Quantum Complexity Theory and its Implications in Cryptography

Ann He, Charissa Plattner

2019

1. Introduction

Richard Feynman, an American physicist most notably known for his contributions to quantum mechanics, claimed in his paper *Simulating physics with computers*, that "The only difference between a probabilistic classical world and the equations of the quantum world is that somehow or other it appears as if the probabilities would have to go negative, and that we do not know, as far as I know, how to simulate" [6]. Coming from a world of traditional complexity theory, it's hard to accept the idea that the probability of some event can be negative. However, as we will describe this paper, it is this phenomena (as well as other quantum properties) that underlies much of the supposed power that is gained in quantum computation models over traditional computation models.

Quantum computing is a recent area of active research in physics and computer science, predicated on the theory of quantum mechanics, which describes the interaction of particles and energy at the smallest of scales, reconciling observations escaping earlier theories. In the 1980s, the notion of a quantum computer, a machine that used the effects of quantum mechanics to its computational advantage, was still a fantasy in the mind of Richard Feynman. As recent efforts in experimental quantum physics hint at the viability of quantum computers, we think it is especially interesting to consider the consequences of such a technological advance.

A simple but illustrative experiment to see the power of quantum computation is the parity game. In the parity game, a challenger sends a random bit x to Alice and a random bit y to Bob, who are unable to communicate with each other. Alice and Bob respond with, respectively a and b and they win just in case $a \oplus b = x \wedge y$. In a classical setting, it can be shown that no strategy of Alice and Bob can win more than $\frac{3}{4}$ of the time, but in the quantum setting, by preparing a quantum register in the EPR state, Alice and Bob can leverage quantum entanglement to win the game at least $\frac{4}{5}$ of the time.

In our survey paper, we will start with a basic introduction to the quantum mechanics necessary to understand quantum computing at a high level. We will then introduce the quantum complexity class BQP, the class of decision problems that can be efficiently computed on a quantum Turing machine. We then compare and contrast various properties of quantum complexity classes with traditional complexity classes, as well as show where BQP exists with respect to those classes. We will then consider the practical and theoretical ramifications of quantum computing. Specifically, we discuss Shor's algorithm, a polynomial time quantum algorithm for integer factorization, which would break modern cryptographic systems, and the connection of quantum complexity theory to the Church-Turing Thesis.

2. Foundations of Basic Quantum Computation

2.1 Qubits and quantum states

In classical computing, we consider the bit, which can either be 0 or 1, as the most atomic unit of information. In quantum computing, the analogous unit is the qubit, which can simultaneously

occupy the state 0 or 1, with different amplitudes. Amplitudes can be thought of as a generalization of probability measures, except that they can take on negative values. Whereas probabilities must sum to 1, the $l2$ norm of amplitudes evaluates to 1. More specifically if the qubit is in state 0 with amplitude α_0 and state 1 with amplitude α_1 then we must have $\alpha_0^2 + \alpha_1^2 = 1$. We represent a superposition of states as a sum weighted with corresponding amplitudes, i.e. $\alpha_0 |0\rangle + \alpha_1 |1\rangle$.

As long as no outside influences interfere, the qubit remains in its superposition with the corresponding amplitudes. When measured (observed), the qubit collapses to state 0 with probability $| \alpha_0 |^2$ and state 1 with probability $| \alpha_1 |^2$. After observation, information about the amplitude wave collapses, and the amplitude values are irrevocably lost.

2.2 Quantum operations and quantum interference

Quantum operations are $l2$-norm preserving linear operations on amplitude vectors. In the language of linear algebra, quantum operations correspond to unitary matrices. Importantly, this means that all quantum operations must be reversible.

In traditional computing, there is the notion of an *m-bit register*, or a string of m bits in $\{0,1\}^m$. The entire state of a m-bit register can be described in a single string of m bits. In quantum computing, the analog is a quantum register. A *quantum register* is a string of m qubits, whose state is a *superposition* of all 2^m basic states. As briefly described in section 2.1 as an "amplitude vector", the superposition is a unit vector of probabilities, such that $v = (v_{0^m}, v_{0^{m-1}1}, ...v_{1^m}) \in \mathbb{C}^{2^m}$. The important thing to note here is that the superposition of the qubit register is in the space $\mathbb{C}^{2^m}$.

