Introduction to Quantum Algorithms

Short course report

Alberto Lazari – 2089120

June 2024

Index

1. Computational model

In order to understand quantum algorithms it is necessary to define the tensor product operation between matrices and to present some primitive constructs.

1.1. Tensor product

Let $A \in \mathbb{C}^{m \times n}, B \in \mathbb{C}^{p \times q}, D \in \mathbb{C}^{mp \times nq}$

The tensor product $\otimes$ is defined as a bilinear operator between vector spaces:

$$\otimes : A \times B \to D$$

In the course the $\otimes$ operator always refers to the Kronecker product, a special case of the more generic tensor product.

Kronecker product

$$D := A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ a_{21}B & \dots & a_{2n}B \\ \vdots & & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{pmatrix}$$

1.2. Primitive constructs

Bra-kets Ket $|\psi\rangle$ denotes a column vector in $\mathbb{C}^n$, while bra $\langle\psi|$ denotes a row vector in $(\mathbb{C}^n)^\dagger$, such that $\langle\psi| = |\psi\rangle^\dagger$

Properties:
- $|\psi\rangle|\phi\rangle = |\psi\rangle \otimes |\phi\rangle$
- $\langle\psi|\phi\rangle = \langle\psi, \phi\rangle \Rightarrow$ inner product

Binary strings $\vec{j} \in \{0,1\}^q$ denotes a binary string of q digits, such that $j = \sum_{k=1}^{q} \vec{j}_k \cdot 2^{q-k}$

$|\vec{j}\rangle = |j_1 j_2 \dots j_q\rangle = |j_1\rangle|j_2\rangle \dots |j_q\rangle =$ basis vector of $(\mathbb{C}^2)^{\otimes q}$ with a 1 in position j

1.3. Quantum states

A state in a quantum computer coincides with a single (possibly composite) register of qubits.
A register of q qubits can be represented with a vector $|\psi\rangle$ in $(\mathbb{C}^2)^{\otimes q}$ (which is $\subseteq \mathbb{C}^{2^q}$).

A state $|\psi\rangle$ can be represented as $\sum_{\vec{j} \in \{0,1\}^q} \alpha_j \, |\vec{j}\rangle, \alpha_i \in \mathbb{C}$, which is equivalent to $\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{2^q} \end{pmatrix}$

Basis states / superpositions A q-qubits register $|\psi\rangle$ represents a basis state if $|\psi\rangle = |\vec{j}\rangle$, for some $\vec{j} \in \{0,1\}^q$. Otherwise, it is a superposition.

Product states / entanglements A q-qubits quantum state $|\psi\rangle$ is a product state if $|\psi\rangle = |\psi_1\rangle|\psi_2\rangle \dots |\psi_q\rangle$, where q_i is a 1-qubit state. Otherwise, it is an entangled state.

1.4. Operations on qubits

Operations An operation, or gate, is a unitary matrix $U \in \mathbb{C}^{2^q \times 2^q}$

$\Longrightarrow$ Operations on quantum states are always:
- Linear
- Reversible

Measurement Gate allowing to gather information on the state. Applying a measurement to a q-qubits state $|\psi\rangle = \sum_{\vec{j} \in \{0,1\}^q} \alpha_j |\vec{j}\rangle$ returns the binary string $\vec{k} \in \{0,1\}^q$ with probability $|\alpha_k|^2$

After applying a measurement gate on a state, the original state is not recoverable.

1.4.1. Basic operations

There are some basic gates that are used to create more complex operations:

- X gate: performs a bit flip $\Rightarrow X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- H (Hadamard) gate: returns uniform superposition of basis states (when run in parallel on all qubits):

$$H^{\otimes q}|\vec{0}\rangle = \frac{1}{\sqrt{2^q}} \sum_{\vec{j} \in \{0,1\}^q} |\vec{j}\rangle$$

- CX (Controlled X/NOT) gate: two-qubit gate that can create or destroy entanglement.

  CX_{12} flips target qubit (qubit 2), when control qubit (qubit 1) is $|1\rangle$, leaves qubit 2 as it is otherwise: $\mathrm{CX}_{21}|1\rangle = |01\rangle, \mathrm{CX}_{21}|11\rangle = |10\rangle$
  CX_{21} is the same, but with target qubit 1 and control qubit 2: $\mathrm{CX}_{21}|10\rangle = |10\rangle, \mathrm{CX}_{21}|11\rangle = |01\rangle$

$$\mathrm{CX}_{12} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad \mathrm{CX}_{21} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2. Amplitude amplification

Amplitude amplification is a technique used to distinguish a good state $|\psi_G\rangle$ from a bad state $|\psi_B\rangle$ by flipping the sign of $|\psi_G\rangle$, in order to find a state that has a large overlap with $|\psi_G\rangle$.

2.1. Grover's black-box search algorithm

A special case of amplitude amplification is Grover's black-box search algorithm.

Let $f : \{0,1\}^n \to \{0,1\}$ be an unknown function, that takes as input an n-qubits binary string and outputs 1 for a unique string $\vec{l}$, 0 otherwise.

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} = \vec{l} \\ 0 & \text{otherwise} \end{cases}$$

The good state $|\psi_G\rangle$ searched by the algorithm is the one in which $f(\vec{x})$ returns 1. It does it by repeatedly brute forcing every possible string and amplifying the good state at every step, until the probability of obtaining $\vec{l}$ from a measurement is almost 1.

2.1.1. Complexity

The optimal number of iterations k is around $\frac{\pi}{4}\sqrt{2^n} \Longrightarrow O\left(2^{\frac{n}{2}}\right)$. Compared to a classic algorithm with complexity $O(2^n)$, the quantum version provides a quadratic speedup.

