Concepts and Algorithms of Scientific and Visual Computing

–Quantum Algorithms–

CS448J, Autumn 2015, Stanford University

Dominik L. Michels

Grover's Search Algorithm

The NP-complete SAT problem is defined as follows.

Input: n-variable Boolean Formula φ in CNF
Output: $a \in \{0,1\}^n$ such that $\varphi(x) = 1$

As we will see, Grover's search algorithm published in [Grover 1996] solves this problem in

$$\text{poly}(n)2^{n/2}\text{-time},$$

in contrast to methods from classical computing which need

$$\text{poly}(n)2^{n}\text{-time}$$

in the case of a function φ computable in polynomial time.

Grover's Search Algorithm

We assume a single satisfying assignment x and make use of a n-qubit register with uniform state vector

$$u = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle.$$

The angle $\theta = \measuredangle(u, |a\rangle)$ is given by

$$\theta = \arccos(\langle u, a\rangle) = \arccos\left(\frac{1}{2^{n/2}}\right) > 0.$$

Hence $\theta < \pi/2$ and we consider $\pi/2 - \theta$ with $\sin\theta = 2^{-n/2} \leq \theta$.

Grover's Search Algorithm

The algorithm starts in state u and gets in each step closer to the state $|a\rangle$. If an angle $\pi/2 - \alpha$ is applied in a current state, an angle $\pi/2 - \alpha - 2\theta$ follows in the next step.

Therefore in

$$\mathcal{O}(1/n) = \mathcal{O}(2^{n/2})$$

steps it will be in a state v whose inner product with $|a\rangle$ is larger than $1/2$ implying that a measurement of the register will yield a with a probability of at least $1/4$.

Grover's Search Algorithm

To rotate a vector w towards $|a\rangle$ by θ. It is sufficient to perform two reflections around u and $e = \sum_{x \neq a} |a\rangle \perp |a\rangle$.

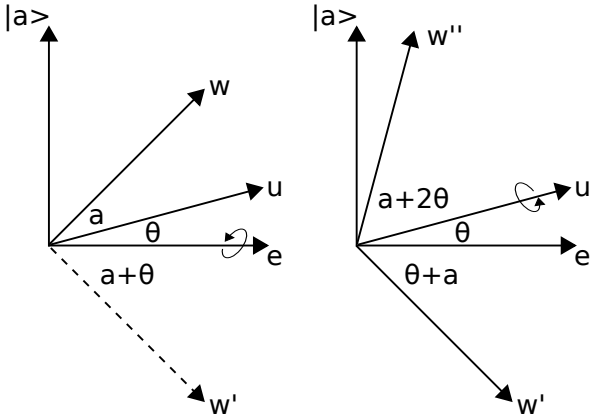

Figure : Illustration of the two reflection around e and u.

Grover's Search Algorithm

As we can see, we are able to transform w in the plane spanned by $|a\rangle$ and u into a vector w'' with a distance of 2θ radians to $|a\rangle$ using just two reflections around e and u.

Such operations can be performed in polynomial time.

Because of that we end up with an overall complexity of

$$\mathcal{O}\left(\text{poly}(n)2^{n/2}\right)$$

for the presented method.

Simon's Algorithm

Simon's problem is defined as follows.

Input: polynomial size circuit for a function $f : \{0,1\}^n \to \{0,1\}$, such there exists $a \in \{0,1\}^n$ with $f(x) = f(y)$ if and only if $x = y \oplus a$ for every $x, y \in \{0,1\}^n$
Output: a

Is can be seen as a black box model, in which the algorithm has to learn an appropriate a with an exponential or subexponential number of queries to the black box which returns $f(x)$ for a given x.

In classical computing we only know methods which determine such an a in exponential time. In contrast, Simon's algorithm published in [Simon, 1994] computes the result in subexponential time and is therefore in BQP.

