Quantum algorithms (CO 781, Winter 2008)

Prof. Andrew Childs, University of Waterloo

LECTURE 11: From random walk to quantum walk

We now turn to a second major topic in quantum algorithms, the concept of quantum walk. In this lecture we will introduce quantum walk as a natural analog of classical random walk, and we'll see a few preliminary examples of how the two kinds of processes differ.

Randomized algorithms Randomness is a ubiquitous tool in algorithms and complexity theory. You could take an entire course on randomized algorithms (and if you haven't, you probably should), but here are a few highlights.

In query complexity, there are functions that can be computed with dramatically fewer queries using a randomized algorithm. A simple example is provided by the Deutsch-Jozsa problem (distinguishing constant functions from balanced functions $f : \{0,1\}^n \to \{0,1\}$): any deterministic algorithm must make $\Omega(2^n)$ queries to solve the problem, yet only a constant number of random queries suffice to determine the solution with bounded error (in fact, one-sided bounded error).

Even total functions can have a large separation between the deterministic and randomized query complexity. A classic example is the problem of *game tree evaluation*. Consider a two-player game in which players alternate turns, with each player having two possible moves on each turn, for a total of n turns. Given a black box function indicating which player wins when a certain sequence of moves is played, the goal is to determine which player can win the game. Equivalently, we want to evaluate the formula corresponding to a balanced binary tree of height n with the 2^n-bit black box input at the leaves, and internal vertices at alternate levels of the tree corresponding to AND and OR gates, respectively. Since it is possible to change the value of the formula by changing only one bit of the input, a deterministic algorithm must query all $N = 2^n$ leaves. However, there is a randomized strategy that evaluates this formula by querying only about $N^{0.753}$ leaves. One can show that the optimal strategy for evaluating the root of the tree is to pick a child at random and evaluate it recursively. The improvement from randomness comes because it is possible to get lucky (e.g., if we are evaluating an AND gate and one of its children evaluates to 0, we do not have to evaluate the other child), and in fact this must happen quite often throughout the tree (since the outputs of necessarily unlucky inputs to an AND gate are potentially lucky inputs to an OR gate, and vice versa).

Of course, we are more interested in speedups for explicit (not black-box) computational problems. An example along these lines is provided by the well-known 3SAT problem, the problem of deciding whether a 3CNF formula (a conjunction of disjunctions of 3 literals each) on n variables has a satisfying assignment. This problem is NP-complete. While a brute force search requires time 2^n, it is possible to solve the problem with a running time c^n for certain constants $c < 2$. However, better values of this constant are possible if we consider randomized algorithms. In particular, Schöning gave a simple randomized algorithm (which has subsequently been improved) that runs in time $(\frac{4}{3})^n$. The strategy is as follows: start from a random assignment of the n bits, and while the formula is not satisfied, pick a random literal in a random unsatisfied clause and flip its value. If this procedure fails to find a satisfying assignment after $O(n)$ steps, start again, and repeat the entire procedure about $(\frac{4}{3})^n$ times (or until a satisfying assignment is found). By analyzing this random process, Schöning showed that if no satisfying assignment is found, the formula is very likely to be unsatisfiable.

Of course, it would be even more interesting if there were a problem that could be solved in polynomial time by a randomized algorithm, but not by a deterministic one. This is the famous "P vs. BPP" question. Nowadays, the majority opinion is that P = BPP (i.e., that any randomized algorithm can be "derandomized"), but no proof is known. You have probably heard that primality testing, a problem for which polynomial-time randomized algorithms have been known for a long time, was recently derandomized; but there are other candidate problems to separate P from BPP, such as polynomial identity testing. In any event, even if P = BPP, there will still be reasons to consider randomized algorithms: such algorithms could be easier to discover, easier to implement, and might run faster in practice (or even in principle, up to polynomial factors).