Quantum weirdness dictates that when the state of the register is measured, the register collapses down into a single state x, where $x \in \{0,1\}^m$, with probability $|v_x|^2$. To put this concretely, let's consider a 2-qubit register. The superposition of this vector would contain $2^m = 4$ probabilities, such that the superposition would look like $[v_{00}, v_{01}, v_{10}, v_{11}]$ and $\sum_x |v_x|^2 = 1$. The probability that, when observed, the value of the register is $x = 01$, is equal to the value of $|v_{01}|^2$ in the superposition vector.

Formally, a quantum operation for an m-qubit register is a function $F : \mathbb{C}^{2^m} \to \mathbb{C}^{2^m}$ which is

- Linear: For every $v \in \mathbb{C}^{2^m}$, $F(v) = \sum_x v_x F(|x\rangle)$
- Norm preserving: For every $v \in \mathbb{C}^{2^m}$ such that $v_2 = 1$, $F(v)_2 = 1$

Understanding exactly why linearity is required is not necessary for the scope of this paper, but know that it is a requisite from quantum theory. Norm preservation is, of course, required because the superpositions are probability vectors, so the norm must be preserved to maintain the integrity of the probabilities.

Quantum interference refers to the wave-properties of quantum mechanics where paths to the same state can impact each other constructively or destructively. The mode of interference is additive, whereby two paths to the same state which have the same sign amplitude interfere constructively whilst two paths to the same state with amplitudes that have opposing signs interfere destructively.

2.3 Examples of quantum operations

- Flipping qubits: Flipping qubits corresponds to a permutation matrix, which is a unitary matrix since each pair of columns is orthogonal and has norm 1.

  Consider the example of flipping the first bit on a 2-bit register. Then we need to perform the linear operation

  - $\alpha_0 |00\rangle \mapsto \alpha_0 |10\rangle$
  - $\alpha_1 |01\rangle \mapsto \alpha_1 |11\rangle$
  - $\alpha_2 |10\rangle \mapsto \alpha_2 |00\rangle$

– $\alpha_3 |11\rangle \mapsto \alpha_3 |01\rangle$

Which can be implemented by the matrix $\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

- Reordering qubits: Reordering qubits, ie permuting the bits, also corresponds to permutation matrices.
- Copying qubits / controlled not: Actually copying qubits into each other is not a valid transformation as it is not reversible and hence not unitary. For example, consider the operation that copies the first bit into the second bit. Then $|10\rangle$ and $|11\rangle$ map to $|11\rangle$ and both $|00\rangle$ and $|01\rangle$ map to $|00\rangle$, and there is no way to know the pre-image.

  However, we use a "write once" manner to copy a qubit into a fresh qubit in some pre-arranged state like $|0\rangle$ and copying only once into them. Then, the reverse operation is simply to map that qubit back to $|0\rangle$.

  We can write this operation as $|xy\rangle \mapsto |x(x \oplus y)\rangle$
- The Hadamard operation: The *Hadamard gate* maps $|0\rangle$ to $|0\rangle + |1\rangle$ and $|1\rangle$ to $|0\rangle - |1\rangle$.

