Introduction to Quantum Information Science II Lecture Notes

Scott Aaronson

May 2022

We are picking up right where QIS I left off.

Scott Aaronson wishes to thank the following people for invaluable help in preparing these notes: Daniel Liang, Pablo Cingolani, Ethan Tan, Samuel Ziegelbein, Steven Han, and all the students of QIS II at UT Austin in Spring 2022.

Contents

Lecture 1: The Adiabatic Algorithm and Stoquastic Hamiltonians

1.1 The Adiabatic Algorithm

> In his 1989 book *The Emperor's New Mind*, Roger Penrose suggests that maybe quantum mechanical tunneling lets us solve NP-Hard problems. The adiabatic algorithm is an attempt to cash out this intuition.

Last time, we talked about the **Adiabatic Algorithm**, a "quantum-enhanced" version of **Simulated Annealing**, and how we can use it to solve the **Local Hamiltonians Problem** and other optimization problems. We first defined the following two Hamiltonians,

$$H_{init} = \sum_{j=1}^{n} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}_j \otimes I^{\otimes n-1},$$

$$H_{final} = \sum_{l=1}^{m} H_l,$$

where H_{init} has a known ground state $|+\rangle^{\otimes n}$, and the H_l are k-local Hamiltonians for some k. In our problem, we let H_{final} model an instance of 3-SAT. Each H_l is a constraint assigning an energy penalty to every state that does not satisfy the l-th 3-SAT clause. Our goal was to find the ground state of H_{final}, which would be the state that satisfies (or best-satisfies) the 3-SAT instance. But this is an **NP**-hard problem.

So we then defined

$$H_t = (1-t)H_{init} + tH_{final}$$

which linearly interpolates from our initial to final Hamiltonian. By the **Adiabatic Theorem**, if we start in the state $|+\rangle^{\otimes n}$ and vary t slowly enough, we will continuously remain in the ground state of H_t, such that when $t = 1$, our state vector will be exactly the ground state of H_{final} as desired. But what does "slowly enough" mean?

Before we quantify "slowly enough" let us get a better idea about the nature of H_t. One way to do that is to simply plot the trajectory of its eigenvalues over time (Figure 1.1).

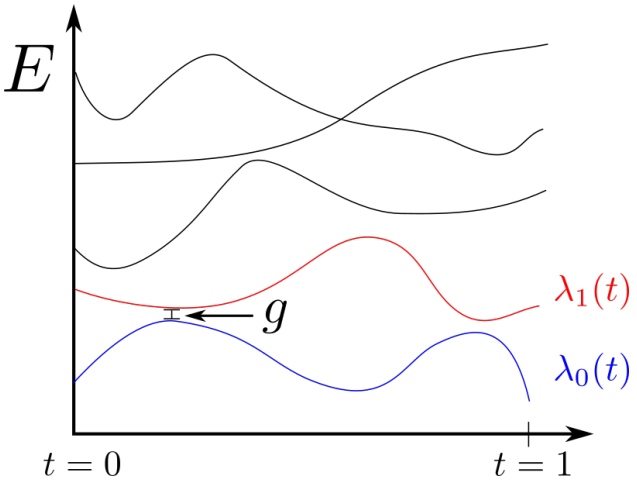


Figure 1.1: Potential plot of energy eigenvalues as a function of time in the adiabatic algorithm.

With the right choice of H_{init} (such as the one we chose above), we can ensure that $\lambda_0(t) < \lambda_1(t)$, or that the energy of the first excited state always exceeds that of the ground state. This is good because we want to prevent such "level crossings" as much as possible.

We can now define the **Minimum Spectral Gap**

$$g = \min_t \left[\lambda_1(t) - \lambda_0(t)\right].$$

By the Adiabatic Theorem, the running time of the algorithm is upper-bounded by $O\left(g^{-2}\right)$. We can already see that this will be efficient only if the gap remains non-exponentially small.

1.1.1 The Grover Landscape

Last semester, we discussed Grover's search algorithm with a black-box oracle, and we can now express it as a Hamiltonian. The Hamiltonian could be a diagonal matrix with zeros on the diagonal everywhere except the cell for which we are searching, where we will put some constant negative value. For example, we could have

$$\begin{pmatrix} 0 & & & \\ & 0 & & \\ & & -1 & \\ & & & 0 \end{pmatrix}.$$

If we plot the energy of each basis state in one dimension, we will get a sort of "golf-course geometry" where the landscape is flat except for a small divot for our desired (unknown) ground state (Figure 1.2).

Figure 1.2: Black-box search style optimization landscape in which only a small region of the landscape is a lower energy than the remaining (flat) landscape.

If we naïvely run the adiabatic algorithm with this Grover fitness landscape, then $g = O\left(\frac{1}{\sqrt{N}}\right)$ and we lose the Grover speedup because our runtime will be something like $O\left(g^{-2}\right) = O(N)$.

Why can't we just run the algorithm with a scaled Hamiltonian of $\sqrt{N}H$, which would have a minimum spectral gap of $\sqrt{N}g = O(1)$, in order to make the runtime $O(1)$? Why can't we go one step further and apply NH, which would have a minimum spectral gap of $Ng = O(\sqrt{N})$, to make the runtime $O\left(\frac{1}{N}\right)$?

- **Engineering answer:** we want the eigenvalues to be constant, representing bounded energy. If we applied $\sqrt{N}H$ or NH then the eigenvalues would blow up and the energy would be unbounded, which is unphysical.
- **Theory answer:** by BBBV, we showed that $O(\sqrt{N}) = O\left(2^{n/2}\right)$ is optimal. If we could beat $\sqrt{N}$ with the adiabatic algorithm, then we could use the Trotterization trick to discretize the adiabatic algorithm and beat Grover, violating BBBV.

So how do we recreate the Grover speedup with the adiabatic algorithm? We can run the algorithm quickly over the regions where the spectral gap is large and run it slowly over the one region where the spectral gap is small.

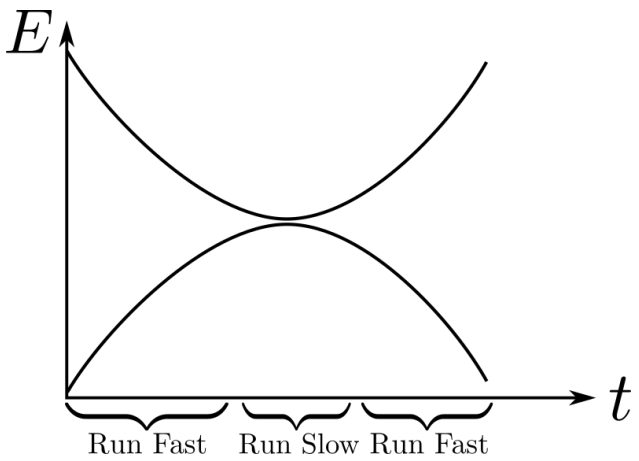


Figure 1.3: High-level sketch of the strategy for getting a Grover speedup using the adiabatic algorithm. The key idea is that we only need to run the algorithm slowly in the vicinity of the minimum spectral gap.

Some more questions ...

- **Do we ever care about $\lambda_1, \lambda_2, \ldots$?** If the implementation of our adiabatic algorithm is not perfectly adiabatic, then we might care, because we might inadvertently jump up to higher energy levels.
- **How do we know when to go fast or slow when varying t?** In theory, we can try to predict where the minimum gap will be. In practice, we would try to run the algorithm many times, varying when we do the slow part each time.

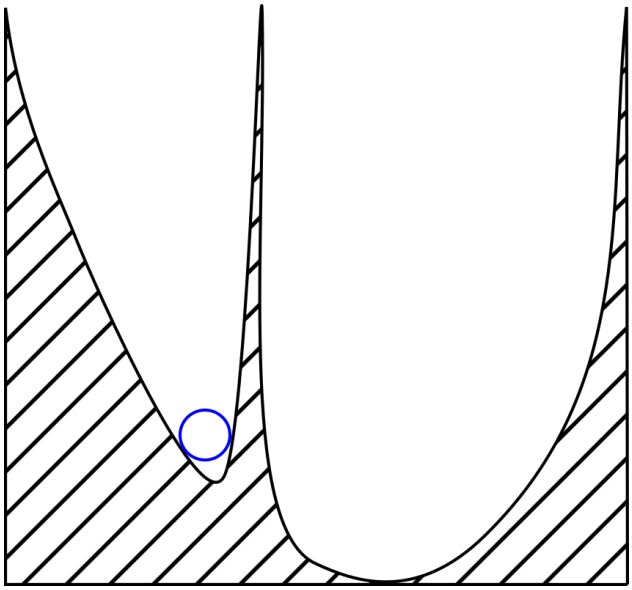


Figure 1.4: Optimization landscape where a sharp thin spike separates the global minimum from the rest of the landscape.

- **Will λ_0 ever cross λ_1?** With the right H_{init}, the adiabatic algorithm guarantees this will never happen. With a different H_{init} there could be a crossing and the adiabatic algorithm could still work, but this is outside the criteria of the Adiabatic Theorem.

- **If we're looking for an approximate answer, can we go faster than $O\left(g^{-2}\right)$?** Yes, but you have to be careful because you might jump to $\lambda_1, \lambda_2, \ldots$.

1.1.2 Some Interesting Fitness Landscapes

Of course, it would be interesting to try to construct some more landscapes to understand how the adiabatic algorithm performs against simulated annealing. Our first example is a landscape with a basin and a spike, where we start in the local minimum part of the basin (Figure 1.4).

We can construct this Hamiltonian by applying an energy penalty depending on the Hamming weight (number of ones) of each input string. Considering input strings of length n, most strings' weights will be $n/2 \pm O(\sqrt{N})$, but there will be a local minimum nearby. Via simulated annealing, we will find our way to the local minimum but will get stuck for exponential time if the spike is too tall. But given a fixed-width spike, quantum tunneling allows the adiabatic algorithm to find its way out of the local minimum, and in the end the algorithm will find its way to the global minimum. So the adiabatic algorithm has a considerable advantage in global-optimum-finding in the case of Hamming weights.

Another interesting landscape is the wide, shallow basin with a thin, deep well hidden off to the side somewhere, where we start near the center of the basin (a local minimum) (Figure 1.5).

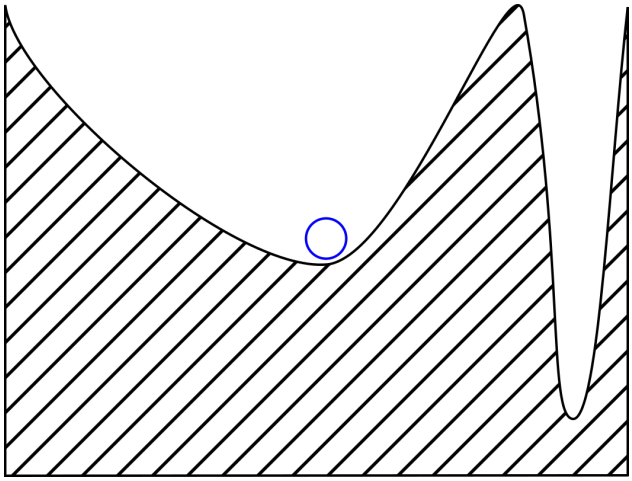


Figure 1.5: An optimization landscape where a very tall thick hill separates the global minimum from the rest of the landscape. The thicker the barrier, the less helpful tunneling is in practice.

In this situation, most of the wavefunction, including most of the tail, will roughly be contained within the shallow basin, so the ability to tunnel will provide little advantage, and both the adiabatic algorithm and simulated annealing will get stuck in the local minimum for exponential time.

A third interesting landscape is another golf-course-like geometry, with two smooth, small, rounded basins of different depths, separated by a flat plateau wide enough to where tunneling provides little advantage (Figure 1.6).

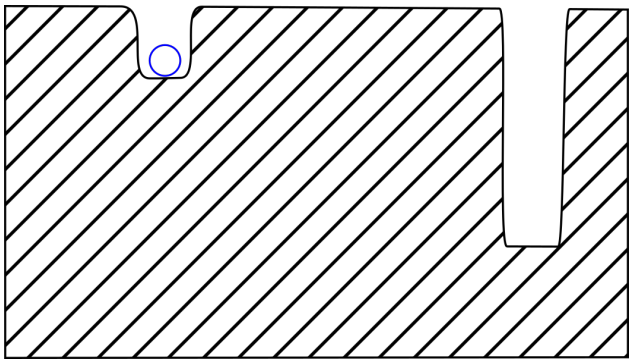


Figure 1.6: The adiabatic algorithm struggles with this sort of optimization landscape, while classical algorithms like simulated annealing have little difficulty.

With simulated annealing, if we start in the shallow basin, there is a random chance that we can jump out of the basin and maybe randomly walk all the way into the second basin. However, the adiabatic algorithm will never make it across the plateau until after an exponentially long time, so in this landscape the classical algorithm wins.

How do these algorithms fare with *random* Hamiltonians? By BBBV, in the case of a random Hamiltonian, neither simulated annealing nor the adiabatic algorithm can beat brute force search using Grover's algorithm.

1.2 D-Wave

We now talk a little about D-Wave, the first quantum computing startup. D-Wave has been designing specialty superconducting qubits that are optimized to stay in their ground states, in order to perform a noisy version of the adiabatic algorithm, called **Quantum Annealing**. (We can think of the adiabatic algorithm as the zero-temperature special case of quantum annealing.) In 2007, D-Wave built a 16-qubit system, and their latest system implements 5000 qubits.

Figure 1.7: D-wave's room-sized 2000-qubit quantum annealing device.

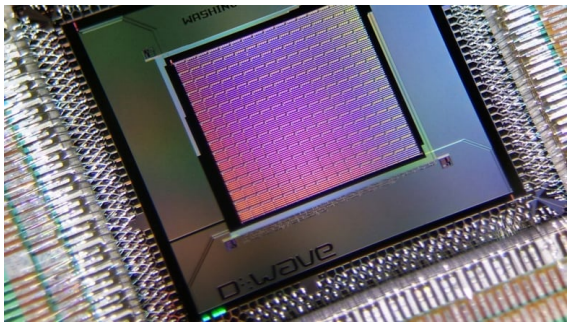


Figure 1.8: Close-up view of one D-Wave's quantum annealing chips.

Let us now consider some practical considerations when running quantum annealing or the adiabatic algorithm. In physical systems, there is always some nonzero temperature. So we use things like dilution refrigerators to go to sub-Kelvin temperatures in order to try to reduce noise. However, this still might not be cool enough to harness the full power of the adiabatic algorithm.

In particular, if the temperature is greater than our minimum spectral gap g, then it is likely that jumps will occur, no matter how slowly we vary our time parameter t. And even if the temperature $T = 0$, there might not be a speedup for specific Hamiltonians. Finally, just because we're getting an exponential speedup over simulated annealing, this doesn't necessarily mean we will get an exponential speedup over all classical algorithms - which brings us to our next topic.

1.3 Stoquastic Hamiltonians

Let us define another class of Hamiltonian matrices, **Stoquastic Hamiltonians**, which are Hamiltonians where all off-diagonal entries are real and nonpositive. An example of a stoquastic Hamiltonian is $\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ and two examples of Hamiltonians that are not stoquastic are $\begin{pmatrix} 1 & i \\ -i & 1 \end{pmatrix}$ and $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$. We will now state a few linear algebraic theorems.

Our first theorem is that if H is stoquastic, then it has a ground state

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

with $\alpha_x \in \mathbb{R}_{\geq 0}$. (We can think of ground states of stoquastic matrices as "quasi-classical," so to speak, due to each amplitude being a nonnegative real.)

We'll deduce this theorem as a consequence of the **Perron-Frobenius Theorem**, which states that if A is a positive matrix (or at least non-negative and well-behaved [irreducible]), then the eigenvector $\vec{v}$ corresponding to the largest eigenvalue $\lambda_{\max}$ has all nonnegative entries. An example would be the stationary distribution of a Markov chain (stochastic matrix). With this in mind, let $A_\beta = I - \beta H$, where the parameter β is sufficiently small to ensure that each entry on the diagonal of βH is less than one. Then, A_β will be a positive matrix, and because of the diagonal entries, it's sufficiently well-behaved that the Perron-Frobenius Theorem applies. Note that A_β is not necessarily stochastic, so the eigenvalue in question is not necessarily 1.

The $\vec{v}$ which maximizes $\langle v|A|v\rangle$ will correspond to the maximal eigenvalue of A and will have all nonnegative entries. Then, $H = \frac{1}{\beta}(I - A_\beta)$ and $H\vec{v} = \frac{1}{\beta}(1-\lambda)\vec{v}$, so $\vec{v}$ is an eigenvector of H corresponding to its *minimum* eigenvalue (ie ground state energy). This proves the theorem about ground states of stoquastic Hamiltonians.

1.3.1 Quantum Monte Carlo

It turns out that stoquastic eigenvectors work well when trying to use Quantum Monte Carlo methods. Quantum Monte Carlo (QMC) refers to a general class of (classical) techniques used to learn information about quantum systems using classical sampling. We can use the QMC method on H to try to approximate the ground state.

One way to think about Monte Carlo methods is that we start with the Gibbs (Boltzmann) distribution $e^{-H\tau}$ for our Hamiltonian, learn everything we can about it for $\tau \in \mathbb{R}$, and then make the substitution $\tau = it$ as we analytically extend the function to imaginary time to obtain our desired matrix exponential.

However, for reasons that we won't go into, this works well only when H is stoquastic, in which case we can define a classical Markov chain that converges to a limiting distribution related to H's ground state.

Lecture 2: Glued Trees and Universal Adiabatic Quantum Computing

2.1 Overview

Last time, we talked about some energy landscapes for which the adiabatic algorithm performs better than classical simulated annealing. Today, we'll discuss landscapes for which the adiabatic algorithm outperforms all possible classical algorithms. We'll start with the so-called **Glued Trees Problem**, discovered by Childs et al. in 2003 [27], which yields an exponential black-box quantum speedup via a non-adiabatic quantum walk. Then, we'll discuss recent breakthroughs by Hastings and Gilyén-Vazirani [40, 31], which turned this into an exponential black-box speedup for the adiabatic algorithm itself. We'll also see how adiabatic computing can be used to simulate universal quantum computation.

2.2 The Glued Trees Problem

Suppose we have a graph G consisting of n vertices, and which includes an "entrance" node s and an "exit" node t. The goal is to construct an algorithm that can efficiently walk from s to t. There exist a plethora of classical algorithms to tackle this problem, such as breadth-first and depth-first searches to systematically explore the entire graph. However, let's also assume that n is exponentially large, so any of these types of classical search methods would take an exponentially long time. Additionally, in this problem, we know nothing about the structure of parts of the graph that haven't been visited yet. We are given a black-box oracle to which we can input the *address* (which is just some label that uniquely identifies a node) of a node that's already been visited and are given the addresses of all the input node's neighbors. The graph can thus only be explored through queries to this oracle, and the search is for the particular label that corresponds to t.

To approach the problem classically, one could imagine another solution that is much less memory-intensive than BFS or DFS: a simple random walk, where a neighbor is picked at random with each step taken. There's a theorem, which we won't prove here, that states that a random walk in an n-vertex undirected graph will visit all vertices within $\mathcal{O}(n^3)$ steps with high probability.

With these problem constraints, the question posed by Childs et al. was if there existed a graph where a random walk, or any other classical query algorithm, would take exponential time to find the exit node, but a quantum algorithm on the same graph could query in superposition to reach the exit faster. The graph they came up with is illustrated in Figure 1. From the entrance and exit

nodes, respectively, the graph branches out into two binary trees consisting of l levels, and thus $\mathcal{O}(2^l)$ vertices. The leaves of the first tree are then connected to those of the second by a random cycle.

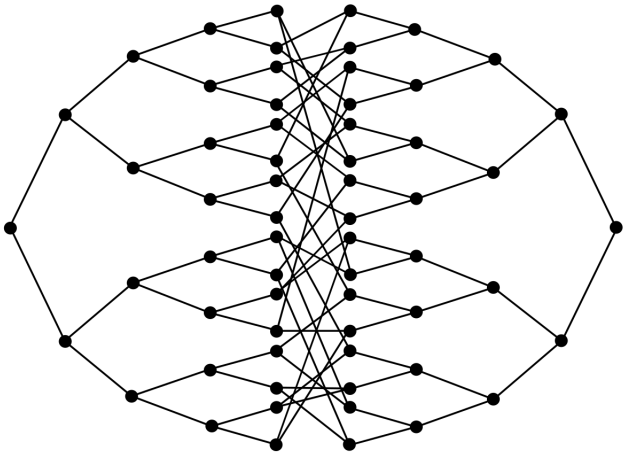


Figure 2.1: The glued trees graph configuration

Using a classical "random walk" approach, starting the traversal at s and travelling down the first tree by randomly choosing a fork of the branch with each step is no problem. However, when you reach the randomly connected middle section, it becomes nearly impossible to guarantee any progress is made towards t. Because there's a $1/2$ probability of making the wrong choice (away from the exit) at each vertex once you cross the center portion of the graph, and since once you do, you inevitably then return to the center portion, you'll remain walking around for an exponential amount of time. It turns out you can prove that any randomized algorithm has a query complexity of $\Omega(2^{\frac{n}{2}})$ to find the exit of a glued trees graph.

As a challenge, you can try to come up with the classical algorithm that achieves this $\mathcal{O}(2^{\frac{n}{2}})$ query complexity. A hint: try starting by doing BFS on the first $\frac{n}{2}$ vertices (the left half of the first binary tree).

But if we take advantage of querying the oracle quantumly to go into a superposition over all the different neighboring vertices, it can be shown that it's possible to reach the exit with high probability in polynomial time, obtaining an exponential speedup over classical methods.

The idea is that, instead of a classical random walk, we'll take a quantum random walk, which is conceptually similar except for the fact that we're evolving into a superposition over nodes. The initial single amplitude of the entrance vertex will then become continually split amongst nodes as we progress down the first binary tree. It gets more complicated once we get to the random cycle in the center; without getting too far into details, it has been shown that this evolution of states can be projected down to one dimension, with a kind of barrier in the middle that corresponds to

the random cycle (Figure 2).

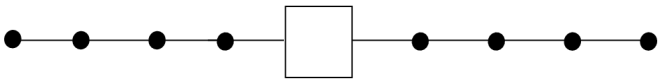


Figure 2.2: Projecting glued trees onto one dimension, with the random cycles represented by a barrier.

This is where adiabatic computation helps us. Once the complicated mess of edges in the center is reduced to a barrier in a one-dimensional walk, all we have to do is quantum tunnel through it! From there, it's straightforward to barrel to the end vertex, as the paths going backwards interfere destructively.

It is important to note that, while this quantum walk is reminiscent of the adiabatic algorithm, it's not strictly speaking adiabatic. The reason for this is that the system doesn't remain in the ground state for the entirety of the algorithm; rather, it can be implemented in such a way that it jumps up to a more excited state, then back down to the ground state at a later point, making it "nearly adiabatic". It's also extremely improbable that this graph configuration would ever be encountered in real life, and a generalization to a whole class of graph properties for which quantum walks are superior is not clear.

Nevertheless, very recently, building on ideas from the glued trees, Hastings [40] gave an example where the adiabatic algorithm gets a provable superpolynomial speedup in query complexity over any classical algorithm, and Gilyén and Vazirani [31] then improved the speedup to an exponential one.

2.3 Universality of Adiabatic Quantum Computing

The idea that adiabatic quantum computing could simulate any quantum circuit may seem counter-intuitive; it isn't immediately obvious why a method used for specific optimization and search problems should be universal. However, a result from 2004 [14] demonstrated that, if we relax the earlier constraint that final Hamiltonians obtained through adiabatic computation must be stoquastic, universal quantum computation through adiabatic computation is possible. Note that despite changing what the H_{final} looks like, we'll still be doing linear interpolation between $H_{initial}$ and H_{final}, and the Hamiltonian whose ground state forms an equal superposition ($|+\rangle^{\otimes n}$) can still be used as $H_{initial}$.

So why aren't stoquastic Hamiltonians sufficient? A result by Bravyi et al. [22] showed that, while using stoquastic Hamiltonians might be able to outperform *some* classical computations, "stoquastic quantum polynomial time" is contained in the complexity class Arthur-Merlin (AM).

> AM *is a classical complexity class of problems that can be decided using a probabilistic polynomial-time verifier (Arthur) that submits random challenges to an all-knowing prover (Merlin) who returns a proof corresponding to the challenges. Arthur's job is then to either accept or reject Merlin's proof.* AM *contains* NP.