The success of randomness as a tool for classical algorithms motivates us to consider quantum analogs of random processes as tools for quantum algorithms. Specifically, we will consider quantum analogs of random walks on undirected graphs, as the unitarity of quantum mechanics makes it difficult to define a genuinely quantum analog of a random walk on a directed graph. This may seem like a serious limitation, since many randomized algorithms (including the game tree evaluation and 3SAT algorithms described above) are essentially random walks on *directed* graphs. Nevertheless, we will see that quantum analogs of random walks on undirected graphs have many algorithmic applications. For example, there is a quantum walk algorithm that speeds up game tree evaluation.

Continuous-time quantum walk Random walks come in two flavors: discrete- and continuous-time. It is easiest to define a quantum analog of a continuous-time random walk, so we will consider this case first. Given a graph $G = (V, E)$, we define the continuous-time random walk on G as follows. Let A be the *adjacency matrix* of G, the $|V| \times |V|$ matrix with

$$A_{j,k} = \begin{cases} 1 & (j,k) \in E \\ 0 & (j,k) \notin E \end{cases} \tag{1}$$

for every pair $j, k \in V$. In particular, if we disallow self-loops, then the diagonal of A is zero. There is another matrix associated with G that is nearly as important: the *Laplacian* of G, which has

$$L_{j,k} = \begin{cases} -\deg(j) & j = k \\ 1 & (j,k) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\deg(j)$ denotes the degree of vertex j. (The Laplacian is sometimes defined differently than this—e.g., sometimes with the opposite sign. We use this definition because it makes L a discrete approximation of the Laplacian operator ∇^2 in the continuum.)

The continuous-time random walk on G is defined as the solution of the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t} p_j(t) = \sum_{k \in V} L_{jk}\, p_k(t). \tag{3}$$

Here $p_j(t)$ denotes the probability associated with vertex j at time t. This can be viewed as a discrete analog of the diffusion equation. Note that

$$\frac{\mathrm{d}}{\mathrm{d}t} \sum_{j \in V} p_j(t) = \sum_{j,k \in V} L_{jk}\, p_k(t) = 0 \tag{4}$$

(since the columns of L sum to 0), which shows that an initially normalized distribution remains normalized: the evolution of the continuous-time random walk for any time t is a *stochastic process*. The solution of the differential equation can be given in closed form as

$$p(t) = e^{Lt} p(0). \tag{5}$$

Now notice that the equation (3) is very similar to the Schrödinger equation

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}t}|\psi\rangle = H|\psi\rangle \tag{6}$$

except that it lacks the factor of i. If we simply insert this factor, and rename the probabilities $p_j(t)$ as quantum amplitudes $q_j(t) = \langle j|\psi(t)\rangle$ (where $\{|j\rangle : j \in V\}$ is an orthonormal basis for the Hilbert space), then we obtain the equation

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}t} q_j(t) = \sum_{k\in V} L_{jk}\, q_k(t), \tag{7}$$

which is simply the Schrödinger equation with the Hamiltonian given by the Laplacian of the graph. Since the Laplacian is a Hermitian operator, these dynamics preserve normalization in the sense that $\frac{\mathrm{d}}{\mathrm{d}t}\sum_{j\in V}|q_j(t)|^2 = 0$. Again the solution of the differential equation can be given in closed form, but here it is $|\psi(t)\rangle = e^{-iLt}|\psi(0)\rangle$.

We could also define a continuous-time quantum walk using any Hermitian Hamiltonian that respects the structure of G. For example, we could use the adjacency matrix A of G, even though this matrix cannot be used as the generator of a continuous-time classical random walk.

Discrete-time quantum walk It is trickier to define a quantum analog of a discrete-time random walk. In the simplest discrete-time random walk on G, at each time step we simply move from any given vertex to each of its neighbors with equal probability. Thus the walk is governed by the $|V| \times |V|$ matrix M with entries

$$M_{jk} = \begin{cases} 1/\deg(k) & (j,k) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

for $j, k \in V$: an initial probability distribution p over the vertices evolves to $p' = Mp$ after one step of the walk.

