Graphs in Quantum Information Theory

David Feder, University of Calgary

1 Introduction

The fascinating properties of graphs have been studied since ancient times. Modern 'graph theory' originated fairly recently, with Leonhard Euler's famous paper on the Seven Bridges of Königsberg, published in 1736. The problem was to find a way to wander about Königsberg, crossing each bridge exactly once. The layout of the bridges in town is given in Fig. 1.

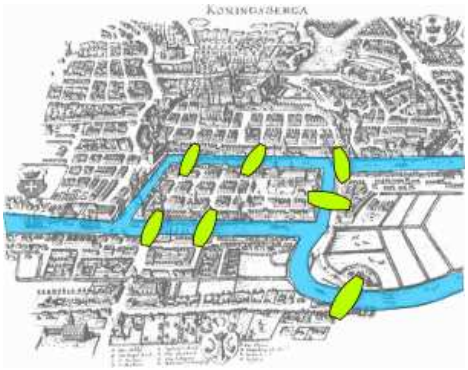

Figure 1: The seven bridges problem, from wikipedia.

The solution of the problem will be discussed during the summer school; you can use the following space to fill it in! But I'll give you a hint: the problem reduces to analyzing the following graph:

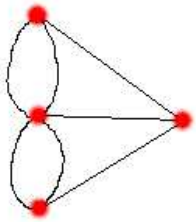

Figure 2: The graph associated with the seven bridges problem.

Solution:

By recasting the above topological problem in terms of a graph consisting of vertices and edges, Euler laid the foundations of modern graph theory. His original paper can be found on the summer school website. Since then, numerous physical and mathematical problems have been phrased in terms of an underlying graph. One of the most famous problems in graph theory over the past couple of centuries was the so-called 'four-colour problem,' posed by Francis Guthrie in 1852. This asked if it is possible to colour a map using only four colours, without any two adjacent countries sharing the same colour. Though it was intensely studied by many leading mathematicians, who developed many important graph theoretic concepts and tools, the four-colour problem was only solved in 1977 [1], in part using computers to perform exhaustive search.

There are many important problems in graph theory that remain to be solved. A wide class are graph representations of combinatorial optimization problems, such as the *travelling salesman problem* (finding the lowest-cost path that takes you through each vertex exactly once and brings you back to the start) which is known to be NP-hard. An efficient algorithm for this type of problem (also known as Hamiltonian cycle problems) would have far-reaching practical consequences, such as for scheduling and manufacturing. Another class of problems involves finding patterns. One of the most famous examples is the *graph isomorphism problem*, where one needs to determine if two different graphs can be made the same by permuting (or relabelling) the vertices of one of them. An important class of this type of problem is to enumerate all of the subgraphs of a given type (such as all *complete* subgraphs with n vertices, where each vertex is connected to all others in the subgraph by a single edge); many of these problems, such as the *clique problem* which is to determine the largest complete subgraph, are also NP-hard. But since networks pervade our society (think of the WWW, or even networks representing society itself!), efficient algorithms would allow us to identify and predict the properties of highly complex systems.

In this short course, I will discuss some of the relationships between quantum information and graph theory that have been developed over the past few years. The ultimate goal is the development of quantum algorithms for problems in graph theory that can outperform their classical counterparts. There are two main points of contact between these seemingly disparate fields. One of these is to consider a graph as a collection of vertices in either real or configuration space, connected by edges. The basics of graph theory are covered in Section 2. A major effort in this case is to construct quantum algorithms that can efficiently determine properties of the graph, and in Section 3 I will describe in detail one such approach which is based on quantum walks. The other point of contact is to consider each vertex as a qudit, with edges representing an entangling operation. In this picture, each graph is uniquely associated with a highly entangled quantum state known as a graph state or stabilizer state. The theory of stabilizers will be introduced in Section 4 and their relationship with quantum error correction will be discussed in Section 5. In Section 6 I will also show that certain graph states, known as cluster states, are a resource for universal quantum computation based only on measurements.

2 Graph Basics

One can be very formal when talking about graphs, but I prefer to make things as simple as possible. Basically, graphs are characterized by a set of vertices V and the set of edges E that connect them. The resulting graph is sometimes designated $G(V, E)$. Sometimes the vertices are called 'nodes' and the edges 'links.' The *degree of a vertex* equals the number of edges emanating from it. Examples of graphs are shown in Fig. 3. (Notice that the maximum clique number is 3 for the graph on the right in Fig. 3). If there is a link $E(1, 2)$ between vertex 1 and 2, and also $E(2, 1)$ between 2 and 1, and the same is true for all the other edges E, then the graph is 'undirected.' If the edges go one way but not the other, then the graph is said to be 'directed,' and is sometimes referred to as a digraph. If two vertices share more than a single edge, the graph is called a 'multigraph.' If the weights of all the edges (described below) are all equal, then the graph is 'regular'; otherwise it is 'weighted.' That'll do for now!!

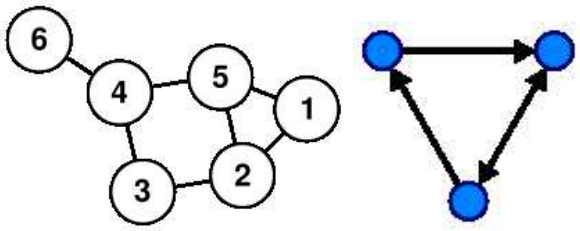

Figure 3: Examples of graphs. A labelled undirected regular graph is shown on the left while an unlabelled directed regular graph is shown on the right.

The *adjacency matrix* A for a simple graph is a matrix whose rows and columns are labelled by the vertices and the non-zero entries give the edges. For an undirected regular graph with no self-loops, the matrix is symmetric and traceless, with all entries either 0 or 1. To be concrete, the adjacency matrices for the graphs shown in Fig. 3 are given by

$$A_{\text{left}} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}; \quad A_{\text{right}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

3 Quantum Walks on Graphs

One of the most efficient classical ways to explore graphs is the random (or drunken) walk, which underpins many important algorithms in physics and computer science, among other things. Random walks also provide a nice description of how classical particles diffuse toward some kind of equilibrium. Because quantum particles can become delocalized (i.e. can go many directions simultaneously), one might expect algorithms based on a quantum version of a random walk to outperform those based on classical random walks. Indeed, polynomial speed-ups are standard and some exponential improvements have also been found. See Refs. [2, 3, 4, 5] for some nice introductions to quantum walks.

3.1 Discrete-Time Random Walk on a Line

Consider a one-dimensional (1D) lattice, which is a regular unweighted graph with each vertex connected to its neighbours on the left and right (except for the first and last vertices, of course!). Suppose that there is a walker that at every step will randomly choose between moving right or left. Maybe he flips a coin and moves one step to the right with probability p if it comes up heads, and left with probability $q = 1 - p$ otherwise. The question is: what is the probability of finding him m steps away after N steps are taken? Put another way, if he is a very precise drunk, so that the length of step is always exactly ℓ, then what is his mean displacement after N steps?

We know that $N = n_1 + n_2$, where n_1 and n_2 are the number of steps to the right and left, respectively. Evidently, the distance after N steps is given by the difference, $m = n_1 - n_2$. These two equations can be inverted, $n_1 = \frac{1}{2}(N+m)$ and $n_2 = \frac{1}{2}(N-m)$, so that the probability of being m steps away is

$$P_N(m) = \frac{N!}{n_1!n_2!}p^{n_1}q^{n_2} = \frac{N!}{\left[\frac{1}{2}(N+m)\right]!\left[\frac{1}{2}(N-m)\right]!}p^{\frac{1}{2}(N+m)}q^{\frac{1}{2}(N-m)}.$$

Let's calculate the mean number of steps taken to the right, and the standard deviation.

$$\begin{aligned}
\overline{n_1} = \sum_{n_1=0}^{N} p(n_1) n_1 &= \sum_{n_1=0}^{N} \frac{N!}{n_1!(N-n_1)!} p^{n_1} q^{N-n_1} n_1 && (1)\\
&= \sum_{n_1=0}^{N} \frac{N!}{n_1!(N-n_1)!} q^{N-n_1} \left[n_1 p^{n_1}\right] \\
&= \sum_{n_1=0}^{N} \frac{N!}{n_1!(N-n_1)!} q^{N-n_1} \left[p \frac{d}{dp}\left(p^{n_1}\right)\right] && (2)\\
&= p \frac{d}{dp} \sum_{n_1=0}^{N} \frac{N!}{n_1!(N-n_1)!} q^{N-n_1} p^{n_1} \\
&= p \frac{d}{dp} \left[(p+q)^N\right] \qquad \text{from binomial theorem} \\
&= pN(p+q)^{N-1} = pN. \qquad \text{since } p+q=1.
\end{aligned}$$

So the mean number of steps to the right is just the total number of steps times the probability of taking a step to the right. No surprises here! Obviously we also know that $\overline{n_2} = qN$ and so $\overline{m} = N(p-q)$.