It follows from this result that the complexity class corresponding to stoquastic computation– let's call it StoqQP– is contained in the polynomial hierarchy:

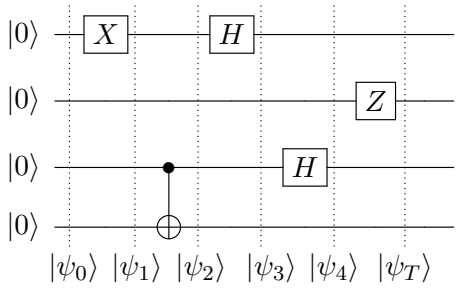
$$\mathsf{StoqQP} \subseteq \mathsf{AM} \subseteq \mathsf{BPP}^{\mathsf{NP}} \subseteq \mathsf{PH}$$

However, it is conjectured that bounded-error quantum polynomial time (BQP), which StoqQP would have to equal to be universal, is not contained in the polynomial hierarchy. This was shown to be true in the black-box setting by Raz and Tal [58]. From this evidence, it seems unlikely that StoqQP = BQP. Once we allow non-stoquastic final Hamiltonians, however, we'll see that this deficiency in stoquastic computation can be eliminated.

For starters, it's straightforward to see how AdiabaticQP $\subseteq$ BQP, where AdiabaticQP represents what we can compute via the adiabatic algorithm with linear interpolatoin, a non-stoquastic final Hamiltonian, and a $\Omega(\frac{1}{\mathsf{poly}(n)})$ spectral gap. Recall that trotterization allows us to decompose a Hamiltonian into many pieces, each of which can be approximated as a product of unitary transformations with a quantum circuit once the pieces are small enough.

What about the other direction, BQP $\subseteq$ AdiabaticQP? To prove this, we'll need to introduce the concept of **Feynman-Kitaev History States**. Say we had a quantum circuit C like the one shown below in which one gate is applied at a time, or per "time step" of the circuit out of T total time. The initial state $|\psi_0\rangle$ is the all-0 state, and subsequent states up to $|\psi_T\rangle$ are achieved after the application of each gate. The *history state* of this circuit encapsulates its entire history $\{|\psi_0\rangle, ..., |\psi_T\rangle\}$, in a single superposition, like so:

$$|\psi_c\rangle = \frac{1}{\sqrt{T}} \sum_{t=1}^{T} |t\rangle |\psi_t\rangle$$



Here, $|\psi_t\rangle$ is the state of the system at time step t, also referred to as the "computation register", while $|t\rangle$, the "clock register", encodes the time step t.

Our goal now is, given some circuit C, to construct a Hamiltonian H_C as a sum of local terms such that the unique ground state of H_C is $|\psi_C\rangle$. Since we're limited to running the adiabatic algorithm,

we can't perform transformations on Hamiltonians as one does in circuit-based quantum computing using unitary gates. Instead, what we want to do is "run" the desired circuit by interpolating from our initial Hamiltonian to H_C.

This construction would be pretty useless if we were unable to measure the final state, $|\psi_T\rangle$, of the simulated circuit. However, if H_C is in an equal superposition over every t states, measuring it would just result in some random time step's state. A simple "hack" to ameliorate this issue is simply to put some relatively large number of identity transformations at the end of the circuit. Now when H_C is measured, $|\psi_T\rangle$ will be measured with high probability.

[12pt] In prior work, Kitaev had already shown how to construct a Hamiltonian whose ground state is a history state. To encode $|t\rangle$, Kitaev used unary notation. Unary notation is a simple way to represent natural numbers, where the quantity of 1s determines the number. So, if $T = 4$, the unary representation of each time step would be

$$|t_1\rangle = |1000\rangle\,,$$
$$|t_2\rangle = |1100\rangle\,,$$
$$|t_3\rangle = |1110\rangle\,,$$
$$|t_4\rangle = |1111\rangle\,.$$

While unary is inefficient, it's useful because the time can be confirmed by measuring only a few bits. For example, given $|1100\rangle$, measuring a $|1\rangle$ on the second qubit and a $|0\rangle$ on the third qubit would indicate that $t = 2$.

Next, we can construct a local Hamiltonian by assigning an energy penalty to local measurement outcomes that indicate that the state is anything other than $|\psi_C\rangle$. To do this, we will perform a local check on each $|t\rangle$ in the following way. First, measure the two qubits adjacent to the t-th qubit. As an example, to perform a check on

$$|t_2\rangle = |1100\rangle\,|\psi_2\rangle\,,$$

we would measure the first and third qubits. Assume that the outcomes of these measurements are $|1\rangle$ and $|0\rangle$ respectively. In that case, this measurement leaves the t-th qubit in the $|+\rangle$ state, and since it is entangled with the computation register, the state collapses to

$$\frac{|t\rangle\,|\psi_t\rangle + |t+1\rangle\,|\psi_{t+1}\rangle}{\sqrt{2}}.$$

We are now in an equal superposition over two consecutive time steps. Observe also that another way to write $|\psi_{t+1}\rangle$ is as $|\psi_t\rangle$ after the t-th gate is applied:

$$|\psi_{t+1}\rangle = U_t |\psi_t\rangle .$$

Now, we can perform a local check by ignoring all other parts of the clock besides the ones that could vary. If U_t^{-1} is applied, controlled on the t-th clock and computation registers being $|1\rangle$, then the state should be brought to $|+\rangle |\psi_t\rangle$. All that has to happen now is a measurement of the t-th clock qubit in the Hadamard basis, which should return $|+\rangle$ if we're in the correct state. This check can be applied for every t, and doing so enforces propagation of the circuit at each time step. Finally, we can add local checks to ensure that the initial state, at t_0, is indeed $|0^n\rangle$.

To perform one check, we had to observe 3 clock qubits and 2 computation qubits– a "5-local test". Rewriting each test as a Hamiltonian that imposes an energy penalty if the measured state is not what's expected will yield H_C.

The last step is to prove that adiabatic evolution can actually reach the ground state of H_C in polynomial time. Indeed, it was proven by Aharonov et al. [17] that the spectral gap remains $\frac{1}{\mathsf{poly}(T)}$, where T is the number of gates. The gap size is inverse polynomial in the size of the circuit, which means the adiabatic algorithm will run in polynomial time, and proves $\mathsf{AdiabaticQP} = \mathsf{BQP}$.

Lecture 3: The Stabilizer Formalism, Part 1

3.1 Introduction

The stabilizer formalism has played a large part in research in quantum computing theory since its development in 1996. However, we only briefly mentioned it in QIS I (in our discussion of universal gate sets).

It turns out that there is a rich subset of quantum circuits that displays many interesting features of QC (entanglement, teleportation, superdense coding, etc.) and is central to quantum error correction, but is not universal and can be simulated using a (non-obvious) efficient algorithm. Understanding this algorithm can give us a conceptually different point of view on quantum computation itself. Stabilizer circuits can be thought of as an "easy subset" of QC (as we'll see, this is literally true when we do fault-tolerant quantum computation with so-called transversal operations). From the stabilizer point of view, the hard part of quantum computing is figuring out how to inject the "non-stabilizerness"—injecting non-stabilizer gates, states and/or measurements that allow us to do full, universal quantum computation.

3.2 Stabilizers and the Pauli group

A unitary U "stabilizes" a pure state $|\psi\rangle$ if $U|\psi\rangle = |\psi\rangle$, i.e. if $|\psi\rangle$ is an eigenvector of U with eigenvalue $+1$. Note that phase **does** matter here, so, if $U|\psi\rangle = -|\psi\rangle$ or $U|\psi\rangle = i|\psi\rangle$, then U **does not** stabilize $|\psi\rangle$. The set of all unitaries that stabilize a state $|\psi\rangle$, denoted $\mathrm{Stab}(|\psi\rangle)$, is known as the *stabilizer group* of $|\psi\rangle$. $\mathrm{Stab}(|\psi\rangle)$ forms a group because, if $U, V \in \mathrm{Stab}(|\psi\rangle)$, then U^{-1} and V^{-1} are also in $\mathrm{Stab}(|\psi\rangle)$, $UV \in \mathrm{Stab}(|\psi\rangle)$, and the identity I is always in $\mathrm{Stab}(|\psi\rangle)$. This group may seem to be very complex and hard-to-characterize, with exponentially large matrices, so keeping track of a state by its stabilizer group may seem like a non-starter at first. However, it turns out that there is a special class of states where we only need to look at the intersection of $\mathrm{Stab}(|\psi\rangle)$ with a group known as the Pauli group. In the special case of one qubit, the Pauli group is the group generated by the Pauli matrices:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

You might remember from QIS I that any 2×2 Hermitian matrix, whether a density matrix or a Hamiltonian, can be written as a real linear combination of these four matrices. This allows us to represent 1-qubit mixed states using the Bloch sphere and parameterize all possible 1-qubit

Hamiltonian, although this doesn't even begin to exhaust the importance of the Pauli matrices. The Pauli matrices have the following relations:

$$X^2 = Y^2 = Z^2 = I$$

$$XY = iZ; \quad YZ = iX; \quad ZX = iY$$

$$YX = -iZ; \quad ZY = -iX; \quad XZ = -iY$$

Note that the Pauli matrices have the same algebraic relationships as the four units of the quaternions $(1, i, j, k)$. Within the group, we have:

Pauli matrix	I	X	Z	Y
States(s) stabilized	all states	$\|+\rangle$	$\|0\rangle$	$\|i\rangle$

Pauli matrix	$-I$	$-X$	$-Z$	$-Y$
States(s) stabilized	no state	$\|-\rangle$	$\|1\rangle$	$\|-i\rangle$

These six states are, in some sense, "picked out" (i.e. stabilized) as being special by the Pauli group. Note that they are also the six axis directions of the Bloch sphere.

3.3 Generalizing the Pauli group

We can generalize the 1-qubit Pauli group to P_n, the Pauli group on n qubits. Elements of P_n have the following form:

$$b(P_1 \otimes P_2 ... \otimes P_n)$$

where P_1 through P_n are elements of the single-qubit Pauli group P_1 and $b \in \{1, -1, i, -i\}$. Note that, in the literature on stabilizer states, $b(P_1 \otimes P_2 ... \otimes P_n)$ is often written as just $bP_1P_2...P_n$. Be sure to keep in mind that, in this context, $bP_1P_2...P_n$ represents a tensor product of Pauli matrices, **not** a matrix multiplication of unitaries.

Multiplying elements in P_n is fairly simple. Since elements of P_n are tensor products, we can multiply elements component-wise, keeping in mind the global phase as well. For example, with $ZXXIY, XYZXX \in P_5$, we have

$$(ZXXIY)(XYZXX) = (ZX)(XY)(XZ)(IX)(YX) = (iY)(iZ)(-iY)X(-iZ) = YZYXZ$$

Elements of P_n can also be represented using "tableaus". These are matrices consisting of two $n \times n$ blocks of 1s and 0s. In a tableau, each Pauli matrix is represented by two-bit strings:

Pauli matrix	I	X	Z	Y
String	00	10	01	11

Each row in the matrix represents a generator of P_n. One of the bits of the two bit string is placed in the left matrix, and the other in the right matrix. This effectively creates the following scheme: the presence of a 1 in the ith entry of a row indicates the presence of a X in the ith component of the element of P_n, and a 0 indicates the absence of an X. Similarly, for the right block, the presence of a 1 in the ith entry of the row indicates the presence of a Z in the ith component of the element of P_n, and a 0 indicates the absence of an Z. If there is a 1 in the ith entries of both the

left and right blocks, this indicates a Y in the ith component of the element of P_n. For example, the generating set $\{ZXXIY, XYZXX\}$ would be represented as

$$\left[\begin{array}{c} ZXXIY \\ XYZXX \end{array}\right] = \left[\begin{array}{ccccc|ccccc} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{array}\right]$$

and their product is represented as

$$\left[\begin{array}{c} YZYXZ \end{array}\right] = \left[\begin{array}{ccccc|ccccc} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right]$$

Notice that multiplying two elements together is done by simply bitwise XOR-ing the $2n$-bit strings.

3.4 Stabilizer states

Note that $\text{Stab}(|\psi\rangle \cap P_n)$, being an intersection of two groups, is itself a group, which we call the Pauli stabilizer group of $|\psi\rangle$. Now let S_n, the n-qubit stabilizer states, be the set of n-qubit states such that $|\text{Stab}(|\psi\rangle) \cap P_n| = 2^n$, i.e., the set of states where the intersection of their stabilizer group and P_n contains 2^n elements. For a randomly chosen state $|\psi\rangle$, $\text{Stab}(|\psi\rangle) \cap P_n$ will most likely be trivial, so states in S_n are fairly special. In some sense, the states in S_n are the ones where the Pauli group P_n gives us as much information as it possibly could.

3.4.1 The Gottesman-Knill Theorem and its implications

Theorem 1 (Gottesman-Knill [33])**.** *The elements of S_n are exactly those states that are reachable from the $|0\rangle^{\otimes n}$ state using only the CNOT, Hadamard and Phase gates. Furthermore, every state in S_n is **uniquely** determined by the group $P_n \cap Stab(|\psi\rangle)$.*

The fact that the set $\{CNOT, H, P\}$ preserves membership in S_n is why it isn't universal for quantum computing. As a result of the Gottesman-Knill Theorem, if we know the Pauli stabilizer group of a state, we know the state itself. How useful is this representation for algorithms? Since an n-qubit stabilizer state has a Pauli stabilizer group of size 2^n, with elements that are $(2n+1)$-bit strings, it may initially seem that keeping track of a Pauli stabilizer group is no easier than keeping track of a vector of amplitudes. However, there's an easier way to keep track of a group than to just keep track of all of its elements, namely by keeping track of its generators. For an n-qubit state, we only need n generators to keep track of its Pauli stabilizer group, since n linearly independent elements will generate a group of size 2^n. For example, consider a generating set with two different strings, 110 and 011. Bitwise XORing these together gives us 101, and the trivial string 000 will also be included in the group generated by 110 and 011. Since elements of a Pauli stabilizer group are multiplied by bitwise XOR in the tableau representation in a similar way to the example just given (ignoring global phases), this property also holds for elements of a Pauli stabilizer group. So, for the Pauli stabilizer group of an n-qubit stabilizer state, we need to keep track of n $2n+1$-bit elements ($2n+1$ because the global phases of an element of a Pauli stabilizer group will only ever be 1 or -1, never i or $-i$). Therefore, we can keep track of a stabilizer state using $n(2n+1)$ bits, which, crucially, is polynomial in the number of qubits n. However, we don't yet know how to update the generating set of a stabilizer state as operations are performed on it, so we don't yet have an efficient algorithm. We'll see that shortly.

3.4.2 Pauli stabilizer groups are always abelian

Now, there is no reason that $\text{Stab}(|\psi\rangle)$ has to be abelian (i.e. all elements commute with each other), and we have already seen that the Pauli group P_n is nonabelian. However, their intersection, $\text{Stab}(|\psi\rangle) \cap P_n$, will always be an abelian group. To see this, note that, while P_n is nonabelian, all of its elements either commute or anticommute. Consider $XX, ZZ \in P_2$:

$$(XX)(ZZ) = (-iY)(-iY) = -YY$$

$$(ZZ)(XX) = (iY)(iY) = -YY$$

so XX and ZZ commute. But, for $Z, X \in P_1$ (in fact, for any odd number of Xs and Zs)

$$XZ = -iY$$

$$ZX = iY$$

$$XZ = -ZX$$

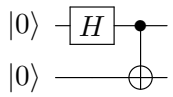
So, for some $\text{Stab}(|\psi\rangle) \cap P_n$ to be nonabelian, there must be two elements in the group that anticommute. But if these two elements, call them P_1 and P_2, are both in $\text{Stab}(|\psi\rangle) \cap P_n$ for some $|\psi\rangle$, then P_1P_2 and $P_2P_1 = -P_1P_2$ must both stabilize $|\psi\rangle$ and therefore

$$|\psi\rangle = P_1P_2 |\psi\rangle = P_2P_1 |\psi\rangle = - |\psi\rangle$$

which can only be the case if $|\psi\rangle$ is the zero vector, which is not a valid quantum state. Therefore, $\text{Stab}(|\psi\rangle) \cap P_n$ cannot have anticommuting elements and therefore must be abelian.

3.5 Using the stabilizer formalism

Consider the stabilizer representation of the following circuit:



We begin in the $|00\rangle$ state, which has $P_2 \cap \text{Stab}(|\psi\rangle) = \{II, ZI, IZ, ZZ\}$, which is generated by the set $\langle ZI, IZ\rangle$ (although any two of the three non-identity elements would also work). The H gate takes us to the state $|+\rangle \otimes |0\rangle$, which has $P_2 \cap \text{Stab}(|\psi\rangle) = \{II, XI, IZ, XZ\}$ and generating set $\langle XI, IZ\rangle$. Notice that the effect of applying H to the first qubit in the stabilizer representation is to exchange the Z in the first components of the $P_2 \cap \text{Stab}(|\psi\rangle)$ group's elements with an X. The function of the Hadamard gate is to swap between the X and Z bases, which is reflected in its action on stabilizer representations.

Applying the CNOT gate gives us the Bell state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ with $P_2 \cap \text{Stab}(|\psi\rangle) = \{II, XX, ZZ, -YY\}$ and generating set $\langle XX, ZZ\rangle$. Notice that the CNOT gate has changed XI in the generating set of $|+\rangle \otimes |0\rangle$ into XX. This may seem fairly intuitive since the CNOT gate in this circuit has the second qubit as its target. But also, something else has happened. Specifically, IZ in the generating set of $|+\rangle \otimes |0\rangle$ has become ZZ. It seems that the CNOT gate is acting in the "wrong direction", affecting the stabilizer representation of the control qubit! This is a reflection of the fact that a

CNOT from control qubit a to target qubit b is the same as a CNOT from control qubit b to target qubit a when viewed in the Hadamard basis. In other words, this is a reflection of the hidden symmetry in the CNOT gate which can only be seen in different bases.

In tableaus, the same evolution looks as follows: we start in $|00\rangle$, which has tableau representation

$$\left[\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

To apply an H gate to the jth qubit, we swap x_j (the jth column of the X matrix on the left) and z_j (the jth column of the Z matrix on the right). So, for this circuit, we'll swap the first columns of the X and Z matrices to get:

$$|+\rangle \otimes |0\rangle = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

To apply a CNOT with qubit j as the control and qubit k as the target, we first set x_k equal to $x_j \oplus x_k$. We then, in the Z matrix, set z_j equal to $z_j \oplus z_k$ (note that this is the opposite direction). For example:

$$|EPR\rangle = \left[\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right]$$

To apply a Phase gate P to the jth qubit, the correct rule turns out to be to set $z_j := z_j \oplus x_j$. Note that we haven't handled the global phases yet; these have more complicated, nonlinear rules.

Lecture 4: Tableau Representation, Measurement, and QEC

4.1 Overview

In the last lecture we discussed stabilizer quantum computing. We went over what it means for a unitary matrix U to stabilize a quantum state $|\psi\rangle$. We also went over the definition of $\text{Stab}(|\psi\rangle)$, the *stabilizer group* of a state $|\psi\rangle$.

Remember that an n-qubit Pauli group P will have size

$$|P| = 4^{n+1}$$

because of n qubits having 4 different possible stabilizers, and an extra $(n+1)^{\text{th}}$ being the phase $b \in \{1, -1, i, -i\}$. The number of bits needed to represent an element of this group will just be the log of the size, so $2n + 2$ bits.

We then went on to introduce the Gottesman-Knill Theorem (1996), which implies the existence of a poly-time classical algorithm to simulate stabilizer circuits [33].

Finally, we briefly discussed the tableau representation of stabilizer states with their generators.

In this lecture we will further elaborate on the tableau representation of the stabilizer states. In particular, we will discuss how the phase is incorporated into the tableau representation, as well as how to handle measurement.

We will also discuss Aaronson and Gottesman's paper on simulating stabilizer circuits without having to do Gaussian elimination [10].

The lecture will end with a brief discussion of Quantum Error Correction in the stabilizer formalism. In particular, we will discuss the Shor Code and the 5-qubit code in that language.

4.1.1 Warmup

As a warmup from the paper, Aaronson and Gottesman calculate the number of stabilizer states for n qubits.

This can be started by considering the number possibilites for the first generator, which is $4^n - 1$,

since the first one can be any product of the four Pauli matrices, excluding the all I generator. Then the next generator will have $\frac{4^n}{2} - 2$ possibilites, to account for the fact that it must commute with the first generator and that it cannot be I or M_1.
This pattern can be extended to the following product:

$$(4^n - 1)\left(\frac{4^n}{2} - 2\right)\left(\frac{4^n}{4} - 4\right)\cdots\left(\frac{4^n}{2^{n-1}} - 2^{n-1}\right)$$

Now we have to divide the whole product by the number of equivalent generating sets for a stabilizer S, since this product by itself would overcount. There are $2^n - 1$ choices for the first generator, $2^n - 2$ for the second, and so on, assuming each generator needs to be independent of the previous ones,

$$(2^n - 1)(2^n - 2)\cdots(2^n - 2^{n-1})$$

Dividing the two we get

$$2^n \prod_{i=0}^{n-1}(2^{n-i} + 1)$$

stabilizer states in total.

For example, when $n = 1$ we have 6 and when $n = 2$ we have 60 two-qubit stabilizer states, namely $6 \times 6 = 36$ unentangled stabilizer states ($|0\rangle, |1\rangle, |+\rangle, |-\rangle, |i\rangle, |-i\rangle$) and 24 entangled EPR pairs.

4.2 Stabilizer Simulation

There is a fast classical algorithm to simulate any stabilizer circuit. The algorithm works by keeping track of the generators of the state's Pauli stabilizer group. Note that monitoring a list of amplitudes would *not* be efficient, since there are exponentially many of them. Maintaining a list of all elements of the stabilizer group, rather than just the generators, would likewise be exponentially inefficient.

4.3 Tableau Representation

4.3.1 Keeping Track of signs

It turns out that to keep track of the signs, all we have to do is add an extra column of bits on the left side, with 0 representing + and 1 representing − for each row.

$$\left[\begin{array}{c|cc|cc} r_1 & 1 & 1 & 0 & 1 \\ r_2 & 0 & 0 & 1 & 1 \end{array}\right]$$

We have the following rules for updating these bits:

- Hadamard and Phase on i^{th} qubit:
  For every row j, flip r_j if the two entries in the columns being flipped are equal to 1.

In other words, if x_{ji} is the entry on the j^{th} row and i^{th} column of the left matrix, and z_{ji} is the entry on the j^{th} row and i^{th} column of the right matrix, we have

$$r_j = r_j \oplus x_{ji} z_{ji}$$

So we must have that $x_{ji} = z_{ji} = 1$ to perform the flip.

- CNOT from qubit i to k:
  Flip r_j iff $x_{ji} = z_{jk} = 1$ and $x_{jk} = z_{ji}$

An important side note here is that row operations *do not* change the state, but merely the set of generators. In other words, we are just exchanging the Pauli generators we have for others in the stabilizer group. Consider the following example where we perform a row operation of adding the first row to the second row.

$$\left[\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right] \rightarrow \left[\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{array}\right]$$

The generators go from $\{XX, ZZ\}$ to $\{XX, YY\}$.

4.3.2 Measurement

Now let's talk about how one would go about making measurements on qubits in this tableau representation. How will we know what the outcome is? Without loss of generality, assume we want to measure the first qubit of a state $|\psi\rangle$, which is $\in \{|0\rangle, |1\rangle\}$. It's not hard to see that if the stabilizer $+ZIII... \in P_n \cap \text{Stab}(|\psi\rangle)$, then the measurement outcome will be $|0\rangle$, seeing as that operator stabilizes $|0\rangle$ and does not stabilize $|1\rangle$. By the same logic, if the stabilizer $-ZIII... \in P_n \cap \text{Stab}(|\psi\rangle)$, then the measurement outcome will be $|1\rangle$. Now we claim that if neither of these are $\in P_n \cap \text{Stab}(|\psi\rangle)$, then the outcome will be $|0\rangle$ or $|1\rangle$ with equal probability. **Why is this?** The reason being that the stabilizer is always an equal superposition over some subspace S:

$$\frac{1}{\sqrt{|S|}} \sum_{x \in S} i^{f(x)} |x\rangle$$

where f is a function that determines the phase of $|x\rangle$. The proof is deferred to when we discuss inner products between stabilizer states. **So what happens when the outcome is random?** Well, in the case of our simulation we can just flip a coin and pick $+ZI$ or $-ZI$ for that qubit, updating the tableau accordingly. *However,* a key caveat is that we must maintain that the stabilizer group be abelian. For this to be maintained, we *must throw out* anything that does not commute with our new generator. As an example, let's say we had a 2 qubit generating set $\{XX, ZZ\}$ and we want to measure to $|0\rangle$, so we put ZI into this set. We need to get rid of XX because it does not commute with ZI. In fact, it anti-commutes. In the actual tabeleau, we can check for this using a **symplectic inner product**, defined below
Given a tableau representation like the one below,

$$\left[\begin{array}{ccc|ccc} a_1 & \cdots & a_n & a_{n+1} & \cdots & a_{2n} \\ b_1 & \cdots & b_n & b_{n+1} & \cdots & b_{2n} \end{array}\right]$$

we have the following formula to see if two rows commute or anticommute,

$$a_1 b_{n+1} \oplus \cdots \oplus a_n b_{2n} \oplus a_{n+1} b_1 \cdots a_{2n} b_n = \begin{cases} 0 \to & \text{commute} \\ 1 \to & \text{anticommute} \end{cases}$$

Putting it all together: So we now have analysis that tells us whether to expect a $|0\rangle$ or $|1\rangle$, but how can we deduce this from a tableau representation? One answer is **Gaussian Elimination**: we can solve a system of linear equations to see if our row space contains $(0000|1000) = ZIII$. If so, we then multiply the appropriate generators to determine its sign.