  Applying a Hadamard gate to every qubit on an m-bit register we map the state $|x\rangle$ for $x \in \{0,1\}^m$ to

$$\begin{aligned}(|0\rangle + (-1)^{x_1} |1\rangle)(|0\rangle + (-1)^{x_2} |1\rangle)...(|0\rangle + (-1)^{x_m} |1\rangle) = \\ = \sum_{y \in \{0,1\}^m} (\prod_{i:y_i=1} (-1)^{x_i}) |y\rangle \\ = \sum_{y \in \{0,1\}^m} -1^{x \odot y} |y\rangle\end{aligned}$$

- Reversible AND: The *Toffoli gate* uses an additional bit in a quantum register to compute a reversible AND. It maps the state $|b_1\rangle |b_2\rangle |b_3\rangle \to |b_1\rangle |b_2\rangle |b_3 \oplus (b_1 \wedge b_2)\rangle$. The bit b_3 must be in a fresh $|0\rangle$ state in order for the operation to be performed correctly. We see that we are first taking the AND of b_1 and b_2, and then XOR'ing that result with $b_3 = 0$, which gives us $b_3 = b_1 \wedge b_2$.
- Reversible OR: One can construct a reversible OR gate in a method similar to the Toffoli gate described above.

A quantum operation is considered to be *elementary* if it operates on three or fewer qubits of a register at a time. Thus, the realm of elementary quantum matrices is limited to square matrices of size 2×2, 4×4, or 8×8. Note that a 3-qubit system will have a superposition vector of size $2^m = 2^3 = 8$; thus a elementary matrix operating on 3 qubits will be of size 8×8. At this point, stop and try to internalize the concept of an elementary quantum operation. All quantum algorithms and operations discussed in the rest of this paper will rely on the assumption that the operations used are elementary; that is, the operations operate on only a small portion of the entire register at once.

2.4 Universal set of quantum operations

Say that a set S of quantum gates is universal if you can approximate any unitary transformation on any number of qubits to any desired precision by composing gates from S. It is important to note that given finite S the best we can do is approximation, since there are uncountably infinite unitary transformations while the gates in S can be composed in only countably infinite ways.

Ways in which gate sets can fail universality are if they fail to model superposition, interference, entanglement, do not allow for complex amplitudes, or are contained in the stabilizer set, which is the set $M = \{cNOT, Hadamard, P\}$ where

$$\text{Phase} = \text{P} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

The Gottesman-Knill Theorem shows that M generates only a discrete subset of unitary transformations and any circuit made of gates in M and applied to the initial state $|0...0\rangle$ can be stimulated in polynomial time by a classical Turing Machine.

Whereas these are all ways in which a set of gates can fail to be universal, the Solovay-Kitaev theorem gives insight into an essential property of universal gates. It states that any set of single-qubit quantum gates that generates a dense subset of the group $SU(2) = \{\begin{bmatrix} \alpha & -\bar{\beta} \\ \beta & \bar{\alpha} \end{bmatrix}\} : \alpha, \beta \in \mathbb{C}, |$ α $|^2 +$ | β $|^2 = 1\}$ is such that short sequences of gates from the generating set will approximate any other quantum gate well.

Importantly, small finite sets of quantum gates which each operate on up to three qubits at a time are sufficient to implement any quantum algorithm.

3. Quantum Computation

In this section, we will provide some important definitions and develop some intuitions that are required to understand the proof of $BPP \subseteq BQP$.

3.1 Definition: Quantum Computation

As explained in the previous section, any unitary matrix can be applied to the superposition of a m-qubit vector as a valid quantum operation. However, in the model we will describe here, we will only consider elementary operations, otherwise known as *quantum gates*.

Quantum computation: Let $f : \{0,1\}^* \to \{0,1\}$ and $T : \mathbb{N} \to \mathbb{N}$ be some functions. We say that f is computable in quantum $T(n)$-time if there is a polynomial-time classical TM that on input $(1^n, 1^{T(n)})$ for any $n \in \mathbb{N}$ outputs the descriptions of quantum gates F_1,...,F_T such that for every $x \in \{0,1\}^n$, we can compute $f(x)$ by the following process with probability at least $\frac{2}{3}$:

1. Initialize an m qubit quantum register to the state $|x0^{n-m}\rangle$, i.e. x padded with zeros, where $m \leq T(n)$.
2. Apply one after the other $T(n)$ elementary quantum operations F_1,...,F_T to the register.
3. Measure the register and let Y denote the obtained value.
4. Output Y_1, the first bit in Y, which is a m-qubit register.

