UNDERSTANDING QUANTUM COMPUTATION AND QUANTUM ANNEALING

BOAZ TAMIR*

The purpose of this paper is to give a non-technical and elementary introduction to the fundamental ideas underlying quantum tunneling, quantum computation, quantum annealing and related topics. We start with the most basic terms such as superposition and tunneling, we then motivate the search for quantum computers. Next we use a toy model to demonstrate the notion of annealing, both classical and quantum. We end with a more formal review of some well-known results.

1. Introduction

In this paper we will describe the main ideas behind quantum annealing. We start with the most basic terms. The text addresses the non-expert reader. In section 2 we present the notion of superposition, a particle can be in several places or states simultaneously. Moreover, a quantum particle has two complementary descriptions, as a wave or as a particle. We present the well-known two-slit experiment suggesting this duality. In section 3 we present one of the most puzzling effects of quantum theory which is the tunneling effect. A quantum particle can find its way through barriers by tunneling. We suggest several uses of such effects. In section 4 we give motivations for the notion of a quantum computer. We start with a most general definition of a computer. The definition suggests the use of many apparatus as computers, whether physical biological or chemical. This suggests the idea of a quantum computer. The most basic model of quantum computation is the 'gate' model. We show how to apply this model on a simple search problem. We present the Grover quantum algorithm and show how it can be used to speed-up all classical search algorithms. In section 5 we describe the possible construction of a real quantum apparatus. We use the construction to present the most important ideas in quantum computation and annealing, such as Hamiltonians, ground states, classical and quantum annealing, adiabatic computation, and more. In section 6 we give a bird's eye view on simulated annealing and quantum annealing, clarifying the language of physicists, all this to allow the reader to follow current research.

2. What is a Quantum Superposition?

Consider the following experiment known as the two slit experiment[1,2,3]. Imagine a cannon that fires electrons into a screen. Suppose there are two slits in the screen (see illustration 2.1).

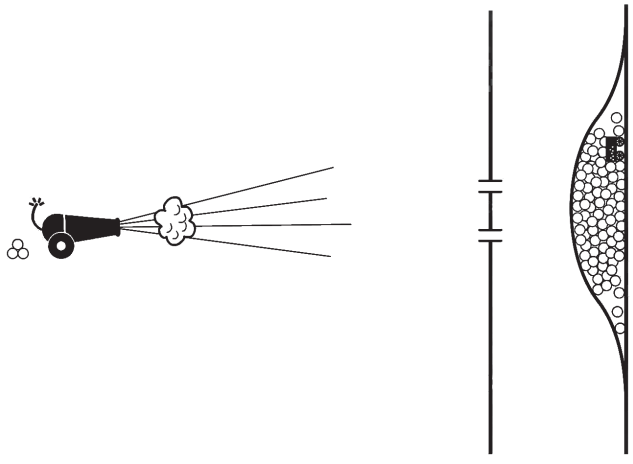

Illustration 2.1 : Electrons or small bullets are fired into a wall with two slits [illustration by Isabelle Zaoui]

At the other side of the screen there is a wall. Suppose we can detect any electron that passes through any of the

* Faculty of interdisciplinary studies, Bar-Ilan University, Ramat-Gan, Israel, Email : canjlm@actcom.co.il

slits and ends somewhere at the wall. We can even count the number of electrons reaching any point at the wall. Now let us repeat the experiment many times while picturing the distribution of electrons reaching the wall. What do we expect to get? We expect that the probability to get the electrons opposite to each of the slits will be high, and far away will decrease. We expect each electron to pass through one of the slits, either the upper or lower, and therefore to reach the wall at a point opposite to the slit with high probability. It could indeed scatter a little bit to the sides while touching the boundaries of the slit.

Well, prepare for a big surprise! On the wall we will see a pattern of interference of two waves. The electrons are distributed as if a wave was originally formed (as if in some medium, say water) and was then split into two sub-waves at the two slits, later to interfere with each other to get the interference pattern at the wall (see illustration 2.2).

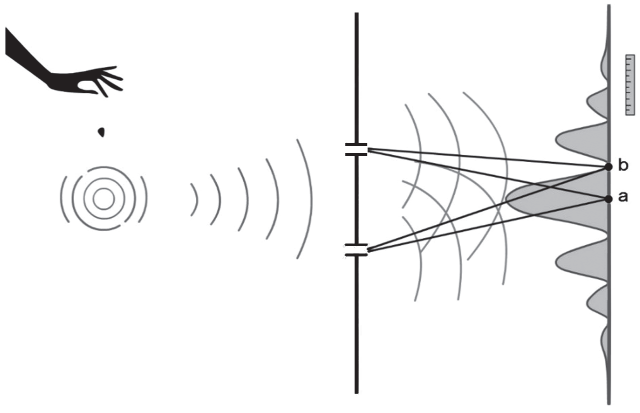

Illustration 2.2 : A wave is formed at left, then the wave is split into two sub-waves that interfere at the wall [Illustration by Isabelle Zaoui]

How come we made an experiment with small bullets, the electrons, and got a wave pattern? There is no medium in which a wave can form! From where these waves are coming? As far as we know, the electrons should resemble small bullets, and not waves in some kind of a medium!

This is one of the main puzzles of early 20^{th} century physics. Is there a way to explain the phenomena? One way is to give up our old concepts of electrons and change the way we think of particles, maybe electrons do have a wave property. If electrons are indeed waves we could easily explain the above results. Each electron is therefore passing through both slits! The probability to find the electron at the wall can be computed from the interference pattern at that point. At the middle point on the wall (point a) both wave appear with the same phase and therefore they strengthen each other, at other points (like point b) there exists some phase difference (180^0 at point b) between the two waves, and the waves could even cancelled out. This could be a good explanation for the results we see, however it opens a Pandora Box full of questions. What is this medium in which our waves propagates? Is there a medium? Do we need a medium? How come electrons change their identity, for most of the 19^{th} century experiments electrons were assumed to behave like small pellets (with mass and charge), how can we explain those old experiments if we now think of the electrons as waves? Could it be that an electron changes its identity according to the experiment in which we try to measure its behavior? This is even worse, a particle that has no clear identity and behaves differently according to the experiment presented. How can we say something about such a particle?

Here are some commonly used terms. We say that the electron is in a superposition of two states 0 and 1, and we denote the states by |0> and |1> ('ket' 0 and 'ket' 1, 'ket' as in the suffix of bracket), where |0> is 'passing through the upper slit', and |1> is 'passing through the lower slit'. We will use the notation |0>+|1>, which denotes the superposition of both states, the electron passes through both slits. Such a superposition is called a 'two state system', or a 'qubit' (quantum bit).

Trying to understand what is going on here, suppose we put a detector between the screen and the wall (see illustration 2.3).

