Quantum Computation and NP-Completeness

Dave Bacon
Summer Undergraduate Research Fellowship Paper
Advisors: Nicholas Cerf and Chris Adami
California Institute of Technology

ABSTRACT

As computers become smaller and smaller the physical laws which govern the computer's operation will shift from the classical arena into the quantum mechanical realm. Since computation is essentially a physical process it is not unreasonable to suggest that the computational power of quantum mechanical processes might be beyond that of traditional classical computational models. Resent results have supported this hypothesis without explicitly describing the relationships between traditional complexity classes and quantum complexity classes. This research examines the relationship between quantum computers and NP-complete problems. Specifically we examine the question as to whether the boolean satisfaction problem(SAT) or the partition problem (PAR) can be solved more efficiently on a quantum computer. The role of interference, measurement, and unitary evolution are examined as possible routes to an efficient quantum algorithm.

1. Introduction

There is a building called the theory of computation. Built during the past hundred years this building houses rooms filled with the wonders of Turing machines, Gödelian conjectures, and amazingly complex algorithms. Like any building, however, these rooms are only as sturdy as the foundation upon which the whole structure rests. Since its construction a hundred years ago, the residents of the building of computation (who we call computer scientists) have routinely inspected the foundation of their building and found the foundation to be utterly solid and immovable. Recently, much to the computer scientist's dismay, a group of visitors from another building, calling themselves physicists, have begun to crawl around in the basement of the computation building and, to put it bluntly, been getting in everyone's way. From these scruffy visitors muted mumblings about "quantum mechanics" and "the superposition principle" nothing short of a rebellion among the inhabitants of the computation building has ensued. "Your foundation is ethereal," the visitors claim, "nothing more than a false perception of a secure foundation." "The very foundation you have built upon, the classical computer," the visitors dare to utter in blasphemy, "suffices only as a crude approximation to a truly secure foundation, the quantum computer."

What these physicists have stumbled upon is the idea[Bennett 1993, Bron 1994, Deutsch 1985, Jozsa 1991] that since computation is inherently a physical process, the basis of computational theory must be derived from fundamental physical laws. Since the world in which the founders of computational theory operated behaved in an essentially classical manner it was easy for these founders to base their theory on *intuitive* classical assumptions. Computational theory has flourished under this perspective due to the fact that every computational experiment–all modern day computers–have remained in the classical realm. However, as computers grow smaller and smaller they will eventually reach the domain where classical physics is no longer a satisfactory physical theory and instead the computers must obey the laws of quantum mechanics. Due to this paradigm shift, the basic principles of computational theory need to be revised. In effect, computational theory must come to grips with a new concept: the quantum computer.

2. What is a Quantum Computer?

Both the deterministic and probabilistic classical computer evolve along pathways of different measurable states. At each step in the computation of these classical computers the computer is completely specified by one definite state which can, in theory, be measured and symbolically displayed to an experimenter. A deterministic classical computer evolves from state to state in a fashion uniquely determined by the initial state of the computer and subsequent input while a probabilistic classical computer, although still partially determined by external input, progresses from the state to state in a probabilistic manner. At any given instant in time a classical computer is in a state which is completely specified by some set of measurable variables which have been arrived at by some deterministic algorithm. Measurement of these variables always tells us the exact state of the computer at a given time.

A quantum computer is completely specified, in analogy with the state of a classical computer, by its wave function. Like a classical computer, a quantum computer has a set of variables which can be measured to draw conclusions as to the state of the system. However, unlike a classical computer, the state of a quantum system is not necessarily completely specified by a measurement of these variables. In other words, what one measures the state of the system to be is not the wave function of the system. The wave function of the computer can be pictured at a given time t as existing in a superposition of the different possible measurable states

$$|\psi(t)>= \sum_i a_i |state_i>$$

where the coefficients a_i are complex numbers whose norm squared, $|a_i|^2$, represent the probability that a measurement of computer's variables yield the variables which represent $|state_i>$. Upon measurement of the state $|state_i>$ the wave function immediately collapses to that state:

$$|\psi>= |state_i> .$$

When, on the other hand, a measurement of the system is not being made, the evolution of the wave function is completely specified (deterministic) by an unitary evolution operator. Thus, like a classical computer the evolution of the wave function is completely specified by an algorithm whenever a measurement is not being made. Because the evolution of the wave function is unitary, every operation on a quantum computer besides measurement, is a reversible operation. (However, see experiments[Scully and Druhl 1982, Englert *et al.* 1994], such as the quantum erasure experiment, for the possibility that measurement is indeed reversible.)

