2020

Quantum Computing and Quantum Supremacy

10555083

Introduction

Quantum computing was first proposed by Richard Feynman in 1982 (Feynman, 1982). Since then countless advances have been made, from the development of basic quantum algorithms such as the Deutsch Oracle Problem which led to the development of Shor's Algorithm (Shor, 1994); a quantum algorithm for factoring large numbers, which fundamentally breaks many of the assumptions modern cryptography relies on. To the creation of physical quantum systems, which can allow for real usage of these algorithms, and now Google demonstrating their quantum supremacy in late 2019 with a sampling problem.

Many of these advancements also bring up a lot of new questions, such as; what is quantum computing and how does it work; what does supremacy mean; what are the current issues with it and what has Google achieved?

Background

When looking into quantum supremacy, it is important to understand the fundamentals of quantum computing and how it differs from classical computation, as well as truly understanding what quantum supremacy means.

The major fundamental difference between quantum and classical computation is the difference between a classical and quantum bit (Qubit). At its simplest, a classical bit is like a switch and can be in one of two states; usually represented in base 2 as 0 and 1. A classical bit can also be represented as a 2-by-1 vector as shown below:

$$|1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \quad |0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$$

The $|x\rangle$ notation used here is the Dirac notation and is the standard notation used for states in a quantum device (Nielsen & Chuang, 2010).

A qubit can exist in both states simultaneously, known as a superposition, until measured, at which point it will collapse to one of the two states. The matrix representation of a qubit can be shown like this:

$$|x\rangle = \begin{bmatrix}c_0\\c_1\end{bmatrix} \text{ where } |c_0|^2 + |c_1|^2 = 1$$

In this representation, c_0 and c_1 are complex numbers, representing the probability a qubit will end up in each of the states when a measurement is taken (Yanofsky & Mannucci, 2008). As a qubit can be represented in this manner and because the sum of the probabilities squared must equal one, a qubit can also be seen as represented by a unit vector on a 2D plane (Nielsen & Chuang, 2010). This much wider array of potential states unlocks many possibilities for quantum computation and is where much of its power over a classical system comes from. Nonetheless, it increases the space required to store the state; this is because when combining a set of qubits the tensor product is required instead of just combining them together as practised with a normal bit.

As the qubit and the classical bit are different, so are the way gates work. In quantum computing all logic gates must be reversible, there is a requirement to understand input from output. For example, the classical OR gate, which takes two inputs and outputs and would return 1 if either of the inputs is 1 or would return 0 if neither of them were. This would not work on a quantum computer; as a result of 1 will not tell you which bit was 1 or if both were. The CNOT gate is an example of a reversible quantum gate, it uses 2 qubits and will apply the NOT operation on the first qubit when the first qubit is $|1\rangle$. This required usage of reversible gates does have an advantage for quantum computers because there is a minimum amount of energy required for a system to get rid of information (Landauer, 1961) which is removed by reversible gates, lowering the energy require to compute information.

Qubit's may also be entangled; entanglement describes two (or more) qubits having very strong correlation. For example, if two entangled qubits in state $|11\rangle$ are present, we could measure the first qubit and immediately know the second qubit would also be in that state (Yanofsky & Mannucci, 2008). This also manifests when those qubits are incredibly distant. Fig. 1 shows an example of entanglement between 2 qubits running off IBMs quantum experience.

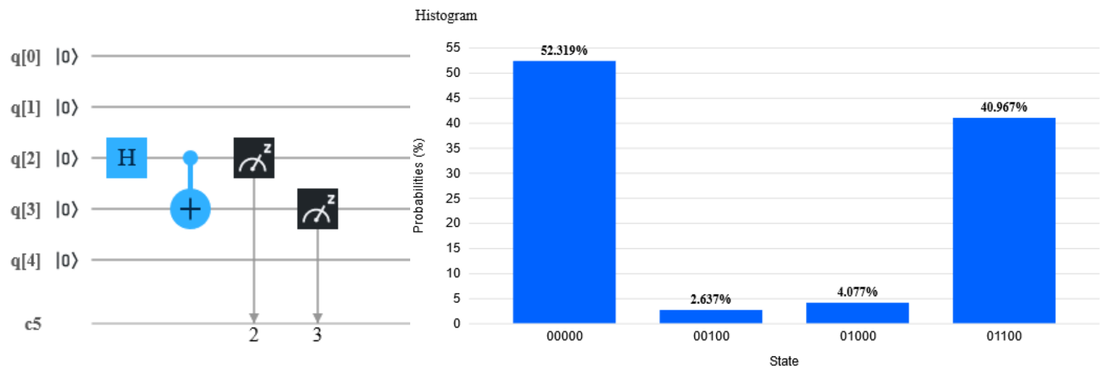

Figure 1: A basic quantum circuit showing entanglement between 2 qubits (with some errors). The first qubit is passed through a Hadamard gate, which puts it in to an equal superposition and then through a CNOT gate, which will flip it depending on the state of the control bit. Which leads to them ending up in either states $|11\rangle$ or $|00\rangle$.