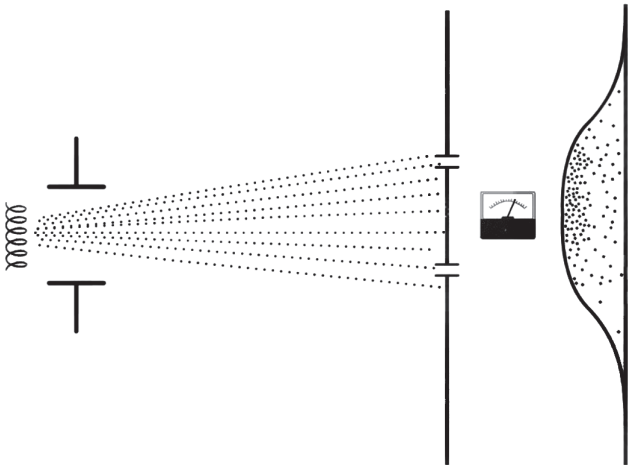

Illustration 2.3 : Trying to measure the electrons in a two slit experiment [Illustration by Isabelle Zaoui]

Will the detector identify the slit through which the electron has passed? Yes, but we will be surprised once more. The electrons will cease to behave like waves and turn back into being particles. On the wall, the distribution will be a simple one with a maximal mode near the center, as if there were no waves and no interference. Note that the detector is a classical measurement device, it could measure classical particles, it could measure the fact that

the electron is passing through one slit or the other, it could not measure a superposition! So if we put such a device it will show one result or the other[4]. So far this is reasonable. The amazing thing is that following the measurement the electron loses its wave identity, and goes back to being a particle! We will say that the superposition has collapsed into either one of its branches. The measurement device has discovered an electron that passed through the upper slit (a state |0>), or an electron that passed through the lower one (a state |1>). We interfered with the quantum superposition with a (classical) measurement apparatus, and thereafter caused the collapse of all super-positions into a classical ensemble of particles, a 'mixed state' of particles, where half of them in state |0> the other half in state |1>.

To sum-up this section, a particle has two complementary descriptions, as a wave or as a particle, it will present only one of its faces, according to the experiment done. This is also known as the wave-particle duality. As a wave the particle is in superposition of all pointwise states, (here only two states |0> and |1>).[a]

3. The Tunneling Effect

Another strange property of quantum physics is the tunneling effect. Quantum particles can find their way through barriers that seem otherwise impenetrable, as if they are tunneling a hole through. When a classical particle is being introduced with a high wall blocking its way, it will bounce back according to its elastic properties. A quantum particle could sometimes pass through the wall to its other side as if it had excavated a tunnel from one side to the other. This effect is called the 'tunneling` effect (see illustration 3.1).

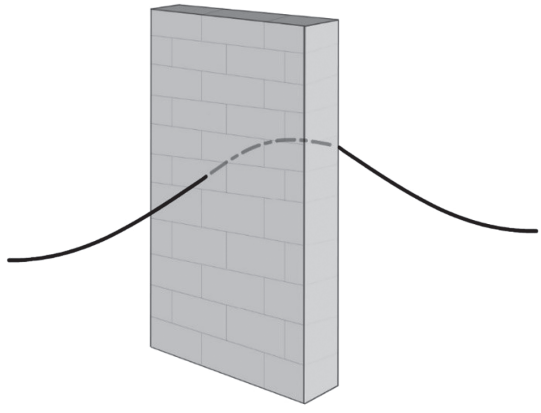

Illustration 3.1 : a wave-particle is tunneling through a barrier [Illustration by Isabelle Zaoui]

Let us try to explain the effect. Clearly the phenomena is connected to the particle's wave property. When the particle approaches the wall, its probability wave is also advancing. We think of a wave function in space as having no clear bounding, its amplitude linger continuously in all direction[b]. We can say that close to the barrier a part of this wave overshoots the wall, coming from the other side. The fact that part of the particle's wave is located at the other side of the wall means that there is some probability to find the particle there. It could indeed be a small probability since the amplitude is decreasing exponential fast at all margins, however this probability is non zero.

It looks like a good explanation, however on a second thought it raises several questions. We saw above that the particle's wave is only a mathematical artifact, how come we use it here as if it actually exists?[c] To say that the wave function extend over the barrier, does this really explain anything, or is it only a rephrasing of the phenomena?

We can get away using some mathematics, we can say that while trying to solve the Schrödinger equation[d] of a particle in a potential having the form of a high barrier, we get solutions with wave amplitudes extending beyond the barrier's boundary[5]. Alternatively we can use the tunneling effect to demonstrate the claim that the particle has wave properties. Instead of using the wave property as an explanation we can use the phenomena to suggest a wave-like behavior.

Experiments do show that such an effect indeed exists, and can be used in several ways as we will shortly describe. As the width of the barrier grows, the effect decreases fast, that is, the probability to find the particle at the other side will decrease fast. If we fire a large amount of particles into the wall we could actually count the number of those who passed the wall, therefore measure the size of the effect. We can use such an experiment to prove that the number of particles decreases exponentially fast as the width of the wall increases.

A tunneling microscope (TMC) uses the effect to map the surface of substances. The head of the TMC approaches the surface, very closely but without touching it. The TMC fires electrons into the surface, the TMC's head and the surface are electrically conducting, the small gap in between constitutes the barrier the electrons should penetrate (see illustration 3.2).

a. And as a pointwise state it could be described by an ensemble of waves also known as wave-packet.

b. In the mathematical language we expect the wave function to be *analytic*.

c. We do believe today that the wave function is physical, see the Aharonov-Bohm effect[1].

d. The differential equation governing the time evolution of the wave function.

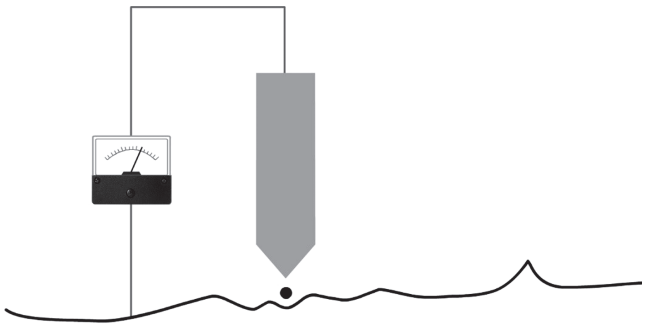

Illustration 3.2 : A tunneling microscope [Illustration by Isabelle Zaoui]

Some of the electrons are tunneling through this gap creating a small electrical current. One can measure this current, its size will be a function of the size of the gap which is a function of the height of the surface. Hence we can picture the landscape of the surface.

Another use of the tunneling effect is in quantum computation. Many computational problem could be translated into finding a minimum of an energy function defined over the problem space. We can think of the computation process as walking over the problem space, looking for a minimal point[a]. We will later see several such examples. Sometimes we will bounce a small barrier preventing us from reaching a much lower point located at the other side of the barrier. Such barriers could be passed if we could tunnel through. It turns out that indeed one can use the tunneling effect to speed the computation process.

Another apparatus that uses the tunneling effect is the 'Josephson Junction'[6]. Such a junction is made of two superconducting wires. The two wires are attached very closely, however there is a small gap in-between. Classical physics do not allow a current to flow through the gap. However if the temperature is low enough, all the particles are forming a coherent wave. The tunneling probability for each one particle is very small, however when a large number of particles are forming the same wave pattern, the small tunneling probability is compensated by the large number of particles and a current is established. At the junction, a phase difference between the two waves is created. It turns out that we can control this phase difference by a DC voltage held between the wires, and hence produce a DC-controlled AC generator.

