Quantum algorithms (CO 781/CS 867/QIC 823, Winter 2013)
Andrew Childs, University of Waterloo

LECTURE 9: Simulating Hamiltonian dynamics

So far, we have focused on quantum algorithms for the hidden subgroup problem, with applications to number-theoretic problems such as factoring, computing discrete logarithms, and performing calculations in number fields. Another major potential application of quantum computers is the simulation of quantum dynamics. Indeed, this was the idea that first led Feynman to propose the concept of a quantum computer. In this lecture we will see how a universal quantum computer can efficiently simulate several natural families of Hamiltonians. These simulation methods could be used either to simulate actual physical systems, or to implement quantum algorithms defined in terms of Hamiltonian dynamics (such as continuous-time quantum walks and adiabatic quantum algorithms).

Hamiltonian dynamics

In quantum mechanics, time evolution of the wave function $|\psi(t)\rangle$ is governed by the Schrödinger equation,

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

Here $H(t)$ is the *Hamiltonian*, an operator with units of energy, and $\hbar$ is Planck's constant. For convenience it is typical to choose units in which $\hbar = 1$. Given an initial wave function $|\psi(0)\rangle$, we can solve this differential equation to determine $|\psi(t)\rangle$ at any later (or earlier) time t.

For H independent of time, the solution of the Schrödinger equation is $|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$. For simplicity we will only consider this case. There are many situations in which time-dependent Hamiltonians arise, not only in physical systems but also in computational applications such as adiabatic quantum computing. In such cases, the evolution cannot in general be written in such a simple form, but nevertheless similar ideas can be used to simulate the dynamics.

Efficient simulation

We will say that a Hamiltonian H acting on n qubits can be *efficiently simulated* if for any $t > 0$, $\epsilon > 0$ there is a quantum circuit U consisting of $\mathrm{poly}(n, t, 1/\epsilon)$ gates such that $\|U - e^{-iHt}\| < \epsilon$. Clearly, the problem of simulating Hamiltonians in general is BQP-hard, since we can implement any quantum computation by a sequence of Hamiltonian evolutions. In fact, even with natural restrictions on the kind of Hamiltonians we consider, it is easy to specify Hamiltonian simulation problems that are BQP-complete (or more precisely, PromiseBQP-complete).

You might ask why we define the notion of efficient simulation to be polynomial in t; if t is given as part of the input, this means that the running time is, strictly speaking, not polynomial in the input size. However, one can show that a running time polynomial in $\log t$ is impossible; running time $\Omega(t)$ is required in general (intuitively, one cannot "fast forward" the evolution according to a generic Hamiltonian). The dependence on ϵ is more subtle. There are no nontrivial lower bounds on the running time in terms of ϵ, and it is an open question to better understand the complexity of quantum simulation as a function of simulation error.

We would like to understand the conditions under which a Hamiltonian can be efficiently simulated. Of course, we cannot hope to efficiently simulate arbitrarily Hamiltonians, just as we cannot

hope to efficiently implement arbitrary unitaries. Instead, we will simply describe a few classes of Hamiltonian that can be efficiently simulated. Our strategy will be to start from simple Hamiltonians that are easy to simulate and define ways of combining the known simulations to give more complicated ones.

There are a few cases where a Hamiltonian can obviously simulated efficiently. For example, this is the case if H only acts nontrivially on a constant number of qubits, simply because any unitary evolution on a constant number of qubits can be approximated with error at most ϵ using $\text{poly}(\log \frac{1}{\epsilon})$ one- and two-qubit gates, using the Solovay-Kitaev theorem.

Note that since we require a simulation for an arbitrary time t (with $\text{poly}(t)$ gates), we can rescale the evolution by any polynomial factor: if H can be efficiently simulated, then so can cH for any $c = \text{poly}(n)$. This holds even if $c < 0$, since any efficient simulation is expressed in terms of quantum gates, and can simply be run in reverse.