Simon's Algorithm

We make use of a register with $2n + m$ qubits, in which m denotes the number of workspace bits needed for the computation of f.
We apply n Hadamard operations to set the first n qubits to the uniform state.
After that we apply

$$|xz\rangle \mapsto |x(z \oplus f(x))\rangle$$

to the register which leads to the state

$$\sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle = \sum_{x \in \{0,1\}^n} (|x\rangle |x \oplus a\rangle) |f(x)\rangle.$$

Now we measure the second n bits of the register, which collapses into the state $|xf(x)\rangle + |(x \oplus a)f(x)\rangle$ for some $x \in \{0,1\}^n$.

Simon's Algorithm

After that we apply n Hadamard operations to the first n bits.
Because of

$$x \mapsto \sum_y (-1)^{x \odot y} |y\rangle$$

the new state of this bits is given by

$$\sum_y \left((-1)^{x \odot y} + (-1)^{(x \oplus a) \odot y} \right) |y\rangle$$

$$= \sum_y \left((-1)^{x \odot y} + (-1)^{x \odot y} (-1)^{a \odot y} \right) |y\rangle.$$

In this state, for each $y \in \{0,1\}^m$ the y-th coefficient is unequal to zero if and only if $a \odot y = 0$. In the case of a measurement the state yields a uniform $y \in \{0,1\}^n$ which satisfies $a \odot y = 0$.

Simon's Algorithm

Repeating this procedure for k times leads to k uniform strings $y_1, \ldots, y_k$ with $y \odot a = 0$. Hence we have k linear equations over GF(2) regarding $a_1, \ldots, a_n$.

For example, for $k \geq 2n$ we have $n-1$ linearly independent equations with a high probability. Hence we are able to determine a from these equations with Gaussian elimination.

We end up with just a linear number of

$$\mathcal{O}(n)$$

operations in contrast to methods with exponential complexity from classical computing. It is also possoble to show, that every quantum algorithm needs $\Omega(n)$ operations to solve this problem.
Therefore Simon's Algorithm is an optimal quantum strategy.

Shor's Factorization Algorithm

Shor's Factorization Algorithm published [Shor 1994] is able to find the prime factors of a given $N \in \mathbb{N}$ with a polynomial complexity in the number of digits. It is therefore in

$$\mathrm{BQP}.$$

In contrast to that the best known method from classical computing needs

$$2^{\log^{1/3}(N)} \text{ steps}$$

to factor N; see [Lenstra 1990].

Shor's Factorization Algorithm

Before we are able to consider Shor's factorization algorithm, we have to focus on the quantum Fourier transform over $\mathbb{Z}_M$.

For that we consider the classical discrete Fourier transform over the finite group

$$\mathbb{Z}_M = (\{0, \ldots, M-1\}, +)$$

in which the "$+$" is interpreted as the addition modulo M.

This transform is defined for a vector $f \in \mathbb{C}^M$ by

$$\hat{f}(x) = \frac{1}{\sqrt{M}} \sum_{y \in \mathbb{Z}_M} f(y) \Omega^{xy},$$

in which $\Omega = e^{2\pi i/M}$ denotes a primitive M-th root of unity.

Shor's Factorization Algorithm

The Fourier transform can be performed by the multiplication of the $M \times M$-matrix

$$(\Omega^{xy})_{x,y \in \{0,\ldots,M-1\}}$$

with the input vector f.

As we have already seen, in order to realize an efficient computation of $\hat{f}$, it is a common approach to make use of the FFT FT_M, which is a

$$\mathcal{O}(M \log(M))\text{-method}$$

to perform the required matrix-vector multiplication.

Shor's Factorization Algorithm

For that we perform a splitting strategy in components with even and components with odd indices:

$$\hat{f}(x) = \frac{1}{\sqrt{M}} \sum_{y \in \mathbb{Z}_M} f(y)\Omega^{xy}$$

$$= \frac{1}{\sqrt{M}} \sum_{y \in \{0,2,\ldots,2\lceil \frac{M-1}{2} \rceil\}} f(y)\Omega^{-2x(y/2)}$$

$$+ \frac{\Omega^x}{\sqrt{M}} \sum_{y \in \{1,3,\ldots,2\lfloor \frac{M-1}{2} \rfloor + 1\}} f(y)\Omega^{2x((y-1)/2)}.$$