3.2 Definition: BQP

In this section, we will discuss the complexity class of "Bounded-Error Quantum Polynomial-Time" decision problems, BQP.

BQP: A boolean function $f : \{0,1\}^* \to \{0,1\}$ is in BQP if there is some polynomial $p : \mathbb{N} \to \mathbb{N}$ such that f is computable in quantum $p(n)$-time.

It can be helpful to think of BQP as the class of decision problems that can be efficiently computed on a quantum computer. To put it informally, a language L would be in BQP if there exists a polynomial time Turing machine that is able to generate a polynomial number of quantum gates, or elementary quantum operation matrices, that will compute the algorithm on a m-qubit register.

It is rather unintuitive, but it is important to note that the power of quantum complexity does not stem from the representation of the m-qubit register in 2^m bits, operated on by matrices of size $2^m \times 2^m$. The much more interesting property of m-qubit registers is that the probabilities in their superpositions vectors are allowed to be negative, as it is their $L2$ norms that are preserved by quantum operations.

3.3 Definition: Quantum circuits

Though the polynomial Turing machine definition is helpful in terms of framing quantum complexity classes in terms of traditional complexity classes, it can be easier to reason about quantum computing in terms of quantum circuits.

Quantum circuits are acyclic graphs with inputs, outputs, and internal gates. The inputs are vertices of in-degree zero and the outputs are vertices with out-degree zero. The overall structure of a quantum circuit is very similar to a boolean circuit. There are, however, major differences regarding the kinds of gates that are allowed in a quantum circuit compared to a boolean circuit. As previously described, our quantum computation model allows for only elementary operations, which means that our quantum functions will be matrices of size 2×2, 4×4, or 8×8. Thus, the quantum gates in our circuit will be matrices of these sizes, as opposed to the traditional AND, OR, and NOT gates of boolean circuits. It follows from this that the in-degree of any gate is at most 3.

3.4 BPP $\subseteq$ BQP

We will begin with a quick refresher of the complexity class BPP. The class BPP stands for "Bounded-error Probabilistic Polynomial" problems. Intuitively, BPP is the class of decision problems that can be probabilistically computed efficiently, with some margin of error. The class BPP is important because it contains all of the problems that we consider to be feasibly computable in a classical computation model.

With the previous definition of quantum circuits, we see that it might be possible to efficiently simulate any classical circuit with a quantum circuit. If this is the case, then we will have shown that all classical computation can be efficiently simulated with quantum operators.

Lemma 10.10 (Arora-Barak): If $f : \{0,1\}^n \to \{0,1\}^m$ is computable by a Boolean circuit of size S, then there is a sequence of $2S + m + n$ quantum operations computing the mapping of $|x\rangle\,|0^{2m+S}\rangle \to |x\rangle\,|f(x)\,|0^{S+m}\rangle\rangle$ [3]

Though we will not go into a full proof of this lemma here, the general idea is that we can replace the Boolean gates in C with their quantum counterparts. As previously described in Section 2, each Boolean gate (AND, OR, and NOT) has a quantum analog.

It immediately follows from this lemma that $BPP \subseteq BQP$. Recall that a Hadamard gate transforms the qubit state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$. When observed, the state collapses to $|0\rangle$ with probability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$. While this implies an algorithm that involves intermediate measurements, it can be shown that any algorithm with intermediate measurement can be converted with minimal overhead to an algorithm with measurement only at the end [3]. Thus, it makes sense that all of the randomized bits required by a BPP algorithm can be generated using Hadamard gates.

4. Shor's Algorithm

The factoring problem asks: Given some integer N, what are N's prime factors?

Factoring is in NP, but is not known to be NP-complete. One of the hardest cases of the factoring problem is when N is the product of two primes, where the primes are roughly equal in size. Today, the security of the Internet relies on factoring being a hard problem, which we will discuss in a later section.

4.1 Shor's

Shor's Algorithm is an algorithm for integer factorization, famous for its repercussions on public key cryptography. The solution takes advantage of quantum superposition and some number-theoretic facts.

