Grover's algorithm

Daan Sprenkels

Radboud University Nijmegen, The Netherlands
dsprenkels@science.ru.nl

1 Introduction

The first notion of *quantum computing* was put forward by the great 20th century physicist Richard Feynman [4]. The underlying problem was that for approximating quantum wave functions with N variables, you need $O(N^N)$ operations. This is not easily computable with classical computers without making huge assumptions about the quantum system (which often just produces bad solutions). So instead of using a classical computer, which will *always* be in some state s, we use a quantum computer which will be in a superposition[1] of multiple states (a quantum state) at the same time. This way we can start in a combination of states, and after executing a program, we get another combination of states. This is what people mean when they say that a quantum computer does all computations at the same time. But this is not really true, because in the end we will have to measure the state and we will find the state to be in only one of them.

But that is not the full story. In the classical world we were limited to tranformations from *one* state to *one* other state. Now with quantum states we can make programs that transform from a combination of states into another combination of states. Later on we will see why this is so powerful when we address the concept of amplitude amplification.

This ability is a problem for modern cryptography. Although counter-intuitive, some quantum algorithms are an order of magnitude faster than their classical counterparts because of this. The most famous example is Shor's algorithm [9], which allows for easy factoring of large prime num-

[1] Physicists like the word *superposition* a lot. Sometimes it is explained as a system being in any of a multiple of states without us knowing which one it is. The accepted interpretation is however that a system is actually a linear combination of multiple states, but collapses to *one* of them when a measurement is performed. The question of how this is possible is a problem that has existed since the beginning of quantum mechanics and is called the "measurement problem". Most physicists don't even bother getting their heads around this, so neither need you [5].

bers (breaking RSA and all discrete logarithm based cryptosystems)[2]. These notes discuss only the algorithm of Grover [6], which reduces the complexity of all symmetric ciphers and all hash functions from $O(N)$ to $O(\sqrt{N})$, effectively halving their security.

2 Quantum computing principles

2.1 Qubits

Where in classical computers we have a bit, which can be in any of two states, in quantum computing we have *qubits*. One qubit is a vector in $\mathbb{C}^2$; some combination of zero and one. In the case of one qubit, we define our vector space with the two orthogonal basis vectors $|0\rangle$ and $|1\rangle$. Here we use the *bra-ket* notation to denote vectors in our vector space. Any qubit $|\psi\rangle$ can be in any linear combination of basis vectors, so mathematically we can write any qubit as $|\psi\rangle = a_0 |0\rangle + a_1 |1\rangle$. Here we call a_0 and a_1 the amplitudes of $|0\rangle$ and $|1\rangle$ respectively.

In this vector space the inner product between $|\psi_a\rangle = a_0 |0\rangle + a_1 |1\rangle$ and $|\psi_b\rangle = b_0 |0\rangle + b_1 |1\rangle$ is defined as:

$$\langle\psi_a|\psi_b\rangle = \overline{a_0} b_0 \langle 0|0\rangle + \overline{a_1} b_1 \langle 1|1\rangle$$

When a qubit $|\psi\rangle$ is measured, the probability that we find it in a basis state $|x\rangle$ is equal to $\langle\psi|x\rangle$. Because this inner product describes a probability, we know that $\langle\psi|\psi\rangle = 1$ for all vectors $|\psi\rangle$. With this knowledge we can simplify the previous equation to:

$$\langle\psi_a|\psi_b\rangle = \overline{a_0} b_0 + \overline{a_1} b_1$$

Multiple qubits can be combined to make systems over bases with more than two elements. This makes it possible to have states like $|101\rangle$. Analogous to regular bits, n qubits can store combinations of 2^n different states.

2.2 The Quantum Circuit Model

These qubits themselves can do little without the means to manipulate them using quantum operations. A common model to describe operations on qubits is the *Quantum Circuit Model* [9, 8]. In this model, a quantum

[2] If you would like to know more about Shor's algorithm I strongly recommend reading "Introduction to Quantum Algorithms" by Shor himself (reference [9]).

program is described as a circuit of quantum gates which operate on a one or more qubits. As long as a gate describes a hermitian operator on the full quantum state, it is a valid operator. As a consequence, any classical operation can also be done by a quantum computer as long as we have enough qubits available. So in theory we can implement hash computations as parts of quantum algorithms.