Thus while a classical computer evolves along a definite singular path a quantum computer has the ability to go through many different paths on the way to a final definite or probabilistic state.

Here it should be noted that a quantum computer can be made perhaps even more powerful by, in direct analogy with a classical probabilistic computer, allowing the deterministic evolution (the algorithm) to be probabilistic. We will not consider this possibility in this paper but will only note that any such algorithm can be simulated by a by a completely quantum system using measurements on the quantum system to determine the algorithm of the quantum computer.

3. The Quantum Computer Explicitly Specified

A quantum computer $\mathcal{Q}$ is composed of a set of registers $\mathcal{R}$ which correspond to physically measurable quantities r. These registers can correspond to any physical observable: the spin of an electron or the energy level of an atom, for example. For simplicity we will take these registers to be n-bit registers so that each bit of a register can be pictured as corresponding one level in a two state system. Throughout we will designate these states by $|0>$ and $|1>$ for 1-bit registers, and by $|x>$ where x is a positive integer for n-bit registers.

At any given time t, the state of $\mathcal{Q}$ is completely specified by a wave function $|\Psi(t)>$ which acts on a Hilbert space corresponding to the registers $\mathcal{R}$. The evolution of the system is physically governed by the Schrödinger equation. We will assume that any possible unitary Hamiltonian is physically possible and thus we will allow evolution via the Schordinger equation to be represented by any unitary transformation. On the most basic level we will take the single bit matrices

$$P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\imath\phi} \end{bmatrix}$$

$$N = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$H(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix}$$

and the multiple bit matrix

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

to be the basic building blocks upon which a quantum computer is to be built.

In order to receive input from a user into a quantum computer we can prepare any of the registers $\mathcal{R}$ to be in a certain user specified state by measuring the register and then rotating the Hilbert space of that register so that the user input is specified on that register.

Output of a quantum computer simply corresponds to measurement of one of the registers $\mathcal{R}$ and recording this measurement on some microscopic storage or output device.

4. Why Are Quantum Computers Special?

There are two major differences between the quantum theory of computation and the classical theory of computation: the superposition principle and interference.

4.1. Obtaining an Exponential Superposition in Polynomial Time

The Hadamard transformation of order 2,

$$H = H\left(\frac{\pi}{2}\right) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

is one of the most useful matrices in quantum computation. This transformation, which has no classical analogy, can be used to create and destroy superpositions. Thus, for instance, if a 1-bit register is in the $|0>$ state, the action of this unitary transformation put the system in an in-phase superposition of the $|0>$ and $|1>$ states:

$$H|0>= \frac{1}{\sqrt{2}}(|0> +|1>).$$

Further, if we apply this transformation to every bit in an n-bit register we will obtain a superposition of states.

$$H_1 H_2 ... H_n |0, 0, 0, \cdots, 0>= H|0>= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k> .$$

The beauty of quantum computers, we immediately notice, is that we have obtained a superpostion of an exponential number of states (2^n) in a polynomial number of steps (n). Thus we see that a quantum computer can deal with an exponentially large data set without using an exponential number of operations.

Suppose further that we have a quantum construction which computes in s steps a function f(x):

$$U|x, 0>= |x, f(x)> .$$

If we apply this function to the above exponential superposition of states,

$$U \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k, 0>= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k, f(k)>$$

we can computer f(k) an exponential number of times on the exponential superposition using only s steps. This massive parallel computing without an exponential overhead has no known classical analogy.

