Solving Hard Scientific Problems Using Quantum Computers

May 16, 2020

Steven Oud
Software for Science
Amsterdam University of Applied Sciences
steven.oud@hva.nl

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo.

1 Introduction

The idea of a quantum computer was first proposed by Richard Feynman in 1982 [15]. Feynman suggested we use a computer that made use quantum mechanical phenomena to solve problems in physics and chemistry intractable by classical computers. He remarked “If you want to make a simulation of nature, you’d better make it quantum mechanical, and by golly it’s a wonderful problem, because it doesn’t look so easy.” In 1994, Peter Shor introduced Shor’s algorithm: a quantum algorithm for factoring integers and computing discrete logarithms that was exponentially faster than any known classical algorithm [35].[1] The discovery of such a speedup, among others, resulted in the field of quantum computation and information. Since then, the field of quantum computation and information has made significant progress with applications in the fields of chemistry [28], cryptography [8], and machine learning [31].

Quantum computers are no longer just a theory, like they were when proposed back in 1982. Companies such as Google and IBM are building actual quantum computers [17, 23]. These are very small and error-prone quantum computers, but quantum computers nonetheless. Google recently demonstrated “quantum supremacy" with their Sycamore quantum processor [5]. This means they have experimentally proven that a quantum computer can do some computation efficiently that classical computers cannot. This is an important milestone to reach to demonstrate the possible power of quantum computers and stimulate future research.

This report will look at how quantum computers can be used to solve hard scientific problems that classical computers cannot. It is structured as follows. First, in Section 2, we will give a high level overview of quantum computation and information. Second, in Section 3, computational complexity and how quantum algorithms relate to it is discussed. Third, in Section 4, we take a look at promising applications for quantum computing in science. Finally, we conclude the results of the research in Section 5.

[1] Note that this does not mean that there is no similarly efficient classical algorithm that we just have not found yet.

2 Quantum Computation and Information

Quantum computers take advantage of quantum mechanical effects such as superposition and interference to solve certain problems faster than classical computers. Instead of bits, which can be in states 0 or 1, they use quantum bits, or qubits, that can exist in superpositions of states. In a superposition of states, each state has a number associated with it which we call its amplitude. A qubit has some amplitude for being 0 and another amplitude for being 1. When we look at the qubit, it randomly collapses to either 0 or 1 and its superposition is lost.

In everyday life we talk about probabilities ranging from 0 to 100 percent. It makes no sense to say there is a negative 50 percent chance of rain tomorrow. However, in the quantum realm we find that elementary particles such as electrons and photons obey different rules. The amplitudes that describe quantum states are closely related to probabilities, with an important distinction: they are complex numbers and can be negative as well as positive. This gives rise to the phenomena of quantum interference. When an event can happen one way with a positive amplitude and another way with a negative amplitude, the two possibilities can cancel out so that the final amplitude is zero and the event never happens. This is an example of destructive interference. On the other hand there is constructive interference: the possibilities of events with two positive or negative amplitudes can reinforce so that the final amplitude is increased and the event is more likely to happen.

Particles can interact and produce entangled states. These entangled states show a correlation in measurement outcomes. For example, we could create an entangled two-qubit state so that when we measure one qubit, we also know the measurement outcome of the other qubit. Entanglement works over any distance: given an entangled state of which one qubit is located on earth and one located in another galaxy, when we measure the qubit on earth we immediately know the state of the qubit in the other galaxy. Einstein, Podolsky, and Rosen [14] famously argued that the theory of quantum mechanics was incomplete because this would allow for faster than light communication, which is forbidden by the theory of relativity. This is however

not the case, as a classical channel is required to communicate the results between the observers [9].

Using the quantum phenomena superposition, interference, and entanglement, we can do certain computations more efficient than classical computers. As Aaronson [2] describes it: "The goal in quantum computing is to choreograph a computation so that the amplitudes leading to wrong answers cancel each other out, while the amplitudes leading to right answers reinforce." We will look further into how to define and compare the efficiency of classical and quantum algorithms in Section 3.

To put the potential computing and storage power of a quantum computer into perspective, imagine that we have a 1000-qubit state. Describing this state's amplitudes requires $2^{1000} \approx 10^{302}$ numbers — more numbers than there are atoms in the observable universe. We could use these 10^{302} numbers to store information and carry out algorithms. Unfortunately, we would not be able to retrieve the information stored in the quantum state efficiently. Remember, when we observe a quantum state in superposition, it collapses to a single state. The 1000-qubit quantum state is in a superposition of 10^{302} states, and when we measure it, we only see one possibility at random. Quantum algorithms needs some clever trick so that the correct answers have a high probability of being measured and the wrong answers have a low probability of being measured.

