)$ clauses, by allowing C_ψ to be the circuit encoding ψ and computing $\sum\limits_{x \in \{0,1\}^n} C_\psi(x)$.

Corollary 17. *Given as input a description of a classical circuit C_f that computes a function $f : \{0,1\}^n \to \{\pm 1\}$, computing $\sum\limits_{x \in \{0,1\}^n} C_f(x)$ is $\#\mathbf{P}$-hard.*

Proof. Given the ability to compute this sum for a $\{\pm 1\}$-valued circuit, we will show that we can obtain $\sum\limits_{x \in \{0,1\}^n} C_g(x)$, where C_g is a circuit that computes $g : \{0,1\}^n \to \{0,1\}$. As stated in Claim 16 this is $\#\mathbf{P}$-hard.

Note by adding an extra dummy variable, we can ensure without loss of generality that $k = \sum\limits_{x \in \{0,1\}^n} C_g(x) \leq 2^{n-1}$. Now we simply produce from C_g a $\{\pm 1\}$ valued circuit $C_{g'}$ defined to be equal to 1 on inputs x for which $C_g(x) = 0$ and -1 on inputs x for which $C_g(x) = 1$. Now note that $\sum\limits_{x \in \{0,1\}^n} C_{g'}(x) = 2^n - 2k$ and so we can obtain k by simply subtracting 2^n and dividing by -2. □

We now prove a lemma that is the primary technical hurdle involved in the proof of this theorem.

Lemma 18. *Given as input the description of an efficient classical circuit C_f computing a function $f : \{0,1\}^n \to \{\pm 1\}$ we can efficiently obtain a matrix X so that we can efficiently find $\sum_{x\in\{0,1\}^n} C_f(x)$ given the ability to compute* $\mathbf{Permanent}[X]$.

Proof. Our sketch proceeds with three Claims which taken together give our proof.

Claim 19. *There exists a classical algorithm that takes as input an efficient classical circuit C_f computing a function $f : \{0,1\}^n \to \{\pm 1\}$ and outputs the description of an efficient quantum circuit Q, running in time $poly(n, |C_f|)$ with the property:*

$$\langle 00...0|Q|00...0\rangle = \frac{\sum\limits_{x\in\{0,1\}^n} C_f(x)}{2^n}$$

Proof. Consider the following quantum circuit Q that is initialized on the all-zeros basis state $|00...0\rangle$ on n qubits:

1. Prepare the state $\frac{1}{2^{n/2}} \sum\limits_{x\in\{0,1\}^n} |x\rangle$
2. We can multiply $C_f(x)$ into the phases, with two quantum queries to C_f resulting in: $|C_f\rangle = \frac{1}{2^{n/2}} \sum\limits_{x\in\{0,1\}^n} C_f(x)|x\rangle$
3. Apply the Hadamard, $H^{\otimes n}$

Note that $H^{\otimes n}|C_f\rangle = \frac{1}{2^n} \sum\limits_{y\in\{0,1\}^n} \sum\limits_{x\in\{0,1\}^n} (-1)^{\langle x,y\rangle} C_f(x)|y\rangle$

The key observation that we are about to use is that $\langle 00...0|Q|00..0\rangle = \langle 00...0|H^{\otimes n}|C_f\rangle = \frac{\sum\limits_{x\in\{0,1\}^n} C_f(x)}{2^n}$, and therefore encodes a $\#\mathbf{P}$-hard quantity in an exponentially small amplitude. It is not hard to see (by fixing a universal quantum gate set), that the classical description of such a quantum circuit can be generated classically. $\square$

Claim 20. *By the quantum universality of "Postselected linear optics", as shown by [MRW+01], there exists a polynomial time classical algorithm that converts any quantum circuit Q to a Linear-optics circuit L so that the amplitude to which L maps its initial state to itself is proportional to $\langle 00...0|Q|00...0\rangle$. In particular, if we know this amplitude we can efficiently obtain $\langle 00...0|Q|00...0\rangle$.*

Using the two Claims together allows us to take a classical circuit C_f that computes a function $f : \{0,1\}^n \to \{\pm 1\}$ and produce a quantum state generated by a Linear-optics circuit, with an amplitude proportional to $\sum\limits_{x \in \{0,1\}^n} C_f(x)$. The next Claim connects this observation to the $\mathbf{Permanent}$ function.

Claim 21. *The amplitude to which an n-photon Linear-optics circuit L maps its initial state to itself can be expressed as the $\mathbf{Permanent}$ of an $n \times n$ matrix. This matrix can be efficiently obtained from the description of the circuit itself.*

Putting all the Claims together we prove Lemma 18, and conclude that if we can compute the $\mathbf{Permanent}$ of an arbitrary matrix, we have an efficient classical algorithm that uses this ability to compute $\sum\limits_{x \in \{0,1\}^n} C_f(x)$ for any efficiently computable $f : \{0,1\}^n \to \{\pm 1\}$. By Corollary 17 we conclude that this allows us to solve $\#\mathbf{P}$-hard problems. As noted in [Aar11] this gives a reproof of Valiant's theorem that computing $\mathbf{Permanent}$ exactly is $\#\mathbf{P}$-hard. $\square$

We can also use this to show our desired result, namely that computing a multiplicative estimate to the $\mathbf{Permanent}$ is $\#\mathbf{P}$-hard. Note that if we can compute the estimate guaranteed in the statement of the theorem, we can certainly compute the $sgn(\mathbf{Permanent}[X]) = \frac{\mathbf{Permanent}[X]}{|\mathbf{Permanent}[X]|}$. We will now show that even computing the $sgn(\mathbf{Permanent}[X])$ function is $\#\mathbf{P}$-hard. As mentioned before, the above Claims allow us to construct a matrix whose $\mathbf{Permanent}$ is proportional to $\sum\limits_x C(x)$. Thus, if we can compute the $sgn(\mathbf{Permanent}[X])$ function we can certainly compute the $sgn(\sum\limits_x C(x))$ function. We now prove that this task is $\#\mathbf{P}$-hard.

Lemma 22. *Given as input a circuit C_f which computes a function $f : \{0,1\}^n \to \{\pm 1\}$, computing $sgn\left(\sum\limits_{x \in \{0,1\}^n} C_f(x)\right)$ is $\#\mathbf{P}$-hard.*

Proof. First we note that by adding extra input bits to the circuit C_f we can create a circuit $C_f^{(k)}$ so that $\sum_{x\in\{0,1\}^n} C_f^{(k)}(x) = \left(\sum_{x\in\{0,1\}^n} f(x)\right) + k$ and likewise a circuit $C_f^{(-k)}$ so that $\sum_{x\in\{0,1\}^n} C_f^{(-k)}(x) = \left(\sum_{x\in\{0,1\}^n} f(x)\right) - k$.

Now we give a binary search procedure that exactly computes $\sum_{x\in\{0,1\}^n} C_f(x)$ given only the ability to compute $sgn\left(\sum_{x\in\{0,1\}^n} C_f(x)\right)$ (which we will refer to in passing as "checking the sign" of the circuit). The procedure proceeds in phases, using our ability to check the sign once in each phase. In the first phase, check the sign of C_f and if it's positive, we know that $0 < \sum_{x\in\{0,1\}^n} C_f(x) \leq 2^n$. Now we create a new circuit, $C_f^{(-2^{n-1})}$. If it's negative, we know that $-2^n \leq \sum_{x\in\{0,1\}^n} C_f(x) < 0$ and we create a new circuit $C_f^{(2^{n-1})}$. The second phase proceeds with whichever new circuit was created, checks the sign, and again creates a new circuit $C_f^{(-2^{n-2})}$ if the sign is positive and $C_f^{(2^{n-2})}$ if the sign is negative. Repeat this process, each time dividing in half until we have found the true value of $\sum_{x\in\{0,1\}^n} C_f(x)$.

□

This concludes our proof of the theorem as stated. □

Now we prove that even computing a multiplicative error estimate to $\mathbf{Permanent}^2[X]$ is $\#\mathbf{P}$-hard.

Theorem 23. *Given as input a matrix $X \in \mathbb{Z}^{n\times n}$, it is $\#\mathbf{P}$-hard to approximate $\mathbf{Permanent}^2[X]$ to within multiplicative error $\epsilon \geq 1/poly(n)$, so that our output α is:*

$$(1-\epsilon)\mathbf{Permanent}^2[X] \leq \alpha \leq (1+\epsilon)\mathbf{Permanent}^2[X]$$

Proof. Note that using the same methodology as in Lemma 18, we have established that, given as input the description of any classical circuit C_f computing a function $f : \{0,1\}^n \to \{\pm 1\}$ we can efficiently obtain a matrix X so that we can efficiently find $\left(\sum_{x\in\{0,1\}^n} C_f(x)\right)^2$ given the ability to compute $\mathbf{Permanent}^2[X]$. It is also clear from our discussion above that a multiplicative estimate to $\mathbf{Permanent}^2[X]$ is also a multiplicative estimate to $\left(\sum\limits_{x\in\{0,1\}^n} C_f(x)\right)^2$. We will show that the latter problem is $\#\mathbf{P}$-hard, which suffices to prove the theorem.

Lemma 24. *Given a circuit C_f that computes a function $f : \{0,1\}^n \to \{\pm 1\}$, it is $\#\mathbf{P}$-hard to approximate $\left(\sum\limits_{x\in\{0,1\}^n} C_f(x)\right)^2$ to within multiplicative error $\epsilon \geq 1/poly(n)$, so that our output α is:*

$$(1-\epsilon)\left(\sum_{x\in\{0,1\}^n} C_f(x)\right)^2 \leq \alpha \leq (1+\epsilon)\left(\sum_{x\in\{0,1\}^n} C_f(x)\right)^2$$

Proof. We will show how to compute $\left(\sum\limits_{x\in\{0,1\}^n} C_f(x)\right)^2$, which we define as β, exactly using the ability to compute this multiplicative error estimation.

