CS 880: Quantum Information Processing 10/19/2010

Lecture 18: Simulating Hamiltonian Dynamics

Instructor: Dieter van Melkebeek Scribe: Tyson Williams

Last lecture, we finished our discussion of discrete quantum walks. Today we discuss continuous-time quantum walks and how to simulate them using quantum gates. Their applications include solving sparse linear systems of equations and formula evaluation.

1 Continuous-time Walks

1.1 Classical

A classical continuous-time random walk on a graph G is specified by a vector $P(t)$ where

$$P_v(t) = \Pr[\text{walk is at vertex } v \text{ at time } t]. \tag{1}$$

Each vertex sends probability to its neighbors proportional to its own probability. This process is described by

$$\frac{dP(t)}{dt} = (A - D)P(t), \tag{2}$$

where A is the adjacency matrix G (not normalized) and D a diagonal matrix with $D_{ii} = deg(v_i)$. The matrix $L = A - D$ is known as the Laplacian of G.

Exercise 1. *Prove that (2) describes a valid probabilistic process. That is, show for all t that each component is never negative and all components sum to 1.*

There is, in fact, a closed form solution for (2), which is

$$P(t) = e^{Lt}P(0). \tag{3}$$

1.2 Quantum

The transition from classical to quantum is easier in the continuous-time setting than in the discrete-time setting. It is

$$i\frac{d\left|\psi(t)\right\rangle}{dt} = (A - D)\left|\psi(t)\right\rangle. \tag{4}$$

While (2) preserves probability, (4) preserves the 2-norm. We can state this using the bra-ket notation

$$\langle\psi(t)|\psi(t)\rangle = \langle\psi(t)|\left|\psi(t)\right\rangle = \left|\psi(t)\right\rangle^{\dagger}\left|\psi(t)\right\rangle = 1, \tag{5}$$

which is just the inner product of $\psi(t)$ with itself. Equation (4) then becomes

$$i\frac{d\left|\psi(t)\right\rangle}{dt} = L\psi(t). \tag{6}$$

Proof that equation (6) is a valid quantum process. Since

$$\begin{aligned}
\frac{d}{dt}\langle\psi(t)|\psi(t)\rangle &= \left(\frac{d}{dt}\langle\psi(t)|\right)|\psi(t)\rangle + \langle\psi(t)|\left(\frac{d}{dt}|\psi(t)\rangle\right) \\
&= iL^\dagger\langle\psi(t)|\psi(t)\rangle - iL\langle\psi(t)|\psi(t)\rangle \\
&= i(L^\dagger - L)\langle\psi(t)|\psi(t)\rangle \\
&= i(L - L)\langle\psi(t)|\psi(t)\rangle \qquad \text{(Since } L \text{ is Hermitian)} \\
&= 0,
\end{aligned}$$

the 2-norm of this quantum process is a constant. Thus, if $\langle\psi(0)|\psi(0)\rangle = 1$, then $\langle\psi(t)|\psi(t)\rangle = 1$ for all t. □

Physicists will recognize (6) as Schrödinger's equation, which holds even when L is replaced with any Hermitian matrix H that varies over time. The closed-form solution for constant H in the quantum case is similar to the classic one, namely

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle. \tag{7}$$

When discussing how to solve well-conditioned systems of linear equations, e^{-iHt} was the operator U that we used. We used the fact that if H is efficiently sparse, then we can compute U efficiently. We now show how to do that.

2 Simulating Sparse Hamiltonians

To simulate a sparse Hamiltonian H efficiently, we need a handle on H. It is not enough for H to be sparse. It needs to sparse in a "efficient" way. When looking at a row, we need to be able to efficiently locate and approximately compute the nonzero entries. We say that H is sparse when it has at most s nonzero entries per row/column where $s = \text{poly}\log(N)$. We can efficiently approximate $U = e^{-iHt}$ when H is efficiently sparse.

Our algorithm will be slightly worse parameters than the one we used while discussing well-conditioned systems of linear equations.

