Quantum Computing (CST Part II)

Lecture 8: Quantum Search

Quantum computation is... a distinctively new way of harnessing nature... It will be the first technology that allows useful tasks to be performed in collaboration between parallel universes.

David Deutsch

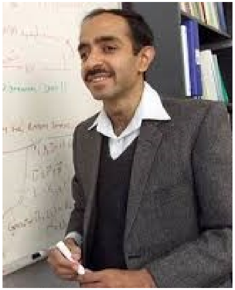
What can we do with quantum parallelism?

Last time we saw how the Deutsch-Jozsa algorithm uses the fact that when using a quantum computer an oracle can be queried by a superposition of computational basis states (rather than by a single state), thus using this *quantum parallelism* to reduce the complexity of some query problems.

This prompts the question of what else we can use this quantum parallelism for: and the simplest thing we may think of is to use a quantum computer to search through a database looking for a single significant entry.

Quantum search

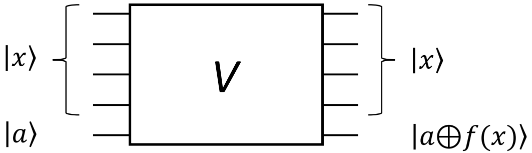
- In 1996 Indian-American computer scientist *Lov Grover* published a quantum search algorithm, which to this day remains one of the most important quantum algorithms.
- The algorithm concerns searching an *unstructured database* with N entries.
- If we are searching for a unique *marked* entry, then classically this would take a maximum of N queries, and $N/2$ queries on average.
- Grover's algorithm enables the task to be completed with only $\mathcal{O}(\sqrt{N})$ queries.


https://www.dotquantum.io

Lov Grover

Oracles revisited

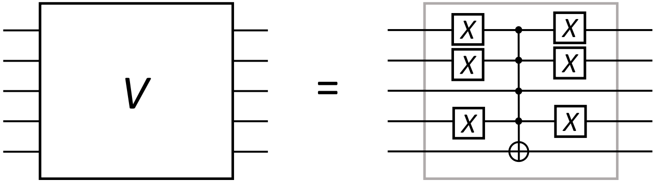
Recall that a core component of the Deutsch-Jozsa algorithm is the "oracle", which can be thought of as something that can *recognise* a correct answer when it sees one. Specifically, we think of an oracle as a black box "marking" some input binary strings as 0 (i.e., $f(x) = 0$) and some as 1 (i.e., $f(x) = 1$), of the form:



- In Grover's algorithm, we consider a *search oracle*, V, which marks a single element as 1, and all the others as 0.
- The goal is to find the element marked 1, and we assume that there is no algorithmic short-cut to find it, so classically we would have to perform a brute-force search.

An example of an oracle

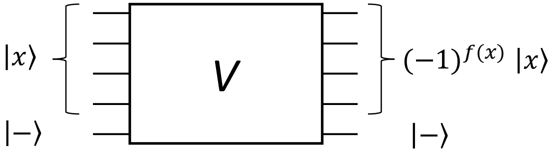
Say that 0010 is the unique "marked" binary string, then an appropriate oracle would be:



Which is in the form of a general control gate.

The action of a search oracle on $|-\rangle$

- Consider an input binary string x and let the final qubit input be set as $|-\rangle$, (i.e., the input is $|x\rangle\,|-\rangle$)
- The oracle transforms this to $(-1)^{f(x)}\,|x\rangle\,|-\rangle$,
  That is, if $f(x) = 0$ then nothing happens, whereas if $f(x) = 1$ then the last qubit is changed from $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ to $\frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) = -\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ because of the modulo-2 addition.



