

Quantum Algorithms

AMS Short Course

Peter Shor
MIT
Cambridge, MA

Outline

- Quantum Fourier Transform
- Phase Estimation
- Grover Search
- Matrix Inversion
- Brief Survey of Other Quantum Algorithms.

Quantum Fourier Transform

The quantum Fourier transform is one of the most widely used techniques in quantum computing. It is used in the algorithms for phase estimation and periodicity finding, which in turn give factoring and discrete log algorithms, algorithms for Pell's equation and class groups, and more.

QFT

- The quantum Fourier transform (over 2^m) is performed on a register containing m qubits, which should be viewed as representing an integer between 0 and $2^m - 1$

$$|x\rangle \rightarrow \frac{1}{2^{m/2}} \sum_{y=0}^{2^m-1} e^{-2\pi i xy / 2^m} |y\rangle$$

Quantum Fourier Transform

$$|x\rangle \rightarrow \frac{1}{2^{m/2}} \sum_{y=0}^{2^m-1} e^{-2\pi i xy / 2^m} |y\rangle$$

It is easy to check that this is a unitary operation.

Unitarity is not sufficient for efficient implementation on a quantum computer.

Why not? For one thing, there is a unitary transformation which takes any desired input into the desired output in one step. This is cheating.

We need to break unitary transformations up into products of 2-qubit gates.

Two-qubit gates

A 2-qubit gate U is a 4×4 matrix.

What does this mean when you're operating on a 2^n dimensional space?

Take the tensor product of U on its 2 qubits with the identity I on the other qubits.

If U acts on the 3rd and 18th qubit, this looks different from the tensor products you may be used to.

Example

Tensor product of Fourier transform on 1st and 3rd qubits with identity I on 2nd.

$$\begin{pmatrix} 1 & 1 & & 1 & 1 & & & \\ 1 & -i & & -1 & i & & & \\ & & 1 & 1 & & 1 & 1 & \\ & & & 1 & -i & & -1 & i \\ 1 & -1 & & 1 & -1 & & & \\ 1 & i & & -1 & i & & & \\ & & 1 & -1 & & 1 & -1 & \\ & & 1 & i & & -1 & -i & \end{pmatrix}$$

The QFT for $m = 3$.

We can rewrite

$$|x\rangle \rightarrow \frac{1}{2^{m/2}} \sum_{y=0}^{2^m-1} e^{-2\pi ixy/2^m} |y\rangle \text{ as}$$

$$|x\rangle \rightarrow \frac{1}{\sqrt{8}} \sum_y (-1)^{x_1y_3} (-1)^{x_2y_2} (-1)^{x_3y_1} (-i)^{x_2y_3} (-i)^{x_3y_2} (-i)^{x_3y_1/2} |y\rangle$$

where x_1, x_2, x_3 , are the bits of x .

continuing...

$$|x\rangle \rightarrow \frac{1}{\sqrt{8}} \sum_y (-1)^{x_1y_3} (-1)^{x_2y_2} (-1)^{x_3y_1} (-i)^{x_2y_3} (-i)^{x_3y_2} (-i)^{x_3y_1/2} |y\rangle$$

Now $|x_1\rangle \rightarrow \frac{1}{\sqrt{2}} (-1)^{x_1y_3} |y_3\rangle$

Is just the unitary matrix $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

This is called a Hadamard gate.

continuing...

$$|x\rangle \rightarrow \sum_y (-1)^{x_1y_3} (-1)^{x_2y_2} (-1)^{x_3y_1} (-i)^{x_2y_3} (-i)^{x_3y_2} (-i)^{x_3y_1/2} |y\rangle$$

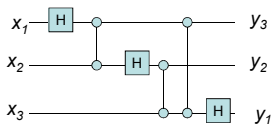
and $|x_2y_3\rangle \rightarrow (-i)^{x_2y_3} |x_2y_3\rangle$

Is just the unitary matrix $\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -i \end{pmatrix}$

continuing...

$$|x\rangle \rightarrow \sum_y (-1)^{x_1y_3} (-i)^{x_2y_3} (-1)^{x_2y_2} (-i)^{x_3y_2} (-i)^{x_3y_1/2} (-1)^{x_3y_1} |y\rangle$$

Can be performed by 6 unitary gates applied in the right order.



