G53NSC and G54NSC Non-Standard Computation

Dr. Alexander S. Green

9th of March 2010

Introduction

- Last week we looked at Grover's algorithm
- We shall start today by looking at an example of Grover's algorithm
- for a search space of size 8 ($N = 3$)
- Then we'll be moving on to Shor's algorithm...
- Looking at a similar algorithm known as Simon's algorithm
- How it relates to Shor's algorithm and period finding
- and how period finding relates to Factorisation
- Next week, we'll look at how the Quantum Fourier transform is used in Shor's algorithm
- and work through an example factorisation using Shor's algorithm

Research Presentations

- Research presentations start in two weeks...
- So it is time to set a schedule for the talks
- There are 7 projects:
  - 4 will take place on the 23rd of March
  - 3 will take place on the 30th of March
- To randomly choose the order of talks, I shall use *QIO*...
- The webpage will be updated accordingly

Part I

Grover's Algorithm

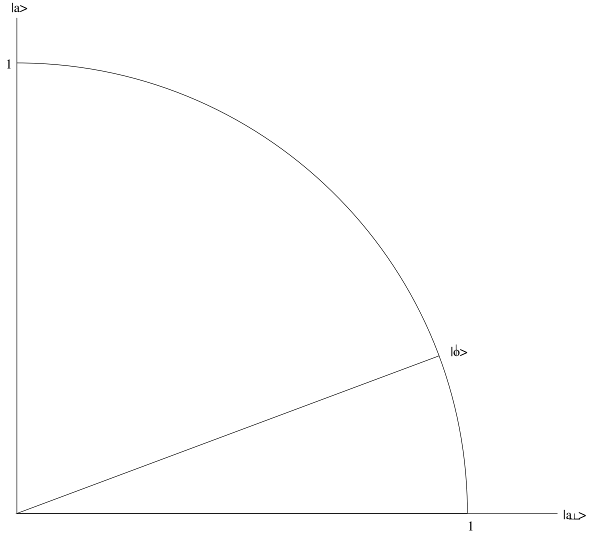
Grover's Algorithm

- An example application of Grover's algorithm for $N = 3$
- We're given a function f of type $(Bool, Bool, Bool) \rightarrow Bool$ which returns *True* only for one input
- We can define a unitary U_f such that $U_f \left|x\right\rangle \otimes \left|y\right\rangle = \left|x\right\rangle \otimes \left|y \oplus f(x)\right\rangle$
- and setting the fourth qubit as an ancilliary qubit, in the state $\left|-\right\rangle$, gives us the unitary V we require
- $V\left|x\right\rangle = (-1)^{f(x)}\left|x\right\rangle = \begin{cases} \left|x\right\rangle, & x \neq a \\ -\left|a\right\rangle, & x = a \end{cases}$
- We now need to define the necessary W unitary too...
- Remember, we could use $-W$
- $-W = H^{\otimes 3} W' H^{\otimes 3}$ where $W'\left|x\right\rangle = (-1)^{x \equiv 0}\left|x\right\rangle = \begin{cases} \left|x\right\rangle, & x \neq 0 \\ -\left|0\right\rangle, & x = 0 \end{cases}$

Grover's Algorithm

- We can define the state $|a\rangle$ as the state we are looking for
- and the state $|a_\perp\rangle$ as the states orthogonal to $|a\rangle$
- We can create the following state using Hadamard rotations:
  $|\phi\rangle = \frac{1}{\sqrt{8}}(|0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + |5\rangle + |6\rangle + |7\rangle)$
- Which can be alternately written as:
  $\sqrt{\frac{7}{8}}\,|a_\perp\rangle + \frac{1}{\sqrt{8}}\,|a\rangle$
- We can visualise this on the plane spanned by $|a\rangle$ and $|a_\perp\rangle$