Now let's calculate the RMS value $(\Delta n_1)^2 = \overline{n_1^2} - \overline{n_1}^2$. Because we already know $\overline{n_1}$, we only need to obtain $\overline{n_1^2}$. But instead of a factor of n_1 appearing in the expression (1), there is now a factor of n_1^2. It would be nice to use a derivative trick like in line (2) above. Using

$$\frac{d}{dp}\left(p^{n_1}\right) = n_1 p^{n_1-1} \quad \text{and} \quad \frac{d^2}{dp^2}\left(p^{n_1}\right) = n_1(n_1-1)p^{n_1-2}$$

one gets

$$p^2 \frac{d^2}{dp^2}\left(p^{n_1}\right) = (n_1^2 - n_1)p^{n_1} \quad \Rightarrow \quad n_1^2 p^{n_1} = p^2 \frac{d^2}{dp^2}\left(p^{n_1}\right) + n_1 p^{n_1}.$$

So right away the result can be written down:

$$\begin{aligned}
\overline{n_1^2} &= p^2 \frac{d^2}{dp^2}\left[(p+q)^N\right] + pN \\
&= p^2\left[N(N-1)(p+q)^{N-2}\right] + pN \\
&= p^2 N(N-1) + pN.
\end{aligned}$$

So

$$(\Delta n_1)^2 = \overline{n_1^2} - \overline{n_1}^2 = p^2N^2 - p^2N + pN - p^2N^2 = Np(1-p) = Npq.$$

The RMS value is therefore $\Delta n_1 = \sqrt{Npq}$. This means that after N steps, the mean displacement of the walker from his starting point will go like $\sqrt{N}$. Put another way, if there are $M+1$ vertices and he starts at the center, then he will take on average $(M/2)^2$ steps to reach either end.

3.2 Discrete-Time Quantum Walk on a Line

The *discrete-time quantum walk* on the line behaves much as the classical version, except that instead of the walker choosing to move right or left at each step, she moves in both directions simultaneously, with some probability amplitude. A simple approach is to allow the walker to carry a 'quantum coin,' which can be put into a superposition of heads and tails. So the walker needs to keep track

of both her coin coordinates, and her spatial coordinates. The coin states will be encoded in a spin degree of freedom, with $|0\rangle_C$ and $|1\rangle_C$ representing heads and tails, respectively. On a 1D lattice with $M+1$ points, her spatial coordinate will be encoded in the vector $|j\rangle_S$, $0 \le j \le M$. So, the total wavefunction can be written

$$|\psi\rangle = \sum_{\sigma=0}^{1} \sum_{j=0}^{M} \alpha_{\sigma j} |\sigma\rangle_C \otimes |j\rangle_S.$$

The quantum walk is effected by repeated application of the operator U, which consists of first applying the Flip operator F on the coin, and then shifting the walker with S in a direction conditional on the spin state: right if $|0\rangle_C$ and left if $|1\rangle_C$. For N steps, the resulting operator is $U = [S(F \otimes I)]^N$ where I is the M-dimensional identity operator and

$$F = [\cos(\theta)|0\rangle_C + \sin(\theta)|1\rangle_C] \langle 0|_C + [-\sin(\theta)|0\rangle_C + \cos(\theta)|1\rangle_C] \langle 1|_C;$$

$$S = \sum_i \left(|0\rangle_C\langle 0|_C \otimes |j+1\rangle_S\langle j|_S + |1\rangle_C\langle 1|_C \otimes |j-1\rangle_S\langle j|_S \right),$$

where θ is some angle defining the fairness of the coin, with $\theta = \pi/4$ giving the balanced coin. (Note that in 1D, a real coin operator is completely general, but this is not the case for more interesting graphs). Because the coin operator isn't really randomizing (the operation is perfectly unitary and the evolution of the wavefunction is coherent), I prefer to call this a 'quantum walk' rather than a 'quantum random walk.' The analysis of this discrete-time quantum walk is a bit involved [6], so I'll simply show the results in Fig. 4 for the random and quantum walks after 20 timesteps. In both of these cases, I chose a balanced coin and started the walks with full probability at the central vertex, and for the quantum case the initial coin state was chosen to be $|0\rangle_C + i|1\rangle_C$ which gives a nice symmetric pattern.

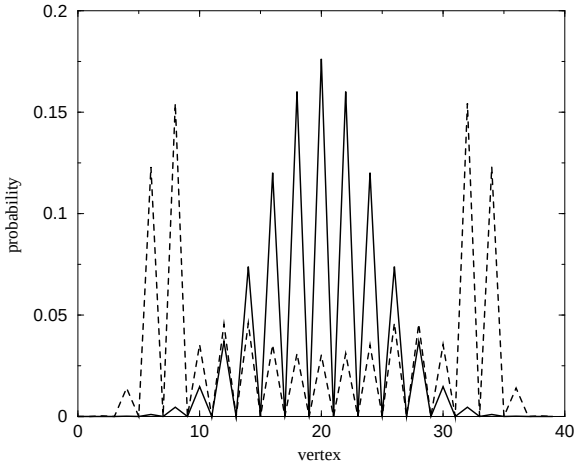

Figure 4: Comparison of the discrete-time random walk and quantum walk on a line. The solid and dashed lines show the probability of finding the classical and quantum walkers at a given vertex, respectively, after 20 timesteps.

It is clear that the probability distributions for the two walks are markedly different. The classical distribution remains peaked at the origin, and the half-width at half-max is around $5 \approx \sqrt{20}$ lattice spaces. In contrast, the quantum probability distribution is strongly peaked *away* from the center, and the spread is much greater. Fig. 5 compares the rms width of the resulting distributions as a function of the number of steps. While the classical rms value spreads like the square root of the number of steps so that $\Delta j \sim \sqrt{N}$ or alternatively $\Delta x \sim \sqrt{t}$ where x and t are position and time respectively, the quantum value spreads *linearly* in the number of steps, $\sqrt{\langle x^2 \rangle} \sim t$. Thus, in

principle the quantum walk provides a quadratic speed-up in the ability to access a given vertex over the classical random walk; that is, starting the walker from the central vertex, a quantum walk will give a high probability of hitting the vertex at either end quadratically faster than will a random walk.

This polynomial speed-up is a bit misleading, though. There is a classical strategy for reaching one of the end vertices starting from the center that also scales linearly with the number of steps. It is simply to move in only one direction! This is a cautionary note when discussing speed-ups using quantum walks: there can be a classical algorithm that looks nothing like a random walk that can still perform as well.

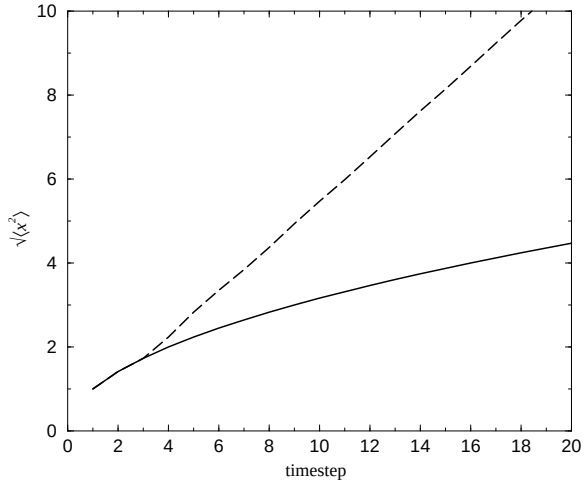

Figure 5: Comparison of the rms spread for the discrete-time random walk (solid line) and quantum walk (dashed line).