Let $R = 2^{2n}$. We start by finding α, a $\left(1 \pm \frac{1}{4}\right)$-multiplicative estimate to β. We know that:

1. If $\alpha \leq \frac{1}{2}R$, then: $\frac{3}{4}\beta \leq \alpha \leq \frac{1}{2}R$, and so, $\beta \leq \frac{1/2}{3/4}R = \frac{2}{3}R$.
2. If $\alpha \geq \frac{1}{2}R$, then $\frac{5}{4}\beta \geq \alpha \geq \frac{1}{2}R_1$, and so, $\beta \geq \frac{1/2}{5/4}R_1 = \frac{2}{5}R$.

Now in either case we have ascertained a bound for β. In case 1, we know that $\beta \in [0, \frac{2}{3}R]$. In case 2, we know that $\beta \in [\frac{2}{5}R, R]$. In case 1, we can repeat with procedure with $R = 2^n$. In case 2, as in Lemma 22, we can produce a padded circuit, C_f' so that $0 \leq \left(\sum\limits_{x\in\{0,1\}^n} C_f'\right)^2 \leq \frac{3}{5}R$ and repeat the process with $R = 2^n$. Continue dividing R in half, repeating the process until β is ascertained exactly. □

□

3.3 The Hardness of Computing the Permanent over $\mathbb{F}$ on Most Matrices

Now we show that it is also $\#\mathbf{P}$-hard to compute $\mathbf{Permanent}^2$ on most matrices over a sufficiently large finite field. An analogous argument works for the $\mathbf{Permanent}$ function.

Theorem 25 (Lipton [Lip91])**.** *If there is a randomized polynomial time algorithm $\mathcal{O}$ such that:*

$$\Pr_X\left[\mathcal{O}(X) = \mathbf{Permanent}^2[X]\right] > 1 - 1/\left(6n+3\right)$$

With each element in X chosen uniformly at random from a finite field $\mathbb{F}$ of size at least $2n+1$, then we can use $\mathcal{O}$ at most a $poly(n)$ number of times to compute $\mathbf{Permanent}^2[X]$ for any $X \in \mathbb{F}^{n\times n}$ in randomized $poly(n)$ time.

Proof. The proof uses only that the $\mathbf{Permanent}^2$ function is of degree $2n$. We need to compute the $\mathbf{Permanent}^2[X]$ for an arbitrary $X \in \mathbb{F}^{n\times n}$. Consider a procedure that chooses a random $Y \in \mathbb{F}^{n\times n}$ and then picks an arbitrary subset $S \subseteq \mathbb{F}$ of cardinality $2n+1$ which doesn't include 0. For each $s \in S$, compute the value $a_s = \mathcal{O}(X + sY)$. Use Lagrange linear interpolation to compute the unique polynomial $p(s)$ of degree $2n$ such that $p(s) = a_s$ for all $s \in S$, and output $p(0)$. First we note that since Y is chosen uniformly at random from $\mathbb{F}^{n\times n}$, it is clear that, for each s, $X + sY$ is uniformly distributed over $\mathbb{F}^{n\times n}$. As a direct consequence of this we know:

Lemma 26. *For each X,*

$$\Pr_Y\left[\forall s \in S : p(s) = \mathbf{Permanent}^2\left[X + sY\right]\right] > 2/3$$

Proof. For each X, invoking a union bound, we have:

$$\begin{aligned}\Pr_Y\left[\exists s \in S : \mathbf{Permanent}^2\left[X+sY\right] \neq p(s)\right] &\leq \sum_{s\in S} \Pr\left[\mathbf{Permanent}^2\left[X+sY\right] \neq p(s)\right] \\ &\leq \frac{2n+1}{6n+3} = 1/3\end{aligned}$$

Where the last inequality comes from the error probability of $\mathcal{O}$. $\square$

Now conditioned on $p(s) = \mathbf{Permanent}^2\left[X + sY\right]$ for all $s \in S$ it follows that $p(0) = \mathbf{Permanent}^2[X]$ because, for fixed X and Y the univariate polynomial $f(s) = \mathbf{Permanent}^2[X + sY]$ and $p(s)$ are of degree $2n$ and agree on $2n + 1$ points. $\square$

Now we will show that we can prove this hardness result even when the matrix values are Real and the entries are distributed from an "autocorrelatable" distribution.

Definition 27 (Autocorrelatable distribution)**.** *We say a continuous distribution $\mathcal{D}$ over $\mathbb{R}$ is autocorrelatable if there exists some constant $c > 0$ so that for all $\epsilon > 0$ and $z \in [-1, 1]$:*

$$\int_{-\infty}^{\infty} |\mathcal{D}(x) - \mathcal{D}\left((1-\epsilon)\,x + \epsilon z\right)|\; dx \leq c\epsilon$$

For example, Aaronson and Arkhipov [AA13] have shown that the Gaussian Distribution with mean 0 and variance 1 is autocorrelatable.

Theorem 28 (Generalizing [AA13])**.** *Suppose $\mathcal{D}$ is some autocorrelatable distribution over $\mathbb{R}$ that can be sampled efficiently, and we are given an oracle $\mathcal{O}$ so that:*

$$\Pr_{Y \sim \mathcal{D}^{n\times n}}[\mathcal{O}(Y) = \mathbf{Permanent}^2[Y]] \geq 3/4 + \delta$$

for some $\delta = 1/poly(n)$. Then given an $X \in [-1, 1]^{n\times n}$, we can use $\mathcal{O}$ at most $poly(n)$ times to compute $\mathbf{Permanent}^2[X]$.

Proof. First we cite the Berlekamp-Welch algorithm on noisy interpolation of univariate polynomials over arbitrary fields, $\mathbb{F}$:

Theorem 29 (Berlekamp-Welch [Ber84])**.** *Let q be a univariate polynomial of degree d over any field $\mathbb{F}$. Suppose we are given m distinct pairs of $\mathbb{F}$-elements $(x_1, y_1), (x_2, y_2)...(x_m, y_m)$ and are* *promised that $q(x_i) = y_i$ for at least $\frac{m+d}{2}$ values of i. There exists a deterministic algorithm to reconstruct q using $poly(n, d)$ field operations.*

Given $X \in [-1, +1]^{n\times n}$, we choose $Y \sim \mathcal{D}^{n\times n}$ and let $X(t) = (1 - t)Y + tX$. Note that $X(1) = X$ and $X(0) = Y$. We define the univariate polynomial $q(t)$ of degree at most $2n$, so that $q(t) = \mathbf{Permanent}[X(t)]^2$.

We can no longer guarantee that the matrix $X(t)$ is distributed from $\mathcal{D}^{n\times n}$, but for small enough values of t, we can show that the distribution it is drawn from is close in statistical distance.

Now let $S = \frac{2n}{\delta}$ and $\epsilon = \frac{\delta}{2cSn^2}$. For every $s \in [S]$ we define $a_s = \mathcal{O}(X(\epsilon s))$. Our goal is to use the Berlekamp-Welch algorithm, Theorem 29, to recover $q(t)$, using the noisy evaluations a_s. Then we simply return $q(1) = \mathbf{Permanent}^2[X]$ as desired.

We know by definition:

$$\Pr_{Y\sim\mathcal{D}^{n\times n}}[\mathcal{O}(Y) = \mathbf{Permanent}[Y]^2] \geq 3/4 + \delta$$

Then if we let $\mathcal{D}_s$ be the distribution the matrix $X(\epsilon s)$ is drawn from, we have:

$$\Pr[\mathcal{O}(X(\epsilon s)) = q(\epsilon s)] \geq 3/4 + \delta - \|\mathcal{D}^{n\times n} - \mathcal{D}_s\| \geq 3/4 + \delta - c\epsilon S n^2 = 3/4 + \delta/2$$

Where the second inequality follows from the triangle inequality and the fact that $\mathcal{D}$ is autocorrelatable.

Let T be the set of s so that $q(\epsilon s) = a_s$.

By Markov:

$$\Pr\left[|T| \geq \left(\frac{1}{2} + \frac{\delta}{2}\right) S\right] \geq 1 - \frac{\frac{1}{4} - \frac{\delta}{2}}{\frac{1}{2} - \frac{\delta}{2}} \geq \frac{1}{2} + \frac{\delta}{2}$$

Since $S = 2n/\delta$ we have that $\left(\frac{1}{2} + \frac{\delta}{2}\right) S = \frac{S+2n}{2} = \frac{m+d}{2}$ when the number of sampled points $m = S$ and the degree $d = 2n$ and so we can use Berlekamp-Welch algorithm from Theorem 29 to compute $\mathbf{Permanent}^2[X]$ whenever $|T| \geq \left(\frac{1}{2} + \frac{\delta}{2}\right) S$.

By repeating $O(1/\delta^2)$ times for different choices of Y and taking the majority vote, we can compute $\mathbf{Permanent}^2[X]$.

$\square$

We now summarize these results in a table, which describes known hardness results for the $\mathbf{Permanent}^2$ (or equivalently, the **Permanent**) function.

Approximation	Entries in Matrix	Success Probability	$\#\mathbf{P}$-hard?
exact	$\{0, 1\}$ or $\mathbb{Z}$	1	Yes! Thm. 13
ϵ-multiplicative	$\mathbb{Z}_+$	1	Easy [JSV04]
ϵ-multiplicative	$\mathbb{Z}$	1	Yes! Thm. 15
exact	$\mathbb{F}_p, p = 2n + 1$	$1 - 1/(6n + 3)$	Yes! Thm. 25
exact	$\mathbb{R}$	$3/4 + 1/poly(n)$ (over autocorr. dist.)	Yes! Thm. 28
ϵ-multiplicative	$\mathbb{R}$	$1 - 1/poly(n)$	?

We also note that the Lipton result, Theorem 25, is known to be true for much stronger settings of parameters, using more sophisticated interpolation techniques (see [AA13] for a complete overview.)

Chapter 4

The Power of Exact Quantum Sampling

In this section we prove that, unless the $\mathbf{PH}$ collapses to a finite level, there is a class of distributions that can be sampled efficiently on a Quantum Computer, that cannot be sampled exactly classically. To do this we (again) cite a theorem by Stockmeyer on the ability to "approximate count" inside the $\mathbf{PH}$.