To define a quantum analog of this process, we would like to specify a unitary operator U with the property that an input state $|j\rangle$ corresponding to the vertex $j \in V$ evolves to a superposition of the neighbors of j. We would like this to happen in essentially the same way at every vertex, so we are tempted to propose the definition

$$|j\rangle \overset{?}{\mapsto} |\partial_j\rangle := \frac{1}{\sqrt{\deg(j)}} \sum_{k:(j,k)\in E} |k\rangle. \tag{9}$$

However, a moment's reflection shows that this typically does not define a unitary transformation, since the orthogonal states $|j\rangle$ and $|k\rangle$ corresponding to adjacent vertices j, k with a common neighbor ℓ evolve to non-orthogonal states. We could potentially avoid this problem using a rule that sometimes introduces phases, but that would violate the spirit of defining a process that behaves in the same way at every vertex.

We can get around this difficulty if we allow ourselves to enlarge the Hilbert space, an idea proposed by Watrous as part of a logarithmic-space quantum algorithm for deciding whether two vertices are connected in a graph. Let the Hilbert space consist of state of the form $|j,k\rangle$ where $(j,k) \in E$. We can think of the walk as taking place on the (directed) edges of the graph; the state $|j,k\rangle$ represents a walker at vertex j that will move toward vertex k. Each step of the walk consists of two operations. First, we apply a unitary transformation that operates on the second register conditional on the first register. This transformation is sometimes referred to as a "coin flip," as it modifies the next destination of the walker. A common choice is the Grover diffusion operator over the neighbors of j, namely

$$C := \sum_{j \in V} |j\rangle\langle j| \otimes \bigl(2|\partial_j\rangle\langle\partial_j| - I\bigr). \tag{10}$$

Next, the walker is moved to the vertex indicated in the second register. Of course, since the process must be unitary, the only way to do this is to swap the two registers using the operator

$$S := \sum_{(j,k) \in E} |j,k\rangle\langle k,j|. \tag{11}$$

Overall, one step of the discrete-time quantum walk is described by the unitary operator SC.

Now let's consider some basic examples of quantum walks on graphs, focusing on the continuous-time definition. The behavior of such walks is typically easier to calculate, and they often have similar behavior to discrete-time quantum walks. However, discrete-time quantum walks have certain implementation advantages that make them useful, so we will return to them later. In particular, we will introduce Szegedy's formulation of discrete-time quantum walk, which turns out to be a remarkably versatile tool for quantum search algorithms.

Random and quantum walks in one dimension Perhaps the best known example of a random walk is for the case of an infinite one-dimensional line, with $V = \mathbb{Z}$ and $(j,k) \in E$ iff $|j-k| = 1$. It is well known that the random walk on this graph starting from the origin (in either continuous or discrete time) typically moves a distance proportional to $\sqrt{t}$ in time t. Now let's consider the corresponding quantum walk.

To calculate the behavior of the walk, it is helpful to diagonalize the Hamiltonian. The eigenstates of the Laplacian of the graph are the momentum states $|\hat{p}\rangle$ with components

$$\langle j|\hat{p}\rangle = e^{\mathrm{i}pj} \tag{12}$$

where $-\pi \le p \le \pi$. We have

$$\begin{aligned}
\langle j|L|\hat{p}\rangle &= \langle j+1|\hat{p}\rangle + \langle j-1|\hat{p}\rangle - 2\langle j|\hat{p}\rangle & (13)\\
&= (e^{\mathrm{i}p(j+1)} + e^{\mathrm{i}p(j-1)} - 2e^{\mathrm{i}pj}) & (14)\\
&= e^{\mathrm{i}pj}(e^{\mathrm{i}p} + e^{-\mathrm{i}p} - 2) & (15)\\
&= 2(\cos p - 1)\langle j|\hat{p}\rangle, & (16)
\end{aligned}$$

