Quantum Computation

Antoine Groudiev

December 23, 2023

One of the most widely used system for secure data transmission is the RSA scheme; its security relies on the assumption that the integer factorization problem is hard. To this day, there is indeed no known efficient algorithm to factor an integer *on a classical computer*.

Nevertheless, this problem can be solved in polynomial time on a quantum computer, using Shor's algorithm. A large enough quantum computer would therefore be able to break the RSA encryption scheme.

We will define the behavior of a quantum computer using two quantum computational models, and delve into quantum complexity theory, the study of complexity classes of problems solved using these quantum models. We will end by describing the Deutsch-Jozsa algorithm, to prove that a quantum computer can be exponentially faster than a deterministic classical computer.

Contents

1 Introduction to Quantum Computers

A classical computer uses the *bit* as a basic unit, a system which can be in exactly one of the two states `0` and `1`. Similarly, a quantum computer uses a basic unit called *qubit*, which can be in a *superposition* of two states. A qubit can be described as a linear combinaison of both states, each coefficient being related to the probability that the qubit is in the given state. This way, a qubit can store more information than a simple bit, by storing information for both states at a time.

1.1 Dirac notation

While representing the state of a classical bit is quite simple, manipulating qubits requires more advanced notation. The state of a qubit is often represented using the Dirac notation, also known as the Bra-ket notation.

A qubit of state Ψ is noted $|\Psi\rangle$, and is in a superposition of the two state $|0\rangle$ and $|1\rangle$. This superposition can be written as:

$$|\Psi\rangle = \alpha|\Psi\rangle + \beta|\Psi\rangle$$

where $\alpha, \beta \in \mathbb{C}$ are called the states amplitudes.

Example (Schrödinger's cat)**.**

$$|\text{Schrödinger's cat}\rangle = \alpha|\text{dead}\rangle + \beta|\text{alive}\rangle$$

The probability to measure the qubit in the state $|0\rangle$ (and respectively $|1\rangle$) is $|\alpha|^2$ (respectively $|\beta|^2$). This interpretation gives us a normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1$$

since the qubit must be mesured in one on the two states.

Example (LFCC exam)**.** Assume that you have a probability $p \in [0, 1]$ of passing the next LFCC exam. Your result at the exam can be represented using one qubit in the following state:

$$|\text{exam}\rangle = \sqrt{p}\,|\text{pass}\rangle + \sqrt{1-p}\,|\text{fail}\rangle$$

We can verify that the normalization condition is respected:

$$(\sqrt{p})^2 + (\sqrt{1-p})^2 = 1$$

1.2 Vector representation

Writing qubits is nice, but we would like to be able to change their state too. The Dirac notation is not the most practical one when the possible states ($|0\rangle$ and $|1\rangle$) are explicit. Instead, we can use a vector representation:

$$|\Psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

This notation allows us to use linear algebra tools to express physical conditions and changes. The normalization condition simply becomes:

$$\||\Psi\rangle\|_2 = 1$$

Furthermore, we can model an action that changes the qubit's state simply by a matrix U. If a qubit of initial state $|\Psi\rangle$ goes through some quantum gate (more on this later) represented by U, the final state $|\Psi'\rangle$ would be:

$$|\Psi'\rangle = U|\Psi\rangle$$

Obviously, we want $|\Psi'\rangle$ to respect the normalization condition for every qubit Ψ. This requires U to be a unitary matrix, that is $UU^\dagger = I_2$, where $U^\dagger$ is the conjugate transpose of U[1].

1.3 The Bloch sphere

When dealing with simple amplitudes, like $\alpha, \beta \in [0, 1]$, the state of a qubit can be simply represented using a 1-dimensionnal line:

Nevertheless, visualizing the state of a qubit - and the transformations of this state as physical transformations are applied to it - can be challenging. A common way to visualize a qubit is the Bloch sphere.