Theorem 30 (Stockmeyer [Sto85]). *Given as input a function $f : \{0,1\}^n \to \{0,1\}^m$ and $y \in \{0,1\}^m$, there is a procedure that outputs α such that:*

$$(1-\epsilon) \Pr_{x \sim U_{\{0,1\}^n}}[f(x) = y] \leq \alpha \leq (1+\epsilon) \Pr_{x \sim U_{\{0,1\}^n}}[f(x) = y]$$

In randomized time $poly(n, 1/\epsilon)$ with access to an $\mathbf{NP}$ oracle.

Note that as a consequence of Theorem 30, given an efficiently computable $f : \{0,1\}^n \to \{0,1\}$ we can compute a multiplicative approximation to $\Pr_{x \sim U_{\{0,1\}^n}}[f(x) = 1] = \frac{\sum_{x \in \{0,1\}^n} f(x)}{2^n}$ in the $\mathbf{PH}$.

Despite this, we have shown, as a consequence of Theorem 22, that the same multiplicative approximation becomes $\#\mathbf{P}$-hard if f is $\{\pm 1\}$-valued.

Now we show the promised class of quantumly sampleable distributions:

Definition 31 ($\mathcal{D}_f$). *Given $f : \{0,1\}^n \to \{\pm 1\}$, we define the distribution $\mathcal{D}_f$ over $\{0,1\}^n$ as* *follows:*

$$\Pr_{\mathcal{D}_{f,n}}[y] = \frac{\left(\sum\limits_{x\in\{0,1\}^n} (-1)^{\langle x,y\rangle} f(x)\right)^2}{2^{2n}}$$

The fact that this is a distribution will follow from the proceeding discussion.

Theorem 32. *For all efficiently computable $f : \{0,1\}^n \to \{\pm 1\}$ we can sample from $\mathcal{D}_f$ in $poly(n)$ time on a Quantum Computer.*

Proof. Consider the following quantum algorithm:

1. Prepare the state $\frac{1}{2^{n/2}} \sum\limits_{x\in\{0,1\}^n} |x\rangle$
2. Since by assumption f is efficiently computable, we can apply f to the phases (as discussed in Section 2.3), with two quantum queries to f resulting in:

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

3. Apply the n qubit Hadamard, $H^{\otimes n}$
4. Measure in the standard basis

Note that $H^{\otimes n}|f\rangle = \frac{1}{2^n} \sum\limits_{y\in\{0,1\}^n} \sum\limits_{x\in\{0,1\}^n} (-1)^{\langle x,y\rangle} f(x)|y\rangle$ and therefore the distribution sampled by the above quantum algorithm is $\mathcal{D}_f$. $\square$

As before, the key observation is that $(\langle 00...0|H^{\otimes n}|f\rangle)^2 = \frac{\left(\sum\limits_{x\in\{0,1\}^n} f(x)\right)^2}{2^{2n}}$, and therefore encodes a $\#\mathbf{P}$-hard quantity in an exponentially small amplitude. We can exploit this hardness classically if we assume the existence of a classical sampler, which we define to mean an efficient random algorithm whose output is distributed via this distribution.

Theorem 33 (Folklore, e.g., [Aar11]). *Suppose we have a classical randomized algorithm B, which given as input 0^n, samples from $\mathcal{D}_f$ in time $poly(n)$, then the* $\mathbf{PH}$ *collapses to* $\mathbf{BPP^{NP}}$.

Proof. The proof follows by applying Theorem 30 to obtain an approximate count to the fraction of random strings r so that $B(0^n, r) = 00..0$. Formally, we can output an α so that:

$$(1-\epsilon)\frac{\left(\sum\limits_{x\in\{0,1\}^n} f(x)\right)^2}{2^{2n}} \leq \alpha \leq \frac{\left(\sum\limits_{x\in\{0,1\}^n} f(x)\right)^2}{2^{2n}}(1+\epsilon)$$

In time $poly(n, 1/\epsilon)$ using an $\mathbf{NP}$ oracle. Multiplying through by 2^{2n} allows us to get a multiplicative approximation to $\left(\sum\limits_{x\in\{0,1\}^n} f(x)\right)^2$ in the $\mathbf{PH}$. This task is $\#\mathbf{P}$-hard, as proven in Theorem 23. Since we know by Theorem 12, $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$, we now have that $\mathbf{P}^{\#\mathbf{P}} \subseteq \mathbf{BPP^{NP}} \Rightarrow \mathbf{PH} \subseteq \mathbf{BPP^{NP}}$ leading to our theorem. Note that this theorem would hold even under the weaker assumption that the sampler is contained in $\mathbf{BPP^{PH}}$. $\square$

We end this Chapter by noting that Theorem 33 is extremely sensitive to the exactness condition imposed on the classical sampler, because the amplitude of the quantum state on which we based our hardness is only exponentially small. Thus it is clear that by weakening our sampler to an "approximate" setting in which the sampler is free to sample any distribution Y so that the Total Variation distance $\|Y - \mathcal{D}_f\| \leq 1/poly(n)$ we no longer can guarantee any complexity consequence using the above construction. Indeed, this observation makes the construction quite weak– for instance, it may even be unfair to demand that any physical realization of this quantum circuit *itself* samples exactly from this distribution! In the proceeding sections we are motivated by this apparent weakness and discuss the intractability of approximately sampling in this manner from quantumly sampleable distributions.

Chapter 5

The Hardness of Approximate Quantum Sampling

5.1 Approximate Sampling Definitions

In this section we define some simple terms which we will refer to throughout. As discussed in the prior section, we will be interested in demonstrating the existence of some distribution that can be sampled exactly by a uniform family of quantum circuits, that cannot be sampled approximately classically. Approximate here means close in Total Variation distance, where we refer to the Total Variation distance between two distributions X and Y by $\|X - Y\|$. Thus we define the notion of a Sampler to be a classical algorithm that approximately samples from a given class of distributions:

Definition 34 (Sampler)**.** *Let $\{D_n\}_{n>0}$ be a class of distributions where each D_n is distributed over $\mathbb{C}^n$. Let $r(n) \in poly(n), \epsilon(n) \in 1/poly(n)$. We say S is a $Sampler$ with respect to $\{D_n\}$ if $\|S(0^n, x \sim U_{\{0,1\}^{r(n)}}, 0^{1/\epsilon(n)}) - D_n\| \leq \epsilon(n)$ in (classical) polynomial time.*

In the next sections we will show a general class of distributions in which the existence of a Sampler implies the existence of an efficient approximation to an Efficiently Specifiable polynomial in the following two contexts:

Definition 35 ($\epsilon-$additive δ-approximate solution)**.** *Given a distribution D over $\mathbb{C}^n$ and $P : \mathbb{C}^n \rightarrow$*

$\mathbb{C}$ *we say* $T : \mathbb{C}^n \to \mathbb{C}$ *is an* $\epsilon-$*additive approximate* $\delta-$*average case solution with respect to* D*, to* $P : \mathbb{C}^n \to \mathbb{C}$*, if* $\Pr_{x \sim D}[|T(x) - P(x)| \leq \epsilon] \geq 1 - \delta$*.*

Definition 36 ($\epsilon-$multiplicative δ-approximate solution)**.** *Given a distribution* D *over* $\mathbb{C}^n$ *and a function* $P : \mathbb{C}^n \to \mathbb{C}$ *we say* $T : \mathbb{C}^n \to \mathbb{C}$ *is an* $\epsilon-$*multiplicative approximate* $\delta-$*average case solution with respect to* D*, if* $\Pr_{x \sim D}[|P(x) - T(x)| \leq \epsilon |P(x)|] \geq 1 - \delta$*.*

These definitions formalize a notion that we will need, in which an efficient algorithm computes a particular hard function approximately only on most inputs, and can act arbitrarily on a small fraction of remaining inputs. We conclude the section by giving two more definitions.

Definition 37 ($\mathbb{T}_\ell$)**.** *Given* $\ell > 0$*, we define the set* $\mathbb{T}_\ell = \{\omega_\ell^0, \omega_\ell^1 ..., \omega_\ell^{\ell-1}\}$ *where* ω_ℓ *is a primitive* ℓ*-th root of unity.*

We note that $\mathbb{T}_\ell$ is just ℓ evenly spaced points on the unit circle, and $\mathbb{T}_2 = \{\pm 1\}$.

Definition 38 ($\mathcal{B}(0,k)$)**.** *For* k *an even integer, we define the distribution* $\mathcal{B}(0,k)$ *over* $[-k,k]$*, so that:*

$$\Pr_{\mathcal{B}(0,k)}[y] = \begin{cases} \frac{\binom{k}{\frac{k+y}{2}}}{2^k} & if\ y\ is\ even \\ 0 & otherwise \end{cases}$$

5.2 Efficiently Specifiable Polynomial Sampling on a Quantum Computer

In this section we describe a general class of distributions that can be sampled efficiently on a Quantum Computer.

Lemma 39. *Let* $h : [m] \to \{0,1\}^n$ *be an efficiently computable one-to-one function, and suppose its inverse can also be efficiently computed. Then the superposition* $\frac{1}{\sqrt{m}} \sum\limits_{x \in [m]} |h(x)\rangle$ *can be efficiently prepared by a quantum algorithm.*

Proof. Our quantum procedure with first register consisting of m qubits and second of n qubits proceeds as follows:

1. Prepare $\frac{1}{\sqrt{m}} \sum\limits_{x \in [m]} |x\rangle|00...0\rangle$
2. Query h using the first register as input and the second as output:
$$\frac{1}{\sqrt{m}} \sum_{x \in [m]} |x\rangle|h(x)\rangle$$
3. Query h^{-1} using the second register as input and the first as output:
$$\frac{1}{\sqrt{m}} \sum_{x \in [m]} |x \oplus h^{-1}(h(x))\rangle|h(x)\rangle = \frac{1}{\sqrt{m}} \sum_{x \in [m]} |00...0\rangle|h(x)\rangle$$
4. Discard first register

□

Definition 40 (Efficiently Specifiable Polynomial)**.** *We say a multilinear homogenous n-variate polynomial Q with coefficients in $\{0,1\}$ and m monomials is* Efficiently Specifiable *via an efficiently computable, one-to-one function $h : [m] \to \{0,1\}^n$, with an efficiently computable inverse, if:*
$$Q(X_1, X_2..., X_n) = \sum_{z \in [m]} {X_1}^{h(z)_1} {X_2}^{h(z)_2} ... {X_n}^{h(z)_n}$$

Definition 41 ($\mathcal{D}_{Q,\ell}$)**.** *Suppose Q is an Efficiently Specifiable polynomial with m monomials. For fixed Q and ℓ, we define the class of distributions $\mathcal{D}_{Q,\ell}$ over ℓ-ary strings $y \in [0, \ell-1]^n$ given by:*
$$\Pr_{\mathcal{D}_{Q,\ell}}[y] = \frac{|Q(Z_y)|^2}{\ell^n m}$$

Where $Z_y \in \mathbb{T}_\ell^n$ is a vector of complex values encoded by the string y.