3 Quantum Computational Complexity

To talk about the efficiency of an algorithm, we would like to be able to classify the difficulty of the given algorithm. We note that the time taken by a given algorithm generally grows with the size of the input. For example, sorting a million numbers takes longer than a thousand numbers. Because of this, when we analyze the efficiency of a certain algorithm, it is traditional to describe the running time of an algorithm as a function of the size of the input. We usually concentrate on the worst-case running time of an algorithm. Given an algorithm whose running time grows by some constant k given an input size of n, we write that this algorithm has a worst-case running time of $O(n^k)$. We call the set of problems that run in $O(n^k)$ polynomial-time algorithms. Generally, we refer to problems that can be solved in polynomial time as easy, and problems that require superpolynomial time as hard [12].

We define complexity classes as collections of problems which share some common feature with respect to the computation resources required to solve those problems. Two of the most important complexity classes are P (polynomial time) and NP (nondeterministic polynomial time). The class P consists of problems that are solvable in polynomial time, and NP consists of problems that are *verifiable* in polynomial time. Verifiable means we can check that the answer to the problem is correct in polynomial time. For example, we can check if a completed Sudoku is correct in polynomial time, but actually solving a Sudoku can not be done in polynomial time. Any problem in P is also in NP, since if it is in P, we can solve it in polynomial time and thus also verify it in polynomial time. A major unsolved problem in computer science is whether P = NP. That is, can every problem whose solution can be verified in polynomial time also be solved in polynomial time?[2]

[2] According to polls, most researchers believe P ≠ NP [21, 22, 20].

The class NP captures a huge amount of practical problems. NP-complete problems are the hardest problems in NP. If an efficient algorithm for any of the NP-complete problems would be found, it could be adapted to solve all other NP problems as well [3]. So if we were able to find an efficient algorithm for any NP-complete problems, it would mean that P = NP.

Where do quantum algorithms fit in these complexity classes? We define BQP (bounded-error quantum polynomial time) to be the class off all problems which can be solved in polynomial time on a quantum computer, where a bounded probability error of at most 1/3 is allowed [29]. A corresponding complexity class for classical probabilistic computers is BPP (bounded-error polynomial time). We know that quantum computers can solve some problems in polynomial time that classical probabilistic computers cannot, so BPP ⊆ BQP [11]. It is still uncertain where BQP exactly fits with respect to P and NP. What we do know is that quantum computers can solve all problems in P in polynomial time [7]. That is, a quantum computer can solve all problems in polynomial time that a classical computer can. What about problems in NP? Quantum computers can solve some NP problems efficiently like factoring, but it seems unlikely they can solve NP-complete problems efficiently [1]. Of course if a quantum computer could solve an NP-complete problem in polynomial time, they could in turn solve *all* NP problems efficiently, which would mean P = NP.

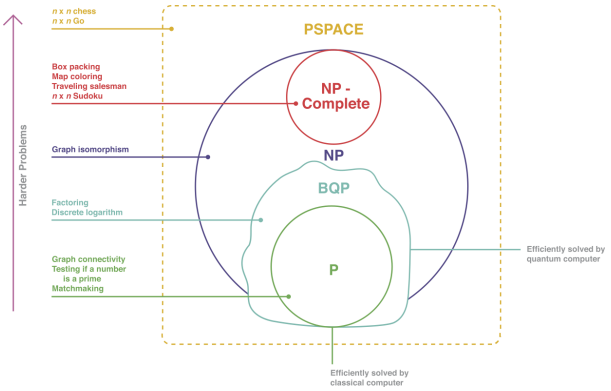

Figure 1: A diagram illustrating the hierarchy of several important complexity classes. The exact relation between NP and BQP is not yet known. Image by MIT OpenCourseWare.

3.1 *Quantum Speedups*

We talk about a quantum speedup if there exists a quantum algorithm that is more efficient than any known classical algorithm. It is useful to distinguish between two different kind of speedups: polynomial speedups and superpolynomial speedups. The holy grail of quantum computing is a superpolynomial speedup. This means there is an efficient quantum algorithm for a problem that is hard for classical computers. Polynomial speedups are less powerful, but still practically useful. For example, unstructured database search takes $O(n)$ time on a classical computer, but using Grover's algorithm we can do it in $O(\sqrt{n})$ time on a quantum computer [19]. This is a quadratic speedup, but the time taken by the quantum algorithm is still linear in n. A superpolynomial speedup would give us a time complexity of $O(\log n)$, which is more desirable. For the database search problem, we

know that $O(\sqrt{n})$ is the best we can do [10]. This is another reminder of the limitations of quantum computers: we cannot expect superpolynomial speedups for all algorithms, and even polynomial speedups are not just up for grabs.

