Quantum Computation

March 29, 2016

Notation

State $|\alpha\rangle$

Dual Vector $\langle\alpha|$

Hibert Space $\mathcal{H}$

Computational Basis

- $|0\rangle$
- $|1\rangle$
- $|2\rangle$ etc.

1 Elementary Concepts

A state $|\alpha\rangle$ is a vector in a Hilbert space $\mathcal{H}$. The smallest non-trvial space is $\mathbf{C}^2$, with two basis states. The most common basis choice is called 'computational', which has matrix representaiton

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$\mathbb{C}^2$ is commonly reffered to as a qubit. Spaces combine under the tensor product[1]; a system of n qubits lives in space $\mathbb{C}^{2^n}$, with 2^n basis vectors. 'Qudit' is used to refer to a $\mathbb{C}^d$ dimensional system.

The tensory product is linear, as is the action of an operator $\hat{O}$.

$$\alpha A \otimes \beta B = \alpha\beta(A \otimes B) \tag{1}$$

$$\hat{O}(\alpha|\psi\rangle + \beta|\phi\rangle) = \alpha\hat{O}|\psi\rangle + \beta\hat{O}|\phi\rangle \tag{2}$$

Aside: We define the Schur-Hadamard or 'Bad-Student' product as the direct, entrywise product of two matrices. For a unitary matrix, this product gives a doubly stochastic matrix, as each row and column sums to 1. It is also 'Unistochastic', as the resulting matrix is unitary. Unistochastic matrices are distinct as they cannot be decomposed into permutation matrices (as doubly stochastic matrices can).

Evolution

Axiomatically, evolution in Qauntum mechanics is reversible. These operators are represented by unitary matrices U, generated by the system Hamiltonian H

$$U(t) = e^{-iHt} \tag{3}$$

[1] This is in contrast to classical probabilities which combine by the direct sum

Superposition and Denisty Operators

A vector in the Hilbert space represents a state. 'Pure' states are extremal points in the the space, and a set of mutually orthogonal prue states can form a basis. Any general state can be decomposed into a mixutre of basis states

$$|\psi\rangle = \sum_n c_n |\phi_n\rangle \tag{4}$$

where the amplitudes c_n have the property $\sum_n c_n c_n^* = 1$.

Any state in a d-dimensional space has a corresponding density operator ρ, represented by a $d \times d$ unitary matrix

$$\rho = |\psi\rangle\langle\psi| \tag{5}$$

, where the diagonal entries are called the 'populations', and give the probability of a given basis state, and the off-diagonal entries are called the 'coherences'.

In quantum computation, we extend the computational basis, defining a general basis vector as a binary string $x \in \{0,1\}^n$, such that

$$|x\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle \tag{6}$$

where $x_i \in \{0,1\} \forall i$.

Gates

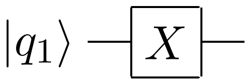
Gates are the terms used for unitary operations acting on one or more qubits during a computations. Some elementary gates, their quantum circuit and matrix representations are given below.

Pauli X

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Flips a computational basis state. Has eigenvalues

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

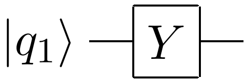


Pauli Y

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Flip plus phase difference. Has eigenvalues

$$|+i\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ i \end{pmatrix} \qquad |-i\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix}$$



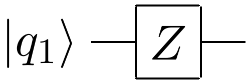
Pauli Z

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Add a phase difference to the $|1\rangle$ state. Eigenvalues are the computaitonal states.

Phase Gate R_θ

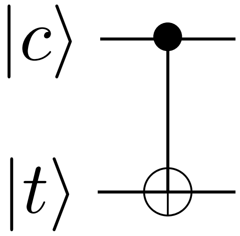
$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$



CNOT

$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$$

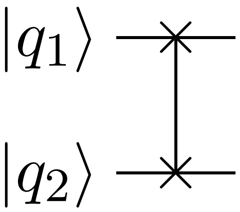
Flips the target qubit, conditioned on the control qubit.



SWAP

$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

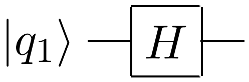
Swap qubits 1 and 2.



Hadamard

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

Rotates between the equator ($|+,-\rangle$) and poles ($|0,1\rangle$) of the Bloch sphere.



