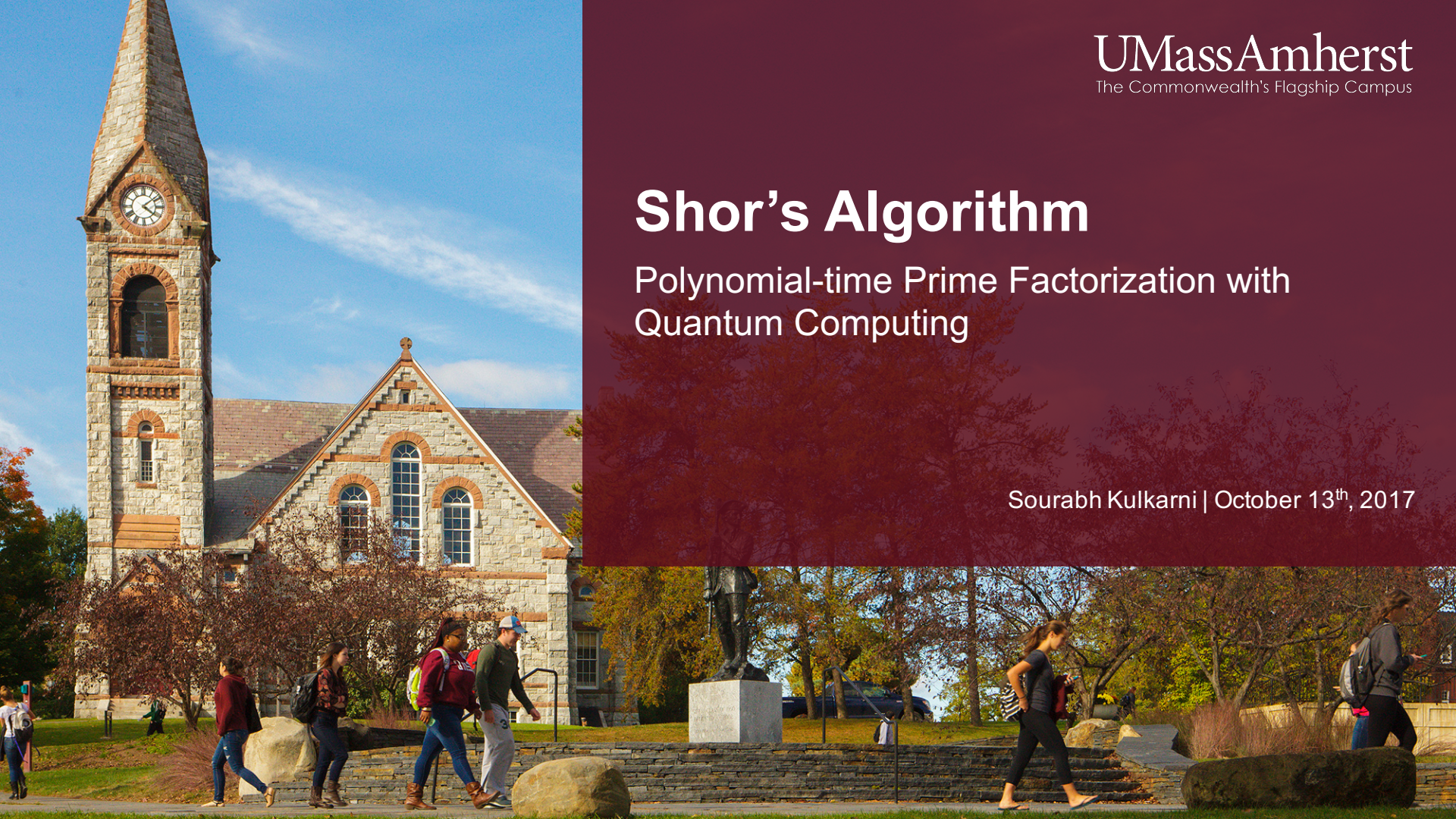


UMassAmherst
The Commonwealth's Flagship Campus

Shor's Algorithm

Polynomial-time Prime Factorization with Quantum Computing

Sourabh Kulkarni | October 13th, 2017

Content

- Church Thesis
- Prime Numbers and Cryptography
- Overview of Shor's Algorithm
- Implementation with Quantum Computing

Church Thesis

- "*Any physical computing device can be simulated by a Turing machine in a number of steps polynomial in the resources used by the computing device.*"
- Means that any physical computer does essentially what a Turing machine does
- This statement initially thought to encompass the computation power of all computers