4. Quantum Computation

A quantum computer is a quantum apparatus using quantum physical properties to improve computation. We need to distinguish between the use of quantum physics in the improvement of current computer gates, or computer hardware, and the use of quantum physics in writing new algorithms, where we use quantum effects to get computational benefits. Current computers use semi-conductors based transistors, a theory which is based on our understanding of quantum physics, however, in a quantum computer we mean something deeper. We want to use quantum effects to produce better algorithms. Here quantum physics plays an important role in the *logic of computation*!

This suggests the evolution of well-established terms in computer science such as computer, computation, algorithm, computational complexity, etc.

For computer scientists a computer is a Turing Machine, an abstract notion invented by Turing in 1937[7].

A Turing machine has a 'head` and a strip of paper. The machine's head has a set of inner states[b]. The strip of paper is used to read inputs from and write outputs on. The machine's head goes over the strip, reads a sign and changes its inner state according to the sign read and its previous inner state. It could then move and write a sign in a different place on the strip. One can show that using these simple operations one can describe all plausible computations. In fact we humans use those exact operations to compute, think of multiplying two big numbers using a paper to write intermediate results.

Implicit in all this is a *physical definition* (and reduction) of the notion of 'computation'. We can say that a certain function is *computable* if we can describe or build a Turing machine that will eventually stop and print the outcome of the function. We therefore translate the function into a set of simple physical operations. Turing had the idea that any computation, even a very complex one, could be described by his machine, given enough time and a long enough strip of paper. A Turing machine could therefore compute what any other machine could compute, sophisticated as it could be. This is known as the '*Turing-Church thesis'*. The thesis has no proof, being only a conjecture, however so far it was not refuted. We cannot find an example of a computation that could not be done by a Turing machine and could be done by some other machine. What would then be 'some other machine'? Let us dwell on this idea.

a. A minimal point is a point where the value of the function is minimal with respect to its values at neighboring points (local minima) or a point where the value of the function is minimal with respect to its values at all space points (global minima).

b. You can think of the head's states as a short term memory.

We will present here a somewhat different physical definition for the notion of a 'computer'[8].

A computer is a physical or biological or chemical apparatus satisfying the following conditions:

1) The apparatus has a computational basis in its input and in its output.

2) The apparatus goes through an evolution from some initial time to a predetermined final time.

What is this '*computational basis*'? It's a set of physical distinguished states of the apparatus that we can read off as far as our resolution power allows us. We can mark the first state by 1, the second by 2, etc. We can then use the computational basis at the input to 'write' the data on which the apparatus will operate. We can 'read' the result (of the computation) off the computational basis at the output, it could be the same computational basis or a different one.

The following example stresses the idea that a computer could be a most general device. Suppose we want to compute the average of three numbers k, m and n. Our 'computer' will be a tank, separated into three compartments. We will fill the tank with water. Suppose we warm the first compartment up to k degrees, the second up to m degrees, and the third up to n degrees, see illustration 4.1.

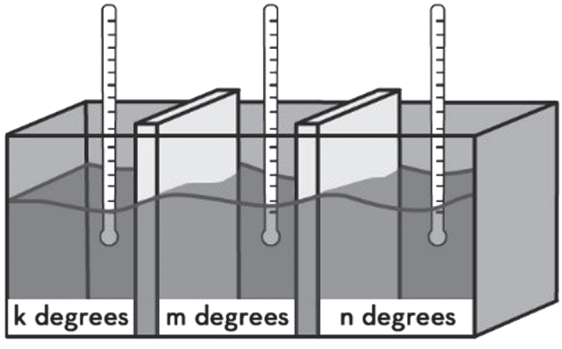

Illustration 4.1 : A thermodynamic computer computing the average of three numbers [Illustration by Isabelle Zaoui]

Now let us remove the two barriers dividing the tank into three. Wait for a while to let the tank 'compute'. The temperature in the tank will eventually reach an average temperature. All is left to do is measure the average temperature which is the number we want. The 'computational basis' here (the same at the input and output) is the set of macro-states, those states (of water molecules) that can be identified by a thermometer, i.e. by their degree. We waited for a while to let the system reach a new equilibrium, this is what is meant by the second part of the definition above, i.e. we let the system evolve up to a predetermined final time. Clearly this is not an efficient way to compute averages, however the example stresses the wide and physical interpretation we can give the notion of a 'computer'.

The definition above could be applied to a wide range of 'machines', be it biological, physical, chemical, almost anything. Indeed almost anything could be a computer, the question is therefore; given an apparatus, what would be the family of functions that is natural (or easy) to compute with this apparatus? Thermodynamic machines are most natural if we want to compute averages. Alternatively we can search for the 'computational power' of physical laws, these are the computational benefits we can gain by using these physical laws. For example what could we easily compute using gravitation? This is indeed a legitimate question.

One of the principles of quantum physics that we can use for computation is the superposition. Perhaps we can implement several tasks in parallel, a superposition of computations! David Deutsch, one of the fathers of quantum computation describes the quantum computer as a superposition of computers, each is doing its task separately and at the end they all interfere to give a final results[2]. Indeed we shall use this metaphor.

The quantum computer cannot help us in solving problems that cannot be solved by a classical computer such as the halting problem[a,7,9]. However, it could be that some problems that are hard to compute by a classical computer will turn out to be easier for the quantum computer. One of the most important problems in the theory of cryptography is the problem of factoring a number into its prime components[b]. Factoring is known to be a hard problem for a classical computer; increasing the number we want to factor will exponentially increase the number of resources we need, the resources are time and memory space. For example adding one more digit to the number we need to factor will multiply the resources by 10. Encryption of information in military and financial systems uses factorization-based algorithms. A huge number (several hundred decimal digits long) is very hard (time consuming)

a. Consider a universal module with two inputs and one output, at the first input we insert the code of any machine, at the second any data, the output of the universal module will tell us if that machine while processing that data will eventually stop. It turns out that such a module is impossible to construct.

b. What are these prime factors? For example 51 is 17*3, where 17 and 3 are prime numbers and cannot be factored any more.

to factor into its primes, however we can easily present such a number by multiplying several big primes. This is known as a '*trap door'*; it is easy to create such a number, however very hard to back engineer. It turned out that quantum computers are good at factoring numbers, therefore quantum computers can easily decrypt such codes. This is one of the reasons for the great interest in quantum computation.

We will try to explain in a very schematic way the most basic working of a quantum algorithm[10]. We will not go into the physical details of the gates. Scientists are not sure yet what would be the natural basic component, the quantum 'transistor' of such a computer.

Let us start with a very simple example, the addition of two bits, what is known to computer scientists as a XOR gate. In the quantum context it is called a CNOT gate for reasons that will become clear immediately. Take a quick look at illustration 4.2.

