Quantum computing in continuous time

Andrew Childs*
Center for Theoretical Physics
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
(25 September 2001)

I. INTRODUCTION

This lecture is about an alternative paradigm for the description and design of quantum algorithms. In particular, I will discuss algorithms that have a continuous description in time. First, I will describe how the quantum search problem can be viewed in this way. I will also present a general technique for solving computational problems on a quantum computer based on quantum adiabatic evolution.

The conventional way of describing a quantum algorithm is to specify a sequence of unitary transformations, each acting on one or two qubits at a time. In this description, time is discretized into steps, with one gate applied per step. This is a convenient way to think to think about algorithms, and indeed, nearly all of the work on quantum algorithms so far has been done using this picture. But it is not the only possible picture, and we might expect to gain some insight by looking at things in a different way.

In quantum mechanics, the time evolution of the state vector $|\psi(t)\rangle$ is given by the Schrödinger equation,

$$i\hbar\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = H(t)|\psi(t)\rangle\,. \tag{1}$$

For this rest of this lecture, I will set $\hbar = 1$ for convenience. This equation says that time evolution is generated by an operator $H(t)$, which is called the *Hamiltonian* of the system. The spectrum of $H(t)$ gives the energy levels of the system at time t. Since energy is a real quantity, $H(t)$ must be a Hermitian operator. This in turn implies that the time evolution is unitary. In a physical quantum computer, the unitary gates that make up the computation are nothing more than solutions of this equation over a certain period of time. But instead of thinking about the transformations induced by particular unitary operators, we are free to think about continuous time evolution in terms of the Hamiltonian $H(t)$. Indeed, since real physical systems evolve continuously in time, we might expect this to be advantageous in some cases.

What does the time evolution look like? The Schrödinger equation is a linear, first-order differential equation. Thus, if we specify the state of the system $|\psi(0)\rangle$ at $t = 0$ (or indeed at any time), this fully specifies the state of the system at all other times. If $H(t)$ is actually *independent* of time, then the solution is particularly simple:

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle\,. \tag{2}$$

You might think that continuous-time quantum computation is simply analog quantum computation. Indeed, one of the first papers on this subject was titled *An analog analogue of*

---

*`amchilds@mit.edu`

Architecture	States	Dynamics	Measurement
Conventional classical computer	D	D	D
Conventional quantum computer	C	D	D
Continuous time quantum computer	C	C	D
Classical or quantum analog computer	C	C	C

TABLE I: Degrees of analogness. D=discrete, C=continuous.

a digital quantum computation. Classically, an analog computer is something that makes use of continuously varying parameters to solve a computational problem. A slide rule is a good example of an classical analog computer. But in the quantum case, there are actually several different ways in which a computer can be analog. First of all, since quantum mechanics takes place in a vector space, the states are *always* capable of taking continuous values, even if the space is finite-dimensional. However, in a te-dimensional. However, in a finite-dimensional Hilbert space, a particular measurement can result in only a finite number of different outcomes, so the states are in some since discretized upon measurement. In both cases, we are free to describe the dynamics either in discrete or a continuous terms. So a quantum computer is not necessarily either analog or digital, but can have aspects of both. Some of the possible "degrees of analogness" are summarized in Table I.

II. THE GROVER PROBLEM

The first example I will give is a continuous-time version of Grover's algorithm for quantum search. The Grover problem is as follows: you are given a function $f(x)$ that takes the values 0 and 1 on integers $x \in \{0, 1, \ldots, N-1\}$. On $N-1$ values of x, $f(x) = 0$, but on one special value w (the *winner*), it takes the value $f(w) = 1$. The problem is to determine w as efficiently as possible.

Classically, it is easy to see that on average, this problem will require us to query the function f on half of its inputs. The best possible algorithm is to simply try all of the possible inputs in turn. The running time of this algorithm is $O(N)$. But in the quantum case, it is actually possible to solve this problem using only $O(\sqrt{N})$ queries of f. Grover's original description of his algorithm used a discrete sequence of gates. However, the same essential idea has a convenient description in terms of continuous time evolution.