4.4 Alternative to Gaussian Elimination

We discussed earlier how Gaussian elimination can be used to simulate measurements, but it has a complexity of $O(n^3)$. It turns out that there is a faster way to simulate stabilizer circuits, including their measurements, which was shown by Aaronson and Gottesman [10].

4.4.1 "Destabilizer Generators"

At a high level, the idea of the A-G simulation is to store n "destabilizer" generators in addition to the n stabilizer generators. These will be Pauli operators as well. This requires a cost of twice as many bits to specify a stabilizer state. But simulating a measurement can be done in only $O(n^2)$ time, rather than $O(n^3)$ from Gaussian elimination. The destabilizer generators, together with the stabilizer generators, generate the whole Pauli group P_n.

4.5 Stabilizer Codes and Quantum Error Correction

Stabilizer codes are a form of quantum error correcting code in which the encoding and decoding can be done by stabilizer circuits. Quantum error correction was a topic we discussed in QIS 1, but now we can discuss it in the language of stabilizers.

As an example, recall Shor's 9-qubit code from QIS 1, whose codewords are

$$\left(\frac{|000\rangle \pm |111\rangle}{\sqrt{2}}\right)^{\oplus 3}.$$

This code turns out to be a stabilizer code, in that we can specify the logical $|\bar{0}\rangle$ and $|\bar{1}\rangle$ states using the stabilizer formalism, like so:

$$ZZIIIIIII,$$
$$IZZIIIIII,$$
$$IIIZZIIII,$$
$$IIIIZZIII,$$
$$IIIIIIZZI,$$

$$IIIIIIIZZ,$$
$$XXXXXXIII,$$
$$IIIXXXXXX,$$
$$ZZZZZZZZZ$$

The first 8 generators determine the 2-dimensional subspace of logical codewords. The 9^{th} generator is $ZZZZZZZZZ$ or $-ZZZZZZZZZ$ depending on whether we want $|0_L\rangle$ or $|1_L\rangle$ respectively. If we wanted $|+_L\rangle$ or $|-_L\rangle$ we would use $XXXXXXXXX$ or $-XXXXXXXXX$.

As another example, consider the 5-qubit code, which is the smallest code that can detect and correct an error on any one qubit. We can likewise write it using the stabilizer formalism:

$$XZZXI,$$
$$IXZZX,$$
$$XIXZZ,$$
$$ZXIXZ,$$
$$\pm ZZZZZ$$

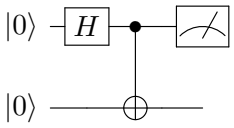
The sign of the last generator will once again yield either logical $|0\rangle$ ($+ZZZZZ$) or logical $|1\rangle$ ($-ZZZZZ$).

Lecture 5: More Fun with Stabilizers

5.1 Overview

We've been talking about the Gottesman-Knill Theorem [33], which shows that stabilizer circuits (i.e., quantum circuits consisting of H, CNOT, Phase gates, and computational basis measurements) can be simulated efficiently on a classical computer.

Consider the following quantum circuit.



We can represent what's happening at each point in the circuit by maintaining a list of generators of the Pauli stabilizer group:

$$\{ZI, IZ\} \overset{H}{\mapsto} \{XI, IZ\} \overset{\text{CNOT}}{\mapsto} \{XX, ZZ\}.$$

Equivalently, we can write the stabilizer tableau at each point in the circuit:

$$\left(\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right) \overset{H}{\mapsto} \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right) \overset{\text{CNOT}}{\mapsto} \left(\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right).$$

To simulate measurement of the first qubit, we have to choose $\pm ZI$ uniformly at random, and insert the choice into the Pauli stabilizer group. Then, we remove any elements that do not commute with our choice. In the example above, XX does not commute with $\pm ZI$ and would be removed from the list of generators. Therefore, the post-measurement state can be represented as

$$\{\pm ZI, ZZ\} \qquad \text{or, alternatively,} \qquad \begin{array}{c} \pm \\ + \end{array} \left(\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right)$$

If we randomly select $+ZI$, then the post-measurement state is $|00\rangle$, the same state we started in. One way to see this is that the initial tableau and final tableau differ only by row operations. Similarly, if we randomly select $-ZI$, one can check that the post-measurement state is $|10\rangle$.

Say we measure the first qubit of an arbitrary stabilizer state $|\psi\rangle$, then the measurement outcome can be determined as follows.

1. If $+ZII\ldots I \in \mathcal{P}_n \cap \text{Stab}(|\psi\rangle)$, then the measurement outcome is $|0\rangle$ with certainty.

2. If $-ZII\ldots I \in \mathcal{P}_n \cap \text{Stab}(|\psi\rangle)$, then the measurement outcome is $|1\rangle$ with certainty.

3. Otherwise, the measurement outcome is $|0\rangle$ or $|1\rangle$ with equal probability.

In general, given the tableaus of two stabilizer states $|\psi\rangle$ and $|\phi\rangle$, one can show that (1) if the Pauli stabilizer groups of $|\psi\rangle$ and $|\phi\rangle$ contain the same Pauli operator with opposite sign, then the two states must be orthogonal (since this implies that there is a measurement that perfectly distinguishes the two states); (2) otherwise, $|\langle\psi|\phi\rangle| = (1/\sqrt{2})^s$, where s is the number of generators required to generate the group $\mathcal{P}_n \cap \text{Stab}(|\psi\rangle) \cap \text{Stab}(|\phi\rangle)$.

To see this, first note that if the states are non-orthogonal, then $|\langle\psi|\phi\rangle|$ must depend only on s because s is the only thing that is invariant between the two stabilizer states when we apply arbitrary unitary transformations to them.

We give an example to illustrate where the $1/\sqrt{2}$ term comes from. Recall from above that measuring the first qubit of an EPR pair has the following effect on the stabilizer tableau:

$$\left(\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right) \overset{\text{measure first qubit}}{\mapsto} \begin{array}{c} \pm \\ + \end{array} \left(\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right)$$

The first qubit will be projected onto $|0\rangle$ or $|1\rangle$. Call the pre-measurement state $|\psi\rangle$, and let $|\psi_0\rangle$ be the post-measurement state if the measurement outcome is $|0\rangle$ and $|\psi_1\rangle$ if the measurement outcome is $|1\rangle$. Observe that for either post-measurement state, $s = 1$ (the phase on the stabilizer tableau does not affect s). Therefore, we conclude that $|\langle\psi|\psi_0\rangle| = |\langle\psi|\psi_1\rangle|$, and, because $|\psi_0\rangle$ and $|\psi_1\rangle$ are orthogonal, it must be that $|\langle\psi|\psi_0\rangle| = |\langle\psi|\psi_1\rangle| = 1/\sqrt{2}$. See [11] for more detail on computing the inner product between stabilizer states.

In this lecture, we continue discussing the stabilizer formalism. First, we will see that simulating a stabilizer circuit is a complete problem for the complexity class $\oplus\mathsf{L}$ (pronounced "parity L"). Then, we explore applications of the stabilizer formalism in quantum error correcting codes and fault-tolerant quantum computation.

5.2 The Computational Power of Stabilizer Circuits

We know from the Gottesman-Knill Theorem that simulating stabilizer circuits can be done in classical polynomial time. So one can ask: are stabilizer circuits just equivalent to classical computation? Aaronson and Gottesman studied this question and found that the answer is no — stabilizer computation is weaker than universal classical computation [11].

Let Simulate-Stabilizer-Circuit be the following computational problem: you are given as input a description of a stabilizer circuit and you have to decide whether a measurement outcome on some qubit is $|1\rangle$ with certainty after the stabilizer circuit is applied to the initial state $|0\rangle^{\otimes n}$. Aaronson and Gottesman showed that Simulate-Stabilizer-Circuit is complete for a complexity class called $\oplus\mathsf{L}$ (pronounced "parity L"), the class of problems that reduce to simulating a polynomial-size circuit comprised only of NOT and CNOT gates acting on the initial state $|0\rangle^{\otimes n}$.

From this definition of $\oplus\mathsf{L}$, it is clear that $\oplus\mathsf{L} \subseteq \mathsf{P}$. $\oplus\mathsf{L}$ can equivalently be defined as the class of problems that can be solved by a nondeterministic logarithmic-space Turing machine, that accepts

if and only if the number of accepting paths is odd. From this definition, it is clear that $\mathsf{L} \subseteq \oplus\mathsf{L}$, where L is the class of problems that can be solved by a deterministic logarithmic-space Turing machine.

It is conjectured that $\oplus\mathsf{L} \neq \mathsf{P}$. Intuitively, this is because a circuit comprised only of NOT and CNOT gates is rather weak. Such a circuit is essentially just a chain of XORs, i.e., a straight-line program (no branches or loops) whose only operation is XOR. Note also that it is impossible to create superposition and there's no access to even an AND gate. Despite this, Aaronson and Gottesman showed that simulating a stabilizer circuit is computationally equivalent to the seemingly easier problem of simulating a circuit of NOT and CNOT gates only.

Before we explain how they showed this, we first must explain what we mean by "computationally equivalent". When we say that different NP problems are equivalent to each other, we mean that they are reducible to each other in polynomial time. In other words, if you have an oracle for one, then in polynomial time you can solve the others. But that cannot be what we mean when we talk about $\oplus\mathsf{L}$-completeness because $\oplus\mathsf{L} \subseteq \mathsf{P}$! Reductions in P could solve $\oplus\mathsf{L}$ problems outright and make them all trivially equivalent. When we talk about equivalence, we mean equivalence under reductions that are weaker than the class in consideration. So, for $\oplus\mathsf{L}$-completeness, a good choice is reductions in the complexity class L, or the class of problems that can be solved by a Turing machine with logarithmic space, which is the notion used in [11]. Specifically, they showed that a machine with only logarithmic read/write memory, read-only input of size n, write-only output of size n, and an oracle for evaluating CNOT circuits can predict the outcome of any stabilizer circuit (i.e., Simulate-Stabilizer-Circuit $\in \oplus\mathsf{L}$).[1] The other direction of the equivalence is very easy to see. Clearly, simulating a CNOT circuit is reducible to simulating a stabilizer circuit because CNOT circuits are a special case of stabilizer circuits (i.e., Simulate-Stabilizer-Circuit is $\oplus\mathsf{L}$-hard).

How do we show Simulate-Stabilizer-Circuit $\in \oplus\mathsf{L}$? First, note that almost every step of the Gottesman-Knill algorithm can be expressed as a chain of XORs (or CNOTs) acting on the stabilizer tableau, all of which can be simulated by a circuit comprised only of CNOT gates. The one complication is keeping track of the phases, which, if you recall from the previous lecture, are the only nonlinear terms that appear in the Gottesman-Knill algorithm. The key observation is that the phases never feed back into the rest of the computation. Stabilizer operations can cause the phases to change, but a phase update never causes any other part of the tableau to change. The only time phases become relevant is when we measure at the end of the circuit. So, the idea is to just keep track of the contributions to the phases throughout the computation, and let the logarithmic-space reduction handle the nonlinear terms.

5.3 Stabilizer Codes

Despite the severe limitations of stabilizer computation, it has become a central part of quantum computing in large part because of the huge role it plays in quantum error correcting codes (QECCs). Almost all of the QECCs that are studied are *stabilizer codes* and are presented using the stabilizer formalism. Below are three common examples of stabilizer codes presented with the stabilizer formalism.

[1] This also uses a result of Hertrampf, Reith, and Vollmer [41]: $\mathsf{L}^{\oplus\mathsf{L}} = \oplus\mathsf{L}$.

Shor Code:

$ZZI\ III\ III$
$IZZ\ III\ III$
$III\ ZZI\ III$
$III\ IZZ\ III$
$III\ III\ ZZI$
$III\ III\ IZZ$
$XXX\ XXX\ III$
$III\ XXX\ XXX$
$\pm ZZZ\ ZZZ\ ZZZ$

5-Qubit Code:

$XZZXI$
$IXZZX$
$XIXZZ$
$ZXIXZ$
$\pm ZZZZZ$

Steane Code:

$IIIXXXX$
$IXXIIXX$
$XIXIXIX$
$IIIZZZZ$
$IZZIIZZ$
$ZIZIZIZ$
$\pm ZZZZZZZ$

Each of these codes can detect and correct a single error on any one of the physical qubits and encodes a single logical qubit. The final $\pm ZZ\ldots Z$ determines whether the encoded state is $|0_L\rangle$ or $|1_L\rangle$ (a logical $|0\rangle$ or logical $|1\rangle$).

There is a special type of stabilizer code called a Calderbank-Steane-Shor (CSS) code. A CSS code is a stabilizer code where you can choose generators such that each generator is either all I's and Z's or all I's and X's. Looking at the stabilizer codes above, we can see that the Shor code and Steane code are CSS codes. It turns out that the 5-qubit code is not a CSS code, but one would need to check all generating sets to confirm this. CSS codes are nice because they arise from combining two different classical error correcting codes that are dual to one another.

5.4 Fault-Tolerant Quantum Computation

5.4.1 Transversal Gates

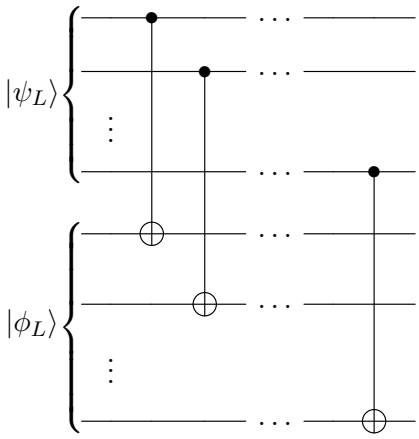
We are interested in quantum error correction because quantum error correction is the best known path to known quantum computation. But, it is not enough to encode your qubits with a QECC and have them sit around! We also want to apply gates to the encoded qubits and have the result be the encoded outcome.

Say we are using a code where a single logical qubit is encoded with n physical qubits,

$$|\psi_L\rangle = \alpha\,|0_L\rangle + \beta\,|1_L\rangle\,,$$

and we want to apply a Hadamard gate to the logical qubit. It's not immediately clear how this should be done. Can we just apply a Hadamard gate to each physical qubit? If that's true for a given code, we say that the Hadamard gate is *transversal* (or, *acts transversely*) for that code. If we have two logical qubits $|\psi_L\rangle$ and $|\phi_L\rangle$ and we can apply CNOT $|\psi_L\rangle\,|\phi_L\rangle$ by applying CNOT to

each pair of physical qubits like so,



then the CNOT gate is transversal for that code.

Transversal gates are the simplest we could hope for. If a gate is not transversal, then applying it will be a more complicated (and, thus, expensive) operation. Therefore, a useful code is a code where we have as many transversal gates as possible. Several interesting observations were made in this direction, which we state without proof.

- The CNOT gate is transversal for all CSS codes.
- The Hadamard gate is transversal for all CSS codes where there the X part and Z part are dual.[2]
- The Phase gate is transversal for doubly-even CSS codes, where doubly-even means that the number of X's and the number of Z's in a generator is a multiple of 4.

It turns out that the Steane code satisfies all of these properties. Therefore, all stabilizer operations are transversal in the Steane code. Or, more broadly, there exists a QECC where all stabilizer operations are transversal. While this is an awesome result, it falls short of what we wanted because of the Gottesman-Knill Theorem! Stabilizer circuits are exactly a class of circuits that we can simulate in classical polynomial time. It would be *really* awesome if there were a QECC where the set of transversal gates was universal. If such a code existed, this is plainly what we would want to use for fault-tolerant quantum computation. No such code was found, and, in 2009, Eastin and Knill proved a no-go theorem for a QECC with this property.

Theorem 2 (Eastin-Knill Theorem [28])**.** *For any quantum error-correcting code that can detect 1-qubit errors, the set of unitary transformations generated by the transversal gates is a discrete set and, therefore, not universal for quantum computation.*

Note that the Eastin-Knill Theorem does not rule out the possibility that transversal circuits for a code are hard to simulate classically. It rules out the possibility that you can have a continuous set of transversal gates.

[2]The duality is via the Hadamard. The X part of the code is taken to the Z part and vice versa by Hadamarding each Pauli in each generator. The Steane code is an example of this.

The formal proof involves Lie algebras, which we will not go into here. Instead, we will try to give some intuition about why a continuous set of transversal gates is incompatible with error detection. Consider the single-qubit gate

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

that rotates a qubit by θ radians. Now, suppose we have a code where a single logical qubit is encoded with n physical qubits,

$$|\psi_L\rangle = \alpha\,|0_L\rangle + \beta\,|1_L\rangle\,,$$

and the code has a universal transversal set of gates. Then, R_θ is transversal, and we apply it to the logical qubit by applying R_θ to each individual qubit:

$$(R_\theta)_L = R_\theta^{\otimes n}.$$

The key is to think about what happens when θ is on the order of $1/n$. Each physical qubit is being rotated by $1/n$ radians, which is impossible to distinguish from an error occurring on approximately one of the physical qubits in expectation. But, the whole point of the QECC is to detect and correct an error on a physical qubit, which would nullify the effect of the R_θ gate! The proof of the Eastin-Knill Theorem is essentially formalizing this fundamental tension between applying R_θ for small θ and detecting single-qubit errors.

5.4.2 Stabilizer Circuits and T Gates

To sum up, we can only hope for fault-tolerant quantum computation where the stabilizer gates will be transversal. In order to achieve universal quantum computation, we must get some "non-stabilizerness" into the encoded qubits. Figuring out how to inject non-stabilizerness has been a major focus of quantum computer design for the past 20 years.

Although we won't show it here, it is known that stabilizer gates and any one non-stabilizer gate is universal for quantum computation. A common choice for the non-stabilizer gate is the T gate:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}.$$

Our current situation for fault-tolerant quantum computation is as follows. We express quantum circuits using stabilizer and T gates. We have QECCs where applying stabilizer gates is essentially free (because they act transversely). However, since the T gate cannot also be transversal by the Eastin-Knill Theorem, each time we apply the T gate will be expensive. The question is: given a quantum circuit, how many T gates will we need? That is, how much non-stabilizerness do we need to do something interesting? Aaronson and Gottesman studied this question in 2004. They generalized the Gottesman-Knill algorithm to give a simulation of quantum circuits with only stabilizer gates plus a few non-stabilizer gates [11]. The running time of the algorithm is polynomial in the number of qubits and stabilizer gates and exponential in the number of non-stabilizer gates.

Theorem 3 ([11])**.** *A quantum circuit with n qubits, m stabilizer gates, and k single-qubit non-stabilizer gates can be classically simulated in* $\mathrm{poly}(n, m, 16^k)$ *time.*

For our discussion, the relevant conclusion to draw from this theorem is: If you want to do something that is exponentially hard for a classical computer, then you should expect that the minimum number of non-stabilizer gates is linear in the number of qubits.

We conclude by giving a sketch of the proof of Theorem 3. The idea is to run the Gottesman-Knill algorithm, but modify it as necessary to deal with non-stabilizer gates. Suppose we are simulating a circuit of stabilizer gates and, say, T gates. We start in the initial state $|0\rangle^{\otimes n}$ and we run the Gottesman-Knill algorithm to simulate the stabilizer gates, maintaining an $O(n^2)$-bit tableau. When we encounter the first T gate (without loss of generality, on the first qubit), suppose our quantum state is some stabilizer state $|\psi\rangle$ with the following form:

$$|\psi\rangle = \frac{|0\rangle|\psi_0\rangle + |1\rangle|\psi_1\rangle}{\sqrt{2}}.$$

(The case where the first qubit is definitely $|0\rangle$ or definitely $|1\rangle$) is simpler.) After applying the T gate, the resulting quantum state will live within a 4-dimensional subspace spanned by $|0\rangle|\psi_0\rangle$, $|0\rangle|\psi_1\rangle$, $|1\rangle|\psi_0\rangle$, and $|1\rangle|\psi_1\rangle$. Therefore, the resulting quantum state can be expressed as follows.

$$(T \otimes I^{\otimes n-1})|\psi\rangle = a|0\rangle|\psi_0\rangle + b|0\rangle|\psi_1\rangle + c|1\rangle|\psi_0\rangle + d|1\rangle|\psi_1\rangle.$$

Observe that all four of these basis states are stabilizer states and can be represented with stabilizer tableaus. So, now we need to keep track of a superposition of four stabilizer states, each one with a complex amplitude. By linearity we can track all four of them separately.

The next time we encounter a non-stabilizer gate, we repeat. So, we will have 16 tableaus and 16 complex amplitudes to keep track of. After k non-stabilizer gates, we end up in a state of the form

$$\sum_{j=1}^{4^k} a_j |\psi_j\rangle,$$

where each $|\psi_j\rangle$ is a stabilizer state. The remaining difficulty is calculating the probability of a measurement outcome. In this case, it's convenient to think of a measurement as corresponding to a Hermitian, positive semi-definite matrix E, where

$$\mathbf{Pr}[E \text{ accepts } |\psi\rangle] = \langle\psi| E |\psi\rangle.$$

In the simulation, when we measure the state after k non-stabilizer gates, we get

$$\mathbf{Pr}[E \text{ accepts } |\psi\rangle] = \sum_{j,k}^{4^k} a_j^* a_k \langle\psi_j| E |\psi_k\rangle.$$

We can compute each $\langle\psi_j| E |\psi_k\rangle$ using the Gottesman-Knill algorithm, assuming E is a stabilizer measurement. Then, we compute the linear combination of all 16^k terms that appear in the sum.

Next time, we will talk about magic initial states, a proposal from Bravyi and Kitaev for injecting non-stabilizerness into fault-tolerant quantum computations. Then, we'll talk about major improvements to this simulation algorithm.

Lecture 6: From Stabilizer QC to Universal QC using magic states

6.1 Overview

Last time we continued our discussion of stabilizer states. We will finish this discussion today by presenting so-called *magic states*. They introduce a tiny bit of non-stabilizerness into stabilizer circuits. We'll show how this lets us break out of stabilizer prison and obtain universal quantum computing. We'll also talk about the recent notion of *stabilizer rank* and how it led to improved classical simulations of quantum circuits with stabilizer gates and T gates.

6.2 Who lives inside stabilizer prison?

As discussed, there are two equivalent characterizations of the stabilizer states: they are the n-qubit states $|\psi\rangle$ whose Pauli stabilizer group is maximal, meaning $|\mathcal{P}_n \cap \mathrm{Stab}(|\psi\rangle)| = 2^n$ and, equivalently, they are also the states that can be produced from the all-zero state $|0^n\rangle$ using only CNOT, H and P gates.

In some sense, the Gottesman-Knill algorithm gave us one direction of this equivalence: it gave us that applying CNOT, H and P gates is never going to change that the Pauli stabilizer group has n generators. That is special about these gates: applying them is just going to change what those generators are, but never the number of them. If you start in the stabilizer set, they will always keep you there.

For the other direction, we need to show that any state from the stabilizer set can be prepared using CNOT, H and P gates only. That is "just" a circuit construction problem: you have to take any valid stabilizer tableau and give a way to create it starting from the tableau of the all-zero state. In the paper of Aaronson and Gottesman in 2004 [12], they give one way for doing this, where they first do a round of Hadamards, then a round of CNOTs, then a round of phase gates, then another round of Hadamards, ... they had 11 layers. Then you can produce any stabilizer state that you want, which thereby proves the universality of these gates – of course not universality for general quantum computations, but for stabilizer operations.

6.3 How many gates do we need in stabilizer prison?

How many stabilizer gates are needed to create an arbitrary stabilizer state? We can easily see that this number has to be at most $2^{\mathcal{O}(n^2)}$: There are only $2^{\mathcal{O}(n^2)}$ different n-qubit stabilizer states, therefore the circuit size has to be at most $2^{\mathcal{O}(n^2)}$, otherwise you would enter an infinite cycle.[1] That's an upper bound.