4.2. Inference

The second aspect of a quantum computer which separates it from a classical computer is interference. In a classical probabilistic algorithm the probability that a certain state is reached is the sum of all of the different probabilities of each of the different pathways which lead to the certain state. Thus

$$P_i^{classical} = \sum_j P_{j,i}$$

where $P_{j,i}$ is the probability of reaching state i via path j.

Quantum computers, on the other hand, do not follow the same rules of probability that classical probabilistic computers follow. For a quantum computer, the probability that a certain state is reached is the sum of all of the different probability *amplitudes* of the different pathways *squared*. These probability amplitudes are the main difference between quantum and classical systems. Further we should note that when we talk about probabilities in quantum mechanics what we mean is the probability *if* we measure the system. Thus,

$$P_i^{quantum} = \left| \sum_j a_{j,i} \right|^2 ,$$

where $a_{j,i}$ is the probability amplitude of reaching the state i via the state j. Because probabilities in quantum mechanics are probability amplitudes squared, probability amplitudes can be negative. This in turn leads to the fact that the above sum can have terms which cancel and hence interfere with each other.

Consider for example the following algorithm

1. Start in the 1-bit state $|0>$.
2. Apply $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ to the state to obtain a superposition.
3. Compute U, a user specified function,

$$U|0>= f_0|0>$$

$$U|1>= f_1|1>$$

where f_0 and f_1 are either ± 1 depending on the nature of U.

4. Apply $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ to the state once again.
5. Measure the bit.

We therefore see that if $f_0 = f_1$ then at step five we will measure $|0>$ and if $f_0 = -f_1$ then we will measure $|1>$. Thus this algorithm can use interference to distinguish between two different types of user inputed functions U. Algebraically the algorithm proceeds along the following path

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

$$U\frac{1}{\sqrt{2}}(|0> + |1>) = \frac{1}{\sqrt{2}}(f_0|0> + f_1|1>)$$

$$H\frac{1}{\sqrt{2}}(f_0|0> + f_1|1>) = \frac{f_0 + f_1}{2}|0> + \frac{f_0 + f_1}{2}|1> .$$

Interference on a grander scale has been demonstrated recently in Shor's algorithm for factoring numbers[Shor 1994]. In Shor's algorithm, a fast fourier transform serves to create interference between different periodic states in order to find the frequency of a given function.

5. The First Quantum Myth

Evolution of a quantum computer via the Schrödinger equation is unitary. Thus as long as the Schrödinger equation governs the quantum computer the computer will be a reversible norm-conserving system. However, when a measurement is made on a quantum computer the system appears to collapse in a completely non-unitary manner. Thus it is often misunderstood that the evolution of a quantum computer must be strictly unitary with the only measurement occurring at the final stage of the computation. We will see that this view holds both and ounce of truth and and ounce of fallacy.

First of all we will establish that a measurement which occurs part way through the evolution of the system can serve to make the overall evolution of the computer non-unitary. Suppose that we wish to construct the following transformation:

$$|k, n>\rightarrow |k \cdot n, n>$$

where $|k>$ and $|n>$ are 1-bit registers. This transformation is given by

k	k	$k \cdot n$	n
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	1

This unitary transformation is clearly non-unitary. It can be made unitary by adding an extra *junk* bit,

k	n	c	$k \cdot n$	n	$c = k$
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	0	0	1
1	1	0	1	1	1

and starting the system in in the state $|k, n, 0>$. However, even though the transformation is still unitary, the extra bit is now a hinderence. Basically since the extra bit is either 0 or 1 and not one or the other it will be a hinderence to other operations. In other words, the $|k, n> \to |k \cdot n, n>$ transformation is essentially different from the $|k, n, 0> \to |k \cdot n, n, k>$ transformation because the $|k, n> \to |k \cdot n, n>$ sub-product cannot be used to interfere with each other in the later transformation. We must always recall that any left-over junk bits which are entangled with the input superposition in such a manner that they cannot be written as a tensor product with the whole superposition will disallow later interference.