3.3 Continuous-Time Quantum Walks

The discrete-time walks described above have continuous-time versions that share much the same behaviour, but don't generally require a coin degree of freedom. If γ is the probability per unit time of hopping from a given vertex to one of its neighbours, then the time-evolution of the probabilities for the random walk is given by the following equation:

$$\frac{dp_j(t)}{dt} = \gamma \sum_{k=0}^{M} A_{jk} p_k(t).$$

Here, A_{jk} are the non-zero entries of the graph adjacency matrix, and conservation of probability requires that

$$\sum_j p_j(t) = 1 \quad \forall t.$$

It is easy to verify by direct simulation that the distribution of probabilities with time closely follows the behaviour shown in Fig. 4.

The continuous-time quantum walk has very much the same form. Given the wavefunction for some quantum particle $|\psi(t)\rangle$, the Schrödinger equation is

$$i\hbar \frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle,$$

where H is the *Hamiltonian* for the system. The Hamiltonian is the generator of time translations in classical mechanics, and is equal to the total energy operator for conservative systems (i.e. when

closed paths in Hilbert space do no work). The amplitude to find the particle on vertex $|j\rangle$ is simply $\langle j|\psi\rangle$; inserting a complete set using the identity $\sum_j |j\rangle\langle j| \equiv 1$, one then obtains

$$i\hbar \frac{d}{dt}\langle j|\psi(t)\rangle = \sum_k \langle j|H|k\rangle\langle k|\psi(t)\rangle. \tag{3}$$

But for discrete systems, the elements of the Hamiltonian effectively constitute the adjacency matrix of a graph, $\langle j|H|k\rangle \equiv -\tau A_{jk}$, where τ is a constant (for unweighted graphs) with the appropriate units of energy and length (according to the spatial dimension). Defining $\langle j|\psi(t)\rangle \equiv \psi_j(t)$, Eq. (3) can then be written in the following convenient form:

$$i\hbar \frac{d\psi_j(t)}{dt} = -\tau \sum_k A_{jk}\psi_k(t). \tag{4}$$

Thus, a continuous-time quantum walk on a graph is indistinguishable from ordinary Schrödinger time-evolution under the influence of some Hamiltonian/adjacency matrix. This implies that any efficient algorithm to simulate a Hamiltonian can be effected by a quantum walk over some graph, and vice versa.

3.4 Example: Quantum Walk Search Algorithm

To illustrate the power of quantum walks, I'll describe a way to implement Grover's algorithm [7] using quantum walks, designed to find with high probability a marked element of some unsorted database. The analysis is somewhat simpler to describe in terms of a continuous-time quantum walk [8, 9], though a discrete-time version is also possible [10]. It is well-known that Grover's algorithm gives a quadratic speed-up over a classical brute-force search, i.e. a marked item in a database with N elements can be found quantum mechanically using only $\sqrt{N}$ operations.

Consider the complete graph with N vertices, K_N. The adjacency matrix is given by

$$A_{ij} = \begin{cases} 1 & i \neq j \\ 0 & i = j \end{cases},$$

so that each vertex is connected to all others. The ground state of the associated Hamiltonian $H = -\tau A$ with $\tau < 0$ is the uniform distribution

$$|\psi_{\rm gs}\rangle = \frac{1}{\sqrt{N}} \sum_j |j\rangle.$$

This is because other than a factor of $(\tau/N)I$, the Hamiltonian itself is $H = -N\tau|\psi_{\rm gs}\rangle\langle\psi_{\rm gs}|$.

Now suppose that the vertex q is marked, and we would like to evolve the initial ground-state wavefunction under the influence of some Hamiltonian, so that $|\psi_{\rm gs}\rangle \to |q\rangle$ after some time t. In effect, we would like to turn the problem into one involving only two states, $|\psi_{\rm gs}\rangle$ and $|q\rangle$. It turns out that for the complete graph, one simply needs to use the following Hamiltonian

$$H = -\tau A - (\tau I - \tau I) - |q\rangle\langle q| = -N\tau|\psi_{\rm gs}\rangle\langle\psi_{\rm gs}| - |q\rangle\langle q| + \tau I.$$

The extra factor of τI just gives a constant shift to the energy and has no effect on the time-dynamics except to shift the wavefunction's phase. The solution requires inverting the Schrodinger equation (4), so that $\psi_j(t) = \exp(-iHt)\psi_j(0) \equiv U(t)\psi_j(0)$, where j is some state index. But to obtain an analytical solution for $U(t)$ we need to expand in an orthogonal basis, and the vectors $|\psi_{\rm gs}\rangle$ and $|q\rangle$ are not orthogonal; rather, $\langle\psi_{\rm gs}|q\rangle \equiv x = 1/\sqrt{N}$ for any k. To make further progress, we first need to use an orthogonal basis; let's define a vector $|p\rangle$ that is explicitly orthogonal to $|q\rangle$:

$$|p\rangle = \frac{1}{\sqrt{1-x^2}} \left(|\psi_{\rm gs}\rangle - x|q\rangle\right).$$

Now in (q, p) space, the Hamiltonian and initial state are respectively:

$$\frac{H}{(-\tau)} = \begin{pmatrix} 1+x^2 & x\sqrt{1-x^2} \\ x\sqrt{1-x^2} & 1-x^2 \end{pmatrix}; \qquad |\psi(0)\rangle = \begin{pmatrix} |q(0)\rangle \\ |p(0)\rangle \end{pmatrix} = \begin{pmatrix} x \\ \sqrt{1-x^2} \end{pmatrix}.$$

A simple way to find $U = e^{-iHt}$ is to diagonalize H, i.e. find V such that $VHV^\dagger = D$ where D is the (diagonal) matrix of the eigenvalues λ of H. Then, $H = V^\dagger DV$ and $U = V^\dagger e^{-iDt}V$. It's easy to find that $\lambda = 1 \pm x$ and

$$V = \frac{1}{\sqrt{2}} \begin{pmatrix} \sqrt{1+x} & -\sqrt{1-x} \\ \sqrt{1-x} & \sqrt{1+x} \end{pmatrix}.$$

We quickly obtain

$$\begin{aligned} \begin{pmatrix} |q(t)\rangle \\ |p(t)\rangle \end{pmatrix} &= \begin{pmatrix} \cos(xt) - ix\sin(xt) & -i\sqrt{1-x^2}\sin(xt) \\ -i\sqrt{1-x^2}\sin(xt) & \cos(xt) + ix\sin(xt) \end{pmatrix} \begin{pmatrix} |q(0)\rangle \\ |p(0)\rangle \end{pmatrix}. \\ &= \begin{pmatrix} x\cos(xt) - i\sin(xt) \\ \sqrt{1-x^2}\cos(xt) \end{pmatrix}. \end{aligned}$$

So, the probability of being in the marked state $\langle q(t)|q(t)\rangle = x^2\cos^2(xt) + \sin^2(xt)$ is exactly one when $t = \pi/2x = \pi\sqrt{N}/2$, so that the search time scales like $\sqrt{N}$ as hoped.

While similar algorithms have been devised for other kinds of graphs, the efficiency with which the marked state can be found varies considerably. For example, quantum search algorithms for elements on a line give no speed-up over classical methods, since sites can be found using $\sqrt{N}$ steps, but $\sqrt{N}$ iterations are needed to determine if the result is correct. In fact, unless special tricks are used, a regular lattice needs to have dimension greater than 3 for a quantum algorithm to give any advantage over brute-force searching [9].

