

Quantum Search + quantum Zeno effect

0.0.1 NP-Completeness

Most interesting computational search problems share an important feature: given a proposed solution, it is easy to verify whether its correctness. i.e. it is possible to check in polynomial time whether a proposed solution is indeed a solution to the problem. Unfortunately there are exponentially many potential solutions to choose from, and this is the combinatorial explosion that makes the design of efficient algorithms so challenging.

Let us consider an example — satisfiability. Here we are given a boolean function $f(x_1, \dots, x_n)$ mapping $\{0, 1\}^n$ to $\{0, 1\}$. The challenge is to find an input (a_1, \dots, a_n) such that $f(a_1, \dots, a_n) = 1$. Such an input is called a satisfying assignment. In the satisfiability problem the boolean function is specified very explicitly via a formula (or sometimes even a circuit). For example, in the case 3SAT (or 3-satisfiability), the function $f(x_1, \dots, x_n) = c_1 \wedge \dots \wedge c_m$, where each of the c_i 's (called clauses) are simple functions of three of the x_j 's. More precisely each clause is of the form $(u \vee v \vee w)$, where u, v, w (called literals) each stand for either a variable x_j or its complement $(1 - x_j)$. The clause evaluates to 1 if any of u, v or w is 1.

3SAT is a prototypical example of an NP-complete problem. What this term means is that 3SAT is in the class NP of problems where a proposed solution can be efficiently checked — in this case by checking that for each clause there is a literal that evaluates to 1. The challenge, of course, is finding a satisfying assignment among the 2^n possible assignments of values to x_1, \dots, x_n . Saying that 3SAT is NP-complete means that it is one of the hardest problems in NP, in the sense that if you can solve 3SAT efficiently (i.e. in polynomial time) then you can solve any problem in NP efficiently. There are now thousands of useful computational problems that are known to be NP-complete. Coping with them in practice is one of the big challenges in the field of Algorithms.

The question we will focus on today is whether the exponential power of quantum algorithms can help solve NP-complete problems efficiently. The most naive hope is this: start with an n -qubit register in the state $|0^n\rangle$. Apply the Hadamard transform to get the superposition $\sum_{x \in \{0,1\}^n} |x\rangle$. Now compute the function f to get the superposition $\sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$. The quantum computer has now "in parallel" computed $f(x)$ for every value of x and "should somehow" be able to find an assignment such that $f(a) = 1$. The problem with this naive hope is that we have no interference between the different values of x , and if we were to measure the quantum computation would be no different than it would have been if we had just randomly chosen x . So is there any quantum algorithm that can do better? How much better?

0.0.2 Unstructured Search

To answer these questions we will abstract our problem as follows: we represent f by a table of $N = 2^n$ entries (one for each input) where exactly one of the entries is 1, and all the rest are 0. The task is to find index of the unique entry that is 1. Ideally we want to solve the problem in polynomial time. i.e. $O(\text{poly}(n))$. The quantum algorithm is allowed to query the table in superposition. i.e. it is allowed to create an arbitrary superposition $\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$, and query the table to get the answer $\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle |f(x)\rangle$.

Unfortunately we can show that any quantum algorithm to solve this problem must make at least $\sqrt{N} = 2^{n/2}$ queries to the table. The proof of this fact is beyond the scope of this course. What we will see is an algorithm, due to Grover, that matches this bound. It give a quantum algorithm for solving this problem in $O(\sqrt{N})$ steps.

Here's the problem: You are given a boolean function $f : \{1, \dots, N\} \rightarrow \{0, 1\}$, and are promised that for exactly one $a \in \{1, \dots, N\}$, $f(a) = 1$. Think of this as a table of size N , where exactly one element has value 1, and all the others are 0. Since we assume f can be computed classically in polynomial time, we can also compute it in superposition:

$$\sum_x \alpha_x |x\rangle |0\rangle \rightarrow \sum_x \alpha_x |x\rangle |f(x)\rangle$$

As we saw before, we can use circuit for f to put information about $f(x)$ in the phase by effecting the transformation:

$$\sum_x \alpha_x |x\rangle \rightarrow \sum_x \alpha_x (-1)^{f(x)} |x\rangle$$

0.1 Grover's Algorithm

The quantum search algorithm starts with the superposition $|\psi_0\rangle = \sum_x \frac{1}{\sqrt{N}} |x\rangle$. For \sqrt{N} iterations it applies a sequence of two operations. Each such iteration increases the amplitude of $|a\rangle$ by at least $\frac{1}{2\sqrt{N}}$. It follows that at the end of the algorithm the amplitude of $|a\rangle$ is a constant, and therefore measuring the register yields a with constant probability.

The first operation is a phase reflection of $|a\rangle$: i.e. it maps the superposition $|\psi\rangle = \sum_x \alpha_x |x\rangle$ to $|\psi'\rangle = \sum_{x \neq a} \alpha_x |x\rangle - \alpha_a |a\rangle$. Exercise: show how to use the quantum circuit for computing f to implement a phase reflection of $|a\rangle$.