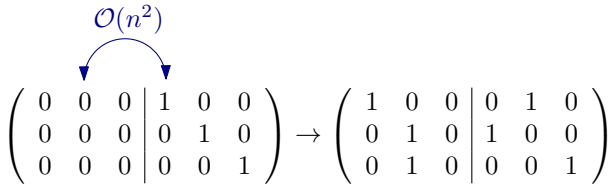


Figure 6.1: We can obtain the tableau representation of any stabilizer state from the tableau representation of the all-zero state by performing $\mathcal{O}(n^2)$ column operations. This reasoning provides an upper bound on the number of gates necessary to create an arbitrary stabilizer state.

To get a better upper bound, we recognize that each of the operations corresponds to some column operation on the stabilizer tableaus (see also Fig. 6.1). Just from linear algebra, how many column operations do you typically need to get from a given $n \times n$-matrix to another given (target) $n \times n$-matrix? By doing Gaussian elimination (there it would usually be row operations of course, but it is the same concept) you typically do it in $\mathcal{O}(n^2)$ operations. We are just talking about a generalization of Gaussian elimination here. So it stands to reason that, whatever you can do at all with stabilizer gates on n qubits you can do with only $\mathcal{O}(n^2)$ of them! And indeed one can show this using the Aaronson-Gottesman decomposition: each H and P round uses $\mathcal{O}(n)$ gates, while each CNOT round uses $\mathcal{O}(n^2)$ gates. This is very different from universal quantum computing. From QIS I we know that you would need an exponential number of gates ($\Theta(4^n)$) to apply an arbitrary n-qubit unitary transformation.

Aaronson and Gottesman even improved this result for stabilizer circuits in their 2004 paper [12]: Actually $\mathcal{O}(\frac{n^2}{\log n})$ gates suffice. And this is also the necessary number, hence this result is optimal. The proof of the lower bound is counting: in order to specify a n-qubit stabilizer circuit with k stabilizer gates (exemplarily shown in Fig. 6.2), how many bits of information do you need? To answer this, we first need to know how many bits we need for each gate. We need to specify whether it is CNOT, H or P, but furthermore also which qubit (or pair of qubits) it is acting on. For a 1-qubit gate, that means $\log_2(n)+\mathcal{O}(1)$ bits, for a 2-qubit gate (here CNOT) it is $2\cdot\log_2(n)+\mathcal{O}(1)$. Hence $\mathcal{O}(\log_2 n)$ bits per gate are necessary. How many different stabilizer circuits are therefore possible with k gates? From the number of bits used to describe one circuit we can immediately conclude $2^{\mathcal{O}(k\cdot\log n)}$. Since the number of possible stabilizer states still is 2^{n^2}, we can only hope we have covered all of them if $k\cdot\log n$ is of order n^2, the number of bits needed to describe one state (giving a circuit is one way to specify the state).

> This kind of information argument should just become second nature. It is just counting. Whenever you encounter a new situation in computer science, start by counting: how many programs are there of a given length?

[1] We had more distinguishable circuits than distinguishable states we used to create them, therefore we for sure have circuits with more gates than necessary.

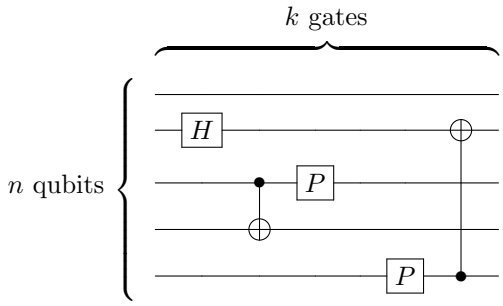


Figure 6.2: Stabilizer circuit with n qubits and k stabilizer gates.

In conclusion, an arbitrary stabilizer circuit must consist of $\Theta(n^2/\log n)$ gates, otherwise there would not be enough circuits to go around.

6.4 Transversality of CSS codes and stabilizerness

Another thing we talked about last time are CSS codes. This is the subclass of quantum error-correcting codes where we can partition the generators of our Pauli stabilizer group into two sets: one set only involving I and X, and the other set only involving I and Z. The examples were therefore the Shor code and the Steane code.

CSS codes are particularly useful because we can often find transversal quantum gates for them. For example, the Hadamard gate acts transversally on a code word as $H_L |\psi_L\rangle = H |\psi_1\rangle \otimes H |\psi_2\rangle \otimes \cdots \otimes H |\psi_n\rangle$. It is apparent that transversal gates are particularly easy to apply to encoded qubits.

We know that there are CSS codes for which all stabilizer gates act transversally. But unfortunately, there is also the Eastin–Knill theorem [29] for which we sketched a proof in the last lecture. It says that there is no quantum error-correcting code that is actually able to correct errors and has a universal set of transversal gates. Indeed, the set of transversal gates for a given code is always discrete, so it cannot include all possible one qubit gates.

That is the picture of quantum computing we've had for the past 20 years: **stabilizer operations are easy, non-stabilizer operations are hard.**[2]

6.5 How much non-stabilizerness do we need?

So practical ideas for designing fault-tolerant quantum computers for the past 20 years have focused on this issue: how to inject non-stabilizerness into the computation?

The first question you could ask is: how many non-stabilizer gates do you need to do something interesting? For n qubits, m stabilizer gates, and k non-stabilizer gates, then there is the classical simulation of [12] which uses time $\text{poly}(m, n) \cdot \exp(k)$, as we discussed last time.

[2] It is important to emphasize that easy and hard do **not** mean polynomial vs. exponential here, they should all be polynomial/efficient in principle. We are talking about practical issues here.

This implies that if you want an exponential speedup from your quantum computation[3] then, at a minimum, you would need $\mathcal{O}(n)$ non-stabilizer gates.

> The best currently known classical running time for factoring algorithms is roughly $\mathcal{O}(n^{1/3})$ (Number Field Sieve). It is not ruled out that – without improving the classical factoring algorithm – maybe a quantum computer could factor an n-digit number using only $\mathcal{O}(n^{1/3})$ non-stabilizer gates. It is not known how to do that currently. Presumably, it would need to be something beyond Shor's algorithm – something that would have to know about elliptic curve theory. If not – how would the classical simulation work, without knowing about elliptic curve theory? Finding a quantum factoring algorithm using any number of non-stabilizer gates less than $\mathcal{O}(n)$ is therefore an interesting open problem.

We conclude: we need a significant number of non-stabilizer gates to do something exciting with a quantum computer.

6.6 How magic states inject non-stabilizerness

Now, how do we inject non-stabilizerness into fault tolerant quantum computing?

Of course, there would be a brute-force approach: decode the state, apply the non-stabilizer gate, encode again. That works in principle, but people who run the numbers realized that it would be horrifically expensive – and of course, it is error-prone.

But there are very clever and less expensive ways to get non-stabilizerness into otherwise stabilizer-based quantum computations: in 2004, Bravyi and Kitaev introduced ***magic states*** [24]. This concept is motivated by the goal to keep our gates transversal, to make the engineers' lives bearable. So what else can we do to get up to universal computation? To answer this, let's review the assumptions we made for the Gottesman-Knill Theorem. We assumed that our initial state (the all-zero state) was a stabilizer state and we assumed that our measurements were stabilizer measurements (for example in the standard basis). What happens if we relax those assumptions? What Bravyi and Kitaev noticed is that, when we take a non-stabilizer circuit and feed it some non-stabilizer initial states, that can be good enough to simulate the effect of non-stabilizer operations. The states that do this are what we call *magic states*.

> Even more magic in quantum mechanics?

We will now explore how this works.

[3] Meaning, as usual, you want to do something in polynomial time quantumly that took exponential time classically.

6.6.1 An example

Assume we want to apply some non-stabilizer gate. For example

$$P_\theta = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

will be non-stabilizer for most values of θ. Since the stabilizer gates are a "maximal pre-universal set," just one of these non-stabilizer gates is already enough to boost us up to universality. E.g. $P_{\pi/4} = T$ is a popular choice.

How could we do that? As part of our input, in addition to the $|0\rangle$ states, we could also manufacture one specific non-stabilizer state. Let us call it

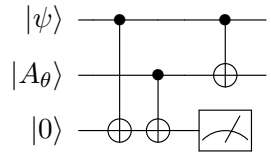
$$|A_\theta\rangle = \frac{|0\rangle + e^{i\theta}|1\rangle}{\sqrt{2}}.$$

So just an equal superposition of $|0\rangle$ and $|1\rangle$, but with an additional $e^{i\theta}$ phase. We claim: by using only (some number of copies of) this state and stabilizer operations, we can simulate the effect of a non-stabilizer gate. But, we need one further trick: *adaptive measurements*. What do we mean by that? The ability to change the gates we apply based on the outcome of measurements taken.

> In QIS I we spent some time proving that adaptive measurements are actually useless. They did not give us any more power, because they could always be simulated by just using CNOTs. But the proof for the uselessness of adaptive measurements already assumed that we had a universal set of gates! What we will see now is that, if we do not already have a universal set, then adaptive measurements (together with magic states) can boost us up to universality.

Without adaptive measurements, we would still be stuck in stabilizer land. To further motivate why this provides us with more computational power, we can just emphasize that CNOT is a stabilizer gate, unlike controlled CNOT, which is the very powerful Toffoli gate! So providing control based on the outcome of a measurement is kind of like boosting a CNOT to a Toffoli... And finally breaks us free from the prison of stabilizerness.

How does the Bravyi-Kitaev construction work? It is beautifully simple. Let us say we want to apply the P_θ gate to some state $|\psi\rangle = a|0\rangle + b|1\rangle$ to obtain $P_\theta|\psi\rangle = a|0\rangle + be^{i\theta}|1\rangle$. We have an ancilla qubit initialized to $|0\rangle$. All we need then are three CNOT gates and one measurement:



Measuring the ancilla qubit is basically measuring the parity of the two qubits – it does not tell us more. In more detail, but ignoring the ancilla qubit, before the measurement we have:

$$|\psi\rangle|A_\theta\rangle = (a|0\rangle + b|1\rangle)\left(\frac{|0\rangle + e^{i\theta}|1\rangle}{\sqrt{2}}\right) \tag{6.1}$$

$$= \frac{a|00\rangle + ae^{i\theta}|01\rangle + b|10\rangle + be^{i\theta}|11\rangle}{\sqrt{2}}. \tag{6.2}$$

Measuring the parity of this state yields two options (depending on the parity) with equal probabilities:

$$a\,|00\rangle + be^{i\theta}\,|11\rangle \quad \text{or} \quad ae^{i\theta}\,|01\rangle + b\,|10\rangle\,. \tag{6.3}$$

To reduce this to one-qubit states we apply the last CNOT gate

$$a\,|00\rangle + be^{i\theta}\,|10\rangle \quad \text{or} \quad ae^{i\theta}\,|01\rangle + b\,|11\rangle\,. \tag{6.4}$$

and ignore the last qubit, which is now for sure $|0\rangle$ or $|1\rangle$:

$$a\,|0\rangle + be^{i\theta}\,|1\rangle \quad \text{or} \quad ae^{i\theta}\,|0\rangle + b\,|1\rangle \equiv a\,|0\rangle + be^{-i\theta}\,|1\rangle\,. \tag{6.5}$$

Since we measured the parity, we also know which of the two states we have. One way to think about this result is that with probability $\frac{1}{2}$ we have achieved our goal – and with probability $\frac{1}{2}$ we have done the opposite, applied $P_{-\theta}$. So not quite perfect yet.

In the paper of Bravyi and Kitaev, they suggest just repeating this with the same $|\psi\rangle$, using more $|A_\theta\rangle$ magic states (each time a fresh one). When we do this, half the time we add θ to our phase, half the time we subtract it. We keep repeating this. This random walk will, before too long, end up where we want it to be, at $a\,|0\rangle + be^{i\theta}\,|1\rangle$ – and furthermore, once we end up there, we will know, because each time we take a step we know whether it was a positive or a negative step from the measurement outcome. We keep repeating this until we did exactly one positive step in total. Then we have applied P_θ.

What we have achieved is, that all the non-stabilizerness can be concentrated into the creation of the magic initial states – as long as we have adaptive measurements. So the problem now is to create the $|A_\theta\rangle$ states, or – if this is on encoded qubits – encoded $|A_\theta\rangle$. But that's just one fixed task. If you look at the design of quantum computers in the last 20 years, most of the effort in the quantum computation actually goes into what is known as *magic state factories*.

> This approach always seemed a little bit crazy to Prof. Aaronson. Maybe there is a better way – but so far applying non-stabilizer gates directly is even worse!

Even if we don't have magic state factories, there is a different way to proceed. We can get the boost from stabilizer to universality using non-stabilizer *measurements*, meaning measurements in a basis of non-stabilizer states. How to prove it?

> In computer science or math, the first way to prove anything is to try to reduce it to something you already know...

You can use non-stabilizer measurements in order to prepare non-stabilizer states. To create our $|A_\theta\rangle$ state for example, we could just take a $|0\rangle$ qubit and measure it in a basis that includes $|A_\theta\rangle$ as one of the basis vectors, the other one just being the orthogonal state $|B_\theta\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - e^{i\theta}\,|1\rangle\right)$. Then, half the time we will obtain $|A_\theta\rangle$ from the measurement – and we will know when from the measurement result.[4]

We conclude that, if we take a stabilizer circuit and adaptive measurements, then the addition of the tiniest grain of non-stabilizerness will boost us up to quantum universality – whether the non-stabilizerness is in the initial states or measurements.

[4] For encoded qubits we would need to do the encoded version of the measurement.

6.6.2 Intermediate strength without adaptive measurements?

What happens if we have non-stabilizer magic initial states, let us say n copies of $|A_\theta\rangle$, and stabilizer gates only – but no adaptive measurements? What is this able to do? It seems to be something that is not universal quantum computing – but still beyond what a classical computer can do. Something of intermediate strength.

We can see why it is very unlikely to be universal quantum computing: We are still limited to a discrete set of unitary transformations. The only difference is that this discrete set is now acting on non-stabilizer initial states, but it is still not like general quantum circuits. On the other hand, the fact that we are operating on non-stabilizer initial states is already enough to make the Gottesman-Knill algorithm no longer work.

Later, when we talk about bosons and fermions, we'll talk about quantum supremacy experiments such as BosonSampling and we will see how there are various models of quantum computation that seem to be non-universal (they can't do all of BQP), but that can sample certain probability distributions that a classical computer does not seem to be able to sample easily. The BosonSampling model seems one example of that – and stabilizer circuits with magic initial states and no adaptive measurements seem seems to be another such example where you can sample classically hard distributions.

6.7 Stabilizer rank and the battle between classical and quantum computing

Recently, in 2016, Bravyi and Gosset gave a radical new approach to taking a quantum circuit and simulating it classically [23]. This was specifically for quantum circuits that consisted of mostly stabilizer gates with only a limited number of $T = P_{\pi/4}$ gates. Their theorem says:

Theorem 4. *If we have quantum circuit with n qubits, m stabilizer gates and k T-gates then there is a classical algorithm to simulate it, meaning calculate the probabilities of the measurement outcomes, that uses time* $\text{poly}(n,m) \cdot 2^{k/2}$. *If by simulate we merely mean that we can sample from the probability distribution over measurement outcomes, then we only need* $\text{poly}(n,m) \cdot 2^{0.23k}$ *time.*

Structurally, these running times look like those from Aaronson and Gottesman's 2004 paper [12]. But the exponent here is way better: it is still exponential in the number of T-gates, but the base is better. They even get below 2^k. These are state-of-the-art algorithms now, even in practice, to simulate circuits with a limited number of T-gates. The running time can be improved even further: in 2021 Qassim, Pashayan and Gosset presented an algorithm to approximate probabilities of the outputs in $\text{poly}(n,m) \cdot 2^{0.3963k}$ time [55].

How did they do that? At a high level, by using the concept of magic states and the fact that T gates can be replaced by stabilizer gates and magic initial states, e.g. again $|A_\theta\rangle$.

They introduced a new notion that has become very important. It is called the *stabilizer rank* and is defined by

$$\text{sr}(|\psi\rangle) = \text{ min \# of stabilizer states } |v_1\rangle, \ldots, |v_k\rangle \text{ s.t. } |\psi\rangle = \sum_{i=1}^{k} \alpha_i |v_i\rangle \tag{6.6}$$

for any given n-qubit state $|\psi\rangle$. In other words, it's the minimum number of stabilizer states that need to be superposed to get $|\psi\rangle$.

What if $|\psi\rangle$ itself is a stabilizer state? Of course $\mathrm{sr}(|\psi\rangle)$ is then just 1.

For any state $|\psi\rangle$, clearly $\mathrm{sr}(|\psi\rangle) \leq 2^n$. Why? We could just use the classical basis states, which are stabilizer states, and we know that any state is a superposition of those. But since we can use any stabilizer state here, maybe we can represent an arbitrary state $|\psi\rangle$ in a more compact way. Of course, the stabilizer states are not just a basis, they are more than a basis. They are overcomplete.

If we now look at a tensor product of n magic states, what Bravyi and Gosset did was to prove an upper bound on its stabilizer rank. The best current bound, from [55], is something like

$$\mathrm{sr}(|A_\theta\rangle^{\otimes n}) \leq 2^{0.3963n}.$$

The typical way to show this is just by using computer search. So we program a computer to represent, say, $|A_\theta\rangle^{\otimes 5}$ by linear combination of fewer than $2^5 = 32$ stabilizer states. As soon as you got that, you can just take many products of this construction... That will give you a general upper bound on the stabilizer rank of n copies of $|A_\theta\rangle$.

What is the ultimate limit to this sort of approach? If you could show that this $|A_\theta\rangle^{\otimes n}$ had a stabilizer rank that was only polynomial in n, then this would imply that BQP = P and hence all quantum computations could be classically simulated efficiently – at least as long as you can actually find this stabilizer-representation efficiently. Therefore this emerged as a very fundamental question: presumably that is not true, but **can we rule it out unconditionally**?

So far, there is a hard result from very recently, by Peleg, Shpilka and Volk [54] and independently Lovitz and Steffan [48], giving the best lower bound so far. They show that the stabilizer rank of the Magic State $|A_{\pi/4}\rangle^{\otimes n}$ is at least linear in n: $\mathrm{sr}(|A_{\pi/4}\rangle^{\otimes n}) \in \Omega(n)$. This result does not help much with settling the discussion – but they also give some explanation why this bound is so low by showing how this is related to circuit lower bound problems in classical complexity theory. In classical complexity, it happens again and again that we believe that the best circuits for a problem are of exponential size, but the best lower bounds we can find are pathetically weaker than that... there are some explanations for why we are currently stuck with these problems, including something called the *Natural Proofs Barrier* of Alexander Razborov and Steven Rudich [59].

We see that a new front has opened in the battle of classical versus quantum computation! What is the best lower bound on stabilizer rank? Maybe we will be able to answer this better.

6.8 Next time

Next time we will start talking about bosons and fermions. The topics will include Boson/FermionSampling, the Knill-Laflamme-Milburn Theorem, and getting from a non-universal model to a universal one using adaptive measurements!

Lecture 7: Introduction to Bosons and Fermions

7.1 Overview

In the last lecture, we finished learning about stabilizers. In this lecture we introduce the idea of modes and the two types of particles in our universe: *bosons* and *fermions*. Bosons are force carrier particles, examples of which include photons, gluons, gravitons, and the Higgs boson. Fermions are particles that make up matter. Examples include quarks, electrons, and neutrinos.

7.2 Modes

Up until this point in the course, we have thought about a quantum system as being built out of qubits. But not all quantum systems are most naturally described this way. As another possibility, of great importance for nature, we could have multiple identical particles distributed in a discrete set of bins. We refer to these bins as *modes*.

Figure 7.1: a collection of four particles placed into $m = 8$ modes.

To be sure, if we have n identical particles each placed in one of m possible nodes, we can still represent the state of our system using qubits. One way to do so is simply to list the mode that each particle is in. This would look like:

$$|x_1, x_2, \ldots, x_n\rangle \text{ where each } x_i \in \{1, \ldots, m\} \tag{7.1}$$

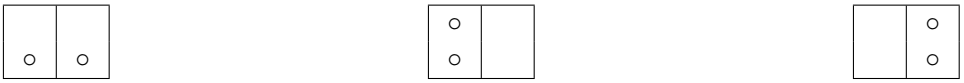
This representation requires $n \log_2 m$ qubits.

More concretely, consider the case where there are two particles and two modes ($n = m = 2$). If our particles have identifiers attached to them, then there would be four possible configurations:

In equilibrium each of these outcomes should be equally likely, so we can expect to see the particles in the same mode 50% of the time.

7.3 Bosons

But the way bosons work is different. They do not have identifiers attached to them. So in particular, there are only three possible states of 2 bosons in 2 modes:



Once again, when our system is in equilibrium we can expect each of these outcomes to be equally likely. But this time there is a $\frac{2}{3}$ chance that both of our particles end up in the same mode! Already we see a central property of bosons: they tend to "bunch up" in the same mode more than classical particles.

Now suppose we have n particles and $m = 2$ modes. In the classical setting in equilibrium, we expect the number of particles in the first mode to have a binomial distribution with mean $\frac{n}{2}$ and standard deviation of order $\sqrt{n}$. However if our particles are bosons, every possible number of particles in the first mode is equally likely, giving us a uniform distribution.

Now suppose we have m modes. How many particles do we need before it is likely, in equilibrium, that some two particles land in the same mode? In the classical setting, the Birthday Paradox tells us that $\Theta(\sqrt{m})$ particles are necessary and sufficient. With bosons it is also $\Theta(\sqrt{m})$, but now the probability of a collision is enhanced by a factor of 2, meaning that we only need $\frac{1}{\sqrt{2}}$ times the number of particles. When we make our particles indistinguishable, all states representing permutations of the particle labels become the same. This means that the number of configurations where n particles are in n distinct modes decreases by a factor of $n!$. On the other hand, the number of configurations containing n particles in $n-1$ modes only decreases by a factor of $n!/2$, as each pair of permutations that differ only in a transposition of the two particles that end up in the same mode give the same state.

7.3.1 How to represent bosons with qubits?

So far we have talked about bosons as particles that are indistinguishable from one another, but what exactly does this mean? If we take the representation of a system given in (7.1), we need the state to remain the same if we permute the qubits. So for example if our state is

$$\sum_{i,j=1}^{m} \alpha_{i,j} \left|i,j\right\rangle \tag{7.2}$$

where i and j are the locations of our two particles, then we require that $\alpha_{ij} = \alpha_{ji}$ holds for all pairs i, j.

When we add this restriction to our state the representation we use in (7.1) becomes inefficient. It overcounts the number of possible states because the encoding treats $|i,j\rangle$ and $|j,i\rangle$ as if they were different. To fix this, we can encode our state of n bosons in m modes in a different way:

$$\left|s_1,\ldots,s_m\right\rangle \text{ where } s_i \geq 0 \text{ and } s_1 + \ldots + s_m = n \tag{7.3}$$

In this representation, the s_i are known as the *occupation numbers*; s_i represents the number of particles in the i'th mode. By a simple calculation, the number of basis states of that form is:

$$\binom{m+n-1}{n}$$

which grows exponentially with $\min(m, n)$.

7.3.2 How to represent qubits with bosons?

Conversely, we can represent a single qubit using a single boson with $m = 2$ modes. The particle appearing in the first mode corresponds to our qubit being $|0\rangle$ and the particle appearing in the second mode corresponds to our qubit being $|1\rangle$. To represent a basis state $|x_1, \ldots, x_n\rangle$, we can use n bosons and $m = 2n$ modes. If $x_i = 0$ then we place a boson in mode $2i - 1$ and if $x_i = 1$ we place a boson in mode $2i$. This is known as the *dual rail* representation, as the two modes per qubit resemble parallel railway tracks:

7.3.3 Entanglement in bosons

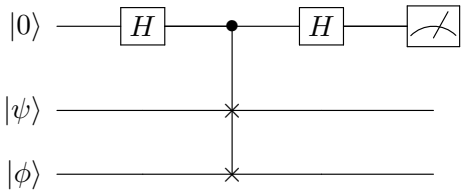
We learned that entanglement is a concept that fundamentally requires two or more qubits. But it turns out that we can violate the Bell inequality using just a single boson! Consider the state:

$$|\psi\rangle = \frac{|1, 0\rangle + |0, 1\rangle}{\sqrt{2}}$$

This state features a single boson that is in one of two possible modes. If Alice and Bob can each measure one of the modes of $|\psi\rangle$, then they can treat this state like a Bell pair for the CHSH game.

7.3.4 How do we know bosons are identical?

The swap test is a common way to check whether two quantum states are identical. Here is the circuit for the swap test:

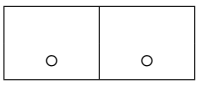


If $|\psi\rangle = |\phi\rangle$ then the swap test will always return a $|0\rangle$. If $\langle\psi|\phi\rangle = 0$, on the other hand, then the measurement will return either $|0\rangle$ or $|1\rangle$ with equal probability. When we apply the swap test to bosons (such as photons), if we find that the outcome is always $|0\rangle$, we have learned experimentally that those bosons really were identical - they are not distinguishable even by properties beyond the knowledge of current physics. There is no analogous way to prove that two particles are truly identical in classical physics.