3.5 Example: Glued-Trees Graph

Quantum walks also have the capability to give an exponential speed-up over classical methods for certain kinds of problems. One class of these can be phrased as the problem of finding the 'exit' vertex of some graph (this will be clearer below), given full initial amplitude on some 'entrance' vertex. Consider again the 1D lattice with $M + 1$ sites, with the entrance vertex the first site on the left, and the exit the last site on the right. A quantum walk will hit the rightmost vertex in a time $t \propto M$ with probability decreasing polynomially with M (more specifically, like $M^{-1/2}$), while a classical walker will hit the rightmost vertex in a time $t \propto M^2$.

The so-called 'glued-trees' graph [11], shown in Fig. 6, provides an example of an exponential separation between classical and quantum behaviour. Without randomization (left image), a classical random walker will take a time to reach the exit vertex that grows exponentially with the graph width, because of the exponential growth in the number of vertices. A quantum walker, in contrast, is effectively able to take all directions simultaneously. In effect, the quantum walk on the original graph maps onto a quantum walk on a 1D line (with a defect), which can be traversed in linear time. That said, there are again efficient classical strategies that allow traversal of the glued-trees graph in time that grows only polynomially with the width; the random walker accomplishes this by checking the vertex degree at each step.

To foil this method, it suffices to randomize the connections at the graph center [12], shown on the right of Fig. 6. An intelligent random walker will rapidly find its way toward the center of the graph, since it has two ways to move right and only one to move left. But once there, it will get lost in the maze of random connections and will take an exponentially increasing time to find the way out as a function of the tree-depth. One can phrase the attempt to hit the exit vertex as an *oracle problem*. Suppose you are given the label of the entrance vertex, and your task is to find the label of the exit vertex. At each vertex, you can ask an oracle for the names of the neighbouring vertices. Success requires an exponential number of queries using the optimal classical algorithm,

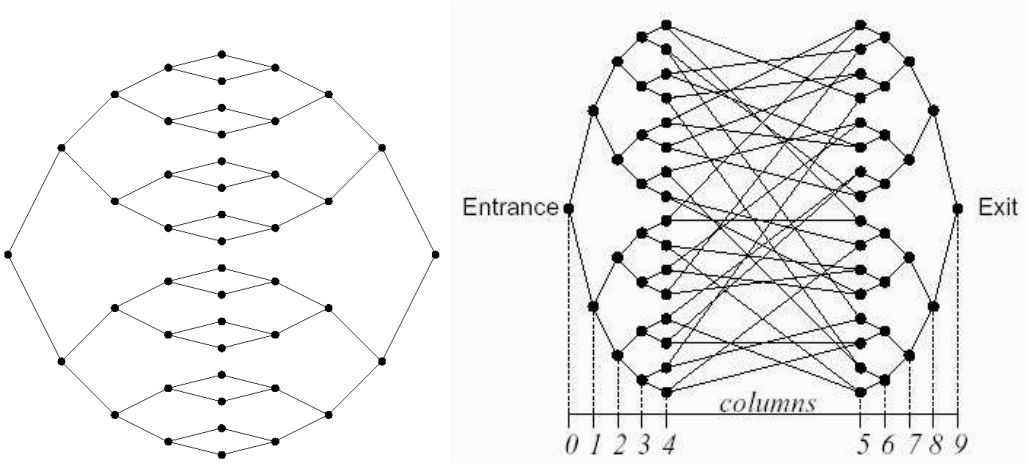

Figure 6: The glued-trees graph without randomization (left, from Ref. [3]) and with randomization (from Ref. [6]).

but only a polynomial number using a quantum walk. Note that this isn't the first example of an exponential improvement using a quantum algorithm; just about anything based on a quantum Fourier transform, such as Shor's factoring algorithm [13], gives this improvement.

4 Graph States and Stabilizers

Let's switch gears, to see another way to think about quantum graphs. Instead of quantum particles traversing edges between vertices, now the vertices themselves will correspond to qubits (generally qudits, but I'll restrict the story to qubits for the purposes of these notes), and the edges correspond to entangling operations between qubits. This will turn out to be a powerful way to think about highly entangled pure and mixed states, and will lead ultimately to a new approach to quantum computing, based entirely on measurements of vertices in graphs. For an introduction to graph-state theory see Ref. [14]; for stabilizer theory, I recommend Dan Gottesman's Ph.D. thesis [15].

4.1 Graph States

To be specific, let each of the N vertices of a given graph be initialized in the $|+\rangle$ state, so that

$$|\psi_{\text{init}}\rangle = |+\rangle^{\otimes N} = \left[\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)\right]^{\otimes N} = \bigotimes_{j=1}^{N} |+\rangle_j.$$

Now two-qubit controlled-phase operations, otherwise known as CZ gates, are applied between qubits j and k, making vertex k the 'neighbour' of vertex i. Recall that the CZ gate is defined as

$$\text{CZ} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} = \sum_{j,k\in\{0,1\}} (-1)^{jk} |jk\rangle\langle jk|. \tag{5}$$

The simplest non-trivial graph one can construct this way has two vertices that share one edge. The initial state is $|\psi\rangle = |+\rangle \otimes |+\rangle \equiv |++\rangle$ and after the entangling gate the result is

$$|\psi_-\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle - |11\rangle\right), \tag{6}$$

which is shown in Fig. 7.

Figure 7: The two-vertex graph state is generated (a) by initializing each qubit in the $|+\rangle$ state and then operating with a CZ gate, yielding (b).

Note that if we apply a Hadamard gate to either qubits 1 or 2, the result is a Bell state:

$$|\psi_{\rm Bell}\rangle = H_1|\psi'\rangle = H\otimes I|\psi'\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\otimes\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) = H_2|\psi'\rangle.$$

Now suppose that we position three qubits in an equilateral triangle and apply a CZ to qubits 1 and 2 and also to qubits 2 and 3; in this case we obtain the 'Λ' graph and the state

$$|\psi_\Lambda\rangle = \frac{1}{\sqrt{8}}\left(|000\rangle + |001\rangle + |010\rangle - |011\rangle + |100\rangle + |101\rangle - |110\rangle + |111\rangle\right). \tag{7}$$

If we further apply a CZ between qubits 1 and 3 we obtain the graph K_3 and the state

$$|\psi_\triangle\rangle = \frac{1}{\sqrt{8}}\left(|000\rangle + |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle - |110\rangle - |111\rangle\right). \tag{8}$$

Both of these graphs are shown in Fig. 8. If we subsequently applied a Hadamard to qubits 1 and 3 of Λ, we would obtain the GHZ_3 state $\left(|000\rangle + |111\rangle\right)/\sqrt{2}$ which looks much simpler. (Note that the transformation from △ to GHZ_3 is a combination of this and other 'local complementation' operations, discussed below). Don't be fooled into thinking that we will always make GHZ_N for N qubits this way, however! For example, for four qubits the resulting state is $\left(|0000\rangle + |0011\rangle + |1100\rangle - |1111\rangle\right)/2$.

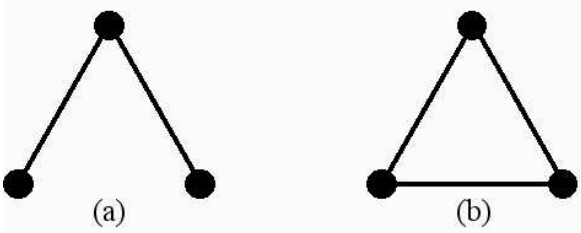

Figure 8: The Λ (a) and △ (b) graphs are the only two three-vertex fully connected graphs.

