Quantum Computing (CST Part II)

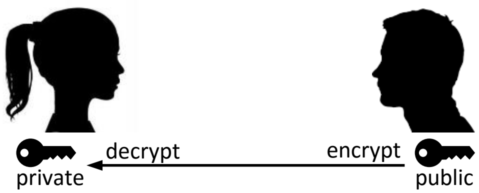
Lecture 10: Application 1 of QFT / QPE: Factoring

The problem of distinguishing prime numbers from composites, and of resolving composite numbers into their prime factors, is one of the most important and useful in all of arithmetic.
Carl Friedrich Gauss

Resources for this lecture

Nielsen and Chuang p226–235 covers the material of this lecture.

Public-key cryptography



- In public-key cryptography Alice publishes a public key, which Bob uses to encode a message. Alice then uses her private key to decrypt the message.
- This relies on the asymmetry of the cryptography: it is easy to encrypt the message using the public key, but hard to decrypt it without knowledge of the private key.
- This in turn relies on the existence of one-way functions.

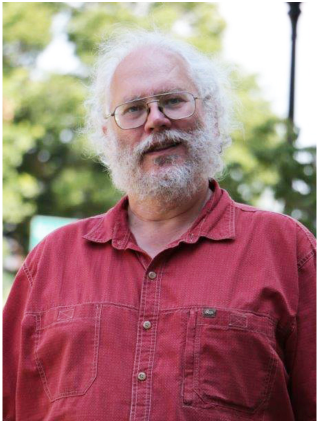
One-way functions

- One way function are functions that are easy to perform "forward", but hard to invert.
- For example, the RSA public-key cryptosystem uses factoring as a one-way function: multiplying two prime number p and q to give a composite number N is easy, but the inverse, factoring a composite number N into factors p and q is mathematically difficult.
- In fact, the best known classical algorithm for factoring, the *number field sieve*, requires $\exp(\Theta(n^{1/3}\log^{2/3} n))$ operations, where $n = \lceil \log_2 N \rceil$, i.e., the number of bits required to express N.
- For cryptographic applications it is crucial that the mathematical definition of one-way functions is that they must be hard to invert on average, not just in the worst case.

Factoring quantum mechanically

In 1994 Peter Shor invented a quantum algorithm which can factor numbers in polynomial time.

This remains one of the (or probably *the*) most important and impressive potential application of quantum computing.



www.quora.com/profile/Peter-Shor

Peter Shor

Shor's factoring algorithm

For factoring a n-bit composite integer N.

1. Is N even? If so, output 2 and stop.
2. There is an efficient classical algorithm to check whether $N = c^l$ for some integers $c, l \geq 2$ and compute c if so. Run this classical algorithm and output c if obtained and stop.
3. If N is neither even nor a prime power, randomly choose $1 < x < N$, and compute $s = \mathsf{gcd}(x; N)$ (i.e., the greatest common divisor) using Euclid's division algorithm. If $s \neq 1$ output s, which is a factor of N, and stop.
4. If $s = 1$ (i.e. x and N are co-prime), find the order r of the function $x \mod N$.
5. If r is odd, go back to step 3 and pick a different random number x. If r is even then perform efficient classical post-processing to extract a factor of N to output and stop.

Order finding

For co-prime positive integers x and N, such that $x < N$, the *order of* x *modulo* N is defined to be the least positive integer r such that $x^r = 1 \mod N$.

- Order finding is itself believed to be a hard problem classically.
- We will now show how quantum phase estimation can be used to find the order efficiently using a quantum computer.

Order finding using quantum phase estimation

To find the order of x modulo N quantumly, we simply apply QPE using the unitary:

$$U |y\rangle = |(xy) \mod N\rangle$$

where y is an integer such that $0 \leq y < N$.

U *is* unitary when x and N are co-prime because U is merely a permutation matrix (i.e., it has exactly one 1 in each column and each row). To see this, consider the following argument (in which $y_1 \neq y_2$):

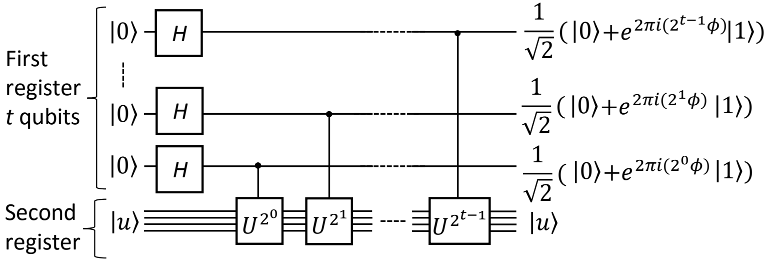
If $U |y_1\rangle = U |y_2\rangle$, then $xy_1 = xy_2 + kN$ for some integer $k \neq 0$. Therefore $y_1 - y_2 = \frac{kN}{x}$, i.e., $\frac{kN}{x}$ is an integer. However, as N and x are co-prime, the least (positive) integer k which satisfies this is $k = x$, i.e., $y_1 - y_2 = N$ (or $y_2 - y_1 = N$). So it cannot be the case that $0 \leq y_1, y_2 < N$.