Since $\||\Psi\rangle\|_2 = 1$, we can uniquely[2] write $|\Psi\rangle$ as:

$$|\Psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle$$

where $0 \le \theta \le \pi$ and $0 \le \phi < 2\pi$. This parametrization of $|\Psi\rangle$ corresponds to spherical coordinates, and uniquely define a vector $\vec{a} \in \mathbb{R}^3$:

$$\vec{a} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}$$

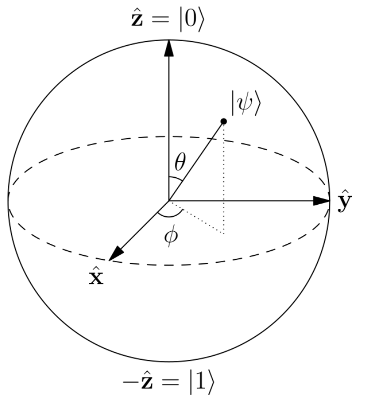

Figure 1: A Bloch sphere

The mesure of the qubit $|\Psi\rangle$ can be seen as the projection of this vector on either $|0\rangle$ or $|1\rangle$, i.e. on $\vec{z}$ or $-\vec{z}$. The closer the point on the sphere is to a pole, the higher the probability to be mesured in this state. When the vector is pointing directly to the top, the probability to be mesured in $|0\rangle$ is 1; when pointing directly down, the probability is 0, and when the vector is orthogonal to $\vec{z}$, the probability is $\frac{1}{2}$.

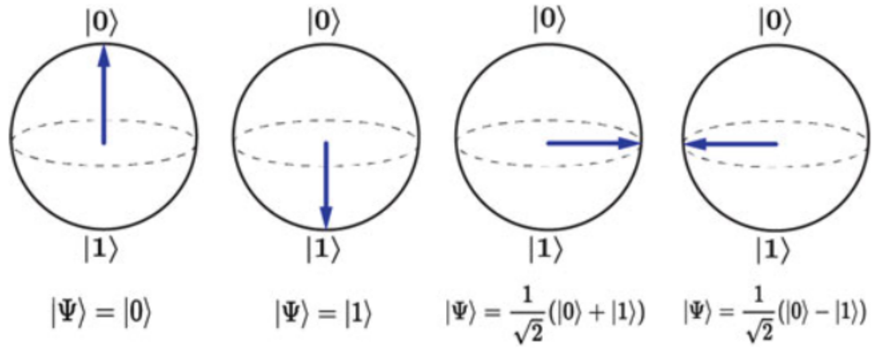
$|0\rangle$
$|1\rangle$
$|\Psi\rangle = |0\rangle$
$|0\rangle$
$|1\rangle$
$|\Psi\rangle = |1\rangle$
$|0\rangle$
$|1\rangle$
$|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
$|0\rangle$
$|1\rangle$
$|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

Figure 2: Examples of vectors on the Bloch sphere for some simple states

[1]The conjugate transpose is usually noted $U^\star$, but I guess that quantum physicists prefer the medieval vibe of †.

[2]Except when $|\Psi\rangle = |0\rangle$ or $|\Psi\rangle = |1\rangle$, since ϕ can take any value.

2 Quantum Computational Models

To study the complexity of different problems with a quantum point of view, we need a model for a quantum computer. Much like classical computers, two approaches coexist: the representation of a quantum computer as a circuit, using quantum logic gates, and an abstract model called quantum Turing machine, by analogy with the classical Turing machine.

2.1 Quantum Circuits

A *quantum circuit* is the counterpart of the classical circuit; from an abstract point of view, it is composed of a multiple qubits, a sequence of *quantum gates*, and special gates called *measurements*, the quantum phenomenon which projects the qubit in one pure state. Each line of the circuit represents one qubit, and the x-axis represents time.

A quantum gate takes into input n qubits, and modifies the probability of each qubit to be in each state. This can be seen visualy: the quantum gate changes the position of the qubit on the surface of the Bloch sphere. Therefore, quantum gates are often represented as unitary matrices of size $2^n \times 2^n$, i.e. elements of $U(2^n, \mathbb{C})$. We will now present a few basic quantum gates.

2.1.1 The X Gate

The X gate is the quantum equivalent of the classical *NOT* gate. It takes one qubit as an input, and permutes its amplitudes. Formally:

$$X : \alpha|0\rangle + \beta|1\rangle \mapsto \beta|0\rangle + \alpha|1\rangle$$

The unitary matrix associated to X is:

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

2.1.2 The Z gate

The Z gate does not have a classical equivalent; its action over one qubit is simply to leave the $|0\rangle$ component untouched, and to multiply the $|1\rangle$ component by -1:

$$Z : \alpha|0\rangle + \beta|1\rangle \mapsto \alpha|0\rangle - \beta|1\rangle$$

Or, written in matrix form:

$$Z \cong \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

2.1.3 Hadamard Gate

The Hadamard gate is a one-qubit gate without classical equivalent; one of its most basic use is to generate superposition on a qubit. The unitary matrix associated to the Hadamard gate, written H, is the following:

$$H \cong \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

An immediate result is that:

$$\begin{cases} H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases}$$

meaning that a qubit starting in a *pure* state ($|0\rangle$ or $|1\rangle$) is transformed into a qubit in a balanced superposition of $|0\rangle$ and $|1\rangle$.