In any case, if we don't apply any Hadamards after the entangling gates, then for many qubits it rapidly becomes tedious to explicitly write down the various graph states. Thankfully, there are two very compact ways to express them. The first is to use the so-called 'quadratic form' which is based on the graph adjacency matrix; for the graphs (7) and (8), the adjacency matrices are respectively

$$A_\Lambda = \begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 0\end{pmatrix}; \qquad A_\triangle = \begin{pmatrix}0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 0\end{pmatrix}. \tag{9}$$

The corresponding states are written

$$|\psi_\Lambda\rangle = \sum_{\mathbf{x}\in\{0,1\}^{\otimes 3}} (-1)^{x_1x_2+x_2x_3}|x_1x_2x_3\rangle; \qquad |\psi_\triangle\rangle = \sum_{\mathbf{x}\in\{0,1\}^{\otimes 3}} (-1)^{x_1x_2+x_2x_3+x_1x_3}|x_1x_2x_3\rangle.$$

These expressions become obvious when comparing with the definition of the CZ operator (5). So, in general the graph states can be represented as

$$|\psi_{\rm graph}\rangle = \sum_{\mathbf{x}\in\{0,1\}^{\otimes N}} \exp\left(-i\pi\sum_{i=1}^{N}\sum_{j>i} A_{ij}x_ix_j\right)|\mathbf{x}\rangle$$

But this still seems a bit unwieldy. A more illuminating approach is to use the stabilizer formalism, discussed in the next section.

4.2 Stabilizers

A stabilizer of a state $|\psi\rangle$ is a fancy word for a set of operators, all of which have $|\psi\rangle$ as an eigenvector with eigenvalue $+1$. For example, the state $|+\rangle$ is the eigenvector of both the X and I operators with eigenvalue $+1$, so the group $\{X, I\}$ is the stabilizer for $|+\rangle$. Likewise, the stabilizer for the product state $|++\rangle$ is $S = \{X\otimes X, X\otimes I, I\otimes X, I\otimes I\} \equiv \{XX, XI, IX, II\}$. Before I get too far I should pause to remind you what the Pauli matrices are:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Any single-qubit unitary can be constructed from a linear combination of these X, Y, and Z, but the eigenvectors of these latter Pauli matrices are different from $|+\rangle$. This means that only X and I stabilize $|+\rangle$; likewise, an arbitrary single-qubit state $|\varphi\rangle = U|+\rangle$ also has only two elements in its stabilizer, which are I and $UXU^\dagger$.

Things get more interesting when the states are entangled. Consider again the two-qubit graph state $|\psi_-\rangle$. We know that

$$|\psi_-\rangle = \mathrm{CZ}|++\rangle = \mathrm{CZ}(XX)|++\rangle = \mathrm{CZ}(XI)|++\rangle = \mathrm{CZ}(IX)|++\rangle = \mathrm{CZ}(II)|++\rangle,$$

so to obtain the stabilizer for $|\psi_-\rangle$ it is nice to know that $\mathrm{CZ}(XX) = (YY)\mathrm{CZ}$, $\mathrm{CZ}(XI) = (XZ)\mathrm{CZ}$, and $\mathrm{CZ}(IX) = (ZX)\mathrm{CZ}$. So immediately we obtain the stabilizer for $|\psi_-\rangle$ as $S_- = \{XZ, ZX, YY, II\}$, a result that can be easily checked by multiplying any of these operators on the ket. More concisely, if we know that some N-qubit unitary U (which may be separable or not) transforms the state $|\psi\rangle$ with stabilizer S into the state $|\psi'\rangle$ with stabilizer S', then $S' = USU^\dagger$. For example, because $|\psi_{\mathrm{Bell}}\rangle$ is obtained by applying a Hadamard on (say) the first qubit, the stabilizer for this state is transformed by $(HI)\{XZ, ZX, YY, II\}(HI)^\dagger$ yielding the set $S_{\mathrm{Bell}} = \{ZZ, XX, -YY, II\}$.

Though I haven't bothered to prove it formally, it should be evident that any N-qubit pure state has a stabilizer containing 2^N commuting elements. The advantage of this formalism is that, while the stabilizer is often not unique (there can be many different choices for U that yield the same final state), each 2^N-element stabilizer nevertheless uniquely defines the N-qubit state. In general, however, the stabilizer elements will not be separable (i.e. written as a tensor product of single-qubit unitaries). An obvious disadvantage is that enumerating these is at least as tedious as writing down the original state, which generally has 2^N different coefficients. For the two-qubit examples above, however, you may have noticed that two of the elements in each stabilizer (one of which is the identity matrix) can be derived from the other two elements by multiplication. In fact, one can treat the stabilizer as a group, whose elements can be generated by a subset of elements called the *generators*. For N qubits, exactly N generators (not including the identity) are required to produce all 2^N elements.

For the two-qubit graph state $|\psi_-\rangle$ defined in (6), we could choose the generators to be $G_- = \{XZ, ZX\}$. Likewise for the graph states $|\psi_\Lambda\rangle$ defined in (7) and $|\psi_\triangle\rangle$ defined in (8) we could choose $G_\Lambda = \{XZI, ZXZ, IZX\}$ and $G_\triangle = \{XZZ, ZXZ, ZZX\}$, respectively. These follow directly from the ways that the (mutually commuting) CZ operators transform the original X stabilizers. Comparison of these with the adjacency matrices given in (9) leads to an immediate connection between the adjacency matrix of a graph and its stabilizer generators. Here is the rule:

- Replace all the diagonal elements of the adjacency matrix A with an X.
- Replace each element in A with value 1 by a Z.
- Replace each off-diagonal element in A with value 0 by an I.

When you are done, each row of the adjacency matrix looks like a stabilizer generator. To be concise, the elements M_j of the generator $G = \{M_j | j = 1, \ldots, N\}$ are all in the form

$$M_j = X_j \bigotimes_{k=\mathcal{N}(j)} Z_k,$$

where $\mathcal{N}(j)$ denotes 'the neighbourhood/neighbours of vertex j.'

4.3 Local Complementation

One of the many useful decomposable (i.e. lots-o-single-qubit or multi-local or separable) unitary operations on quantum graphs such as those described above is *local complementation* (LC). In mathematical terms, LC about the vertex j about is accomplished by the transformation

$$LC_j = \sqrt{X_j} \bigotimes_{k=\mathcal{N}(j)} \sqrt{Z_k} \qquad \sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & \pm i \end{pmatrix}; \quad \sqrt{X} = \sqrt{\frac{i}{2}} \begin{pmatrix} -i & 1 \\ 1 & -i \end{pmatrix}.$$

Evidently, $(LC_j)^2|\psi\rangle = |\psi\rangle$, so that two successive LC's bring the graph state back to itself. The sudden appearance of square-roots of Pauli matrices might seem odd, but in fact these are two operators that belong to the *local Clifford group*, which is a set of single-qubit operators that maps the set of Pauli matrices to itself. In particular, they give rise to the following useful identities:

$$\sqrt{Z}X(\sqrt{Z})^\dagger = Y; \quad \sqrt{Z}Y(\sqrt{Z})^\dagger = -X; \quad \sqrt{X}Y(\sqrt{X})^\dagger = -Z; \quad \sqrt{X}Z(\sqrt{X})^\dagger = Y.$$

Let's apply local complementation on qubit 2 of the Λ graph to illustrate what happens to the stabilizer:

$$\begin{aligned} & \sqrt{Z}\sqrt{X}\sqrt{Z}\{XZI, ZXZ, IZX, YYZ, XIX, ZYY, -YXY, III\} \left(\sqrt{Z}\sqrt{X}\sqrt{Z}\right)^\dagger \\ = \; & \{YYI, ZXZ, IYY, XZZ, YIY, ZZX, -XXX, III\} \\ = \; & \{XZZ, ZXZ, ZZX, YYI, YIY, IYY, -XXX, III\}, \end{aligned}$$

where the second line is the result of the transformation, and the third line is the stabilizer of the $\triangle$ graph. Apparently miraculously, a series of completely local operations has transformed a graph with two edges into a graph with three, as if we had explicitly applied another CZ gate between qubits 2 and 3!

In graph theory, this particular multi-local unitary LC_j corresponds to *complementing the neighbourhood of vertex j*. Here's the recipe that doesn't require any math. First, determine the neighbours of vertex j, i.e. the set of all vertices connected to j by an edge, denoted $\{\mathcal{N}(j)\}$. Second, if an edge already connects two vertices in $\{\mathcal{N}(j)\}$, remove it; if no edge connects them, then add it. In the example above, $\mathcal{N}(2) = \{1, 3\}$, but qubits 1 and 3 have no edge between them, so we add it to make the $\triangle$ graph. Because local unitary operations cannot change the entanglement of a given state, the LC operation implies that complicated graph states can potentially be derived from a small number of entangling operations. Conversely, there is a minimum number of entangling operations needed to create any given graph state; unfortunately, for general graphs this number is not necessarily easy to compute.