The algorithm is described by a time-independent Hamiltonian. The winner will be specified through a given Hamiltonian

$$H_w = |w\rangle\langle w| \,. \tag{3}$$

In other words, the state $|w\rangle$ is distinguished by having energy one, whereas all other states have energy zero. We know that the Hamiltonian is of this form, but we do not know the value of w, and we would like to find it. It may help to think of H_w as a "black box Hamiltonian": we can apply it to quantum states, but we cannot look inside the box.

The algorithm will work by preparing the system in some standard state (independent of w), allowing the system to evolve, and making a measurement that will tell us the value of

w. A sensible w-independent starting state is

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{z=0}^{N-1} |z\rangle \,, \tag{4}$$

the uniform superposition of all possible answers. But if we just apply the Hamiltonian H_w to this starting state, the probability that we measure w will not change with time. Instead, we will add an additional *driving Hamiltonian*

$$H_D = |s\rangle\langle s| \tag{5}$$

so that the total Hamiltonian is

$$H = H_w + H_D = |w\rangle\langle w| + |s\rangle\langle s| \,. \tag{6}$$

Now we have to find the probability of finding $|w\rangle$ as a function of time. Note that there are only two states involved in this problem, $|w\rangle$ and $|s\rangle$. These span a two-dimensional space, which means the calculation will be particularly simple. But $|w\rangle$ and $|s\rangle$ are not orthogonal; they have inner product

$$\langle w|s\rangle = \frac{1}{\sqrt{N}} = a \,. \tag{7}$$

To get an orthonormal basis, we can use Gram-Schmidt orthonormalization, which just amounts to removing the part of $|s\rangle$ along $|w\rangle$ and normalizing:

$$|r\rangle = \frac{1}{\sqrt{1-a^2}}(|s\rangle - a|w\rangle) \,. \tag{8}$$

In terms of this state, we have

$$|s\rangle = a|w\rangle + \sqrt{1-a^2}|r\rangle \,, \tag{9}$$

so the Hamiltonian is

$$H = \begin{bmatrix} 1+a^2 & a\sqrt{1-a^2} \\ a\sqrt{1-a^2} & 1-a^2 \end{bmatrix} = I + a(\sqrt{1-a^2}X + aZ) \,. \tag{10}$$

Note that the squares of the numbers multiplying X and Z sum to one. The state as a function of time is

$$\begin{aligned} |\psi(t)\rangle &= e^{-iHt}|s\rangle && (11) \\ &= \cos at|s\rangle - i\sin at(\sqrt{1-a^2}X + aZ)(a|w\rangle + \sqrt{1-a^2}|r\rangle) && (12) \\ &= \cos at|s\rangle - i\sin at(a\sqrt{1-a^2}|r\rangle + (1-a^2)|w\rangle + a^2|w\rangle - a\sqrt{1-a^2}|r\rangle) && (13) \\ &= \cos at|s\rangle - i\sin at|w\rangle && (14) \end{aligned}$$

(up to an overall phase). At a time given by

$$at = \frac{\pi}{2} \quad \Rightarrow \quad t = \frac{\pi}{2a} = \frac{\pi\sqrt{N}}{2} \,, \tag{15}$$

the state is $|w\rangle$, so a measurement of the state gives the correct answer. Note that the time is $O(\sqrt{N})$, just as in Grover's algorithm, and that the success probability is periodic in time.

One of the nice things about the Grover problem is that it is easy to show that Grover's algorithm is optimal. It is not possible to find w in fewer than $O(\sqrt{N})$ calls to f. Similarly, it is easy to show that the continuous-time search algorithm just described cannot find w in time less than $O(\sqrt{N})$.

The proof goes as follows. We will allow a completely arbitrary (possibly time-dependent) driving Hamiltonian $H_D(t)$ and w-independent starting state $|\psi(0)\rangle$. We will denote the time evolution under $H_w + H_D(t)$ by $|\psi_w(t)\rangle$, and under $H_D(t)$ alone by $|\phi(t)\rangle$. We will bound the rate at which the two states $|\psi_w\rangle$ and $|\phi\rangle$ can separate. Since $|\phi\rangle$ is independent of w, the algorithm cannot be successful unless $|\psi_w\rangle$ is very different from $|\phi\rangle$.