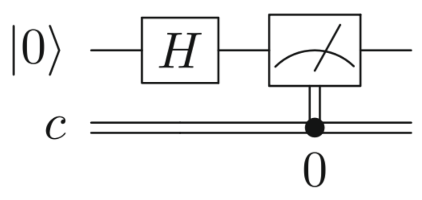

Figure 3: A basic circuit

If we consider the circuit 2.1.3 on the left, the probability to measure $|0\rangle$ is exactly $\frac{1}{2}$. (Note that this is possibly the only method to have true random numbers.) If we were to modify the circuit to add an X gate between the Hadamard gate and the measuring "gate", the outputs probabilities would not change.

Note that like every quantum gate represented by an unitary matrix, and unlike many classical gates, the Hadamard gate is reversible. This might seem counter-intuitive since the previous example suggests that we "lost" the information on the qubit.

2.2 Quantum Finite Automaton

2.2.1 Quantum Languages

Let Σ be an alphabet, and $L \subseteq \Sigma^\star$ a language over the alphabet Σ. Another way to see this language, apart from a subset of $\Sigma^\star$, is using its characteristic function χ_L:

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

By analogy to this, we can define a *quantum language* as a function mapping words to probabilities.

Definition (Quantum language)**.** A quantum language over the alphabet Σ is a function f such as:

$$f : \Sigma^\star \to [0, 1]$$

f is a classical alphabet if $f(\Sigma^\star) \subseteq \{0, 1\}$.

2.2.2 Definition of QFAs and QRLs

Definition (QFA)**.** A *Quantum Finite Automaton* $Q = (H, s_{\text{init}}, H_{\text{accept}}, P_{\text{accept}}, \Sigma, \delta)$ consists of:

- a Hilbert space[3] H of dimension n. (This will generalize the set Q of states of a classical DFA.)
- an initial normalized vector $s_{\text{init}} \in H$ (i.e. $\|s_{\text{init}}\|^2 = 1$). (This is the equivalent of the initial state in a DFA.)
- a subspace $H_{\text{accept}} \subseteq H$, and an operator P_{accept} which projects onto it
- an alphabet Σ
- a function $\delta : \Sigma \to U_n(\mathbb{C})$[4], mapping each letter of $a \in \Sigma$ to an unitary matrix $U_a \in \Sigma$.

We will also write $\delta^\star(w = w_1 \cdots w_{|w|}) = \delta(w_{|w|}) \cdots \delta(w_1) = U_{w_{|w|}} \cdots U_{w_1}$. Finally, the language recognized by Q is the following function:

$$f_Q : w \mapsto \|P_{\text{accept}} \delta^\star(w) s_{\text{init}}\|^2$$

(It is convenient to see the vector and matrix multiplications being computed from right to left.)

[3]Recall that a Hilbert space is an inner product space complete for the distance induced by the inner product
[4]$U_n(\mathbb{C})$ is the set of unitary matrices of size n

Intuitively, we start from the s_{init} vector (or state), and we follow w by multiplying the current vector by the matrix associated to the current letter of the input word. When the word is finished, we project the current vector (which remains of norm 1) on the space H_{accept} using the operator P_{accept}. If $\delta^\star(w)s_{\text{init}} \in H_{\text{accept}}$, the P_{accept} will left $\delta^\star(w)s_{\text{init}}$ untouched, and the output probability will be 1; otherwise, the output probability decreases as the norm projection of $\delta^\star(w)s_{\text{init}}$ decreases.

The end of the process corresponds to a measurement, and $f_Q(w)$ can be interpreted as the probability that this measurement has an acceptable outcome.

Definition (QRL)**.** A *Quantum Regular Language* is a quantum language recognized by a QFA.