It turns out that for N-qubits there are a number of fully connected graphs (i.e. no vertices without a neighbour, and the graph is not separable into disconnected subgraphs) that cannot be mapped to one another through the local complementation. It is an open question if there exists some other LU outside of the Clifford group that is able to effect the transformation. For $N = 3$, there are 4 distinct fully connected graphs, namely Λ and rotations of this by $2\pi/3$ and $4\pi/3$, and $\triangle$. Clearly, LC can connect all of these. For $N = 4$ there are a total of 38 distinct fully connected graphs (by my count anyhow!). Of these, I obtain 4 classes of graphs that are not convertible by LC. What are they?

Solution:

If I allow qubit positions to be swapped, i.e. where I interchange the roles of qubits j and k through the SWAP gate

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

then I can group together graphs that are *isomorphic* to one another. How many classes of 4-qubit graphs remain non-convertible under the combined operations of SWAP and LC?

Solution:

Things rapidly get worse than this as N increases.

One of the key advantages of graph states is that they allow the effects of complicated manipulations on quantum states to be directly visualized. For example, suppose that Alice and Bob each hold one qubit, but wanted these to be entangled with each other. Unfortunately, they are too far away to directly entangle their qubits. One way to accomplish this is for Charlie to create a two-qubit graph state, and then for him to *lend* one of these qubits to Alice and the other to Bob. We could stop there, because now Alice and Bob share entanglement, but suppose that Charlie insists that his original qubits come back (this is getting a bit artificial, I know!). Now, Alice can entangle her two qubits with a CZ, and Bob can do the same with his two qubits. The result is the first graph depicted in Fig. 9. Once this is done, a series of LC operations will transform the initial graph into the final one shown in this figure. Now Alice and Bob can each disentangle the qubits they borrowed from Charlie by again operating with CZ's, and send them back. Voila!

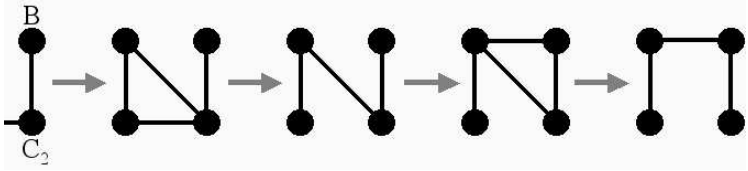

Figure 9: Entanglement transfer by local complementation. The vertex labels shown for the first graph are the same for all others; A stands for Alice, B for Bob, and C_1 and C_2 are the two qubits donated by Charlie.

Even more interesting are scenarios where the various parties are also allowed to make measurements (projective or otherwise); in fact, such non-unitary operations are central to a way to perform universal quantum algorithms using graph states, to be discussed in Section 6. But first, let's discuss quantum error correction using stabilizers.

5 Quantum Error Correcting Codes

One of the many powerful applications of graph or stabilizer states is quantum error correction. For a nice discussion, see Refs. [15] and [16]. Consider a noisy channel through which one wanted to send a single qubit. Suppose that with some probability, this noise would tend to cause a bit flip, i.e. turning $|0\rangle$ into $|1\rangle$ and vice versa, or $|\psi\rangle \rightarrow X|\psi\rangle$. How can we diagnose if a bit flip occurred, and fix it? Unfortunately, we can't just measure the qubit, because doing so destroys the state.

A simple approach is to entangle the qubit of interest (qubit 1) in the state $|\psi\rangle = a|0\rangle + b|1\rangle$ with two *ancillary qubits*, labelled qubits 2 and 3. If the ancillae are both initialized in $|0\rangle$, then performing a CNOT gate

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

between qubits 1 and 2, and another between 1 and 3, yields the encoded state $|\psi'\rangle = a|000\rangle + b|111\rangle$. Now, if a single bit is flipped, the state will be transformed into one of $a|100\rangle + b|011\rangle$, $a|010\rangle + b|101\rangle$, or $a|001\rangle + b|110\rangle$. Which of these three errors occurred (if any) can be checked using a *syndrome measurement*, without destroying the initial state. In this example, the syndrome measurement corresponds to the projective measurements ZZI and IZZ. To see how it works, let's write out ZZ explicitly:

$$\begin{aligned} ZZ &= Z \otimes Z = (|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes (|0\rangle\langle 0| - |1\rangle\langle 1|) \\ &= |00\rangle\langle 00| - |01\rangle\langle 01| - |10\rangle\langle 10| + |11\rangle\langle 11| \\ &= (|00\rangle\langle 00| + |11\rangle\langle 11|) - (|01\rangle\langle 01| + |10\rangle\langle 10|)\,. \end{aligned} \tag{10}$$

Thus, a measurement of ZZI (accomplished mathematically by the taking the expectation value $\langle\psi'|ZZI|\psi'\rangle$) will give $+1$ if qubits 1 and 2 are the same, corresponding to the left parenthesis of Eq. (10), and -1 if they are different (right parenthesis); likewise IZZ for qubits 2 and 3. After taking both measurements, correcting the error is simply a matter of flipping the appropriate bit.

How does this example relate to stabilizers? The generators for the state $|\psi'\rangle = a|000\rangle + b|111\rangle$ are $G_{|\psi'\rangle} = \{ZZI, IZZ, ZIZ\}$. If an XII error occurs, the generators are transformed into $G'_{|\psi'\rangle} = \{-ZZI, IZZ, -ZIZ\}$; likewise, errors IXI and IIX yield $G'_{|\psi'\rangle} = \{-ZZI, -IZZ, ZIZ\}$ and $G'_{|\psi'\rangle} = \{ZZI, -IZZ, -ZIZ\}$, respectively. Thus, measuring only two of the three stabilizer generators unambiguously yields the error syndrome, because $\langle\psi'|M_j|\psi'\rangle = \pm 1$ depending on the nature (or absence) of the error.

You might still be puzzled about how we perform the projective measurement without destroying the initial state. In practice, only the *ancillae* are actually measured to diagnose the error syndrome, so let's look at how things work more closely. The stabilizer for the initial state $|\psi\rangle \otimes |00\rangle$ is $\{IIZ, IZI, IZZ, III\}$ where an I remains in position 1 because $|\psi\rangle$ is unknown. The two CNOTs convert each of these to the stabilizer $\{IZZ, ZZI, ZIZ, III\}$. If there is an X error on qubit 1, this introduces -1 prefactors to the second and third elements, so after *removing* the entanglement with the ancillae the original stabilizer becomes $\{IIZ, -IZI, -IZZ, III\}$. Thus the stabilizer for the two ancillae is $\{IZ, -ZI, -ZZ, II\}$ which defines the state $|10\rangle$. For X errors on qubits 2 and 3, their state becomes $|11\rangle$ and $|01\rangle$, respectively, and for no error it remains in $|00\rangle$. So, computational-basis measurements of the two ancillae yield four bits of information, revealing the error syndrome without having to measure the qubit of interest.

In general, quantum states can suffer from an error that takes the form of a general single-qubit unitary U. Because we can write U as a linear combination of I, X, Y, and Z, a generic error can be thought of as some combination of bit flips (X), phase flips (Z), or both (Y). To correct phase flips requires a trivial modification of the protocol above. Since the errors are now Z's rather X's, we simply need to construct the state with stabilizer generators $G = \{XXI, IXX, XIX\}$ instead. But this is simply a transformation of each qubit from the Z (computational) basis to the X basis, which is accomplished by application of Hadamards.