No Cloning

This is a fundamental theroem of quantum information and a consequence of linearity of quantum mechanics. Consider a ‘cloning’ unitary U such that

$$U(|0\rangle \otimes |0\rangle) = |0\rangle \otimes |0\rangle \tag{7}$$

$$U(|1\rangle \otimes |0\rangle) = |1\rangle \otimes |1\rangle \tag{8}$$

$$\tag{9}$$

Then we might expect to clone an arbitrary qubit state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$. But,

$$U(\alpha\,|0\rangle \otimes |0\rangle + \beta\,|1\rangle \otimes |0\rangle) = \alpha\,|00\rangle + \beta\,|11\rangle \tag{10}$$

$$\neq |\psi\rangle \otimes |\psi\rangle \tag{11}$$

as $|\psi\rangle \otimes |\psi\rangle = \alpha^2\,|00\rangle + \alpha\beta(|01\rangle + |10\rangle) + \beta^2\,|11\rangle$

2 Universal Computing

We can conceptualise a computation as a boolean function $f : \{0,1\}^n \to \{0,1\}^m, n \geq m$, and this operation can be broken up into a circuit if there exist a sequence of one- and many-qubit gates U_i such that

$$U_n U_{n-1} \cdots U_1 |\psi_0\rangle = |\psi_{out}\rangle$$

where $|\psi_0\rangle$ and $|\psi_{out}\rangle$ are computational states belonging to $\mathbb{C}^n$ and $\mathbb{C}^m$ respectively.

If $n > m$, then the computation is said to be irreversible. The Landauer principle states that there is a thermondynamic cost $k_B T \ln_2$ associated to erasing a bit, and so an irrevesrible circuit generates $(n-m)k_B T \ln_2$ of energy.

Reversible computation, clasically, is achieved by adding additional ancillae bits, and using the Toffoli or 'CCNOT' gate.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{12}$$

It can be shown that this gate is universal for classical computation, meaning with enough bits and a sequence of Toffoli's any classical computation can be implemented.

It turns out that, for quantum computation, we only need to add the Hadamard gate to achieve universal computation. This makes it clear that quantum computing is an extension of the classical, and namely that $P \subseteq BQP$.

The Solovay-Kitaev Theorem

While the set $\{H, CNOT\}$ is one example, there are many different universal sets, some of which are 'overcomplete'. Examples include $\{\text{All Single Qubit Gates}, CNOT\}$ and $\{CNOT, H, R_{\frac{\pi}{4}}\}$.

The Solovar-Kitaev Theorem defines what it means for a set of gates to be 'universal' for quantum computing.

A set of t normalised, unitary matrices $\{U_i\}$ is an ϵ-approximation of a Unitary U if

$$\|U - U_t U_{t-1} \cdots U_1\| \leq \epsilon \tag{13}$$

where $\|U\| \equiv \max \frac{\|U|\psi\rangle\|}{\||\psi\rangle\|}, \| \, |\psi\rangle \, \| \equiv \sqrt{\langle\psi|\psi\rangle}$.

The Solovay-Kitaev theorem states that a set of gates is universal if it can ϵ approximate any Unitary. The theorem also states that any universal set is ϵ-equivalent to each other, meaning that any 'Quantum speedup' can be achieved irrespective of our choice of gates.

3 The Deutsch-Jozsa Algorithm

This is the protoypical quantum algorithm, demonstrating how a technique called phase-kickback, in combination with superposition, can be used to significantly reduce the 'query complexity' of a problem.

Consider a function $f : \{0,1\}^n \to \{0,1\}$. We are given a black box or 'oracle' that can evaluate this function for any input. We want to characterise if the function is 'balanced' (outputs 0 for half the inputs or 1 for the other half), or 'constant' (output always 0 or 1). How many times do we have to query the oracle to find out?

Clasically, we can show that for $N = 2^n$ possible inputs, we need at least $\frac{N}{2} + 1$ queries. But quantum algorithms require only 1!

Quantum Parallelism

Consider a 'register' of n qubits, all initialised in the $|0\rangle$ state. We can then consider applying an n-fold Hadamard to each of states $H^{\otimes n}|00\cdots 0\rangle$. An n-fold Hadamard on superposition of binary strings $|x\rangle$ has the effect

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_y (-1)^{x\cdot y}|y\rangle \tag{14}$$

where

$$x \cdot y \equiv x_1 \wedge y_1 \oplus x_2 \wedge y_2 \oplus \cdots \oplus x_n \wedge y_n.$$

For the state $|0\rangle^{\otimes n}$, this initialises the register in a uniform superpostion of all binary strings $x \in \{0,1\}^n$

$$H^{\otimes n}|00\cdots 0\rangle = \frac{1}{\sqrt{2^n}}\sum_x |x\rangle$$

often denoted simply $|x\rangle$.

To perform reversible computing, like in the classical case, we need additional qubits to keep track of the outputs. If we consider a second register $|0\rangle^{\otimes m}$, we can construct an oracle gate $\hat{O}_f$ for a function $f : \{0,1\}^n \to \{0,1\}^m$ such that:

$$\hat{O}_f(|x\rangle \otimes |00\cdots 0\rangle) = \frac{1}{\sqrt{2^n}}\sum_x |x\rangle \otimes |0 \oplus f(x)\rangle\,. \tag{15}$$

Effectively, we now have a superposition of the inputs, and the evaluation of the function for each one. This ability to evaulate the oracle for all inputs in a single query is commonly reffered to as Quantum Parallelism.