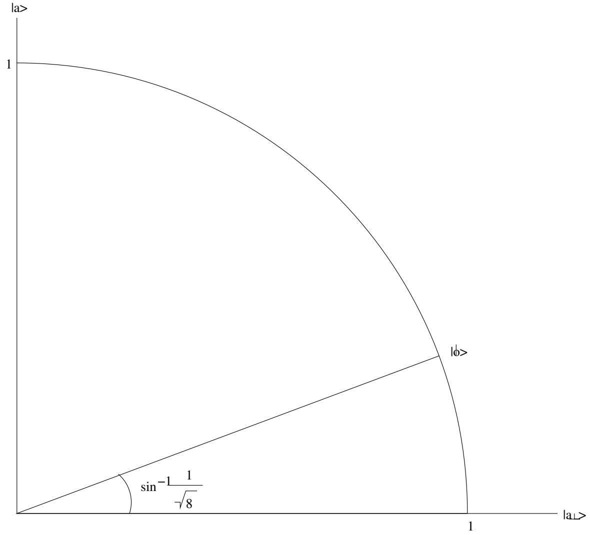
Grover's Algorithm



Grover's Algorithm

- The angle (θ) between $|a_{\perp}\rangle$ and $|\phi\rangle$ can now be calculated...
- Remembering, for a right angled triangle that $sin\theta = \frac{Opposite}{Hypotenuse}$
- $|\phi\rangle$ is a unit vector, so the hypotenuse is 1
- The opposite is the amplitude of $|a\rangle$ in $|\phi\rangle$, which we have seen is $\frac{1}{\sqrt{8}}$
- So, $sin\theta = \frac{1}{\sqrt{8}}$ and $\theta = sin^{-1}\frac{1}{\sqrt{8}}$

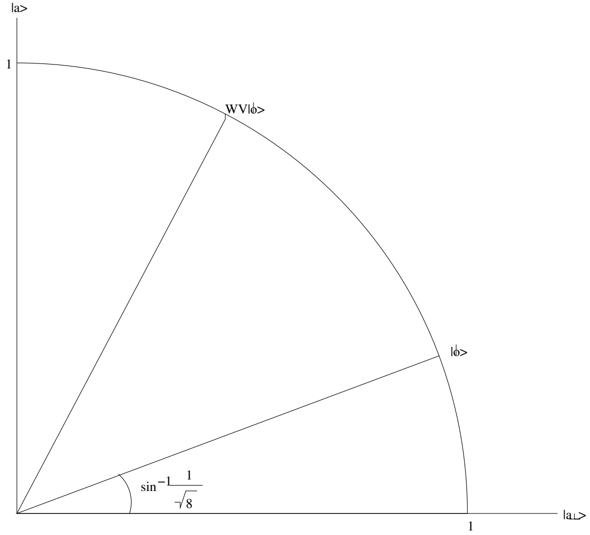
Grover's Algorithm



Grover's Algorithm

- Remember, V reflects about the $|a_\perp\rangle$ axis
- and W reflects about $|\phi\rangle$
- Combined, they form what is known as a Grover iteration, which rotates the state by 2θ
- How many iterations do we require here?
- For large N, $|\phi\rangle$ is so close to $|a_\perp\rangle$ that we can say the number of iterations required is $\frac{\pi}{4}\sqrt{N}$
- However, for small N the original angle θ is enough to make a difference
- We can calculate the number of iterations n by noting that $\theta + 2n\theta$ needs to be as close to $\frac{\pi}{2}$ as possible
- So, we need n to be the nearest integer to $\frac{\frac{\pi}{2}-\theta}{2\theta}$
- $\frac{\frac{\pi}{2}-\theta}{2\theta} \approx 1.673$, so we have $n = 2$ iterations

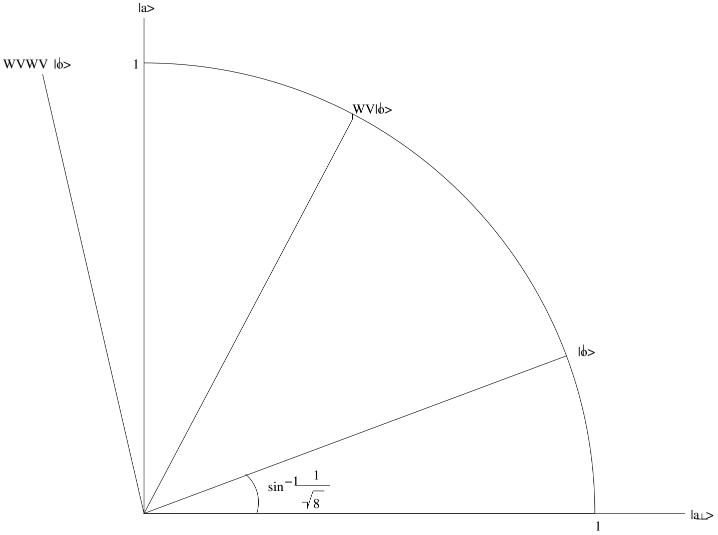
Grover's Algorithm



Grover's Algorithm

- After only a single iteration, we can see the state is getting close to $|a\rangle$
- The angle is 3θ, so we can calculate the measurement probabilities...
- the amplitude of $|a\rangle$ can be calculated using $sin3\theta = \frac{Opposite}{Hypotenuse}$
- Again, the hypotenuse is 1, so the opposite is $sin3\theta$
- ≈ 0.88393
- So, the probability of measuring $|a\rangle$ after a single iteration is $|sin3\theta|^2 \approx 0.781$