7.4 Fermions

So far in this lecture we have been looking at bosons, which have the property that if your state looks like (7.2) then $\alpha_{ij} = \alpha_{ji}$ (i.e. the state is restricted to the so-called *symmetric subspace*). If instead we have that $\alpha_{ij} = -\alpha_{ji}$ (we are restricted to the *anti-symmetric subspace*) then we get a different type of particle: the fermion. At a high level, bosons like to bunch together in the same mode. By contrast, two fermions can *never* be in the same mode.

This idea is captured in the Pauli exclusion principle. Since $\alpha_{ii} = -\alpha_{ii}$ we can conclude that $\alpha_{ii} = 0$, so the amplitude on states where both fermions are in the same mode must always be zero. If there are m modes and n fermions, then the number of basis states they can occupy is given by $\binom{m}{n}$. Revisiting our $m = 2$ modes and $n = 2$ particles case, the only possible state is:



Lecture 8: Beamsplitter Networks

8.1 Lecture 7 Review: Occupation Numbers

In the last lecture we discussed the properties of bosonic and fermionic states and how they can be represented.

One way to represent a state with m modes and n particles is to list the mode of each particle. If for all $i \in \{1, \ldots, n\}$, the ith particle is in mode x_i, then such a state would look like

$$|x_1, x_2, \ldots, x_n\rangle .$$

Bosons are indistinguishable, so any permutation of the particles yields the same state:

$$|x_1, x_2, \ldots, x_n\rangle = |x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}\rangle .$$

For fermions, swapping any two particles results in the state taking on a phase of -1:

$$|x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n\rangle = -|x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n\rangle .$$

These rules put restrictions on the amplitudes of bosonic and fermionic states. For instance, since $|2, 3\rangle = |3, 2\rangle$ for bosons, the amplitudes of those states, $\alpha_{2,3}$ and $\alpha_{3,2}$, must be equal in any bosonic state. Similarly, in any fermionic state, $\alpha_{2,3} = -\alpha_{3,2}$. Notably, since $\alpha_{x,x} = -\alpha_{x,x}$ in any fermionic state, we see $\alpha_{x,x} = 0$ for all modes x, so no two fermions can occupy the same mode. This is the **Pauli exclusion principle**.

In order to avoid having these redundant amplitudes, we introduced the occupation number representation. States written in this form are also known as **Fock states**. Instead of listing the modes for each particle, we list the number of particles (the occupation number) for each mode. A Fock state with m modes and n particles will take the form

$$|s_1, s_2, \ldots, s_m\rangle , \text{ where } \sum_{i=1}^{m} s_i = n.$$

For bosons, $s_i \geq 0$ for all i, while for fermions, $s_i \in \{0, 1\}$ for all i.

We can even have a continuum of modes. Quantum fields, the object of study in Quantum Field Theory, are just superpositions over lists of occupation numbers at each location in space. These fields can interact with each other, and particles are bundles of effects traveling through them. In practice, you can never measure the fields directly (the occupation numbers themselves), only these bundles of field effects back-reacting against each other.

8.2 Beamsplitter Networks

For qubit states, we've seen that circuits of unitary gates govern how quantum states evolve. The equivalent of a circuit for bosons and fermions is something called a **beamsplitter network**.

Imagine we have a state with a single particle. (Note that with only one particle, it doesn't matter whether that particle is a boson or fermion.) Then, we can represent our state as

$$\sum_{i=1}^{m} \alpha_i \left|i\right\rangle ,$$

where $\left|i\right\rangle = \left|0,\ldots,1,\ldots,0\right\rangle$ is the state that has occupation number 1 in the ith mode and 0 everywhere else. We can't use gates (tensors of a gate G at some location and I everywhere else) on this state, because our Hilbert space is no longer a tensor product space. Instead, we will use beamsplitters and phase shifters.

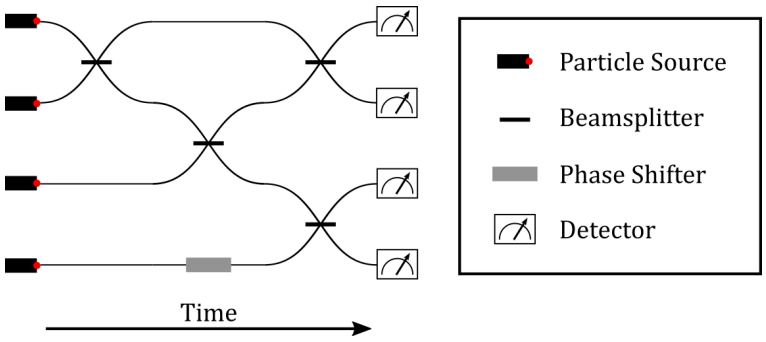


Figure 8.1: An example beamsplitter network.

8.2.1 Beamsplitters

Beamsplitters act on two modes – in practice, often two adjacent modes. A beamsplitter can be represented by a 2×2 unitary matrix. Experimentalists typically restrict these to real matrices of the form

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

The corresponding $m \times m$ unitary (applying R_θ to the first two modes) would be the direct sum

$$R_\theta \oplus \mathbb{I}_{m-2} = \begin{pmatrix} R_\theta & 0 \\ 0 & \mathbb{I}_{m-2} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & & & \\ \sin\theta & \cos\theta & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}.$$

Note that these unitaries are direct sums, not tensor products.

8.2.2 Phase Shifters

Phase shifters apply a complex phase shift to a specific mode. As $m \times m$ unitaries, these are diagonal matrices of the form

$$\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & e^{i\theta} & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix},$$

where the (i,i)th entry, acting on the ith mode, is $e^{i\theta}$, and all other diagonal entries are 1.

Note that since our Hilbert space is only m dimensional (rather than $\exp(m)$-dimensional, like it would be for m qubits) we can classically simulate both beamsplitters and phase shifters efficiently, at least in the 1 particle case.

8.2.3 What can we do with them?

Consider the following beamsplitter network, illustrated here in two equivalent ways:

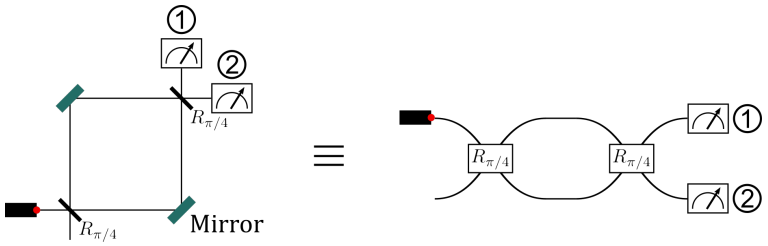


Figure 8.2: A beamsplitter network involving two $R_{\pi/4}$ gates.

With a particle source only on the first mode, the initial state of the network is $|1,0\rangle$. After the first $R_{\pi/4}$ beamsplitter, the state is

$$\frac{|1,0\rangle + |0,1\rangle}{\sqrt{2}}.$$

In other words, the particle is in an equal superposition of being in the first and second modes. After both $R_{\pi/4}$ beamsplitters are applied, the state will be $|0,1\rangle$ due to interference, and a particle will only be detected by the second detector. In other words, $(R_{\pi/4})^2 = X$, as expected.

Classical light (lots of photons—e.g., a laser) acts the same way, but deriving this behavior from the single photon case is non-trivial. Roy J. Glauber won the Nobel Prize in 2005 for doing just this [32]. Physicists have understood classical light for a long time, but understanding the behavior of single photons required amplitudes and quantum mechanics.

We might ask for more general unitary transformations acting on the modes. Is it possible to make a Hadamard? A Quantum Fourier Transform? These questions are answered in the positive by the following theorem from Reck et. al. in 1994.

Theorem 5. *Any unitary $U \in U(m)$ can be realized using $O(m^2)$ beamsplitters and phase shifters [60].*

The proof of this theorem boils down to the fact that Gaussian elimination on an $n \times n$ matrix requires $O(n^2)$ row operations, and we can embed any row operation into beamsplitters and phase shifters. We can then see that this upper bound is tight by a counting argument. Contrast this with the fact that 4^n gates are needed to produce arbitrary $n \times n$ unitaries in the qubit model of computation.

Building off of this, Aaronson and Bouland proved the following in 2014:

Theorem 6. *Every non-trivial 2-mode beamsplitter—i.e. not the product of a permutation and diagonal matrix—densely generates either $U(m)$, the set of all $m \times m$ unitary matrices, or $O(m)$, the set of all $m \times m$ orthogonal matrices [21].*

This implies that there is no beamsplitter analogue of stabilizer gates/states/circuits, because a beamsplitter either generates a trivial subset of unitaries or is universal. This also tells us that the Hadamard beamsplitter is universal.

In practice, beamsplitter networks can be implemented as a mess of mirrors on a table or—as companies like PsiQuantum are starting to do—through etching these networks into silicon.

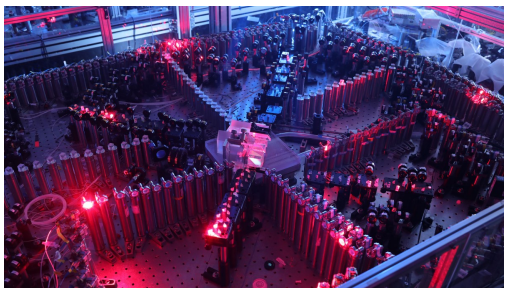


Figure 8.3: A typical linear optical network.

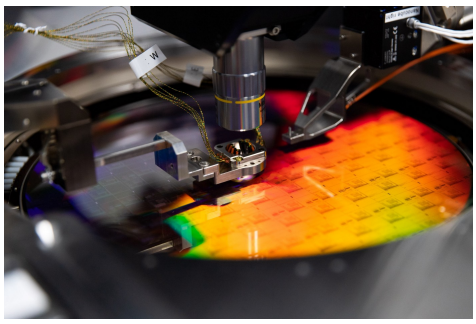


Figure 8.4: PsiQuantum's linear optical networks etched in silicon.

Now what happens if we introduce multiple particles? Nature has two different answers for the two fundamental types of particles, bosons and fermions.

8.2.4 Bosons

Let's imagine passing two bosons through the following simple beamsplitter network, starting with the state $|1,1\rangle$, that is, one boson in each mode.

The Hilbert space for two bosons is 3 dimensional, spanned by the basis states $|1,1\rangle$, $|2,0\rangle$, and $|0,2\rangle$. How do we generalize

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

to this 3D Hilbert space? To answer this question, let's look at amplitudes one at a time.

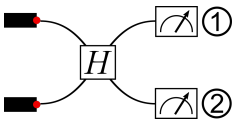


Figure 8.5: A beamsplitter network with 2 modes and a single Hadamard beamsplitter.

Each entry in the Hadamard matrix gives us an amplitude associated with a single particle transitioning from a certain mode to another. For instance, the bottom right entry tells us there is a $-\frac{1}{\sqrt{2}}$ amplitude associated with a particle on the second mode remaining in the second mode, while the top right entry tells us there is a $\frac{1}{\sqrt{2}}$ amplitude associated with a particle on the second mode transitioning to the first mode. What is the amplitude of transitioning from the two-particle state $|1,1\rangle$ to $|1,1\rangle$? There are two ways for this to happen: either the bosons stay in their original modes or they swap. If we add the amplitudes of both of these events, we see

$$\left(\frac{1}{\sqrt{2}}\right)\left(-\frac{1}{\sqrt{2}}\right)+\left(\frac{1}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}}\right)=0,$$

so $\mathbb{P}[|1,1\rangle] = 0$. Then, by symmetry, $\mathbb{P}[|2,0\rangle] = \mathbb{P}[|0,2\rangle] = \frac{1}{2}$. Next lecture, we will see the normalization factor which would let us calculate this $\frac{1}{2}$ probability explicitly.

Notice that after passing through this beamsplitter network, the bosons have become perfectly correlated! This is a famous effect called the "**Hong-Ou Mandel Dip**." Beamsplitters do not cause bosons to interact with each other (through a force, for instance), but it often looks as if they do because of boson statistics.

For general 2×2 beamsplitters

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

we find the probability of remaining in the state with one particle in each mode is

$$\mathbb{P}[\langle 1,1| \rightsquigarrow |1,1\rangle] = ad + bc.$$

With three particles and three modes, applying a general beamsplitter

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

leads to the probability

$$\mathbb{P}[\langle 1,1,1| \rightsquigarrow |1,1,1\rangle] = aei + afh + bfg + bdi + cdh + ceg = \mathrm{Per}(A),$$

where $\mathrm{Per}(A)$ is the so called **permanent** of A. For $A \in \mathbb{C}^{n\times n}$,

$$\mathrm{Per}(A) = \sum_{\sigma\in S_n}\prod_{i=1}^{n} a_{i,\sigma(i)}.$$

The connection between bosons and the permanent has been known by physicists since the 1950's, but it wasn't at the forefront of physicists' attention. In fact, in his quantum mechanics textbook,

physicist Steven Weinberg referred to the permanent as "a determinant, but with all minus signs replaced with plus signs," thinking readers wouldn't have heard of something as obscure as the permanent [74]. However, the permanent is a useful function in many other contexts too—particularly in algorithms and complexity theory. For instance, if a $\{0,1\}$ matrix is the adjacency matrix of a bipartite graph, then its permanent gives the number of perfect matchings in that graph.

In 1979, Leslie Valiant proved the following:

Theorem 7. *The problem of computing a matrix's permanent is* #P*-complete [69].*

#P is a complexity class believed to be above even NP. Informally, it's the class of all combinatorial counting problems: for example, "given a Boolean formula $\varphi(x_1,\ldots,x_n)$, how many satisfying assignments does it have?" In a future lecture, we will use quantum optics to give a more elegant proof of Valiant's result, but for now, we'll just accept it.

This leads to a natural next question: do bosons give you a way to calculate permanents more efficiently? As it turns out, if you have a specific matrix in mind, there's no obvious speedup, but if you want to sample random matrices with a bias toward matrices with large permanents, you *can* get an apparent speedup—this is **BosonSampling**.

8.2.5 Fermions

Now let's look at the same beamsplitter network from Figure 8.5 applied to fermions. We know from the Pauli exclusion principle that we'll end up with the state $|1,1\rangle$ with probability 1, but let's derive this. Like before, we need to add up the amplitudes associated with both particles remaining in their original modes and both particles swapping. This time, however, we need to flip the sign on the amplitude associated with swapping, because fermions pick up a phase of -1 when they swap modes.

$$\left(\frac{1}{\sqrt{2}}\right)\left(-\frac{1}{\sqrt{2}}\right)-\left(\frac{1}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}}\right)=-1,$$

so $\mathbb{P}[|1,1\rangle] = |-1|^2 = 1$. In general, fermion amplitudes are given by determinants. If we apply a beamsplitter A to n fermions in n modes, the probability of remaining in the state with one fermion in each mode is given by $|\mathrm{Det}(A)|^2$. Since A is a unitary matrix, $|\mathrm{Det}(A)| = 1$, so that probability is 1. This is exactly what the Pauli exclusion principle suggests—there is only one possible way to put n fermions in n modes. By comparison, for any unitary A, we'll prove later that $|\mathrm{Per}(A)| \leq 1$, which makes sense because $\mathrm{Per}(A)$ must be an amplitude.

We've seen that with n particles and m modes, beamsplitters are represented by $m \times m$ unitaries, but the Hilbert spaces they act on can have a much larger dimension, say M, depending on n, m, and whether our particles are bosons or fermions. In the next lecture, we will learn how to explicitly construct the appropriate $M \times M$ unitary from any given $m \times m$ beamsplitter unitary.

Lecture 9: Fermionic Linear Optics

9.1 Overview

9.1.1 Last Lecture

In the last lecture introduced the use of linear-optical networks (also known as "beamsplitter networks") to perform unitary transformations on boson and fermion states. Linear-optical networks are made up of beamsplitters, which are essentially 2×2 unitary transformations that act on 2 out of the m total modes. We can also have another optical element called a phaseshifter, which can apply a complex phase to a single mode of our choice.

Crucially, beamsplitters and phaseshifters are not combined via tensor product — they're instead combined via direct sum. In other words, if we have m modes, then we apply a sequence of $m \times m$ unitaries that look like

$$\begin{pmatrix} \cos\theta & -\sin\theta & & & \\ \sin\theta & \cos\theta & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} \text{ or } \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & e^{i\theta} & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix}$$

which represent individual beamsplitters and phaseshifters, respectively. Those matrices are multiplied together to obtain the overall $m \times m$ unitary transformation applied by a linear-optical network. We also saw last lecture that any unitary U can be implemented by a beamsplitter network using $O(m^2)$ beamsplitters and phaseshifters.

This picture perfectly captures the behavior of individual particles, but we're interested in what happens when multiple particles travel through a beamsplitter network together. Specifically, suppose we have a starting state $|S\rangle = |s_1, s_2, \ldots, s_m\rangle$, where $s_i \geq 0$ and $s_1 + \cdots + s_m = n$, that represents the occupation numbers of n particles in m modes. What happens when the particles pass through a beamsplitter network that applies the unitary transformation U? Let's call that action $\varphi(U)$. Then for fermions we have

$$\langle S| \varphi(U) |T\rangle = \det(U_{S,T}).$$

In that formulation, $|T\rangle$ is a possible state after the beamsplitter network is applied, and $U_{S,T}$ is the submatrix of U formed by the rows and columns corresponding to nonzero occupation numbers

in $|S\rangle$ and $|T\rangle$, respectively. In the case of n fermions with m modes, $\varphi(U)$ is an $M \times M$ matrix where $M = \binom{m}{n}$, which is the number of possible configurations of the fermions.

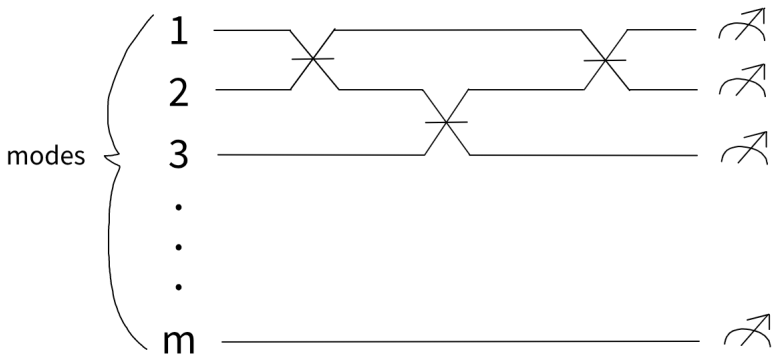
9.1.2 This Lecture

In this lecture we will continue discussing the $M \times M$ matrix $\varphi(U)$ which describes the transformation of n particles with m modes in a linear-optical network that applies a unitary transformation U. We're specifically interested in the case of fermions in this lecture, and we want to answer two basic questions: Is $\varphi(U)$ unitary? And is φ a homomorphism, i.e. $\varphi(UV) = \varphi(U)\varphi(V)$? We will also describe a fast classical algorithm for simulating fermionic linear optics.

9.2 Fermionic Linear Optics

9.2.1 Ordering the Modes

Perhaps the most subtle part in our analysis of fermions in a linear-optical network is that we must first assign some order to the m modes, labeling them from 1 to m. The exact order chosen does not matter, but must be held consistent. Recall that any $m \times m$ unitary U can be decomposed into a product of $O(m^2)$ simple linear-optical elements (beamsplitters and phaseshifters). Interestingly, an even stronger assertion holds: we can decompose U into a product of linear-optical elements where each beamsplitter, a 2×2 unitary transformation, acts only on two adjacent modes, and adjacency is defined by the ordering of modes we chose. Our linear-optical network then looks like



The reason for this is that if we wanted to implement an operation on two non-adjacent modes i and j, we could add in swaps to make i adjacent to j. Alternatively, we could just prove the decomposition theorem using only row operations that act on adjacent rows.

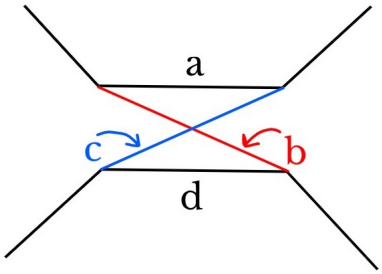
Already with a single particle, we can think of a final amplitude as a sum over the exponentially many different possible paths that the particle could take through the linear-optical network. As we add more beamsplitters, the possible paths multiply. In QIS I, we talked a bit about the Feynman view of quantum mechanics, where we calculate each final amplitude as a sum over all possible contributions to that amplitude. That's exactly how we're thinking about summing the

contributions of all possible paths a particle can take through a linear-optical network. Each path contribution is a product of a bunch of numbers, one for each beamsplitter passed on that path.

Each beamsplitter in a linear-optical network has an associated 2×2 unitary matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

In the sum-over-paths picture, the interpretation of that matrix is that it gives the number to multiply by when computing the amplitude contribution of a path, as a function of where the particle enters and exits the beamsplitter. Visually, that interpretation looks like



So, if a particle enters from the top and exits out the top, then we multiply by a. Likewise, if a particle enters from the bottom and exits out the bottom, we multiply by d.

Now consider the case of two particles. We still want to consider all possible paths the particles can take, but we now need to sum over all possible **pairs** of paths. In the case of fermions, however, the defining property is that we have to multiply by -1 for every swap of the particles.

9.2.2 Qubit Representation

Suppose that instead of fermions we have qubits, where each qubit corresponds to the occupation number of a mode. For example, the string $|1001\rangle$ means that the first and fourth modes are occupied while the second and third are unoccupied. Our goal is to simulate beamsplitters using 2-qubit gates. Consider encoding the behavior of a beamsplitter with the 2×2 unitary given earlier using the following 2-qubit gate:

$$\begin{array}{c} \\ 00 \\ 10 \\ 01 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 10 & 01 & 11 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & a & b & 0 \\ 0 & c & d & 0 \\ 0 & 0 & 0 & ad-bc \end{pmatrix} \end{array} \tag{9.1}$$

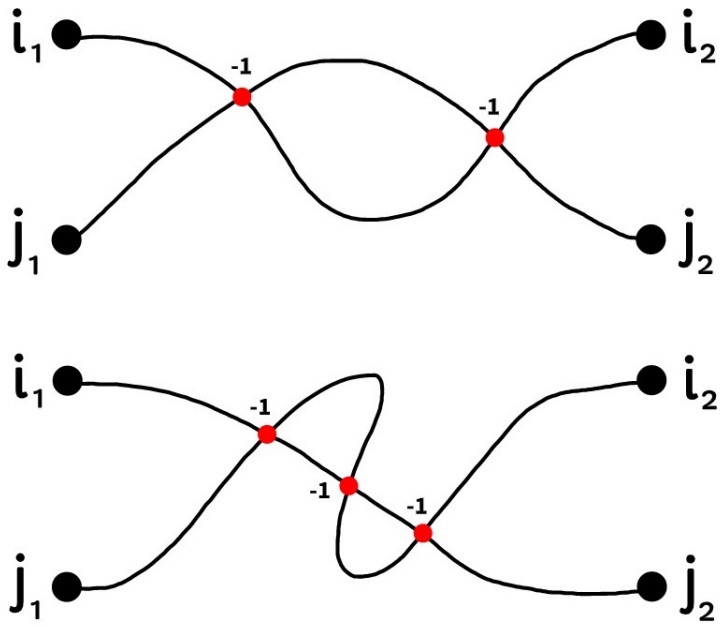
Note that beamsplitters don't create or destroy particles, so we must have a 0 entry between any two bit strings of different Hamming weights. The center entries a, b, c, d correspond exactly to the behavior of the beamsplitter in the single particle case. The reason for the bottom-right $ad - bc$ entry is that two fermions passing through the beamsplitter can either stay in the modes in which they entered, giving a contribution of ad, or they can swap, picking up a -1 phase and giving a

contribution of $-bc$. This is why we enforced an ordering on the modes — in doing so, we've given a well-defined meaning for when two fermions swap.

Importantly, the entry $ad - bc$ is exactly the determinant of the beamsplitter's associated 2×2 unitary matrix, so we have $|ad - bc| = 1$. It then follows that the given 4×4 matrix is unitary, as it's made up of 3 unitary blocks. So our 4×4 unitary matrix is a 2-qubit gate, and we can then imagine applying that gate to a pair of qubits whenever we want to simulate a beamsplitter acting on the two corresponding modes.

Now consider the m-qubit unitary applied by our quantum circuit of 2-qubit gates, which represent beamsplitters acting on adjacent modes. We're interested in that m-qubit unitary's action on the subspace spanned by bit strings of Hamming weight n, representing the n fermions with m possible modes. The unitary's action on that subspace is exactly equivalent to the action of $\varphi(U)$, namely $\langle S| \varphi(U) |T\rangle = \det(U_{S,T})$. From this, we can conclude that $\varphi(U)$ is unitary and also a homomorphism, as it corresponds to a block of our constructed m-qubit unitary transformation.