The Algorithm

The Deutsch-Jozsa algorithm is the name given to the generalisation of this constant/balanced problem for n-bit input strings; the Detusch algorithm refers specifcally to the $n = 2$ case.

In this algorithm, the second register is instead intialised into the $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ state.

If the function $f(x)$ evaluates to 1, then the second register becomes (dropping normalisation): $|0 \oplus 1\rangle - |1 \oplus 1\rangle = |1\rangle - |0\rangle = -|-\rangle$ i.e that state picks up a π phase.

We then apply a set of Hadamards to the first register, and we can throw away the second register

$$\sum_x H^{\otimes n}|x\rangle \otimes (-1)^{f(x)} \tag{16}$$

$$= \sum_y (\sum_x (-1)^{f(x)+x\cdot y}|y\rangle) \tag{17}$$

$$= \begin{cases} \pm|0\rangle^{\otimes n} & \text{consntant} \\ |\phi\rangle \neq |0\rangle^{\otimes n} & \text{balanced} \end{cases} \tag{18}$$

To evaluate, we simply need to read out each of the qubits in the first register. If we encounter a $|1\rangle$, we know the function was balanced, otherwise it was constant.

This algorithm is deterministic. Not only that, but it requires only a linear number of Hadamard gates and single oracle query. It has been shown that any boolean oracle can be implemented in a polynomial number of gates, and so this algorithm is computationally efficient.

Aside: Boolean Functions and Quantum Circuits A boolean function $f : \{0,1\}^n \to \{0,1\}^m$ has an 'algorithm', which describes how to create a family of circuits C that can evaluate the function for different input sizes n. This algorithm is efficient if these circuits have a size that scales polynomially or less with n. The circuits are typically represented as acyclic graphs using irreversible operations AND, NOT and OR, along with the FANOUT gate.

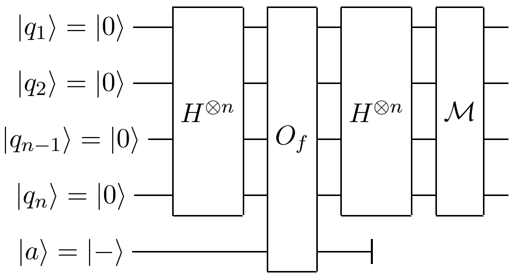

$|q_1\rangle = |0\rangle$
$|q_2\rangle = |0\rangle$
$|q_{n-1}\rangle = |0\rangle$
$|q_n\rangle = |0\rangle$
$|a\rangle = |-\rangle$
$H^{\otimes n}$
O_f
$H^{\otimes n}$
$\mathcal{M}$

Figure 1: Simplified Circuit for the Deutsch-Jozsa algorithm.

These circuits have a reversible implementation, as these elementary operaitons can all be approximated using a sequence of Toffoli gates with enough ancillae bits.

If we consider an archetypal operation e.g. modulo addition, then such a circuit will need n bits to define the Modulo, n for input x and n for input y. However, the ouput is only n bits, and so a reversible example must have an additional n 'worker' bits w, such that the inputs are left unchanged and the worker bits contain the result of the computation.

Generally, for a function $f : \{0,1\}^n \to \{0,1\}^m$ with an irreversible circuit C, there exists a reversible circuit R with n input bits and l ancilla bits, which gives an output with m-data bits and $n + l - m$ 'garbage' bits.

This gives a template to implement an oracle; with an additional m 'readout' bits, the result can be mapped to these bits with a sequence of CNOTs, and then the inverse operation R^{-1} applied to restore the input and the ancillae.

---

4 Hidden Subgroup Problems

The Hidden Subgroup Problem is a general state of a class of computational problem. A certain case of these problems, which are clasically known to be in NP, can be solved in polynomial time with a quantum algorithm.

Formal Statement

For a group G and a subgroup $H : |H| \leq |G|$, we say that a function $f : G \to X$ onto the set X 'hides the group' H if $\forall g_1, g_2 \in G, f(g_1) = f(g_2) \leftrightarrow g_1 H = g_2 H$. Equivalent, we say that f is constant within the cosets of H.

Given such a function f, implemented as an oracle using $O(\log |G| + \log |X|)$ bits, how many queries are required to determine a generating set for H?

Simons's Problem

Simon's Problem is an example of the Hidden Subgroup problem, similar to the Deutsch-Jozsa problem. In this case, the gidden group is a single element $s \in \mathbb{Z}_2^N$, the group of N bit binary strings.