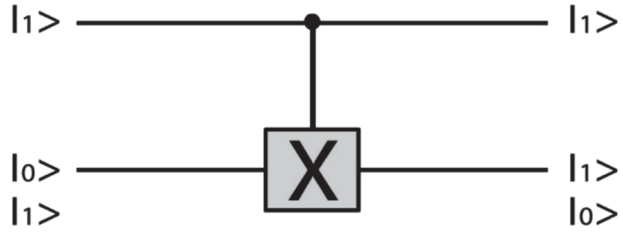

Illustration 4.2 : A XOR or CNOT quantum gate

The upper bit (actually qu-bit) is controlling the lower one, on the lower one there is an X gate, a NOT gate. This is the meaning of the black dot on the upper line, and the vertical line between the two horizontal ones. If the upper qubit is in the state |0> nothing is happening. At the upper output we will still get |0>, and at the lower output we will get whatever we inserted, the X gate is not activated. Now, when the upper qubit is in state |1>, it activates gate X on the lower qubit. The gate X will swap |0> by |1>. If we check the operation of this CNOT gate on each of the possible inputs we will get:

|0>|0>————> |0>|0>

|0>|1>————> |0>|1>

|1>|0>————> |1>|1>

|1>|1>————> |1>|0>

It is easy to see that the output of the second bit (the lower one) is the sum modulo 2 of the two input bits[a], which is also the action of XOR. Now, since the above gate is a quantum one its input accepts a superposition of all four cases. The quantum gate will process the above four computations in parallel, in one step. Several such primitive quantum gates do exists[10].

Similarly we can *quantum-transform* other classical gates. The NAND gate, (an AND gate followed by a NOT gate) also known by physicist as the Toffoli gate[11], has a similar quantum counterpart in the form of CCNOT[b]. It is well known that the NAND gate is universal for classical computation, in other words we can write any logic gate using only NAND gates, hence its importance for classical and quantum computation.

Let us now build a quantum computer. Suppose we have a classical computer, write the computation using only NAND gates. Replace any NAND gate by its quantum counterpart CCNOT. Our new quantum computer will carry out the same computation, only now all operations are carried out simultaneously. Let us present a computational problem where such a quantum computer could be helpful.

Suppose we are given a non-ordered array of inputs, it could be names and addresses. Suppose we need to find the address of someone. Since the array is not ordered we might have to go over the whole list before we reach the person whose address we need. This is like searching for a needle in a haystack. Such a problem demands a large set of simple checks. Each such check could be very easy however there is a large number of them.

At the input to the computer we will introduce a superposition of all states representing integral numbers from 1 to N, the size of the array. Now a classical computer will compute a result for each of the inputs. It will write 0 on some special bit (a flag bit) to denote that this input (array cell) is not the one we are looking for (is not a solution for our search problem), and it will write 1 on this flag bit to denote that the corresponding input number is indeed a solution for the search problem. A classical computer will go over each of the inputs, one by one. Our quantum computer will compute all of the above using only one step. At the output of the quantum computer we will get a huge superposition, each branch contains the values of two registers, one holds a serial number of one of the array's cell, the other holds the result of the search on that cell, see illustration 4.3.

a. Meaning 1+0 = 1 and 1+1 = 0.

b. A CCNOT gate is a controlled CNOT, as if the gate in illustration 4.2 is controlled by another qu-bit, therefore CCNOT is a three qubit gate.

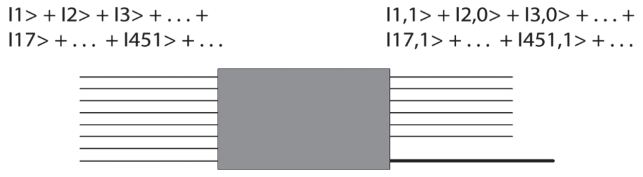

Illustration 4.3 : A classical computer transformed into a quantum one

In the example (see illustration 4.3) cells number 1, 17, and 451 are indeed solutions to the search problem, 3 is not. This huge superposition is the outcome of one computational step, this is the power of quantum computation. This looks very promising. All we have to do now is measure the superposition and find those places where the flag register holds the number 1.

Well, not so simple, how can we read the superposition? If we attach a measuring device the whole superposition will collapse into one of its branches. Now since we have vastly more branched with 0 at their flag register, we will probably get something like '352 is not a solution'. We could have reached such a result by a simply guess, for that we don't need a quantum computer. Note that in the superposition above all amplitudes are equal. We have to somehow increase the amplitudes of all the branches with the 1 flag register, while decreasing all others. But how can we do that, we have no idea where they are placed in the array (otherwise we would have had the solution). This is like having a probability amplifier, or an amplitude amplifier. Such an amplifier cannot go over all the branches one by one, since then it will resemble a classical computer. The quantum operations that are allowed in quantum computation are all global. Any action that separates between some of the branches will cause the collapse of the computation to a classical one.

Could we build such an amplifier? Classically this sounds an absurd! Consider a virtual opaque box where I have many beads, one is colored red, all the other black. For some reason I can draw out only one bead at a time. I want the red one. It could indeed be that I would have to draw out almost all the beads before I reach the red one. Suppose now I have an algorithm by which I shake the box, turn it upside down and shake again, perhaps followed by some other operations. All the actions on the box are global ones, on all the beads simultaneously. Suppose this series of global actions increases the probability that I will pick the red one the next time I try, this is the algorithm we need!

Surprisingly, such a quantum algorithm do exists and is known as the 'Grover algorithm'[12]. The amplification is made by several iterations. Each iteration has two steps. Let us try to demonstrate one such iteration for a two qubit quantum computer. Our superposition has four states |00>+|01>+|10>+|11>. Suppose now the third state |10> is the one we are looking for. Following the above arguments, the state |10> should be marked by a flag register in a state |1>. Therefore at the input to the Grover algorithm we should have:

|00>|0>+|01>|0>+|10>|1>+|11>|0>

The first step of the Grover iteration uses the flag register to mark the state |10> by a minus sign:

|00>|0>+|01>|0>+|10>|1>+|11>|0> ————>

|00>|0>+|01>|0>–|10>|1>+|11>|0>

What is the physical meaning of such a state? We can think of a wave function that splits into four waves (as in the slit experiment above) whereas the third one has a phase shift (is lagging by) of 180^0. How do we carry out such an operation? It can be done using the CNT gate we introduced above. Observe illustration 4.4 below, note that the lower qubit is in the state |-> which is |0>–|1>.

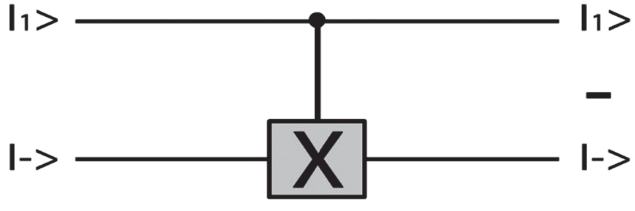

Illustration 4.4 : a CNOT gate for marking a state

If the **X** (NOT) gate is being activated (when the upper qubit is in state |1>), the lower qubit is marked by a minus sign:

|0> – |1> ———> |1> – |0> = – (|0> – |1>),

where the minus sign is attributed to the whole branch of the superposition at the output. You have probably guessed, the upper qubit of the CNOT gate is the flag bit that testify that this branch is a solution. In this way we can change the phase of the marked branch without any knowledge of its identity. This was an exercise in quantum algorithm!