In order to rescue ourselves from this dilemma, we call upon the probabilistic and non-unitary nature of measurement. If we construct a unitary transformation:

$$|0,0,0> \to \frac{1}{\sqrt{2}}(|0,0,0> + |0,0,1>)$$

$$|0,1,0> \to \frac{1}{\sqrt{2}}(|0,1,0> + |0,1,1>)$$

$$|1,0,0> \to \frac{1}{\sqrt{2}}(|0,0,0> - |0,0,1>)$$

$$|1,1,0> \to \frac{1}{\sqrt{2}}(|1,1,0> + |1,1,1>)$$

then we see that after the transformation if we start with $|k, n, 0>$ we are in the state $\frac{1}{\sqrt{2}}(|k \cdot n, n, 0> + |junk, junk, 1>)$. Thus if we measure the final register and obtain $|0>$ then we have achieved the desired transformation without obtaining any incorrectly entangled junk bits. If, on the other hand, we measure $|1>$ for the final register then the first two registers contain values which contain the correct result with the exception of a phase flip. Notice, however, this phase shift cannot be reversed because the phase shift is between the two identical $|0, 0, 1>$ states. This fact renders the state unusable via the fact that we cannot reconstruct the initial state. Half of the time, however, we will obtain a $|0>$ and thus complete the correct non-unitary transformation on the bits which concern us.

Let us define N as the number of operations need to obtain the state at which we wish to apply the non-unitary nature of $|k, n> \rightarrow |k \cdot n, n>$. Then if we measure $|1>$ in the final bit of our probabilistic routine we can measure the first two registers, return them to their original state and go through the N operations needed to return the system to the $|k, n, 0>$ state and again apply the above transformation. If we apply this procedure s times, we will obtain a zero in the last register at least once with probability $1 - \frac{1}{2}^s$ which goes to 1 exponentially fast in s. This will take a total of Ns operations. Hence as long as N and s are polynomials of low enough order we can obtain a non-unitary transformation within polynomial time.

The drawback to this procedure is that one must repeat all of the previous procedure which has led up to a precarious superposition s times in order to obtain a decent probability of having a successful algorithm. Thus if this procedure is applied $O(n)$ times, where n is the size parameter of the problem, we will run into an exponential slowdown.

6. The Second Quantum Myth

The second quantum myth relates to the act of measurement. In the previous section we noted that a midway measurement could drastically effect the evolution of a quantum computer. We will show, however, that this or any measurement could be put off until the end of the computation.

In order to accomplish this feat we must construct "conditional" operators. A conditional operator acts on one set of registers conditional on another set of separate registers, i.e.

$$U_c|a_1, \cdots, a_x, b_1, \cdots, b_y, c_1, \cdots, c_z >= |a_1, \cdots, a_x, f_1, \cdots, f_y, c_1, \cdots, c_z >$$

where the a_i's are the conditional registers which determine what operations will be performed on the b_i's, the c_i's are the unaffected registers, and the f_i's are dependent on the a_i's and b_i's and represent the conditional operation on the b_i's. Thus,

$$f_i(a_1, a_2, \cdots, a_x, b_1, b_2, \cdots, b_y) = U(a_1, a_2, \cdots, a_x)|b_1, b_2, \cdots, b_y >$$

for some unitary transformation U.

An example of a conditional operator is the exclusive-or (XOR) of two bits.

x	y	x	$x \oplus y$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

If we treat the first register as the conditional register then we find that the second registers is acted on by

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

if $x = 0$, or by

$$N = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

if $x = 1$. Thus XOR is often referred to as a conditional NOT.

Now, since we can always construct a conditional matrix we will now show that this implies that we can put a measurement off until the end of the computation. In order to achieve this we make the following substitutions into the original (midway measurement allowed) algorithm:

1. Whenever there is a measurement replace this measurement by a copy of the measured register to a new "copy" register, i.e.

$$C|a_i, 0 >= |a_i, a_i > .$$

2. Replace all actions on conditional classical information by conditional operations (quantum mechanical) on the quantum copied bits.