We are given a binary function $f : \{0,1\}^n \to \{0,1\}^n$, with the 'promise' that it is periodic with period s. Namely, for two string $x, y \in \mathbb{Z}_2^n$,

$$f(x) = f(y) \implies x \oplus y = s \implies y = x \oplus s\,. \tag{19}$$

The initial step of the algorithm is idetical to the Deutsch-Jozsa algorithm; a set of n data qubits are initialised in a coherent superposition of all inputs, and an oracle is acted on the n data qubits and n ancillae to give the state

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle \, |f(x)\rangle \tag{20}$$

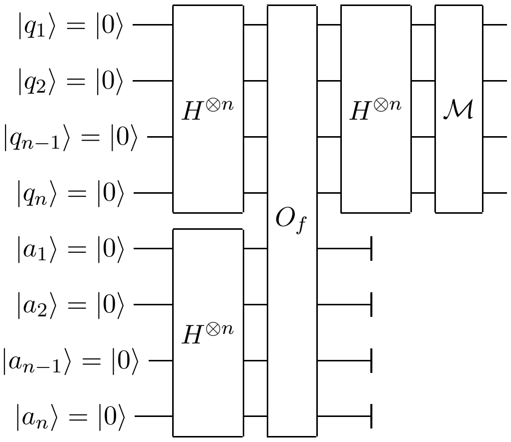

$|q_1\rangle = |0\rangle$
$|q_2\rangle = |0\rangle$
$|q_{n-1}\rangle = |0\rangle$
$|q_n\rangle = |0\rangle$
$|a_1\rangle = |0\rangle$
$|a_2\rangle = |0\rangle$
$|a_{n-1}\rangle = |0\rangle$
$|a_n\rangle = |0\rangle$
$H^{\otimes n}$
$H^{\otimes n}$
O_f
$H^{\otimes n}$
$\mathcal{M}$

Figure 2: Simplified Circuit for Simon's Algorithm

which can be written as a sum over the range of the function f and the hidden group s as

$$\frac{1}{\sqrt{2^n}}\sum_{r\in R}(|x\rangle + |x\oplus s\rangle)\,|r\rangle$$

We then apply a second set of Hadamard operations on the first register, and perform a computational basis measurement on the first register. The propbability of obtaining a given string y on readout is then given by

$$\|\frac{1}{2^n}\sum_{r\in R}((-1)^{x\cdot y} + (-1)^{(x\oplus s)\cdot y})\,|r\rangle\,\|^2 \tag{21}$$

which gives

$$P(y) = \begin{cases} 2^{-(n-1)} & \text{if}\, s\cdot y = 0 \\ 0 & \text{if}\, s\cdot y = 1 \end{cases} \tag{22}$$

Or, if $s = 0$, this is equal to a uniform superpositon over all possible strings.

Determing s then requires an iteration of this algorithm to generate $n-1$ strings, which is enough to solve for $s\cdot y = 0$.

The problem here is that the string we obtain on readout is random, and so there is a non-zero probability that we obtain a duplicate string on a given round of the algorithm.

For a given run of the algorith, the probability of obtaining $n-1$ linearly independent string is bounded by

$$\prod_{k=1}^{\infty} 1 - \frac{1}{2^k} > \frac{1}{4}$$

. If we repeat our $n-1$ trials $4m$ times, then we have a bounded failure probability

$$(1-\frac{1}{4})^{4m} < e^{-m}$$

which, for $m = 10$, is $\frac{1}{20000}$. Thus, with a linear number of queries, we can find s.

Quantum Fourier Transform

Before discussing other instances of the HSP, we will consider in detail the Quantum Fourier Transform, a key element of these algorithms.

The general statement of a fourier transform is that it takes an input vector of N complex numbers x, and transforms it onto a vector y such that

$$y_k \equiv \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} x_j e^{2\pi ijk/N} \tag{23}$$

Quantum mechanically, this corresponds to a change of basis such that

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |K\rangle$$

We can write $e^{2\pi ijk/N}$ as the Nth root of unity, $\omega_N^{i\times j}$, and write the Fourier transform as F_N as a matrix $[F_N]_{i,j} = \omega_N^{i\times j}$, noting that $\omega_N^M = \omega_N^{M \mathrm{mod} N}$ for $M > N$. This matrix is unitary.

We can write the action of the QFT on a computation state $|x_1x_2\cdots x_n\rangle$

$$QFT\,|x\rangle = \frac{(|0\rangle + e^{2\pi i0.x_n}|1\rangle) \otimes \cdots (|0\rangle + e^{2\pi i0.x_1x_2\cdots x_n}|1\rangle)}{2^{\frac{n}{2}}} \tag{24}$$

where $0.x = \frac{x}{2^{|x|}}$.