The next step of the iteration is a reflection operation over the average of all amplitudes. In the case above the sum of the amplitudes is 1/2+1/2–1/2+1/2 =1 and the average is ¼. A reflection of the amplitudes over ¼ will result in the cancellation of three amplitudes and the amplification of the fourth one to unity. The amplified one is exactly the third one (see illustration 4.5 below). Now is the time to measure the superposition. It will definitely yield the third one, the one we were looking for.

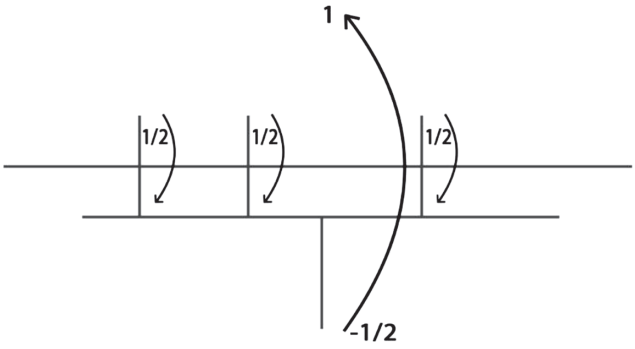

Illustration 4.5 : The Grover algorithm on a two qubit superposition

This second step of the Grover iteration is also a global action, it operates on all branches simultaneously, and therefore is legitimate in the quantum context (we will not go here into its details).

For the two bit quantum computer one Grover iteration was enough, for a three qubit superposition, (a superposition of eight states) we will need two Grover iterations. If we continue to iterate we might deteriorate the relation between the marked state and the other states. In general, if the array has N states and there is only one marked solution, we need to iterate $\sqrt{N}$ times before we can be sure that the marked state has a considerably high amplitude. In a classical computer we might need N operations to find the solution since the array is not sorted. The quantum computer will therefore be more efficient. The decrease in the computation time was due to the fact that we have introduced a new physical element into the algorithm, the superposition. We can use this method to speed-up any classical computation. For several other problems this will not be the best way to use quantum computation. In fact, for the problem of Factoring there is a better algorithm suggested by P. Shor[13]. The algorithm uses sub-modules that can identify hidden symmetries in a function, this is closely related to the Fourier Transform of that function. Its complexity is of the order of a polynomial of degree three in the number of decimal digits. This is exponentially better that what we can achieve today using any classical algorithm.

All in all this suggests that some problems that are hard to compute using a classical computer could be easier to carry out on a quantum computer. Note however that the exponential difference of efficiency for Factoring is only ad-hoc. So far we know of no classical algorithm that can solve Factoring efficiently, that is without a need for resources that grow too fast (exponentially). We have no proof that Factoring *should* be hard on a classical computer. As for the Search problem it is indeed easy to prove the difference of efficiencies, however such a difference is consider minor. In other words improving the efficiency from N to $\sqrt{N}$ does not change the `complexity class' as defined in computer science.

5. Quantum Computation by Annealing

In the following we will suggest a quantum apparatus, a quantum annealing computer. In this section we will use a somewhat schematic toy example to demonstrate the annealing process, both classical and quantum. In the next section we will use a more formal language to review some recent results.

Here is a description of our virtual 'quantum microprocessor'.

Imagine an ensemble of elements, such that each can be in one of two states, 0 or 1. Assume there are couplings between some of the elements. For example we can think of small magnets, each has a north and a south pole. A north pole is attracted to a south pole. If we wish to attach two such magnets such that a north pole is close to a north pole we will have to invest some energy, when released they will go back to their natural state, a north pole side by side with a south pole. Suppose we denote a north pole by 1 and a south pole by 0 then the two magnets will stay at 01 or 10 state with no energy invested. We will call such a conjugation or coupling a negative one (this is only a name). Now suppose those elements have some other type of couplings, a positive one, in which we can hold each pair together at the state 00 or 11 without investing any energy, for such a coupling we will have to invest some energy to hold the elements in the state 01 or 10.

Suppose now we have an array of such elements[a], organized in a 2 dimensional grid. Suppose we can pre fix the type of coupling between any two elements. So any two elements with a negative conjugation will probably be found in a state 01 or 10 and any two elements with a positive conjugation will be probably found in a state 00 or 11. Why probably? Well, it could be that one element has a positive conjugation from one side and a negative conjugation at the other side, it could be that the constraints from one side oppose the constraints from the other side, we will say that such an element is in a state of frustration. It could be that it is better to hold one such element 'against its will' (at least partly) in order to satisfy several other conjugations.

At the illustration below (illustration 5.1) if the state of the element in the center is 0, it satisfies the demands

a. You can think of an ensemble of such magnet monopoles, in reality magnets are dipoles coupling two monopoles.

of the conjugations from left, right and below. It will not satisfy the upper conjugation.

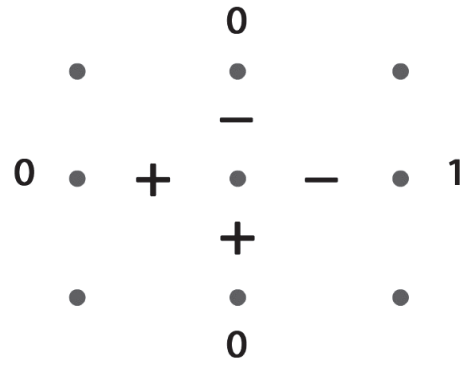

Illustration 5.1 : What is the preferred state of the center element?

So we will have to invest some energy to hold it in its position against the detraction of the upper element. However, if we try to set the state of the center element to be 1, it will not satisfy the demands of 3 conjugations (left, bottom and right), and we will have to invest much more energy to hold it in that (the 1) position. Clearly the system prefers the less energetic solution, that is, the configuration where the center element is in state 0. It is therefore clear that for each possible configuration of the elements we can compute the energy needed to hold that configuration stable. The system by itself looks for the configuration with lowest energy.

We will now demonstrate how such a grid can define a computational problem, and how the lowest energy configuration of the elements defines the solution to that problem. First look at the following game-theoretical example. Suppose there is a factory where two types of workers are hired, red and green. The workers are ordered in a grid such that between each two neighboring workers there is a professional relationship. The professional relations are important to the efficiency of the factory. Some of the relations are monochromatic, in such case the two workers have to be of the same color, both green, or both red. Some of relations are heterochromatic and therefore the two workers having those relations have different colors, one red the other green. All professional relations are predetermined and define the essence of the factory. The problem is to place the workers such that as much as possible professional relations are satisfied.