Theorem 42. *Given a polynomial, Q with m monomials and $\ell \leq exp(n)$, Efficiently Specifiable relative to h, the resulting $\mathcal{D}_{Q,\ell}$ can be sampled in $poly(n)$ time on a Quantum Computer.*

Proof. Note that h maps from $[m]$ to $\{0,1\}^n$ and we note that $\{0,1\}^n \subseteq [0,\ell-1]^n$.

1. We start in a uniform superposition over qudits of dimension ℓ, $\frac{1}{\sqrt{m}} \sum\limits_{z\in[m]} |z\rangle$.
2. We then apply Lemma 39 to prepare $\frac{1}{\sqrt{m}} \sum\limits_{z\in[m]} |h(z)\rangle$.
3. Apply Quantum Fourier Transform over $\mathbb{Z}_\ell^n$ to attain
$\frac{1}{\sqrt{\ell^n m}} \sum\limits_{y\in[0,\ell-1]^n} \sum\limits_{z\in[m]} \omega_\ell^{<y,h(z)>} |y\rangle$

Notice that the amplitude of each y basis state in the final state is proportional to the value of $Q(Z_y)$.

$\square$

5.3 Classical Hardness of Efficiently Specifiable Polynomial Sampling

In this section we use Stockmeyer's Theorem 30, together with the assumed existence of a classical sampler for $\mathcal{D}_{Q,\ell}$ to obtain hardness consequences.

Theorem 43. *Given an Efficiently Specifiable polynomial Q with n variables and m monomials, and a Sampler S with respect to $\mathcal{D}_{Q,\ell}$, there is a randomized procedure $T : \mathbb{C}^n \to \mathbb{C}$, an $(\epsilon \cdot m)-$additive approximate $\delta-$average case solution with respect to the uniform distribution over $\mathbb{T}_\ell^n$, to the $|Q|^2$ function, that runs in randomized time $poly(n, 1/\epsilon, 1/\delta)$ with access to an* $\mathbf{NP}$ *oracle.*

Proof. We need to give a procedure that outputs an ϵm-additive estimate to the $|Q|^2$ function evaluated at a uniform setting of the variables, with probability $1 - \delta$ over choice of setting. Setting $\beta = \frac{\epsilon\delta}{16}$, suppose S samples from a distribution $\mathcal{D}'$ such that $\|\mathcal{D}_{Q,\ell} - \mathcal{D}'\| \leq \beta$. We let p_y be $\Pr_{\mathcal{D}_{Q,\ell}}[y]$ and q_y be $\Pr_{\mathcal{D}'}[y]$.

Our procedure picks a uniformly chosen encoding of a setting $y \in [0, \ell - 1]^n$, and outputs an estimate $\tilde{q}_y$. Note that $p_y = \frac{|Q(Z_y)|^2}{\ell^n m}$. Thus our goal will be to output a $\tilde{q}_y$ that approximates p_y within additive error $\epsilon \frac{m}{\ell^n m} = \frac{\epsilon}{\ell^n}$, in time polynomial in n, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$.

We need:

$$\Pr_y[|\tilde{q}_y - p_y| > \frac{\epsilon}{\ell^n}] \leq \delta$$

First, define for each y, $\Delta_y = |p_y - q_y|$, and so $\|\mathcal{D}_{Q,\ell} - \mathcal{D}'\| = \frac{1}{2}\sum_y[\Delta_y]$.

Note that:

$$E_y[\Delta_y] = \frac{\sum_y[\Delta_y]}{\ell^n} = \frac{2\beta}{\ell^n}$$

And applying Markov, $\forall k > 1$,

$$\Pr_y[\Delta_y > \frac{k2\beta}{\ell^n}] < \frac{1}{k}$$

Setting $k = \frac{4}{\delta}, \beta = \frac{\epsilon\delta}{16}$, we have,

$$\Pr_y[\Delta_y > \frac{\epsilon}{2} \cdot \frac{1}{\ell^n}] < \frac{\delta}{4}$$

Then use approximate counting (with an $\mathbf{NP}$ oracle), using Theorem 30 on the randomness of S to obtain an output $\tilde{q}_y$ so that, for all $\gamma > 0$, in time polynomial in n and $\frac{1}{\gamma}$:

$$\Pr[|\tilde{q}_y - q_y| > \gamma \cdot q_y] < \frac{1}{2^n}$$

Because we can amplify the failure probability of Stockmeyer's algorithm to be inverse exponential. Note that:

$$E_y[q_y] \leq \frac{\sum_y q_y}{\ell^n} = \frac{1}{\ell^n}$$

Thus,

$$\Pr_y[q_y > \frac{k}{\ell^n}] < \frac{1}{k}$$

Now, setting $\gamma = \frac{\epsilon\delta}{8}$ and applying the union bound:

$$\begin{aligned}
\Pr_y[|\tilde{q}_y - p_y| > \frac{\epsilon}{\ell^n}] &\leq \Pr_y[|\tilde{q}_y - q_y| > \frac{\epsilon}{2} \cdot \frac{1}{\ell^n}] + \Pr_y[|q_y - p_y| > \frac{\epsilon}{2} \cdot \frac{1}{\ell^n}] \\
&\leq \Pr_y[q_y > \frac{k}{\ell^n}] + \Pr[|\tilde{q}_y - q_y| > \gamma \cdot q_y] + \Pr_y[\Delta_y > \frac{\epsilon}{2} \cdot \frac{1}{\ell^n}] \\
&\leq \frac{1}{k} + \frac{1}{2^n} + \frac{\delta}{4} \\
&\leq \frac{\delta}{2} + \frac{1}{2^n} \leq \delta.
\end{aligned}$$

□

Now, as proven in Section 5.5, we note that the variance of the distribution over $\mathbb{C}$ induced by an Efficiently Specifiable Q with m monomials, evaluated at uniformly distributed entries over $\mathbb{T}_\ell^n$ is m, and so the preceeding Theorem 43 promised us we can achieve an $\epsilon \mathrm{Var}[Q]$-additive approximation to Q^2, given a Sampler. We now show that, under a conjecture, this approximation can be used to obtain a good multiplicative estimate to Q^2. This conjecture effectively states that the Chebyshev inequality for this random variable is tight.

Conjecture 1 (Anti-Concentration Conjecture relative to an n-variate polynomial Q and distribution $\mathcal{D}$ over $\mathbb{C}^n$)**.** *There exists a polynomial p such that for all n and $\delta > 0$,*

$$\Pr_{X\sim\mathcal{D}}\left[|Q(X)|^2 < \frac{\mathrm{Var}[Q(X)]}{p(n, 1/\delta)}\right] < \delta$$

Theorem 44. *Assuming Conjecture 1, relative to an Efficiently Specifiable polynomial Q and a distribution $\mathcal{D}$, a $\epsilon \mathrm{Var}[Q]$-additive approximate δ-average case solution with respect to D, to the $|Q|^2$ function can be used to obtain an $\epsilon' \leq poly(n)\epsilon$-multiplicative approximate $\delta' = 2\delta$-average case solution with respect to $\mathcal{D}$ to $|Q|^2$.*

Proof. Suppose λ is, with high probability, an $\epsilon \mathrm{Var}[Q]$-additive approximation to $|Q(X)|^2$, as guaranteed in the statement of the Theorem. This means:

$$\Pr_{X\sim\mathcal{D}}\left[\left|\lambda - |Q(X)|^2\right| > \epsilon \mathrm{Var}[Q]\right] < \delta$$

Now assuming Conjecture 1 with polynomial p, we will show that λ is also a good multiplicative approximation to $|Q(X)|^2$ with high probability over X.

By the union bound,

$$\Pr_{X\sim\mathcal{D}}\left[\frac{\left|\lambda-|Q(X)|^2\right|}{\epsilon p(n,1/\delta)} > |Q(X)|^2\right] \leq \Pr_{X\sim D}\left[\left|\lambda-|Q(X)|^2\right| > \epsilon \mathrm{Var}\left[Q\right]\right] + \Pr_{X\sim\mathcal{D}}\left[\frac{\epsilon\mathrm{Var}\left[Q\right]}{\epsilon p(n,1/\delta)} > |Q(X)|^2\right] \leq 2\delta$$

Where the second line comes from Conjecture 1. Thus we can achieve any desired multiplicative error bounds (ϵ', δ') by setting $\delta = \delta'/2$ and $\epsilon = \epsilon'/p(n, 1/\delta)$.

□

For the results in this section to be meaningful, we simply need the Anti-Concentration conjecture to hold for some Efficiently Specifiable polynomial that is $\#\mathbf{P}$-hard to compute, relative to any distribution we can sample from (either U_n, or $\mathcal{B}(0,k)^n$). We note that Aaronson and Arkhipov [AA13] conjectures the same statement as Conjecture 1 for the special case of the $\mathbf{Permanent}$ function relative to matrices with entries distributed independently from the complex Gaussian distribution of mean 0 and variance 1.