Shor's Factorization Algorithm

We exploit the facts, that Ω^2 is a $M/2$-th primitive complex root of unity and $\Omega^{M/2} = -1$. With the use of W, which denotes the $M/2 \times M/2$ diagonal matrix

$$\mathrm{diag}(\Omega^0, \ldots, \Omega^{M/2-1}),$$

we obtain the recursive scheme given by

$$FT_M(f)_{\mathrm{low}} = FT_{M/2}(f_{\mathrm{even}}) + W \cdot FT_{M/2}(f_{\mathrm{odd}})$$

and

$$FT_M(f)_{\mathrm{high}} = FT_{M/2}(f_{\mathrm{even}}) - W \cdot FT_{M/2}(f_{\mathrm{odd}}),$$

which leads to a divide et impera algorithm with logarithmic deep. Its runtime can be described by the recurrence relation

$$T(M) = 2 \cdot T(M/2) + \mathcal{O}(M)$$

leading to the complexity of $T(M) \in \mathcal{O}(M \log(M))$.

Shor's Factorization Algorithm

The quantum Fourier transform changes the state of a quantum register from $f \in \mathbb{C}^M$ to its Fourier-transformed version $\hat{f}$.

For every m and $M = 2^m$, it uses $\mathcal{O}(m^2)$ quantum gates to transform a quantum register in state

$$f = \sum_{x \in \mathbb{Z}_m} f(x)|x\rangle$$

into the state

$$\hat{f} = \sum_{x \in \mathbb{Z}_m} \hat{f}(x)|x\rangle,$$

where $\hat{f}(x) = \frac{1}{\sqrt{M}} \sum_{x \in \mathbb{Z}_m} f(x)\omega^{xy}$.

Shor's Factorization Algorithm

With the use of the scheme, which we known from the classical FFT, we are allowed to split the problem to compute FT_M into two identical subproblems of size $M/2$ involving computation of $FT_{M/2}$ which can be carried out recursively using the same elementary operations.

Let us run FT_M on the $m-1$ most significant qubits.
The state is given by

$$(FT_{M/2} f_{\text{even}})|0\rangle + (FT_{M/2} f_{\text{odd}})|1\rangle.$$

If the least significant bit is 1, compute W on the $m-1$ most significant qubits:

$$(FT_{M/2} f_{\text{even}})|0\rangle + (W \cdot FT_{M/2} f_{\text{odd}})|1\rangle.$$

Shor's Factorization Algorithm

The transform W on $m-1$ qubits can be defined by

$$|x\rangle \mapsto \Omega^x = \Omega^{\sum_{i=0}^{m-2} 2^i x_i}.$$

Application of a Hadamard gate H to the least significant qubit and moving it to the most significant position leads to

$$(FT_{M/2} f_{\text{even}})(|0\rangle + |1\rangle) + W \cdot FT_{M/2} f_{\text{odd}}(|0\rangle - |1\rangle) = \hat{f}.$$

The overall "trivial" complexity of the quantum Fourier transform is quadratic in m. In addition, the beste known algorithm only requires

$$\mathcal{O}(m \log(m)) \text{ [Hales 2000]}$$

gates to achieve an efficient approximation as in the case of the classical FFT.

Shor's Factorization Algorithm

By using the quantum Fourier transform we are able to specify Shor's order-finding algorithm.

This is a polynomial-time quantum method, which finds the smallest r such that

$$A^r = 1 \bmod N$$

for a given binary representation of A and N.

The factoring problem can be polynomially reduced to order finding. With the use of this classical result we get Shor's factorization algorithm–a

$$\mathrm{BQP}$$

–method, which finds the prime factors of a given integer with a polynomial complexity in the number of digits.

Shor's Factorization Algorithm

In order to prove this fact, we will sketch Shor's order-finding algorithm, which finds the order of $A \in \mathbb{N}$ modulo $N \in \mathbb{N}$. Let $m = \lceil 5 \log(M) \rceil$ and $M = 2^m$. The register

consists of

$$m + \mathrm{polylog}(N)$$

qubits.

The function

$$x \mapsto A^x \bmod N$$

can be computed with complexity $\mathrm{polylog}(N)$.