3. Measure the copy registers as the final step in the computation

Using this procedure it is clear that the evolution of the system will proceed in the same manner as if there had been a measurement. This is because quantum information is as robust as classical information (This from both the correspondence principle and the fact that classical bits can be simulated, disregarding non-unitary evolution for the moment, by orthogonal sets of quantum states). Thus midway measurements can be put off until a later point in the computation.

7. The Satisfaction Problem

One of the most often sited NP-complete problems[Garey and Johnson 1975] is the satisfaction problem (SAT). NP-complete problems are a set of problems which have no known classical polynomial time solution but can all be mapped onto each other in a polynomial amount of time. We will examine a specific strain of the SAT problem, the boolean-SAT problem. The boolean SAT problem can be stated as:

Given: A boolean expression $f(x_1, x_2, \cdots, x_n)$ where $x_i \in \{0, 1\}$

Question: Does there exist a combination $\vec{x} = (x_1, x_2, \cdots, x_n)$ such that $f(\vec{x}) = 1$?

This problem is difficult to answer classically because one must search almost the entire phase space of $(x_1, x_2, \cdots, x_n)$, a total of 2^n possible inputs, in order to find if there exists a $\vec{x}$ such that $f(\vec{x}) = 1$.

It can be shown[Garey and Johnson 1975] that any boolean function $f(\vec{x})$ can be written in what is called 3-CNF form (3-conjunctive normal form) in polynomial (in n) steps. 3-CNF expressions are simply the product (ANDs) of 3 term additions (ORs) with negations on any of the terms. Thus, for example,

$$f(x_1, x_2, x_3, x_4) = (x_1 + \bar{x_2} + x_3) \cdot (x_1 + \bar{x_3} + \bar{x_4})$$

is a 3-CNF expression. Since any boolean function is transformable in polynomial time into a 3-CNF function, the 3-CNF boolean SAT problem is also NP-complete.

8. SAT-Brute Force Failure

It would appear that most obvious way to solve the SAT problem on a quantum computer would be the following procedure:

Given: State $|0,0>$ with an n-bit first register and a 1-bit second register.

Algorithm:

1. Apply the Hadamard matrix H to each bit in the first register to obtain an exponential superposition

$$|\Psi>= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |n,0>$$

2. Apply U_f to the state. U_f is constructed from classical gates and is given by

$$U_f|n,0>=|n,f(n)> .$$

Here we recall that any classical algorithm can be implemented on a quantum computer. Hence the state is now

$$|\Psi>= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k,f(k)>$$

At this point we could naively claim that we are done. We have computed $f(\vec{x})$ for all possible $\vec{x}$'s. The problem, of course, is in obtaining information from this superposition. Suppose that there existed only one $\vec{x}$ such that $f(\vec{x})=1$. We then see that if we measure the last register the probability that we measure $|1>$ is

$$P = \left|\frac{1}{\sqrt{2^n}}\right|^2 = \frac{1}{2^n}$$

which is exponentially small in the input size n. Thus, the naive, brute force approach does not yield an acceptable solution.

Notice, however, that this approach is better than a straight search over the phase space of $\vec{x}$ but is also only as good as a straight probabilistic search. This is to say that the probability of our success is not dependent on the location of the $f(\vec{x})=1$ solutions.

9. SAT-a Second Approach

Upon examining the brute force method, it becomes clear that one possible remedy for solving the problem of an exponentially small chance of measuring an $f = 1$ if it exists, is to somehow exponentially multiply the number of $f = 1$'s in the superposition. One way to do this would be the following transformation:

$$\frac{1}{\sqrt{2}}(|0,0> +|1,0>) \to \frac{1}{\sqrt{2}}(|0> +|1>) \otimes |0>$$

$$\frac{1}{\sqrt{2}}(|0,0> +|1,1>) \to \frac{1}{\sqrt{2}}(|0> +|1>) \otimes |1>$$

$$\frac{1}{\sqrt{2}}(|0,1> +|1,0>) \to \frac{1}{\sqrt{2}}(|0> +|1>) \otimes |1>$$

$$\frac{1}{\sqrt{2}}(|0,1> +|1,1>) \to \frac{1}{\sqrt{2}}(|0> +|1>) \otimes |1>$$

acting on the i^{th} and last bit. This transformation, acting on all n-bits of the first register would serve to multiply a single $f = 1$ solution exponentially. We see, unfortunately, that this transformation is, in fact, non-unitary. Further, it can be shown that in order to use the non-unitary evolution described above, the algorithm must take an exponential amount of time.