Church Thesis and Quantum Computers

- While natural computing power of classical mechanics corresponds to that of Turing machines, the natural computing power of quantum mechanics might be greater
- Feynman - quantum mechanics computationally more powerful than Turing machines; cannot be simulated by Turing machine in polynomial time
- Example - Shor's Algorithm takes polynomial time in QC vs. super-polynomial time in classical computers

Prime Numbers and Cryptography

- Cryptography requires one-way functions
  - Easy to encode; hard to decode(without key)
- Products of large primes satisfy this requirement
  - Finding two large primes and multiplying them – easy
  - Finding prime factors of a very large number – hard
- How is this used in cryptography?
  - Product of primes (**N**) → public key; prime factors (**p**, **q**) → private key
  - Public key to encode information; private keys to decode
  - Used in RSA

Finding Prime Factors

- Finding a prime factor of **N** = breaking the RSA; But how?
- Naïve approach
  - Trail & error (check all primes less than $\sqrt{N}$)
  - Highly Inefficient; *exp (log **N**)*
- Advanced approach – Modulo arithmetic

Modulo Arithmetic

- Basics: $a \equiv x \pmod{N}$ means x is remainder of a/N
  - e.g., $10 \equiv x \bmod 4$, so x=?
- Modulo arithmetic and exponentiation
  - Periodicity - $f_{\mathbf{a},\mathbf{N}}(x) = \mathbf{a}^x \pmod{\mathbf{N}}$ is periodic if **a**, **N** are relatively prime
  - e.g. 3 and 10, 3^k Mod 10 for k = 1,2,… has a repeating pattern

Modulo Arithmetic

3^k	$3^k \pmod{10}$
3	3
9	9
27	7
81	**1**
243	3
729	9
2187	7
6561	**1**

Period **r** = 4

Modulo Arithmetic

- Modulo arithmetic and exponentiation
  - If two numbers are relatively prime, e.g. 3 and 10, $3^k \pmod{10}$ for k = 1,2,… has a repeating pattern
  - This pattern has a ‘period’, denoted as **r**
  - **r** has the property that $3^r \bmod 10 = 1$; or $3^r - 1 \bmod 10 = 0$
  - In other words, $3^r - 1$ is divisible by 10
  - This property is used in Shor’s algorithm

Prime factors with modulo arithmetic

N = **p**.**q** ; where **p**, **q** are primes we need to find

let **a** be integer smaller than **N**

Let '**r**' be the period of $\mathbf{a}^{x}$ **(**mod **N)**

Now,

$\mathbf{a}^{r}$-1 = **k**.**N** = **k.p.q ,** where **k** is some integer

$(\mathbf{a}^{r/2}-1).\ (\mathbf{a}^{r/2}+1) = \mathbf{k.p.q}$

$(\mathbf{a}^{r/2}-1)/\mathbf{p}.\ (\mathbf{a}^{r/2}+1)/\mathbf{q} = \mathbf{k}$

p = gcd($\mathbf{a}^{r/2}$**-1, N**); **q** = gcd($\mathbf{a}^{r/2}$**+1, N**)

Shor's Algorithm

- Problem: Given an odd composite number **N**, find an integer **a**, that divides **N**.
- Shor's algorithm has 2 parts:
  - Convert a factoring problem into a period finding problem (classical, using properties of modulo arithmetic)
  - Use Quantum Fourier Transform to find the period (this achieves the speedup vs. classical)

Shor's Algorithm

Has 5 steps:

- Step 1 : Choose **a** < **N** such that **a**, **N** are co-prime (gcd (**a**, **N**)=1)
- Step 2 : Use Quantum Fourier Transform to find the period **r** of the function

  $f_{a,N}(x) = a^x \pmod{N}$

- Step 3 : If,

  r = odd: go to Step 1

  r = even: go to Step 4

Shor's Algorithm

- Step 4 : $(\mathbf{a}^{r/2}-1).(\mathbf{a}^{r/2}+1) = \mathbf{k.p.q}$

  If $\mathbf{a}^{r/2}+1 = 0 \bmod \mathbf{N}$;means trivial solution (will give answers 1, **N**) go to Step 1;

  If $\mathbf{a}^{r/2}+1 \neq 0 \bmod \mathbf{N}$; go to Step 5.

