Quantum Computation

quinten tupker

October 8 2020 - October 28, 2020

Introduction

These notes are based on the course lectured by Professor Richard Jozsca in Michaelmas 2020. Due to the measures taken in the UK to limit the spread of Covid-19, these lectures were delivered online. These are not meant to be an accurate representation of what was lectures. They solely represent a mix of what I thought was the most important part of the course, mixed in with many (many) personal remarks, comments and digressions... Of course, any corrections/comments are appreciated.

This course is meant to be a second course on quantum computation. In particular, all the prerequisite knowledge is covered in Cambridge's Part II Quantum Information and Computation course. Lecture notes for this course can be found online.

Now, to start describing course content. Quantum Computation studies algorithms that can be run on quantum computers. Although they have yet to be implemented in practice (but are definitely being developed at a rapid pace), and in particular how they differ with classical computation. A remarkable result, is that at least superficially, quantum algorithms appear to be more powerful than classical algorithms, although it remains unclear if this is definite fact, or if it simply easier for humans to solve complex problems using quantum algorithms instead of classical algorithms. This is remarkable in many ways in a philosophical sense, but here we focus on how to take advantage of these changes. As such, we will begin with a review and extension of one of the most famous quantum algorithms, which is Schur's factoring algorithm.

1 Schur's Algorithm Revisited and the Hidden Hidden Subgroup Problem

Schur's factoring algorithm finds a factor for an arbitrary number N. It does not perform a complete factorisation. It merely computes a factor, which of course can be repeated arbitrarily to get a complete factorisation, but that is not the point here. The complexity of such an algorithm is typically measured in terms of the number of digits of N, which we may denote $n = \ln(N)$. In these terms Schur's algorithm is $O(n^3)$, which means it is "efficient" (computationally feasible) or

Definition 1 (efficient algorithm)**.** An algorithm is efficient if it runs in polynomial time, which generally means it is considered doable in practice.

By comparison, the fastest known classical algorithm runs in $O(e^{n^{1/3}\ln(n)^{1/3}})$. Anyways, here is an outline of Schur's algorithm:

1. choose $a < N$ such that $(a, N) = 1$ (coprime). This can be done efficiently, since the probability of a being coprime is fixed, and we can quickly calculate the GCD using Euclid's algorithm. Then consider $f(x) = a^x(\text{mod}N)$.

2. Use quantum algorithms to calculate the period of the this function (so we have converted a factoring problem into period finding). Since a is coprime, it is guaranteed to be periodic.

3. compute the factor using number theory

The crucial component here is the period finding, which cannot be done efficiently using a classical algorithm. So let's review quantum period finding. Also, note that as usual, quantum oracles are implemented as unitary operators by converting $f : \mathbb{Z}_M \to \mathbb{Z}_N$ to $U_f \left|x\right\rangle \left|0\right\rangle \mapsto \left|x\right\rangle \left|f(x)\right\rangle$. Then, if f has period r (unknown), and f is one-to-one on every period, then period finding can be done using

1. make $\frac{1}{\sqrt{M}} \sum_0^{M-1} \left|i\right\rangle \left|0\right\rangle$

2. apply U_f

3. measure the output register to get

$$\frac{1}{\sqrt{A}}(\left|x_0\right\rangle + \left|x_0 + r\right\rangle + \cdots + \left|x_0 + (A-1)r\right\rangle) \left|f(x_0)\right\rangle$$

   Now the next step is the tricky part, and really is what uses the "quantum magic" here, and that involves the use of the Quantum Fourier Transform (QFT). [This ends lecture 1]

4. We then apply the QFT, which maps $\left|k\right\rangle \mapsto \sum_{y=0}^{M} e^{xy} \left|y\right\rangle$, which, after some calculation (use $\sum e^{2\pi kx/y} = y\delta_{xy}$) leaves us with

$$\text{QFT}\left|\text{per}\right\rangle = \sqrt{A/M} \sum_{k=0}^{r-1} \omega^{x_0 kM/r} \left|kM/r\right\rangle$$

5. Making a measurement we get $C = k_0 M/r$, so $\frac{k_0}{r} = \frac{C}{M}$. If k_0, r are coprime, we are done, since we can reduce $\frac{C}{M}$ to simplest terms (use Euclid's algorithm to cancel out the gcd). Now, number theory tells

us that the probability of being coprime is finite and shrinks slowly (as $O(1/\log\log(M))$), and so we can just repeat until we get the right period. Since f is one-to-one on each period, it is easy to check if our period is correct.

