Grover's Algorithm

Gerardo García Moreno

February 26, 2018

Imagine you are given a telephone list where each person's telephone number is given, suppose it is ordered alphabetically and you are asked to search for someone's number. It will be easy because it is alphabetically ordered and you will probably find it fast even if there are thousands of items in the list. However, suppose the problem is the opposite: you are given the telephone list which is ordered alphabetically, a telephone number and you are asked to look for the name of the person associated to that telephone number. Obviously, this problem turns out to be much more difficult and the unique technique that you can use is to look for one by one, hoping you find it soon. More formally, we can say that the problem takes $\mathcal{O}(N)$ operations to be solved where N is the number of elements that are in the list. However, all we have discussed before holds on in the classical case. If we introduce the laws of quantum mechanics it can be showed that the problem can be solved in $\mathcal{O}\left(\sqrt{N}\right)$ operations applying the quantum search algorithm known as Grover's algorithm that we will explore. This shows again the supremacy of the quantum computer, it allows us to speed up a problem that classically becomes hard to treat.

1 Introduction

Let's suppose we have to search through a space of N elements (0...N-1) and, for convenience, we will asume that $N = 2^n$ in order to use n bits to save the information. Our results can be easily generalized to the case where this condition is not verified. Suppose we are looking for M elements (that is, the search problem has M solutions) with $1 \leq M \leq N$. We will consider that we have a function $f(x)$ where $x \in \{0...N-1\}$ such that $f(x) = 0$ if x is not a solution to the problem and $f(x) = 1$ if x is a solution to the problem.

Let us consider that quantum mechanically, we have a qubit $|q\rangle$ and an operator O such that:

$$O|x\rangle|q\rangle = |x\rangle|q \oplus f(x)\rangle, \tag{1}$$

where $\oplus$ denotes addition modulo 2. We call $|q\rangle$ the oracle qutbit and O the oracle operator because they mark the solutions to the problem; that is, given x is a solution to the problem, if the qubit $|q\rangle$ is on the state $|0\rangle$ or $|1\rangle$ it changes its value after the application of the oracle operator. An important thing that must be noticed is the fact that if $|q\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ we can write the action of the operator as:

$$O|x\rangle|q\rangle = (-1)^{f(x)}|x\rangle|q\rangle. \tag{2}$$

With this in mind, if we fixed the state of the oracle qubit as that written above, we can forget about it and just say that the oracle operator marks the solutions of the search problem by changing the sign of the inputs.

2 Quantum Search Proccedure

Let's detail how the quantum search works in this section. The algorithm begins with the computer in the state $|0\rangle^{\otimes n}$ and then we apply the Hadamard gate to each qubit ($H^{\otimes n}$). That leaves the system in the state $|\psi\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$. Once we have that state, we must perform R operations of the G operator where R is an integer that will be specified and the G operator consists of four suboperations:

1. We apply the O operator.
2. We apply again the Hadamard gate to each qubit $H^{\otimes n}$.
3. We must perform the following transformation: $|x\rangle \rightarrow -(-1)^{\delta_{x0}}|x\rangle$. This is just applying the operator $2|0\rangle\langle 0| - I$ where I is the identity operator.
4. We apply one more time the Hadamard gate to each qubit $H^{\otimes n}$.

Summing up, the G operator can be expressed as:

$$G = H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n}O = (2|\psi\rangle\langle\psi| - I)O \tag{3}$$

In a pictorical way, it can be expressed as the following quantum circuit:

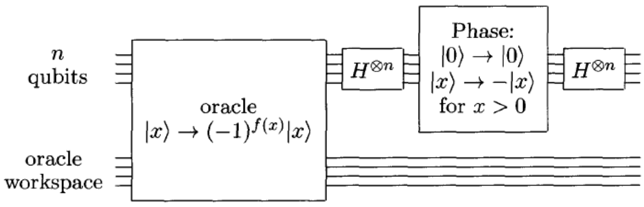
n
qubits
oracle
$|x\rangle \rightarrow (-1)^{f(x)}|x\rangle$
oracle
workspace
$H^{\otimes n}$
Phase:
$|0\rangle \rightarrow |0\rangle$
$|x\rangle \rightarrow -|x\rangle$
for $x > 0$
$H^{\otimes n}$

Figure 1: Quantum circuit showing the action of G.