Additionally, we acknowledge a result of Tao and Vu who show:

Theorem 45 (Tao & Vu [TV08])**.** *For all $\epsilon > 0$ and sufficiently large n,*

$$\Pr_{X\in\{\pm1\}^{n\times n}}\left[|\mathbf{Permanent}[X]| < \frac{\sqrt{n!}}{n^{\epsilon n}}\right] < \frac{1}{n^{0.1}}$$

Which comes quite close to our conjecture for the case of the $\mathbf{Permanent}$ function and uniformly distributed $\{\pm1\}^{n\times n} = \mathbb{T}_2^{n\times n}$ matrix. More specifically, for the above purpose of relating the additive hardness to the multiplicative, we would need an upper bound of any inverse polynomially large δ, instead of a fixed $n^{-0.1}$.

5.4 Sampling from Distributions with Probabilities Proportional to $[-k, k]$ Evaluations of Efficiently Specifiable Polynomials

In the prior sections we discussed quantum sampling from distributions in which the probabilities are proportional to evaluations of Efficiently Specifiable polynomials evaluated at points in $\mathbb{T}_\ell^n$. In this section we show how to generalize this to quantum sampling from distributions in which the probabilities are proportional to evaluations of Efficiently Specifiable polynomials evaluated at polynomially bounded integer values. In particular, we show a simple way to take an Efficiently Specifiable polynomial with n variables and create another Efficiently Specifiable polynomial with kn variables, in which evaluating this new polynomial at $\{-1, +1\}^{kn}$ is equivalent to evaluation of the old polynomial at $[-k, k]^n$.

Definition 46 (k-valued equivalent polynomial)**.** *For every Efficiently Specifiable polynomial Q with m monomials and every fixed $k > 0$ consider the polynomial $Q'_k : \mathbb{T}_2^{kn} \to \mathbb{R}$ defined by replacing each variable x_i in Q with the sum of k new variables $x_i^{(1)} + x_i^{(2)} + ... + x_i^{(k)}$. We will call Q'_k the k-valued equivalent polynomial with respect to Q.*

Theorem 47. *Suppose Q is an n-variate, homogeneous degree d Efficiently Specifiable polynomial with m monomials relative to a function $h : [m] \to \{0, 1\}^n$. Let $k \leq poly(n)$ and let Q'_k be the k-valued equivalent polynomial with respect to Q. Then Q'_k is Efficiently Specifiable with respect to an efficiently computable function $h' : [m] \times [k]^d \to \{0, 1\}^{kn}$.*

Proof. We first define and prove that h' is efficiently computable. We note that if there are m monomials in Q, there are mk^d monomials in Q'. As before, we'll think of the new variables in Q'_k as indexed by a pair of indices, a "top index" in $[k]$ and a "bottom index" in $[n]$. We are labeling each variable in Q' as $x_i^{(j)}$, the j-th copy of the i-th variable in Q. We are given $x \in [m]$ and $y_1, y_2, ..., y_d \in [k]$. Then, for all $i \in [n]$ and $j \in [k]$, we define the output, $z = h'(x, y_1, y_2, ..., y_d)_{i,j} = 1$ iff:

1. $h(x)_i = 1$

2. If $h(x)_i$ is the $\ell \leq d$-th non-zero element of $h(x)$, then we require $y_\ell = j$

We will now show that h'^{-1} is efficiently computable. As before we will think of $z \in \{0,1\}^{kn}$ as being indexed by a pair of indices, a 'top index" in $[k]$ and a "bottom index" in $[n]$. Then we compute $h'^{-1}(z)$ by first obtaining from z the bottom indices $j_1, j_2, ..., j_d$ and the corresponding top indices, $i_1, i_2, ..., i_d$. Then obtain from the bottom indices the string $x \in \{0,1\}^n$ corresponding to the indices of variables used in Q and output the concatenation of $h^{-1}(x)$ and $j_1, j_2, ..., j_d$.

□

5.5 Computation of the Variance of Efficiently Specifiable Polynomial

In this section we compute the variance of the distribution over $\mathbb{C}$ induced by an Efficiently Specifiable polynomial Q with assignments to the variables chosen independently from the $\mathcal{B}(0,k)$ distribution. We will denote this throughout the section by $\mathrm{Var}[Q]$. Recall, by the definition of Efficiently Specifiable, we have that Q is an n variate homogenous multilinear polynomial with $\{0,1\}$ coefficients. Assume Q is of degree d and has m monomials. Let each $[-k,k]$ valued variable X_i be independently distributed from $\mathcal{B}(0,k)$.

We adopt the notation whereby, for $j \in [m], l \in [d]$, x_{j_l} is the l-th variable in the j-th monomial of Q.

Using the notation we can express $Q(X_1, ..., X_n) = \sum\limits_{j=1}^{m} \prod\limits_{l=1}^{d} X_{j_l}$. By independence of these random variables and since they are mean 0, it suffices to compute the variance of each monomial and multiply by m:

$$\text{Var}\ [Q(X_1, ..., X_n)] = \text{E}\left[\sum_{j=1}^{m}\prod_{l=1}^{d} X_{j_l}^2\right] = \sum_{j=1}^{m}\text{E}\left[\prod_{l=1}^{d} X_{j_l}^2\right] \tag{5.1}$$

$$= m\text{E}\left[\prod_{l=1}^{d} X_{1_l}^2\right] = m\prod_{l=1}^{d}\text{E}\ \left[X_{1_l}^2\right] \tag{5.2}$$

$$= m\left(\text{E}\ \left[X_{1_l}^2\right]\right)^d \tag{5.3}$$

Now since these random variables are independent and identically distributed, we can calculate the variance of an arbitrary X_{j_l} for any $j \in [m]$ and $l \in [d]$:

$$\text{E}\left[X_{j_l}^2\right] = \frac{1}{2^k}\sum_{i=0}^{k}\left[(k-2i)^2\binom{k}{i}\right] \tag{5.4}$$

$$\tag{5.5}$$

Thus, the variance of Q is:

$$m\frac{1}{2^{kd}}\left(\sum_{i=0}^{k}\left[(k-2i)^2\binom{k}{i}\right]\right)^d$$

It will be useful to calculate this variance of Q in a different way, and obtain a simple closed form. In this way we will consider the k-valued equivalent polynomial $Q'_k : \mathbb{T}_2^{nk} \to \mathbb{C}$ which is a sum of $m' = mk^d$ multilinear monomials, each of degree d. As before we can write $Q'_k(X_1, ..., X_{nk}) = \sum_{j=1}^{m'}\prod_{l=1}^{d} X_{j_l}$. Note that the uniform distribution over assignments in $\mathbb{T}_2^{kn}$ to Q'_k induces $\mathcal{B}(0,k)^n$ over $[-k, k]^n$ assignments to Q. By the same argument as above, using symmetry and independence of random variables, we have:

$$\begin{aligned}
\mathrm{Var}\left[Q(X_1, X_2, ..., X_n)\right] =& \mathrm{Var}\left[Q'_k(X_1, X_2, ..., X_{nk})\right] && (5.6)\\
&= m' \prod_{l=1}^{d} \mathrm{E}\left[X_{1_l}^2\right] && (5.7)\\
&= m' \mathrm{E}\left[X_{1_l}^2\right]^d = 1^d m' = m' = k^d m && (5.8)
\end{aligned}$$

Chapter 6

Examples of Efficiently Specifiable Polynomials

In this Chapter we give two examples of Efficiently Specifiable polynomials.

6.1 Permanent is Efficiently Specifiable

Theorem 48. $\mathrm{Permanent}\,(x_1, ..., x_{n^2}) = \sum\limits_{\sigma \in S_n} \prod\limits_{i=1}^{n} x_{i,\sigma(i)}$ *is Efficiently Specifiable.*

Proof. We note that it will be convenient in this section to index starting from 0. The Theorem follows from the existence of an $h_{\mathbf{Permanent}} : [0, n! - 1] \to \{0,1\}^{n^2}$ that efficiently maps the i-th permutation to a string representing its obvious encoding as an $n \times n$ permutation matrix. We will prove that such an efficiently computable $h_{\mathbf{Permanent}}$ exists and prove that its inverse, $h_{\mathbf{Permanent}}^{-1}$ is also efficiently computable.

The existence of $h_{\mathbf{Permanent}}$ follows from the so-called "factorial number system" [Knu73], which gives an efficient bijection that associates each number in $[0, n! - 1]$ with a permutation in S_n. It is customary to think of the permutation encoded in the factorial number system as a permuted sequence of n numbers, so that each permutation is encoded in $n \log n$ bits. However, it is clear that we can efficiently transform this notation into its permutation matrix (using, for example, the trivial algorithm that searches for the positions of each of n elements in the $n \log n$ bit encoding), and vice-versa.

To go from an integer $j \in [0, n! - 1]$ to its permutation we:

1. Take j to its "factorial representation", an n number sequence, where the i-th place value is associated with $(i-1)!$, and the sum of the digits multiplied by the respective place value is the value of the number itself. We achieve this representation by starting from $(n-1)!$, setting the leftmost value of the representation to $j' = \lfloor \frac{j}{(n-1)!} \rfloor$, letting the next value be $\lfloor \frac{j - j' \cdot (n-1)!}{(n-2)!} \rfloor$ and continuing until 0. Clearly this process can be efficiently achieved and efficiently inverted, and observe that the largest each value in the i-th place value can be is i.

2. In each step we maintain a list ℓ which we think of as originally containing n numbers in ascending order from 0 to $n-1$.

3. Repeat this step n times, once for each number in the factorial representation. Going from left to right, start with the left-most number in the representation and output the value in that position in the list, ℓ. Remove that position from ℓ.

4. The resulting n number sequence is the encoding of the permutation, in the standard $n \log n$ bit encoding

To go from a permutation to its factorial representation, we can easily invert the process:

1. In each step we maintain a list ℓ which we think of as originally containing n numbers in order from 0 to $n-1$.

2. Repeat this step n times, once for each number in the encoding of the permutation. Going from left to right, start with the left-most number in the permutation and output the position

of that number (where we start with the 0-th position) in the list, ℓ. Remove that number from ℓ.

$\square$

6.2 The Hamiltonian Cycle Polynomial is Efficiently Specifiable

Given a graph G on n vertices, we say a Hamiltonian Cycle is a path in G that starts at a given vertex, visits each vertex in the graph exactly once and returns to the start vertex.

