Shor's Algorithm

Elliott Pryor, Benjamin Bushnell, REID HELP

13 November, 2019

1 The Problem

The problem of factoring large numbers has existed for centuries. Euclid's algorithm provides a very efficient way to determine the greatest common divisor of two numbers (GCD). Say we look for $GCD(a, b)$ then a is a factor of b if the $GCD(a, b) > 1$. If we have one of the factors, it is very easy to find the other factor as we can just divide the number by its factor. However, factoring very large numbers using Euclid's algorithm is very time consuming as we have to try every single number less than the value we are factoring. We can make a few improvements to our guesses; however, classically this problem cannot be solved in polynomial time.

This is very important to modern security. We use RSA encryption as a way to encrypt our data.

For sections 2 and 3 assume we are trying to find:

$$ab = C$$

where a and b are factors of n and $a, b, C \in \mathbf{N}$.

2 Classical Computation

The classical part of this algorithm is really simple. We just have to guess a random number as our factor. Then we use Eucild's algorithm to verify that we didn't get extremely lucky and guess a factor. If we guess a factor, then we don't need to use the quantum part of the algorithm. Then we feed our guess (which we know does NOT share factors with the number in question) into the quantum algorithm.

3 The Quantum Algorithm

The quantum part of this algorithm is what turns the random number that we guessed into the actual factor. In short, it does this by finding the period of some function that we can relate to the factors. Given that that is a gross oversimplification of what happens, we will cover some of the math supporting Shor's Algorithm.

First, we must define a few relations.

Theorem 1. *Given $a, b \in \mathbf{Z}$ and a, b share no common factors. Then*

$$a^p = mb + 1$$

for some $m, p \in \mathbf{Z}$.

Theorem 2. *Given $a^x = mN + r$ for $a, x, m, N, r \in \mathbf{Z}$. Then*

$$a^{x+yp} = kN + r$$

for some $y, p, k \in \mathbf{Z}$.

We know that the guess given to the quantum portion of this algorithm, g, does not share any factors with the number we are trying to factor, C. Then by theorem 1 we can express this as $r^p = mC + 1$. We can rearrange this to $(r^{p/2} + 1)(r^{p/2} - 1) = mC$. Then we know that our factors of C are related to $(r^{p/2} + 1)$ and $(r^{p/2} - 1)$. Once we find these numbers, we can use Euler's formula to calculate the factors.

In order to find p we use theorem 2 and some properties of quantum computers. By theorem 2, we know that $g^{x+yp} = kC + r \text{ mod(C)} = r\ \forall y \in \mathbf{Z}$. So we can raise our guess to integer powers and search for when the remainder repeats. Classically, we cannot do this efficiently. However, with quantum computers we have a superposition of states that we can exploit to efficiently compute this. If we poll a random remainder value from our modulus calculation we get a superposition of the states that result in this value: g^x, g^{x+p}, g^{x+2p}... This has a period of p which we can by taking the Fourier transform of this superposition. The Fourier transform returns the period of the function. Assuming that p is even, then we are done! We have now found the factors of C. If p is odd, then $r^{p/2}$ is not an integer, and we have to restart the process with a new guess.