Chapter 6

Quantum Supremacy
The problem Google solved

On October 24 2019, a research group lead by John Martinis announced to have achieved Quantum Supremacy, meaning that they performed a task using a programmable quantum architecture - calling this a quantum computer is still premature - that can’t be solved with current classical computers. More precisely, they constructed an architecture of 53 qubits that they used to solve a sampling problem in 3 minutes which would take 10000 years to solve on a desktop computer. IBM, Google’s main competitor, announced that this problem could be solved in only a few days using the best existing super computers, diminishing Google’s claims.

In this Chapter, we will present the sampling problem solved for this Quantum Supremacy Experiment, the associated quantum architecture that was built and a discussion on the classical difficulty of this problem. A good part of this lecture is a rewriting of parts of the Google article.

6.1 Sampling Random Quantum Circuits

Circuit Sampling: The input is a description of a $n$-qubit quantum circuit $U$, described by a sequence of one and two-qubit gates. The task of the problem is to sample from the probability distribution of outcomes

$$p_U(x) = \langle x|U|0^n\rangle^2.$$ 

If we run $U$ on input $|0^n\rangle$ and measure the whole output, $p_U(x)$ is the probability that we measure $x$.

A classical algorithm for (perfect) circuit sampling can be thought of, without loss of generality, as a function $A$, mapping $m \in \text{poly}(n)$ bits $r = (r_1, \ldots, r_m)$ to $n$ bits such that

$$\forall x \in \{0, 1\}, \frac{1}{2^m} |\{(r_1, \ldots, r_m) : A(r_1, \ldots, r_m) = x\}| = p_U(x).$$

6.1.1 Brief overview of the difficulty of the problem

In the quantum setting, it is easy to sample from distribution $p_U$: run $U$ on input $|0^n\rangle$, and measure the output. By definition of $p_U$, the output will be $x$ with probability exactly $p_U(x)$ (on a perfect quantum computer). However, it is believed to take an exponential time in $n$ to perform with a classical computer. This is the starting point of the quantum supremacy experiment: build a small quantum unitary $U$ from which we will be able to sample $p_U$ while the problem should be hard for a classical computer. These are the hurdles that have to be overcome in order for this approach to work:
• First, build a quantum architecture for which we will be able to compute the unitary $U$ for a randomly chosen $U$. The quantum supremacy experiment uses $n = 53$ but there are a lot of practical imperfections so the sampling problem solved has a lot of errors.

• Taking into account these imperfections, the classical problem should still be hard.

6.2 The Quantum Sampling Problem

6.2.1 The architecture used

When we look at circuits of 1 and 2 (qu)bit, we usually consider the case where any pair of wires can be put together in order to perform a 2 (qu)bit gate. However, in the quantum setting, this is challenging and we only have near neighbor interactions. In the quantum supremacy experiment, the interactions can be done as follows:

Each square represents a qubit and only 2 squares (i.e. qubits) that have a common edge can be used together to perform 2 qubit gates. Now that we have our qubit layout, what unitary $U$ could be performed by the Google team? They apply 20 times the following procedure:

• For each qubit, apply one of the 3 following single qubit unitaries, which hasn’t been applied in the previous round

$$X' = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ -i & 1 \end{pmatrix} ; \quad Y' = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} ; \quad W' = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & \sqrt{-i} \\ \sqrt{-i} & 1 \end{pmatrix}$$

• Then, we apply the following 2 qubit gate $G_2$, on pairs, according to one of the layouts below.

$$G_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -i\sin(\theta) & 0 \\ 0 & -i\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix}$$

with $\theta \approx \frac{\pi}{2}$ and $\phi \approx \frac{\pi}{6}$. 

30
The device here constructed is *programmable* in the sense that we can choose between the different gates $X', Y', W'$ as well as the places where we apply $G_2$. Moreover, they showed that the set \{ $X', Y', W', G_2$ \} forms a universal set of quantum gates so if they could scale up this technology then they could construct a full quantum computer.

### 6.2.2 Errors in the quantum sampling

We showed how to construct this random unitary $U$ and we said that by running $U$ quantumly, we could sample from the distribution $p_U$. However, in the real quantum supremacy experiment, we can only sample $U$ with a lot of errors. The quantum supremacy experiment says that each of their have around 0.3\% of errors. We will consider here a simplified model where we assume that a gate function correctly with probability 99\% and that it outputs a random state otherwise\(^1\).

In each of the 20 rounds, there are 53 single qubit gates and $\leq 24$ 2-qubit gates, depending on the layout. This means that the whole unitary is correctly performed with probability at least $0.997^{(53+24)\cdot 20} \approx 1\%$. The rest of the time, because the noise is a random noise, the measured output is a uniform random string. Therefore, the quantum supremacy experiment can sample from the distribution $\tilde{p}_U$ defined as

$$\tilde{p}_U := Fp_U + (1 - F)\text{Unif.}$$

with $F = 1\%$.

**Verifying that succeeded the sampling task.** Notice that $p_U$ (resp. $\tilde{p}_U$) are each characterized by $2^{53}$ probabilities corresponding to the different $p_U(x)$ (resp. $\tilde{p}_U(x)$). For a given $U$, we therefore don’t know the full distributions $p_U$ and $\tilde{p}_U$ so how can we check that the quantum supremacy experiment indeed samples from $\tilde{p}_U$? Well right now, we can’t. The only thing we can do is to analyze the behavior of the quantum architecture on smaller values of $n$, for which we can compute $\tilde{p}_U$, and then, to extrapolate this behavior for $n = 53$. There are different advanced extrapolation arguments to show that the quantum supremacy experiment does what is says it does\(^2\) but only a future calculation of $\tilde{p}_U$ will determine if they indeed sampled from $\tilde{p}_U$ (or something approaching) or not.

