SHOR'S ALGORITHM

GIOVANNI CIAMPI

CONTENTS

1 PRELIMINARY ALGORITHMS

1.1 Deutsch's Algorithm

The Deutsch's algorithm is a quantum algorithm for solving the Deutsch's Problem. The Deutsch's problem can be formalized in the following way: suppose we are given a black-box capable of computing a function

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

such that f is either constant or balanced. Formally, for f to be constant means that $f(0) = f(1)$; while for f to be balanced it means that $f(0) \neq f(1)$. Then the problem is to determine if f is constant or balanced by *querying* this black-box.
In a classical setting, in order to find a solution to this problem we would need to query the black-box twice: one query for each input of the domain. It is easy to understand that knowing only one of the input-output pairs of the function would be useless. Surprisingly, if we switch to the quantum setting, it is possible to solve the Deutsch's problem with just one query to our oracle.
From a very general standpoint, the algorithm works by making use of the *quantum superposition*; that is the ability of qubits of being in many states at once. The superposition is exploited in order to give to a *quantum oracle* for f both the inputs 0 and 1 in just one query. A detailed diagram of a quantum circuit implementing Deutsch's algorithm is presented in the following figure:

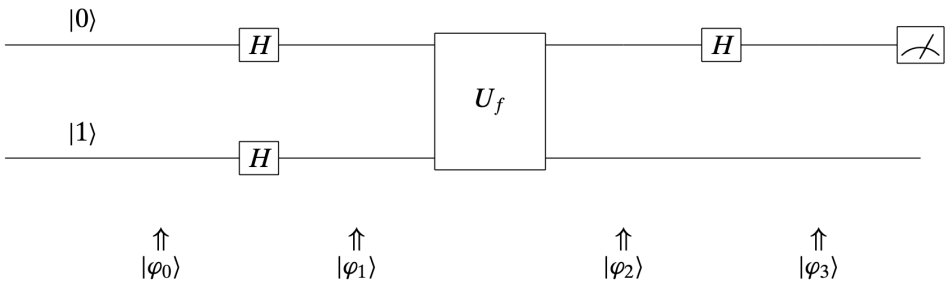

Figure 1: A quantum circuit for Deutsch's algorithm

The U_f block in the figure represents a *controlled* quantum circuit that computes f. Formally, U_f takes two input qubits: the first input, denoted by $|x\rangle$, will be the input of f, while the second input qubit, $|y\rangle$, will control f. So if $y = 1$ f will be computed on $|x\rangle$, otherwise, if $y = 0$, the identity operation will be applied to $|x\rangle$ instead of f. Formally:

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

As we can see from the *Figure 1* the circuit is prepared by setting the input bits $|x, y\rangle$ to the state $|0, 1\rangle$; this state corresponds to the state $|\varphi_0\rangle$ of the

system. As a first operation, two Hadamard gates are applied on both $|x\rangle$ and $|y\rangle$, bringing the state of the system to

$$|\varphi_1\rangle = \Big(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\Big)\Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

At this point the U_f will be applied. It is important to note that by applying the U_f gate to the input $|x\rangle$ $(\frac{|0\rangle - |1\rangle}{\sqrt{2}})$, the result will be

$$(-1)^{f(x)}\,|x\rangle\,\Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

So by applying U_f to the superposition $|\varphi_1\rangle$ we get the superposition

$$|\varphi_2\rangle = \Big(\frac{(-1)^{f(0)}\,|0\rangle + (-1)^{f(1)}\,|1\rangle}{\sqrt{2}}\Big)\Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

Now if we concentrate on the superposition of the first qubit, the fact that f is constant would mean that the value would be $(\pm 1)(\frac{|0\rangle + |1\rangle}{\sqrt{2}})$, while, in the case of a balanced f, the value of the superposition would be $(\pm 1)(\frac{|0\rangle - |1\rangle}{\sqrt{2}})$. It is easy to see that by applying a Hadamard gate to this qubit, it would yield $|0\rangle$ in the case of a constant f and it would yield $|1\rangle$ in the case of a balanced f.
So we have just seen that quantum superposition allows us to solve the Deutsch's problem by calling a quantum version of the f black-box just once; something that is absolutely impossible in the classic context.