At its simplest, quantum supremacy is the point at which a quantum computer can do something which cannot be efficiently simulated by a classical computer, whether the task has any real value or not. Harrow and Montanaro (Quantum computational supremacy, 2017) wrote of four requirements which any supremacy experiment should have. These are:

1. A well-defined computational task,
2. A plausible quantum algorithm for the problem,
3. An amount of time/space that is allowed for the classical competitor,
4. A complexity-theoretic assumption,
5. (Optional) A verification method that can distinguish the quantum algorithm from the classical competitor.

The requirements clearly define how any quantum supremacy experiment should be run and lay out a set of rules for how quantum supremacy can be unassailably achieved, which is consistent with many of the tasks being used as potential ways to demonstrate quantum supremacy.

Main Issues

The problems with quantum computing

As quantum computing is still a new field, there are doubts about whether it will be able to meet the theory expectation of what it can achieve and whether real quantum computers large enough to solve problems such as Shor's Algorithm for large number factoring (Shor, 1994), can even be realised.

One major problem with quantum computing is the presence of noise and other errors when dealing with qubits. Many algorithms exist for quantum computation and often estimate the number of qubits needed to work. However, these use a logical qubit in their assumptions. Noise and errors cause this problem because to implement a single logical qubit can take thirteen physical qubits minimum and to create a reasonably fault-tolerant qubit, which can be used effectively, could take 10^3 to 10^4 physical qubits (Fowler, Mariantoni, Martinis, & Cleland, 2012). The large number of qubits can cause more problems with the feasibility of a system as there is potential for the error rates to increase as more qubits are added to the system, a larger qubit count may be required as a consequence.

When approaching quantum supremacy, verifying the algorithm works correctly and is in fact better than its classical counterpart is another problem with quantum computers. Harrow and Montanaro (2017) argue, as quantum computers approached the point of quantum supremacy, it would become harder to ensure they were working correctly as it would become difficult to simulate them classically within a realistic time frame. Some quantum algorithms do not have this problem, for instance, Shor's algorithm could be verified by simply multiplying the factors back out and making sure they reach the correct number; most do. One potential solution would be to test small parts of the system and use as the basis of assumptions for error rates when the system is scaled up although, this carries the risk of not giving a representation of how the system functions as a whole. An analogous comparison would be the role of unit testing in software engineering, where it confirms an individual function works although, it gives no guarantees the function will work in the context of the whole system.

Current achievements

In 2018, researchers working with Google released a paper which discussed performing a sampling problem with a system of 9 qubits as a blueprint for demonstrating quantum supremacy if the system could be scaled up to 50 qubits (Neill, et al., 2018). In late 2019 Google achieved quantum supremacy using this sampling problem with a quantum computer which had 53 qubits. Their results (see Fig. 2) indicated their quantum computer completed the task in 200 seconds and estimated that a classical supercomputer would have taken over 10,000 years to generate the same results (Arute, et al., 2019).