3. Quantum Fourier transform

The quantum Fourier transform algorithm (QFT) aims to compute the DFT of a vector $x \in \mathbb{C}^{2^n}$.

The classic DFT is defined as

$$y_j = \sum_{k=0}^{2^n-1} x_k \cdot e^{\frac{2\pi i j k}{2^n}} \quad \forall j = 0, 1, ..., 2^n - 1$$

The matrix Q_n that implements the n-qubit QFT is

$$(Q_n)_{jk} = \frac{1}{\sqrt{2^n}} \omega_n^{jk} \quad \forall \vec{j}, \vec{k} \in \{0, 1\}^n \quad \omega_n = e^{\frac{2\pi i}{2^n}}$$

3.1.1. Complexity

It is not easy to declare a precise complexity for the algorithm that implements Q_n, but it uses a polynomial amount of resources. By using the *elementary gate complexity* measure the complexity can be declared as polynomial, which-in any case at all-is a huge improvement over the classical DFT algorithm (fast Fourier transform), that has a complexity of $O(n \cdot e^n)$.

This leads to an exponential speedup of the QFT over the classic algorithm.

4. Gradient estimation

Classical computers are able to estimate the gradient of a d-dimension function $f : [0, 1]^d \to \mathbb{R}_+$ at the origin by evaluating f in $O(d)$ time.

Quantum computers are able to do it by evaluating f once, a constant amount of time.

4.1. Jordan's quantum gradient algorithm

Given a linear d-dimension function $f(x) = a^\top x + b, \quad a \in \mathbb{R}^d$ the algorithm returns $a = \nabla f(0)$.

A binary oracle U_f of the function is needed: a matrix such that

$$U_f |\vec{x}\rangle |\vec{y}\rangle = |\vec{x}\rangle |\vec{y} \boxplus \overrightarrow{f(x)}\rangle$$

Where $|\vec{x}\rangle$ is a register composed of d q-qubit registers, representing the arguments $x_1, x_2, ..., x_d$ of $f \implies |\vec{x}\rangle = |\vec{x_1}\rangle |\vec{x_2}\rangle ... |\vec{x_d}\rangle$

In particular, when $|\vec{y}\rangle$ is the initial quantum state

$$U_f |\vec{x}\rangle |\vec{0}\rangle = |\vec{x}\rangle |\overrightarrow{f(x)}\rangle$$

The algorithm returns a, which is $\nabla f(0)$, the gradient of f at the origin, by applying the H gate on $|\vec{x}\rangle$, QFT on $|\vec{y}\rangle$, then U_f on the result. In order to measure a, QFT^{-1} is applied.

4.2. Nonlinear functions

If f is nonlinear, an approximation can be constructed as $g(x) = \nabla f(0)^\top x$. Then Jordan's algorithm can be applied to g, in order to estimate $\nabla f(0)$.

$g(x)$ can be expressed (for m large enough) as

$$\sum_{k=-m}^{m} a_n f(kx)$$

The complexity is $O(m)$.

5. Linear systems solving

Linear systems can be solved by a quantum algorithm.

Given a linear system $Ax = b, \quad A \in \mathbb{C}^{2^n \times 2^n}, b \in \mathbb{C}^{2^n}$
The solution x of the system is encoded in a quantum state $|\psi\rangle$ such that

$$\||\psi\rangle - |\text{amp}(x)\rangle\| \leqslant \varepsilon$$

For some precision parameter $\varepsilon > 0$.

$|\text{amp}(x)\rangle$ is the amplitude encoding of the solution as a quantum state, equivalent to

$$|\text{amp}(x)\rangle := \sum_{\vec{j} \in \{0,1\}^n} \frac{x_j}{\|x\|} |\vec{j}\rangle = \begin{pmatrix} x_1/\|x\| \\ x_2/\|x\| \\ \vdots \\ x_{2^n-1}/\|x\| \end{pmatrix}$$

By encoding the result in a quantum state it cannot be immediately measured. The algorithm is supposed to be used as a stepping stone for other operations, as there is no performance improvement over classical methods if the aim is to measure the solution.

6. Shor's factorization algorithm

An interesting quantum algorithm, that brings great performance improvements and has real-world implications is Shor's integer factorization algorithm.

6.1. Classical integer factorization

Factorizing an integer n is generally intractable on classical computational models, since the fastest known algorithm has a sub-exponential complexity of $O\left(e^{(\log n)^{\frac{1}{3}}(\log \log n)^{\frac{2}{3}}}\right)$.

Many cryptography-related algorithms and protocols exploit this fact to ensure (algorithmically)-unbreakable encryption.

6.2. Quantum factorization

Shor's algorithm provides an exponential speedup over classical approaches: it uses a number of quantum gates of order $O\left((\log n)^2 (\log \log n)(\log \log \log n)\right)$.

The high-level steps of Shor's algorithm are:

1. The problem of finding n's integer factors is reduced to finding the period of a function $f(x) = a^x \mod n$. The objective is finding the smallest positive integer r, such that $a^r \equiv 1 \mod n$, r being the period of f
2. Quantum phase estimation process: applies modular exponentiation and the inverse QFT to a uniform superposition of all possible states (initialized with a H gate)
3. Post processing process: uses the period r to find the factorization of n

6.3. Implications

Implementing this quantum algorithm would lead to severe consequences in a big section of current cryptography and cyber security in general. It could be used to break various cryptography mechanisms, like private/public-key schemes, most notably:

- The RSA algorithm
- Diffie-Hellman key exchange

However, current quantum computers seem to lack a sufficient number of qubits and results are not so stable due to noise and errors, for Shor's algorithm to pose a serious threat in real-world scenarios.