We will measure the distance between $|\psi_w\rangle$ and $|\phi\rangle$ using

$$d_w = \Big\| |\psi_w\rangle - |\phi\rangle \Big\|^2 = 2(1 - \mathrm{Re}\langle\psi_w|\phi\rangle)\,. \tag{16}$$

To be able to determine w with a reasonable probability of success, we demand

$$d_w \geq \epsilon \tag{17}$$

for some fixed $\epsilon > 0$. Since we require the algorithm to work for all inputs, this implies

$$\sum_w d_w \geq N\epsilon\,. \tag{18}$$

Now

$$\begin{aligned} \frac{\mathrm{d}d_w}{\mathrm{d}t} &= 2\,\mathrm{Im}\langle\psi_w|H_w|\phi\rangle && (19)\\ &\leq 2|\langle\psi_w|H_w|\phi\rangle| && (20)\\ &\leq 2\Big\| H_w|\phi\rangle \Big\| && (21)\\ &= 2|\langle w|\phi\rangle|\,, && (22) \end{aligned}$$

where the second inequality follows from the Cauchy-Schwartz inequality,

$$\vec{a}\cdot\vec{b} \leq |\vec{a}||\vec{b}|\,. \tag{23}$$

Summing on w gives

$$\begin{aligned} \frac{\mathrm{d}\sum_w d_w}{\mathrm{d}t} &\leq 2\sum_w |\langle w|\phi\rangle| && (24)\\ &\leq 2\sqrt{N}\,, && (25) \end{aligned}$$

where we have used the Cauchy-Schwartz inequality again. Integrating this inequality with the initial condition $d_w = 0$ at $t = 0$, we find

$$\sum_w d_w \leq 2\sqrt{N}t\,, \tag{26}$$

which gives

$$t \geq \frac{\epsilon}{2}\sqrt{N}\,. \tag{27}$$

Thus the running time of the algorithm must be $O(\sqrt{N})$.

Of course, this continuous-time algorithm is not very useful in practice unless we have a physical system whose Hamiltonian is $|s\rangle\langle s|$. I do not know of such a system. However, it is possible to *simulate* this Hamiltonian on a conventional quantum computer. But this runs counter to our present goal of thinking about algorithms in continuous time, so we will be content with the insight provided by the above algorithm.

III. ADIABATIC COMPUTATION

Our second example is known as quantum computation by adiabatic evolution. This is an example of a problem that has a very natural description in terms of a Hamiltonian, but is not so easy to describe in the circuit model of computation. It has the advantage that it is easy to formulate for a very wide class of computational problems, even very hard ones. However, it has the disadvantage that it is difficult to determine how efficient the algorithm actually is. Indeed, no one has yet discovered a problem for which the adiabatic algorithm is faster than classical methods, except for the Grover problem discussed in the previous section.

Adiabatic computation can be used to solve any problem that can be described as minimizing a function of n bits. Let $z = (z_0, z_1, \ldots, z_{n-1})$ be an n-bit number, and suppose we would like to minimize a function

$$h(z) : \{0,1\}^n \to \mathbb{R}\,. \tag{28}$$

We can associate with this function the *problem Hamiltonian*

$$H_P = \sum_{z=0}^{2^n-1} h(z)|z\rangle\langle z|\,, \tag{29}$$

which is diagonal in the computational basis and has the possible values of $h(z)$ as its eigenvalues. Thus the minimum of $h(z)$ corresponds to the ground state of H_P. In other words, we have turned the problem of minimizing a function of bits into the problem of finding the ground state of a Hamiltonian.