so the corresponding eigenvalue is $2(\cos p - 1)$. Thus the amplitude for the walk to move from j to

k in time t is

$$\langle k|e^{-\mathrm{i}Lt}|j\rangle = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{-2it(\cos p-1)}\langle k|\hat{p}\rangle\langle\hat{p}|j\rangle\,\mathrm{d}p \tag{17}$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{ip(k-j)-2it(\cos p-1)}\,\mathrm{d}p \tag{18}$$

$$= e^{2\mathrm{it}}(-\mathrm{i})^{k-j}J_{k-j}(2t) \tag{19}$$

where $J_\nu(\cdot)$ is the Bessel function of order ν. This expression can be understood using basic asymptotic properties of the Bessel function. For large values of ν, the function $J_\nu(t)$ is exponentially small in ν for $\nu \gg t$, of order $\nu^{-1/3}$ for $nu \approx t$, and of order $\nu^{-1/2}$ for $\nu \ll t$. Thus (19) describes a wave propagating with speed 2.

We can actually use a similar calculation to exactly describe the corresponding continuous-time classical random walk, which is simply the analytic continuation $t \to it$ of the quantum case. Here the probability of moving from j to k in time t is

$$[e^{Lt}]_{kj} = e^{-2t}I_{k-j}(2t), \tag{20}$$

where $I_\nu(\cdot)$ is the modified Bessel function of order ν. For large t, this expression is approximately $\frac{1}{\sqrt{4\pi t}}\exp(-(k-j)^2/4t)$, a Gaussian of width $\sqrt{2t}$, in agreement with our expectations for a classical random walk in one dimension.

Similar calculations can be carried out for the t-step discrete-time random and quantum walks; again, the random walk moves a distance proportional to $\sqrt{t}$ and the quantum walk moves a distance proportional to t.

Random and quantum walks on the hypercube Finally, let's investigate a simpler, yet more dramatic example of a difference between the behavior of random and quantum walks. Consider the Boolean hypercube, the graph with vertex set $V = \{0,1\}^n$ and edge set $E = \{(x,y) \in V^2 : \Delta(x,y) = 1\}$, where $\Delta(x,y)$ denotes the Hamming distance between the strings x and y. When $n = 1$, the hypercube is simply an edge, with adjacency matrix

$$\sigma_x := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{21}$$

For general n, the graph is the direct product of this graph with itself n times, and the adjacency matrix is

$$A = \sum_{j=1}^{n} \sigma_x^{(j)} \tag{22}$$

where $\sigma_x^{(j)}$ denotes the operator acting as σ_x on the jth bit, and as the identity on every other bit.

For simplicity, let's consider the quantum walk with the Hamiltonian given by the adjacency matrix. (In fact, since the graph is regular, the walk generated by the Laplacian would only differ by an overall phase.) Since the terms in the above expression for the adjacency matrix commute, the unitary operator describing the evolution of this walk is

$$e^{-\mathrm{i}At} = \prod_{j=1}^{n} e^{-\mathrm{i}\sigma_x^{(j)}t} \tag{23}$$

$$= \bigotimes_{j=1}^{n} \begin{pmatrix} \cos t & -\mathrm{i}\sin t \\ -\mathrm{i}\sin t & \cos t \end{pmatrix}. \tag{24}$$

After time $t = \pi/2$, this operator flips every bit of the state (up to an overall phase), mapping any input state $|x\rangle$ to the state $|\bar{x}\rangle$ corresponding to the opposite vertex of the hypercube.

In contrast, consider the continuous- or discrete-time random walk starting from the vertex x. It is not hard to show that the probability of reaching the opposite vertex $\bar{x}$ is exponentially small at any time: the walk rapidly reaches the uniform distribution over all 2^n vertices of the hypercube.

This simple example shows that random and quantum walks can exhibit radically different behavior. In subsequent lectures, we will see how such differences can be harnessed to give quantum algorithms that outperform classical computation.