Essentially, Shor's Algorithm takes advantage of a simple reduction from finding the prime factors of n to finding the order of a random element x in the multiplicative group mod$n$, given a few constraints on x, so that Shor's algorithm is also a randomized algorithm.

Let x be an element of $\mathbb{Z}_n^*$ such that $x^r \equiv 1 \bmod n$ and x is nontrivial, i.e. $x \neq 1$ and $x \neq n-1$. Suppose, furthermore, that r is even so that $y = x^{r/2}$ and $y^2 \equiv 1 \bmod n$. Then $(y+1)(y-1) \equiv 0 \bmod n$, i.e. $(y+1)(y-1) = kn$. Then $gcd(y+1, n)$ or $gcd(y+1, n)$, which can be computed by the Euclidean algorithm, will yield a prime factor of n. (We can then divide by the factor and repeat the algorithm at most $\sqrt{n}$ times for the full factorization.)

It can be shown that with high probability, r will be even for a nontrivial x. As we have reduced factorization to order-finding of x in $\mathbb{Z}_n^*$ let us now discuss how to order-find in polynomial time.

Integral to this process is the Quantum Fourier Transform. For a such that $0 < a \leq q$ the QFT performs the mapping

$$|a\rangle \mapsto \frac{1}{q^{1/2}} \sum_{c=0}^{q-1} |c\rangle \exp(2\pi iac/q)$$

I.e. it performs a linear mapping using the unitary matrix which has (a, c) entry as $\frac{1}{q^{1/2}} \exp(2\pi iac/q)$. Let us denote the matrix as Q.

Given x, n let us first find q, the power of 2 such that $n^2 \leq q < 2n^2$.

Then, use q Hadamard operations on the first part of the register so that it is in state

$$\frac{1}{q^{1/2}} \textstyle\sum_{a=0}^{q-1} |a\rangle |0\rangle$$

Then compute $x^a \bmod n$ in the second register, using repeated squaring, so that the machine is now in the state

$$\frac{1}{q^{1/2}} \textstyle\sum_{a=0}^{q-1} |a\rangle |x^a \bmod n\rangle$$

Now, apply QFT operation Q on the first part of the register, such that the machine is now in state

$$\frac{1}{q} \textstyle\sum_{a=0}^{q-1} \sum_{c=0}^{q-1} \exp(2\pi iac/q) |c\rangle |x^a \mod n\rangle$$

At this point, we perform an observation, which lands us in some state $|c, x^k \bmod n\rangle$. Because of the modular arithmetic in the states, there are multiple paths to a given state $|c, x^k \bmod n\rangle$. For a fixed c, k the probability of observing it is the square of the summed amplitudes of all states where $x^a \equiv x^k \bmod n$, i.e.

$$| \frac{1}{q} \textstyle\sum_{a:x^a \equiv x^k} \exp(2\pi iac/q) |^2$$

After bounding the range of a and turning the sum into an integral, it can be shown that the probability of seeking some state $|c, x^k \bmod n\rangle$ is greater than $\frac{1}{3r^2}$. With high probability, the obesrved value of c is near some integral multiple of q/r. If d/r is in lowest terms and $gcd(d, r) = 1$ then we have r. The number of states $|c, x^k \bmod n\rangle$ which allow us to compute r as such is the number of relatively prime d to r, given by $\phi(r)$, Euler's totient function. Furthermore, there are r choices of x^k, as r is the order of x in $\mathbb{Z}_n^*$. Each of these $r\phi(r)$ choices occur with probability $\frac{1}{3^r}$, so we can find r with probability $\phi(r)/3r$. Since $\phi(r)/r > \delta/\log\log r$ (Hardy and Wright 1979), repeating $O(\log\log r)$ times gives us r with constant probability δ.

The time complexity of Shor's algorithm is bottlenecked at the exponentiation of x^k and the application of the QFT transform Q. With a special version of repeated squaring (Schönhage-Strassen algorithm), we can compute the powers of $x \bmod n$ in $poly(\log n)$ time. And although the matrix Q is size $2^{poly(n)}$, with a bit of work [8], the QFT can also be performed in $poly(\log n)$ time.