Algorithm	Quantum Speedup	Technique
Factoring	Superpolynomial	[35]
Quantum Simulation	Superpolynomial	[36, 27, 6]
Searching	Polynomial	[19]
Constraint Satisfaction	Polynomial	[4]

Table 1: A non-exhaustive list of some important algorithms that have a quantum speedup. For a more detailed list of quantum speedups see [24].

4 Quantum Computing for Scientific Problems

While we are still in the very early stage of quantum hardware development, theorists have been finding applications in a variation of (scientific) fields. As of writing this, we are currently in the noisy intermediate-scale quantum (NISQ) era of quantum hardware [34]. These current quantum computers have about 50-100 noisy qubits. Due to noise, errors may occur in these quantum systems and greatly limit their ability to do useful computations. On this scale there are limited applications for these machines (such as exploring many-body quantum physics), and it will take more and better qubits to really change the world. When we discuss applications of quantum computers, we assume access to a fully universal fault-tolerant quantum computer[3], unless noted otherwise.

4.1 Quantum Chemistry

Quantum chemistry is probably the most promising application for quantum computing, both for NISQ and fault-tolerant quantum computers. Since the founding of the field of quantum mechanics, we have been able to describe the state of a quantum-mechanical system by solving the Schrödinger equation [18]. However, the complexity of a quantum system grows exponentially with the number of particles. This makes classical computers unfit for simulating quantum systems efficiently. As Dirac [13] noted: "*The exact application of these laws leads to equations much too complicated to be soluble.*"

It is believed that quantum computers will have applications in chemistry, biology, drug discovery, and materials science [28]. Several quantum algorithms have been proposed to solve problems in chemistry more efficiently [26, 25, 6, 33]. However, currently available NISQ hardware restricts these experiments to focus only on small molecules that we can already simulate classically.

A fundamental problem in computational chemistry is finding the ground-state energy level of molecules [6]. This has also been a main area of research in the field of quantum computational chemistry. There are two well-known quantum algorithms for finding the ground-state of a molecule. The quantum phase estimation algorithm (QPE) is one that offers an exponential speedup over classical algorithms. However, it requires a quantum computer to stay coherent for millions to billions of quantum operations, compared to the tens to hundreds achievable in the near-term. An alternative algorithm suited for NISQ devices named the variational quantum eigensolver (VQE) was recently proposed [33]. It greatly reduces the quantum resources required by using classical optimization methods combined with a short quantum subroutine for measuring the expectation value of a Hamiltonian. The QPE and VQE algorithms can more generally be described as finding eigenvalues of a unitary operator. This has use cases outside chemistry such as determining the results of internet search engines [32], and designing new materials and drugs [16].

4.2 Machine Learning

5 Conclusion

[3] Fault-tolerant quantum computers are quantum computers capable of handling any errors [30]. This is necessary to carry out meaningful computations.

References

[1] Scott Aaronson. "BQP and the polynomial hierarchy". In: *Proceedings of the forty-second ACM symposium on Theory of computing*. 2010, pp. 141–150.

[2] Scott Aaronson. "Quantum Computing Promises New Insights, Not Just Supermachines". In: *Quantum Computing since Democritus*. Cambridge University Press, 2011. Chap. Quantum, pp. 109–131. ISBN: 9780511979309. DOI: 10.1017/cbo9780511979309.010.

[3] Scott Aaronson. "The limits of quantum". In: *Scientific American* 298.3 (2008), pp. 62–69.

[4] Andris Ambainis. "Quantum search algorithms". In: (2005). arXiv: quant-ph/0504012.

[5] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (2019), pp. 505–510. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1666-5. URL: https://doi.org/10.1038/s41586-019-1666-5.

[6] A. Aspuru-Guzik. "Simulated Quantum Computation of Molecular Energies". In: *Science* 309.5741 (2005-09), pp. 1704–1707. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1113479.

[7] Charles H Bennett. "Logical reversibility of computation". In: *IBM journal of Research and Development* 17.6 (1973), pp. 525–532.

[8] Charles H Bennett and Gilles Brassard. "Quantum cryptography". In: *Theoretical Computer Science* 560.P1 (2014), pp. 7–11.

