The Broad State of Quantum Computing
A Computer Scientist's Perspective
CIS 407

Mateo Minato

June 4, 2020

University of Oregon
Instructor: Jee Choi

1 Abstract

Quantum computation is one of the hottest topics in modern science. Earlier this year, Google announced that they had, for the first time, achieved Quantum Supremacy; the quantum solving of a problem infeasible on classical computers. Perhaps unsurprisingly, the development of these quantum computers has been driven largely by physicists. As quantum computation becomes more and more feasible, computer scientists must begin to make an effort to understand at least the broad strokes. Here, I discuss the general state of quantum computation, a basic introduction to the principles behind quantum mechanics, and the importance of quantum development for the future of computation.

2 Background

2.1 Benefits of Quantum Computing

It is a common misconception that quantum computing is an avenue to replacing classical computation. In truth, the benefits of quantum computation are specific, theoretical, and, to some degree, unknown. Computational complexity theory refers to the set of problems a quantum computer can solve in polynomial time with an error probability no higher than 1/3 as BQP (analogous to BPP in classical computation). BQP does not overlap entirely with BPP–not all problems that are classically polynomial are also quantum mechanically polynomial, and vice-versa. While the exact scope of BQP is unknown, there are many well-known problems that have been shown to be in BQP and not in BPP. We'll discuss the exact nature of these problems in a moment–what's important to note now is that there exist problems which quantum computers can quickly solve that classical computers could not solve in a feasible amount of time. This is otherwise known as Quantum Supremacy, and was demonstrated by Google for the first time earlier this year.

While the problems that a QC can solve efficiently can't be specifically defined, the most common and impactful problems can be broadly described according to our knowledge of what precisely makes a quantum computer different. That is, the following problems are some examples of problems much better suited to a quantum computation.

- Large N Factoring; $O(N^2)$, roughly.
- Item with property X selection; $O(N/2)$ classically, but $O(\sqrt{N})$ quantum mechanically.
- Quantum Simulation
- Many or most NP-Complete problems; as a direct result of reversibility.

2.2 A (Very Brief) Introduction to Quantum Mechanics

So what is it, specifically, that makes quantum computers so different from classical computers? To answer that question, we need (at least) a cursory understanding of quantum mechanics. Quantum mechanics provides a new form of parallelism. While classical computations can also perform computations in parallel, the result will be the same (albeit derived more efficiently) as if the computation was performed serially. This is not true for quantum computers; parallelism is intrinsic to linear superposition, or the foundation of a quantum state.

Qubits can exist in a probabilistic superposition of 0's and 1's, as opposed to classical bits which have a decided and discrete value. A qubit in superposition then is essentially both 0 AND 1, with both 0 and 1 states having a complex coefficient corresponding to the probability of the qubit being in that state upon observation. The state of a qubit can therefore be described as a vector in the two dimensional complex space.

It's also worth nothing that this superposition is inherent, NOT just a quirk of our measurement or observation. In the context of the famous beam splitter experiment, the photon must actually exist or have travelled (in some sense) both possible paths, not simple picked one or the other. This is similarly observable with light interference patterns. (Sidenote: it's not worth thinking too much about why this is true or what is happening physically at the atomic/sub-atomic level, because the answer to that question is dependent on interpretation, and is really more a matter of philosophy).

Our understanding of correlation of events must also be adjusted to accommodate quantum mechanics, and specifically as a result of superposition. Quantum affected objects can also be 100% correlated, in that if one has decayed, the other nuclei will have decayed as well. One cannot decay without the other. However, since these nuclei can exist in a superposition of decayed and not-decayed, our correlation between the two must accommodate that as well; essentially, 100% correlation must include correlation of the states comprised through superposition. This is colloquially known as entanglement, and was originally postulated by Schrodinger in his famous feline experiment.

This postulate in turn led to the infamous EPR paradox, which implied that two antiparallel electrons shot off a decaying Helium atom in opposite directions must have some instantaneous knowledge of each others state (if one is observed, the superposition collapses and the other electron MUST have the corresponding opposite spin of the observed electron). This bothered Einstein because it seemed to violate relativity; information traveling at faster than the speed of light. The EPR paradox shows us that our understanding of either relativity or QM is incomplete, and there are many theories that attempt to resolve this inconsistency; however, it's clear through observation that this collapse does in fact happen instantaneously, which is one of the primary reasons quantum mechanics is capable of providing enormously enhanced computing power. (Sidenote: entanglement is notable specifically because the particles state IS in superposition until observed; it's not as simple as each particle had definitive spin the whole time. Instead, the observation itself causes superposition collapse forcing the other particle into a particular state. In QM, it's useful to think as observation not as passive but as active).

