Quantum Computing

Presenter: Qishen

November 4, 2020

1 Background

1.1 Quantum Computing

Quantum computing is entirely different from regular computing. It has the additional state, the uncertain state. This is demonstrated in the notation $x|0> + y|1>$ to indicate that there is probability x of the bit being 0 and probability y of it being 1. However, you can stop flipping the qubits to trap them into 1 state. Bell states are the combination of 2 qubits, and are generated with the CNOT gate. There are 4 possible bell states: $1/\sqrt{2}(|00> \pm |11>)$ and $1/\sqrt{2}(|01> \pm |10>)$

1.2 Shor's Algorithm

Schor's Algorithm is used to find prime factors in an efficient amount of time: $O(n^2(log(n))(log(log(n))))$. The consequences of such an algorithm include breaking RSA encryption since it is based on multiplying primes.

1.2.1 The algorithm:

1. Pick an a in $[2, N-1]$
2. Compute $gcd(a, N)$
3. for each i $[1, N]$ compute $a^i modN$
   - if result == 1 break
   - else set $r = r + 1$
4. Compute $x = a^{(r/2)}$
   - if $x + 1 modN == 0$ then set result $(p, q) = (gcd(x + 1, N), gcd(x - 1, N))$

Shor is not a pure quantum algorithm. It uses a combination of both classical and Quantum computing.

2 Questions

1. The paper mentioned the algorithm needed to be run 8 times to get high certainty, but this was just to factor 15. What would happen if it was a much larger number? Would you need to run more times to get same accuracy? There are no good answers to this questions because Quantum computers that can do such a computation have not been constructed yet.

2. How should an ideal a be determined such that you get a nontrivial result? Currently, it seems like you just have to keep trying different number, but how long would this take for large primes? Is it realistic?

3. Is the confidence level of Shor's algorithm acceptable? It's higher than other implementations. Especially if run multiple times. This may have high overhead, but is better than classical algorithms which simply do not terminate for large values of N.

4. What makes Shor's algorithm so usable in comparison to other factorization algorithms? Shor is the most efficient factoring algorihtm out there so far. However, none yet are efficient enough to be usable with large primes.