In addition, we can rotate the basis in which a Hamiltonian is applied using any unitary transformation with an efficient decomposition into basic gates. In other words, if H can be efficiently simulated and the unitary transformation U can be efficiently implemented, then $UHU^\dagger$ can be efficiently simulated. This follows from the simple identity

$$e^{-iUHU^\dagger t} = Ue^{-iHt}U^\dagger. \tag{2}$$

Another simple but useful trick for simulating Hamiltonians is the following. Suppose H is diagonal in the computational basis, and any diagonal element $d(a) = \langle a|H|a\rangle$ can be computed efficiently. Then H can be simulated efficiently using the following sequence of operations, for any input computational basis state $|a\rangle$:

$$\begin{aligned} |a,0\rangle &\mapsto |a,d(a)\rangle && (3)\\ &\mapsto e^{-itd(a)}|a,d(a)\rangle && (4)\\ &\mapsto e^{-itd(a)}|a,0\rangle && (5)\\ &= e^{-iHt}|a\rangle|0\rangle. && (6) \end{aligned}$$

By linearity, this process simulates H for time t on an arbitrary input.

Note that if we combine this simulation with the previous one, we have a way to simulate any Hamiltonian that can be efficiently diagonalized, and whose eigenvalues can be efficiently computed.

Product formulas

Many natural Hamiltonians have the form of a sum of terms, each of which can be simulated by the techniques described above. For example, consider the Hamiltonian of a particle in a potential:

$$H = \frac{p^2}{2m} + V(x).$$

To simulate this a digital quantum computer, we can imagine discretizing the x coordinate. The operator $V(x)$ is diagonal, and natural discretizations of $p^2 = -\mathrm{d}^2/\mathrm{d}x^2$ are diagonal in the discrete Fourier basis. Thus we can efficiently simulate both $V(x)$ and $p^2/2m$. Similarly, consider the Hamiltonian of a spin system, say of the form

$$H = \sum_i h_i X_i + \sum_{ij} J_{ij} Z_i Z_j$$

(or more generally, any k-local Hamiltonian, a sum of terms that each act on at most k qubits). This consists of a sum of terms, each of which acts only only a constant number of qubits and hence is easy to simulate.

In general, if H_1 and H_2 can be efficiently simulated, then $H_1 + H_2$ can also be efficiently simulated. If the two Hamiltonians commute, then this is trivial, since $e^{-iH_1t}e^{-iH_2t} = e^{-i(H_1+H_2)t}$. However, in the general case where the two Hamiltonians do not commute, we can still simulate their sum as a consequence of the Lie product formula

$$e^{-i(H_1+H_2)t} = \lim_{m\to\infty} \left(e^{-iH_1t/m}e^{-iH_2t/m}\right)^m. \tag{7}$$

A simulation using a finite number of steps can be achieved by truncating this expression to a finite number of terms, which introduces some amount of error that must be kept small. In particular, if we want to have

$$\left\| \left(e^{-iH_1t/m}e^{-iH_2t/m}\right)^m - e^{-i(H_1+H_2)t} \right\| \leq \epsilon, \tag{8}$$

it suffices to take $m = O((\nu t)^2/\epsilon)$, where $\nu := \max\{\|H_1\|, \|H_2\|\}$. (The requirement that H_1 and H_2 be efficiently simulable means that ν can be at most $\mathrm{poly}(n)$.)

It is somewhat unappealing that to simulate an evolution for time t, we need a number of steps proportional to t^2. Fortunately, the situation can be improved if we use higher-order approximations of (7). For example, one can show that

$$\left\| \left(e^{-iH_1t/2m}e^{-iH_2t/m}e^{-iH_1t/2m}\right)^m - e^{-i(H_1+H_2)t} \right\| \leq \epsilon \tag{9}$$

with a smaller value of m. In fact, by using even higher-order approximations, it is possible to show that $H_1 + H_2$ can be simulated for time t with only $O(t^{1+\delta})$, for any fixed $\delta > 0$, no matter how small.

A Hamiltonian that is a sum of polynomially many terms can be efficiently simulated by composing the simulation of two terms, or by directly using an approximation to the identity

$$e^{-i(H_1+\cdots+H_k)t} = \lim_{m\to\infty} \left(e^{-iH_1t/m}\cdots e^{-iH_kt/m}\right)^m. \tag{10}$$

Another way of combining Hamiltonians comes from commutation: if H_1 and H_2 can be efficiently simulated, then $i[H_1, H_2]$ can be efficiently simulated. This is a consequence of the identity

$$e^{[H_1,H_2]t} = \lim_{m\to\infty} \left(e^{-iH_1\sqrt{t/m}}e^{-iH_2\sqrt{t/m}}e^{iH_1\sqrt{t/m}}e^{iH_2\sqrt{t/m}}\right)^m, \tag{11}$$

which can again be approximated with a finite number of terms. However, I don't know of any algorithmic application of such a simulation.