Phase estimation

Algorithm: Start with unitary U , and $|\varphi\rangle$ an eigenvector of U so

$$U |\varphi\rangle = e^{i\theta} |\varphi\rangle$$

Find a good approximation to the eigenvalue $e^{i\theta}$ of $|\varphi\rangle$.

We need to be able to do a controlled U^n , so

$$|0\rangle |\psi\rangle \rightarrow |0\rangle |\psi\rangle$$

$$|1\rangle |\psi\rangle \rightarrow |1\rangle U^n |\psi\rangle$$

Phase estimation

Suppose we start with qubits $|k\rangle |\varphi\rangle$ where k encodes a number in binary.

For each t , we apply a controlled U^{2^t} with the control being the t^{th} bit of k .

We get $|k\rangle U^k |\varphi\rangle = e^{ik\theta} |k\rangle |\varphi\rangle$.

Phase estimation (cont.)

With controlled U , we can do

$$\frac{1}{2^{n/2}} \sum_k |k\rangle |\varphi\rangle \rightarrow \frac{1}{2^{n/2}} \sum_k e^{ik\theta} |k\rangle |\varphi\rangle$$

With an inverse Fourier transform, we have

$$|j\rangle |\varphi\rangle \rightarrow \frac{1}{2^{n/2}} \sum_k e^{2\pi ijk/2^n} |k\rangle |\varphi\rangle$$

So if $\theta = 2\pi j/2^n$, by following the first transform by the inverse of the second, we find θ .

Phase estimation

Even if θ is not an integer multiple of $2\pi/2^n$, we get a very good approximation of θ this way. (The analysis involves the sum of a geometric series.)

Periodicity finding

- Suppose we have a periodic function (e.g., one that takes $|k\rangle \rightarrow |k+1 \bmod N\rangle$)
- Then it's eigenvalues are $e^{2\pi ij/N}$ for integer j
- We can use the phase estimation algorithm to approximately find j/N , and then use continued fractions to find N exactly.
- This enables us to factor and find discrete logs.

Grover's algorithm

Suppose we have a number of "good" answers (we want to find one of these), and we have a transformation G that takes

$G |x\rangle \rightarrow -|x\rangle$ if x is good, and

$G |x\rangle \rightarrow |x\rangle$ if x is not good.

Then, if the universe has size N , we can find a marked state in time $O(\sqrt{N})$

Classically, $O(N)$ is optimal.

Grover's algorithm

Let the space size N be 2^n .

We need several unitary transformations.

$$H^{\otimes n} \text{ where recall } H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

G , which applies -1 to the good states.

G_0 , which applies -1 to the state $|0\rangle$.

Grover iteration

Grover's algorithm consists of repeatedly applying the transformation

$$H^{\otimes n} G_0 H^{\otimes n} G$$

The transformation $H^{\otimes n} G_0 H^{\otimes n}$ takes

$$\sum \alpha_x |x\rangle \rightarrow \sum (2\bar{\alpha} - \alpha_x) |x\rangle$$

"Invert around average."

Proof of "invert around average"

We have $G_0 = \frac{2}{N} \sum |j\rangle\langle j| - I$

Since $H^{\otimes n} |0\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle$

we get

$$H^{\otimes n} G_0 H^{\otimes n} = \frac{2}{N} \left(\sum |j\rangle \right) \left(\sum \langle j| \right) - I$$

or invert around average.

Grover's algorithm

Grover's algorithm starts in an equal superposition of all states.

Each iteration increases the amplitude of the good states by roughly twice the average amplitude.

After approximately $O(\sqrt{N/M})$ iterations, the good states contain nearly all the amplitude, where M is the number of good states

Grover's algorithm

A more careful analysis can be done by using the fact that the algorithm always leaves the system in a two-dimensional subspace.

We need roughly $\frac{\pi}{4} \sqrt{N/M}$ iterations

Matrix inversion

Suppose A is a Hermitian matrix, and $|b\rangle$ is a unit vector. How fast can we find $A^{-1}|b\rangle$?

If A is $n \times n$, a quantum computer might be able to find it in only $O(\log n)$ steps.

