Quantum computing - a brief review from algorithms to platforms

Kuei-Lin Chiu

Department of Physics
Massachusetts Institute of Technology

Key Lab of Quantum Information, University of Science and Technology of China (current position)

2017/12/15 NCTU, Taiwan
Outline

• Introduction to quantum computing

• The Algorithms
  ➢ Deutsch-Jozsa algorithm (judging)
  ➢ Grover’s algorithm (searching)
  ➢ Shor’s algorithm (factoring)

• The platforms
  ➢ Superconducting circuits
  ➢ Semiconductor quantum dots

• Progress and prospect
What is quantum computing?

- Quantum computing uses the properties of quantum mechanics to design hardware and algorithms, and to perform certain calculations which are usually difficult for classical computers to complete.
- The unit of quantum computing is quantum bits ("qubits"), in comparison with the "bits" used in classical computing.

**Classical Computation**

Data unit: bit

- $\blacksquare = '1'$
- $\Box = '0'$

Valid states:

- $x = 0$ or $1$

<table>
<thead>
<tr>
<th>$x = 0$</th>
<th>$x = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="0" alt="0" /></td>
<td><img src="1" alt="1" /></td>
</tr>
</tbody>
</table>

**Quantum Computation**

Data unit: qubit

- $\uparrow = |1\rangle$
- $\downarrow = |0\rangle$

Valid states:

- $|\psi\rangle = c_1|0\rangle + c_2|1\rangle$

| $|\psi\rangle = |0\rangle$ | $|\psi\rangle = |1\rangle$ | $|\psi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ |
|-------------------|-------------------|-------------------|
| ![0](0)           | ![1](1)           | ![0](0)           |
When we measure a quantum state, it can be quite different. Classical Computation

<table>
<thead>
<tr>
<th>State</th>
<th>Result of measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '0'</td>
<td>'0'</td>
</tr>
<tr>
<td>x = '1'</td>
<td>'1'</td>
</tr>
</tbody>
</table>

Quantum Computation

<table>
<thead>
<tr>
<th>State</th>
<th>Result of measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>\psi\rangle =</td>
</tr>
<tr>
<td>$</td>
<td>\psi\rangle =</td>
</tr>
<tr>
<td>$</td>
<td>\psi\rangle =</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Single qubit</th>
<th>Hilbert space $\mathcal{H}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>0\rangle,</td>
</tr>
<tr>
<td>$\mathcal{H}_2 = \begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Two qubits</th>
<th>Hilbert space $\mathcal{H}_2 \otimes \mathcal{H}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>00\rangle,</td>
</tr>
<tr>
<td>$\mathcal{H}_2 \otimes \mathcal{H}_2 = \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td></td>
</tr>
</tbody>
</table>

| Arbitrary state | $|\psi\rangle = c_1|0\rangle + c_2|1\rangle = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$ |
|----------------|-------------------------------------------------|

| Operator $U$ | $U|\psi\rangle = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$ |

| Operator $U$ | $U|\Psi\rangle = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & u_{44} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$ |
Operation of qubits is through quantum gates

### One-qubit Gate

**Hadamard gate** (rotate state around $y$ by $\pi/2$)

\[
H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}
\]

- Hadamard gate \[H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\]
- Hadamard gate \[H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\]

**Pauli-X gate** (rotate state around $x$ by $\pi$)

\[
\alpha|0\rangle + \beta|1\rangle \rightarrow X \rightarrow \beta|0\rangle + \alpha|1\rangle
\]

**Pauli-Z gate** (rotate state around $z$ by $\pi$)

\[
\alpha|0\rangle + \beta|1\rangle \rightarrow Z \rightarrow \alpha|0\rangle - \beta|1\rangle
\]

### Two-qubit Gate

**Controlled-NOT gate**

\[
\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\]

Input | Output
--- | ---
0 0 | 0 0
0 1 | 1 1
1 0 | 1 0
1 1 | 0 1

Arbitrary quantum logic gate can be decomposed into one-qubit and two qubit gates.
Quantum algorithm

- The difference between polynomial-time and exponential-time algorithm

If the dimension of a question is $n$, and the steps (or said the time) required to solve this question is $T(n)$, which can be polynomial or exponential function of $n$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n=10$</th>
<th>$n=20$</th>
<th>$n=30$</th>
<th>$n=40$</th>
<th>$n=50$</th>
<th>$n=60$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$10^{-5}s$</td>
<td>$2 \times 10^{-5}s$</td>
<td>$3 \times 10^{-5}s$</td>
<td>$4 \times 10^{-5}s$</td>
<td>$5 \times 10^{-5}s$</td>
<td>$6 \times 10^{-5}s$</td>
</tr>
<tr>
<td>$n^2$</td>
<td>$10^{-4}s$</td>
<td>$4 \times 10^{-4}s$</td>
<td>$9 \times 10^{-4}s$</td>
<td>$16 \times 10^{-4}s$</td>
<td>$25 \times 10^{-4}s$</td>
<td>$36 \times 10^{-4}s$</td>
</tr>
<tr>
<td>$n^3$</td>
<td>$10^{-3}s$</td>
<td>$8 \times 10^{-3}s$</td>
<td>$27 \times 10^{-3}s$</td>
<td>$64 \times 10^{-3}s$</td>
<td>$1.25 \times 10^{-1}s$</td>
<td>$2.16 \times 10^{-1}s$</td>
</tr>
<tr>
<td>$2^n$</td>
<td>$1024 \times 10^{-6}s$</td>
<td>$\sim 1s$</td>
<td>$\sim 1000s$</td>
<td>12.7 days</td>
<td>35.7 years</td>
<td>366 centuries</td>
</tr>
<tr>
<td>$3^n$</td>
<td>$5.9 \times 10^{-2}s$</td>
<td>58 mins</td>
<td>6.5 years</td>
<td>3855 centuries</td>
<td>$2 \times 10^8$ centuries</td>
<td>$1.3 \times 10^{13}$ centuries</td>
</tr>
</tbody>
</table>