Sparse Hamiltonians

We will say that an $N \times N$ Hermitian matrix is *sparse* (in a fixed basis) if, in any fixed row, there are only $\mathrm{poly}(\log N)$ nonzero entries. The simulation techniques described above allow us to efficiently simulate sparse Hamiltonians. More precisely, suppose that for any a, we can efficiently determine all of the bs for which $\langle a|H|b\rangle$ is nonzero, as well as the values of the corresponding matrix elements;

then H can be efficiently simulated. In particular, this gives an efficient implementation of the continuous-time quantum walk on any graph $G = (V, E)$ whose maximum degree is $\mathrm{poly}(\log |V|)$.

The basic idea of the simulation is to edge-color the graph, simulate the edges of each color separately, and combine these pieces using (7). The main new technical ingredient in the simulation is a means of coloring the edges of the graph of nonzero matrix elements of H. A classic result in graph theory (Vizing's Theorem) says that a graph of maximum degree d has an edge coloring with at most $d+1$ colors (in fact, the edge chromatic number is either d or $d+1$). If we are willing to accept a polynomial overhead in the number of colors used, then we can actually find an edge coloring using only local information about the graph.

Lemma. *Suppose we are given an undirected graph G with N vertices and maximum degree d, and that we can efficiently compute the neighbors of any given vertex. Then there is an efficiently computable function $c(a, b) = c(b, a)$ taking $\mathrm{poly}(d, \log N)$ values such that for all a, $c(a, b) = c(a, b')$ implies $b = b'$. In other words, $c(a, b)$ is a coloring of G.*

Here is a simple proof showing that $O(d^2 \log N)$ colors are sufficient (note that stronger results are possible):

Proof. Number the vertices of G from 1 through N. For any vertex a, let $\mathrm{idx}(a, b)$ denote the index of vertex b in the list of neighbors of a. Also, let $k(a, b)$ be the index of the first bit at which a and b differ. Note that $k(a, b) = k(b, a)$, and $k \leq \lceil \log_2 N \rceil$.

For $a < b$, define the color of the edge ab to be the 4-tuple

$$c(a, b) := \big(\mathrm{idx}(a, b), \mathrm{idx}(b, a), k(a, b), b_{k(a,b)}\big) \tag{12}$$

where b_k denotes the kth bit of b. For $a > b$, define $c(a, b) := c(b, a)$.

Now suppose $c(a, b) = c(a, b')$. There are four possible cases:

1. Suppose $a < b$ and $a < b'$. Then the first component of c shows that $\mathrm{idx}(a, b) = \mathrm{idx}(a, b')$, which implies $b = b'$.

2. Suppose $a > b$ and $a > b'$. Then the second component of c shows that $\mathrm{idx}(a, b) = \mathrm{idx}(a, b')$, which implies $b = b'$.

3. Suppose $a < b$ and $a > b'$. Then from the third and fourth components of c, $k(a, b) = k(a, b')$ and $a_{k(a,b)} = b_{k(a,b)}$, which is a contradiction.

4. Suppose $a > b$ and $a < b'$. Then from the third and fourth components of c, $k(a, b) = k(a, b')$ and $a_{k(a,b)} = b'_{k(a,b)}$, which is a contradiction.

Each case that does not lead to a contradiction gives rise to a valid coloring, which completes the proof. □

Given this lemma, the simulation proceeds as follows. Write H as a diagonal matrix plus a matrix with zeros on the diagonal. We have already shown how to simulate the diagonal part, so we can assume H has zeros on the diagonal without loss of generality.

It suffices to simulate the term corresponding to the edges of a particular color c. We show how to make the simulation work for any particular vertex x; then it works in general by linearity. By computing the complete list of neighbors of x and computing each of their colors, we can reversibly

compute $v_c(x)$, the vertex adjacent to x via an edge with color c, along with the associated matrix element:

$$|x\rangle \mapsto |x, v_c(x), H_{x,v_c(x)}\rangle. \tag{13}$$

Then we can simulate the H-independent Hamiltonian defined by the map

$$|x, y, h\rangle \mapsto h|y, x, h^*\rangle \tag{14}$$

since it is easily diagonalized, as it consists of a direct sum of two-dimensional blocks. Finally, we can uncompute the second and third registers. Before the uncomputation, the simulation produces a linear combination of the states $|x, v_c(x), H_{x,v_c(x)}\rangle$ and $|v_c(x), x, H^*_{x,v_c(x)}\rangle$. Since

$$|v_c(x), x, H^*_{x,v_c(x)}\rangle = |v_c(x), v_c(v_c(x)), H_{v_c(x),x}\rangle, \tag{15}$$

the uncomputation works identically for both components.