Likewise we define an n-cycle to be a Hamiltonian cycle in the complete graph on n vertices. Note that there are exactly $(n-1)!$ n-cycles in S_n.

Theorem 49. $\mathbf{HamiltonianCycle}\,(x_1, ..., x_{n^2}) = \sum\limits_{\sigma:\, n-cycle} \prod\limits_{i=1}^{n} x_{i,\sigma(i)}$ *is Efficiently Specifiable.*

Proof. We can modify the algorithm for the Permanent above to give us an efficiently computable $h_{HC} : [0, (n-1)! - 1] \rightarrow \{0, 1\}^{n^2}$ with an efficiently computable h_{HC}^{-1}.

To go from a number $j \in [0, (n-1)! - 1]$ to its n-cycle we:

1. Take j to its factorial representation as above. This time it is an $n-1$ number sequence where the i-th place value is associated with $(i-1)!$, and the sum of the digits multiplied by the respective place value is the value of the number itself.

2. In each step we maintain a list ℓ which we think of as originally containing n numbers in ascending order from 0 to $n-1$.

3. Repeat this step $n-1$ times, once for each number in the factorial representation. First remove the smallest element of the list. Then going from left to right, start with the left-most

number in the representation and output the value in that position in the list, ℓ. Remove that position from ℓ.

4. We output 0 as the n-th value of our n-cycle. The resulting n number sequence x is the n-cycle, in which the value of each x_i indicates the node to which the i-th node is mapped.

To take an n-cycle to a factorial representation, we can easily invert the process:

1. In each step we maintain a list ℓ which we think of as originally containing n numbers in order from 0 to $n-1$.

2. Repeat this step $n-1$ times. Remove the smallest element of the list. Going from left to right, start with the left-most number in the n-cycle and output the position of that number in the list ℓ (where we index the list starting with the 0 position). Remove the number at this position from ℓ.

$\square$

Chapter 7

Using the “Squashed” QFT

7.1 Efficient Quantum Sampling

In this section we begin to prove that Quantum Computers can sample efficiently from distributions with probabilities proportional to evaluations of Efficiently Specifiable polynomials at points in $[-k, k]^n$ for $k = exp(n)$. Note that in the prior quantum algorithm of Chapter 5 we would need to invoke the QFT over $\mathbb{Z}_2^{kn}$, of dimension doubly-exponential in n. Thus we need to define a new Polynomial Transform that can be obtained from the standard Quantum Fourier Transform over $\mathbb{Z}_2^n$, which we refer to as the “Squashed QFT”. Now we describe the unitary matrix which implements the Squashed QFT.

Consider the $2^k \times 2^k$ matrix D_k, whose columns are indexed by all possible 2^k multilinear monomials of the variables $x_1, x_2, ..., x_k$ and the rows are indexed by the 2^k different $\{-1, +1\}$ assignments to the variables. The (i, j)-th entry is then defined to be the evaluation of the j-th monomial on the i-th assignment. We note in passing that, defining $\bar{D}_k$ to be the matrix whose entries are the entries in D_k normalized by $1/\sqrt{2^k}$ gives us the Quantum Fourier Transform matrix over $\mathbb{Z}_2^k$.

Theorem 50. *The columns (and rows) in D_k are pairwise orthogonal.*

Proof. Here we will prove that the columns of D_k are pairwise orthogonal. A symmetric argument can be used to prove that the rows of D_k are pairwise orthogonal. Note that we can equivalently label the rows and columns of D_k by strings $\{0,1\}^n$, so that, for any $x, y \in \{0,1\}^n$, the (x,y)-th element of D_k is $-1^{\langle x,y\rangle}$. Take any pair of columns c_1, c_2 in the matrix, which are indexed by the strings $x_1, x_2 \in \{0,1\}^n$. Then:

$$\langle c_1, c_2\rangle = \sum_{y\in\{0,1\}^n} -1^{\langle x_1+x_2,y\rangle}$$

If $c_1 = c_2$ then $\langle x_1 + x_2, y\rangle = \langle 2x_1, y\rangle = 2\langle x_1, y\rangle$ is always even, and so $\langle c_1, c_2\rangle = 2^n$. If $c_1 \neq c_2$, it can be verified that there are as many strings y for which $\langle x_1 + x_2, y\rangle$ is even as odd, and so $\langle c_1, c_2\rangle = 0$. $\square$

Now we define the "Elementary Symmetric Polynomials":

Definition 51 (Elementary Symmetric Polynomials)**.** *We define the j-th Elementary Symmetric Polynomial on k variables for $j \in [0,k]$ to be:*

$$p_j(X_1, X_2, ..., X_k) = \sum_{1\leq \ell_1<\ell_2<...<\ell_j\leq k} X_{\ell_1}X_{\ell_2}...X_{\ell_j}$$

In this work we will care particularly about the first two elementary symmetric polynomials, p_0 and p_1 which are defined as $p_0(X_1, X_2, ..., X_k) = 1$ and $p_1(X_1, X_2, ..., X_k) = \sum\limits_{1\leq\ell\leq k} X_\ell$.

Consider the $(k+1)\times(k+1)$ matrix, $\tilde{D}_k$, whose columns are indexed by elementary symmetric polynomials on k variables and whose rows are indexed by equivalence classes of assignments in $\mathbb{Z}_2^k$ under S_k symmetry. We obtain $\tilde{D}_k$ from D_k using two steps.

First obtain a $2^k \times (k+1)$ rectangular matrix $\tilde{D}_k^{(1)}$ whose rows are indexed by assignments to the variables $x_1, x_2, ..., x_k \in \{\pm 1\}^k$ and columns are the entry-wise sum of the entries in each column of D_k whose monomial is in each respective elementary symmetric polynomial.

Then obtain the final $(k+1) \times (k+1)$ matrix $\tilde{D}_k$ by taking $\tilde{D}_k^{(1)}$ and keeping only one representative row in each equivalence class of assignments under S_k symmetry. We label the equivalence classes of assignments under S_k symmetry $o_0, o_1, o_2, ..., o_k$ and note that for each $i \in [k]$, $|o_i| = \binom{k}{i}$. Observe that $\tilde{D}_k$ is precisely the matrix whose (i,j)-th entry is the evaluation of the j-th symmetric polynomial evaluated on an assignment in the i-th symmetry class.

Theorem 52. *The columns in the matrix $\tilde{D}_k^{(1)}$ are pairwise orthogonal.*

Proof. Note that each column in the matrix $\tilde{D}_k^{(1)}$ is the sum of columns in D_k each of which are orthogonal. We can prove this theorem by observing that if we take any two columns in $D_k^{(1)}$, called c_1, c_2, where c_1 is the sum of columns $\{u_i\}$ of D_k and c_2 is the sum of columns $\{v_i\}$ of D_k. The inner product, $\langle c_1, c_2 \rangle$ can be written:

$$\langle \sum_i u_i, \sum_j v_j \rangle = \sum_{i,j} \langle u_i, v_j \rangle = 0$$

□

Theorem 53. *Let L be the $(k+1) \times (k+1)$ diagonal matrix with i-th entry equal to $\sqrt{o_i}$. Then the columns of $L \cdot \tilde{D}_k$ are orthogonal.*

Proof. Note that the value of the symmetric polynomial at each assignment in an equivalence class is the same. We have already concluded the orthogonality of columns in $\tilde{D}_k^{(1)}$. Therefore if we let a and b be any two columns in the matrix $\tilde{D}_k$, and their respective columns be $\bar{a}, \bar{b}$ in $\tilde{D}_k^{(1)}$, we can see:

$$\sum_{i=0}^{k} (a_i b_i |o_i|) = \sum_{i=0}^{2^k} \bar{a}_i \bar{b}_i = 0$$

From this we conclude that the columns of the matrix $L \cdot \tilde{D}_k$, in which the i-th row of $\tilde{D}_k$ is multiplied by $\sqrt{o_i}$, are orthogonal. □

Theorem 54. *We have just established that the columns in the matrix $L \cdot \tilde{D}_k$ are orthogonal. Let the $k+1 \times k+1$ diagonal matrix R be such that so that the columns in $L \cdot \tilde{D}_k \cdot R$ are orthonormal, and thus $L \cdot \tilde{D}_k \cdot R$ is unitary. Then the first two nonzero entries in R, which we call r_0, r_1, corresponding to the normalization of the column pertaining to the zero-th and first elementary symmetric polynomial, are $1/\sqrt{2^k}$ and $\frac{1}{\sqrt{\sum_{i=0}^{k}\left[\binom{k}{i}(k-2i)^2\right]}}$.*

Proof. First we calculate r_0. Since we wish for a unitary matrix, we want the ℓ_2 norm of the first column of $\tilde{D}_k$ to be 1, and so need:

$$r_0^2 \sum_{i=0}^{k} \left(\sqrt{o_i}\right)^2 = r_0^2 \sum_{i=0}^{k} \binom{k}{i} = 1$$

And so r_0 is $1/\sqrt{2^k}$ as desired.

Now we calculate r_1, the normalization in the column of $\tilde{D}_k$ corresponding to the first elementary symmetric polynomial. Note that in i-th equivalence class of assignments we have exactly i negative ones and $k-i$ positive ones. Thus the value of the first symmetric polynomial is the sum of these values, which for the $i-th$ equivalence class is precisely $k-2i$. Then we note the normalization in each row is $\sqrt{\binom{k}{i}}$. Thus we have

$$r_1^2 \sum_{i=0}^{k} \left[\sqrt{\binom{k}{i}}\,(k-2i)\right]^2 = 1$$

Thus $r_1 = \frac{1}{\sqrt{\sum_{i=0}^{k}\left[\binom{k}{i}(k-2i)^2\right]}}$ as desired.