2.2.3 A few properties of QRLs

Theorem (Quantum Pumping Lemma)**.** *If f is a QRL, then for any word $w \in \Sigma^\star$ and any $\varepsilon > 0$, there is a $k \in \mathbb{N}^\star$ such that $\|f(uw^kv) - f(uv)\| < \varepsilon$ for any words u, v. Moreover, if f's automaton is n-dimensional, there is a constant c such that $k < (c\varepsilon)^{-n}$.*

2.3 Quantum context-free languages

Definition (QPDA)**.** A *Quantum Push-Down Automata*

Definition (Quantum Grammar[5])**.**

Definition (Quantum Context-Free Grammar)**.**

Definition (Quantum Context-Free Grammar)**.**

2.4 Quantum Turing Machine

Similarly to the classical case, the previous constructions can be generalized to gain expressivity. This leads to the following definition of a Quantum Turing Machine, the equivalent of a classical Turing machine, but using the previous constructions using Hilbert spaces and transition matrices:

Definition (QTM)**.** A *Quantum Turing Machine* (or QTM) $M = (H, \Gamma, b, \Sigma, \delta, q_0, F)$ consists of:

- a Hilbert space H (replacing the classical set of states Q)
- another Hilbert space Γ (replacing the classical set of tape symbols)
- a blank symbol $b \in \Gamma$
- an alphabet of input and output symbols Σ

(Note: the previous definition of a QTM restricts the head movements to $\{L, R\}$, but a generalized version $\{L, N, R\}$ can similarly be introduced. While this does not matter in term of expressivity for a classical TM, there is a difference for QTMs.)

3 Quantum Complexity Theory

Similarly to the study of classical algorithms, we can regroup quantum problems in classes depending on their solvability by a quantum computational model under time and space constraints.

[5]`https://xkcd.com/1090/`

3.1 Quantum complexity classes

3.1.1 The BQP class

An important class of problem is the BQP class; it is the equivalent of the BPP class (Bounded-error, Probabilistic Polynomial time) for classical probabilistic computers.

Definition (BQP)**.** The *Bounded-error Quantum Polynomial time* (BQP) class is the set of decision problems which can be solved in polynomial time by a quantum Turing machine, with an error of at most $\frac{1}{3}$.

The formal definition often involves a maximum error of $\frac{1}{3}$ in one run, but any bound strictly lower than $\frac{1}{2}$ is sufficient, since multiple runs of the algorithm can be chained to reduce the error.

3.1.2 A BQP-complete problem

Definition (APPROX-QCIRCUIT-PROB problem)**.**

3.1.3 Relationship to classical complexity classes

3.2 Relationship with the Church-Turing Thesis

Definition (Church-Turing Thesis)**.** A function on the natural numbers can be calculated by an effective method if and only if it is computable by a Turing machine.

Any quantum algorithm can be simulated either by hand, or on a classical computer, even if the resulting complexity of the computation is changed. Formally, it can be shown that a quantum Turing machine can be simulated on a Universal Turing Machine. Therefore, the existence of quantum computers do not disprove the Church-Turing thesis.

Nevertheless, the Church-Turing thesis do not state anything about the time complexity of the simulation. Historically, this lead to the following extension of the Church-Turing Thesis:

Definition (Extended Church-Turing Thesis)**.** A probabilistic Turing machine can efficiently simulate any realistic model of computation.

Showing that $BPP \subsetneq BQP$ would imply that quantum computers are in certain cases more efficient than classical computers. This would therefore invalidate the extended Church–Turing thesis. A quantum-extended version of the thesis has therefore been proposed.

4 The Deutsch-Jozsa Algorithm

The Deutsch-Jozsa is an important quantum algorithm, not because of any practical use, but because it shows the interest of a quantum computer over a classical computer. The existence of the problem and algorithm proves that there is an *oracle separation* between the EQP and P class: this is a black box problem that can be solved in polynomial time using a quantum computer, but that cannot be solved by a classical computer in polynomial time.

4.1 Problem description

Let's consider a function f operating on n bits or qubits. That is:

$$f : \{0,1\}^n \to \{0,1\}$$

This function is assumed to have a specific property: it is either *constant* or *balanced*:

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

This means that either the function produces only `0`, or it produces only `1`, or it produces exactly half of `0` and half of `1`.

The f function behaves like a black box, or an *oracle*: it is able to output the outcome of $f(x)$ is an single operation for any x. The computer is asked to determine whether the function is constant or balanced.

4.2 Classical solution

A deterministic computer needs in the worst case to mesure more than half the 2^n possible outcomes for f; the best time complexity is therefore exponential. Nevertheless, this problem can be solved efficiently on a probabilistic computer with a high probability.

4.3 Quantum algorithm

Conclusion