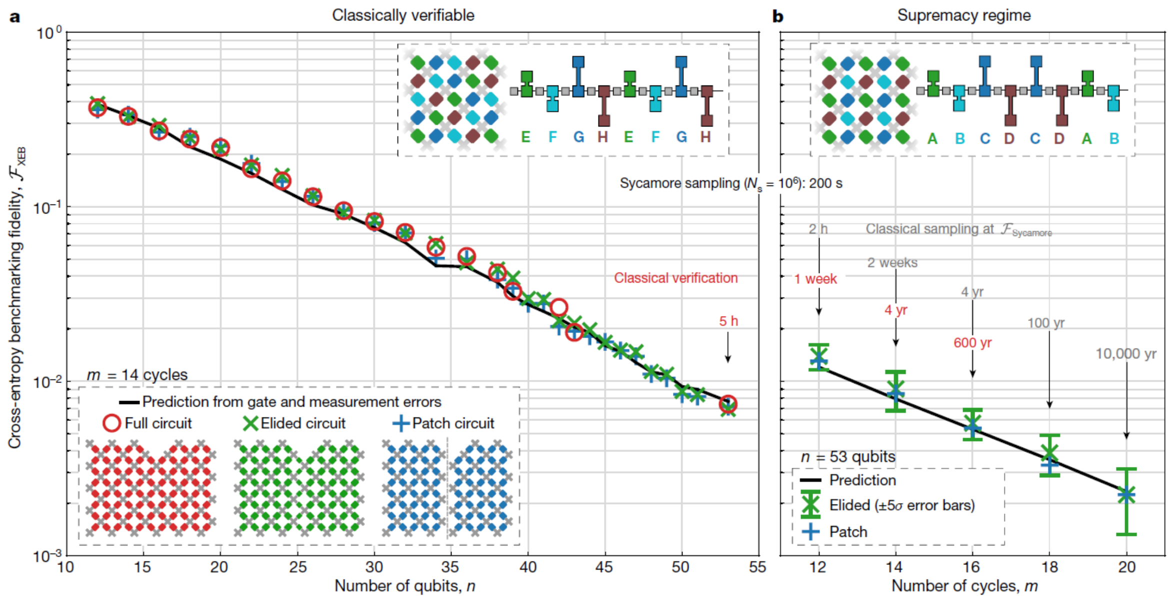

Figure 2: The results of Google's quantum supremacy experiment, showing the time that would be taken by a classical computer to solve the problem. (Arute, et al., 2019)

The reason for this task's suitability as a demonstration of quantum supremacy is because the lack of structure within the random circuits means the problem has some guarantee of computational hardness; where computational hardness assumes the problem cannot be solved in polynomial time. Subsequently, simulating the task on a classical supercomputer would become exponentially more difficult as the number of qubits grows (Arute, et al., 2019). One reason for this is the number of bits required to represent the state of a set of qubits is 2^n where n is the number of qubits the system needs to simulate (Yanofsky & Mannucci, 2008). This huge growth in state storage size alone means systems will need more than a petabyte of RAM (Random Access Memory) just to able to hold the state of their qubits without needing to resort to other techniques of managing it.

Google have solved the error and verification problems mentioned above by using a technique they have called "cross entropy benchmarking." This technique allows them to estimate the fidelity of the system as the average of the probabilities of bit strings they expect to get from the sampling (Boixo, et al., 2018). Using this technique provides an idea of potential system returns and allows them to continue verifying the data as the number of qubits grows. Google have also modelled the error rates per qubit by testing each qubit and coupler individually and in small groups (Fig. 3). In doing so allows them to understand the error rate of the system as well as the error rate of individual qubits, thus providing a better understanding of the data and the ability to filter out or find errors within.

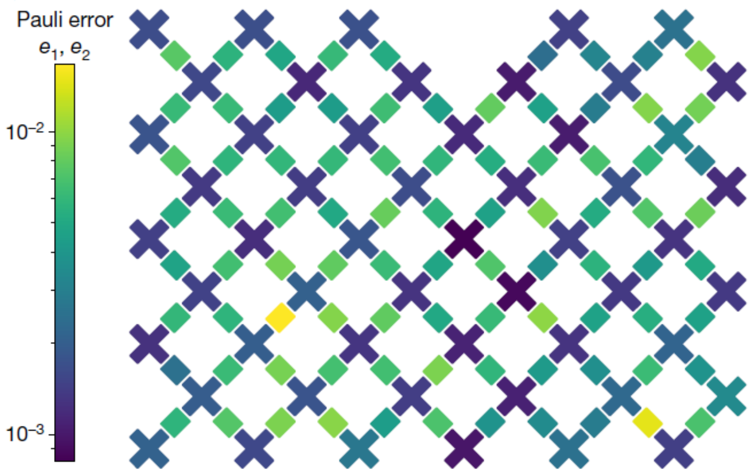

Figure 3: Results of googles error measurement on their system, the X's represent each individual qubit and the squares are the couplers between them. One of the qubits in the top row was removed as it was not functioning correctly. (Arute, et al., 2019)

This result is not without criticism. IBM responded to the current level of quantum supremacy achieved by Google, where they estimate their summit supercomputer can complete the simulation in just 2.5 days instead of the 10,000 years proposed by Google by leveraging solid state storage to compensate for the lack of random access memory (Pednault, Gunnels, Nannicini, Horesh, & Wisnieff, 2019). Notwithstanding, 200 seconds is still a large speedup over 2.5 days and IBM's own results (Fig. 5) do show the time taken increases at a much faster rate as the depth and number of number of qubits grows, potentially meaning the speedup will only increase as the system grows larger and the system still will maintain a claim to quantum supremacy.