I feel that just being able to check if the period is correct is a somewhat lame reason to require that the function be one-to-one on each period, but improvements although not difficult, would complicate this explanation.

Anyways, let's see if we can motivate the Quantum Fourier Transform a bit better. The challenge we face is that our state, $|R\rangle$ takes the form

$$|R\rangle = \sum_k a_k \, |x_0 + kr\rangle$$

for an arbitrary x_0. In other words, we have an arbitrary shift that we want to ignore some how. How do we do that? A natural way to spot "things that ignore shifts" would be to define the shhift operator

$$U\,|x\rangle = |x + 1 \bmod M\rangle$$

and then, how do we say, "we don't care about " U? We look for the eigenvectors of U, which are by definition, the states least affected by U. Fortunately, U is a permuatation matrix, so unitary, so is a quantum gate. Then, the eigenbasis of U is what we may call the set of shift-invariant states χ_k. If we write R in terms of this basis we are bound to get a state of the form

$$\sum_k a_k \lambda_k^{x_0} \, |\chi_k\rangle$$

where $\mathbb{P}(k) = |a_k \lambda_k^{x_0}|^2 = |a_k|^2$ since as eigenvalues of a unitary matrix, $|\lambda_k| = 1$ always. So as expected, probabilities are preserved (this is not that crucial - but it's important they don't differ that much). Anyways, important is that this transformation allows us to express our state as a sum of multiples of the period, which is what we want.

All that remains is to find this basis, and the operation that expresses them in terms of it. As eigenvectors of a unitary matrix are orthogonal, all we need to do is zip the eigenvectors into a matrix. These eigenvectors are just of the form $e^{2\pi ikl/M}$, so we get the Quantum Fourier Transform we expect. [End of lecture 2]

1.1 The Hidden Subgroup Problem (HSP)

The hidden subgroup problem asks the question how can we find subgroup K of group G (this course only considers finite G) given a function $f : G \to X$ that is an invertible function of the left cosets of K. Our goal is to solve this problem in $O(poly(\ln(|G|)))$.

Examples of this include Schur's algorithm where we have $G = \mathbb{Z}_p^*$, then $K = 0, r, 2r, \ldots$, and then f used before works for our purpose. Another example is calculating **discrete logarithm**, which if done efficiently could break encryption methods. This involves calculating logarithms on $\mathbb{Z}_p^*$, so given x finding y such that for group generator g, $x = g^y$. To formulate this as an HSP consider

$$\begin{aligned} f : \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} &\to \mathbb{Z}_p^* \\ (a, b) &\mapsto g^a x^{-b} = g^{a-yb} \end{aligned}$$

where we can see that $f(a_1, b_1) = f(a_2, b_2)$ iff $(a_2.b_2) - (a_1, b_1) = \lambda(y, 1)$ some λ. So using $K = \{\lambda(y, 1) | \lambda \in \mathbb{Z}_{p-1}\}$ works.

Those both involve abelian groups, but a big area of interest is solving the problem for non-abelian groups. Examples of such problems include finding the Automorphism group of a graph. For graph A this means finding $\mathrm{Aut}(A)$ a subgroup of the group of permutations of the vertices of A such that the overall structure (edges) are preserved. This can be formulated quite naturally into an HSP by taking X to be the set of all n vertex graphs, and then to consider the function $f_A(\pi)$ which applies permutation π to A. Clearly the result depends only on the coset of $\mathrm{Aut}(A)$.