But why does this equivalence hold? Because, crucially, each contribution to an amplitude only depends on all the particles' beginning and ending states, not the specific paths taken through the beamsplitter network. Whenever two fermions swap their order, their paths must cross an odd number of times and hence they must pick up a -1 phase. Likewise, whenever two fermions do not swap their order, their paths must cross an even number of times and hence the phase must be 1. Example diagrams for two fermions swapping their order an even or odd number of times and picking up a 1 or -1 phase, respectively, are given below:



So if $\langle i_1| U |i_2\rangle \langle j_1| U |j_2\rangle$ denotes the sum over the contributions of all paths with an even number of crossings and $\langle i_1| U |j_2\rangle \langle j_1| U |i_2\rangle$ denotes the sum over the contributions of all paths with an odd number of crossings, then the amplitude for two fermions ending in modes i_2, j_2 after starting in modes i_1, j_1 is given by

$$\langle i_1| U |i_2\rangle \langle j_1| U |j_2\rangle - \langle i_1| U |j_2\rangle \langle j_1| U |i_2\rangle .$$

This is what leads to the determinant as the overall amplitude of going from an n-fermion starting state $|S\rangle$ to an n-fermion ending state $|T\rangle$.

9.2.3 Reinventing Fermions

The 2-qubit gate picture we developed is known by physicists as the Jordan-Wigner representation. Amusingly, it was later rediscovered by Leslie Valiant, who also proved that computing the permanent is #P-complete and won the Turing award. In his 2002 paper "Quantum Circuits That Can Be Simulated Classically in Polynomial Time," he gives a classical polynomial algorithm for simulating quantum circuits that are restricted to 2-qubit gates of the form given in 9.1 where each gate can only act on two neighboring qubits [71]. Valiant called them **matchgates**.

Not long after Valiant's paper, Terhal and DiVincenzo published a paper stating that what Valiant had actually done was rediscover fermions [67]. In other words, Valiant's subclass of quantum computation can be interpreted such that qubits are just occupation numbers for fermionic modes.

More precisely, Valiant's subclass of quantum computation also allowed for nonzero values (satisfying certain restrictions) in the top right and bottom left entries of the 4×4 matrices for the 2-qubit gates. But this also has a physical interpretation: it corresponds to the possibility of a particle-antiparticle pair being created out of the vacuum or annihilating itself.

9.3 Classically Simulating Fermions in Polynomial Time

We now describe how to simulate a fermionic beamsplitter network classically in polynomial time. Consider the $m \times m$ unitary U that represents a linear-optical network. If we take its leftmost n columns, corresponding to the first n modes, then we have an $m \times n$ column-orthonormal matrix A. Now assume without loss of generality that our initial state looks like $|I\rangle = |1, \ldots, 1, 0, \ldots, 0\rangle$ where the first n modes are occupied by fermions and the remaining $m - n$ modes are unoccupied. After applying our fermionic beamsplitter network, the amplitude of the output state $|S\rangle$ is

$$\langle I| \varphi(U) |T\rangle = \det(A_S)$$

where A_S is the $n \times n$ submatrix of A formed by the n rows determined by S. So in particular, our output state is a superposition given by

$$|\psi\rangle = \sum_{\substack{S \subseteq \{1,\ldots,m\} \\ |S|=n}} \det(A_S) |S\rangle .$$

The question then becomes: what can we know about $|\psi\rangle$ in classical polynomial time? Clearly, it is an exponentially large quantum state involving an exponential number of amplitudes, so we have no hope of writing out all the amplitudes explicitly. We can, however, compute the amplitude of any specific basis state $|S\rangle$, since the determinant is in P.

Let us instead define our simulation task to be: sample an $|S\rangle$ with probability $|\det(A_S)|^2$. It turns out that this can be done in classical polynomial time as well. Why?

Well, one thing we know about A is that its columns are orthonormal, since they came from a unitary matrix. Let $|v_1\rangle, \ldots, |v_m\rangle$ be the row vectors of A. Observe then that

$$\sum_{i=1}^{m} \langle v_i | v_i \rangle = n$$

because it is equivalently the sum of the squared norms of the n columns of A, which are unit vectors. Let us then sample a mode for our first output fermion, where mode i is sampled with probability

$$\frac{\langle v_i | v_i \rangle}{n}.$$

We then project each $|v_j\rangle$ onto the orthogonal complement of the sampled $|v_i\rangle$, yielding $|v'_j\rangle$. Note that the projection of $|v_i\rangle$ onto its orthogonal complement is the zero vector, which is a direct implementation of the Pauli Exclusion Principle: once a mode is chosen for a fermion, it's no longer available for any other fermion. We'll leave as an exercise to prove that

$$\sum_{j=1}^{m} \langle v'_j | v'_j \rangle = n - 1.$$

We then sample the mode of the next fermion where mode j is sampled with probability

$$\frac{\langle v'_j | v'_j \rangle}{n-1}.$$

This process is repeated until n modes have been sampled, one for each of the n fermions, leaving us—as we'll see next time—with a final state $|S\rangle$ that was sampled with probability $|\det(A_S)|^2$. This classical algorithm runs in polynomial time, specifically $O(mn^2)$ time.

Lecture 10: Fock Space

10.1 Overview

In the last lecture we talked about unitary transformations on the Hilbert spaces of bosons and fermions. Formally, any unitary transformation on this space is allowed and is physically implementable in principle, but the vast majority of the unitaries require us to do things that interact the particles with each other. Once we have such interactions, we get universal quantum computation.

When we talk about bosons and fermions, there is a special subclass of unitary transformations that can be implemented using what are called *linear-optical networks*. Particles are passed through a network of linear optical elements called beamsplitters and phaseshifters. The bosons and fermions never actually interact even though it appears so (as we saw with the Hong-Ou-Mandel effect). This behavior occurs because of the way boson and fermion statistics work.

Any unitary obtained by using linear optical elements can be specified by an $m \times m$ unitary U acting on the 1-photon subspace, and is much smaller than the size of a general unitary acting on the n-particle Hilbert space, where m is the total number of modes. Indeed, the dimension of the latter Hilbert space is $\binom{m}{n}$ for fermions and $\binom{m+n-1}{n}$ for bosons, both of which can be exponentially large. We now face the following question: given the $m \times m$ unitary matrix U corresponding to a linear-optical network acting on the 1-particle Hilbert space, how do we induce from it the unitary acting on the n-particle Hilbert space? We claim that this induced unitary, call it $\varphi(U)$, can be defined as follows: for any n-particle states $|S\rangle = |s_1, \ldots, s_m\rangle$ and $|T\rangle = |t_1, \ldots, t_m\rangle$,

$$\langle S|\varphi(U)|T\rangle = \begin{cases} \mathrm{Det}(U_{S,T}) & \text{for fermions, and} \\ \\ \frac{\mathrm{Per}(U_{S,T})}{\sqrt{s_1!s_2!\ldots s_m!t_1!t_2!\ldots t_m!}} & \text{for bosons.} \end{cases}$$

In order to justify these formulas, it must be that

1. $\varphi(U)$ is a unitary transformation, and
2. φ is a homomorphism, that is, $\varphi(U)\varphi(V) = \varphi(UV)$ for unitaries U and V.

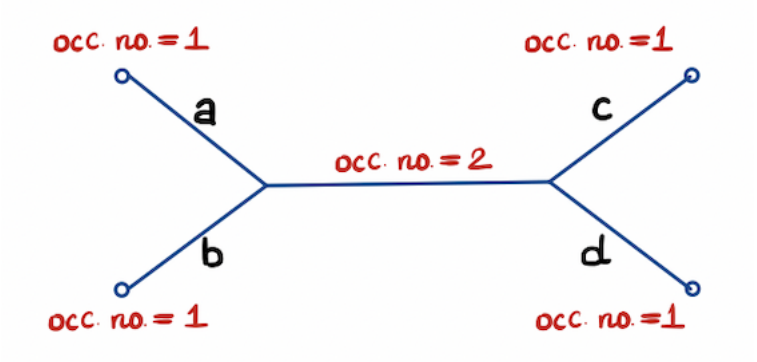
Last lecture, we justified these relations for fermions. We also discussed the FermionSampling problem and showed that there is a probabilistic classical algorithm that takes as input a column-orthonormal matrix $M \in \mathcal{C}^{m \times n}$ and samples from a certain "fermionic" distribution $\mathcal{F}_M$ in $O(mn^2)$ time.

In this lecture, we justify the formula for bosons. In particular, we will attempt to demystify the peculiar form of the formula and show that $\varphi(U)$ is indeed a unitary transformation.

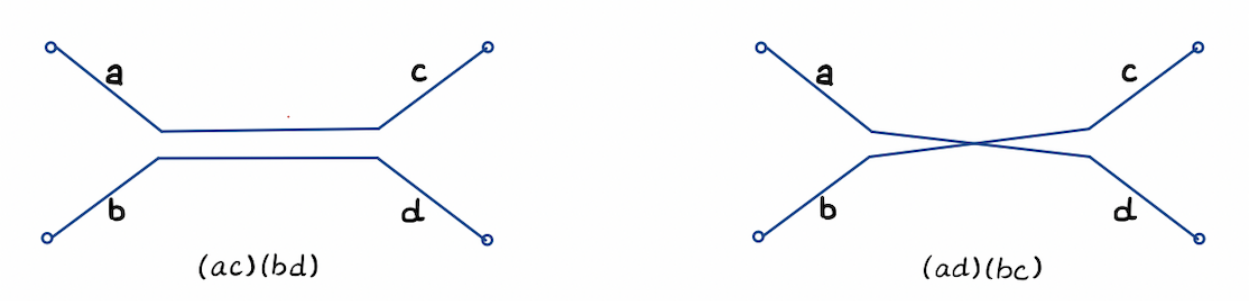
10.2 Back to the Feynman Picture

Similar to what we did last lecture, we will sum over all possible collections of paths that can be followed by n bosons from an initial state $|S\rangle$ to a final state $|T\rangle$. Recall that for fermions, we argued that each time two paths cross we get a multiplier of -1, and that the overall sign of a particular tuple of paths is determined completely by the sign of the underlying permutation.

In the case of bosons, we do not need to worry about paths crossing because there is no change in sign. What, then, is the difficulty with bosons? It is this: since bosons can share a mode, we may have two paths that merge together. Consider the following picture, where the labels a, b, c and d are the amplitudes.



The contribution of this structure to the overall amplitude should be $2abcd$, because the bosons end up in their original positions with amplitude $(ac)(bd) = abcd$ and end up swapping positions with amplitude $(ad)(bc) = abcd$. Recall that if paths don't merge together and each boson ends up in a different state, then we have the permanent (similar to the determinant for fermions) that tells us what the corresponding amplitude of the event is. How do we generalize this when the bosonic paths do end up merging and splitting?

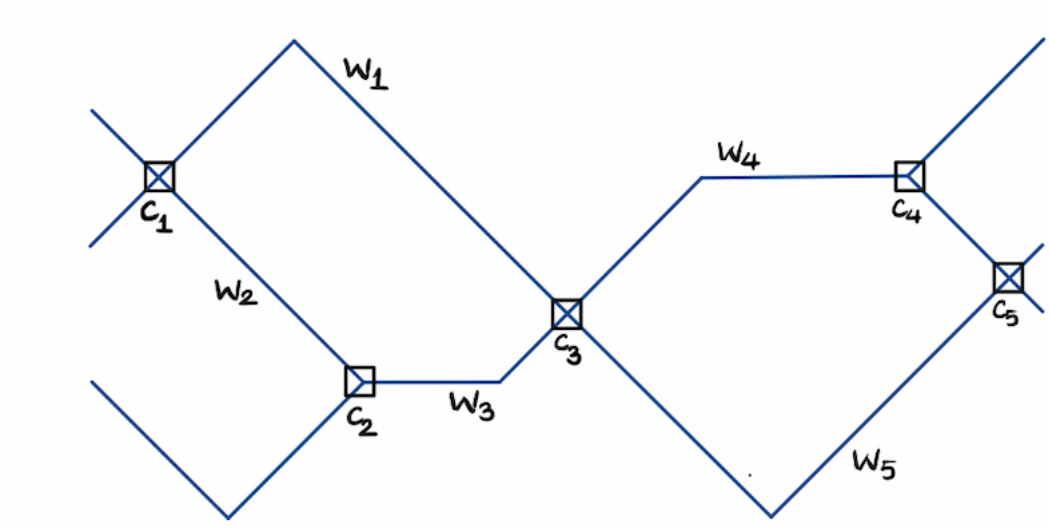


Recall how we defined $U_{S,T}$ for bosonic states $|S\rangle$ and $|T\rangle$: we take s_i copies of the ith column and t_j copies of the jth row in U for all i, j. We then take the permanent of the resulting matrix.

For example, in the beginning of the structure above when the paths collide, the permanent to be considered is $\text{Per}\left(\begin{pmatrix} a & b \\ a & b \end{pmatrix}\right) = 2ab$. Similarly, at the end of the structure when the paths split, the permanent to be considered is $\text{Per}\left(\begin{pmatrix} c & d \\ c & d \end{pmatrix}\right) = 2cd$. However, $(2ab)(2cd) = 4abcd$, rather than $2abcd$ as we wanted. So, there is a factor of $\frac{1}{2}$ that we need to account for. We account for it symmetrically by simply dividing the "labor" onto the merge and the split by multiplying both permanents by $\frac{1}{\sqrt{2}}$.

Now suppose we have n input modes each with an occupation number of 1, and suppose they all combine and then all split up. There are $n!$ sets of paths we are dealing with here. But if we naïvely write the permanent on both ends, we will get an $(n!)^2$ multiplicative factor ($n!$ from each). We account for this extra $n!$ factor by dividing by $\sqrt{n!}$ on both ends.

Generalizing even more, we can have a much more complicated structure. If there are k internal vertices in this structure, where the i-th vertex has c_i incoming bosons, then we want to multiply by $c_1!c_2!\dots c_k!$ overall. This is because at each of these vertices there is an arbitrary permutation of the bosons that gives us a valid set of paths. But this overcounts! We do not care about permutations between bosons that can only be distinguished within some internal wire. Therefore, if the edges between two vertices in this structure carry $w_1, w_2, \dots, w_l$ bosons, then we want to divide by $w_1!w_2!\dots w_l!$ overall. If we naively write the permanent on each vertex then we get an overall multiplicative factor of $c_1!c_2!\dots c_k!$ only. We need to put in the $\frac{1}{w_1!w_2!\dots w_l!}$ factor by hand, and we do this by dividing the work of $\frac{1}{w_i!}$ on both ends of the edge as $\frac{1}{\sqrt{w_i!}}$ on both vertices that the i-th edge is incident on. This explains the idea behind the normalization factor of $\frac{1}{\sqrt{s_1!s_2!\dots s_m!t_1!t_2!\dots t_m!}}$ in the formula for $\langle S|\varphi(U)|T\rangle$ in the case of bosons.



It is possible to extend this to show that φ is a homomorphism. Howeover, it is still not clear why $\varphi(U)$ is a unitary matrix. In order to explain this, we switch our way of thinking about bosons by introducing *creation operators*, a favorite tool of the physicists.

10.3 Viewing Bosonic and Fermionic States as Polynomials

We define formal variables $x_1, x_2, \ldots, x_m$, one for each mode. These formal variables are called *creation operators*; we can think of x_i as taking an action on mode i by adding a particle to that mode. There are also operators that delete a particle from a mode, called *annihilation operators*.

The creation operators satisfy certain algebraic rules. Bosons satisfy $x_i x_j = x_j x_i$ for all i, j, and fermions satisfy $x_i x_j = -x_j x_i$ for all i, j. Note that this implies that $x_i^2 = 0$ for fermions. The state $|S\rangle = |s_1, \ldots, s_m\rangle$ is represented by the monomial $x_1^{s_1} x_2^{s_2} \ldots x_m^{s_m}$. The effect of a linear-optical network can be viewed as a unitary matrix U that transforms a polynomial $p(x)$ in $x_1, \ldots, x_m$ to another polynomial $p(Ux)$; this kind of a transformation is called a *Bogoliubov transformation*. Let us look at an explicit example of this transformation.

Consider the input state to be $|1, 1\rangle$, with corresponding polynomial $x_1 x_2$. Now, suppose we apply the Hadamard beamsplitter

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Since x_1 maps to $\frac{x_1+x_2}{\sqrt{2}}$ and x_2 maps to $\frac{x_1-x_2}{\sqrt{2}}$, the output polynomial is

$$\left(\frac{x_1 + x_2}{\sqrt{2}}\right)\left(\frac{x_1 - x_2}{\sqrt{2}}\right) = \frac{x_1^2}{2} - \frac{x_2^2}{2} + \frac{x_2 x_1}{2} - \frac{x_1 x_2}{2}.$$

For fermions, this simplifies to $x_2 x_1$ because $x_1^2 = x_2^2 = 0$ and $x_1 x_2 = -x_2 x_1$. For bosons, this simplifies to $\frac{x_1^2}{2} - \frac{x_2^2}{2}$ because $x_1 x_2 = x_2 x_1$. We need to fix the formula for bosons because the formula currently says that the amplitudes should be $\frac{1}{2}$ and $\frac{1}{2}$ which is not right. Following the prescription from the previous section, we can fix it by multiplying $\sqrt{2}$ and obtaining the correct polynomial $\frac{x_1^2}{\sqrt{2}} - \frac{x_2^2}{\sqrt{2}}$.

In general, a homogeneous degree n polynomial on $x_1, \ldots, x_m$ looks like

$$\sum_{S=(s_1,\ldots,s_m)} a_S \; x_1^{s_1} x_2^{s_2} \ldots x_m^{s_m}$$

where S varies over all *allowed* n-particle states on these m modes. This polynomial corresponds to the quantum state $|\psi\rangle = \sum_S \alpha_S |S\rangle$, where the amplitude α_S depends only on a_S and S. In fact, the amplitude α_S satisfies

$$\alpha_S = a_S \cdot \sqrt{s_1! s_2! \ldots s_m!}.$$

Now, we want to view the set of polynomials as a vector space with an inner product, a special one, called the *Fock inner product*.

10.3.1 Fock Inner Product

The Fock inner product of two polynomials

$$p(x_1, \ldots, x_m) = \sum_{S=(s_1,\ldots,s_m)} a_S \; x_1^{s_1} x_2^{s_2} \ldots x_m^{s_m}$$

and

$$q(x_1, \ldots, x_m) = \sum_{S=(s_1,\ldots,s_m)} b_S \; x_1^{s_1} x_2^{s_2} \ldots x_m^{s_m},$$

where S varies over all m-tuples of non-negative integers with sum n, is defined as

$$\langle p, q \rangle = \sum_S a_S^* b_S \; s_1! s_2! \ldots s_m!$$

The space formed by these polynomials along with the Fock inner product is called the *Fock space*. Now, let $U[p]$ be the Bogoliubov transformation on the polynomial p, that is, $U[p](x) = p(Ux)$ for all x.

Theorem 8. *For all polynomials p and q, $\langle p, q \rangle = \langle U[p], U[q] \rangle$.*

Proof. We first claim that

$$\langle p, q \rangle = \mathbb{E}_{x \sim \mathcal{G}^m}[p(x)^* g(x)],$$

where $\mathcal{G} = \mathcal{N}(0,1)_{\mathbb{C}}$ is the complex Gaussian measure with mean 0 and variance 1. Due to linearity of expectation, it suffices to prove the claim for monomials p and q with leading coefficients 1.

Let $S = (s_1, \ldots, s_m), R = (r_1, \ldots, r_m)$, $p(x) = x_1^{s_1} \cdot \ldots \cdot x_m^{s_m}$ and $q(x) = x_1^{r_1} \cdot \ldots \cdot x_m^{r_m}$. If $S \neq R$ then there is some index i such that $s_i \neq r_i$. Without any loss in generality let $i = 1$ and $r_1 > s_1$. Then, $p(x)^* q(x)$ contains the term $x_1^{s_1 *} x_1^{r_1} = |x_1|^{2s_1} x_1^{r_1 - s_1}$. Since $x_1^{r_1 - s_1} \neq 0$, we get by radial symmetry that

$$\mathbb{E}_{x_1 \sim \mathcal{G}}[|x_1|^{2s_1} x_1^{r_1 - s_1}] = 0.$$

Since the x_i are sampled independently, $\mathbb{E}_{x \sim \mathcal{G}^m}[p(x)^* q(x)] = 0$ in this case, as required.

Instead, let $S = R$. Then $p(x)^* q(x) = |x_1|^{2s_1} \cdot \ldots \cdot |x_m|^{2s_m}$. It can be shown by doing a suitable integral that $\mathbb{E}_{x_1 \sim \mathcal{G}}[|x_1|^{2s_1}] = s_1!$. Using the fact that the x_i's are independent, $\mathbb{E}_{x \sim \mathcal{G}^m}[p(x)^* g(x)] = s_1! \cdot \ldots \cdot s_m!$, as required.

To finish the proof, note that a unitary transformation U only rotates on the complex plane. Since the Gaussian measure has radial symmetry,

$$\langle p, q \rangle = \langle U[p], U[q] \rangle$$

□

Now, the point of defining the Fock inner product is this: the homogeneous degree n polynomial $\sum_{S=(s_1,\ldots,s_m)} a_S \; x_1^{s_1} x_2^{s_2} \ldots x_m^{s_m}$, where S varies over all *allowed* n particle states on these m modes, corresponds to the quantum state $|\psi\rangle = \sum_S \alpha_S |S\rangle$, where the amplitude $\alpha_S = a_S \cdot \sqrt{s_1! s_2! \ldots s_m!}$. Therefore, if these amplitudes correspond to probabilities that add up to 1, then a Bogoliubov transformation preserves this property of the amplitudes. More generally, we want that a Bogoliubov transformation should preserve angles between any two quantum states. Theorem 1 above proves exactly this, from which we conclude that φ is unitary.

Lecture 11: BosonSampling Basics

11.1 Overview

In the last lecture we talked about unitary transformations that act on bosons. In this lecture we will start talking about BosonSampling and its computational complexity.

11.2 Recap

Suppose we have some $m \times m$ unitary transformation U, which can be physically implemented by a network of beamsplitters, acting on some number of bosonic modes.

We can take U and lift it to a larger unitary $\varphi(U)$ acting on a much larger Hilbert space, spanned by basis states of the form $|S\rangle = |s_1, \ldots, s_m\rangle$ where the s_i's are occupation numbers.

We define $\varphi(U)$ as follows:

$$\langle S| \varphi(U) |T\rangle = \frac{\mathrm{Per}(U_{S,T})}{\sqrt{s_1! \ldots s_m! t_1! \ldots t_m!}} \tag{11.1}$$

where $|S\rangle = |s_1, \ldots, s_n\rangle$, $|T\rangle = |t_1, \ldots, t_n\rangle$, Per is the permanent, and $U_{S,T}$ is an $n \times n$ matrix obtained from U by taking the s_i copies of the i^{th} row and t_j copies of the j^{th} column for all i, j.

In the case of fermions, this is just an $n \times n$ submatrix of U. For bosons there's the complication that there can be repeated rows.

Last class we proved two properties of $\varphi(U)$:

1. $\varphi(U)$ is unitary
2. $\varphi(UV) = \varphi(U)\varphi(V)$ for all U, V (φ is a homomorphism)

11.3 Bosons as polynomials

Let p be a homogeneous polynomial of degree n in variables $x_1, \ldots, x_m$:

$$p(x_1, \ldots, x_m) = \sum_{S=(s_1,\ldots,s_m)} a_S x_1^{s_1} x_2^{s_2} \cdots x_m^{s_m} \tag{11.2}$$

where $S = (s_1, \ldots, s_m)$ ranges over all lists of nonnegative integers such that $s_1 + \cdots + s_m = n$. We assert that there is a correspondence between polynomials such as p and bosonic quantum states:

$$|\varphi_p\rangle = \sum_{S=(s_1,\ldots,s_m)} a_S \sqrt{s_1! \ldots s_m!}\, |S\rangle \,. \tag{11.3}$$

We can say that p is tracking the same information as $|\varphi_p\rangle$. Why is this useful? One thing we can do when dealing with a polynomial is to apply a linear transformation to its variables. Given the multi-variate polynomial p and a $n \times n$ unitary transformation U, we can define a new polynomial $U[p]$ as follows:

$$p \longrightarrow U[p] = p(U\vec{x}). \tag{11.4}$$

All this means is that, letting

$$\begin{pmatrix} x'_1 \\ \vdots \\ x'_m \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mm} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix},$$

we use the vector of x'_is as input to p instead of the vector of x_is. We'll see later that this directly corresponds to $\varphi(U)$, and to what actually happens when we apply unitaries to our states of n bosons.