- Step 5: Compute $\mathbf{p} = \gcd(\mathbf{a}^{r/2}-1, \mathbf{N})$

Shor's Algorithm – Quantum Portion

1. Choose $q=2^L$ such that $N^2 \leq q \leq 2N^2$; (ensure sufficient cycles)
2. Choose x such that $x<N$, $\gcd(x,N)=1$
3. Create two quantum registers of size q, and **entangle** them.

Shor's Algorithm – Quantum Portion

3. Set the initial state $|Reg1\rangle|Reg2\rangle$ as

   Reg1 has equal superposition of all numbers from 0 to q-1

   Reg2 has the quantum modulo exponentiation of 0 to q-1 w.r.t **N**

$$|Reg1\rangle|Reg2\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|x^a (mod\ N)\rangle$$

Shor's Algorithm – Quantum Portion

4. Apply Quantum Fourier Transform on Reg1

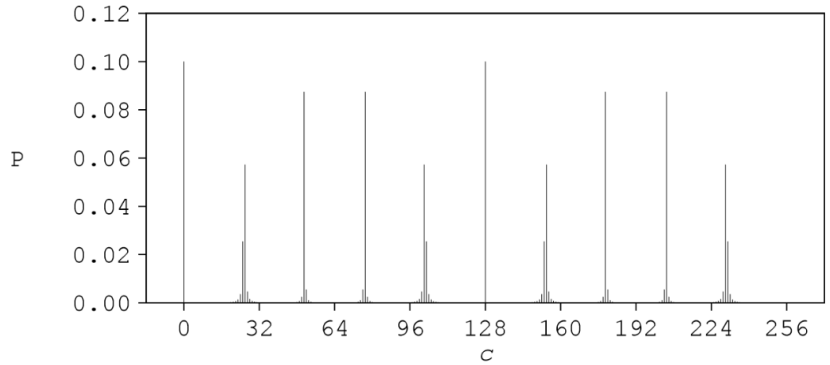
$$|Reg1'\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{\frac{(2\pi i a c)}{q}} |c\rangle$$

This makes the new entangled state as:

$$|Reg1'\rangle|Reg2\rangle = \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} e^{\frac{(2\pi i a c)}{q}} |c\rangle\,|x^a (mod\ N)\rangle$$

Shor's Algorithm – Quantum Portion

5. Observe the entangled state, which collapses it in one of **r** possible states;



The probability P of observing values of c between 0 and 255, given q = 256 and r = 10

we now calculate the probability **P** that the observed state is, say,

$|c\rangle|x^k(mod\ N)\rangle$ where 0<k<**r**

Shor's Algorithm – Quantum Portion

5. The expression for probability is as follows:

$$\left|\frac{1}{\mathrm{q}}\sum_{a:x^a\equiv x^k} e^{\frac{(2\pi iac)}{q}}\right|^2$$

This is essentially the sum over every r^{th} term k, k+r, k+2r, …

We need to extract r from this probability expression.

Shor's Algorithm – Quantum Portion

6. Solving this expression obtained in previous stage:

$$\left|\frac{1}{\mathrm{q}}\sum_{a:x^a\equiv x^k} e^{\frac{(2\pi iac)}{q}}\right|^2$$

We end up with this expression:

$$\left|\frac{c}{\mathrm{q}}-\frac{d}{r}\right|\leq\frac{1}{2q}$$

Shor's Algorithm – Quantum Portion

7. With c, q known, we solve the equation

$$\left|\frac{c}{\mathrm{q}} - \frac{d}{r}\right| \leq \frac{1}{2q}$$

for $\frac{d}{r}$ such that the denominator is less than **N**. This is done by '*continued fraction*' process (polynomial time classical algorithms exist).

Once we obtain such fraction $\frac{d}{r}$, which takes *O(log log r)* repetitions, the denominator is the period of the function, which is what we need.

Shor's Algorithm – Discussions

- Algorithm is polynomial time in high probability
  - Best classical factorisation is $exp\ (c\ (log\ n)^{1/3}(log\ log\ n)^{2/3})$
  - Shor's algorithm is $O((log\ n)^2(log\ log\ n)(log\ log\ log\ n))$
- Period is found with high probability with the algorithm repeated $O(log\ log\ r)$ times
- First number factorized with this algorithm in a QC was 15 in 2001 using 8 qubits
- New algorithm called the 'minimization' algorithm is currently used; it has factored the number 56,153 in 2012 using just 4 qubits

UMassAmherst

The Commonwealth's Flagship Campus