Let us try to state the above example in a mathematical language. Assume we have n variables $X_1,\ldots,X_n$. Suppose the variables are ordered on a grid. Each variable 'knows' its neighbors. We assume each variable is Boolean (0 or 1). Suppose also we have a set of equations, for example, $X_i=X_j$ for the neighbors i and j, and $X_l \neq X_k$ for the neighbors l and k. We are looking for a configuration of the variables solving as much equations as possible. For example assume we have 5 variables $X_0,\ldots,X_4$ and 4 equations:

$$X_1 \neq X_0,\ X_2 \neq X_0,\ X_3 = X_0,\ X_4 = X_0$$

We can try several configurations, it could be that some configurations do not satisfy some of the equations, but we search for those configurations that satisfy the most. Why not try all possible configurations? Clearly if there are n variables there are 2^n configurations, we could work very hard even for a low number of variables. Is there a way to solve such problems? The system of elements we described above is exactly constructed for that purpose. For each Boolean variable we match an element of the system. We will build the couplings between the elements according to the above equations. For an inequality we will use a negative coupling, and for an equality we will use a positive coupling. We actually write the equations into the 'computer' using the couplings, in other words the couplings are turned into physical constraints. The above example (of 5 variables) is solvable using the system of elements described in illustration 5.1 above. All we need is to match X_0 to the element at the middle and $X_1,\ldots,X_4$ to the elements above, on the right and clockwise. The system will stabilize on a solution. The solution is read off the configuration of the elements after the system had stabilized. As lower is the energy of the system, the better is the solution. Satisfying the conditions does not cost energy, only dissatisfaction cost energy. So we need to force the system into a lower energetic state, as lower as we can. How can we do it? By putting the 'computer' in a refrigerator. The system will lose its energy and thereafter we can pull the 'computer' out and read the final configuration. Will this always work?

We should be careful; if we look at the space of all possible configurations we can compute an energy level to each point, the minimal energy needed to hold that configuration stable. We can even picture an energy landscape of the space, see illustration 5.2.

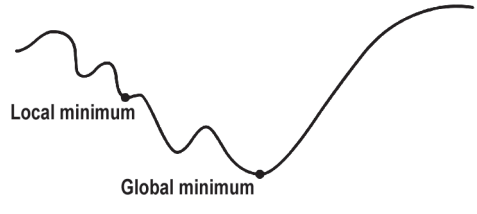

Illustration 5.2 : The energy landscape

Our 'computer' walks over the space looking for a minimum point. We have to differentiate between local and

global minima. If we cool the system too fast the computer might get stuck in a local minimum. To get out of such local minima one has to warm up the system, give it enough energy to climb out. Later we can re-cool the system, our goal is to reach a lower energy point, possibly the global minimum. The process resembles an annealing one, therefore called '*simulated annealing'*.

So far everything looks as if it could all be classical. We could perhaps implement the 'computer' using some set of strings and springs. What is quantum in all this?

Imagine now a similar quantum system, where the elements are qubits, in particular we can use electrons' spins[a]. Spin is a natural two state system. The couplings between the spins might force them to align or anti-align locally. The evolution of the system from its initial state to its final state will be governed by a quantum Hamiltonian. Well, how could all that help us?

During evolution time we can use several quantum principles, one is known by the name '*adiabatic computation*'. Suppose we have a similar spin-computer as above solving a most simple problem, not the problem we want to solve but a different one which is very simple (where we know its solution). We write the constraints of that simple problem into the set of couplings, and this is our *initial Hamiltonian*. Assume also that we start with a configuration of spins that solves those constrains, i.e. a *ground state* of the initial Hamiltonian. Now let us do a very strange thing, we will change the set of *couplings*, very slowly from the set defining the simple problem to the set defining the problem we want to solve, in other words we are changing our initial Hamiltonian towards a final Hamiltonian, known also as the *problem Hamiltonian*. Changing the set of couplings, very slowly, is like changing the computer itself (its hardware). The principle of adiabatic quantum mechanics states[b,14];

If we start from the ground state of an initial Hamiltonian and evolve the Hamiltonian slowly enough towards a final Hamiltonian, then the ground state will miraculously evolve into the ground state of the final Hamiltonian, the solution we seek!

This is a quantum magic! We start with a computer that solves a simple problem, the initial state of the system is a solution of this simple problem, we change the computer itself (its hardware), until it defines a new problem. We don't have to worry about the solution, it will show up, as long as we do everything slowly enough.

It turns out that such a system is even more efficient than a classical one. In an adiabatic computer the computational complexity is measured by the time needed to evolve the system from its initial state to its final state. We are forced to evolve slowly. If we try to rush the process we will end up in a configuration which is not the solution we want. Therefore the minimal time needed is the computational complexity. If we compare this minimal time to the classical complexity we will find that the adiabatic computer is indeed better for a large family of problems. It is also a known fact that the adiabatic computer is equivalent to the standard gate–type computer that we discussed above (in section 4)[15].

Another quantum principle we can use during the evolution towards the final state is the tunneling effect (see section 3 above). Adding some quantum noise, (a transverse field) our spin-computer can cross energy barriers in its way to lower energy points. Such a computer is named a quantum annealing computer (see the discussion below).

6. A Bird's Eye View of Quantum Annealing

Let us formalize our discussion above in the language used by physicists. This will enable the reader to follow current research. For a recent and most general review of the topic see[16] (for a more pedagogical introduction see[17]).

Our computer, the machinery used to solve the computational problem is coined '*Hamiltonian*'. This could be the set of rules governing the apparatus' behavior, we can also think of the Hamiltonian as the algorithm, or the software, or even the physical hardware. The Hamiltonian is acting on a space of states, taking an initial state to a final state, which is the solution to the computational problem.

It is also useful to think of the Hamiltonian as an energy function. In fact we write the energy function such that the lowest energy state, the '*ground state*', is the solution we seek (see section 5 above). The Hamiltonian, once finding this ground state should leave it as it is (the

a. A spin is a two state system of an electron, spinning both clockwise and anticlockwise. Pick any direction, align a spin-measurement device in that direction, and you will find the electron spinning parallel or anti-parallel to that direction. The electron is therefore in a superposition which collapses to either one of its branches, whenever a measurement device is presented. Such a quantum measurement is known as a Stern-Gerlach measurement.

b. In fact the adiabatic principle is much more general and this is its interpretation to the theory of computation.

computer stops processing when finding a solution). Such a state which is left invariant under the Hamiltonian is coined an '*eigenstate*'. Writing the Hamiltonian is no more than writing the set of clauses the solution should satisfy. We saw an example above with a small number of equations and a simple energy function (see illustration 5.2). The fact that we can write a set of clauses does not mean that we know how to solve it, writing a set of equations is different than solving it. We can indeed construct a Hamiltonian, i.e. can build the machinery that is equivalent to the set of equations, however to find the solution we have to turn on the machinery and let it process the initial data until it reaches the solution, the ground state. In the language of computer scientists we say that the Hamiltonian can be built 'efficiently', or the circuit complexity grows only polynomialy with the size of the problem space.

If our Hamiltonian is indeed an energy function of some physical system then all we need is to cool the system into its ground state. This is the main intuition behind the annealing process.