The quantum fourier transform F_N can be efficiently decomposed into the Hadamard gate, which is equivalent to F_2, controlled-phase gates and the transform F_{N-1} The phase gates are denoted R_k where

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}.$$

An example is shown below for F_3.

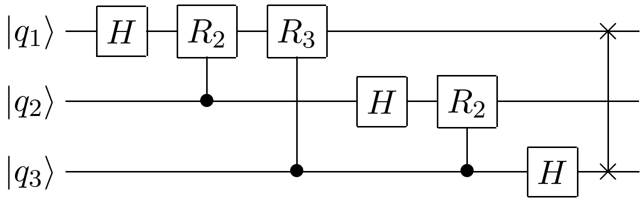
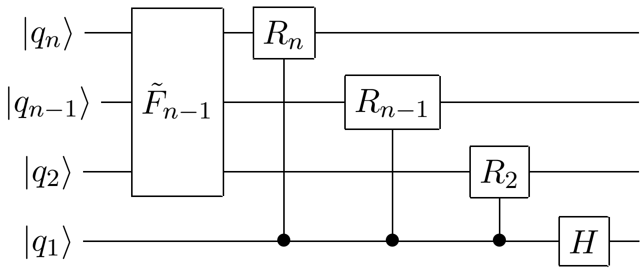


Figure 3: Quantum Circuit for the F_3 QFT

More generally, we consider a 'flipped' QFT $\tilde{F_n}$ acting on a state $|\tilde{x}\rangle = |x_nx_{n-1}\cdots x_1\rangle$

$$\tilde{F_n}\,|\tilde{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=1}^{2^n} \omega_{2^n}^{jk} |x_1x_2\cdots x_n\rangle \tag{25}$$

Then we can write $\tilde{F_n}$ as $\tilde{F_{n-1}}$, and a squence of controlled phase gates and a final Hadamard. A simplified circuit, skipping many qubits, is shown in the figure below. F_n can be obtained from $\tilde{F_n}$ by adding a squence of swap gates to change the order of the bits.



From the circuit above and the circuit for F_3, we can build any QFT gate using $\frac{n(n+1)}{2}$ phase gates and Hadamards, and a final $\frac{n}{2}$ swap gates, meaning it can be achieved in $\Theta(n^2)$ gates.

Phase Estimation

Consider an oracle implementing some large unitary operation U. We know that this $2^n \times 2^n$ matrix has $2^n = N$ orthonormal eigenvectors with associated eigenvalues $e^{2\pi i\theta_i}$. Given a state $|\psi\rangle$ that is promised to be an eigenvector of U, can we find its eigenvalue phase θ_i?

We begin with two registers. The first register has n qubits, the number of which limits our precision of the estimate for θ_i, and the second register stores $|\psi\rangle$. We seek to implement a channel $\Lambda : |k\rangle \otimes |\psi\rangle \to |k\rangle \otimes U^k |\psi\rangle$.

We then perform the usual trick of initialising the first reigster in a coherent superposition of all $N = 2^n$ inputs, and applying this operation. We can throw away the state $|\psi\rangle$ as it is a product with the result, leaving

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i k \theta_i} |k\rangle \tag{26}$$

Case 1: $\theta_i = \frac{j}{2^n}$

In this case, our phase can be exactly represented in the bits we use for the first register. The state we have reduces to

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i k \theta_i} |k\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \omega^{jk} |k\rangle \tag{27}$$

where ω is the 2^nth root of unity. Thus, we need simply act the inverse QFT F_n^*, and we obtain a computational state $|x_1 x_2 \cdots x_n\rangle = |x\rangle$, where $\theta_i = 0.x$!

Case 2: $\theta_i \neq \frac{j}{2^n}$

If θ_i cannot be exactly represented in n bits, then we consider the best approximation $b : \theta_i - \frac{b}{2^n} = \delta \in [0, 2^{-n}]$. In this case, applying F_n^* gives the state

$$\frac{1}{\sqrt{2^n}} \sum_{k,l=0}^{2^n-1} e^{-2\pi i k l / 2^n} e^{2\pi i \theta_i k} |l\rangle \tag{28}$$

The amplitude of a state $|(b+l) \mod 2^n\rangle$ can be shown to equal

$$\frac{1}{2^n} \left(\frac{1 - e^{2\pi i (2^n \delta - l)}}{1 - e^{2\pi i (\delta - l/2^n)}} \right)$$

We can show that the probability of an output string $m : |m - b| > e$, the error tolerance, is $\frac{1}{2(e-1)}$. Thus, to approximate θ_i to t bits with $p_{success} \geq 1 - \epsilon$ requires $n = t + [\log(2 + \frac{1}{2\epsilon})]$ qubits.