The theory that observation CANNOT impact outcome, and instead state is related to some hidden variable is provably false through measurement. That is, outcomes of quantum mechanical experiments are commonly observed that violate Bell's inequality, which is itself a consequence of this hidden variable theory. The distinction/proof can be subtle, so we'll avoid the details here, but what's important to understand is that the hidden variable theory has been repeatedly shown to be experimentally non-viable.

Quantum Teleportation is a theoretical possibility based on entangled states, although it has not been accomplished in a lab at faster than the speed of light because the method described by Bennet et all requires information to be transmitted through conventional means. This is still notable as it shows that, as a result of entangled states, an EXACT replica can be produced–Something that is impossible otherwise, due to Heisenberg's Uncertainty Principle (it would be impossible to even gather enough information to generate a perfect replica).

One inescapable form of heat generation in computers stems from thermodynamics; anytime information is irreversibly erased (such as resetting a bit or series of bits from any value to 0), the entropy of the system increases, and heat correspondingly increases. Quantum Computers are reversible, unlike classical computers, so they can theoretically operate without generating heat. The reversibility of quantum computers stems from design; all irreversible gates are disposed off, and only reversible gates are employed. Indeed, as proven by Charles Bennet, quantum computers are capable of performing any computation using only reversible gates.

2.3 Physical Implementations

There are many current iterations of real, functioning quantum computers. I'll provide a brief overview of some of the more notable efforts.

- **Ion Trap Computers** are a common realization of quantum computers. They encode information via the energy states of ions and in the vibrational modes between those ions. This allows Fourier transforms to be evaluated, which, in turn, leads to Shor's factoring algorithm.
- **Quantum Electrodynamics Cavity Computers** consists of a cavity filled with Cesium atoms, and uses lasers, phase shift detectors, polarizers, and mirrors to generate a truly quantum computer capable of creating, manipulating, and preserving superposition and entanglement.
- **NMR Computers** use molecules in liquid and perform operations by sending radio pulses to the sample and reading its response, where each qubit is implemented as a spin state of the nucleus of atoms comprising the molecules. This form of QC allows solving of many NP-complete problems in polynomial time. This form of QC has led to most current practical accomplishments in quantum computing.
- **Quantum Dot Computers** are a simpler implementation, comprised of an array of quantum dots each connected to their nearest neighbor via gated tunneling barriers. The advantage here is that the quantum dots are controlled electronically; the disadvantage is that they can only communicate with their nearest neighbor, resulting in complex data readout.

Other implementations include Josephson Junctions, The Kane Computer, and Topological Quantum Computers, all which are promising candidates for future implementations of quantum computing, despite being in various stages of theory vs. implementation.

2.4 Quantum Computing Simulators

Quantum computers are currently severely limited by the expense and difficulty of their realization. Physically, most designs require the QC to operate at temperatures nearing absolute zero. The largest functioning Quantum Computers in existence hold at around 50-60 qubits, and attempts to build larger quantum computers have proved enormously difficult. It is of great interest, then, for computer scientists and physicists both to build effective quantum simulators, so that quantum computing specific algorithms and methodologies can be studied, tested, and tweaked. In some sense, this is analogous to the development of classical computing–inevitably; when the physical limitations of implementation are extreme, theory leads realization. But building a quantum computer simulator on a classical system is no easy task.

The state of an L-qubit quantum computer can be represented by a complex valued vector of length $D = 2^L$.[1] If we limit the bit size of our floating point arithmetic to 13-15 digits, we need at least 2^{L+4} bytes to represent a state of the quantum system of L qubits. As you can see, this leads to exponential increase in memory requirements. For example, to simulate a 24 qubit QC requires 256 MB of memory, and a 36 qubit QC requires 1 TB, to store a single arbitrary state. As you can see, this simulation is already becoming extremely expensive.