11.4 Fock inner product

Suppose we have two polynomials p and q, where p has the coefficients a_S and q has the coefficients of b_S. Then we can define the Fock-space inner product as follows:

$$\langle p, q\rangle = \sum_{S=(s_1,\ldots,s_m)} a_S^* b_S s_1! \ldots s_m! \tag{11.5}$$

In the previous lecture, we proved that unitary transformation preserves the Fock inner product:

Theorem 9 (Unitary Invariance of Fock Inner Product). *$\langle p, q\rangle = \langle U[p], U[q]\rangle$ for all polynomials p,q and all unitary transformations U.*

11.5 How do we get between polynomials and permanents?

To see the connection, start by imagining we have a monomial $x_1 x_2 \cdots x_n$. The x_is are creation operators that act on the first n modes, yielding a state where there is one photon in each of the first n modes and none in the remaining $m - n$ modes. Now let's imagine we perform a unitary on these creation operators such as that from equation (11.4); then we have the following:

$$U[x_1 \cdots x_n] = (U_{11}x_1 + \ldots + U_{m1}x_m) \cdots (U_{1n}x_1 + \ldots + U_{mn}x_m) \tag{11.6}$$

$$= \sum_{S=(s_1,\ldots,s_m)} \mathrm{Per}(U_S) x_1^{s_1} x_2^{s_2} \cdots x_m^{s_m} \tag{11.7}$$

where $\mathrm{Per}(U_S)$ is the permanent of U_S, and U_S is the $n \times n$ submtarix formed from U by first taking the leftmost n columns, then taking the s_i copies of the i^{th} row for all i. We get $\mathrm{Per}(U_S)$ because when we multiply the linear terms, we need to sum over $n!$ possible ways of getting the monomial $x_1^{s_1} x_2^{s_2} \cdots x_m^{s_m}$.

More generally, suppose we have a monomial corresponding to the basis state $|S\rangle$, namely $x_1^{s_1} \cdots x_m^{s_m}$, and a monomial corresponding to the basis state $|T\rangle$, namely $x_1^{t_1} \cdots x_m^{t_m}$. Applying the Fock inner product on these two monomials while applying a U to the second monomial we have:

$$\langle x_1^{s_1} \cdots x_m^{s_m}, U[x_1^{t_1} \cdots x_m^{t_m}] \rangle = \mathrm{Per}(U_{S,T}) \tag{11.8}$$

$$= \langle S | \varphi(U) | T \rangle \sqrt{s_1! \ldots s_m! t_1! \ldots t_m!} \tag{11.9}$$

where (11.9) follows from the definition of $\varphi(U)$. This, finally, proves that $\varphi(U)$ is unitary, since we've already seen that U induces a unitary transformation on Fock polynomials.

11.6 Using bosons to do quantum computation

Consider the following facts:

1. The permanent of an $m \times m$ matrix is a #P-complete function [69]
2. Permanents arise as bosonic amplitudes.

Putting these two facts together, it would seem that in order to calculate their own evolution, bosons have to solve a #P-complete problem. This seems like a staggering claim! Indeed, it seems to be positing some immense amount of computational power on the part of nature. Certainly much much more than in the case of fermions; fermions "got the easy job" and only have to deal with determinants. This naturally invites the speculation that we could build some bosonic system in order to calculate a permanent of our choice, and thereby solve a #P-complete problem in polynomial time. This means in particular that we could solve NP-complete problems in polynomial time. What goes wrong if we try?

Suppose we have a matrix $X \in \mathbb{C}^{n \times n}$, and we would like to calculate $|\mathrm{Per}(X)|^2$. Note that there is no loss of #P-completeness here; this is already a #P-complete problem. Naturally, we could build some beamsplitter network that gives rise to the matrix X. However, there are two difficulties.

11.6.1 Embed X into a unitary matrix

The first difficulty is that X need not be unitary. To solve this problem, we can construct a $2n \times 2n$ unitary matrix U such as X arises as the top-left submatrix of U:

$$U = \left(\begin{array}{c|} X & \\ \hline & \end{array}\right).$$

Strictly speaking, of course, in order for this to work, we will have to normalize X. Let

$$Y = \frac{X}{||X||} \tag{11.10}$$

where $||X||$ is the spectral norm of X (i.e. the largest singular value of X).

Claim 10. *There exists a $2n \times 2n$ unitary U that has Y as its top-left $n \times n$ submatrix,*

$$U = \left(\begin{array}{c|} Y & \\ \hline & \end{array}\right)$$

Proof. As a first observation, it suffices to construct a $2n \times n$ matrix $\left(\begin{array}{c} Y \\ \hline \end{array}\right)$, with Y as the top $n \times n$ matrix whose columns are orthonormal. This gives us the first n columns of a unitary matrix. Clearly, if we have this, we can just complete the matrix by adding in some more orthonormal columns to get a full basis.

We know that $||Y|| = 1$. Therefore, $\mathbb{I} - Y^\dagger Y$ is a positive semidefinite matrix, because all the eigenvalues of $Y^\dagger Y$ are between 0 and 1. And any semidefinite matrix can be written in the form of $Z^\dagger Z$ for some other matrix Z (this is called the Cholesky Decomposition).

So if we form a new matrix $W = \begin{pmatrix} Y \\ Z \end{pmatrix}$, then we have

$$W^\dagger W = Y^\dagger Y + Z^\dagger Z = \mathbb{I}.$$

This is equivalent to saying that all the columns of W are orthonormal, as desired. □

We conclude that, for any matrix Y with $||Y|| = 1$, we can run a linear-optical experiment, where the probability of some particular outcome is equal to $|\operatorname{Per}(Y)|^2$:

$$U = \left(\begin{array}{c|} Y & \\ \hline & \end{array}\right) \Rightarrow |\langle I|\varphi(U)|T\rangle|^2 = |\operatorname{Per}(Y)|^2, \tag{11.11}$$

where $|I\rangle = |1, \ldots, 1, 0, \ldots, 0\rangle$ is the standard initial state.

11.6.2 Glynn's formula

But now we come to the second difficulty: this is not the same thing as being able to explicitly calculate $|\operatorname{Per}(Y)|^2$. In particular, the permanent, and hence the probability, might be exponentially small. We know that we could get an estimate of $|\operatorname{Per}(Y)|^2$ by running the experiment multiple times and calculating the mean of all the measurement outcomes. But in order to get any nontrivial

estimate, we might need to repeat the experiment an exponential number of times, in which case we might as well just calculate the permanent using a classical computer.

Formally, by repeating the linear optics experiment (sending the bosons through a network of beamsplitters) $O(\frac{1}{\varepsilon^2})$ times, with high probability, we can approximate $|\operatorname{Per}(Y)|^2$ to additive error $\pm\varepsilon$.

It now behooves us to ask: if we have a matrix Y, and its norm is at most 1, then is it hard to estimate its permanent up to small additive error using a classical computer? Alas, it turns out that it is not hard.

Theorem 11. *[36] Let $Y \in \mathbb{C}^{n\times n}$ with $||Y|| \leq 1$. Then there's a classical randomized algorithm that approximates $|\operatorname{Per}(Y)|$ to additive error $\pm\varepsilon$ in $O(\frac{n^2}{\varepsilon^2})$ time.*

To prove this, we're going to use a famous formula for the $n \times n$ permanent called Ryser's formula, or rather, a slight variant called Glynn's formula.

Lemma 12 (Glynn's formula)**.** *For any $n \times n$ matrix Y,*

$$\operatorname{Per}(Y) = \mathop{\mathbb{E}}_{z_1,\ldots,z_n\in\{1,-1\}}[z_1 \cdots z_n \prod_{i=1}^{n}(y_{i1}z_1 + \ldots + y_{in}z_n)] \tag{11.12}$$

where $z_1, \ldots, z_n$ are either 1 or -1 with equal probability. This is true because, when we expand the product, all terms where a z_i is raised of the power of 2 or higher cancel out (because of the $z_1 \cdots z_n$), and the remaining $n!$ terms give exactly the permanent.

Glynn's formula yields the fastest known algorithm to calculate the permanent exactly. It takes time $O(2^n n^2)$, which can be reduced to $O(2^n n)$ if we iterate over all of the possible of $z_1, \ldots, z_n$ in Gray code order.

Now we can get a second cool consequence from Glynn's formula:

Claim 13 (General inequality for the permanent)**.** *For all $Y \in \mathbb{C}^{n\times n}$*

$$|\operatorname{Per}(Y)| \leq ||Y||^n \tag{11.13}$$

In particular this means that the permanent of any unitary matrix is at most 1: $|\operatorname{Per}(U)| \leq 1$.

Proof. We can prove this inequality by using Glynn's formula. But before we do that, we have to introduce the AGM inequality:

Lemma 14 (Arithmetic and geometric means (AGM) inequality)**.**

$$(a_1, \ldots, a_n)^{1/n} \leq \frac{a_1 + \ldots + a_n}{n}. \tag{11.14}$$

Using Glynn's formula (11.12) and the AGM inequality (11.14), we get:

$$\sqrt{\prod_{i=1}^{n}|y_{i1}z_1 + \ldots + y_{in}z_n|^2} \leq \left(\frac{\sqrt{\sum_{i=1}^{n}|y_{i1}z_1 + \ldots + y_{in}z_n|^2}}{\sqrt{n}}\right)^n \tag{11.15}$$

$$= \left(\frac{||Y\vec{z}||}{||\vec{z}||}\right)^n \tag{11.16}$$

$$\leq ||Y||^n \tag{11.17}$$

□

Now we can get an efficient classical algorithm to estimate the permanent of Y. The algorithm consists of picking, for example, $\frac{1}{\varepsilon^2}$ vectors $Z = (z_i, \ldots, z_n)$ uniformly at random, and then taking the empirical mean of the term inside the expectation in equation (11.12). We can use that to approximate $\mathrm{Per}(Y)$ to additive error $\pm\varepsilon$ in only $O(\frac{n^2}{\varepsilon})$ time.

11.7 BosonSampling

So, have we shown that beamsplitter networks yield no advantage over classical computers after all? This is where BosonSampling comes into the picture. We can revisit the experiment and observe that it samples from *some* probability distribution over exponentially many possible outcomes. We know that this distribution somehow "implicitly knows" about permanents. So let's suppose that our problem is to sample from this permanent-based probability distribution; then maybe a quantum computer can give an advantage. We will talk more about this in the next lecture!

Lecture 12: Complexity of BosonSampling

12.1 Review

Recall that given n particles and m modes: If $|S\rangle$ is a list of occupation numbers $|s_1, s_2, \ldots, s_m\rangle$, $\sum_{i=1}^m s_i = n$ and $|T\rangle$ is also a list of occupation numbers $|t_1, t_2, \ldots, t_m\rangle$, $\sum_{i=1}^m t_i = n$, for any unitary U we can create a $\phi(U)$ with the following properties:

1. $\varphi(U)$ is unitary
2. $\varphi(U)$ is a homomorphism (i.e. $\varphi(UV) = \varphi(U)\varphi(V)$)

However, what if the matrix we care about isn't unitary?

Lemma 15. *For any $X \in \mathbb{C}^{n\times n}$, it is possible to embed $\frac{X}{\|X\|}$ as a submatrix of a $2n \times 2n$ unitary U.*

In addition, it would be possible to implement this $2n \times 2n$ unitary with only $\mathcal{O}(n^2)$ beamsplitters. The probability of us finding the output state in the standard initial state $|I\rangle = |1, 1, \ldots, 1, 0, 0, \ldots, 0\rangle$ is equal to

$$\left(\frac{|\operatorname{Per}(X)|}{\|X\|^n}\right)^2$$

Does this mean that we can directly calculate the permanent of any arbitrary matrix X? Unfortunately not.

It is not possible to directly observe probabilities/amplitudes! We can only sample from the distribution, but each measurement will give only a single outcome. In other words, Nature only needs to know *implicitly* about the solutions to #P-complete problems.

For reasons having to do with standard statistics, we can approximate $|\operatorname{Per}(X)|$ to $\pm\varepsilon\|X\|^n$ precision with $\frac{1}{\varepsilon^2}$ measurements. However, Gurvits's algorithm can do just as well classically. So there seems to be no hope in getting an exponential speedup.

12.2 BosonSampling

BosonSampling can be thought of as an instance where the researchers shoot the arrow first and then draw the target around it!

Given a column-orthonormal matrix $A \in \mathbb{C}^{m \times n}$, let $\mathcal{D}_A$ be the following distribution defined by A:

$$\Pr_{\mathcal{D}_A}[S] = \frac{|\operatorname{Per}(A_S)|^2}{s_1! s_2! \dots s_m!}$$

where A_S is the $n \times n$ matrix formed from A by taking s_i copies of its ith row for all i. Note that $\mathcal{D}_A$ has an exponentially large support. We want to sample from $\mathcal{D}_A$.

This problem is easy to solve using a beamsplitter network, as we discussed previously. What is the computational complexity of BosonSampling? Is BosonSampling in BQP? It essentially is, though it is important to make the distinction that BosonSampling is a *sampling* problem and not a *decision* problem. Nevertheless, it would be fair to say BosonSampling $\in$ SampBQP, where SampBQP is the class of sampling problems solvable in quantum polynomial time.

How would one simulate a beamsplitter network using qubits? First, we can see that it is possible to represent the state as

$$\sum_S \alpha_S |s_1, s_2, \dots, s_m\rangle$$

with $\mathcal{O}(m \log(n))$ qubits. Then, each beamsplitter can be naively implemented as an $n^2 \times n^2$ unitary, which in turn can be decomposed into 1- and 2-qubit gates using the Solovay-Kitaev Theorem. Thus, it is easy to simulate BosonSampling with a quantum computer.

Is BosonSampling complete for SampBQP? It is not known, though believed to be extremely unlikely, because the group of beamsplitter unitaries is exponentially smaller than the group of general n-qubit unitaries.

Is there a polytime randomized algorithm for BosonSampling? Probably not! Indeed, if there were to be one, we can break a lot of things.

We can prove this by assuming the opposite and seeing what happens. Assume there exists a fast classical algorithm M, that given $X \in \mathbb{C}^{n \times n}$, $\|X\| \leq 1$ as well as a classical random string r satisfies:

$$\Pr_r[M(X, r) \text{ accepts}] = |\operatorname{Per}(X)|^2$$

Let's call the probability p for convenience. Then p is the probability that M accepts, summed over all r, but also the squared magnitude of $\operatorname{Per}(X)$, which is a sum of $n!$ terms. While these are both sums of exponentially many terms, they're also extremely different. The first is a sum of *nonnegative* terms, while the second is a sum of *complex* terms, which can interfere destructively and cancel, leaving only a tiny residue. As we'll see, if the latter type of sum could always be reduced to the former, there would be dramatic consequences.

12.3 The argument

We'll now summarize the argument for the classical hardness of BosonSampling, from the paper by Aaronson and Arkhipov [7]. The argument proceeds by combining four facts, all of them standard in classical complexity theory:

1. p can be approximated to a multiplicative error in the class $\mathsf{BPP}^{\mathsf{NP}}$ – that is, probabilistic polynomial time with an NP oracle. The high level idea of how to do it goes like this:

Pick a hash function h. Then, ask the oracle "Does there exist an r such that $M(X,r)$ accepts and $h(r)=0$?" Then, keep changing the range of the hash function until the answer is equally likely to be "yes" or "no", to get an estimate of how many r's cause $M(X,r)$ to accept and therefore of the probability.

2. It is #P-hard even to *approximate* $|\operatorname{Per}(X)|^2$ to some multiplicative error.

Proof. We already know that exactly computing the permanent is #P-complete. We need to show how to compute $\mathsf{Per}(X)$ exactly, given an oracle that multiplicatively approximates permanents. Let

$$X = \begin{bmatrix} x_{11} & \\ & [Y] \end{bmatrix}$$

$$X^{[r]} = \begin{bmatrix} x_{11}-r & \\ & [Y] \end{bmatrix}$$

Here, X is an $n \times n$ matrix, and Y is the bottom-right $(n-1)\times(n-1)$ submatrix of X. $X^{[r]}$ is X but with a number r that is subtracted from the top left entry. By the definition of the permanent, we see that

$$\operatorname{Per}(X^{[r]}) = \operatorname{Per}(X) - r\operatorname{Per}(Y)$$

Then, binary search over all r's until one is found such that

$$\begin{aligned} 0 &= \operatorname{Per}(X) - r\operatorname{Per}(Y) && \text{(to exponential precision)} \\ \implies \operatorname{Per}(X) &= r\operatorname{Per}(Y) && \text{(to exponential precision)} \end{aligned}$$

Note that we've reduced the problem to calculating $\operatorname{Per}(Y)$, where Y is an $(n-1)\times(n-1)$ matrix. So now we can recurse. Ultimately, we will end up with a solution that looks like:

$$\operatorname{Per}(X) = r_1 r_2 \ldots r_n$$

□

3. $\mathsf{BPP}^{\mathsf{NP}} \subseteq \mathsf{PH}$, where

$$\mathsf{PH} = \mathsf{P} \cup \mathsf{NP} \cup \mathsf{NP}^{\mathsf{NP}} \cup \ldots$$

is the Polynomial Hierarchy, which consists of P, NP, and any constant-sized stacks of NP oracles. It is *conjectured* that the Polynomial Hierarchy is infinite, i.e. each level is more powerful than the previous. If PH were to collapse at some finite level, it would be almost as surprising as if $\mathsf{P}=\mathsf{NP}$. Sipser, Gács, and Lautemann showed in 1983 that BPP is within the Polynomial Hierarchy (see e.g. [65]):

$$\mathsf{BPP} \subseteq \mathsf{NP}^{\mathsf{NP}}$$

Their proof generalizes to show that

$$\mathsf{BPP}^{\mathsf{NP}} \subseteq \mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}}}$$

4. Finally, we have Toda's Theorem from 1991, which states that $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$ [68].

Putting all four facts together, we can see that:

If there is a fast randomized algorithm for BosonSampling

$\implies$ #P-hard approximation problems are in $\mathsf{BPP}^{\mathsf{NP}}$

$\implies$ $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{BPP}^{\mathsf{NP}}$

$\implies$ $\mathsf{PH} = \mathsf{BPP}^{\mathsf{NP}}$

$\implies$ PH collapses to $\mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}}}$

And this would be a huge deal, because that would mean the Polynomial Hierarchy is finite! This is why we consider it improbable that there is a fast randomized algorithm for BosonSampling—at least in the exact setting. We'll discuss *approximate* BosonSampling in the next lecture.

Lecture 13: Complexity of Approximate Boson-Sampling

13.1 Complexity of Exact BosonSampling and its Limitation

In the last lecture we gave the first theoretical evidence for BosonSampling being hard for classical algorithms. Namely, we showed that if there was an efficient classical algorithm for *exact* BosonSampling, then the polynomial hierarchy would collapse. The proof followed by combining the following two facts.

Fact 1. It's #P-hard to *multiplicatively* approximate the value of $|\mathrm{Per}(X)|^2$.

Fact 2. If there was an efficient classical algorithm for *exact* BosonSampling, then $|\mathrm{Per}(X)|^2$ would be approximable in $\mathsf{BPP}^{\mathsf{NP}}$.

13.1.1 Permanent and Counting

It's worth mentioning here that it's #P-hard to compute the *exact* value of the permanent even for 0-1 matrices, as shown in the original paper by Valiant [69]. A famous corollary is that counting the number of perfect matchings in a bipartite graph is #P-hard.

By contrast, we have learned a polynomial-time algorithm for *finding* a perfect matching in undergraduate algorithms class. So we see that search can be totally different from counting. Searching for a solution can be in P even while counting the number of solutions is #P-hard, no easier than counting the satisfying assignments to a 3-SAT formula.

Interestingly, the converse is still an open problem: if searching for a solution is NP-hard, is counting the number of solutions necessarily #P-hard?

13.1.2 Approximation of Permanent

The situation is very different for approximation. A seminal discovery by Jerrum, Sinclair and Vigoda [42] showed a polynomial-time randomized algorithm to *approximate* the permanent of any **non-negative** matrix to any fixed constant multiplicative error. The algorithm is based on Markov Chain Monte Carlo (MCMC), which is a very successful and important technique in both theory and practice. It takes a random walk on the set of all possible perfect matchings; the major technical part is proving the random walk mixes in a polynomial number of steps. Using

standard techniques, one can reduce the problem of approximate counting to that of sampling a nearly-random solution.

To summarize, with regards to the Permanent:

	Appproximating	Exactly Computing
Non-negative matrices	P	#P-hard
General Matrices	#P-hard	#P-hard

The intuition behind this contrast between the two approximation problems is that the presence of negative elements may cause most of the contributions in the sum to get cancelled out. This, of course, is exactly the behavior of amplitudes in quantum mechanics.

13.1.3 Limitations of Exact BosonSampling

Our hardness results for exact BosonSampling seem solid, as complexity theorists strongly believe the polynomial hierarchy doesn't collapse. However, when we want to push BosonSampling experiments to the real world, many new issues arise.

Maybe the most obvious issue is that our previous hardness result only relies on the probability of a *single* outcome (among exponentially many outcomes), which in general is exponentially small. Therefore, any tiny error in the quantum device may totally destroy that probability and our argument.

As our quantum experiment is subjected to noise, it is only fair if we allow the classical simulation to be subjected to noise also. Eventually, we want to know how hard it is to classically simulate whatever it is that the BosonSampling experiment does.

Therefore, in this lecture we will study the complexity of *approximate* BosonSampling. To this end, we have to introduce several new assumptions that are not rock-solid. However, a decade since the original paper [7] was published, those conjectures are still standing.

13.2 Complexity of Approximate BosonSampling

In the first place, what makes bosons attractive for quantum supremacy experiments is that we know the permanent is not just #P-hard in the worst case; the hardness of it has some *robustness* properties, e.g., being #P-hard in average.

Denote the true distribution in BosonSampling by D. Now suppose we have a classical algorithm $\mathcal{A}$ that samples from a distribution $D_{\mathcal{A}}$ with $\|D - D_{\mathcal{A}}\|_{\text{TV}} \leq \epsilon$, where $\|\cdot\|_{\text{TV}}$ is the *total variation* distance[1], a standard metric measuring the distance between two distributions. Our goal will be to show that such an algorithm would already yield unlikely complexity consequences. Aaronson and Arkhipov [7] managed to do this with the help of the following conjecture.

[1]Formally, for any two distributions D_1, D_2, $\|D_1 - D_2\|_{\text{TV}} := \frac{1}{2}\sum_x |p_x - q_x|$, where p_x, q_x are the probability of sampling x from distributions D_1, D_2 respectively.

Conjecture 16 (Permanent of Gaussians Conjecture (PGC))**.** *Suppose matrix X is randomly drawn from $N(0,1)_{\mathbb{C}}^{n\times n}$, i.e, each element is drawn from $N(0,1)_{\mathbb{C}}$ independently. Then, it is $\#\mathsf{P}$-hard to approximate $|\mathrm{Per}(X)|^2$ for at least $1-\delta$ fraction of X with additive error $\epsilon\sqrt{n!}$ in time* $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$.

PGC is also interesting in classical complexity theory, besides its motivation from BosonSampling. [7] showed that assuming PGC, any fast classical algorithm for approximate BosonSampling would imply the collapse of polynomial hierarchy.

We now give some intuition for this result. Assume we have m modes and n photons in the state $|1,\ldots,1,0,\ldots,0\rangle$. The true distribution D is determined by an $m\times n$ rectangular matrix A, where the amplitude of each outcome is defined by the permanent of an $n\times n$ sub-matrix of A (possibly with repeated rows).

We first present some basic facts from random matrix theory in Section 13.2.1. Then, we show that it's possible to "hide" an $n\times n$ Gaussian random sub-matrix X in the rectangular matrix A without being noticed.

13.2.1 Facts about Random Matrices

Suppose we want to pick an $m\times m$ unitary matrix uniformly at random. But what do we mean by a *uniform* unitary? There is a notion called *Haar measure*. Informally, here is a natural way to generate a Haar-random matrix:

1. choose a uniformly random unit vector in $\mathbb{C}^m$ as the first row;
2. for each succeeding row, choose a uniformly random unit vector in the subspace of $\mathbb{C}^m$ that's orthogonal to all the rows above it.

An important property of Haar-random matrices (or indeed, an equivalent way to define them) is that an infinite sequence of random beam-splitters will eventually converge to the Haar measure.

Now consider a single element in an $m\times m$ Haar random matrix U. Its distribution measure will look like a complex Gaussian of mean zero and variance of $1/n$ with the tails cut off. In general, any sufficiently small sub-matrices also exhibit this property.