Assuming we don't know an eigenstate of U, this can be shown to work for a state $|\psi\rangle = \sum_n c_n |u_n\rangle$, giving the associated eigenvalue θ_n with probability $|c_n|^2$.

Order Finding

Order finding is a number theoretic problem that is most commonly studied in the context of prime factorisation, namely finding two prime numbers p, q such that $N = pq$.

Given a number N and the range of position integers below it $\mathbb{Z}_N = \{0, 1, 2 \cdots N\}$, we define $\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N : gcd(a, N) = 1\}$. This, combined with modulo multplication, forms a group such that there is one unique element $b : ab \mod N = 1$. Each element a also has an order, $r > 0, \in \mathbb{Z}_N : a^r \mod N = 1$. This order $r = |\mathbb{Z}_N^*|$.

Clasically, finding this order is computationally hard. However it turns out this problem can be reduced efficiently to solving a phase estimation problem, which as we showed requires a single quantum oracle query.

We consider a reversible transformation

$$\Lambda_n(M_a) |k\rangle |x\rangle = |k\rangle M_a^k |x\rangle = |k\rangle \left| a^k x \text{mod} N \right\rangle$$

which has eigenvectors

$$|\psi_j\rangle = \frac{1}{\sqrt{r}} \sum_{l=0}^{r} \omega_r^{-jl} \left| a^l \text{mod} N \right\rangle \tag{29}$$

and associated eigenvalues $e^{\frac{2\pi ik}{r}}$. This makes it clear that, by finding the eigenvalue of such a state using phase estimation, we can then obtain r.

The operation above can be constructed via 'modular exponentiation', a classical boolean algorithm. We can use the method described at the end of Section 3 to calculate $\left|a^k \text{mod} N\right\rangle$, and then multiply this result by the second register, before applying the inverse operation to restore inputs and ancillae used. This requires $O(n^3)$ operations to complete, as $|k\rangle$ has n bits and for each bit we require $O(n^2)$ operations to calculate $x^{k_l 2^{l-1}} \text{mod} N$ for each bit of k.

For n qubits we have some $j \in \{0, 1 \cdots 2^n\}$, such that $r \approx \frac{2^n}{j} + \frac{1}{2}$. We want to be within ϵ of r, which will give the correct r by rounding to the nearest integer if $|\epsilon| \leq \frac{1}{2N^2}$, which in turn implies $n = 2\log(N)$.

Alternatively, we can use the fact that $\frac{k}{r}$ must be rational, as it is the product of two integers, we can obtain r without requiring knowledge of k. Any given rational number x has an expression as a sequence of integers $[a_0, a_1, \cdots a_m]$ such that

$$x = a_0 + \cfrac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \cdots}}} \tag{30}$$

And so, we can use the continued fractions algorithm to identify this expansion and thus obtain $\frac{k}{r}$, in $O(n^3)$ operations.

How can we obtain these (mutually orthogonal) eigenstates of the modular exponentiation operator? We can exploit that fact that

$$|11 \cdots 1\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |\psi_j\rangle \tag{31}$$

and so after performing the phase-estimation procedure, when reading out the first register we pick one phase θ_j associated to $|\psi_j\rangle$ uniformly at random.

Given that we can make the error of phase estimation arbitrarily small for a small increase in the circuit size, the main source of error is if j and r have a common factor, and so the algorithm will return r' which is not necessarily r. However, the probability that j and r are co-prime is very high, and in these intances the continued fractions method is guaranteed to work.

This is because there are at least $\frac{r}{2\log r}$ prime numbers less than r, meaning the probability j is prime is $> \frac{1}{2\log r} > \frac{1}{2n}$, and so $2n$ repetitions will suffice to find the correct r. Another method which also requires an additional $O(n)$ repetitions, is to repeat the procedure to find the order of $a' = (a^{r'}) \text{mod} N$.

Alternatively, we can take the trials r_i froma given run of the algorithm, and as long as the s_i do not have common factors, we can find r from the lowest common multiples of r_i. The probability this succeeds in two trials is at least $\frac{1}{4}$.

The total cost of this procedure is then $O(n^3)$, most of which is consumed during the modular exponentiation step.

Shor's Algorithm

Shor's algorithm can achieve prime factorisation, a problem clssicaly solvable only in sub-exponential time, also in $O(n^3)$ operations, an exponential speed up.

This is because there exists a 'reduction' of prime factorisation to the problem of order finding, and we can use the above algorithm to solve the problem in $O(n^3)$.

For an integer input N, the result is easy if N is prime or a 'prime power' p^k, or if N is even. We can consider a recursive algorithm that keeps going until we find some non-trivial case, and which then outputs two factors $N = pq$. The procedure can be repeated for these factors, to find all the prime factors of any input N.