Operationally, performing an update on qubit j requires an update of 2^L elements of the state.[1] A single qubit phase shift is an exception, requiring (only) 2^{L-1} updates. It is worth noting that all of these operations can be done in place. Performing 2-qubit operations, such as the CNOT, require 2^{L-1}, while 3-qubit gates require 2^{L-2} amplitude updates.

In practice, simulating an L-qubit QC requires $N = 2^L/2^M$ MPI processes, where M is the number of qubits for which we're tracking the basis states. Upon implementation of this algorithm in Fortran 90, researchers found near-perfect scaling with the number of CPU's and problem size, indicating that this strategy of massive parallelization may find application in the future realization of quantum simulation.[1]

An accurate simulation must also account for the error probability and decoherence inherent to quantum mechanics. To do so artificially (classically) adds even more computational complexity. In 2002, researchers found that they could include a reasonable error model by representing inaccuracies as small deviations in the angles of rotation that comprise all single qubit gates. [2] Using this methodology, and a similarly parallelized approach, they were able to simulate 30-qubit quantum register states. They were also able to implement Shor's Algorithm to factor the smallest viable integer, 15. They found their efforts were slowed dramatically by modular exponentiation, and as a result noted that the effectiveness of scaling this method of simulation to larger integers required further experimentation.

3 Quantum Algorithms

3.1 Shor's Algorithm

Shor's Algorithm for large N factoring is one of, if not the, most seminal quantum computing algorithms yet devised. It originally garnered attention in 1997 for its promise to break RSA encryption, an encryption standard that relies on the difficulty of large N factoring, and that is still widely used today. Peter Shor postulated that his algorithm would allow a quantum computer to factor large numbers.[3] Below, I attempt to walk through the algorithm in broad strokes, referencing the original study as well as a followup conducted by John Cooper in 2007.[4] This algorithm would be more technically discussed through the use of quantum notation and superposition, but by treating the known quantum operations as black-box operations, we can make the algorithm more digestible to non-physicists.

If A, B are integers, $A^p = m * B + 1$ for some integers p and m. Assuming N is the number, we can take any integer guess g as to a factor of N (in fact, all g needs to do is SHARE a factor with N, as Euclid's algorithm allows us to quickly find shared factors). We can then transform this equation into:

$$g^p = m * N + 1$$
$$g^p - 1 = m * N$$
$$(g^{p/2} + 1)(g^{p/2} - 1) = m * N$$

$(g^{p/2} + 1)(g^{p/2} - 1)$ are now much, much better guesses than our original, arbitrary guess g. Its worth repeating that a core advantage to this algorithm is a direct result of the efficiency of Euclid's algorithm.

However, if one of these two factors is itself a multiple of N, the other is a factor of m, and our guesses are useless. Likewise, if p is odd, $p/2$ is correspondingly fractional, and our guesses are similarly useless. Luckily, it turns out that 37.5% of the time, for any random guess g, neither of these problems occur. This means that we can successfully factor large N in less than 10 guesses 99.9+% of the time.[3, 4]

The most glaring problem, classically, is finding our exponential term p. This problem is extremely resource demanding on a classical computer (more demanding, in most cases, then just factoring N by brute force). Quantum computers solve this problem by taking a superposition of possible values x for p as input, taking our random guess g to the power of p for each element of the superposition, and then finding how much bigger that that result (which, as always, is represented by a superposition) is than a multiple of x (this is essentially a modulus operation). We are looking for a value x that gives us a remainder of 1. If we tried to directly measure this superposition output, we'd get a random value as output due to superposition collapse.[3]

It's also a mathematical fact that if $g^x = m*N+r$, then $g^{x+p} = m*N+r$. This is known as the repeating property, and works for any multiple of p. If we measure any random value of the output superposition described above (assuming the input superposition included all possible values), the superposition will collapse into a superposition of ONLY those values $|x, +r>$. Notably, each of these values x will be separated by value p.[3, 4]

We now have a superposition series of inputs separated by frequency $1/p$. We can pass this superposition through a quantum fourier transform, which will output a superposition of superpositions. The resulting sin waves will destructively interfere, giving you a state of $|1/p>$, which you can subsequently invert to find p. Assuming p is even and $g^{p/2} \pm 1$ is not an exact multiple of N, we are guaranteed to have a number that shares factors with N, which, through Euclid's algorithm, is good enough to find factors for large N, and we've done it in polynomial time.[3, 4]