Here we are running up against a fundamental barrier. If there is no solutions, the brute force algorithm will yield

$$|\Psi_0 >= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k,0> .$$

If on the other hand there is one $\vec{x}$ such that $f = 1$, the brute force algorithm will yield

$$|\Psi_1 >= \frac{1}{\sqrt{2^n}} \sum_{k=0,k\neq x}^{2^n-1} |k,0> + \frac{1}{\sqrt{2^n}}|x,1> .$$

The overlap between these function is

$$| < \Psi_1|\Psi_0 > | = 1 - \frac{1}{\sqrt{2^n}}$$

and thus these states can only be distinguished with an exponential number of operations.

10. SAT-Reverse Engineering?

Since we can write any boolean function in 3-CNF form we can force each clause in the 3-CNF form to be separately 1 in order that $f = 1$. For example if

$$f(x_1, x_2, x_3, x_4) = (x_1 + \bar{x_2} + x_3) \cdot (x_1 + \bar{x_3} + \bar{x_4})$$

then we can restate the SAT problem for this function as determining if we can simultaneously set

$$(x_1 + \bar{x_2} + x_3) = 1$$

and

$$(x_1 + \bar{x_3} + \bar{x_4}) = 1.$$

Using a quantum computer we can construct gates which "uncompute" each of these clauses and leaves the system in a state which is a superposition of the three possible x_i's which satisfy each clause. In order to see if there exists an $f = 1$ solution we must then only compare each like term in these clauses to see if they are consistent with each other.

The transformation which does this consistency checking is

$$|0,0> \rightarrow |1,0>$$

$$|0,1> \rightarrow |0,0>$$

$$|1,0> \rightarrow |0,1>$$

$$|1,1> \rightarrow |1,1>$$

$$\frac{1}{\sqrt{2}}(|0,0> + |1,0>) \rightarrow |1,0>$$

$$\frac{1}{\sqrt{2}}(|0,1> + |1,1>) \rightarrow |1,1>$$

$$\frac{1}{\sqrt{2}}(|0,0> + |0,1>) \rightarrow |1,0>$$

$$\frac{1}{\sqrt{2}}(|1,0> + |1,1>) \rightarrow |1,1>$$

$$\frac{1}{2}(|0,0> + |0,1> + |1,0> + |1,1>) \rightarrow \frac{1}{\sqrt{2}}(|1,0> + |1,1>).$$

We immediately note that this transformation is non-unitary and that it cannot be saved by the non-unitary probabilistic algorithm detailed above. Thus we once again run up against non-unitary evolution. Here we see that, while the overlap between these different states is at most $\frac{1}{4}$, will will have to apply this transformation $O(n)$ times so that, since we will have to repeat each measurement if we get an incorrect answer, there will be an exponential effective overlap.

11. What Has Gone Wrong?

There are three major points, at this point, which we see are holding us back from discovering a quantum algorithm to solve the SAT problem:

1. While we have used an exponential superposition we could not extract specific data. This problem is intrinsic to the way in which quantum computers work and no way of getting around this appears obvious.
2. We have not used interference. Interference is the one aspect of quantum computers which differs radically from any classical concept. Conceptually we are looking for an algorithm which will allow us to use phase shifts to destroy, via interference, $f = 0$ pathways.
3. The evolution we needed was non-unitary. In most cases this comes about because the overlap between the states we are examining is exponentially near 1. As in 1, there appears to be no obvious way to get around this.

Examining this list, one possible path to examine is to look for a problem which requires the manipulation of an exponential data structure but whose answers depend on a structure in that data set.