There are several ways to reach the ground state. Classical annealers use a '*gradient descent'*. If we look at the energy landscape, gradient descent resembles a blind man's walk. Using his stick the blind man can sense the direction of the gradient. Similarly, we can start with any configuration, a candidate for the ground state, we can compute the energy of that initial state. Next we make a small perturbation on the candidate. We then check the energy of this perturbed state. If the energy is lower than the energy of our initial state then this perturbed state is our new candidate. We can go on like this until we reach a minimal state. This is a state whose perturbations in all directions only raise the energy. The problem with such a gradient descent is obvious, we could end in a '*local minimum'*, far from our true ground state, the '*global minimum'* whose energy is the lowest (see illustration 5.2). The way to solve this problem is to add some heat (noise) into the system. We must let the system overcome those local minima. The system must continue exploring the landscape. This is the question of exploiting *versus* exploring. The way to do it is by letting the system explore a state even if the energy of that state is higher than the energy of the current candidate. So let us improve the gradient descent algorithm, we can toss a coin, and with some probability let the algorithm go into a perturbed state even if its energy is higher. This is very similar to an annealing process, in which we alternate between long phases were we cool the material, and short phases where we warm it up. A system that explores all possible configurations, not necessarily with the same probability, is coined an '*ergodic* system'. We therefore improve the ergodicity by adding this stochastic phases.

Let us control the amplitudes of the warm ups; at the beginning of the evolution we can let it be relatively high, reducing the amplitude towards the end; let the probability to go to a configuration with a higher energy be proportional to $e^{-\beta\Delta E}$ where ΔE is the increase in the energy, and β is proportional to 1/T (where T is the temperature)[18]. Therefore for room temperature (high temperature, low β) the probability to visit any configuration will be relatively high and the system will be ergodic, for low temperature (high β) the probability to jump to higher energetic states will decrease sharply. Therefore by controlling the rate of cooling the system, we also control the warm ups (and exploration).

Suppose the temperature T does not decrease faster than the order of n/log(t) where n is the dimension of the system and t is the time. It was proved in[19] that such a protocol will end with the right ground state. This algorithm is known as '*simulated annealing*'. The minimal evolution time needed to assure that the system will end in the ground state is the '*computational complexity'* of the algorithm.

Another way to reach the ground state is by using '*adiabatic computation'*. We start with an initial simple Hamiltonian having a simple ground state that we know how to construct. We evolve the system very slowly into a final Hamiltonian also known as the '*Problem Hamiltonian'*. This is like evolving the computer from a very simple one into our more complex one. If the evolution is slow enough (and we have started from the ground state of the initial Hamiltonian) the quantum adiabatic theorem says that the final state will be the ground state of the final Hamiltonian, i.e. the solution.

Yet another way to reach the ground state is by '*quantum annealing'*. We replace the warm ups short phases of the simulated annealing by a '*fluctuation field'*. This is a small component we add to the Hamiltonian. The state governed by the Hamiltonian behaves as if it 'looks' around and explore neighboring configurations, or even tunnel through barriers. One can control the amplitude of the fluctuation field, reducing it towards the end of the process, this is similar to a simulated annealing algorithm only now the control knob is a different one. It was proved in[20] that such a process is bound to yield the ground state. Again, the amplitude of the fluctuation field should decrease no faster than $1/t^{1/n}$. This is similar to the computational complexity time for simulated annealing, and even a little

bit better[a]. Quantum annealing is usually done in low temperature, and sometimes also adiabatically.

For some specific computational problems one can show a clear preference for quantum annealing over simulated annealing. In[21] a toy model of 8 qubits was tested. The quantum annealing algorithm yielded the ground state with much higher probability than that of a simulated process with the same time schedule[b]. In[22] a traveling salesman problem was discussed for 1002 cities, there the error rate (between the known solution and the final state) reduced much faster for quantum annealing as compared to the error rate for simulated annealing.

In the following several paragraphs we discuss a map between statistical mechanics and quantum adiabatic computation presented in[23], this will demonstrate some of the principles discussed above. Consider an ensemble of classical elements satisfying a set of clauses similar to the set presented above in section 5. This is an ensemble of n-configurations (n-tuples where n is the size of the system). Assume also that the number of configurations of the same type is inversely proportional to the energy of that configuration (the energy needed to hold that configuration) with an exponential factor; $e^{-\beta E}$. This means that there are exponentially more elements with low energy than elements with high energy. Such a distribution is known as the '*Gibbs distribution*'[24]. When sampling from such an ensemble we will probably get a configuration with very low energy, or a configuration that solves many clauses.

Assume now that this coefficient β is high. This will increase the probability to end with the configuration we need. In fact this is what we get when we cool the system. We increase the probability to get the low energy states. However at room temperature the probability to get any of the configuration is the same, otherwise we could have easily guessed the solution. Therefore we can imagine that β is a function of temperature, at room temperature β is low and the exponent is flat, and at low temperature β is high and the exponent is sharp.

Let us now map everything into the quantum realm. The above ensemble of elements are '*mixed states*' in the sense discussed in section 2. A measurement process maps a superposition into a classical state, here we need an opposite map, from mixed states into superpositions. In fact we can map our ensemble into a superposition by replacing probabilities with amplitudes (at least on the paper by square rooting all probabilities[c]). Now we need to find an initial Hamiltonian such that the room-temperature mixed state is its ground state, and a final Hamiltonian such that the low temperature mixed state is its ground state. It is easy to find a Hamiltonian for the room-temperature mixed state, since such a state has almost equal amplitudes. As for the final Hamiltonian this could be some variant of our problem-Hamiltonian. Next we can use adiabatic computation, slowly evolving into the ground state of the final Hamiltonian. All in all we have started with a classical ensemble of particles and mapped the classical search into a quantum adiabatic process. It turns out that the minimal time needed for such an adiabatic computation is close to the time needed for a classical simulated annealing computer.

Consider again our grid of spins discussed in section 5 above, and let us take it one step further. Assume each of the spins 'knows' all the others. Each spin is coupled to all other spins, like a network of connections. Above we assumed only positive +1 or negative -1 couplings between the spins, however suppose now the couplings are taken randomly from a normalized Gaussian distribution. We prefix the strength of each of the coupling by random sampling from a Gaussian distribution, only then we let the system evolve, this is known as a '*quenched*' randomness. Such a system is known as the Sherrington Kirkpatrick model[25]. We can define a similar model where each spin only 'knows' its nearest neighbors on the grid, and this is known as the Edwards Anderson model[26]. Such models have the property that each of the spins sees the same similar set of couplings, and we can therefore replace the spins' interactions with some mean value, hence the name '*mean field theory*'[27].

We coin such grids '*spin glasses*', they are glass-like amorphous. The spins show many irregularities, the energy landscape has many local minima with high barriers in-between. It is hard for the system to '*relax*', that is, to pull itself out towards a global minimum. It will indeed eventually relax and therefore these local minima states are also known as '*metastable states*'. This resembles the process of '*vitrification*' where a substance is cooled by some fast procedure to produce a glassy type solid. The glassy solid is extremely amorphous. It will eventually relax

a. The above computational complexity estimations are general, for some specific problems one can get better results.

b. Since the model has low dimension one can compute the solution in advance. The model only compares the complexity times of both models, the classical and quantum annealing.

c. The coefficients of a superposition are the amplitudes, by squaring the amplitude we get the probabilities to collapse to the corresponding classical state.

into its crystalline phase but this might take a very long time.

We can compute the correlations between the configurations of those sets of local minima, or their '*overlap*'. These correlations are also known as '*order parameters*' of the system, they characterize the system. A very simple system with zero external magnetic field and low temperature below some '*critical point'* will find itself with all spins up or all spins down, therefore two final configurations with an obvious simple symmetry. In contrast, a glassy state will be much more complex, breaking this symmetry[28].