While incomplete in details, the structure of Shor's algorithm was included to show the reliance on superposition and interference in obtaining r. Here we relied on the periodicity of $x^k \bmod n$, which creates multiple copies of a given state.

4.2 Relationship to cryptography

RSA

Public key cryptography is a means of communication between two parties where the distribution of secret keys is asymmetric. If a party Bob wishes to communicate with a party Alice, we assume that Alice has run a key-generation function which produces a public key and a secret key, both specific to her. The public key e is published for others to use to encrypt their messages to her. The secret key d is kept by Alice as the sole means to decrypt messages encrypted with e.

Currently, most of public key cryptography is implemented using RSA key generation, which relies on the difficulty of factoring an integer n, the product of two primes. Below we include the RSA key generation pseudocode [5] to illustrate how one would use integer factorization to crack RSA secrets.

RSAGen(l, e) :=

generate a random l-bit prime p such that $\gcd(e, p-1) = 1$
generate a random l-bit prime q such that $\gcd(e, q-1) = 1$ and $q \neq p$
$n \leftarrow pq$
$d \leftarrow e^{-1} \bmod (p-1)(q-1)$
output (n, d)

If a malicious party is able to factor n, and obtain p, q, then it can also derive d as $e^{-1} \bmod (p-1)(q-1)$ using the extended Euclidean algorithm. Then, it is able to decrypt any messages sent to Alice.

Discrete log

Other forms of cryptography, such as signatures or zero knowledge proofs, are based on the Diffie Hellman assumption. The most basic form of the Diffie Hellman assumption is that it is difficult to compute a given a generator x of a cyclic group ($\mathbb{Z}_p^*$ where p is prime) and x^a (this challenge is also known as the discrete log problem). One may have noticed the connection to order finding, and indeed, a simple modification to Shor's algorithm, which also uses the Quantum Fourier Transform, can be used to solve for the secret a [8].

5. Quantum Complexity Hierarchy

Let's start by reviewing the hierarchy of complexity classes that we are familiar with:

$$L \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$$

We recall that L is the class of decision problems that can be solved in logarithmic space, P is the class of problems that can be solved in polynomial time, NP is the class of problems that can be solved in non-deterministic polynomial time.

5.1 BQP and its relation to classical complexity classes

Hopefully it is clear from Shor's algorithm that quantum computing can provide at least an exponential speed up on classical computation. This begs the question – how much speed up is possible from quantum computing? Is there an upper bound on how powerful quantum computing is compared to classical computing?

Yes! In fact, theorists have shown that $BQP \subseteq EXP$. Recall that an m-qubit vector is represented as a superposition vector of max size 2^m, which is transformed by a series of unitary matrices of size $2^m \times 2^m$ (or smaller for elementary operations). It makes sense that a classical computational model can simulate this behavior with at most an exponential speedup. Thus, we know that BQP provides at most an exponential speed up over classical computing.

Recall that $PSPACE$ is the class of problems that can be computed using only polynomial space. At first glance, it would appear that BQP is not contained within $PSPACE$; after all, the superposition vector of an m-qubit vector requires space of 2^m. However, Bernstein and Vazirani proved that, in fact, $BQP \subseteq PSPACE$. We will not go into a full explanation of the proof; however, it is important to note that the proof relies on the reuse of space between each step in the quantum computation, and requires computing over only the relevant strings in $\{0,1\}^m$ (pruned down according to which 3 bits the current elementary quantum operation is acting upon), instead of writing out the entire quantum superposition.

The tightest upper bound we have on BQP is the complexity class PP. A language $L \in PP$, the class of languages solvable in probabilistic polynomial-time, if there exists a polynomial-time nondeterministic Turing machine M for which an input $x \in L$ if $M(x)$ has more accepting paths than rejecting paths. Adleman, DeMarrais, and Huang famously proved $BQP \subseteq PP$. The proof has been left as an exercise for the reader.