Even more famous is the **Graph Isomorphism** problem (GI), which is to check whether or not two labelled graphs are isomorphic (so whether there exists a permutation turning one into the other). This can be converted into an HSP, although the process is more complicated. There also exists few good classical algorithms, although in 2017 someone found an algorithm doing it in quasi-polynomial time.

But how far will we get with quantum algorithms. The abelian case has been fully solved, but unfortunately the non-abelian case (which the above two problems belong to) remains an important unsolved problem in the field. [End of lecture 3]

Briefly, another example of an HSP involving the Dihedral group is how to calculate the shortest vector within a lattice. No good quantum algorithm exists to solve this problem.

Now, let's develop the formalism to describe the quantum algorithm to solve the hidden subgroup problem for any finite abelian group. Contrary to my expectations, we jump straight into representation theory. Well, that is not really contrary to my expectations, but I had imagine it would be more natural to look at the problem as permutations, and to define invariant states from there. Apparently, while it is not mentioned, it does not seem to be the central approach thought of here.

Instead we rely on the fact that for a finite abelian group, all representations can be seen as one dimensional representations $\chi : G \to S^1$, the unit circle embedded in the complex plain. Being one dimensional they are all irreducible, and also satisfy the following three properties:

- $\chi(g) = e^{2\pi i k/|G|}$ some $k \in \mathbb{Z}$
- $\langle \chi_i, \chi_j \rangle = \frac{1}{|G|} \sum_{g \in G} \chi_i(g) \chi_j^*(g) = \delta_{ij}$

- There are exactly $|G|$ distinct such representations, which we may label χ_g
- the restriction of an irreducible representation to a subgroup is also irreducible

A particular application of this is that **trivial representation** $\chi_0(g) = 1$ is orthogonal to all other representations, so in general $\sum_g \chi_k(g) = 0$. This will be helpful later on. Also, note that since this is an abelian group, we will use $+$ to denote the group operation.

Now the crucial advantage that the abelian case has over the non-abelian case is that all elements commute, meaning in particular that all representations can be simultaneously diagonalised, which means a universal shift invariant state exists. In particular, we notice that

$$|\chi_k\rangle = \frac{1}{\sqrt{|G|}} \sum_{g\ inG} \overline{\chi_k(g)}\, |g\rangle$$

Here the complex conjugate is really just convention, but it makes things look similar to Schur's algorithm. Here it is easy to see that the eigenvalues of a shift by g, denoted $U(g)$ is $U_k(g)\,|\chi_k\rangle = \chi_k(g)\,|\chi_k\rangle$.

The Quantum Fourier Transform (QFT) then as expected swaps the standard unit basis with the shift invariant basis meaning that $QFT\,|\chi_g\rangle = |g\rangle$ so considering components approrpiately, and transposing to get the inverse we find

$$QFT\,|g\rangle = \frac{1}{\sqrt{|G|}} \sum_{k\in G} \chi_k(g)\,|k\rangle$$

Now, since we can classify all finite abelian groups as the product of cyclic groups, the following result on representations of such groups is quite valuable

Example 1. For $G = \mathbb{Z}_M$, $\chi_a(b) = e^{2\pi i ab/M}$ describes all representations of G. Even better is that for $G = \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{M_r}$, $\chi_a(b) = e^{2\pi i (a_1 b_1/M_1 + \cdots + a_r b_r/M_r)}$. Furthermore, in this case

$$QFT_G = QFT_{M_1} \otimes \cdots \otimes QFT_{M_r}$$

Now let's start to truly describe the HSP algorithm for finite abelian groups. Here we are given oracle $f : G \to K$ and we work on state space $\mathcal{H}_{|G|} \otimes \mathcal{H}_{|K|}$

1. form the state space $\frac{1}{\sqrt{|G|}} \sum_{g\in G} |g\rangle\,|0\rangle$
2. Apply U_f for form $\frac{1}{\sqrt{|G|}} \sum_{g\ inG} |g\rangle\,|f(g)\rangle$
3. measure second register to get $|g_0 + K\rangle$ where $|K\rangle = \frac{1}{\sqrt{|K|}} \sum_{k\in K} |k\rangle$.