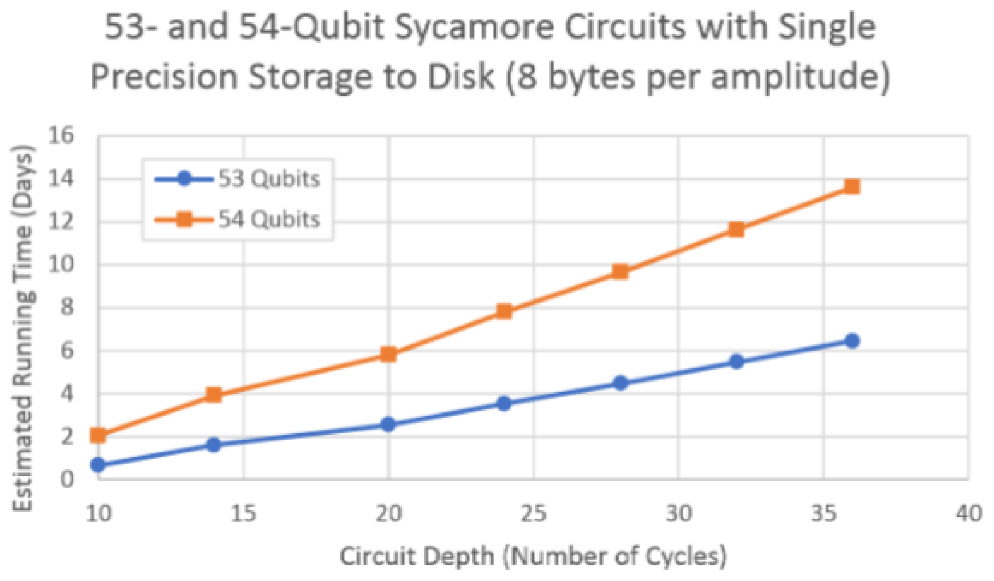

Figure 4: Graph from IBM's response showing the estimated running time of their classical simulation. The overall rate of increase does go up as the number of qubits and circuit depth increase. (Pednault, Gunnels, Nannicini, Horesh, & Wisnieff, 2019)

Conclusion

In conclusion, quantum computing has come a long way in a short time. While Google's recent achievements may not fit under some definitions of quantum supremacy, they have achieved it by the requirements set out in this report, defined by Harrow and Montanaro (2017). Their quantum supremacy experiment used a well-defined task with a solid algorithm and strong explanation, as well as a clearly defined time bounds, for the classical computer alongside a strong assumption of its computational hardness due to the unstructured nature of the problem. Atop this, they used a strong verification model, modelling the error rate down to the individual qubit and gates so the true fidelity of the system could be understood allowing them to strongly assert their output was correct and in line with the predictions made by also simulating a much smaller version of the system.

Overall these achievements show quantum supremacy is possible at some scale and their confidence the error rate is only additive as the number of qubits rises (Arute, et al., 2019), should mean that quantum systems can become even larger over time and that we could even witness something resembling a quantum computing version of Moore's Law, where the number of qubits in a system will double ever few years. However, it still must be kept in mind while quantum computation can be useful, not everything will be able to see an improvement from quantum computing and only some application will ever be able to use its actual potential.

References

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., & Martinis, J. M. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 505-510. doi:https://doi.org/10.1038/s41586-019-1666-5

Boixo, S., Isakov, S. V., Smelyanskiy, V. N., Babbush, R., Ding, N., Jiang, Z., . . . Neven, H. (2018). Characterizing Quantum Supremacy in Near-Term Devices. *Nature Physics*, 595-600. doi:https://doi.org/10.1038/s41567-018-0124-x

Feynman, R. P. (1982). Simulating physics with computers. *International Journal of Theoretical Physics*, 467-488.

Fowler, A. G., Mariantoni, M., Martinis, J. M., & Cleland, A. N. (2012). Surface codes: Towards practical large-scale quantum computation. *Physical Review A*. doi:10.1103/PhysRevA.86.032324

Harrow, A. W., & Montanaro, A. (2017). Quantum computational supremacy. *Nature*, 203-209. doi:https://doi.org/10.1038/nature23458

Landauer, R. (1961). Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development*, 183-191.

Neill, C., Roushan, P., Kechedzhi, K., Boixo, S., Isakov, S. V., Smelyanskiy, V., . . . Martinis, J. M. (2018). A blueprint for demonstrating quantum supremacy with superconduction qubits. *Science*, 195-199. doi:DOI: 10.1126/science.aao4309

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum Computation and Quantum Information, 10th Anniversary Edition.* Cambridge University Press.

Pednault, E., Gunnels, J. A., Nannicini, G., Horesh, L., & Wisnieff, R. (2019). Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits. *arXiv:1910.09534 [quant-ph]*.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science* (pp. 124-134). Santa Fe, NM, USA: IEEE.

Yanofsky, N. S., & Mannucci, M. A. (2008). *Quantum Computing for Computer Science.* Cambridge University Press.