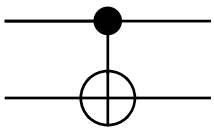

Fig. 1. Example of a controlled NOT (CNOT) gate [11]. Depending on the value of the upper wire, the value of the lower wire will be flipped.

2.3 Simulating the quantum gate model

Physical quantum computers do not yet exist, so at the moment we wil have to do with simulating them using classical methods. The conventional way to do this is by using column vectors for the qubits and matrix multiplications for the operations. An example of the matrix representation of the CNOT gate (on a two-qubit system) from figure 1 is[3]:

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

If we need to apply multiple different quantum operation—first $\mathbf{A}$, then $\mathbf{B}$—on a quantum state we can simulate this by applying them after each other, i.e. $|\psi'\rangle = \mathbf{BA}\,|\psi\rangle$.

[3] I do not expect you to see this immediately. As a matter of fact, it is not really important that you do. You may just believe me on my word that this matrix indeed represents the CNOT gate. But if you feel that you do not understand why this could work you may want to grab a sheet of paper and apply this matrix to a couple of example quantum state vectors like

$|10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ or $|11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ or the combined state $\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$.

3 Grover's algorithm

Grover's algorithm is mostly known for its usage on hash functions. A more general description is that the algorithm is good for finding a specific element in an unordered list. In the case of finding a preimage for some hash h this element is the element x for which $\text{Hash}(x) = h$. In the general case, when searching for the target value t, we define an oracle function f for which $f(x) = 1$ if $x = t$ and $f(x) = 0$ for all other values of x.

In essence, the algorithm works by amplifying the amplitude of the state for which $f(x) = 1$, so that in the end, when we measure what state ψ is in, the probability of measuring t state is larger than the probability of measuring another x.

3.1 Initialization

First, to get all the amplitudes to the same level the algorithm applies the Hadamard operator H (also called the coin flip operator) on each qubit. The result is a quantum state in which the amplitude of all basis states is equal to $\frac{1}{\sqrt{N}}$.

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

3.2 Iteration

The iteration part starts by inverting the amplitude of *only* the t component. To simulate this we apply the operator Z_t:

$$Z_t \left|x\right\rangle = (-1)^{f(x)} \left|x\right\rangle$$

on all separate qubits in the system.

Note that we do not need to do a measurement to check if x is the to-be-found value (which would collapse our state). We only have to keep in mind that the oracle function is implemented as part of the quantum algorithm.

The second part of the iteration is to apply what Grover calls the "diffusion transform" D. This transformation performs *inversion about the average* and it performs the following operation on all separate state components $\left|x\right\rangle$ in $\left|\psi\right\rangle$:

$$D \left|x\right\rangle = (2m - 1) \left|x\right\rangle$$

where m is the current mean amplitude of all the states[4].

These two steps are iterated until the amplitude of $|t\rangle$ is large enough so that the probability that we measure it is $O(1)$. We measure to get the output. Because there's some probability that the output was incorrect, the outcome is checked with a classical algorithm. If the algorithm produced a bad result, we try again until it gives back the correct output state.

3.3 Complexity

In a classical search algorithm, you expect to find t after $\frac{N}{2}$ iterations on average, which is $O(N)$. This is (of course) not very efficient if N is large.

Grover is different. Each iteration of Grover's algorithm amplifies the amplitude of the t state with $O(\frac{1}{\sqrt{N}})$. This means we need to do the iteration $O(\sqrt{N})$ times to crank the amplitude up to the point where the probability of measuring $|t\rangle$ is $O(1)$. This is a major speedup relative to the classical algorithm.

3.4 Example iteration

We will do an example iteration of Grover's algorithm on a quantum state of two bits. We will take the target state to be $|t\rangle = |10\rangle$ (the third basis vector).