1.2 The Deutsch-Josza Algorithm

The Deutsch-Josza algorithm addresses a natural generalization of the Deutsch's problem that occurs if we extend the domain of the function from $\{0,1\}$ to $\{0,1\}^n$. So this time we are given a black-box that computes

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

As in the previous case, f is said to be constant if for every string in $\{0,1\}^n$ the output is the same, and it is said balanced if for exactly half of the input strings it yields one of the two possible outputs, while for the other half of the input strings it yields the opposite output. Formally we have that f is balanced if the following holds

$$\forall x, y \in \{0,1\}^n, f(x) = f(y)$$

while if f is balanced all of the followings are true

$$\exists A, B \mid A, B \subset \{0,1\}^n$$

$$|A| = |B| = 2^{n-1}$$

$$A \cap B = \emptyset$$

$$\forall x, y \in A, f(x) = f(y)$$

$$\forall x, y \in B, f(x) = f(y)$$

$$\forall x \in A, \forall y \in B, f(x) \neq f(y)$$

This simple generalization, in a classical context, to be solved requires highly inefficient algorithms that have running time exponential in n. In fact it is straightforward to note that in order to determine if f is constant or balanced one needs to compute f on *at least* $2^{n-1}+1$ elements. Suppose in fact that f is balanced: by evaluating f on a number of elements that is $\leqslant 2^{n-1}$, we could get the same result each time, and this may lead us to think that f is constant, but in reality we could just be unlucky to select all elements in the A input subset of a balanced function. So in order to have a definite answer we need to evaluate f on the half of the input plus one elements. This has running time that is $O\left(2^n\right)$ given an efficient implementation of the f black-box.
Switching again to the quantum context, we can see that the Deutsch-Jozsa algorithm can effectively solve the problem in constant time. The Deutsch-Jozsa algorithm is implemented in the circuit depicted in the following figure: The structure of the circuit is very similar to the one for Deutsch's

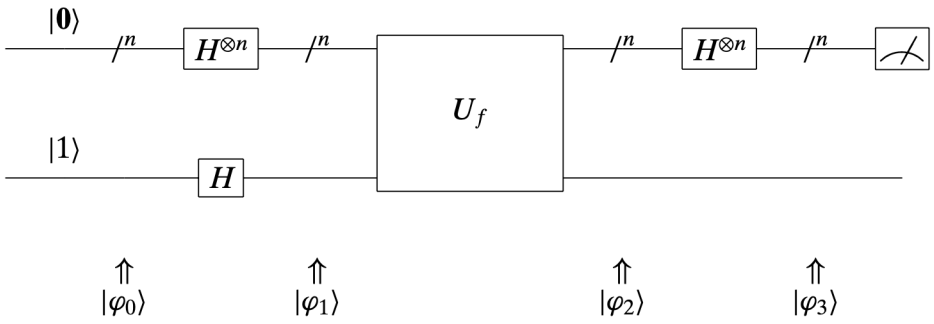

Figure 2: A quantum circuit for Deutsch-Jozsa algorithm

algorithm presented in *Figure 1*, the main difference is that this time the input of f is a vector, so the boldface is used to denote it, and, consequentially, the one-qubit Hadamard gates are extended to be n-qubits Hadamard gates. At the beginning of the circuit, the registers are prepared by setting them to $|\mathbf{0}\rangle|1\rangle$ for the same reasons explained in the previous section. After that, as in the circuit for Deutsch's algorithm the two Hadamard gates are applied. This time, the first register will be left in an equally weighted superposition of all the states in $\{0,1\}^n$. Formally we have that

$$|\varphi_1\rangle = \left(\frac{\sum_{x\in\{0,1\}^n}|x\rangle}{\sqrt{2^n}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

In the same way that with the previous circuit, by applying U_f we get the following state:

$$|\varphi_2\rangle = \left(\frac{\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle}{\sqrt{2^n}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