So, to protect against an arbitrary error, one needs at most 9 qubits: 3 qubits standing for one to protect against phase-flip errors, and each of these made three-fold degenerate to protect against bit-flip errors. The result is the *Shor code*, which is an example of *concatenation*. The 'codewords' are:

$$|0\rangle \to \frac{1}{\sqrt{2}}\left(|000\rangle + |111\rangle\right)\left(|000\rangle + |111\rangle\right)\left(|000\rangle + |111\rangle\right);$$

$$|1\rangle \to \frac{1}{\sqrt{2}}\left(|000\rangle - |111\rangle\right)\left(|000\rangle - |111\rangle\right)\left(|000\rangle - |111\rangle\right).$$

Now, recall that any N-qubit pure state can be uniquely defined by a stabilizer with N generators and 2^N elements. But for the Shor code, we would like to define *two* states/codewords that share the same stabilizer. It turns out that for N qubits, a stabilizer with $N-k$ generators will stabilize 2^k orthogonal states, so the Shor code requires 8 generators. These turn out to be:

$$\begin{aligned} G_{\text{Shor}} &= \{ZZIIIIIII, IZZIIIIII, IIIZZIIII, IIIIZZIII, IIIIIIZZI, IIIIIIIZZ, \\ &\quad XXXXXXIII, IIIXXXXXX\}. \end{aligned}$$

You should convince yourself that this code indeed stabilizes the two Shor codewords, and furthermore that transforming the stabilizer by any single Pauli matrix changes the eigenvalue of at least one of the elements to -1.

What is the smallest number of qubits required to diagnose and fix arbitrary errors? Each qubit can suffer from three kinds of Pauli errors, so the total number of syndromes is $3N+1$ including the identity. Measurements of the $N-1$ ancillae give 2^{N-1} possible outputs, so we require $2^{N-1} \geq 3N+1$. The equality is first satisfied for $N = 5$, corresponding to the *Steane code*. The stabilizer generator is

$$G_{\text{Steane}} = \{XZZXI, ZZXIX, ZXIXZ, XIXZZ\} \equiv \{M_1, M_2, M_3, M_4\}.$$

Together with the logical $X \equiv M_5 = XXXXX$ operator, which commutes with the generators, we obtain a unique state that must be LC-equivalent to a graph state. Indeed, the generators

$$\begin{aligned} G'_{\text{Steane}} &= \{M_5M_2M_3, M_2M_1M_5, M_4M_5M_2M_3, M_1M_5M_2M_3, M_4M_3M_5\} \\ &= \{XZIIZ, ZXZII, IZXZI, IIZXZ, ZIIZX\} \end{aligned}$$

define the pentagon graph! Simply removing one of the generators from the above list define two codewords that can encode the $|0\rangle$ and $|1\rangle$ computational states. It is straightforward to verify that this code is robust against arbitrary errors, but apart from its obvious symmetry it is a bit mysterious why this particular 5-vertex graph has this useful property. Why not K_5, for example? To get a better understanding, it is important to explore more deeply the process of making measurements on vertices in a graph.

6 One-Way Quantum Computing

Two approaches to universal quantum computation, based entirely on measurements, were developed more or less simultaneously about five years ago [17, 18]. The first of these is commonly referred to as Linear Optical Quantum Computation, or KLM after the first initials of the three authors. The second is called *one-way quantum computation* and *cluster-state quantum computation*, the former because of the irreversibility of the measurement process, and the latter because the fundamental

resource used is a highly entangled 'cluster' of states. I will focus on the one-way model for lack of time (and expertise!). For a recent review of this, see Ref. [19].

Consider again the two-qubit graph state, Fig. 7. Suppose now that qubit 1 is rotated into the state $|\psi_1\rangle = |\psi_{\rm in}\rangle = a|0\rangle + b|1\rangle$, so that the two-qubit state becomes $a|00\rangle + a|01\rangle + b|10\rangle - b|11\rangle$. If we now measure qubit 1 in the ξ-basis, defined by orthonormal eigenvectors in the XY plane

$$|\xi\pm\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle \pm e^{-i\xi}|1\rangle\right),$$

then the resulting state $|\psi_{\rm out}\rangle$ on qubit 2 becomes

$$|\psi_{\rm out}\rangle = X^m H R_z(\xi)|\psi_{\rm in}\rangle$$

up to an overall phase. Here,

$$R_z(\xi) = \begin{pmatrix} e^{-i\xi/2} & 0 \\ 0 & e^{i\xi/2} \end{pmatrix}$$

is the Z-rotation operator and $m \in \{0,1\}$ is the measurement result. The cluster-state representation and circuit diagram for this teleportation protocol are shown in Fig. 10. The presence of the X for outcome $m = 1$ is known as a *byproduct operator*. This is perhaps the simplest way of implementing quantum teleportation; if Alice and Bob hold qubits 1 and 2, respectively, and Alice measures her qubit, then Bob can recover $|\psi_{\rm in}\rangle$ by applying an appropriate unitary to his qubit if Alice tells him her measurement angle and result. A computational (Z-basis) measurement, in contrast, completely destroys the information, yielding $Z^m|+\rangle$ on qubit 2.

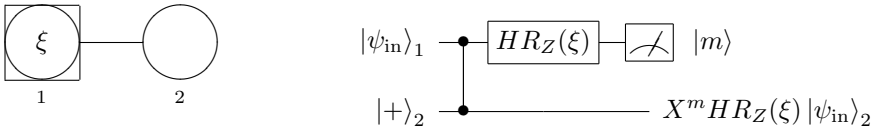

Figure 10: The one-bit teleportation protocol with the graph-state representation (left) and the equivalent quantum circuit (right).

If the initial graph were a linear chain of qubits, then the ξ-basis measurement would teleport the state of qubit 1 and the attendant operators to qubit 2, leaving all other qubits unchanged; one could then repeat the measurement with a different choice of angle ξ_2, and so on. After three teleportations the result is

$$\begin{aligned} |\psi_{\rm out}\rangle &= X^{m_3} H R_z(\xi_3) X^{m_2} H R_z(\xi_2) X^{m_1} H R_z(\xi_1)|\psi_{\rm in}\rangle \\ &= X^{m_3} Z^{m_2} H R_z\left[(-1)^{m_2}\xi_3\right] Z^{m_1} H R_z\left[(-1)^{m_1}\xi_2\right] H R_z(\xi_1)|\psi_{\rm in}\rangle \\ &= X^{m_3} Z^{m_2} X^{m_1} H R_z\left[(-1)^{m_2}\xi_3\right] R_x\left[(-1)^{m_1}\xi_2\right] R_z(\xi_1)|\psi_{\rm in}\rangle, \end{aligned}$$

where I have made use of the identities $R_z(\xi)X = XR_z(-\xi)$, $HX = ZH$, $ZR_z(\xi) = R_z(\xi)Z$, and $HR_z(\xi)H = R_x(\xi)$. The three angles ξ_1, ξ_2, and ξ_3 serve as Euler angles, able to rotate a vector on the Bloch sphere in an arbitrary direction and therefore able to simulate an arbitrary single-qubit unitary. Note that the presence of byproduct operators means that the angles have to be adjusted during the computation to effect the desired unitary!

To summarize the results above, a linear chain of qubits is the real-space analog of a single wire of a quantum circuit, except that time on the horizontal axis is replaced by space. Each computational qubit is mapped to a string of physical qubits, and single-qubit unitaries are replaced by concatenated teleportations. The non-unitary evolution of the total quantum graph because of measurements distinguishes one-way quantum computation from most other methods that are quantum circuit-equivalent.

Of course, computational universality also requires two-computational-qubit gates. To build this in we need the other crucial graph, which is Λ with the apical vertex initialized in $|\psi_{\rm in}\rangle = a|0\rangle + b|1\rangle$.

A bit of algebra reveals that measuring the apical vertex yields the state

$$|\psi_{\rm out}\rangle = [X^m H R_z(\xi) \otimes I]\, CZ |\psi_{\rm in}\rangle \otimes |+\rangle = [I \otimes X^m H R_z(\xi)]\, CZ |+\rangle \otimes |\psi_{\rm in}\rangle.$$

In graph language, measuring the apical vertex in the ξ-basis produces a two-qubit graph where the unitary transformed input state can be chosen to be on either vertex. This leads to the following useful fact. If there are two parallel linear graphs each representing a different computational qubit that are connected by a single vertical link, then this link would act like a $(U_1 \otimes U_2)CZ|\psi_1\rangle|\psi_2\rangle$. This process is depicted in Fig. 11. We have therefore obtained a two-qubit gate that yields computational universality when used together with single-qubit rotations.