With this, we must just specify R in order to tell how the algorithm works. However, we will determine R by first introducing a geometrical interpretation of Grover's Algorithm in the next section.

3 Geometrical Interpretation of Grover's Algorithm

Let's first define the following vectors:

$$|\alpha\rangle := \frac{1}{\sqrt{N-M}}\sum_{x''}|x''\rangle, \tag{4}$$

$$|\beta\rangle := \frac{1}{\sqrt{M}}\sum_{x'}|x'\rangle, \tag{5}$$

where x' denotes the sum over the solutions to the problem and x" denotes the sum over the elements which are not solutions. We can see that $|\psi\rangle = a|\alpha\rangle + b|\beta\rangle$ with $a = \sqrt{\frac{N-M}{N}}$ and $b = \sqrt{\frac{M}{N}}$.

Thus, the vector $|\psi\rangle$ is spanned in the subspace spanned by $\{|\alpha\rangle, |\beta\rangle\}$. Let's now see what is the action of the G operator onto the subspace spanned by $|\alpha\rangle$ and $|\beta\rangle$. First, we can notice that $O|\psi\rangle = a|\alpha\rangle - b|\beta\rangle$; that is, the oracle operator performs a reflexion of the state vector about the vector $|\alpha\rangle$ in the subspace spanned by $|\alpha\rangle$ and $|\beta\rangle$. Second, we can notice that the operator $2|\psi\rangle\langle\psi| - I$ performs another reflexion about the vector $|\psi\rangle$ in the same subspace. From classical results of linear algebra, we know that the product of two reflexions is just a rotation; that is, the G operator simply performs a rotation of the vector in the space spanned by the vectors $|\alpha\rangle$ and $|\beta\rangle$. Because of that, applying the G operator R times, will leave the result in the space spanned by $|\alpha\rangle$ and $|\beta\rangle$.

If we define $\cos\left(\frac{\theta}{2}\right) = \sqrt{\frac{N-M}{N}}$, the state $|\psi\rangle$ can be expressed as $|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|\alpha\rangle + \sin\left(\frac{\theta}{2}\right)|\beta\rangle$ and the application of the G operator has the following action on the state: $G|\psi\rangle = \cos\left(\frac{3\theta}{2}\right)|\alpha\rangle + \sin\left(\frac{3\theta}{2}\right)|\beta\rangle$. Moreover, if we make the identification $|\alpha\rangle \longrightarrow \begin{pmatrix}1\\0\end{pmatrix}$ and $|\beta\rangle \longrightarrow \begin{pmatrix}0\\1\end{pmatrix}$ the operator G can be represented as:

$$\begin{pmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{pmatrix},$$

with $\sin(\theta) = \frac{2\sqrt{M(N-M)}}{N}$ and let's assume $M \leq N/2$ for simplicity. We can visualize it with the following diagram:

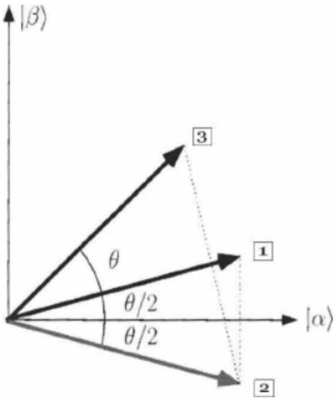

Figure 2: Geometrical interpretation of Grover's Algorithm.

With this image, it is clear that what we are doing is rotating $|\psi\rangle$ near $|\beta\rangle$ by applying the G operator. Let's see how many times we have to apply G to do it. We want to perform a rotation of $\Delta = \frac{\pi}{2} - \frac{\theta}{2} \longrightarrow \cos(\Delta) = \sin\left(\frac{\theta}{2}\right) = \sqrt{\frac{M}{N}}$. If each application of G gives a rotation of θ we can see that we must apply it R times where :

$$R = CI\left(\frac{\arccos\left(\sqrt{M/N}\right)}{\theta}\right), \tag{6}$$

where CI represents the closest integer. After performing R rotations, a measurement in the computational basis gives as a solution with probability at least $\frac{1}{2}$. If $M << N$, we can give an approximate bound for the probability of success. First, notice that: $\theta \approx \sin(\theta) \approx 2\sqrt{\frac{M}{N}}$. With this we can see that the error will be less than $\frac{\theta}{2} = \sqrt{\frac{M}{N}}$ with a probability of error less than $\frac{M}{N}$ approxximately.