So the information about $f(x)$ has been encoded in the relative phase of the $|x\rangle$ states in the superposition. Now we can apply the second Hadamard

gate in order to *retrieve* this information. By applying the second Hadamard gate, we get the following state

$$|\varphi_3\rangle = \Big(\frac{\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\sum_{z\in\{0,1\}^n}(-1)^{\langle z,x\rangle}|z\rangle}{2^n}\Big)\Big(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\Big)$$

We get a slightly more compact expression by adding the exponents:

$$|\varphi_3\rangle = \Big(\frac{\sum_{x\in\{0,1\}^n}\sum_{z\in\{0,1\}^n}(-1)^{f(x)}(-1)^{\langle z,x\rangle}|z\rangle}{2^n}\Big)\Big(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\Big)$$

$$= \Big(\frac{\sum_{x\in\{0,1\}^n}\sum_{z\in\{0,1\}^n}(-1)^{f(x)\oplus\langle z,x\rangle}|z\rangle}{2^n}\Big)\Big(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\Big)$$

Now let's analyse the probability that by measuring the top register we will get the state $|0\rangle$. We can easily do that by setting $|z\rangle = |0\rangle$ in the expression above and realising that $\langle z,x\rangle = \langle 0,x\rangle = 0\ \forall x$. This latter point is very important because it implies that the relative phase of $|0\rangle$ is entirely dependent on $f(x)$. So we can rewrite $|\varphi_3\rangle$ as

$$|\varphi_3\rangle = \Big(\frac{\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|0\rangle}{2^n}\Big)\Big(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\Big)$$

It is now straightforward to check that, if f is constant, the expression collapses to either $|0\rangle$ or $(-1)|0\rangle$; while, if f is balanced, exactly half of the $f(x)$ is 0 and the other half is 1, so the factors cancel each other, and we are left with $0|0\rangle$.
At the end of the operations, when we measure the top register, if the result is $|0\rangle$, then it means that the function is constant; if we measure some other value other than $|0\rangle$, it means that the function is balanced. In fact, as showed, the state $|0\rangle$ has amplitude 0 if f is balanced, so it will not be possible to have it as a result of the measurement.

2 SHOR'S ALGORITHM AND THE FACTORING PROBLEM

The factoring problem is the problem of finding (non-trivial) prime factors of a given integer that is not itself a prime number. This problem has been extensively studied for millennia and no classical algorithm has been found to solve it effectively. These reasons allow to use a function of the form

$$f(p_1,p_2) = p_1\cdot p_2$$

where p_1 and p_2 are two large primes as a *one-way* function in cryptography. A one-way function is a function that is *assumed* to be non-invertible in efficient running time. These functions are of crucial importance in computer science, in fact they allow to design effective security protocols that are mathematically sound, though no definitive proof of security is known yet. The fact that mathematicians tried to solve these problems for so long without success, has led to think that the assumption that it is impossible to solve them is reasonable and safe. So in cryptography these functions are fundamental building blocks for security protocols.

The relevance of the problem of factoring for computer science brought enormous attention to the world of quantum computing when Peter Shor came up with an algorithm to solve the factoring problem in polynomial time. This algorithm allows to solve the factoring problem *indirectly*, in fact it makes use of several *reductions* from factoring to other problems, that are strictly inter-related. In particular, Shor's algorithm makes use of the fact that the problem of factoring a given composite integer can be reduced to the problem of finding the order of a certain function, in polynomial time. In the classical world, such order-finding problem would still require exponential running time. Here quantum computing comes in, since it allows us to solve the order-finding problem efficiently.