It is not known whether $BPP = BQP$. Because mathematicians and theorists have tirelessly sought, but not yet found, an efficient algorithm for the integer factorization problem, Shor's algorithm provides strong evidence that, in fact, $BQP \neq BPP$.

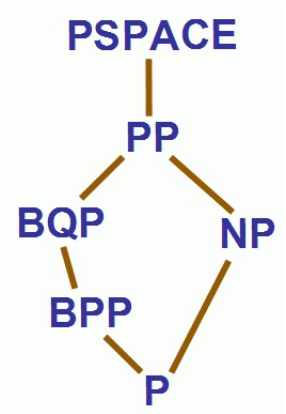

Figure 1: BQP and the classical complexity classes

5.2 Classic misconceptions

Many modern explanations of quantum computing involve some kind of reference to quantum computations being able to "explore all possible paths in parallel", citing this property as the crux of quantum computing's exponential speed up in algorithms such as Shor's algorithm. While this is a compelling statement, it is somewhat misleading when trying to compare BQP to NP.

Often people describe the class NP, as well as non-deterministic Turing machines, as magically being able to try all decision paths and outputting 1 if one path reaches an accepting state. With this in mind, it's tempting to conclude that NP problems should see a significant speed up in

the quantum world. This, however, is not true. There have not been any NP-complete problems proven to be in BQP. The NP problems which have seen significant speedup have all exploited unique mathematical structures within the problem, rather than using a brute force attack on the problem. It is not currently known whether $NP \subseteq BQP$.

6. Feasibility and Philosophical Implications of Quantum Computing

Quantum mechanics opens up the ability to hold and operate on vast amounts of information in a relatively small number of bits. Whereas a classical computer with s bits holds s bits of information, a quantum computer with s qubits can keep track of 2^s states at once.

As previously mentioned, one point to be wary of is the misconception that quantum computing can be leveraged to compute an exponential number of paths in parallel. The reason why this doesn't work is because measurement at the end of a quantum algorithm collapses the state into a single one out of the exponential states, losing information about all other states.

However, researchers are curious about whether other properties of quantum mechanics can be used to solve NP-complete problems. (While Shor's algorithm is an efficient quantum solution to integer factorization, it is not known whether integer factorization is NP-complete.) Specifically, researchers have posed the question of whether the quantum approach which treats the underlying problem as a black box would be able to solve NP-complete problems. One can think of this as searching over an exponential space of possible solutions–or guessing a solution and verifying its correctness.

Grover's algorithm provides a $\sqrt{n}$ speedup, but this only reduces the exponent in the search run-time. Because of this limitation in the ability of quantum computing, it is conjectured that quantum mechanics alone cannot solve NP-complete problems, and that any quantum algorithm to solve an NP-complete problem must leverage the problem structure.

6.1 Implementations

Though much of the research in quantum computing is still purely theoretical, there are already individuals with functioning quantum machines that can simulate these algorithms. In the research sector, physical implementations of Shor's algorithm have been able to factor numbers up to the number 56,153 [13]. Though these current implementations do not pose a threat against RSA, which uses numbers which are the products of very large primes, the fact remains that significant progress is being made in these fields. Private section companies are growing increasingly interested in quantum computing. One of these companies, D-Wave purports to have the "first real-time Quantum computing environment"[10]. They claim to they have been doubling the number of qubits handled in their operations every year for the last ten years [12]. This rate of progress is incredibly reminiscent of Moore's law, which suggested that number of transistors on a chip would double every two years.

The cryptography community has been actively trying to get ahead of the problems that will arise if efficient and scalable quantum computing machines are built. Cloudflare, a networking company that handles massive amounts of Internet traffic including DNS and encrypted content delivery for third parties, has open-sourced a TLS v1.3 library that uses a quantum-resistant encryption scheme [9]. (For those not familiar with TLS: TLS is a networking protocol named Transport Layer Security, which is responsible for the encryption of HTTPS network traffic.)