Also, we can give a bound for R. First, we must notice that $R \leq \lceil \frac{\pi}{2\theta} \rceil$. If we give a lower bound to θ, we will give automatically an upper bound to R. We must notice that:

$$\frac{\theta}{2} \geq \sin\left(\frac{\theta}{2}\right) = \sqrt{\frac{M}{N}}, \tag{7}$$

so $R \leq \lceil \frac{\pi}{4}\sqrt{\frac{N}{M}} \rceil$. With this, we have proved that we can solve the search problem in $\mathcal{O}\left(\sqrt{N}\right)$ operations. That is, a quantum computer will solve the problem in a more efficient way than a classical one. Also, we have assumed we know the number of solutions M before applying the algorithm although it will not be the general case. We can combine this algorithm with the phase estimation procedure (which is a quantum proccedure involving the quantum fourier transform) applied to G in order to obtain M and then have a complete search algorithm.

A Appendix:Hadamard Gate

The Hadamard Gate denoted as H is a unitary operator that acts on a two-level system and whose action in the computational basis $\{|0\rangle, |1\rangle\}$ can be expressed as:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

If we introduce the identification $|0\rangle \longrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle \longrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, the operator is represented as the following matrix: $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

B Appendix: Computational Complexity

In this appendix, we will try to clarify briefly how we quantify the complexity of a given algorithm. It is clear that there are many ways in which we can quantify how difficult it is (from a computational point of view) to solve a problem. For example, we could look up at the amount of internal space a certain computation may occupy or how much RAM memory it will consume in order to accomplish a certain task. However, for our purposes we will restrict ourselves to the analysis of time limitations of an algorithm.

In order to formalize our definition of "time", we will use the extended asymptotic notation for steps. Suppose, for example, that a certain program would take $f(n) = 24n + \log n + \sqrt{n} + 17$ steps, where n is the length, in bits, of our input. Then we may say that our program is $O(n^2)$ or $o(n)$ because, for sufficiently large n, the functions n^2 and n act as upper and lower bounds for $f(n)$ up to a real unimportant constant factor, respectively. However, in general we will use a similar notation $\mathcal{O}(n)$ to denote that, for sufficiently large n, our function behaves *like* n up to an unimportant constant factor; that is, if two constants a, b exist such that $an < f(n) < bn$ when $n \to \infty$.

It is important to notice that the complexity of a problem is very different from the complexity of a given algorithm. As different algorithms may be used to solve the same problem, some of them will be more efficient than others, and therefore have different complexities, but the problem will always be as complex as the most efficient of the algorithms we could design to solve it. That is, if the best algorithm we can think of to solve a particular problem is $\mathcal{O}(n \log n)$, then the problem itself will be classified $\mathcal{O}(n \log n)$.

What makes an algorithm simple or complex? Traditionally, we would classify algorithms in the following way. If the complexity of an algorithm is at most polynomial, that is, if the algorithm is at most $\mathcal{O}(p(n))$ where $p(n)$ is a polynomial on n, we call it *easy*, *tractable* or *feasible.* If, on the other hand, its complexity is greater than that (we call it exponential complexity, abusing of the term, since we may have, for example, an $\mathcal{O}\big(n^{\log n}\big)$ algorithm, which grows faster than any polynomial but slower than any exponential), we say the algorithm is *hard*, *intractable* or *infeasible.* This classification is sometimes rather coarse, as for example we may find an $\mathcal{O}\big(2^{n/1000}\big)$ algorithm more useful than an $\mathcal{O}\big(n^{2000}\big)$ algorithm because only at incredibly huge n is the exponential greater than the polynomial, but for our purposes it would do just fine since we will not consider this kind of subtelties.

References

[1] Michael A. Nielsen, Isaac L. Chuang *Quantum Computation and Quantum Information* (Cambridge University Press, 2000)

[2] Mikio Nakahara, Tetsuo Ohmi *Quantum Computating: From Linear Algebra to Physical Realization* (CRC Press, 2008)

[3] Dorit Aharonov *Quantum Computation* (Annual Reviews of Computational Physics, ed. Dietrich Stauffer, World Scientific, vol VI, 1998)