□

7.2 A Simple Example of "Squashed" QFT, for $k = 2$

In this section we explicitly construct the matrix $L \cdot \tilde{D}_2 \cdot R$ from the QFT over $\mathbb{Z}_2^2$. Note that the matrix we referred to as D_2 is:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Where we can think of the columns as identified with the monomials $\{1, x_1, x_2, x_1x_2\}$ in this order (from left to right) and the rows (from top to bottom) as identified with the assignments $\{(1,1), (-1,1), (1,-1), (-1,-1)\}$ where the first element in each pair is the assignment to x_1 and the second is to x_2. Note that as desired, the (i,j)-th element of D_2 is the evaluation of the j-th monomial on the i-th assignment.

Now we create $\tilde{D}_2^{(1)}$ by combining columns of monomials that belong to each elementary symmetric polynomial, as described in the prior section. We identify the columns with elementary symmetric polynomials on variables x_1, x_2 in order from left to right: $1, x_1 + x_2, x_1x_2$ and the rows remain the same. This gives us:

$$\begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & -2 & 1 \end{pmatrix}$$

It can easily be verified that the columns are still orthogonal. Now we note that the rows corresponding to assignments $(1,-1)$ and $(-1,1)$ are in the same orbit with respect to S_2 symmetry.

And thus we obtain $\tilde{D}_2$:

$$\begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \\ 1 & -2 & 1 \end{pmatrix}$$

Now L is the diagonal matrix whose i-th entry is $\sqrt{o_i}$, the size of the i-th equivalence class of assignments under S_2 symmetry. Note that $|o_0| = \sqrt{\binom{2}{0}} = 1$, $|o_1| = \sqrt{\binom{2}{1}} = \sqrt{2}$, and $|o_2| = \sqrt{\binom{2}{2}} = 1$, and so L is:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

And $L \cdot \tilde{D}_2 =$

$$\begin{pmatrix} 1 & 2 & 1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & -2 & 1 \end{pmatrix}$$

And we note that the columns are now orthogonal. As before, this implies there exists a diagonal matrix R so that $L \cdot \tilde{D}_2 \cdot R$ is unitary. It is easily verified that this is the matrix R:

$$\begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{\sqrt{8}} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

And the first two elements r_0, r_1 can be easily seen to be $\frac{1}{\sqrt{2^k}} = \frac{1}{2}$ and $\frac{1}{\sqrt{\sum_{i=0}^{k}\left[\binom{k}{i}(k-2i)^2\right]}} = \frac{1}{\sqrt{8}}$, as claimed in the prior section. Thus the final $k+1 \times k+1$ matrix $L \cdot \tilde{D}_2 \cdot R$ is:

$$\begin{pmatrix} \frac{1}{2} & \frac{2}{\sqrt{8}} & \frac{1}{2} \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ \frac{1}{2} & -\frac{2}{\sqrt{8}} & \frac{1}{2} \end{pmatrix}$$

Which is unitary, as desired.

7.3 Using our "Squashed QFT" to Quantumly Sample from Distributions of Efficiently Specifiable Polynomial Evaluations

In this section we use the unitary matrix developed earlier to quantumly sample distributions with probabilities proportional to evaluations of Efficiently Specifiable polynomials at points in $[-k, k]^n$ for $k = exp(n)$. Here we assume that we have an efficient quantum circuit for this unitary. The prospects for this efficient decomposition are discussed in Section 8.

For convenience, we'll define a map $\psi : [-k, k] \to [0, k]$, for k even, with

$$\psi(y) = \begin{cases} \frac{k+y}{2} & if\ y\ is\ even \\ 0 & otherwise \end{cases}$$

Definition 55. *Suppose Q is an Efficiently Specifiable polynomial Q with n variables and m monomials, and, for $k \leq exp(n)$, let Q'_k be its k-valued equivalent polynomial. Let* $\mathrm{Var}\left[Q\right]$ *be the variance of the distribution over $\mathbb{C}$ induced by Q with assignments to the variables distributed over $\mathcal{B}(0, k)^n$ (or equivalently, we can talk about* $\mathrm{Var}\left[Q'_k\right]$ *where each variable in Q'_k is independently uniformly chosen from $\{\pm 1\}$), as calculated in Section 5.5. Then we define the of distribution $\mathcal{D}_{Q,k}$*

over n tuples of even integers in $[-k,k]$ by:

$$\Pr_{\mathcal{D}_{Q,k}}[y] = \frac{Q(y)^2 \binom{k}{\psi(y_1)}\binom{k}{\psi(y_2)}\dots\binom{k}{\psi(y_n)}}{2^{kn}\mathrm{Var}[Q]}$$

Theorem 56. *By applying $(L \cdot \tilde{D}_k \cdot R)^{\otimes n}$ in place of the Quantum Fourier Transform over $\mathbb{Z}_2^n$ in Section 5.2 we can efficiently quantumly sample from $\mathcal{D}_{Q,k}$.*

Proof. Since we are assuming Q is Efficiently Specifiable, let $h : [m] \to \{0,1\}^n$ be the invertible function describing the variables in each monomial. We start by producing the state over $k+1$ dimensional qudits:

$$\frac{1}{\sqrt{m}} \sum_{z \in [m]} |h(z)\rangle$$

Which we prepare via the procedure described in Lemma 39.

Instead of thinking of h as mapping an index of a monomial from $[m]$ to the variables in that monomial, we now think of h as taking an index of a monomial in Q to a polynomial expressed in the $\{1, x^{(1)} + x^{(2)} + ... + x^{(k)}\}^n$ basis.

Now take this state and apply the unitary (which we assume can be realized by an efficient quantum circuit) $(L \cdot \tilde{D}_k \cdot R)^{\otimes n}$.

Notice each $y \in [-k,k]^n$ has an associated amplitude:

$$\alpha_y = \frac{r_0^{n-d} r_1^d Q(y) \sqrt{\binom{k}{\psi(y_1)}\binom{k}{\psi(y_2)}\dots\binom{k}{\psi(y_n)}}}{\sqrt{m}}$$

Letting $p_y = \Pr_{\mathcal{D}_{Q,k}}[y]$, note that, by plugging in r_0, r_1 from Section 7.1:

$$\begin{aligned}
\alpha_y^2 &= \frac{Q(y)^2\binom{k}{\psi(y_1)}\binom{k}{\psi(y_2)}\cdots\binom{k}{\psi(y_n)}r_0^{2(n-d)}r_1^{2d}}{m} \\
&= \frac{Q(y)^2\binom{k}{\psi(y_1)}\binom{k}{\psi(y_2)}\cdots\binom{k}{\psi(y_n)}}{m2^{k(n-d)}\left(\sum_{i=0}^{k}\left[\binom{k}{i}(k-2i)^2\right]\right)^d} \\
&= \frac{Q(y)^2\binom{k}{\psi(y_1)}\binom{k}{\psi(y_2)}\cdots\binom{k}{\psi(y_n)}}{2^{kn-kd}\mathrm{Var}\left[Q\right]2^{kd}} = \frac{Q(y)^2\binom{k}{\psi(y_1)}\binom{k}{\psi(y_2)}\cdots\binom{k}{\psi(y_n)}}{2^{kn}\mathrm{Var}\left[Q\right]} = p_y
\end{aligned}$$

□

7.4 The Hardness of Classical Sampling from the Squashed Distribution

In this section, as before, we use Stockmeyer's Theorem (Theorem 30), together with the assumed existence of a classical sampler for $\mathcal{D}_{Q,k}$ to obtain hardness consequences for classical sampling with $k \le exp(n)$.

Theorem 57. *We fix some $k \le exp(n)$. Given an Efficiently Specifiable polynomial Q with n variables and m monomials, let Q'_k be its k-valued equivalent polynomial. Suppose we have a Sampler S with respect to our quantumly sampled distribution class, $\mathcal{D}_{Q,k}$, and let* $\mathrm{Var}\left[Q\right]$ *denote the variance of the distribution over $\mathbb{C}$ induced by Q with assignments distributed from $\mathcal{B}(0,k)^n$. Then we can find a randomized procedure $T : \mathbb{R}^n \to \mathbb{R}$, an $\epsilon\mathrm{Var}\left[Q\right]$-additive approximate δ-average case solution to Q^2 with respect to $\mathcal{B}(0,k)^n$ that runs in time $poly(n, 1/\epsilon, 1/\delta)$ with access to an* $\mathbf{NP}$ *oracle.*

Proof. Setting $\beta = \epsilon\delta/16$, suppose S samples from a class of distributions $\mathcal{D}'$ so that $\|\mathcal{D}_{Q,k} - \mathcal{D}'\| \le \beta$. Let $q_y = \Pr_{\mathcal{D}'}[y]$.

We define $\phi : \{\pm 1\}^{kn} \to [-k,k]^n$ to be the map from each $\{\pm 1\}^{kn}$ assignment to its equivalence class of assignments, which is n blocks of even integral values in the interval $[-k,k]$. Note that, given a uniformly random $\{\pm 1\}^{kn}$ assignment, ϕ induces the $\mathcal{B}(0,k)$ distribution over $[-k,k]^n$.

Our procedure picks a $y \in [-k,k]^n$ distributed[1] via $\mathcal{B}(0,k)^n$, and outputs an estimate $\tilde{q}_y$. Equivalently, we analyze this procedure by considering a uniformly distributed $x \in \{\pm 1\}^{kn}$ and then returning an approximate count, $\tilde{q}_{\phi(x)}$ to $q_{\phi(x)}$. We prove that our procedure runs in time $poly(n, 1/\epsilon, 1/\delta)$ with the guarantee that:

$$\Pr_x \left[\frac{|\tilde{q}_{\phi(x)} - p_{\phi(x)}|}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}} > \frac{\epsilon}{2^{kn}} \right] \leq \delta$$

And by our above analysis of the quantum sampler:

$$p_{\phi(x)} = \frac{Q(\phi(x))^2\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}}{2^{kn}\mathrm{Var}\left[Q\right]}$$

Note that: $\frac{1}{2} \sum\limits_{y \in [-k,+k]^n} |p_y - q_y| \leq \beta$ and thus because we are summing over as many times as the size of its orbit under $(S_k)^n$, we know that:

$$\frac{1}{2} \sum_{x \in \{\pm 1\}^{kn}} \frac{\left|p_{\phi(x)} - q_{\phi(x)}\right|}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}} \leq \beta$$

[1]We can do this when $k = exp(n)$ by approximately sampling from the Normal distribution, with only $poly(n)$ bits of randomness, and using this to approximate $\mathcal{B}(0,k)$ to within additive error $1/poly(n)$ [BM58, Ber41].