2.1 H is Diagonal

If H is diagonal, then e^{-iHt} is just a combination of rotations.

2.2 H is Efficiently Diagonalizable

Being a Hermitian matrix, H has an orthonormal basis of eigenvectors. This implies that there exists a matrix V such that $HV = VD$ where V, whose rows are the eigenvectors of H, is efficiently computable and D is a diagonal matrix. Then

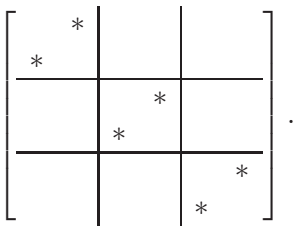
$$e^{-iHt} = Ve^{-iDt}V^{-1} \tag{8}$$

and we have reduced the problem to the case with a diagonal matrix.

2.3 H is a Matching

This is actually a special case of H begin efficiently diagonalizable. We single it out and discuss it further because we will use it later.

If the graph underlying H is a matching, then H has at most one nonzero entry in each row/column. We can simultaneously permute the rows and columns to get a matrix of the form



Since 2×2 matrices are always efficiently diagonalizable when its entries are efficiently computable, we are done.

2.4 Closure Under Addition

If we can efficiently compute $U_1 = e^{-iH_1 t}$ and $U_2 = e^{-iH_2 t}$, then we can efficiently compute $U = e^{-i(H_1+H_2)t}$. This is easy when H_1 and H_2 commute because then

$$U = e^{-i(H_1+H_2)t} = e^{-iH_1 t} e^{-iH_2 t} = U_1 U_2. \tag{9}$$

When H_1 and H_2 do not commute, we take advantage of the fact that we only need to approximate $e^{-i(H_1+H_2)t}$. The Taylor series expansion for e^{-iHt} is

$$e^{-iHt} = I - iHt + O\left(||H||^2 t^2\right). \tag{10}$$

Since

$$e^{-iHt} = \left(e^{-iHt/n}\right)^n, \tag{11}$$

we are also interested in the Taylor series expansion for $e^{-iHt/n}$, which is

$$e^{-iHt/n} = I - iH\frac{t}{n} + O\left(||H||^2 \frac{t^2}{n^2}\right). \tag{12}$$

Then

$$\begin{aligned} e^{-iH_1 t/n} e^{-iH_2 t/n} &= I - i(H_1 + H_2)\frac{t}{n} + O\left(\left(||H_1||^2 + ||H^2||^2\right)\frac{t^2}{n^2}\right) \\ &= e^{-i(H_1+H_2)t/n} + O\left(\left(||H_1||^2 + ||H^2||^2\right)\frac{t^2}{n^2}\right), \end{aligned}$$

so

$$\begin{aligned} e^{-i(H_1+H_2)t} &= \left(e^{-i(H_1+H_2)t/n}\right)^n \\ &= e^{-iH_1 t/n} e^{-iH_2 t/n} + O\left(\left(||H_1||^2 + ||H^2||^2\right)\frac{t^2}{n}\right). \end{aligned}$$

Closure under addition generalizes to

$$e^{-i\sum_{j=1}^{k} H_j t} = \left(\prod_{j=1}^{k} e^{-iH_j t/n}\right)^n + O\left(\left(\sum_{j=1}^{k} ||H_j||^2\right)\frac{t^2 k}{n}\right). \tag{13}$$

In order to make the error term in (13) is no more than ϵ, it suffices for n to be at least

$$\text{poly}\left(\max_j ||H_j||, k, t\right)\frac{1}{\epsilon}.$$

2.5 H is Sparse

When H is sparse, the idea is to efficiently write H as a sum of efficient matchings and apply cases 2.3 and 2.4.

First Attempt Let H_j be the jth nonzero entry in H. This will not work because the H_j's will not always be Hermitian.

Second Attempt Decompose H into matchings. By Vizing's Theorem, every graph G can be edge colored with at most $\Delta(G)+1$ colors, where $\Delta(G)$ is the maximum degree of the graph. Notice that the set of edges for each color is a matching. Unfortunately, we need to efficiently compute an edge coloring but all known constructive proofs of this result are not efficient. If we use $O(s^2 \log^2 N)$ colors, then efficient constructions are known.