So it follows, that for each integer y such that $0 \leq y < N$, $U |y\rangle$ gives a different integer between 0 and $N - 1$ inclusive, so U is a permutation matrix.

For completeness, for $N \leq y < 2^n$, we define $U |y\rangle \rightarrow y$.

Quantum phase estimation with $U\,|y\rangle = |(xy) \mod N\rangle$

Recall that the QPE circuit is thus:



First register
t qubits
Second register
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|u\rangle$
H
H
H
U^{2^0}
U^{2^1}
$U^{2^{t-1}}$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(2^{t-1}\phi)}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(2^{1}\phi)}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(2^{0}\phi)}|1\rangle)$
$|u\rangle$

To perform QPE we need to implement controlled-U^{2^j} gates and prepare the second register in an eigenstate of U. However, regarding the first of these, if we were simply to take a naive approach, we would incur an exponential amount of operations.

Implementing the series of controlled-U^{2^j} gates

There are a few variations on the method used to implement the series of controlled-U^{2^j} gates, so here we present one which can be (relatively) easily understood. We begin by noting:

$$\begin{aligned} U^{2^j}|y\rangle &= |(x^{2^j}y) \mod N\rangle \\ &= |((x^{2^j} \mod N)y) \mod N\rangle \end{aligned}$$

Therefore:

1. We first pre-compute $x^{2^j} \mod N$ for all j, $0 \leq j \leq t-1$ (this is known as *modular exponentiation*).
2. We then use these pre-computed values to implement the unitary operation $|y\rangle \rightarrow |((x^{2^j} \mod N)y) \mod N\rangle$ (controlled as appropriate) for all j, $0 \leq j \leq t-1$ in sequence as required.

Complexity of implementing the controlled-U^{2^j} gates

Noting that multiplication takes $\mathcal{O}(m^2)$ operations, where m is the number of bits needed to specify the numbers being multiplied, the computational complexity of the two steps required to implement the controlled-U^{2^j} gates is therefore as follows:

1. Pre-computing $x^{2^j} \mod N$ for all j, $0 \leq j \leq t-1$ can be achieved by repeated modular squaring of x – i.e., a total of t squaring operations. Note that $x^{2^j} \mod N < N$, so a maximum of n bits are needed to specify the numbers being squared, so the total number of operations required is $\mathcal{O}(n^2 t)$.
2. With the pre-computed values of $x^{2^j} \mod N$ we need to implement the multiplication $((x^{2^j} \mod N)y) \mod N$ for all j, $0 \leq j \leq t-1$, i.e., t times in total. $y < N$, so a total of n bits are required to express each of the numbers being multiplied, therefore the complexity of this step is also $\mathcal{O}(n^2 t)$.

To extract the order from the phase it suffices that $t \in \mathcal{O}(n)$, so putting these together we have that the number of operations required to perform the series of controlled-U^{2^j} gates is $\mathcal{O}(n^3)$.

The eigenvalues of U

QPE also requires that we prepare the input to the series of controlled-U^{2^j} gates in an eigenstate of U. States defined by:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i s k/r} |x^k \mod N\rangle$$

for $0 \leq s \leq r-1$ are eigenstates of U, because:

$$\begin{aligned}
U|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i s k/r} |x^{k+1} \mod N\rangle \\
&= \frac{1}{\sqrt{r}} \sum_{k=1}^{r} e^{-2\pi i s (k-1)/r} |x^k \mod N\rangle \\
&= e^{2\pi i s/r} \frac{1}{\sqrt{r}} \sum_{k=1}^{r} e^{-2\pi i s k/r} |x^k \mod N\rangle \\
&= e^{2\pi i s/r} \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i s k/r} |x^k \mod N\rangle \\
&= e^{2\pi i s/r} |u_s\rangle
\end{aligned}$$