Consider picking a random number $a \in 2, 3, \cdots, N-1$. This number has a unique order in $\mathbb{Z}_N^*$, such that N is dived by $a^r - 1$ (we write $N|a^r - 1)$).

This implies that $N|(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$. We are only interested if $x^{\frac{r}{2}} \text{mod} N \neq -1$, and if r is even.

Assuming this isn't the case, computing the greatest common denominator of $a^{\frac{r}{2}} \pm 1$ and N gives a non-trivial factor.

We can thus proceed by picking a number a at random, and trialing the algorithm. The probability of success for a given run of the order-finding subroutine is $\frac{1}{2}$. Finding the greatest common denomintaor is possible classically in $O(n^2)$ gates, and we typically need $O(\log(n))$ repitions, menaing the limiting factor is the $O(n^3)$ required by the order-finding routine.

Interpreation as a HSP

All of the algorithms dicussed so far can be interpreted as a Hidden Subgroup problem. For an abelian group, qauntum algorithms can solve the HSP in poly time with respect to the order of the whole group to be searched $|G|$.

The general procedure is to use the QFT to intialise a uniform superposition of the group elements

$$\frac{1}{\sqrt{|G|}} \sum_{g} |g\rangle$$

and then apply an oracle which implements the function f hiding the subgroup H, giving

$$\frac{1}{\sqrt{|G|}} \sum_{g} |g\rangle |f(g)\rangle$$

We can rewrite $|f(g)\rangle$ in the Fourier basis

$$|f(g)\rangle = \frac{1}{\sqrt{|G|}} \sum_{l} e^{2\pi i l \frac{g}{|G|}} \left|\hat{f}(l)\right\rangle$$

But as f is constant in the cosets of the subgroup H, $\left|\hat{f}(l)\right\rangle$ has non-zero amplitude only for the cases

$$\sum_{h \in H} e^{-2\pi l \frac{h}{|G|}} = |H|$$

meaning we can apply the inverse QFT to obtain l, and from this obtain elements of H sufficient to find a generating set.

While this understnding works for the cases of Phase estimation, or for simpler cases like Deutsch-Jozsa and Simon's problems, it is difficult to conceptualise how it works generally. The key lies in the fact that any finite abelian group is isomorphic to a product of cyclic groups with prime power orders p_j, such that we can find l_j in the expression

$$e^{2\pi i l \frac{g}{|G|}} = \prod_{j} e^{2\pi i l_j \frac{g_j}{p_j}}$$

from basic phase estimation, and from there obtain l and the group H.

5 Grover's Algorithm

Grovers Algorithm is a quantum computing routine broadly described as a 'quantum search. It allows us to search through a dataset to find entries that match a criteria, encoded by a boolean function. This method can be applied to a broad range of problems, including 'explicit searching, looking for particular records in large databases e.g. web search or real-time processing of financial transactions, or for 'implicit searching, seeking for example the solution that satisfies an NP-Complete problem e.g. satisfiability, or geometric optimisation problems such as the Travelling Salesman.

Grover's Algorithm requires a pair of gates to perform the search. The first is an oracle, which implements a function $f : \{0,1\}^n \to \{0,1\}$, returning 1 only if the input 'matches' the search criteria. This function could be a polynomial-time verifier for an implicit search problem, or a 'comparison' for explicit search.

Using an additional ancilla bit, initialised in the $|-\rangle$ state, we can exploit phase-kickback to mark our solution amongst a coherent superposition of input states.

$$\hat{O}_f |x\rangle |-\rangle = (-1)^{f(x)} |x\rangle |-\rangle \tag{32}$$

We then apply a second gate, called the interpreter or diffusion operator, which inverts the amplitudes of a state around the 'average'

$$\alpha'_k = \langle\alpha\rangle - \delta_k$$

where $\delta_k = \alpha_k - \langle\alpha\rangle$.

The action of these two gates is to 'boost' the amplitude of the state that matches the search criteria, and supress the others. By repeating this procedure, we can increase our probability of finding the 'true' state by reading out the first register. The diffusion operator is performed applying an n-fold Hadamard after the oracle, phase-shifting any state except $|00\cdots 0\rangle$, before reapplying the Hadamard.

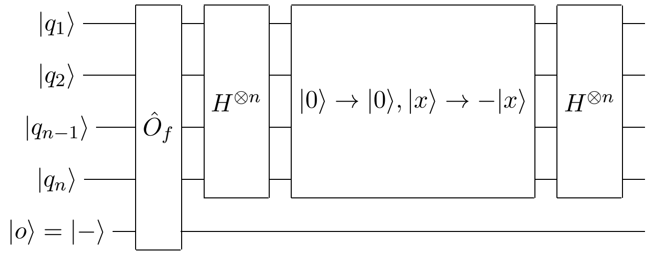


Figure 4: Simplified circuit showing the Grover iteration.