Measuring an operator

So far, we have focused on the simulation of Hamiltonian dynamics. However, it is also possible to view a Hermitian operator not as the generator of dynamics, but as a quantity to be measured. In a practical quantum simulation, the desired final measurement might be of this type. For example, we might want to measure the final energy of the system, and the final Hamiltonian could be a sum of noncommuting terms.

It turns out that any Hermitian operator that can be efficiently simulated (viewing it as the Hamiltonian of a quantum system) can also be efficiently *measured* using a formulation of the quantum measurement process given by von Neumann. In fact, von Neumann's procedure is essentially the same as quantum phase estimation!

In von Neumann's description of the measurement process, a measurement is performed by coupling the system of interest to an ancillary system, which we call the *pointer*. Suppose that the pointer is a one-dimensional free particle and that the system-pointer interaction Hamiltonian is $H \otimes p$, where p is the momentum of the particle. Furthermore, suppose that the mass of the particle is sufficiently large that we can neglect the kinetic term. Then the resulting evolution is

$$e^{-itH\otimes p} = \sum_a \left[|E_a\rangle\langle E_a| \otimes e^{-itE_a p} \right], \tag{16}$$

where $|E_a\rangle$ are the eigenstates of H with eigenvalues E_a. Suppose we prepare the pointer in the state $|x = 0\rangle$, a narrow wave packet centered at $x = 0$. Since the momentum operator generates translations in position, the above evolution performs the transformation

$$|E_a\rangle \otimes |x = 0\rangle \to |E_a\rangle \otimes |x = tE_a\rangle. \tag{17}$$

If we can measure the position of the pointer with sufficiently high precision that all relevant spacings $x_{ab} = t|E_a - E_b|$ can be resolved, then measurement of the position of the pointer—a fixed, easy-to-measure observable, independent of H—effects a measurement of H.

Von Neumann's measurement protocol makes use of a continuous variable, the position of the pointer. To turn it into an algorithm that can be implemented on a digital quantum computer, we can approximate the evolution (16) using r quantum bits to represent the pointer. The full Hilbert

space is thus a tensor product of a 2^n-dimensional space for the system and a 2^r-dimensional space for the pointer. We let the computational basis of the pointer, with basis states $\{|z\rangle\}$, represent the basis of momentum eigenstates. The label z is an integer between 0 and $2^r - 1$, and the r bits of the binary representation of z specify the states of the r qubits. In this basis, p acts as

$$p|z\rangle = \frac{z}{2^r}|z\rangle. \tag{18}$$

In other words, the evolution $e^{-itH\otimes p}$ can be viewed as the evolution e^{-itH} on the system for a time controlled by the value of the pointer.

Expanded in the momentum eigenbasis, the initial state of the pointer is

$$|x = 0\rangle = \frac{1}{2^{r/2}} \sum_{z=0}^{2^r-1} |z\rangle. \tag{19}$$

The measurement is performed by evolving under $H \otimes p$ for some appropriately chosen time t. After this evolution, the position of the simulated pointer can be measured by measuring the qubits that represent it in the x basis, i.e., the Fourier transform of the computational basis.

Note that this discretized von Neumann measurement procedure is equivalent to phase estimation. Recall that in the phase estimation problem, we are given an eigenvector $|\psi\rangle$ of a unitary operator U and asked to determine its eigenvalue $e^{i\phi}$. The algorithm uses two registers, one that initially stores $|\psi\rangle$ and one that will store an approximation of the phase ϕ. The first and last steps of the algorithm are Fourier transforms on the phase register. The intervening step is to perform the transformation

$$|\psi\rangle \otimes |z\rangle \to U^z|\psi\rangle \otimes |z\rangle, \tag{20}$$

where $|z\rangle$ is a computational basis state. If we take $|z\rangle$ to be a momentum eigenstate with eigenvalue z (i.e., if we choose a different normalization than in (18)) and let $U = e^{-iHt}$, this is exactly the transformation induced by $e^{-i(H\otimes p)t}$. Thus we see that the phase estimation algorithm for a unitary operator U is exactly von Neumann's prescription for measuring $i \ln U$.