Grover's Algorithm



Grover's Algorithm

- After two iterations, we can see the state is closer to $|a\rangle$, and it's clear that another iteration would take us further from $|a\rangle$ again
- The angle is 5θ which is more than $\frac{\pi}{2}$, but we can still calculate the measurement probabilities
- the amplitude of $|a\rangle$ can be calculated using $sin(\pi - 5\theta) = \frac{Opposite}{Hypotenuse}$
- Again, the hypotenuse is 1, so the opposite is $sin(\pi - 5\theta)$
- ≈ 0.972
- So, the probability of measuring $|a\rangle$ after a single iteration is $|sin(\pi - 5\theta)|^2 \approx 0.945$
- So, after 2 iterations, we have a probability of ≈ 0.945 of measuring $|a\rangle$, no matter which of the base states it may be.

Moving on

- The last exercise sheet will involve implementing Grover's algorithm for $N = 3$...
- You should compare your results with the results predicted by us today
- For the rest of today, and next week's lecture, we shall be looking at Shor's algorithm
- It is useful to first look at Simon's algorithm...

Part II

Simon's Algorithm

Simon's Algorithm

- Simon's algorithm is said to be one of the main inspirations behind Shor's technique
- Although it is a simpler algorithm, it is very closely related to Shor's algorithm
- It was one of the first algorithms to show an exponential speed-up over the fastest known classical solution
- It was first described by Daniel R. Simon in 1994
- You are given a function $f :: Bool^n \to Bool^{n-1}$, that is defined such that it is periodic under bitwise modulo-2 addition
- That is, if $f(x) = f(y)$, then $x = y$ or for some a, $x = y \oplus a$
- In other words, there exists an a such that $f(x \oplus a) = f(x)$
- Simon's problem involves finding the value of a

Simon's Algorithm

- Classically, the best algorithm for finding a is exponential in the size of n
- Can we use a quantum computer to find a more efficient solution?
- What happens if we define a unitary U_f that implements the following
- $U_f \left|x\right\rangle \otimes \left|y\right\rangle = \left|x\right\rangle \otimes \left|y \oplus f(x)\right\rangle$
- and apply it to the state $\left|\phi\right\rangle \otimes \left|0\right\rangle$?
- We'll be left with the state $\frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} \left|x\right\rangle \left|f(x)\right\rangle$
- Whereby each value of x is entangled with its corresponding $f(x)$
- What is of note now, is what happens when we measure the second register (the $\left|f(x)\right\rangle$)

Simon's Algorithm

- The two-to-one nature of f means that the second register contains each possible value twice...
  - once for the application $f(x)$
  - once for the application $f(x \oplus a)$
- Measuring the second register, will leave the first register in an equal superposition of the two states corresponding to some x_0
- E.g. $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus a\rangle)$
- So, this looks promising, as we're trying to learn a
- How can we extract a from this superposition?

Simon's Algorithm

- Unfortunately its not that straight forward...
- Measuring would just give us a single state, with no hint towards a
- We cannot clone an arbitrary quantum state
- Repeating the experiment will (with high probability) leave us with a different state (E.g. $\frac{1}{\sqrt{2}}(|x_1\rangle + |x_1 \oplus a\rangle)$)
- All we have is a superposition whose states are related by the number a we are trying to calculate
- Fortunately, Simon showed how we are able to learn some partial information about a from the given state...
- All we need to do is apply Hadamard rotations to each qubit in the first register before measuring.
- This doesn't give us a, but gives us (with high probability) enough information to determine a single bit of a.

Simon's Algorithm

- Lets look at this in a little more detail
- We have the state $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus a\rangle)$
- This is an equal superposition of two base states
- Applying a Hadamard rotation to each of the qubits leaves the state $\frac{1}{2^{\frac{n+1}{2}}} \sum_{y=0}^{2^n-1} ((-1)^{x_0 \cdot y} + (-1)^{(x_0 \oplus a) \cdot y}) |y\rangle$
- Where $\cdot$ is the bitwise modulo-2 dot product
- We can note that $(-1)^{(x_0 \oplus a) \cdot y} = (-1)^{x_0 \cdot y}(-1)^{a \cdot y}$
- So, in the cases where $a \cdot y = 1$ the coefficients cancel each other out...
- leaving $\frac{1}{2^{\frac{n-1}{2}}} \sum_{a \cdot y=0} (-1)^{x_0 \cdot y} |y\rangle$

Simon's Algorithm

- What can we get from measuring this state?
- We get a base state $|y\rangle$ with the property that $a \cdot y = 0$
- The state $|y\rangle$ is able to be used to calculate a single bit of a
- Simon went on to show, that you only need to repeat this around $n + 20$ times to have learned all the bits of a (with the proability of failure being less than 1 in a million)
- For more information on Simon's algorithm, please see the course text book
- So, Simon defined an algorithm that finds the period (modulo-2) of a given function, exponentially faster than the best classical solution.
- What has this got to do with Shor's algorithm?