Geometrically, we can estimate how many repetitions of the 'Grover iteration' we need to maximise the probability of finding a solution to the search problem. Assume there are M matches in a data set of N.

We split the space of all computational states into the 'target' and 'junk' subspaces

$$|t\rangle = \frac{1}{\sqrt{M}} \sum_{m\in M} |m\rangle$$
$$|\chi\rangle = \frac{1}{\sqrt{N-M}} \sum_{x\notin m} |x\rangle$$

and the intial coherent superposition can be written

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle = \sin\left(\frac{\theta}{2}\right) |t\rangle + \cos\left(\frac{\theta}{2}\right) |\chi\rangle \tag{33}$$

where $\frac{\theta}{2} = \sin^{-1}(\sqrt{\frac{N-M}{N}})$. Plotting this vector in $|t\rangle, |\chi\rangle$ space, the action of the grover iteration is to first flip the vector in t, χ space around the χ axis, and then flip this fector around the initial state. The total action is a rotation through an angle θ.

The amplitude of the target states is maximal, then, when we have rotated through to an angle $\frac{\pi}{2} \approx \theta' = \frac{2k+1}{2}\theta$ for k applicaitons of the Grover iteration. This means we need approximately

$$k = \lfloor \frac{\cos^{-1}(\sqrt{\frac{M}{N}})}{\theta} \rfloor \tag{34}$$

applications to maximise the probability of finding a target state.

A simpler bound can be obtained by rearranging the expression above to obtain the upper bound

$$k \leq \lceil \frac{\pi}{4}\sqrt{\frac{N}{M}} \rceil. \tag{35}$$

For $M = 1$, this means we need $O(\sqrt{N})$ queries, improving over the $\frac{N}{2}$ searches required for a classical brute force search. This is only a quadratic speedup, rather than the exponential speed up offered by Shor's algorithm, but could be used to

Implicit and Explicit Search

For an implicit search problem, the oracle gate is simply the circuit that implements the efficient polynomial verifier for the problem. Thus, we expect Grover's algorithm to have an efficient implementation. But what about explicit search problems?

We wish to find the target $|t\rangle$ in the register. Assume it is stored in l bits, and there are $N = 2^n$ entries. We need four registers then

- An n qubit register to hold the index
- An l qubit register to hold t
- An l qubit register to hold 'loaded' data
- A single qubit to probide phase kickback

This method suggests a quantum memory, which would allow us to load superpositions of data in to memory, and to implement the oracle by comparing the loaded data to the target t. This would need only $O(\sqrt{N})$ load operations, compared to $O(N)$ clasically.

But, this scheme could also operate with a classical memory, provided we have access to a quantum addressing system. Such a system would need an additional $O(Nn)$ switches.

Optimality

Quantum Search is in fact the optimal algorithm for unstructured search. To illustrate this, consider an initial state $|\psi\rangle$, and an oracle that marks a single target state x. We apply a sequence of unitary operations U_i and the oracle O_x k times

$$\begin{aligned}|\psi_k^x\rangle &= U_k O_x U_{k-1} \cdots U_1 O_x |psi\rangle \\ |\psi_k\rangle &= U_k U_{k-1} \cdots U_1 |\psi\rangle\end{aligned}$$

We want to bound the quantity $D_k \equiv \sum_x \|\psi_k^x - \psi_k\|^2$, as the probability of finding the target is better the larger D is. We show that D grows as k^2, and must be $\Omega(N)$ to distinguish among N items.

As the maximal difference for the Oracle is $O_x |\psi\rangle = -|\psi\rangle$, $D_k \leq 4k^2$. Using $D_{k+1} = \sum_x \|O_x \psi_k^x - \psi_k\|$, we can show that $D_{k+1} \leq D_x + 4\sqrt{D_k} + 4$, and thus

$$D_{k+1} \leq 4(k+1)^2.$$

To bound the required size of D, we consider the situation $|\langle x|\psi_k^x\rangle|^2 \geq \frac{1}{2}$, meaning we are more likely to find the target. We then have

$$\|\psi_k^x - x\|^2 \leq 2 - \sqrt{2}$$

and thus, defining $E_k = \sum_x \|\psi_k^x - x\|^2$ and $F_k = \sum_x \|x - \psi_k^x - \|^2$, we can write

$$D_k \geq E_k + F_k - 2\sqrt{E_k F_k}$$

and using $E_k \leq (2-\sqrt{2})N, F_k \geq (2-\sqrt{2})N$, we can show that to be able to disntinguish $x, D_k \geq cN$.

Combining this with the above result gives

$$k \geq \sqrt{\frac{cN}{4}} \tag{36}$$

showing that $O(\sqrt{N})$ is optimal.