In practice, this is harder than it is in theory because of the current size limitations of quantum computers. It was shown that this method is effaceable on actual quantum computers with the smallest number for which this method actually works, $N = 15$, and the largest number factored using Shor's algorithm on a quantum computer to date is 21. However, the limitations here are physical, and not algorithmic, as discussed in the previous section. The method shows promise for larger quantum computing systems, if and when they become feasible.[4]

3.2 Solving Linear Systems of Equations

More complex algorithms exist for more mathematically challenging problems. Algorithms for solving linear systems of equations is an example. I won't go into as much detail here, but researchers from the University of Bristol proposed the following general methodology in 2009.[5]

- Represent b as a quantum superposition, $|b>$
- Apply the exponential e^{iAt} to b (Hamiltonian simulation) for a superposition of times
- Decompose $|b>$ in the eigenbasis of A and find eigenvalues, λ_j
- Linearly map $|\lambda_j>$ to $C\lambda_j - 1|\lambda_j>$ (C is for normalization). This operation is not unitary, and therefore has some probability of failing.
- Uncompute $|\lambda_j>$ register, and resulting state is proportional to $|x>$

This method only provides notable speedup when A is easily invertible, or when A is not ill-conditioned. To determine all the components of x would require N measurements, but that is a less common use case than using x and x^T to determine some expectation value. A major benefit of the algorithm is that it only needs $O(logN)$ qubit registers, and never needs to write down A, x, or b.

This algorithm takes exponentially less time to compute in its entirety than its classical equivalents take to even write down the output. It is semi-related to the Monte Carlo algorithm in classical computing, in that it uses probabilistic distribution.[5]

The study demonstrates that matrix inversion is a BQP-complete problem, which is of importance to the efficiency of this algorithm. The study also details a wide array of potential ways to relax current requirements, such as not requiring A to be sparse 100% of the time (some other cases are computable), or relaxing the authors initial assumptions regarding the condition number of A.[5]

4 Conclusions

As a direct result of physical limitations, the future of quantum computing is largely influx. As more and more theoretical benefits have been postulated, established, or proven, the interest in pursuing feasible quantum computation has grown exponentially. In recent years, development of more powerful classical computers has been severely limited or even capped by the now atomic scale of transistor development. In fact, the development of classical computers is being limited by the laws of quantum mechanics, particularly quantum tunneling. This has led to further interest in the development of quantum computers as a manner of solving problems that are ill-suited for classical computation to begin with.

As many things titled quantum, there is a tremendous amount of misunderstanding when it comes to the role of quantum computers in our future. Although the set of BQP problems is as of yet nebulous, its clear that the abilities of quantum computers fall short for many use-cases. Its also clear that BQP encompasses problems that are infeasible on classical computers, and some scientists speculate that problems such as protein folding, an all-purpose cure for cancer, could be contained within this set. It seems, then, that the role of quantum computers in our future, whether direct or indirect, will be one of augmentation, and not of substitution. As companies such as Google and IBM continue to invest millions in the development of bigger and better quantum computers, the distinction between the two types of computation will become more pronounced, and the exact nature of the future of computation will become clear.

In the meantime, it is important to understand that the process of quantum computation is so inherently different than that of classical computation as-to-be almost entirely foreign to most computer scientists (at least, those without backgrounds in quantum mechanics). As the field grows, many companies and institutions believe it is important to begin implementing some form of quantum mechanical education to computer science students, through the use of simulation or cloud-based access to small quantum computers. It is my hope that this education continues and the interest in the field becomes more general. The realization of powerful, small, accessible quantum computers will likely occur in our lifetime–and the sooner computer scientists begin to understand them, the better.

References

[1] K. Raedt, K. Michielsen, H. Raedt, B. Trieu, G. Arnold, M. Richter, Th Lippert, H. Watanabe, and N. Ito. Massive parallel quantum computer simulator. *Computer Physics Communications*, 176:121–136, 01 2007.

[2] Jumpei Niwa, Keiji Matsumoto, and Hiroshi Imai. General-purpose parallel simulator for quantum computing. *Physical Review A*, 66, 01 2002.

[3] Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26:1484–1509, 10 1997.

[4] John Cooper. A re-evaluation of shor's algorithm. 01 2007.

[5] Aram Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103:150502, 10 2009.