Part III

Shor's Algorithm

Shor's Algorithm

- Shor's algorithm is usually described as a factorisation algorithm
- In fact, it is a period finding algorithm
- So, why is it described as a factorisation algorithm?
- Well, with a bit of number theory, we can reformulate factorisation into finding the period of a specific function...
- We can restrict ourselves to the specific case of factorisation where we want to factor $N = pq$ with p and q both large primes
- First, it is useful to look at periodic functions of the type b^x in modular arithmetic
- Remembering that $b(modN)$ is the remainder of $\frac{b}{N}$

periodic modular aithmetic

- How about some examples...
- $5^x (mod 7)$

$$5 = 5(mod7) \qquad 5^2 = 4(mod7) \qquad 5^3 = 6(mod7)$$

$$5^4 = 2(mod7) \qquad 5^5 = 3(mod7) \qquad 5^6 = 1(mod7)$$

- So, it is periodic with period 6
- $4^x (mod 7)$

$$4 = 4(mod7) \qquad 4^2 = 2(mod7) \qquad 4^3 = 1(mod7)$$

- So, it is periodic with period 3

periodic modular arithmetic

- In fact, we can state the following theorem...

Theorem

If b shares no factors with N then $b^r = 1(modN)$ for some integer r

- and $b^x(modN)$ is a periodic function of x with period r
- A proof of this theorem follows from Lagrange's theorem, but i won't go into details here
- Now, we can show that if r has two specific properties, then it can be used to calculate p and q as required
- First, where does b come from?
- There are many values that b can take, and the values can be calculated efficiently on a classical computer...
- For b and N to share no factors, we can calculate their GCD, and check that it is 1

periodic modular arithmetic

- GCD can be calculated efficiently using the Euclidean algorithm
- So we can calculate a random value b
- If the period we calculate doesn't have the necessary properties, then we can try again with a different value for b
- Shor showed that the probability of choosing a random b that leads to a period r with the necessary properties is at least 0.5
- So, what are these properties?
- The first one is that r needs to be an even number
- If r is even, then we can calculate a value $x = b^{\frac{r}{2}} (modN)$
- and note that $(x-1)(x+1) = x^2 - 1 = 0 (modN)$

periodic modular arithmetic

- Another thing to note, is we also know that $x - 1 \neq 0(modN)$
- This follows from r being the smallest integer value for which $b^r = 1(modN)$
- Our second requirement is that $x + 1 \neq 0(modN)$
- If both these properties hold, then we know that neither $x - 1$ nor $x + 1$ are divisible by N
- However, we do know that $(x - 1)(x + 1)$ is divisible by N
- As $N = pq$ is the product of two primes, we know that one of $x - 1$ and $x + 1$ is divisible by p, and one is divisble by q
- So, one of p and q is the GCD of N and $(x - 1)$
- the other is the GCD of N and $(x + 1)$

Shor's Algorithm

- So, if we can find the period r of a function $f(x) = b^x (modN)$, we can factorise N.
- To get started, all we need to do is define a unitary operator U_f such that $U_f \left|x\right\rangle \otimes \left|y\right\rangle = \left|x\right\rangle \otimes \left|y \oplus f(x)\right\rangle$
- then applying the unitary U_f to an equal superposition of states $\left|\phi\right\rangle \otimes \left|0\right\rangle$ will leave us with the state $\frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} \left|x\right\rangle \otimes \left|f(x)\right\rangle$
- Measuring the second register will give us, with equal probability, a single base state
- Leaving the first register in the state $\frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} \left|x_0 + kr\right\rangle$
- Where m is the smallest integer such that $mr + x_0 \geqslant 2^n$
- We're left with a similar situation as in Simon's algorithm
- The states in the superposition are related by the period r which we are trying to calculate, but how can we extract this information?

Shor's Algorithm

- Unfortunately, we can't just measure it...
- and Hadamard rotations aren't enough, like they were in Simon's algorithm
- What Shor discovered was that applying the Quantum Fourier transform to this state will allow us to extract the period r.
- We will be looking at the Quantum Fourier transform next week
- along with how we can construct the necessary unitary for the function $f(x) = b^x (mod N)$

Thank you

- Remember, labs are on Thursday...
- This week's exercise sheet will be the last
- Although labs will still run upto the deadline
- I hope to see you there
- Thank you.