*The age of earth is roughly $4.5 \times 10^7$ centuries*

In math, if the complexity of a problem grow exponentially with its input dimension, we refer this problem to NP (non-deterministic polynomial)
Deutsch-Jozsa Algorithm

• Used to determine whether a function is “constant” or “balanced”

• For n-bit input $x=\{0,1\}^n$ (meaning (0 or 1, 0 or 1...,0 or 1), in total $2^n$ possible combinations):
  
  ➢ $f(x)=0$ (or 1) for all $x$, it is called “constant”
  
  ➢ $f(x)=0$ for half of $x$ and $f(x)=1$ for the other half, it is called “balanced”

• In classical algorithm it takes $T(2^n)$ steps to verify while in D-J algorithm it only takes $T(n)$

A simplified example for classical algorithm: if $x=1,2,3...,8$ (i.e. $n=3$); you need to try $f(1)$, $f(2)$...each by each. Let’s say you tried the first half input and found $f(1)=f(2)=f(3)=f(4)=0$; then you need to try the 5th input; if $f(5)=0$ then $f(x)$ is “constant”, however if $f(5)=1$ then $f(x)$ is “balanced”. So the maximal times of tries is 5 if you are unlucky. In general, for a n-bit input, you need to try $(2^n/2)+1$ times.
A simplified example of D-J Algorithm

- For a 1-bit (x=0, 1) input: if f(0)=f(1), f(x) is constant; if f(0)≠f(1), f(x) is balanced. In classical algorithm, you need to try 2 times to find out.
- See how quantum algorithm works differently. If we define an “Oracle” (applying on a 2-qubit state, however the 2nd qubit is an ancilla and will be disregarded in the end) $U_f$: $|x>|y> \rightarrow |x>|y \oplus f(x)>$ and let the input $|y>$ be a superposition state $(|0>-|1>)$:

$$U_f: |x>(|0>-|1>) \rightarrow |x>(|0 \oplus f(x)>-|1 \oplus f(x)>)=(-1)^{f(x)} |x>(|0>-|1>)$$

The effect of Oracle is simply adding a phase factor related with f(x)
- Let $|x>=|0>+|1>$, and operate Oracle $U_f$: $(|0>+|1>)(|0>-|1>) \rightarrow [(-1)^{f(0)} |0> +(-1)^{f(1)} |1>](|0>-|1>)$

First qubit
- Now project the first qubit onto the basis of $|\pm>=(|0>\pm|1>)$
  - If we get $|+>$, meaning $f(0)=f(1)=0$ (or 1), the function is constant
  - If we get $|->$, meaning $f(0)=0$, $f(1)=1$, the function is balanced

➤ We only need to perform Oracle 1 time to determine the function
➤ D-J algorithm is not useful in practical application (i.e., for 2-qubit case, if f(00)=0, f(01)=f(10)=f(11)=1, then f(x) is neither constant nor balanced). However, it serves as a good example to see how quantum algorithm operates differently than classical one
Grover's Search Algorithm

- Imagine we are looking for the solution to a problem with \( N \) possible inputs. We have a black box (or “oracle”) that can check whether a given answer is correct.

**Question:** I’m thinking of a number between 1 and 100. What is it?

**Classical computer**

1. \( \text{Oracle} \) → No
2. \( \text{Oracle} \) → No
3. \( \text{Oracle} \) → Yes

... 

**Quantum computer**

1+2+3+... → No+No+Yes+No+...

Superposition over all \( N \) possible inputs.

Using Grover’s algorithm, a quantum computer can find the answer in \( \sqrt{N} \) queries!
A simple example (search 1 out of 4)

For a 2-qubit input, we have $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. Let’s say $|10\rangle$ is the answer, so $f(00)=f(01)=f(11)=0$ but $f(10)=1$.