Decomposing the uniform superposition

Initially we apply $H^{\otimes n}$ to the search register, that is we put the input in the uniform superposition over all bitstrings of length $n = \log_2 N$ (for convenience we let the number of elements in the search space, N, be a power of 2) to get:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle$$

Which we can write:

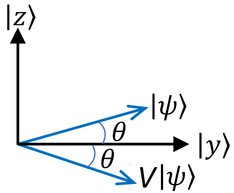
$$|\psi\rangle = \frac{1}{\sqrt{N}} \left(\sqrt{N-1}\, \underbrace{\frac{1}{\sqrt{N-1}} \sum_{x \text{ s.t. } f(x)=0} |x\rangle}_{|y\rangle} + |z\rangle \right)$$

where $|z\rangle$ is the single basis state $|x\rangle$ such that $f(x) = 1$.

Visualising the action of a search oracle on $|-\rangle$

Now we consider the effect of the search oracle on this input, as can be visualised in a two-dimensional space where (from previous slide):

- $|y\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \text{ s.t. } f(x)=0} |x\rangle$, i.e., a unit vector in the direction of the uniform superposition of all unmarked strings.
- $|z\rangle = |x\rangle$ where $f(x) = 1$, i.e., the marked element.

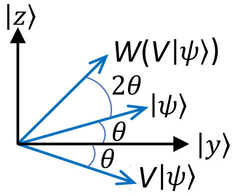


Where $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$ is the uniform superposition.

Applying the search oracle to $|\psi\rangle$ does nothing to the component in the $|y\rangle$ direction, but multiplies the component in the $|z\rangle$ direction by -1. So the action of the search oracle can be seen as a reflection in the $|y\rangle$ axis.

The second step: reflecting about the original superposition

- As we have seen, the search oracle leads to a reflection in the $|y\rangle$ axis.
- But we need to rotate the superposition towards the $|z\rangle$ axis to increase the probability of measuring the marked state (i.e., x s.t. $f(x) = 1$) as is required.
- So we follow up the oracle step with another reflection, about the line of the original uniform superposition, which is given by $W = (2\,|\psi\rangle\,\langle\psi| - I)$ where $|\psi\rangle = |+\rangle^{\otimes n}$.



The operation of W

To see that the unitary W does indeed perform the reflection as claimed, consider an arbitrary (real) vector $|\phi\rangle$, decomposed into a component in the direction of the uniform superposition (denoted $|\psi\rangle$ as previously defined), and a component perpendicular to the uniform superposition, which we denote $|\psi^{\perp}\rangle$:

$$|\phi\rangle = a\,|\psi\rangle + b\,|\psi^{\perp}\rangle$$

So it follows:

$$\begin{aligned}
W\,|\phi\rangle =& (2\,|\psi\rangle\langle\psi| - I)\,|\phi\rangle \\
=& (2\,|\psi\rangle\langle\psi| - I)(a\,|\psi\rangle + b\,|\psi^{\perp}\rangle) \\
=& 2a\,|\psi\rangle \underbrace{\langle\psi|\psi\rangle}_{=1} - 2b\,|\psi\rangle \underbrace{\langle\psi|\psi^{\perp}\rangle}_{=0} - a\,|\psi\rangle - b\,|\psi^{\perp}\rangle \\
=& 2a\,|\psi\rangle - a\,|\psi\rangle - b\,|\psi^{\perp}\rangle \\
=& a\,|\psi\rangle - b\,|\psi^{\perp}\rangle
\end{aligned}$$

That is, the desired reflection about the uniform superposition.

Implementing W with gates from a universal set

We still have to show that W can be implemented with gates from our universal set. We start by decomposing $W = 2\left|\psi\right\rangle\left\langle\psi\right| - I$ (where $\left|\psi\right\rangle = \left|+\right\rangle^{\otimes n}$):

$$\begin{aligned} H^{\otimes n} W H^{\otimes n} =& 2\left|0^n\right\rangle\left\langle 0^n\right| - H^{\otimes n} I H^{\otimes n} \\ =& 2\left|0^n\right\rangle\left\langle 0^n\right| - I \end{aligned}$$

because H is self-inverse, and $H\left|+\right\rangle = \left|0\right\rangle$ (note $\left|0^n\right\rangle = \left|0\right\rangle^{\otimes n}$). We can also use the fact that $X\left|0\right\rangle = \left|1\right\rangle$:

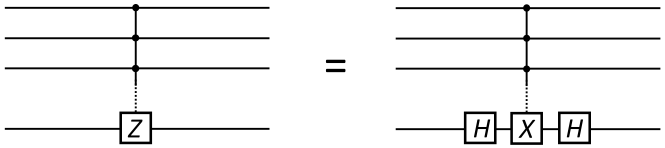
$$\begin{aligned} X^{\otimes n}(H^{\otimes n} W H^{\otimes n}) X^{\otimes n} =& X^{\otimes n}(2\left|0^n\right\rangle\left\langle 0^n\right| - I) X^{\otimes n} \\ =& 2\left|1^n\right\rangle\left\langle 1^n\right| - I \end{aligned}$$

which is the matrix:

$$2\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \end{bmatrix} = -\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & -1 \end{bmatrix}$$

Implementing W with gates from a universal set (cont.)

We have that $(-2\,|1^n\rangle\langle 1^n| + I)$ is a n-qubit generalisation of the CZ gate, which we denote $C_{n-1}Z$:



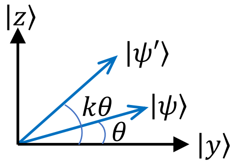
As $Z = HXH$, we have that $C_{n-1}Z = (I \otimes H)C_{n-1}X(I \otimes H)$, which can be efficiently implemented using Toffoli gates with some extra workspace qubits.

Putting this altogether we have:

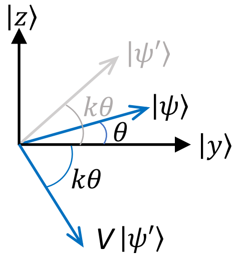
$$W = -H^{\otimes n}X^{\otimes n}(I \otimes H)C_{n-1}X(I \otimes H)X^{\otimes n}H^{\otimes n}$$

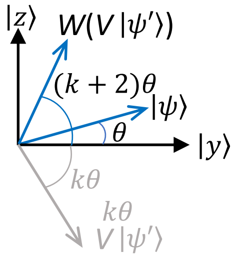
Iterating these two reflections

Say we have rotated to an angle $k\theta$:



Then the action of the matrices V and then W will be:

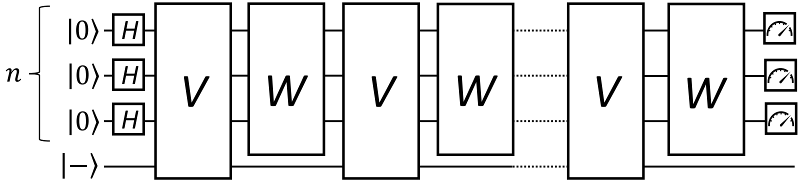




Each occurrence of the Grover iterate, $(W \otimes I)V$, leads to a 2θ rotation.

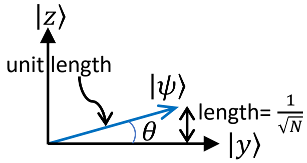
Grover's algorithm

It follows that Grover's algorithm consists of an iteration over V and W, which rotates the uniform superposition $|\psi\rangle$ towards a superposition consisting of only (or dominated by) the marked element, $|z\rangle$:



How many iterations?