2.1 From Factoring to Order Finding

Assume we want to factor a given odd integer N. We can safely assume that N is composite since polynomial-time algorithms exist in order to check for primality of a number. Let's pick a number a uniformly at random in $\{2, 3, ..., N-2\}$. We need that such a is *coprime* with N, that is that $GCD(a, N) = 1$. This is not difficult: in fact, once we picked a random a, if it is not coprime with N, it means that $GCD(a, N)$ is a non-trivial factor of N, so we won the lottery and we are done.
Having such an a that is coprime with N, we can define the function

$$f_{a,N}(x) = a^x \bmod N$$

This function $f_{a,N}$ has some important characteristics, in particular, if $GCD(a, N) = 1$, number theory guarantees us that it is periodic, that is

$$\exists r \in \{0, 1, ..N-1\} \,|\, \forall x \in \mathbb{Z},\ \ a^x \equiv a^{r+x} \equiv a^{2r+x} \equiv ... \equiv a^{kr+x} \bmod N,\ \forall k \in \mathbb{Z}$$

If for the selected a, the order is a certain r that is an even number, and we have that $a^{r/2} \neq -1 \bmod N$, then it holds that $a^r \equiv 1 \bmod N$. This implies directly that $a^r - 1 \equiv 0 \bmod N$ and so $N|(a^r - 1)$. Since r is even, we can rewrite the last expression as $N|(a^{r/2} + 1)(a^{r/2} - 1)$. This implies that at least one of $GCD((a^{r/2} + 1), N)$ and $GCD((a^{r/2} - 1), N)$ will yield a non-trivial factor of N, solving the problem.

2.2 Quantum Order-Finding

It is clear now that given an algorithm to find r given a and N we would be able to solve the factoring problem. In the classical context no algorithms are known yet that efficiently solve this problem, while the quantum circuit in *Figure 3* can. The circuit is prepared by setting the input registers to the state

$$|\varphi_0\rangle = |0_m, 0_n\rangle$$

At this point the Hadamard gate sets the state of the first register to an equally weighted superposition of all possible inputs. It is important to notice that the same result would be achieved by using the QFT operator (the Quantum Fourier Transform) on the input $|0_m\rangle$; anyway, the use of

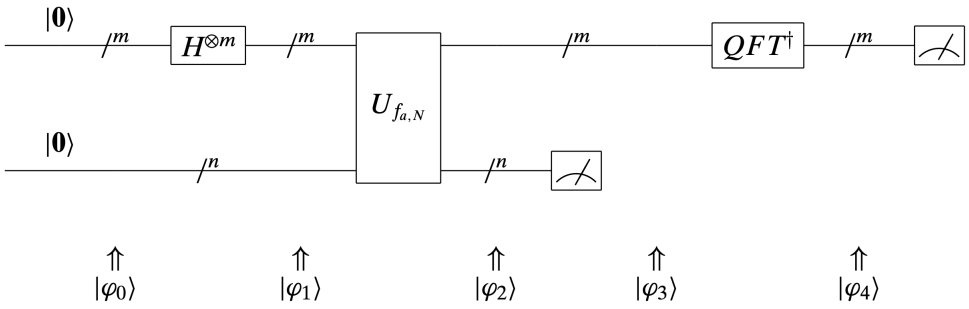

Figure 3: A quantum circuit for Shor's algorithm

the Hadamard gate is much more straightforward; though the awareness of this equivalence makes the use of the QFT^{-1} before the measurement much more clear. After the H is applied, we get the following superposition

$$|\varphi_1\rangle = \frac{\sum_{x\in\{0,1\}^n} |x, 0_n\rangle}{\sqrt{2^m}}$$

After applying $f_{a,N}$ to this state, we get the following result:

$$|\varphi_2\rangle = \frac{\sum_{x\in\{0,1\}^n} |x, f(x)\rangle}{\sqrt{2^m}}$$

that is equivalent to the formulation

$$|\varphi_2\rangle = \frac{\sum_{x\in\{0,1\}^n} |x, a^x \bmod N\rangle}{\sqrt{2^m}}$$

the above expression means that at this point the circuit *already* computed all the values $a^x \bmod N$, in constant time! Now, we need to find a way of retrieving r from $|\varphi_2\rangle$.
It is obvious to note that the values of the function are present several times each in the superposition $|\varphi_2\rangle$. Since we input 2^m values to $f_{a,N}$, and the period of the function is r, we have that a given value $a^{\bar{x}} \bmod N$ is present $\lfloor \frac{2^m}{r} \rfloor$ times.
Now suppose we measure the value of the second register, and we find it to be a certain $\bar{x}$. Then we will have that the first register will be in an equally weighted superposition of all the values x such that the value $f_{a,N}(x)$ will be equal to the measured value. So we will have that