What if we now try to use a quantum annealing process by adding some fluctuation - a '*transverse field'* - to our Hamiltonian? Could we reintroduce exploration into the system? Will the system be able to tunnel its way through the landscape of many local minima and high barriers?

The story of quantum annealing begins at exactly that point some 30 years ago when such Hamiltonians were being introduced and investigated. In[29] it was shown that indeed the transverse field brings back ergodicity into the system, the number of order parameters reduces (in the language of physicists Replica Symmetry Breaking is lost), as if the system is a much simpler one. This suggests that the transverse field helps the system 'look' over barriers.

To sum up this section: our machinery has several control knobs, one for the fluctuation field, one for the cooling process, and one for the pace of the adiabatic evolution. On this 'phase space' of control knobs we can select the path we want. Quantum annealing is a more general concept than adiabatic computation involving the control of the fluctuation field and the control of the adiabatic evolution. A D-wave computer[30] uses both processes, adiabatic and annealing, on a superconductor hardware at a very low temperature.

7. Concluding Remarks

We wish to conclude with some philosophical insights. Underlying all the above is a map between mathematical abstract computational problems and physical machinery designed to solve those problems. Turing was the first to define the abstract notion of computation in terms of a physical machine, starting with some initial condition and ending in a final state that defines the solution. Under the above map, computational problem are mapped into Hamiltonians, solutions are mapped into ground states, computational complexity is mapped into relaxation time etc.

The main motivation for the search for quantum computers stem from the need to reduce the computational complexity. If computation is physical it is only natural to introduce physical principles into the computational process. In quantum computation theory we introduce quantum principles such as superposition and entanglement. Introducing physical principles into algorithms change the way we think of algorithms.

As an example of the intricate relation of abstract and physical computation observe the problem of scaling-up[a]. Problems should be scaled up such that the (circuit) complexity of the Hamiltonian, and the relaxation time will not grow too fast. For example, in the adiabatic computer the (abstract) computational time-complexity is mapped into the gap between the two lowest energy eigenstates of the Hamiltonian (a physical property). If the gap decreases too fast we have to slow down the computation to prevent the system from jumping into the higher energy state.

In particular we discussed the notion of quantum annealing. Quantum annealers can somehow 'see' behind barriers, maybe tunnel through. Such computers can re-introduce ergodicity into a complex energy landscape having many local minima. It is therefore plausible that such computers can reduce the computation (relaxation) time for several computational problems (see the examples above). Here again we convert a physical property into a computational one.

Quantum annealing was introduced some 30 years ago in the context of statistical mechanics, condensed matter physics, spin glass theory etc. It was much later presented as a tool in quantum computation. Yet there are many open questions as to the relation of quantum annealing to quantum computation.

Acknowledgement

I am grateful to Isabelle Zaoui for her help in all the illustrations used in this paper. ❐

References

1. Y. Aharonov and D. Rohrlich, Quantum Paradoxes, Quantum Theory for the Perplexed, Wiley-VCH, (2005).
2. D. Deutsch, The Fabric of Reality. (New York: Viking Press), (1997).
3. R. P. Feynman, R. B. Leighton and M. L. Sands, The Feynman Lectures on Physics, Vol 3 - Quantum mechanics. Reading, MA: Addison Wesley, (1965).

a. Solving the same problem for a higher number of variables

4. D. Albert, Quantum Mechanics and Experience, (Cambridge: Harvard University Press), (1992).
5. A. Messiah, Quantum Mechanics, volume I, (North-Holland Publishing Company, Amsterdam) (1970).
6. B. D. Josephson, "Possible new effects in superconductive tunneling", *Phys. Lett* **1**: 251-253, (1962).
7. A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem", Proceedings of the London Mathematical Society **2**: 230-265 (1937).
8. I. Pitowsky, "The physical Church Thesis and physical computational complexity", *Iyyun* **39**: 81-99, (1990).
9. D. Harel, Computers Ltd. What They Really Can't Do, (Oxford: Oxford University Press), (2000).
10. M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information, (Cambridge: Cambridge University Press) (2000).
11. T. Toffoli, "Reversible computing", in: de Bakker J., van Leeuwen J. (eds.) Automata, Language and Programming. ICALP 1980. Lectures Notes in Computer Science, vol 85. Springer, Berlin, Heidelberg, pp. 632-644, (1980).
12. L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack", *Phys. Rev. Lett.* **79**: 325-328, (1997).
13. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM Journal on Computing,* **26 (5)**: 1484-1509, (1997).
14. E. Farhi, J. Goldstone, S. Gutmann and M. Sipser, (2000). "Quantum computation by adiabatic evolution", arXiv:quant-ph/0001106
15. D. Aharonov, W. van Dam, J. Kempe, et al. "Adiabatic quantum computation is equivalent to standard quantum computation", *SIAM Journal of Computing* **37**: 166-194, (2007).
16. S. Tanaka, R. Tamura, B. K. Chakrabarti, Quantum Spin Glasses, Annealing and Computation, (Cambridge University Press) (2017).
17. E. Cohen and B. Tamir, "D-wave and predecessors: From simulated to quantum annealing", International Journal of Quantum Information 12: 1430002, (2014).
18. S. Kirkpatrick, C. D. Jr., Gelatt and M. P. Vecchi, "Optimization by simulated annealing", *Science*, **vol 220**, 4598, (1983).
19. S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". *IEEE Trans. Patt. Anal. Machine Intel.* **6**: 721 (1984).
20. S. Morita and H. Nishimori, "Convergence theorems for quantum annealing", J.Phys. A: Math. Gen.39: 13903, (2006).
21. T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model". *Phys. Rev.* **E 58**: 5355, (1998).
22. R. Martonak, G. E. Santoro and E. Tosatti, "Quantum annealing of the traveling-salesman problem". *Phys. Rev.* **E 70**: 057701, (2004).
23. R. D. Somma, C. D. Batista and G. Ortiz, "Quantum approach to classical statistical mechanics", *Physical Review. Letters*. **99**: 030603, (2007).
24. L. D. Landau and E. M. Lifshitz, Statistical Physics, (Oxford: Pergamon Press) (1980).
25. D. Sherrington and S. Kirkpatrick, "Solvable model of a spin-glass", Physical Review Letters 35, 1792, (1975).
26. S. F. Edwards and P. W. Anderson, "Theory of spin glasses", Journal of physics, *F: Metal Physics*, **5** (5) 965 (1975).
27. H. E. Stanley, Introduction to Phase Transitions and Critical Phenomena, (Oxford University Press) (1971).
28. G. Parisi, "Order parameter for spin-glasses", *Physical Review Letters,* **vol 50** (24) pp.1946-1948 (1983).
29. P. Ray, B. K. Chakrabarti and A. Chakrabarti, "Sherrington-Kirkpatrick model in a transverse field: Absence of replica symmetry breaking due to quantum fluctuations", Physical Review B, **vol 39**, (16) pp.11828-11832 (1989).
30. D-wave group. 2013. "D-wave: comment on comparison with classical computers", http://www.archduke.org/stuff/d-wave-comment-on-comparison-with-classical-computer.