Some caveats: this algorithm has a polynomial dependence on the condition number κ of A , and we only get the answer to some accuracy.

(Harrow, Hassidim, Lloyd)

Matrix inversion

Suppose you have a Hermitian matrix A . Then e^{itA} is unitary.

Further, if A has small entries, and is sparse, you can perform the transformation e^{itA} efficiently on a quantum computer, in time polynomial in t and $1/\epsilon$. (Berry et al).

The basic idea is to show that if t is small enough, this is straightforward, and then repeatedly apply e^{itA} for small t . Physicists need to do this a lot, so they have developed quite a few methods for improving efficiency by using bigger step sizes, etc.

Using phase estimation

If you can perform e^{itA} , you can use phase estimation. This gives

$$|b\rangle \rightarrow \sum \alpha_{\tilde{\varphi}} |\tilde{\varphi}\rangle |\tilde{v}_{\tilde{\varphi}}\rangle$$

where $\tilde{\varphi}$ is an estimate for eigenvalue of A , and we have decomposed $|b\rangle$ into approximate eigenvectors of A .

Matrix inversion

Now we simply multiply by the complex exponential of our estimate of φ^{-1} and uncompute φ .

$$|b\rangle \rightarrow \sum \alpha_{\tilde{\varphi}} |\tilde{\varphi}\rangle |\tilde{v}_{\tilde{\varphi}}\rangle \rightarrow \sum \alpha_{\tilde{\varphi}} e^{it/\tilde{\varphi}} |\tilde{v}_{\tilde{\varphi}}\rangle$$

This is an approximation to $e^{it/A} |b\rangle$ and how good an approximation it is depends on the condition number κ of A

$$\text{But } e^{itA^{-1}} |b\rangle = |b\rangle + itA^{-1} |b\rangle + O(t^2)$$

So we're getting close to what we want.

Matrix inversion

Instead of A , we can use $C = \sigma_x \otimes A$.

where $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and

$$e^{itC^{-1}} |0\rangle |b\rangle = |0\rangle |b\rangle + it |1\rangle A^{-1} |b\rangle + O(t^2)$$

We can use Grover's algorithm to pick out the component with $|1\rangle$ in the first qubit.

Limits on matrix inversion

If we can perform matrix inversion in time faster than $\kappa^{1/2}$, then we can perform quantum computation faster than we think is possible.

$$A = I - \sum_{t=0}^{T-1} |t+1\rangle \langle t| \otimes U_t e^{-t/T}$$

Then we find $\kappa = T$, and

$$A^{-1} |0\rangle |x\rangle = \sum_{t=0}^{T-1} |t\rangle U_t \dots U_1 |x\rangle e^{-t/T}$$

This has as its largest amplitude the result of the quantum computation starting with x and applying U_1, U_2, \dots, U_t . It is believed impossible to do this much faster than T time.

Limits on matrix inversion algorithms

One problem with the above analysis: A is not Hermitian. We need to add the Hermitian conjugate to make it Hermitian.

When we do, we get that $\kappa = T^2$, so if we could do matrix inversion faster than $\kappa^{1/2}$, we could speed up quantum computation.

The conjugate gradient method runs in $\kappa^{1/2}$, and this shows that it is optimal (this was already known, but maybe not in the same space of algorithms that the quantum proof shows).

Phase estimation

Putting it all together, we get an algorithm that runs in something like $\kappa^2 \log n / \epsilon^{1/2}$.

This is substantially improved from the estimate in the current version of the paper, and shouldn't be completely trusted until it is written down carefully and checked.

Traditional Classification of Quantum Algorithms

- Simulating quantum mechanics (Feynman, Manin)
- Periodicity (e.g., Simon's algorithm, factoring, discrete log, Pell's eq., etc).
- Grover search (solving NP-complete problems more quickly, searching).

Markov chain Grover speedup

- Szegedy showed how to speed up Grover search when the objects you are searching for are on a graph and you can go from one to another via a Markov chain.