H-gate prepare $|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ and $|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$, so the total wavefunction before the oracle is

$$\frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

input  ancilla

Then the whole wavefunction goes through oracle, because of the $(-1)^{f(x)}$ phase, there will be a minus sign in front of the answer state:

$$\frac{1}{2} (|00\rangle + |01\rangle - |10\rangle + |11\rangle) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

When you measure the four basis, they still give you same probability. So the key is to transform the phase difference in $|10\rangle$ into an amplitude difference for us to measure
This can be simply done by a matrix $D$ composed of different quantum gates:

$$D = \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}$$

Apply on wave function:

$$D \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- Hadamard gate
- $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

- Pauli-X gate
- $\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

In classical search:

$$\bar{N} = \frac{N}{\sum_{i=1}^{N} 1} = \frac{1}{N} \sum_{i=1}^{N} t = \frac{N + 1}{2} \approx \frac{N}{2}$$

In Grover’s search, the average times of queries is $\sqrt{N}$.
Shor’s algorithm
To factor an odd integer $N$ (let’s say 21)

- Integer factorization is a NP, and forms the basis of RSA cryptosystem \((\text{Given } N=pq, \text{ find } p \text{ and } q)\).
- 1. Pick up an integer $a$ (said 2), $1 < a < N$. Define a function $f(x) = a^x \mod N$
- 2. Find the periodicity $r$ of $f(x)$ \((2^x \mod 21 \text{ is } 1, 2, 4, 8, 16, 11, 1, 2..; \text{ so } r \text{ for } 2^x \text{ is 6})\)
- 3. If $r$ is odd, go back to step 1 and choose another $a$. If not, compute $f(r/2)$ \((f(3) = 8)\)
- 4. gcd($a^{r/2} + 1$, $N$) and gcd($a^{r/2} - 1$, $N$) are both nontrivial factors of $N$. We are done. (gcd of 9 and 21 is 3, gcd of 7 and 21 is 7; 3 and 7 are the prime factors of 21)

Only step 2 is performed by quantum computing, the rest of the steps are still classical.
• The quantum part of Shor algorithm is a bit complicated. However, let’s get a feeling on how to search period in a given function using quantum algorithm.

• F(x)=½(cos(πx)+1), x could be the states span by 3 qubits (x=|0>,|1>,..,|7>). When x is even, f(x)=1; when x is odd, f(x)=0. The period of f(x) is 2. Our mission is to confirm that

\[
\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle = \frac{1}{\sqrt{8}} \left[ |0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle + |2\rangle|f(2)\rangle + \cdots + |7\rangle|f(7)\rangle \right] \quad (n=3)
\]

• Measure the second register, ex: if we get |0>, then the wavefunction collapses to |φ>=½(|1>+|3>+|5>+|7>)|0>

• Using the quantum Fourier transform (QFT), we expand the function:

\[
|φ\rangle = \left( |0\rangle + e^{i\frac{18\pi}{4}} |1\rangle + e^{i\frac{12\pi}{4}} |2\rangle + \cdots + e^{i\frac{17\pi}{4}} |7\rangle \right)
\]

\[
|X\rangle \rightarrow \frac{1}{\sqrt{8}} \sum_{k=0}^{7} e^{2\pi i k x/8} |k\rangle
\]

QFT

\[
\text{Where QM plays a role: Cancel out only } |0\rangle \text{ and } |4\rangle \text{ survive}
\]

\[
\text{X=1}
\]

\[
\text{X=3}
\]

\[
\text{X=5}
\]

\[
\text{X=7}
\]

\[
\text{only } |0\rangle \text{ and } |4\rangle \text{ survive}
\]
\[ |\varphi\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |4\rangle) \]

We have equal probability to measure \( |0\rangle \) or \( |4\rangle \). If we get \( |0\rangle \) we can’t find the period, but if we get \( |4\rangle \), we can.

Compare with the strictly derived final state after QFT (which we skip), 4=r*1=N/8, r=2 (done)

The power of Shor’s algorithm is to utilize the superposition property of QM, which makes the “unnecessary” information interfere destructively and the “useful” information interfere constructively in FT.

Factoring an integer with n-bits

Classical algorithm takes time \( O(\exp(n^{1/3})) \)

Shor’s quantum algorithm takes time \( O(n^2 \log n) \)

Source: New Enterprise Associates
Superconducting qubit

- A quantum LC resonator using Josephson junction as an inductor provide anharmonic states for a two-level system.

Quantized electrical harmonic oscillator

\[ H = \frac{\hat{\Phi}^2}{2L} + \frac{\hat{q}^2}{2C} \]

- \( q \rightarrow \) capacitor charge (momentum \( p \))
- \( \phi \rightarrow \) inductor flux/phase (position \( x \))

Josephson junction

- Current: \( I = I_c \sin \phi \)
- Voltage: \( V = (\Phi_0/2\pi)(d\phi/dt) \)
- Inductance: \( V = L_j \, dl/dt \)
Transmon qubit

Single qubit gate operation with state-dependent cavity readout

Relaxation and decoherence time

Decoherence time/gate operation time = 1 μs/15 ns

Quantum dot qubits

Single quantum dots

Double quantum dots

Drawback: large static B-field is required

RevModPhys.79.1217
2 qubits CNOT gate in Si QDs

\[ H(t) = J(t) \left( S_L \cdot S_R - \frac{1}{4} \right) + S_L \cdot B_L + S_R \cdot B_R \]
Progress and prospect

Gate operation time ~ ns

Gate operation time ~ 100 ms

Gate operation time ~ 100 ns

Thank you for your attention!