The matrices for Z_t and D are, for a state with two qubits:

$$\mathbf{Z_t} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{D} = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$$

Starting from the initialized state $\psi = \frac{1}{2}(1,1,1,1)^T$ we first apply $\mathbf{Z_t}$:

$$\mathbf{Z_t} \cdot \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

[4] A more formal definition is the one that Grover himself describes, which is that $D = H^{\otimes N} Z_t H^{\otimes N}$, with $H^{\otimes N}$ defined by $H^{\otimes N} = H \otimes H^{\otimes (N-1)}$. Proving why this definition is equivalent to its geometric description is outside of the scope of these notes. For this I (again) refer you to Shor's notes on Grover's algorithm [6] (page 15).

We then apply the diffusion matrix $\mathbf{D}$:

$$\psi' = \mathbf{D} \cdot \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Which—when a measurement is performed—will of course produce $|10\rangle$ as output.

Note that $N = 4$. So for the classical algorithm we expect to do two iterations on average before we find t. For the quantum algorithm, we needed only one iteration to find t.

4 Discussion and conclusion

The Grover algorithm is a clear symptom of the advantage that quantum computers sometimes have over classical computers. Grover defeats the strength of symmetric ciphers and cryptographic hash functions by a factor of two. Futhermore, Grover's algorithm can be used to find collisions in hash functions in $O(\sqrt[3]{N})$ time [2].

When Grover is concerned, these attacks can easily be mitigated against by doubling the security parameter of our cryptographic primitives. This will make all protocols slightly slower and they will use a bit more bandwith, but never is the security of really compromised.

With asymmetric crypto, we actually *do* have a problem. Most asymmetric crypto uses the hardness of the discrete log or prime factorization problems for their security. Shor's algorithm breaks all cryptosystems based on any of these problems, including RSA and (elliptic curve) Diffie-Hellman. The algorithm runs in polynomial time, so increasing the amount of bits is not a solution.

Lucky for us, no sophisticated quantum computers exist yet. This is because physically entangling particles with quantum states is very hard. Currently, the most popular method of building a quantum computer is an ion trap quantum computer [10], which works by using lasers to manipulate ions in a magnetic field. Other implementations include using nuclear magnetic resonance [3] or the polarization of light [7]. Thusfar, no research group has come close to building a quantum computer with enough bits to break any real-world crypto.

Besides, new "post-quantum" cryptographic algorithms are currently being developed. These new algorithms are based on other hard math

problems, which even quantum computers cannot efficiently solve. Most of these algorithms use very large keys, which makes them impractical, but reasonable candidates have been proposed and even implemented in mainstream software [1]. It is clear however that we will eventually have to replace *all* public key crypto by post-quantum algorithms, and that this must happen rather sooner than later.

References

1. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange-A new hope. IACR Cryptology ePrint Archive 2015, 1092 (2015)
2. Brassard, G., Hoyer, P., Tapp, A.: Quantum algorithm for the collision problem. arXiv preprint quant-ph/9705002 (1997)
3. Cory, D.G., Fahmy, A.F., Havel, T.F.: Ensemble quantum computing by nmr spectroscopy. Proceedings of the National Academy of Sciences 94(5), 1634–1639 (1997)
4. Feynman, R.P.: Simulating physics with computers. International Journal of Theoretical Physics 21(6), 467–488 (1982)
5. Griffiths, D.: Introduction to Quantum Mechanics. Pearson international edition, Pearson Prentice Hall (2005)
6. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219. ACM (1996)
7. Lanyon, B., Barbieri, M., Almeida, M., White, A.: Experimental quantum computing without entanglement. Physical review letters 101(20), 200501 (2008)
8. Nielsen, M.A., Chuang, I.L.: Programmable quantum gate arrays. Phys. Rev. Lett. 79, 321–324 (Jul 1997)
9. Shor, P.W.: Introduction to quantum algorithms. In: Proceedings of Symposia in Applied Mathematics. vol. 58, pp. 143–160 (2002)
10. Strubell, E.: An introduction to quantum algorithms (2011)
11. Williams, T.: CNOT gate (March 2011), licenced under CC BY-SA 3.0