Therefore it is possible to compute

$$|x\rangle|y\rangle \mapsto |x\rangle|y \oplus \mathrm{B}(A^x \bmod N)\rangle,$$

in which $\mathrm{B}(X)$ denots the binary representation of X, in polynomial time by extending the register by additional $\mathrm{polylog}(N)$ qubits.

Shor's Factorization Algorithm

The first m qubits of the register are used to encode a number in $\mathbb{Z}_M$.

Apply the quantum Fourier transform to the first m qubits. The state is given by

$$\frac{1}{\sqrt{M}} \sum_{x \in \mathbb{Z}_m} |x\rangle |0^n\rangle .$$

Compute the transform

$$|x\rangle |y\rangle \mapsto |x\rangle |y \oplus \mathrm{B}(A^x \bmod N)\rangle ,$$

so that the state is given by

$$\frac{1}{\sqrt{M}} \sum_{x \in \mathbb{Z}_m} |x\rangle |A^x \bmod N\rangle .$$

Shor's Factorization Algorithm

Measure the second register to get y_0. The state is

$$\frac{1}{\sqrt{M}} \sum_{l=0}^{K-1} |x_0 + l_r\rangle |y_0\rangle,$$

in which x_0 is the smallest number with $A^{x_0} = y_0 \bmod N$ and $K = \lfloor (M-1-x_0)/r \rfloor$.

After that apply the quantum Fourier transform to the first register. The state is given by

$$\frac{1}{\sqrt{MK}} \left(\sum_{x \in \mathbb{Z}_m} \sum_{l=0}^{K-1} \omega^{(x_0+l_r)x} |x\rangle \right) |y_0\rangle.$$

Shor's Factorization Algorithm

Measure the first register to obtain a number $x \in \mathbb{Z}_M$.

Find a rational approximation a/b with coprime a and $b \leq N$, which approximates x/M with an accuracy of $1/(10M)$.

Finally return $r = b$, which satisfies $A^r = 1 \bmod N$.

We will analyze this method for the case

$$r|M.$$

The general case is more technical.

Shor's Factorization Algorithm

We consider the case $M = rc$ with some $c \in \mathbb{N}$. In this case the measured x is equal to ac for a random $a \in \{0, \ldots, r-1\}$.

Before the measurement compute for every $x \in \mathbb{Z}_M$ the absolute value of the coefficient of $|x\rangle$.

Up to normalization factors, this is

$$\left| \sum_{l=0}^{c-1} \omega^{(x_0 + l_r)x} \right| = \left| \omega^{x_0 c' c} \right| \left| \sum_{l=0}^{c-1} \omega^{rlx} \right| = \left| \sum_{l=0}^{c-1} \omega^{rlx} \right|.$$

If not $c|x$ then ω^r is a c-th root of unity.

Shor's Factorization Algorithm

From the geometric series follows

$$\sum_{l=0}^{c-1} \Omega^{rlx} = 0.$$

Hence x would be measured with probability 0.

If $x = cj$ we set

$$\Omega^{rlx} = \Omega^{rcjl} = \Omega^{Mj} = 1.$$

Hence for $j \in \{0, 2, \ldots, r-1\}$, the amplitudes of all x are equal.

Shor's Factorization Algorithm

Therefore the measured x is equal to ac for a random $a \in \{0, \ldots, r-1\}$.

This concludes the proof sketch because it implies

$$x/M = a/r,$$

where $a \leq r$ is a random integer.

For every r, we assume that $\Omega(r/\log(r))$ of the numbers in $[r-1]$ are coprime to r.

The prime number theorem states, that there are at least this many primes in this interval, and since r has at most $\log(r)$ prime factors, all but $\log(r)$ of these primes are coprime to r.

Hence by computing a rational approximation of x/M, the denominator is r.

Shor's Factorization Algorithm

For the case $r|M$ we have analyzed

Shor's order-finding algorithm.

As already mentioned the general case is more technical.

By polynomial reducing the factoring problem to order finding, we would get

Shor's factorization algorithm

which can be performed in polynomial quantum time.