12. The Parition Problem

One possible problem which has this characteristic is the Partition problem (PAR)[Garey and Johnson 1975]. The PAR problem can be stated as

PAR: Given a set of numbers $a_i \in A$, does there exist a set $S \subset A$ such that

$$\sum_{a_i \in S} a_i = \sum_{a_i \notin S} a_i ?$$

We first of all note that we can easily computer $L = \sum_{a_i \in A} a_i$. If L is odd we immediately know that the answer to PAR is no. If on the other hand L is even then this question is equivalent to the question of whether there exists a subset of A such that the sum of the elements in that subset is $L/2$.

We note that in order to solve PAR we can construct a tree which contains every possible combination of adding the a_i's and ask if $L/2$ is in this tree. Thus the first node in this tree is 0 and on each new level in the tree each node splits into two: one with the parent's value and the other with the parent's value plus a_l where l is the level of the parent.

In terms of quantum mechanics we wish to achieve the transformation

$$U(a_i)|x >= \frac{1}{\sqrt{2}}(|x > +|x + a_i >)$$

however we see that this transformation is not unitary. One way to get around this is to use the following transformation:

$$U(a_i)|x, 0 >= \frac{1}{\sqrt{2}}(|x, x > +|x, x + a_i >).$$

If we start with the $(n+1)$-register state $|0, 0, \cdots, 0 >$ and apply $U(a_i)$ to the $i-1$ and i registers, then the final result will be a superposition which holds all possible ways to add the a_i's in the final register. Thus we need only ask if there exists a $L/2$ in this superpostition in order to solve PAR.

It turns out, however, that there appears to be no way to determine this in a polynomial amount of time. The reason for this is that the in the above algorithm the final register is deeply entangled with its history in the other n registers. Thus it appears that once again we need a non-unitary transformation to solve the problem.

13. The Future

The future of solving NP-complete problems on a quantum computer clearly lies in the hands of interference. Shor's algorithm, which factors numbers in polynomial time, is successful because it can use interference. All of the straightforward algorithms above failed to make use of interference and it does not appear that an algorithm which does not use interference can behave better than a probabilistic classical computer. In fact, because a probabilistic classical computer is just a quantum computer with midway measurements, it is obvious that the only difference between a quantum computer and classical computer will occur because of interference. Thus any future researcher attempting to solve NP-complete problems on a quantum computer should attempt to use interference. Along these lines it appears that the PAR problem is more suitable to such an algorithm because of its natural tree-like structure.

14. Conclusion

We have examined quantum computers and how they differ from classical computers. Two myths about quantum computers are examined and explained. The issue of whether a quantum computer can solve an NP-complete problem is specifically addressed. Three ways to solve the satisfaction problem are proposed and each of these is seen to fail for a different reason. An approach to solve the partition problem is likewise shut down. Future approaches to solving NP-complete problems on a quantum computer are addressed.

15. Acknowledgments

This research was supported by funds from the Summer Undergraduate Research Fellowship (SURF) Program at Caltech. We are especially grateful to the SURF donors, John Glanville and Teri Oldknow, for allowing us the opportunity to pursue this exciting research. Finally, we would like to thank our advisors for this SURF research, Chris Adami and Nicholas Cerf, for there excitement and encouragement throughout this project.

REFERENCES

Bennett, C. H. 1993, Nature, 362(22), 694-695

Bron, J. 1994, New Scientist, 9, 21-24

Deutsch, D. 1985, Proc. R. Soc. Lond. A, 400, 97-117

DiVincenzo, D. P. 1995, Science, 270, 255-261

Englert, B. G., Scully, M. O., and Walther, H. 1994, Sci. Am., 271(6), 86-92

Garey, M. R. and Johnson, D. S. 1979, A Guide to the Theory of NP-Completeness (San Fransisco: W. H. Freeman)

Hog, T. 1995, LANL e-print, quant-ph/9508012

Jozsa, R. 1991, Proc. R. Soc. Lond. A, 435, 563-574

Scully, M. O. and Druhl, K. 1982, Phys. Rev. A, 25, 2208

Shor, P. W. 1994, Proceedings,35th Annual Symposium on Fundamentals of Computer Science, 124-134