[9] Charles H Bennett and Stephen J Wiesner. "Communication via one-and two-particle operators on Einstein-Podolsky-Rosen states". In: *Physical review letters* 69.20 (1992), p. 2881.

[10] Charles H Bennett et al. "Strengths and weaknesses of quantum computing". In: *SIAM journal on Computing* 26.5 (1997), pp. 1510–1523.

[11] Ethan Bernstein and Umesh Vazirani. "Quantum complexity theory". In: *SIAM Journal on computing* 26.5 (1997), pp. 1411–1473.

[12] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2009.

[13] P. A. M. Dirac. "Quantum Mechanics of Many-Electron Systems". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 123.792 (1929-04), pp. 714–733. ISSN: 1364-5021, 1471-2946. DOI: 10.1098/rspa.1929.0094.

[14] Albert Einstein, Boris Podolsky, and Nathan Rosen. "Can quantum-mechanical description of physical reality be considered complete?" In: *Physical review* 47.10 (1935), p. 777.

[15] Richard P. Feynman. "Simulating physics with computers". In: *Int J Theor Phys* 21.6-7 (1982-06), pp. 467–488. ISSN: 0020-7748, 1572-9575. DOI: 10.1007/bf02650179.

[16] Gene H. Golub and Henk A. van der Vorst. "Eigenvalue computation in the 20th century". In: *J. Comput. Appl. Math.* 123.1-2 (2000-11), pp. 35–65. ISSN: 0377-0427. DOI: 10.1016/s0377-0427(00)00413-1.

[17] Google AI. *Quantum*. 2019. URL: https://ai.google/research/teams/applied-science/quantum/.

[18] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. Cambridge University Press, 2018-08. ISBN: 9781316995433, 9781107189638. DOI: 10.1017/9781316995433.

[19] Lov K Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[20] Lane A Hemaspaandra. "SIGACT News Complexity Theory Column 100." In: *SIGACT News* 50.1 (2019), pp. 35–37.

[21] Lane A Hemaspaandra. "Sigact news complexity theory column 36". In: *ACM SIGACT News* 33.2 (2002), pp. 34–47.

[22] Lane A Hemaspaandra. "SIGACT news complexity theory column 74." In: *SIGACT News* 43.2 (2012), pp. 51–52.

[23] IBM. *Quantum Computing*. 2019. URL: https://www.ibm.com/quantum-computing/.

[24] Stephen Jordan. "Quantum algorithm zoo". In: (2020). URL: https://quantumalgorithmzoo.org/ (visited on 2020-05-13).

[25] I. Kassal et al. "Polynomial-time quantum algorithm for the simulation of chemical dynamics". In: *Proceedings of the National Academy of Sciences* 105.48 (2008-11), pp. 18681–18686. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0808245105.

[26] Daniel A. Lidar and Haobin Wang. "Calculating the thermal rate constant with exponential speedup on a quantum computer". In: *Phys. Rev. E* 59.2 (1999-02), pp. 2429–2438. ISSN: 1063-651X, 1095-3787. DOI: 10.1103/physreve.59.2429.

[27] Seth Lloyd. "Universal quantum simulators". In: *Science* (1996), pp. 1073–1078.

[28] Sam McArdle et al. *Quantum computational chemistry*. 2018. arXiv: 1808.10402 [quant-ph].

[29] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information. 10th Anniversary Edition*. 10th. New York, NY, USA: Cambridge University Press, 2009. ISBN: 9780511976667. DOI: 10.1017/cbo9780511976667.

[30] Steven Oud. *Introduction to Quantum Computing*. 2019. URL: https://github.com/soudy/introduction-to-quantum-computing/raw/master/introduction-to-quantum-computing.pdf.

[31] Steven Oud. *Simulation of Quantum Algorithms for Solving Machine Learning Problems*. 2019. URL: https://github.com/soudy/quantum-ml-research-paper/blob/master/quantum-ml-research-paper.pdf.

[32] Lawrence Page et al. *The pagerank citation ranking: Bringing order to the web*. Tech. rep. Stanford InfoLab, 1999.

[33] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nat Commun* 5.1 (2014-07), p. 4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[34] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79.

[35] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Rev.* 41.2 (1999-01), pp. 303–332. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/s0036144598347011.

[36] Christof Zalka. "Efficient simulation of quantum systems by quantum computers". In: *Fortschritte der Physik: Progress of Physics* 46.6-8 (1998), pp. 877–879.