---

\(^1\)This is actually not an necessary assumption in the Google experiment. They use the notion of cross entropy that allows to deal with any kind of errors and not just random errors.

\(^2\)In particular, they study a regime where $n = 53$ but the unitary $U$ is a product unitary $U_1 \circ U_2$ over 2 chunks and $\tilde{p}_U$ becomes computable.
6.3 Classical hardness of the sampling problem

6.3.1 A bit of complexity: the language AM.

**Definition 6.** A language $L$ is in AM iff. there exists a randomized algorithm $A(x)$ running in expected polynomial time (in its input’s size $|x|$) and 2 polynomials $p, q$ such that

- $\forall x \in L, \Pr_{y \in \{0,1\}^{p(|x|)}}[\exists z \in \{0,1\}^{q(|z|)}, A(x, y, z) \text{ outputs } 1] \geq 2/3$.
- $\forall x \not\in L, \Pr_{y \in \{0,1\}^{p(|x|)}}[\forall z \in \{0,1\}^{q(|z|)}, A(x, y, z) \text{ outputs } 1] \leq 1/3$.

AM can be seen as the complexity class for which there exists a 2 message interactive proof between a polynomial time verifier and an all powerful prover where:

- $V$ first sends a random string $y \in \{0,1\}^{p(|x|)}$.
- $P$ sends a string $z$ to the verifier.
- $V$ uses $A$ and accepts if $A(x, y, z)$ outputs 1.

![Diagram of AM protocol](image)

$V$ accepts if $A(x,y,z) = 1$

6.3.2 Classical hardness of the problem

In order to argue about the classical hardness, we will use the following proposition and conjectures.

**Proposition 2.** For a random unitary $U$, if one can approximately compute $|\langle 0 | U | 0 \rangle|^2$ then one can compute Permanents\(^3\) of random matrices. This is $\#P$-complete.

$\#P$ is the complexity class that corresponds to counting the number of solutions of a given problem. We present below some examples of $\#P$-complete problems, taken from Wikipedia.

Examples of $\#P$-complete problems include:

- How many different variable assignments will satisfy a given general boolean formula? ($\#\text{SAT}$)
- How many different variable assignments will satisfy a given DNF formula?
- How many different variable assignments will satisfy a given 2-satisfiability problem?
- How many perfect matchings are there for a given bipartite graph?
- What is the value of the permanent of a given matrix whose entries are 0 or 1? (See Sharp-P-completeness of 01-permanent.)
- How many graph colorings using $k$ colors are there for a particular graph $G$?
- How many different linear extensions are there for a given partially ordered set, or, equivalently, how many different topological orderings are there for a given directed acyclic graph?\(^1\)

\(^3\)For a matrix $A = (a_{i,j})$, we have $\text{Perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}$ where $S_n$ is the set of permutations of $\{1, \ldots, n\}$.
Proposition 3. If there is a polynomial classical algorithm to sample from \( \tilde{p}_V \) for a random \( U \) then there is an AM protocol to solve the problem of computing \( |\langle 0|U|0 \rangle|^2 \) for a random \( U \).

Conjecture 1. \( \#P \subseteq AM \).

If this conjecture is false, i.e. if \( \#P \subseteq AM \) then crazy things happen in complexity theory. In particular, the polynomial hierarchy collapses to the third level.

Putting the 2 Propositions together, we conclude that if there exists a polynomial time classical algorithm for the sampling problem then Conjecture 1 is false and the polynomial hierarchy collapses to the third level.

6.3.3 Best classical algorithms of the sampling problem

In the previous section, we sketched the argument that shows that the sampling problem should be hard for quantum computers. We present here what corresponds to the 10000 years claimed by Google and the 2 – 3 days claimed by IBM. Here are the running times expected by Google, taken from their paper. Plots (a) and (c) correspond to running times of the actual sampling problem. Plot (b) is the time to compute a verification circuit, as an indicator that the extrapolation of the behavior of their quantum architecture behaves well.

![Graphs showing running times for different algorithms](image)

FIG. S50. Scaling of the computational cost of XEB using SA and SFA. a, For a Schrödinger algorithm, the limitation is RAM size, shown as a vertical dashed line for the Summit supercomputer. Circles indicate full circuits with \( n = 12 \) to 43 qubits that are benchmarked in Fig. 4a of the main paper. 53 qubits would exceed the RAM of any current supercomputer, and is shown as a star. b, For the hybrid Schrödinger-Feynman algorithm, which is more memory efficient, the computation time scales exponentially in depth. XEB on full verifiable circuits was done at depth \( m = 14 \) (circle). c, XEB on full supremacy circuits is out of reach within reasonable time resources for \( m = 12, 14, 16 \) (stars), and beyond. XEB on patch and elided supremacy circuits was done at \( m = 14, 16, 18, \) and 20.

The plot (a) shows fast running times but the RAM requirements are out of reach. What the IBM team was to find ways to optimize the RAM requirements and push the purple dotted line more on the right to include the quantum architecture from Google. However, as soon as we move
towards 70 or 80 qubits, the RAM requirements will be way too large again in order to be used on a supercomputer and as we can see, other known methods stay very slow.