Introduction to Quantum Computing
Lecture 6: Complexity and Quantum Algorithms I

Petros Wallden

School of Informatics, University of Edinburgh

5th October 2017
- Computational Complexity: Classification of problems according to their difficulty. We usually measure the amount of resources (e.g. time, space, gates) used by an algorithm as a function of the input size.

- $f(n)$ is in $O(g(n))$ if for some constant $m$ there exists a positive constant such that $f(n) \leq cg(n)$ for all $n \geq m$

- $f(n)$ is in $\Omega(g(n))$ if for some constant $m$ there exists a positive constant $c$ such that $f(n) \geq cg(n)$ for all $n \geq m$

- $f(n)$ is in $\Theta(g(n))$ if for some constant $m$ there exists positive constants $c_1 \leq c_2$ such that $c_1g(n) \leq f(n) \leq c_2g(n)$ for all $n \geq m$

- Complexity class: Is a set of problems with related resource-based complexity
Some complexity classes

1. **P**: Solved by a deterministic Turing machine in polynomial time
2. **NP**: Problems verifiable in polynomial time by deterministic Turing machine. Also, “yes” instances can be accepted in polynomial time by a non-deterministic Turing machine
3. **PSPACE**: Problems solved by Turing machine using polynomial amount of space (irrespective of the time needed)
4. **BPP**: Problems solved by probabilistic Turing machine in polynomial time with bounded-error
5. **BQP**: Problems solved by probabilistic quantum computer in polynomial time with bounded-error
- We consider families of circuits $\{C_n\}$ where $C_n$ takes inputs of size $n$
- There is a procedure (e.g. a Turing machine) that generates the circuit diagrams
- Size Complexity of $C_n = |C_n|$ the number of gates

The following two statements hold:
- $\text{BQP} \subseteq \text{PSPACE}$
- Problems solved by quantum computers can be solved by classical computers (but may require exponential long time)
- $\text{BPP} \subseteq \text{BQP}$
- Quantum computers are at least as efficient as probabilistic Turing machines
- There are two versions of this thesis

1. All physical computable functions can be computed by a Turing machine
   - Nothing about the efficiency of this computation.
   - Quantum computers do not affect this statement ($BQP \subseteq PSPACE$)

2. A probabilistic Turing machine can efficiently simulate any realistic model of computation
   - If $BPP \subset BQP$ then (2) is wrong!
     (this seems to be the case)
There are two versions of this thesis:

1. All physical computable functions can be computed by a Turing machine
   - Nothing about the efficiency of this computation.
   - Quantum computers do not affect this statement (BQP ⊆ PSPACE)

2. A quantum Turing machine can efficiently simulate any realistic model of computation
The Oracle Model

- Input: function $f$, given as black box (above)
- Use unitary implementation of this box (below)

- Goal: determine some information about $f$ making as few queries to $f$ (and other operations) as possible
Deutsch-Jozsa Algorithm

- Inspiration for Shor’s and Grover algorithms
- Initial protocol by Deutsch 1985, improved by Jozsa. Current version, is result of further research (Cleve, Ekert, Macchiavello and Mosca)
- Input: A boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$
- Promise: The function is either constant (i.e. for all inputs it gives 0 or for all inputs it gives 1) or balanced (i.e. for half of the bits it gives 0 and for half of them it gives 1).
- Output: The function is constant or is balanced

Classically, to know with certainty (100%) we need at least $2^n/2 + 1$ queries

The oracle maps: $|x\rangle |y\rangle \rightarrow |x\rangle |y \oplus f(x)\rangle$
Deutsch-Jozsa Algorithm

We start with the state $|0\rangle^n |1\rangle$. Apply Hadamard to all

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle (|0\rangle - |1\rangle)$$