6.2 Church-Turing Thesis

The Church-Turing thesis states that "every physically realizable computation device—whether it's based on silicon, DNA, neurons or some other alien technology— can be simulated by a Turing machine" [3]. As it stands, quantum computing do not challenge this thesis, as it is indeed possible to simulate the efficient quantum algorithms with a Turing machine ($BQP \subseteq PSPACE$).

However, some computer science theorists have grown to believe in the *strong* Church-Turing thesis, which states that "every physically realizable computation model can be simulated by a TM *with polynomial overhead*" [3]. It is not currently known whether quantum algorithms can be efficiently simulated by a Turing machine, that is, that they can be simulated with at most a polynomial slow down. If it is indeed proven that the reduction from a quantum algorithm to its classical counterpart is lower bounded by $\Omega(poly(n))$, theorists may need to expand their beliefs on what problems we consider possible to compute.

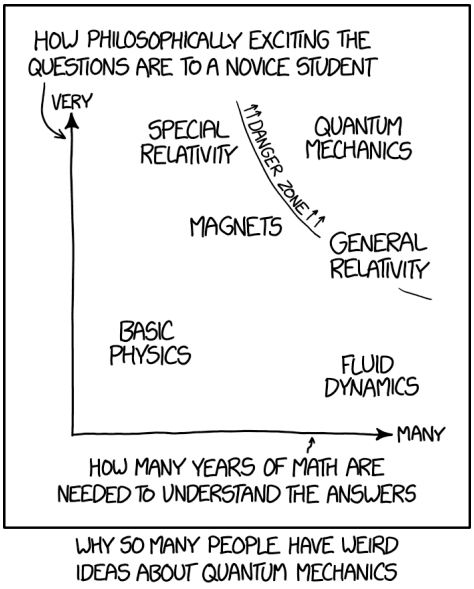

Figure 2: An accurate analysis of our experience writing this paper on quantum complexity [11]

References

[1] Aaronson, Scott. "PHYS771 Lecture 10: Quantum Computing." PHYS771 Quantum Computing Since Democritus, 2006, www.scottaaronson.com/democritus/.

[2] Aaronson, Scott. "Shor, I'll Do It." ShtetlOptimized, 2007, www.scottaaronson.com/blog/?p=208.

[3] Arora, Sanjeev and Boaz Barak, "Ch 10. Quantum Computation." Computational Complexity A Modern Approach, Cambridge University Press, 2016, pp. 201–232.

[4] Bernstein and U. Vazirani (1993), Quantum complexity theory, in Proc. 25th Annual ACM Symposium on Theory of Computing, ACM, New York, pp. 11–20; SIAM J. Comput., 26 (1997), pp. 1411–1473.

[5] Boneh and Shoup, A Graduate Course in Applied Cryptography, 2017. pp. 522-525.

[6] Feynman (1982), Simulating physics with computers, Internat. J. Theoret. Phys., 21, pp. 467–488

[7] Politi, A., Matthews, J. C. F. O'Brien, J. L. Shor's quantum factoring algorithm on a photonic chip. Science 325, 1221 (2009)

[8] Shor, P. W. Polynomial-Time Algorithm for Prime Factorization and Discrete Logarithms on a Quantum Computer, p. 124 in Proceedings of the 35th Annual Symposium on the Foundations of Computer Science, ed. S. Goldwasser (IEEE Computer Society Press, Los Alamitos, CA, 1994); SIAM J. Computing 26, 1484 (1997); quant-ph/9508027.

[9] Valence, Henry de. "SIDH in Go for Quantum-Resistant TLS 1.3." The Cloudflare Blog, The Cloudflare Blog, 29 Aug. 2018, blog.cloudflare.com/sidh-go/.

[10] D Wave Systems, www.dwavesys.com/take-leap.

[11] https://xkcd.com/1861/

[12] "DWave Systems.", www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware.

[13] Zyga, Lisa (28 November 2014). "New largest number factored on a quantum device is 56,153". Phys.org. Science X Network. Retrieved 4 August 2015.