First we define for each x, $\Delta_x = \frac{|p_{\phi(x)} - q_{\phi(x)}|}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}}$ and so $\|\mathcal{D}_{Q,k} - \mathcal{D}'\| = \frac{1}{2}\sum_x \Delta_x$.

Note that:

$$\mathrm{E}_x[\Delta_x] = \frac{\sum_x \Delta_x}{2^{kn}} = \frac{2\beta}{2^{kn}}$$

And applying Markov, $\forall k > 1$,

$$\Pr_x[\Delta_x > \frac{k2\beta}{2^{kn}}] < \frac{1}{k}$$

Setting $k = \frac{4}{\delta}, \beta = \frac{\epsilon\delta}{16}$, we have,

$$\Pr_x[\Delta_x > \frac{\epsilon}{2} \cdot \frac{1}{2^{kn}}] < \frac{\delta}{4}$$

Then use approximate counting (with an $\mathbf{NP}$ oracle), using Theorem 30 on the randomness of S to obtain an output $\tilde{q}_y$ so that, for all $\gamma > 0$, in time polynomial in n and $\frac{1}{\gamma}$:

$$\Pr[|\tilde{q}_y - q_y| > \gamma \cdot q_y] < \frac{1}{2^n}$$

Because we can amplify the failure probability of Stockmeyer's algorithm to be inverse exponential.

Equivalently in terms of x:

$$\Pr_x\left[\frac{|\tilde{q}_{\phi(x)} - q_{\phi(x)}|}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}} > \gamma \cdot \frac{q_{\phi(x)}}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}}\right] < \frac{1}{2^n}$$

And we have:

$$\mathrm{E}_x\left[\frac{q_{\phi(x)}}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}}\right] \le \frac{\sum_x \frac{q_{\phi(x)}}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}}}{2^{kn}} = \frac{1}{2^{kn}}$$

Thus, by Markov,

$$\Pr_x\left[\frac{q_{\phi(x)}}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}} > \frac{k}{2^{kn}}\right] < \frac{1}{k}$$

Now, setting $\gamma = \frac{\epsilon\delta}{8}$ and applying the union bound:

$$\begin{aligned}
&\Pr_x\left[\frac{\left|\tilde{q}_{\phi(x)} - p_{\phi(x)}\right|}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}} > \frac{\epsilon}{2^{kn}}\right] \\
&\quad \le \Pr_x\left[\frac{\left|\tilde{q}_{\phi(x)} - q_{\phi(x)}\right|}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}} > \frac{\epsilon}{2}\cdot\frac{1}{2^{kn}}\right] + \Pr_x\left[\frac{\left|q_{\phi(x)} - p_{\phi(x)}\right|}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}} > \frac{\epsilon}{2}\cdot\frac{1}{2^{kn}}\right] \\
&\quad \le \Pr_x\left[\frac{q_{\phi(x)}}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}} > \frac{k}{2^{kn}}\right] \\
&\quad + \Pr\left[\frac{\left|\tilde{q}_{\phi(x)} - q_{\phi(x)}\right|}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}} > \gamma\cdot\frac{q_{\phi(x)}}{\binom{k}{\psi(\phi(x)_1)}\binom{k}{\psi(\phi(x)_2)}\cdots\binom{k}{\psi(\phi(x)_n)}}\right] + \Pr_x\left[\Delta_x > \frac{\epsilon}{2}\cdot\frac{1}{2^{kn}}\right] \\
&\quad \le \frac{1}{k} + \frac{1}{2^n} + \frac{\delta}{4} \\
&\quad \le \frac{\delta}{2} + \frac{1}{2^n} \le \delta.
\end{aligned}$$

□

Chapter 8

Putting it All Together

In this chapter we put our results in perspective and conclude.

As mentioned before, our goal is to find a class of distributions $\{\mathcal{D}_n\}_{n>0}$ that can be sampled exactly in $poly(n)$ time on a Quantum Computer, with the property that there does not exist a (classical) Sampler relative to that class of distributions, $\{\mathcal{D}_n\}_{n>0}$.

Using the results in Sections 5.3 and 5.4 we can quantumly sample from a class of distributions $\{\mathcal{D}_{Q,k}\}_{n>0}$, where $k = poly(n)$ with the property that, if there exists a classical Sampler relative to this class of distributions, there exists an $\epsilon\mathrm{Var}\left[Q\right]$-additive δ-average case solution to the Q^2 function with respect to the $\mathcal{B}(0,k)^n$ distribution. If we had an efficient decomposition for the "Squashed QFT" unitary matrix, we could use the results from Sections 7.3 and 7.4 to make k as large as $\exp(n)$. We would like this to be an infeasible proposition, and so we conjecture:

Conjecture 2. *There exists some Efficiently Specifiable polynomial Q with n variables, so that $\epsilon\mathrm{Var}\left[Q\right]$-additive δ-average case solutions with respect to $\mathcal{B}(0,k)^n$, for any fixed $k < exp(n)$, to Q^2, cannot be computed in (classical) randomized $poly(n, 1/\epsilon, 1/\delta)$ time with a $\mathbf{PH}$ oracle.*

At the moment we don't know of such a decomposition for the "Squashed QFT". However, we do know that we can classically evaluate a related fast (time $n\log^2 n$) polynomial transform by a theorem of Driscoll, Healy, and Rockmore [DJR97]. We wonder if there is some way to use intuition gained by the existence of this fast polynomial transform to show the existence of an efficient decomposition for our “Squashed QFT”.

Additionally, if we can prove the Anti-Concentration Conjecture (Conjecture 1) relative to some Efficiently Specifiable polynomial Q and the $\mathcal{B}(0,k)^n$ distribution, we appeal to Theorem 44 to show that it suffices to prove:

Conjecture 3. *There exists some Efficiently Specifiable polynomial Q with n variables, so that Q satisfies Conjecture 1 relative to $\mathcal{B}(0,k)^n$, for $k \leq \exp(n)$, and ϵ-multiplicative δ-average case solutions, with respect to $\mathcal{B}(0,k)^n$, to Q^2 cannot be computed in (classical) randomized $poly(n, 1/\epsilon, 1/\delta)$ time with a $\mathbf{PH}$ oracle.*

We would be happy to prove that either of these two solutions (additive or multiplicative) are $\#\mathbf{P}$-hard. In this case we can simply invoke Toda’s Theorem (Theorem 12) to show that such a randomized classical solution would collapse $\mathbf{PH}$ to some finite level.

We note that at present, both of these conjectures seem out of reach, because we do not have an example of a polynomial that is $\#\mathbf{P}$-hard to approximate (in either multiplicative or additive) on average, in the sense that we need. Hopefully this is a consequence of a failure of proof techniques, and can be addressed in the future with new ideas.

Bibliography

[AA13] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. *Theory of Computing*, 9:143–252, 2013.

[Aar10a] Scott Aaronson. Bqp and the polynomial hierarchy. In Leonard J. Schulman, editor, *STOC*, pages 141–150. ACM, 2010.

[Aar10b] Scott Aaronson. A counterexample to the Generalized Linial-Nisan conjecture. *ECCC Report 109*, 2010.

[Aar10c] Scott Aaronson. The equivalence of sampling and searching. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:128, 2010.

[Aar11] Scott Aaronson. A linear-optical proof that the permanent is #p-hard. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:43, 2011.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997.

[Ber41] Andrew C Berry. The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49(1):122–136, 1941.

[Ber84] E. R. Berlekamp. *Algebraic coding theory*. Aegean Park Press, Laguna Hills, CA, USA, 1984.

[BJS10] Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. 2010.

[BM58] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, 29:610–611, 1958.

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

[DHM$^+$05] Christopher M. Dawson, Andrew P. Hines, Duncan Mortimer, Henry L. Haselgrove, Michael A. Nielsen, and Tobias Osborne. Quantum computing and polynomial equations over the finite field z2. *Quantum Information & Computation*, 5(2):102–112, 2005.

[DJR97] James R. Driscoll, Dennis M. Healy Jr., and Daniel N. Rockmore. Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs. *SIAM J. Comput.*, 26(4):1066–1099, 1997.

[FFK91] Stephen A. Fenner, Lance Fortnow, and Stuart A. Kurtz. Gap-definable counting classes. In *Structure in Complexity Theory Conference*, pages 30–42. IEEE Computer Society, 1991.

[FR99] Lance Fortnow and John D. Rogers. Complexity limitations on quantum computation. *J. Comput. Syst. Sci.*, 59(2):240–252, 1999.

[FU11] Bill Fefferman and Chris Umans. On pseudorandom generators and the BQP vs PH problem. *QIP*, 2011.

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[JSV04] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.

[Knu73] Donald E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.

[KSV02] A.Y Kitaev, A.H Shen, and M.N Vyalyi. *Quantum and Classical Computation*. AMS, 2002.

[KvM02] Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[Lip91] Richard J. Lipton. New directions in testing. *DIMACS Distributed Computing and Cryptography*, 2(1):191, 1991.

[MRW$^+$01] Gerard J. Milburn, Timothy C. Ralph, Andrew G. White, Emanuel Knill, and Raymond Laflamme. Efficient linear optics quantum computation. *Quantum Information & Computation*, 1(4):13–19, 2001.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quanum Information*. Cambridge U.P., 2000.

[Shi03] Yaoyun Shi. Both toffoli and controlled-not need little help to do universal quantum computing. *Quantum Information & Computation*, 3(1):84–92, 2003.

[Sho94] Peter W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, page 289. Springer, 1994.

[Sto85] Larry J. Stockmeyer. On approximation algorithms for #p. *SIAM J. Comput.*, 14(4):849–861, 1985.

[Tod91] Seinosuke Toda. Pp is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[TV08] Terence Tao and Van Vu. On the permanent of random bernoulli matrices. In *Advances in Mathematics*, page 75, 2008.

[Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.