Apply the function

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle (|f(x)\rangle - |1 \oplus f(x)\rangle)$$

which can be rewritten as

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle)$$

Apply again Hadamard to the first $n$ qubits (now we ignore the last qubit)

$$\frac{1}{2^n} \sum_{y=0}^{2^n-1} \left( \sum_{x=0}^{2^n-1} (-1)^{f(x)} (-1)^{x \cdot y} \right) |y\rangle$$

where $x \cdot y = x_0 y_0 \oplus x_1 y_1 \oplus \cdots \oplus x_{n-1} y_{n-1}$ is the sum bitwise of the product
Finally we measure the probability of getting $|0\rangle^\otimes n$

$\left| \frac{1}{2^n} \sum_{0}^{2^n-1} (-1)^{f(x)} \right|^2$

and this is 1 if $f(x)$ is constant and is 0 if $f(x)$ is balanced

- Constitutes the first quantum speed up, since we required a single query of the function!
- Algorithm is deterministic (not in BQP, but in EQP (Exact Quantum Polynomial time) which is the quantum version of P)
- Not a speed up compared to BPP (exist efficient classical probabilistic algorithm)
Quantum Fourier Transform

- The Discrete Fourier Transform (DFT) takes a \( N \)-dimensional complex vector \((x_0, \cdots, x_{N-1})\) and maps it to \((y_0, \cdots, y_{N-1})\)

\[
y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \exp\left(2\pi ijk/N\right)
\]

We define \( \omega = \exp(2\pi i/N) \) the \( N \)th root of unity

\[
y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega^{jk}
\]

The Quantum Fourier Transform (QFT) takes a general quantum state

\[
|\psi\rangle = \sum_k x_k |k\rangle \rightarrow |\bar{\psi}\rangle = \sum_k y_k |k\rangle
\]

where the coefficients \( x_j, y_j \) are related as in the DFT

- Notation:

\[
|x\rangle = |x_1 x_2 \cdots x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle
\]

where \( x = x_1 2^{n-1} + x_2 2^{n-2} + \cdots + x_n 2^0 \)

\[
[0.x_1 \cdots x_m] = \sum_{k=1}^{m} x_k 2^{-k},
\]

e.g. \( [0.x_1 x_2] = \frac{x_1}{2} + \frac{x_2}{2^2} \)
The QFT is expressed (note $N = 2^n$ is power of 2):

$$|x_1x_2 \cdots x_n\rangle \rightarrow \frac{1}{\sqrt{N}} \left( |0\rangle + e^{2\pi i [0.x_n]} |1\rangle \right) \otimes \left( |0\rangle + e^{2\pi i [0.x_{n-1}x_n]} |1\rangle \right) \otimes \cdots \otimes \left( |0\rangle + e^{2\pi i [0.x_1x_2 \cdots x_n]} |1\rangle \right)$$

To see that this is the same with

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{2^n-1} e^{2\pi i jk/2^n} |k\rangle$$

write $k = k_1 2^{n-1} + \cdots + k_n 2^0$ and break the sum to sums $k_l \in \{0, 1\}$

This allows us to construct a circuit that implements QFT and since it is composed of unitary gates, proves that QFT is unitary.
Quantum Fourier Transform

The gate \( R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix} \)

In the notation used before \( R_\theta = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \) where \( \theta \rightarrow \frac{2\pi}{2^k} \)

At the end of this circuit the qubits are swapped (and a factor \( \frac{1}{\sqrt{2}} \) is missing from each line)

**Example: Three qubits**
\(|x_1x_2x_3\rangle \rightarrow \frac{1}{\sqrt{2^3}} \times (|0\rangle + e^{2\pi i[0.x_3]} |1\rangle) \otimes (|0\rangle + e^{2\pi i[0.x_2x_3]} |1\rangle) \otimes (|0\rangle + e^{2\pi i[0.x_1x_2x_3]} |1\rangle) \)
Quantum Fourier Transform

- The number of gates required (including the swaps) is $\Theta(n^2)$
- To implement the classical Fast Fourier Transform $\Theta(n2^n)$ gates are needed, so it appears as a significant improvement for a task that is important in many application
- However, we cannot directly access the amplitudes of a quantum state, so we cannot really extract the classical values of the DFT. To achieve something useful, it needs to be part of some other algorithm (see later!)