4. Apply QFT and measure to get some state χ_g. Note that here g does not depend on g_0 due to the QFT, which is the strength of this algorithm. [End of lecture 4] From here we can identify representations that become trivial when restricted to K since we see that

$$\begin{aligned} QFT\,|K\rangle &= \frac{1}{\sqrt{|G||K|}}\sum_{l\in G}\left(\sum_{k\in K}\chi_l(k)\right)|l\rangle \\ &= \sqrt{\frac{|K|}{|G|}}\sum_{l\in G \text{ st } \chi_l|_K \text{ is trivial}}|l\rangle \end{aligned}$$

   using the fact that $\sum \chi_l(k)$ is either $|K|$ or 0 depending on whether or not χ_l becomes trivial.

That ends the quantum part of the algorithm, where one essentially identifies the representation χ_g that is irreducible on G but that becomes the trivial representation when restricted to K. This is enough to find the subgroup of interest.

Example 2. For example, one can show that if K has generators $k_1, \ldots, k_M$ for $K = O(\ln(|K|)) = O(\ln(|G|))$ then if we repeat the algorithm $O(\ln(|G|))$ times we can find the generators (so the subgroup) with probability $> 2/3$.

Example 3. In the $G = \mathbb{Z}_{M_1} \times \cdots \times \mathbb{Z}_{M_r}$, one can find that finding the group is equivalent to solving a set of linear equations.

Now what goes wrong in the non-abelian case? Essentially two things can break down. Firstly, we might not be able to implement the quantum fourier transform in an efficient way. Secondly, because elements don't commute, we cannot simultaneously (block) diagonalise the basis into a set of shift-invariant states. This means one cannot fully recover the representations as one might wish. It has nevertheless been proven when the subgroup is normal, one can solve this efficiently (Hallgren, Russel, Ta-Shma 2003 SIAM J Comp The Hidden Subgroup Problem and Quantum Computation Using Group Representations), and that in fact, even in the most general non-abelian case the information we deduce is sufficient to find K, however there is no efficient way to deduce K from the information we can gain using the more coarse-grained approach here (Ettinger, Hoyer and Knill, Hidden Subgroup States are Almost Orthogonal) (so query complexity is not high, but deducing the right information from it is high). [End of lecture 5]

2 Quantum Phase Estimation

The next problem/aglorithm we discuss is **Quantum Phase Estimation** which is how to find an eigenvalue $e^{2\pi i\phi}$ of eigenstate $|v_\phi\rangle$ of an arbitrary unitary operator U. To do so we need the controlled operator $c-U$ such that $c-U\,|0\rangle\,|\xi\rangle = |0\rangle\,|\xi\rangle$ and $c-U\,|1\rangle\,|\xi\rangle = |1\rangle\,U\,|\xi\rangle$.

Now the issue is that we cannot actually contruct $c - U$ given U if we only have one of its eigenstates. If we have a full implementation, for example in the form of a circuit, we could simply replace every gate with a controlled gate, and that would do the job, however, if we just have a black box operator, there is an ambiguity that arises from the fact that we may consider U and $e^{\imath\alpha}U$ to be same operator. If so we find that

$$c - e^{i\alpha}U(|0\rangle + |1\rangle) |\xi\rangle = |0\rangle |\xi\rangle + |1\rangle e^{i\alpha}U |xi\rangle$$

so these differ by a relative phase, meaning the operator $c - U$ is truly different from $c - e^{i\alpha}U$ despite U and $e^{i\alpha}U$ being the "same". To resolve this, it turns out it is sufficient as long as we have one extra eigenstate $|\alpha\rangle$ with eigenvalue $e^{i\alpha}$, since then we can implement the algorithm as follows:

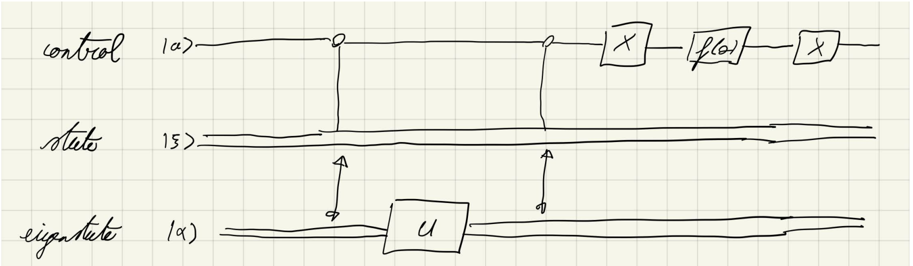

Figure 1: Controlled U

Here the circle with the arrows indicate controlled swap operations between $|\alpha\rangle$ and $|\xi\rangle$, double lines indicate an arbitrary k-qubit state, while a single line denotes a single qubit state. X denotes the swap operator between $|0\rangle$ and $|1\rangle$ and $f(\alpha)$ denotes the matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}.$$

Considering case-by-case we then find that

- when the control is $|0\rangle$ we get $e^{i\alpha - i\alpha} |0\rangle |\xi\rangle$.
- when the control is $|1\rangle$ we get $|1\rangle |\xi\rangle$

(typo in the picture, θ should be $-\alpha$). Now for our actual algorithm we don't need $c - U$ instead we need (what I'll call) $N - U$ which takes the form $N - U |x\rangle |\xi\rangle = |x\rangle U^x |\xi\rangle$. This can be implemented as seen below for decimal expansion $x = x_{n-1} \dots x_1 x_0$.

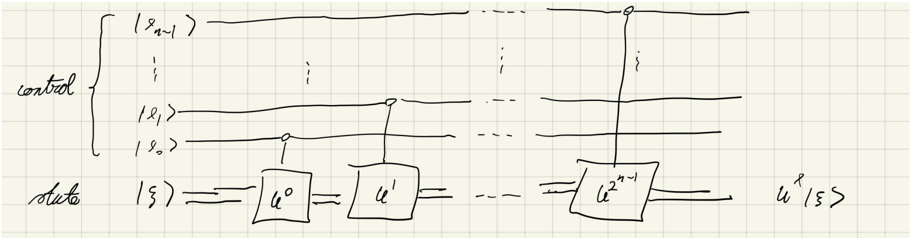

Figure 2: Generalised Controlled U

Now that we've implemented that, the actual quantum phase estimation algorithm is quite easy to implement:

1. start with $|+\rangle\,|v_\phi\rangle$ for $|+\rangle = H^n\,|0\rangle$
2. apply $N - X$ to get $\frac{1}{2^{n/2}}\sum_x e^{2\pi i\phi x}\,|x\rangle = |A\rangle$.
3. Apply $QFT_{2^n}^{-1}$ and measure to get $y_{n-1}\dots y_1 y_1$. Our estimate for ϕ the is
$$\phi \approx 0.y_0\dots y_{n-1}$$

This is exact if $\phi = k/2^n$ some $0 \le k < 2^n, k \in \mathbb{Z}$. If it's not, we have the following handy result

Theorem 1. If the algorithm yields estimate $\theta \approx \phi$ then

- the probability of θ being the estimate to ϕ that is closest possible to ϕ in binary is at least $4/\pi^2 \approx 0.4$.
- $\mathbb{P}(|\phi - \theta| \ge \epsilon) \le \frac{1}{2^{n+1}\epsilon}$.

In particular, we notice that if we want at least m bit accuracy with certainty $1-\eta$ then we should run the algorithm with n bits such that $n = m + \log_2(1/\eta)$. It is interesting here, to me, that the quantity n is sum of the desired accuracy m and the chance of that accuracy (expressed in binary) $\log_2(1/\eta)$. The fact these are equivalent in a sense is curious. [End of lecture 6]