$$|\varphi_3\rangle = \frac{\sum_{a^x \equiv a^{\bar{x}} \bmod N} |x, a^{\bar{x}} \bmod N\rangle}{\lfloor \frac{2^m}{r} \rfloor}$$

that can be also written as

$$|\varphi_3\rangle = \frac{\sum_{j=0}^{\frac{2^m}{r}-1} |t_0 + jr, a^{\bar{x}} \bmod N\rangle}{\lfloor \frac{2^m}{r} \rfloor}$$

where t_0 is an *offset* that indicates the first time that $a^y \equiv a^{\bar{x}} \bmod N$ for an $y \in \{1, 2, .., r-1\}$.
At this point, in order to retrieve r, the QFT^{-1} is applied to the first register. This last operation will conduct to the final state

$$|\varphi_4\rangle = |r, \bar{x}\rangle$$

2.3 The Implementation of $f_{a,N}$

The implementation of the circuit $U_{f_{a,N}}$ is not trivial, as the implementation of quantum circuits require that the gates satisfy some constraints. A first step towards the implementation of $U_{f_{a,N}}$ is a split of the input. Let's start by writing x in binary. We have

$$x = x_{n-1}x_{n-2}\cdots x_1x_0$$

where x can be converted in decimal through the following expression

$$x = 2^{n-1}x_{n-1} + 2^{n-2}x_{n-2} + \cdots + 2x_1 + x_0$$

We can exploit this definition in order to write $f_{a,N}$ as

$$f_{a,N}(x) = a^{2^{n-1}x_{n-1}+2^{n-2}x_{n-2}+\cdots+2x_1+x_0} \bmod N$$

and, equivalently, to

$$f_{a,N}(x) = a^{2^{n-1}x_{n-1}} \times a^{2^{n-2}x_{n-2}} \times \cdots \times a^{2x_1} \times a^{x_0} \bmod N$$

This last expression lets us give an inductive definition of $f_{a,N}$ that is

$$f_{a,N}(x_j) = a^{2^j x_j} \times f_{a,N}(x_{j-1})$$

where

$$f_{a,N}(x) = a^{2^n x_n}$$

and the base case is

$$f_{a,N}(x_0) = a_0^x \bmod N$$

In short, we can get a full $U_{f_{a,N}}$ circuit by putting together a series of $U_{a^{2^j}}$ operators. As long as $\mathrm{GCD}(a, N) = 1$, we are guaranteed that all these operators are unitary and reversible.

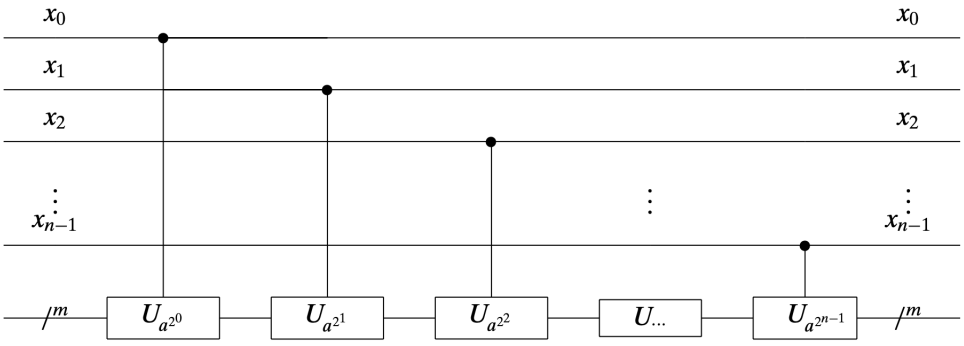

Figure 4: The $U_{f_{a,N}}$ circuit

REFERENCES

[1] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. Oxford University Press, Inc., USA, 2007.

[2] Noson S. Yanofsky and Mirco A. Mannucci. *Quantum Computing for Computer Scientists*. Cambridge University Press, USA, 1 edition, 2008.