The motivation for computation by adiabatic evolution comes from a theorem about quantum mechanics called the quantum adiabatic theorem. This theorem basically states that if a system's Hamiltonian changes very slowly, a system that starts out in the ground state will remain in the ground state. Somewhat more precisely, the adiabatic theorem is as follows: let $\tilde{H}(s)$ be a smoothly varying Hamiltonian such that

$$\tilde{H}(0) = H_B\,, \quad \tilde{H}(1) = H_P\,. \tag{30}$$

Let $|\psi(0)\rangle$ be the ground state of H_B. Evolve with the Hamiltonian $H(t) = \tilde{H}(t/T)$. Then in the limit $T \to \infty$, $|\psi(T)\rangle$ is the ground state of H_P.

The adiabatic theorem can immediately be turned into a continuous-time quantum algorithm. Let H_B be any Hamiltonian whose ground state is easy to construct, and prepare a

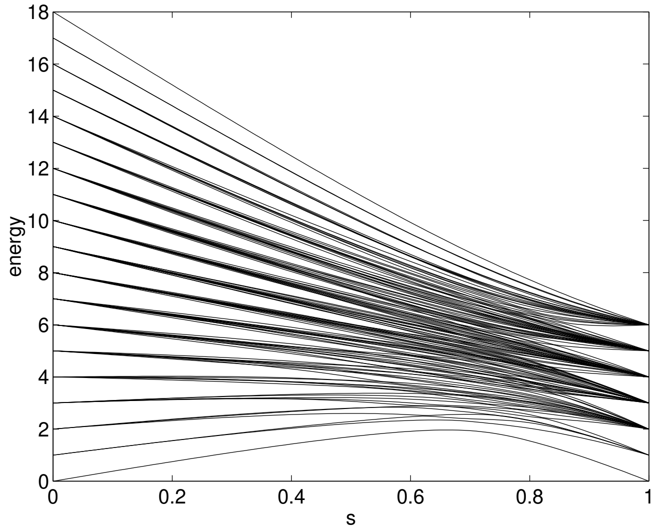

FIG. 1: Spectrum of $\tilde{H}(s)$ for a particular seven-bit instance of Exact Cover, a computational problem one might try to solve by the adiabatic algorithm.

system is this state. Smoothly change the Hamiltonian form H_B to H_P in any way you like — for example, let

$$\tilde{H}(s) = (1-s)H_B + sH_P\,. \tag{31}$$

By evolving for a sufficiently large time T according to the Hamiltonian $H(t)$, you will prepare the ground state of H_P, which can then be measured to give the answer to the problem.

Unfortunately, this algorithm may not be efficient because the required T may be very large. Indeed, if T grows exponentially with n, the algorithm is probably of little value. However, if it turns out that T must grow only polynomially in n, the algorithm could be very useful. It is an open question as to whether there are interesting computational problems for which T only grows polynomially with n.

Although we usually can't say exactly how large a T is large enough, we can express it in terms of the spectrum of $\tilde{H}(s)$. One can show that adiabatic evolution is a good approximation if

$$T \gg \frac{\left\| \frac{\mathrm{d}\tilde{H}}{\mathrm{d}t} \right\|}{\Delta^2}\,, \tag{32}$$

where

$$\Delta = \min_{s\in[0,1]} [E_1(s) - E_0(s)] \tag{33}$$

is the smallest gap in the spectrum between the ground state and first excited state of $\tilde{H}(s)$. The numerator of (32) just enforces the condition that $\tilde{H}(s)$ is smooth; it is easy to ensure that it does not get too big. Thus whether the algorithm is efficient or not depends on the behavior of the gap Δ. If Δ is not too small, as in the example shown in Fig. 1, then the required run time T need not be too long.

To make this discussion concrete, let's consider a particular example of a computational problem one might try to solve by adiabatic evolution. This problem is known as 3SAT. In this problem, you are given a boolean formula that is a conjunction of clauses, each of which involves three bits — for example,

$$(x_0 \vee x_2 \vee \bar{x}_{11}) \wedge (\bar{x}_0 \vee x_5 \vee \bar{x}_7) \wedge \cdots . \tag{34}$$

You are asked to determine whether this formula has a satisfying assignment, i.e., whether there is some assignment of the n bits for which each clause is true.