A superposition of eigenstates of U

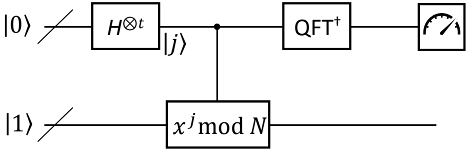
To prepare an eigenstate as defined on the previous slide requires knowledge of r, which we clearly don't have, so instead we prepare an equal superposition of eigenstates for $0 \leq s \leq r-1$:

$$\begin{aligned}
\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i s k / r} \left| x^k \mod N \right\rangle \\
&= \frac{1}{r} \left(\sum_{s=0}^{r-1} e^0 \left| x^0 \mod N \right\rangle + \sum_{k=1}^{r-1} \underbrace{\left(\sum_{s=0}^{r-1} e^{-2\pi i s k / r} \right)}_{=0 \text{ when } k>0} \left| x^k \mod N \right\rangle \right) \\
&= \frac{1}{r} r \, |1\rangle \\
&= |1\rangle
\end{aligned}$$

Which is a state that can be prepared easily.

Order finding: quantum circuit

We previously saw that $|u_s\rangle$ has eigenvalue $e^{2\pi i s/r}$ – i.e., its phase is s/r, so we now run QPE, with the second register in the state $|1\rangle$ (note this is $|1\rangle$ such that 1 is an n-bit binary number, rather than simply $[0,1]^T$).



The measurements will collapse the state into one of the eigenstate components of $|1\rangle$ because each digital eigenvalue phase estimation in the first register will be entangled with its corresponding eigenvector in the second register:

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|0\rangle^{\otimes n}|u_s\rangle \to \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|\widetilde{s/r}\rangle|u_s\rangle$$

Thus QPE returns an estimate of the phase $\widetilde{s/r}$ for some (unknown) integer s, however there is a classical algorithm (the *continued fractions algorithm*) that can extract r from $\widetilde{s/r}$ with high probability.

Reduction of factoring to order-finding

From the order-finding subroutine we have found r such that $x^r \mod N = 1$, i.e.,

$$(x^r - 1) \mod N = 0$$

If r is even this factorises:

$$(x^{r/2} - 1)(x^{r/2} + 1) \mod N = 0$$

For which there are four possibilities:

1. $x^{r/2} = 1 \mod N$: but we know this cannot actually occur, as r is the *least* integer satisfying $x^r = 1 \mod N$.
2. $x^{r/2} = -1 \mod N$: in which case the algorithm fails.
3. $(x^{r/2} - 1)(x^{r/2} + 1) = N$ in which case $(x^{r/2} - 1)$ and $(x^{r/2} + 1)$ are factors that we output.
4. $(x^{r/2} - 1)(x^{r/2} + 1) = kN$ for some $k \geq 2$, in which case there is a result that guarantees that one of gcd$((x^{r/2} + 1); N)$ or gcd$((x^{r/2} - 1); N)$ is a non-trivial factor of N, which we can run Euclid's algorithm to find and then output.

Shor's algorithm: success probability

A single run of Shor's algorithm only returns a factor with a certain probability. In particular:

1. The order-finding subroutine could return even r.
2. The order-finding subroutine could return r such that $x^{r/2} = -1 \mod N$.
3. It is possible that the classical post-processing required to extract the order from the phase can fail.

The probability that r is such that *neither* of the first two occur is at least one half. The probability that the order-finding subroutine is such that the classical post-processing correctly returns the order is at least $1 - \epsilon$ for some constant ϵ.

Therefore a single run of Shor's algorithm correctly returns a factor with probability $\mathcal{O}(1)$, which is acceptable, as the expected number of iterations needed to find a factor does not grow with n.

Shor's algorithm: computational complexity

Recall that the best classical algorithm for factoring requires $\exp(\Theta(n^{1/3}\log^{2/3} n))$ operations.

- Shor's algorithm (as we have expressed it here) calls two classical algorithms as subroutines – the prime power checker, and Euclid's algorithm – both of which require a number of operations that is only polynomial in n.
- The quantum circuit used in Shor's algorithm requires $\mathcal{O}(n^3)$ gates to perform the modular exponentiation, and $\mathcal{O}(n^2)$ gates to perform the QFT.
- As the success probability of a single run of Shor's algorithm is $\mathcal{O}(1)$, the number of iterations we expect does not grow with n.

Therefore Shor's algorithm can factor in a number of operations that is polynomial in the number of bits required to express the number being factored. An exponential speed-up compared to the best classical algorithm.

Summary

In this lecture we have studied Shor's factoring algorithm:

- Performing quantum phase estimation with $U|y\rangle = |(xy) \mod N\rangle$ to achieve order finding:
  - Implementing controlled-U^{2^j} using modular exponentiation.
  - Preparing a superposition of eigenstates $\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle$ as the input to the second register.
- Reduction of factoring to order-finding.