For a graph $G=(V,E)$, label the vertices with $1,\ldots,N=|V|$. If G has any self loops, we can take care of them by adding a diagonal matrix, which is efficiently computable by cases 2.1 and 2.4. Thus we can assume that G has no self loops. To the edge $(v,w)\in E$, assign the color

$$c(v,w) = \begin{cases} (\text{index of } v \text{ as a neighbor of } w, & \\ \ \text{index of } w \text{ as a neighbor of } v, & \\ \ m(v,w), & \\ \ w \bmod m(v,w)) & v<w \\ c(w,v) & v>w, \end{cases}$$

where

$$m(v,w) = \min\{\mu \in \mathbb{Z}^+ \mid v \not\equiv w \pmod{\mu}\},$$

which exists and is $O(\log N)$ since $0<v<w\le N$. This can be seen by contradiction. Suppose that $v \not\equiv w \pmod N$ but $\mu = \omega(\log N)$. Then $v \equiv w \pmod\mu$ for all μ from 1 to $O(\log(N))$. In particular, v and w are equivalent modulo the primes in that range. However, a nontrivial fact from number theory is that the the product all primes less than a number n is at least 2^n. Thus by the Chinese remainder theorem, we get that $v \equiv w \pmod N$, a contradiction.

This coloring is consistent (since $c(v,w)=c(w,v)$) and is efficient to compute. It remains to show that it is a valid coloring.

Proof that this coloring is valid. It suffices to show that $c(v,w)=c(v,w') \implies w=w'$.

Case 1: $v < w$ **and** $v < w'$ The second component of the color implies that $w = w'$.

Case 2: $v > w$ **and** $v > w'$ The first component of the color implies that $w = w'$ (and it is also symmetric to case 1).

Case 3: $v < w$ **and** $v > w'$ The third component of the color is $\mu = m(v, w) = m(w', v)$, so $w \equiv v \bmod \mu$, which is a contradiction with the construction of $\mu = m(v, w)$.

Case 4: $v > w$ **and** $v < w'$ This case is symmetric to case 3. □

Based on these techniques, our runtime for efficiently (and approximately) computing e^{-iHt} when H is sparse is

$$\text{poly}\left(s, \log N, \max_j ||H_j||, t, \frac{1}{\epsilon}\right)$$

for an accuracy of ϵ. We refrain from specifying the exact polynomial since it is not as optimal as the one we mentioned while discussing well-conditioned systems of linear equations.

3 Application: Formula Evaluation

Consider the formula for determining which player in a two-player game (where each player always has two possible moves unless the game is over) has a winning strategy. Say that the formula evaluates to 1 if the first player to move has a winning strategy and 0 if the second player to move has a winning strategy. This type of formula is known as a game tree and has alternating levels of OR and AND gates with leaf nodes that indicate which player wins. The question is this, how many of the N leaves do we need to query to determine who wins?

Deterministically, we need to query $\Theta(N)$. Using randomness to recursively determine which branch to evaluate first, the expected number of leaves we need to query is $\Theta(N^d)$, where $d \approx 0.753$. This improvement comes from the fact that while evaluating an OR (equivalently AND) gate, we might find a branch that evaluates to 1 (equivalently 0) before evaluating the other branch (as long as such a branch exists).

The best known quantum algorithm uses $O(\sqrt{N} \log N)$ queries. The $\sqrt{N}$ term looks likes Grover search. The $\log N$ term would intuitively come from amplifying Grover's success probability. However, Grover cannot get this result. This result is actually an application of discrete random walks that were inspired by continuous random walks.

4 Next Time

In our next lecture, we will discuss adiabatic quantum computation. This is an alternate model of quantum computation that is similar to continue-time quantum walks. We will show that this model is universal and can be simulated by the quantum circuit model with a polynomial amount of overhead in time.