It is easy to come up with Hamiltonians H_B and H_P that give an adiabatic algorithm for 3SAT. H_P is entirely defined by the function $h(z)$, which we can take to be

$$h(z) = \sum_C h_C(z_{i_C}, z_{j_C}, z_{k_C}) , \tag{35}$$

$$h_C(z_{i_C}, z_{j_C}, z_{k_C}) = \begin{cases} 0 , & (z_{i_C}, z_{j_C}, z_{k_C}) \text{ satisfies clause } C \\ 1 , & (z_{i_C}, z_{j_C}, z_{k_C}) \text{ violates clause } C . \end{cases} \tag{36}$$

Every violation of a clause adds one to the energy, so a satisfying assignment exists if and only if the ground state of H_P has energy zero. A sensible choice for the beginning Hamiltonian is

$$H_B = \sum_C H_{B,C} , \tag{37}$$

where

$$H_{B,C} = \frac{1}{2}\left(1 - \sigma_x^{(i_C)}\right) + \frac{1}{2}\left(1 - \sigma_x^{(j_C)}\right) + \frac{1}{2}\left(1 - \sigma_x^{(k_C)}\right) . \tag{38}$$

The ground state of this Hamiltonian is simply

$$|\psi(0)\rangle = \frac{1}{2^{n/2}} \sum_{z=0}^{2^n-1} |z\rangle , \tag{39}$$

the equal superposition of all possible bit assignments.

3SAT is believed to be a very difficult problem. Computer scientists refer to it as *NP-complete*, which means that an efficient algorithm for 3SAT could be turned into an efficient algorithm for a lot of other interesting problems. Thus an efficient algorithm for 3SAT would have profound computational implications. How could we determine whether the adiabatic algorithm is efficient? In the absence of theoretical insight, we might try simulating the algorithm on a classical computer. However, due to the exponential storage requirements for such a simulation, it is hard to go beyond $n \approx 20$ qubits with current computer technology. Such simulations show that the adiabatic algorithm is not an immediate failure, but of course say nothing about asymptotics. Another option would be to simply run the algorithm on a large quantum computer when one becomes available. Of course, even if the algorithm fails to solve 3SAT efficiently, there might be easier problems (or just differently formulated ones) for which it performs well.

Assuming we can find interesting problems where the adiabatic algorithm is useful, it has the additional nice property that it is inherently robust against error. Note that the path from H_B to H_P is irrelevant, so small deviations along the way (or indeed large ones, if they are smooth) will not affect the algorithm. Furthermore, so long as the computer is in an environment with a temperature T_e that satisfies

$$k_B T_e \ll \Delta , \tag{40}$$

thermal fluctuations will be unable to excite transitions to excited states, and hence will not affect the algorithm. Since the algorithm only works if the gap is not too small, the temperature that we have to cool to is not too low.

IV. OPEN QUESTION

Although it has been the most widely used concept of a quantum computer, the circuit model is not the only way to think about quantum algorithms. The paradigm of continuous-time quantum computation provides an alternative approach, either for reevaluating known algorithms (as for the Grover problem) or inventing new ones (as for quantum computation by adiabatic evolution). What other computational tasks might one perform in continuous time, and what other paradigms might prove useful for understanding quantum algorithms?

FURTHER READING

[1] A. M. Childs, E. Farhi, and J. Preskill, *Robustness of adiabatic quantum computation*, quant-ph/0108049.

[2] E. Farhi and S. Gutmann, *An analog analogue of a digital quantum computation*, Phys. Rev. A **57**, 2403 (1998), quant-ph/9612026.

[3] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, *Quantum computation by adiabatic evolution*, quant-ph/0001106.

[4] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, *A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem*, Science **292**, 472 (2001), quant-ph/0104129.

[5] L. K. Grover, in *Proc. 28th Annual ACM Symposium on the Theory of Computation*, 212 (ACM Press, New York, 1996).

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000), Sec. 6.2.

[7] J. Roland and N. J. Cerf, *Quantum search by local adiabatic evolution*, quant-ph/0107015.