The second operation is a reflection about the mean. Before describing this operation let us imagine that we wanted to perform a reflection about $|0\rangle$. i.e. $|0\rangle$ remains unchanged, but every vector orthogonal to it gets reflected. This is implemented by the operator that looks like negative identity except that its top left entry is 1 rather than -1 . i.e. the operator is $R = -I + 2|0\rangle\langle 0|$. To perform a reflection about the mean, we have to do a reflection about $|\psi_0\rangle = \sum_x \frac{1}{\sqrt{N}} |x\rangle$ rather than about $|0\rangle$. This is carried out by first mapping $|\psi_0\rangle$ to $|0\rangle$ by applying $H^{\otimes n}$ then applying R and then mapping $|0\rangle$ back to $|\psi_0\rangle$ by applying $H^{\otimes n}$. i.e. we apply an operator $D = H^{\otimes n} R H^{\otimes n}$. Exercise: show that the operator D maps the superposition $|\psi\rangle = \sum_x \alpha_x |x\rangle$ to $|\phi\rangle = \sum_x \beta_x |x\rangle$, where $\beta_x = 2\mu - \alpha_x$. Explain in what sense β_x is a reflection of α_x about μ .

Grover's algorithm consists of starting in the state ψ_0 and applying \sqrt{N} iterations of:

1. Implement a phase reflection of $|a\rangle$
2. Reflect about the mean using the operator D

This process is illustrated in Figure 0.1.

Suppose we just want to find a with probability $\frac{1}{2}$. Until this point, the rest of the basis vectors will have amplitude at least $\frac{1}{\sqrt{2N}}$. In each iteration of the algorithm, α_a increases by at least $\frac{1}{\sqrt{2N}}$. Eventually, $\alpha_a = \frac{1}{\sqrt{2}}$. The number of iterations to get to this α_a is $\leq \sqrt{N}$.

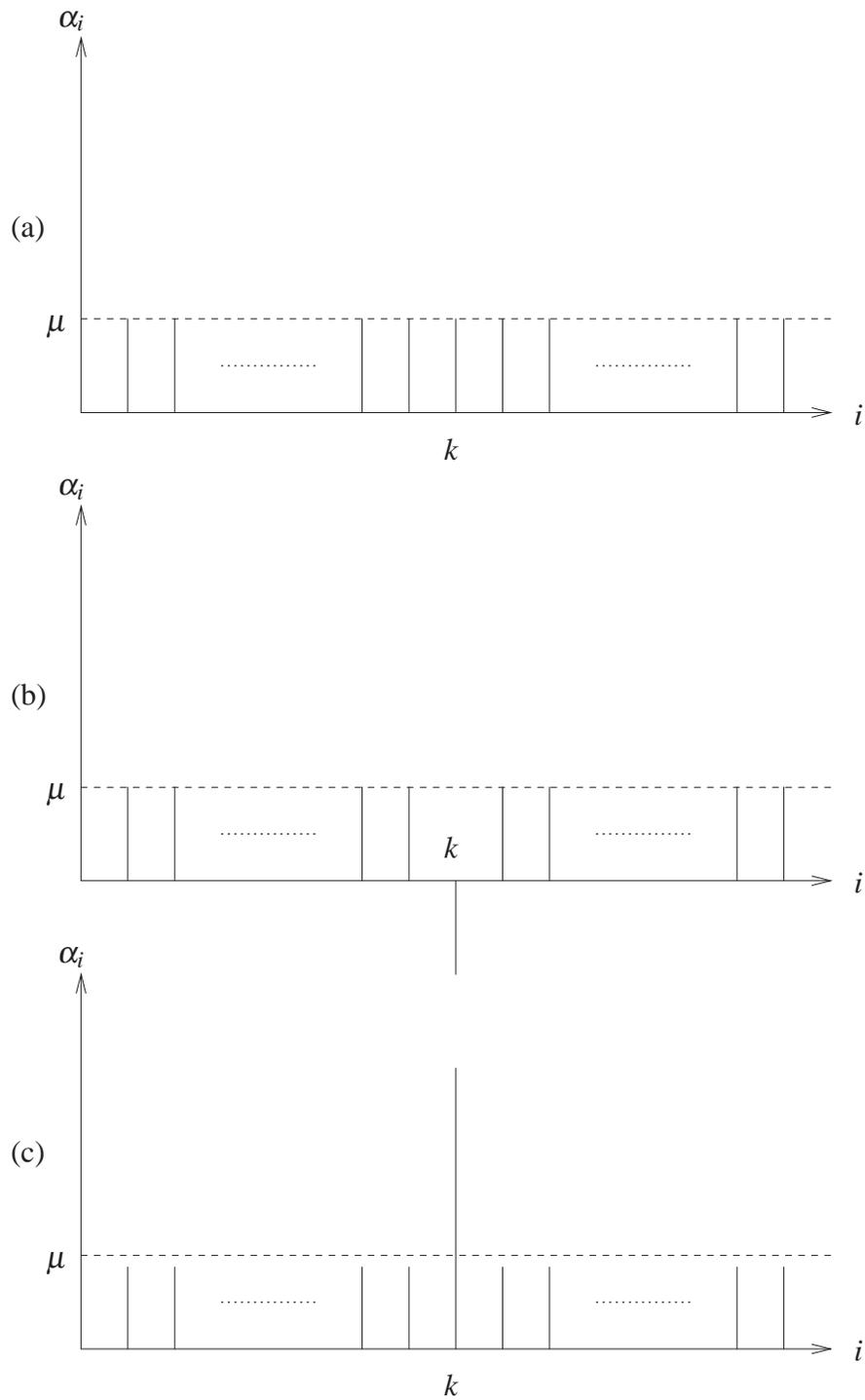


Figure 0.1: The first three steps of Grover's algorithm. We start with a uniform superposition of all basis vectors in (a). In (b), we have used the function f to invert the phase of α_k . After running the diffusion operator D , we amplify α_k while decreasing all other amplitudes.