Speeding up searching with random walks

- Suppose you have a classical algorithm where you are searching for an object using a random walk. It takes time T to initialize the random walk and time S to search once you've initialized it.
- Szegedy showed how you can solve the same problem on a quantum computer in time $O(T + \sqrt{S})$. (See also Magniez, Nayak, Roland, Santha)
- Has applications to various algorithms.

Hidden subgroup problem

- Algorithms have been discovered for various hidden subgroup problems for non-abelian groups.
- Efficient solution to the hidden subgroup problems for other non-abelian groups would provide efficient quantum algorithms for finding short vectors in a lattice and for solving graph isomorphism.

Hidden subgroup algorithms

- Some of these algorithms use representations and characters of finite groups.
- The algorithm for the dihedral group uses sieving over characters of D_N .
- The algorithm for the Heisenberg group can be expressed in terms Klebsch-Gordon transform over the Heisenberg group.

Can hidden subgroup algorithms solve graph isomorphism?

- If you can find hidden subgroups over the symmetric group S_n , then you can solve graph isomorphism
- Several papers have shown severe constraints on how you can use the non-abelian quantum Fourier transform to find hidden subgroups in S_n .

Evaluating game trees (Farhi, Goldstone, Gutman)

- Quantum random walks have led to a new algorithm to evaluate game trees in $O(\sqrt{N})$ time.
- You can set up a Hamiltonian over a graph consisting of a line adjoined to the game tree, and an initial state of this system, so that a wave undergoes reflection or not depending on whether the value is 0 or 1.

Adiabatic algorithms

- Adiabatic quantum algorithms are based on the idea that if you change the Hamiltonian of a system slowly enough, the system will stay in the ground state.
- So start with a Hamiltonian that has a known ground state, and change it slowly to one whose ground state solves the problem you want solved.
- How slowly do you need to change it? It depends on the gap between the lowest and second lowest energy states of the Hamiltonian.

Does this work?

- If this worked in general, then we would be able to solve NP-complete problems on a quantum computer, so this seems unlikely.
- For random satisfiability problems, there are methods which will let you approximate the behavior of several hundred qubits. Some computations have been done which seem to suggest that the energy gap is inverse polynomial. This would imply that the algorithm works for random satisfiability problems. No classical algorithm is known for these problems.

Statistical Mechanics

- Quadratically signed weight enumerators can be computed with a quantum computer.
- Certain partition functions can be computed more efficiently with a quantum computer.

Higher math

- Sean Hallgren gave algorithm for solving Pell's equation. Kirin Kedlaya and Wim van Dam separately gave algorithms for finding zeroes of zeta functions over curves on finite fields.
- Kedlaya's algorithm uses periodicity. Van Dam's algorithm uses quantum computation and phase estimation to find the spectrum of a matrix.
- It seems as though there may be some common generalization of these algorithms which is as yet undiscovered.

Jones polynomial

- In 2002, it was shown that approximating the Jones polynomial at a root of unity is BQP-complete.
- In 2007, it was shown that approximating the Jones polynomial at a root of unity is OQC-complete.
- These are not contradictory, because the closeness of the approximation is important.
- These approximation algorithms have been generalized to hold for other knot invariants and also to computing representations of the symmetric group.

How to compute knot invariants with a quantum computer.

- Many knot invariants are connected to unitary representations of the braid group.
- These are can be approximated using the unitary dynamics of quantum computation.
- You can express the Jones polynomial at a root of unity as the probability of obtaining $|1\rangle$ in the first bit of a quantum computation.
- To find a good approximation of the value of the knot invariant, you need to sample repeatedly (it is quite possible this approximation will be 0).

Approximating knot invariants is as hard as quantum computing.

- For any quantum computation, you need to find a knot, the approximation of whose Jones polynomial gives you the answer to you quantum computation.
- You can simulate the unitary dynamics of the quantum computer with a unitary representation of a braid group.

What important open problems look promising?

Unfortunately, the list has not changed much recently.

- Lattice problems
- Graph isomorphism
- Hidden subgroup problems

So far, we have a history of several years of not solving these. Anything else?

- Once quantum computers are built, maybe playing around with them will give us new and interesting algorithms.
- This won't happen if the funding agencies get discouraged because nobody has discovered any more useful algorithms.
- Conclusion: we need to discover more algorithms before quantum computers are actually built. How?