By simple trigonometry we can express θ:

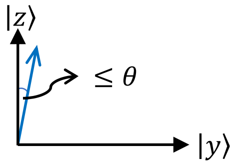


So $\sin\theta = \left(\frac{1}{\sqrt{N}}\right)/1$. If we are searching a large database then $\theta \approx \sin\theta$. Each iteration rotates the superposition 2θ, and we need to rotate $(\frac{\pi}{2} - \theta)$ in total, thus we require n_{it} iterations, where:

$$\begin{aligned} \frac{\pi}{2} - \theta &= n_{it} \times 2\theta \\ \implies n_{it} &= \frac{\pi}{4\theta} - \frac{1}{2} \\ &\approx \frac{\pi\sqrt{N}}{4} \end{aligned}$$

so we require only $\mathcal{O}(\sqrt{N})$ iterations of WV in Grover's algorithm, as opposed to checking all N elements in a classical brute-force search.

Other things to note



- In general N may be such that the superposition never quite aligns with $|z\rangle$, however we can always rotate to within θ of $|z\rangle$. Therefore we will measure the marked answer with probability at least:

$$(\cos\theta)^2 = 1 - (\sin\theta)^2 = 1 - \left(\frac{1}{\sqrt{N}}\right)^2 = \frac{N-1}{N}$$

- Rotating too far reduces the probability of getting the right answer.
- Grover's algorithm also works when there is more than one marked answer (where the aim is to find any marked answer).
- It has been shown that $\Theta\sqrt{N}$ is a lower bound for search of an unstructured database.

Oracles and complexity

Recall that an oracle can be thought of as something that can recognise an answer.

- In general an oracle can be any function that outputs a 0 or 1.
- If we now think about NP-complete problems, by definition the solution can be *verified* by a function which can be computed in polynomial time... thus we can encode such a function as an oracle.

So it follows that, if we want to solve a NP-complete problem by a brute force search through all possible solutions, then we can use the polynomial verifying function as an oracle, and thus Grover's algorithm can yield a quadratic speed-up (note there is a subtlety here regarding the fact that in general a NP-complete problem may have an unknown number of solutions, so we may not know how many iterations of Grover's algorithm to perform to get the correct rotation, but this can be taken care of using other techniques).

If we know that there are M solutions (M marked elements), then the same analysis can be applied to the case in which $|z\rangle$ on Slide 8 is not a single marked element, but the equal superposition of *all* marked elements, and yields $\frac{\pi}{4}\sqrt{N/M}$ iterations needed to find *some* solution.

There is more to quantum computing than superposition

So we can see that we cannot use quantum computing to obtain an exponential speed-up in unstructured search problems. This is one manifestation of the fact that quantum computing is more nuanced that simply preparing a superposition state and thus evaluating each possibility simultaneously.

- In particular, we can only get an exponential quantum speed-up for problems that have a specific structure where we must interfere the superposed terms to get the answer.
- This is the function of the final layer of Hadamard gates in Deutsch-Jozsa, and we will see that the same feature is present in many other quantum algorithms.

But the quadratic advantage is not to be sniffed at

Even though Grover search "only" yields a quadratic speed-up, this may still be very useful in practise. We can understand why we can get a quadratic advantage in the unstructured search problem in the following way.

- Each element in the database has a $1/N$ probability of being the marked element. It follows that if we classically search through the elements, each new element that we inspect increases the probability of us having found the marked element by $1/N$. Thus taking $1/(1/N) = N$ iterations to find the marked state with certainty.
- Quantumly, we put all elements in superposition and then "move the superposition around". Crucially, the (modulus of the) co-efficient of a term in the superposition is the square root of its probability, and so the co-efficient of the marked element is $\sqrt{1/N}$. As we move the superposition such that the co-efficient of the marked element increases in constant increments we only take $\mathcal{O}(1/(\sqrt{1/N}) = \mathcal{O}(\sqrt{N})$ iterations to find the marked state.

Summary

- Grover's search algorithm enables a "marked" element of an unstructured database to be found with high probability with $\mathcal{O}(\sqrt{N})$ queries to the database, as opposed to $\mathcal{O}(N)$ classically.
- Grover's algorithm is of the form of repeated oracle calls V and reflections W.
- The oracle V simply recognises the marked entry.
- The reflection W can be efficiently implemented using gates from a standard universal gate-set.