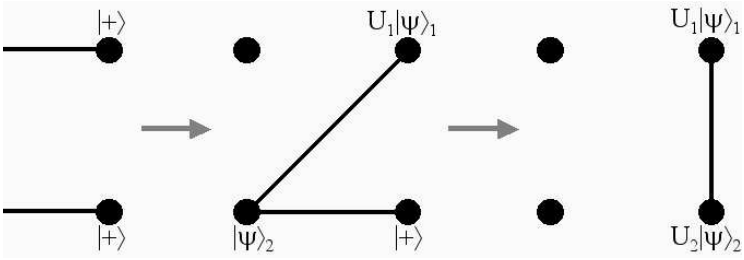

Figure 11: The graph showing how to implement a two-qubit gate in the one-way model.

In contrast, a computational-basis measurement of the apical vertex of Λ completely breaks the links to the other two vertices and all information is lost. The same result holds for a vertex of arbitrarily high degree. For example, K_5 is LC equivalent to the 'star graph,' where one qubit has degree 5 and the others degree 1. This in turn is LU-equivalent to GHZ_5. So it should come as no surprise that a computational measurement of any vertex should completely disentangle the state: by LC, any of the vertices can be made to be the sole degree-5 vertex. This simple observation might begin to reveal why the Steane graph is so special: because all the vertices have small degree 2, it is a stabilizer state that is perhaps the least susceptible to errors caused by inadvertent measurements.

Now that we have all the tools for universal computation, it is time to introduce the *cluster state*, shown in Fig. 12. The cluster state is initially prepared as a 2D square graph-state, where all qubits except for those on the edges are degree four. Once this has been prepared, an algorithm-specific cluster state can be prepared by selectively removing qubits from the graph via Z-measurements, as shown in Fig. 13. At this point, the cluster state looks much like a real-space representation of a quantum circuit, with each horizontal chain designating the computational qubits, and each vertical link effecting a CZ gate.

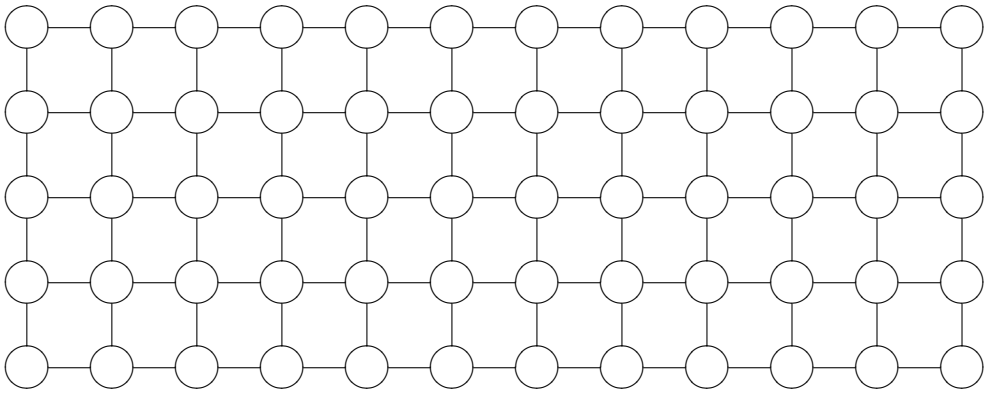

Figure 12: The initial cluster state.

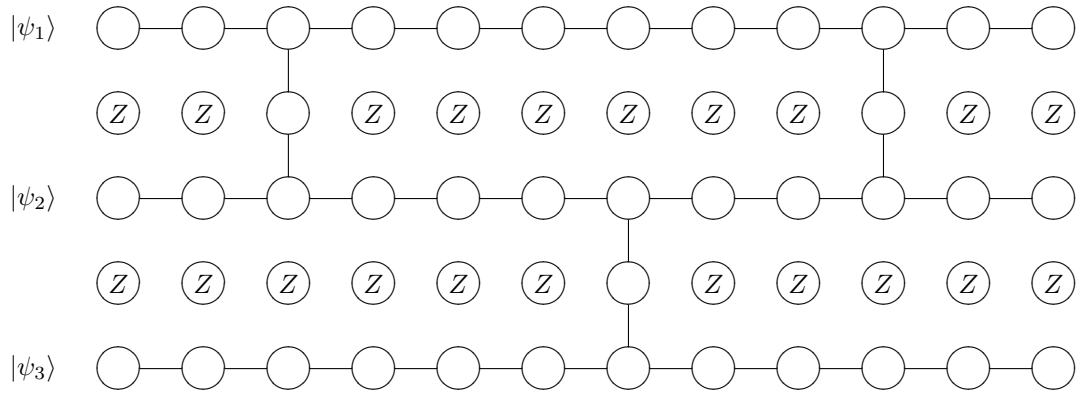

Figure 13: The computation-specific cluster state.

The cluster-state model has several advantages over the traditional unitary-based circuit model. First and foremost, all of the entangling gates are done in a single step before any computation. In many physical systems, two-qubit interactions are difficult to perform on specific qubits during the computation without great effort, and in the one-way model this problem is eliminated. Furthermore, in these same systems it is relatively straightforward to entangle all of the qubits simultaneously. Second, all of the gates contained in the Clifford group can be done in advance, since they do not teleport gates or yield byproduct unitaries. Third, for systems where physical qubits are not long-lived (such as photons), one need only manufacture small parts of the cluster that are directly involved in the next teleportation; and if a given entangling operation to make new cluster sections fails, one can repeat the process until success without affecting the algorithm.

There are several notable disadvantages to this model as well. First, if the initial cluster is not perfectly formed (say the entanglement introduces random or systematic errors), errors quickly propagate after several teleportations, severely reducing output fidelity. Second, standard fault-tolerant approaches are tricky to implement unless the cluster geometry is greatly modified. Third, the square geometry may not be ideally suited for certain important multiqubit operations, such as the three-qubit Fredkin and Toffoli gates, and the multi-qubit quantum fanout gate.

Acknowledgments

I am grateful to Mike Garrett for his help with the figures and proofreading these lecture notes. Thanks also to Nathan Babcock for his insightful comments on error correcting codes.

References

[1] K. Appel, W. Haken, and J. Koch, J. Math **21**, 439 (1977).

[2] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, quant-ph/0012090.

[3] J. Kempe, quant-ph/0303081.

[4] A. Ambainis, quant-ph/0403120.

[5] V. Kendon, quant-ph/0606016.

[6] B. Tregenna, W. Flanagan, R. Maile, and V. Kendon, New J. Phys. **5**, 83 (2003).

[7] L. K. Grover, Phys. Rev. Lett. **79**, 325 (1997).

[8] E. Farhi and S. Gutmann, Phys. Rev. A **57**, 2403 (1998).

[9] A. M. Childs and J. Goldstone, Phys. Rev. A **70**, 022314 (2004).

[10] N. Shenvi, J. Kempe, and K. Birgitta Whaley, Phys. Rev. A **67**, 052307 (2003).

[11] A. M. Childs, E. Farhi, and S. Gutmann, quant-ph/0103020.

[12] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, quant-ph/0209131.

[13] P. W. Shor, quant-ph/9508027.

[14] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. van den Nest, and H. J. Briegel, quant-ph/0602096.

[15] D. Gottesman, quant-ph/9705052.

[16] D. Gottesman, quant-ph/0004072.

[17] E. Knill, R. Laflamme, and G. J. Milburn, Nature **409**, 46 (2001).

[18] R. Raussendorf and H. J. Briegel, Phys. Rev. Lett. **86**, 5188 (2001).

[19] D. E. Browne and H. J. Briegel, quant-ph/0603226.