Lemma 17 (Informal)**.** *If $n\ll\sqrt{m}$, any $n\times n$ sub-matrix of an $m\times m$ Haar random matrix U is statistically close to a matrix of independent complex Gaussians of mean 0 and variance $1/n$.*

Intuitively, when n gets large compared to $\sqrt{m}$, we will start noticing the unitarity constraint. Also, it's not a coincidence that this is the same place we start noticing the bosonic birthday paradox. As long as we keep $n\ll\sqrt{m}$, with high probability our output state doesn't have any collisions, so the distribution is dominated by the states that correspond to $n\times n$ sub-matrices of U.

13.2.2 Hiding Lemma

Suppose we are given a matrix X randomly drawn from $N(0,1)_{\mathbb{C}}^{n\times n}$, and we want to calculate its permanent. We cannot simply embed X as a specific sub-matrix of A (say, the topmost one), because the corresponding probability could be *adversarially* corrupted.

Lemma 17 suggests a clever way to overcome this. Namely, we can embed X as a *randomly chosen* sub-matrix of A. By Lemma 17, X is statistically indistinguishable from other sub-matrices of A, as they all have to look like independent Gaussians in each element. Therefore, with high probability, an approximate BosonSampling algorithm will sample the outcome corresponding to X with probability proportional to $|\mathrm{Per}(X)|^2$. (We omit the details needed to make this precise.)

Finally, applying the same approximate counting argument from the last lecture, any efficient classical algorithm for approximate BosonSampling could be turned into a $\mathsf{BPP}^{\mathsf{NP}}$ algorithm for approximating the permanent of most the Gaussian random matrices, which implies the collapse of polynomial hierarchy if we assume the PGC (i.e., that approximating Gaussian permanents is #P-hard).

13.3 Status of PGC

The entire analysis for approximate BosonSampling hinges on the correctness of the PGC. We now introduce more about the status of the PGC.

13.3.1 Calculate Permanent Exactly in Average-Case

We proved in the last lecture that it's #P-hard to approximate $\mathrm{Per}(X)$ for am arbitrary matrix X. Actually, [7] were also able to prove a converse result.

Theorem 18. *It is #P-hard to* exactly *compute* $\mathrm{Per}(X)$ *for most matrices* $X \sim N(0,1)_{\mathbb{C}}^{n\times n}$.

Theorem 18 is inspired by the result of Lipton [47] for the permanent over a finite field.

Theorem 19 (Informal, [47])**.** *If $p \gg n$ is a prime number, then any efficient algorithm $\mathcal{A}$ which calculates* $\mathrm{Per}(X)$ *for most matrices $X \in \mathbb{F}_p^{n\times n}$ can be transformed into an efficient algorithm $\mathcal{A}'$ to calculate* $\mathrm{Per}(X)$ *for all matrices in $\mathbb{F}_p^{n\times n}$.*

For at least one #P-complete problem, namely the permanent, we know that the average-case is as hard as the worst-case. We have the same situation for *discrete log* problem, which is believed to be an NP-intermediate problem. However, it's still open whether any NP-complete problem is NP-complete in average-case.

Note that when $p = 2$, the permanent is always equal to the determinant, and thus is easy; for any prime $p \geq 3$, the permanent is known to be complete for the class $\mathsf{Mod_pP}$, which is the mod-p version of #P.

Proof of Theorem 19. Suppose we want to calculate the permanent of a matrix $A \in \mathbb{F}_p^{n\times n}$. We first randomly pick a matrix $B \in \mathbb{F}_p^{n\times n}$. Now consider the function $q(t) = \mathrm{Per}(A + tB)$, where $t \in \mathbb{F}_p$ is a scalar.

By construction, $q(t)$ is a polynomial in t of degree at most n. When $t = 0$, we have $q(t) = \mathrm{Per}(A)$; when $t \neq 0$, $A + tB$ is a uniform random matrix in $\mathbb{F}_p^{n\times n}$. Therefore, we can use the algorithm $\mathcal{A}$ to calculate $q(1), q(2), \ldots, q(n)$, and then interpolate the polynomial q to get $q(0) = \mathrm{Per}(A)$. □

Theorem 18 is essentially proved analogously to Theorem 19, but with the complex Gaussian measure instead of the uniform distribution over a finite field.

However, for the PGC, we somehow have to combine "approximation" and "average-case" in the same reduction. For example, if we try to use the same polynomial interpolation trick for Theorem 19, then a small error for estimating $q(1), q(2), \ldots, q(n)$ will blow up to a very large error for $q(0)$.

13.3.2 Permanent Anti-Concentration Conjecture

A kind of necessary but not sufficient step for proving the PGC is understanding the *distribution* of the permanent of a Gaussian matrix. A somewhat stupid example is that, if the permanent of a Gaussian matrix was almost always zero, then PGC would be false because an algorithm could simply output 0 all the time.

It's not hard to show that

$$\mathop{\mathbb{E}}_{X \sim N(0,1)_{\mathbb{C}}^{n \times n}} \left[|\mathrm{Per}(X)|^2 \right] = n!.$$

[7] also calculated the fourth moment, but higher moments are harder to calculate.

Conjecture 20 (Permanent Anti-Concentration Conjecture, PACC)**.** *There exists a polynomial p such that for all n and $\delta > 0$,*

$$\mathop{\mathbb{P}}_{X \sim N(0,1)_{\mathbb{C}}^{n \times n}} \left[|\mathrm{Per}(X)| < \frac{\sqrt{n!}}{p(n, 1/\delta)} \right] \leq \delta.$$

Conjecture 20 plays a key role in the proof framework of [7]. If we rewrite the PGC in a way that claims the multiplicative hardness of approximating $|\mathrm{Per}(X)|$, instead of additive hardness, then we need Conjecture 20 to build the equivalence between these two slightly different statements.

Conjecture 20 has drawn attention from mathematicians, as a problem in random matrix theory. Recent progress by Nezami [52] provides strong estimates for all (integer) moments of the permanent of random Gaussian matrices, using advanced math tools like representation theory. Nezami also argues that in order to pin down the distribution of the permanent and prove Conjecture 20, additional information than integer moments may be needed.

13.4 Obstacle in Real-World BosonSampling Experiment

So far, we've brought the theory a little bit closer to experiment by allowing a small error in total variation distance from the ideal distribution. However, this is still way better than the actual situation in experiments!

The first major obstacle looks trivial in theory, but is huge in practice. Namely, experimentalists cannot currently generate the state $|1, \ldots, 1, 0, \ldots, 0\rangle$ in the start of the BosonSampling experiment. They don't yet have a deterministic single-photon source!

Coherent State A first try might be taking a laser, and then attenuating it until it emits only one photon at a time. However this gives us something called a *coherent state* instead of what we want. A coherent state with parameter α is essentially a superposition of different numbers of photons according to a *Poisson distribution*:

$$|\alpha\rangle \coloneqq e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle .$$

If we use coherent states for a BosonSampling experiment, we'll fail to achieve quantum supremacy in a somewhat dramatic way. Suppose we feed coherent state $|\alpha_i\rangle$ to the i-th mode. Denote the vector $(\alpha_1, \ldots, \alpha_m)$ by $\vec{\alpha}$. Let U be the unitary corresponding to the BosonSampling circuit. Then what we get at mode i is just $|(U\vec{\alpha})_i\rangle$, as if we had just a single photon in a superposition of the modes. We could verify this fact by straightforward calculation using the permanent formula we have learned. Thus, standard laser light is easy to simulate classically, and we need some more exotic way to generate photons. See Figure 13.1 for an illustration.

It feels like Nature conspired to make optical quantum computing very difficult!

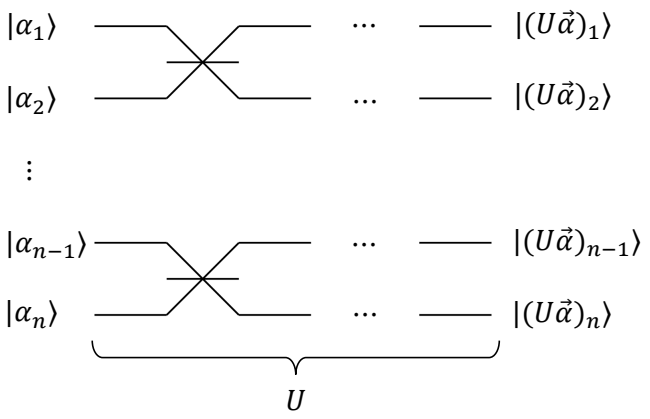


Figure 13.1: If we use coherent states generated by a laser for a BosonSampling experiment, the outcome is easy to simulate classically.

Squeezed State A squeezed state is like a coherent state version of an EPR pair (on two modes):

$$\sqrt{1 - |\lambda|^2} \sum_{n=0}^{\infty} \lambda^n |n\rangle |n\rangle ,$$

where $|\lambda| < 1$.

Squeezed states can be generated by a technique called *spontaneous parametric down conversion* (SPDC): first, use a laser to stimulate a crystal, which makes the electrons in the crystal jump into an excited state. Then, the electrons fall back to the ground state and emit photons traveling in opposite directions in the state above.

When λ is small enough, we can approximate the squeezed state by $|0\rangle |0\rangle + \lambda |1\rangle |1\rangle$. We can then measure one of the modes. If it has a photon in it, then we know there is also a photon in the other mode, and can use that photon in a BosonSampling experiment.

The problem with this method is that if we want to generate n photons, then we have to wait for exponential time in n until we luckily have n parallel SPDC sources each generate one photon simultaneously. In the next lecture, we will explain how people got around this new problem, and continue on other issues that arise in BosonSampling experiments.

Lecture 14: BosonSampling Experiments

14.1 Overview

In the last lecture we discussed the problem of Approximate BosonSampling (Approx-BS) and showed that if a fast classical algorithm exists, then the polynomial hierarchy collapses and $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$. This hasn't been proven false, and is perhaps slightly likelier than $\mathsf{P} = \mathsf{NP}$, but nonetheless is considered very unlikely. In this lecture we discuss practical challenges to BosonSampling and how experimentalists have tried to work around them.

14.2 Practical Approaches to BosonSampling

Recall from previous lectures the general picture of BosonSampling: we have n particles, m possible modes, and an $m \times n$ column-orthonormal matrix U which represents a beamsplitter network. We know that the probability of observing some output state $S = |s_1, s_2, \ldots, s_n\rangle$ starting from the initial state $I = |1, 1, \ldots, 0, 0\rangle$ is given by

$$\langle I| \varphi(U) |S\rangle = \frac{|\mathrm{Per(X)}|^2}{s_1! s_2! \ldots s_n!}$$

where X is a submatrix of U corresponding to the particular outcome S. The goal of BosonSampling is simply to efficiently sample from this distribution, something that should be computationally hard for classical computers.

14.2.1 Coherent States

One key challenge in actually building a BosonSampling experiment is reliably generating a single particle at a time to construct the initial state. One seemingly attractive option is to use a laser and just turn it on for a very short period of time, but it turns out that what lasers generate are called **coherent states**. These can be expressed in the form:

$$|\alpha\rangle = e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle$$

where α is any complex number and $|n\rangle$ represents a state of n photons. Intuitively this can be thought of as a Poisson distribution over the number of photons, plus some extra phase information. As an aside, we can also build these states using **creation operators** $a^\dagger$, which are defined as linear

transformations that satisfy $a^\dagger |n\rangle = \sqrt{n+1}\,|n+1\rangle$ for all n, and which we implicitly introduced before as the formed variables $x_1, \ldots, x_m$ in Fock polynomials. Then we can write:

$$\begin{aligned} |\alpha\rangle &= e^{-|\alpha|^2/2} e^{\alpha a^\dagger} |0\rangle \\ &= e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{(\alpha a^\dagger)^n}{n!} |0\rangle \end{aligned}$$

The problem with coherent states is that they allow for beamsplitter networks to be efficiently simulated with classical computation. If we have some input coherent states $|\alpha_1\rangle, |\alpha_2\rangle, \ldots, |\alpha_n\rangle$ and a unitary transformation U, then the output states are simply $|(U\vec{\alpha})_1\rangle, |(U\vec{\alpha})_2\rangle, \ldots, |(U\vec{\alpha})_n\rangle$, where $U\vec{\alpha}$ is the vector obtained by applying U to $(\alpha_1, \ldots, \alpha_n)$ and $(U\vec{\alpha})_i$ is the ith entry. But this is just matrix multiplication, which can be efficiently computed classically. **Thus, we cannot use coherent states as inputs for BosonSampling.**

14.2.2 Spontaneous Parametric Down Conversion

Another attractive option for generating particles is called **spontaneous parametric down conversion (SPDC).** The general idea is to use a crystal to split one photon beam into two entangled modes - for our purposes we can think of the output state as of the following form, where $|\lambda| < 1$ is a parameter of the system:

$$|\psi_\lambda\rangle = \sqrt{1-|\lambda|^2} \sum_{n=0}^{\infty} \lambda^n |n\rangle_1 |n\rangle_2$$

Alternatively, we can express these states in terms of creation operators as well:

$$|\psi_\lambda\rangle = \sqrt{1-|\lambda|^2} e^{\lambda a_1^\dagger a_2^\dagger} |0\rangle_1 |0\rangle_2$$

We call any state in the form $e^{p(a_1^\dagger, \ldots, a_m^\dagger)} |0\rangle_1 \ldots |0\rangle_m$, where p is a polynomial of degree at most 2, a **Gaussian state** - note that both coherent states and the states produced by SPDC fall into this definition.

The advantage of using SPDC over coherent states is that now we have a verifiable way of generating a single photon. We can simply measure one of the output modes and condition on seeing one photon - if we do, then we know with certainty that the other, unmeasured mode has one photon also (we say that the presence of a single photon in the second mode is "heralded"). Unfortunately, getting this outcome is nondeterministic - and if we let the probability of seeing one photon in an SPDC device be p, then the probability of seeing exactly one photon in each of n SPDC devices run in parallel is only p^n. Since we can only begin our experiment once we get a photon in each of the input modes, it might take time proportional to $\frac{1}{p^n}$ to achieve that. So, even though we can generate initial states for BosonSampling, and those states are heralded, it takes exponential time for the procedure to succeed! This makes it impossible to show quantum supremacy because classical algorithms can already perform BosonSampling in exponential time.

14.2.3 Scattershot BosonSampling

One proposal by Lund et al. [49] - and named by Prof. Aaronson - is called **Scattershot BosonSampling.** The basic idea is to use a large number of SPDC devices – say m of them – and choose

the parameter λ to be small so that each produces a state roughly equivalent to $|0\rangle|0\rangle + \lambda|1\rangle|1\rangle$. As before, we measure one mode of each such state and hope to see one photon, which heralds that the other mode also has exactly one photon. Only a roughly $|\lambda|^2$ fraction of the output modes will have a photon, but as long as m is large enough, $|\lambda|^2 m$ photons will still be sufficient! We won't be starting from the standard initial state, but one can argue that we're still sampling from **some** classically hard distribution under the same assumptions made in the analysis of ordinary BosonSampling. The key practical problem is simply the large number of modes required.

14.2.4 Gaussian BosonSampling

A generalization of Scattershot BosonSampling was proposed by Hamilton et al. [38] who called it **Gaussian BosonSampling**. Instead of measuring half of the modes to look for single photons, you just apply a beamsplitter network to all of them. The key observation is that if the SPDC array generates Gaussian states, then the final amplitudes depend on another #P-hard problem - the Hafnian of a matrix. Unfortunately, no one has formally related the hardness of approximating random Hafnians to that of approximating random permanents, but if you believe that the permanent is hard then it's not too much of a stretch to believe the Hafnian is too. So, there's at least some chance that your experiment can be efficiently simulated on a classical computer - but on the other hand you're guaranteed twice as many single-photon modes as in Scattershot BosonSampling.

14.3 BosonSampling Experiments

A research group at USTC, a university in China, has used Gaussian BosonSampling to actually perform experiments, although with significant limitations to being able to claim quantum supremacy. A few years ago they were able to give results for 14 photons, then the next year they improved their setup to ≈50 detected photons, and more recently their experiments have 100+ detected photons [75].

14.3.1 Experimental Compromises

As we mentioned before, the USTC experiments made several significant compromises that affect potential supremacy claims. The first of these is the photon loss rate - around **70%** of the input photons get lost in the beamsplitter network and aren't picked up by the detectors. Aaronson and Brod [8] showed that losing up to a constant number of photons retains the hardness of the original BosonSampling problem, but losing k requires $\frac{1}{n^k}$ precision where n is the number of photons. Since the experiment has $k = O(n)$ it begs the question of whether the distribution is still classically hard to sample from.

Another significant drawback is that the number of experimental modes is smaller than desired. Ideally we want m to be quadratic with respect to n so that the bosonic birthday paradox applies and it becomes unlikely for modes to end up with multiple bosons - the actual experiments use $m \approx n$. With lots of collisions it becomes easier for a classical computer to sample by computing permanents because the matrices have lots of repeated rows.

This is compounded by the fact that USTC's experiments use non-number-resolving detectors.

Essentially, their sensors only output whether a mode is occupied, not the actual number of bosons present in it. This wouldn't matter with enough modes because collisions would be very uncommon, but since the experiments don't have this it potentially affects the sampled distribution because lots of output states would appear the same to the detectors (e.g. $|n, 0\rangle$ and $|m, 0\rangle$ for $n, m > 0$).

14.3.2 Linear Cross-Entropy

Finally, a serious issue with any sampling-based quantum supremacy experiment, not just BosonSampling, is that the sampled results need to be verified against the actual probability distribution, which requires a classical computer to compute the theoretical probabilities directly. So ironically, in order to demonstrate that quantum computers can solve the classically hard problem of BosonSampling, you need a classical computer to solve the problem first to verify the results!

Google's proposal for doing this was simply to take the measured output states from the experiment - call them $|S_1\rangle, |S_2\rangle, \ldots, |S_k\rangle$ - and compute the following function:

$$\text{LXEB} = \frac{1}{k} \sum_{i=1}^{k} |\langle I| \varphi(U) |S_i\rangle|^2$$

This is called the **Linear Cross-Entropy (LXEB)** score. The idea is that classical computers cannot efficiently spoof it, meaning that they can't cheat and get a high score without actually solving the underlying problem. But this, too, remains an open question.

Google's quantum supremacy experiments - which don't rely on BosonSampling but a different classically hard sampling problem - were specifically designed so that a trivial classical algorithm would get an LXEB of 1 and an ideal quantum computer would get an LXEB of 2. The experimental result? 1.002 - but with a very high confidence of being greater than 1. So, the amount of noise is much more than we'd like, but the ball was still in the skeptics' court to answer the claim of quantum supremacy.

Lecture 15: Warm-up to the KLM Scheme

15.1 Overview

15.1.1 Last Lecture

In the last lecture, we talked about experimental implementations of BosonSampling [7], and in particular the difficulty of achieving quantum supremacy using current experimental techniques.

Experimental Issues with BosonSampling

- Methods of creating input states such as Spontaneous Parametric Down-Conversion (SPDC) produce Gaussian input states instead of the Fock states which the original `BosonSampling` protocol uses. This modifies the probabilities of measurement outcomes to be matrix Hafnians instead of matrix permanents, but the distribution of random matrix Hafnians is still conjectured to be hard to sample from [26].
- Lost photons in experimental setups "average out" the set of matrices that could produce a given output, which reduces hardness if the number of lost photons is great enough. In a 2020 experiment done by the University of Science and Technology of China (henceforth USTC), the proportion of lost photons was about 70% [75].
- Similarly, calibration errors (if they are severe enough) also allow a classical simulation to do better.
- In BosonSampling experiments done by USTC [75], the number of modes was $O(n)$ as opposed to the $\gg n^2$ assumed in the theoretical work.
- The detectors used in the USTC experiments were non-number-resolving, which means that they could only measure the presence or absence of photons in a given mode as opposed to measuring the number of photons in that mode if nonzero.
- Lastly, even if the experiment succeeds and many samples can be drawn from the BosonSampling distribution in a reasonable amount of time, a classical machine will need an exponentially long time to verify that the samples are consistent with the correct distribution. Currently, the Linear Cross-Entropy Benchmark (LXEB) is used, which cost the USTC experiment hundreds of thousands of dollars in supercomputer time for their experiment to verify the results.

A recent paper from Google claims that a fast classical algorithm exists to outperform the recent USTC experiment's LXEB performance mainly by relying on the calibration errors present in their results [73].

On the Bright Side

The USTC experiment had one clear advantage over the kind of supremacy experiment that Google did in 2019 [16] with superconducting qubits: the size of the Hilbert space that the computation takes place in is drastically larger. USTC's experiment using linear optics works within a Hilbert space of dimension $\gg 2^{100}$, whereas Google's has dimension 2^{53}.

This makes a significant difference in terms of classical spoofing abilities: a supercomputer with $> 2^{53}$ classical bits of storage space such as Summit in Oak Ridge National Laboratory is estimated to need only ≈ 3 days to fully simulate Google's circuit and produce the same number of samples. But no existing classical computer has 2^{100} bits of storage.

On this note, a paper by Aaronson and Chen in 2016 [9] describes the space-time tradeoffs of different classical algorithms for quantum simulation:

Note that m is the depth, or number of gates in a circuit while n is the number of qubits.

Method	Time	Space	Description
"Schrodinger Simulation"	$O(m2^n)$	$O(2^n)$	Explicitly calculate each amplitude
"Feynman Simulation"	$O(4^m)$	$O(m+n)$	Calculate each amplitude as a sum over "paths"
[9]	$O(m^n)$	$O(m+n)$	Recursive "best of both worlds" algorithm

By the nature of quantum mechanics, all of these algorithms are extremely parallelizable and so supremacy experiments will always have to be wary of the classical "just throw more cores at the problem" technique for catching up to their results.

Question: Have we so far demonstrated clear quantum supremacy?
Answer: It's up to you to decide how to interpret the above, but certainly there is more to be done!

15.1.2 This Lecture

The focus of this lecture will be on the KLM scheme [43] for achieving universal quantum computing (BQP) experimentally with linear optical networks. Today we will not discuss the full KLM scheme but instead work up to it gradually with a weaker result – namely, the universality of "postselected" linear optics – to try to get a grasp on what's going on.

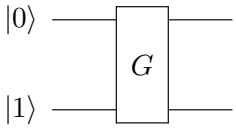
15.2 Getting More Out of Linear Optical Networks

15.2.1 Motivating the KLM Paper

Linear-optical networks, which are circuits composed strictly of beamsplitters and phaseshifters, are not by themselves capable of facilitating universal quantum computation. This intuitively follows

from the fact that each unitary acting on the bosonic state is specified by only m^2 parameters (where m is the number of modes), but the full Hilbert space has dimension $\binom{m+n-1}{n} \gg m^2$. Before giving up, let's see what we can try.

The dual-rail representation of qubits does let us encode arbitrary *one*-qubit gates using linear optics, as follows:



Introducing two-qubit gates is where it gets tricky. We know that the set {CNOT, all one-qubit gates} is universal for BQP, but creating something with the effect of a CNOT gate using just linear optics (deterministically) is known not to be possible [44].

Aside: Why Do We Want Linear Optics So Badly?

Using other quantum mechanical resources such as atoms and electrons, we can achieve universality by way of nonlinear optics. This would involve interacting our bosonic representations of qubits with other fields to inject the nonlinearity and "break out" of the linear optical restrictions.

However, the reason we don't want this is because photons have amazing properties like a long coherence time. Nonlinearity would create **tons** of noise and decoherence in our system which would make building a quantum computer much harder.

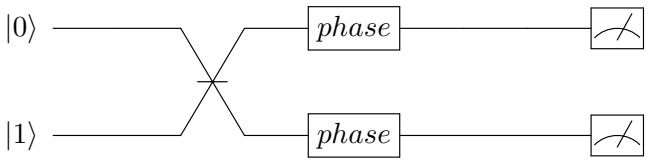
The Punchline

In 2000, Knill, Laflamme, and Milburn (henceforth KLM scheme) found that by adding *adaptive measurements* to linear optical networks, they could simulate CNOT gates and thus achieve BQP-universality [43].

Before jumping into the result of their paper, we will first show a weaker version of their scheme, which relies on *postselected* measurements.

15.2.2 The Dual Rail Representation

To show how we will create a CNOT gate in the universe of linear optics, we first have to revisit the idea of representing qubits in bosonic states with dual rails.



The above diagram represents a linear optical network with two modes, which we call $|0\rangle$ and $|1\rangle$. We restrict these two modes to have exactly one photon between the two of them, so that the above

system at any given time has some state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. This allows us to consider the system as encoding a qubit with exactly that state $\alpha|0\rangle + \beta|1\rangle$.

The object in the middle of the diagram is a beamsplitter with phaseshifters on each output mode, which is a linear optical device able to encode any arbitrary unitary transformation on two modes (Reck et al. [60]). Since there is exactly one photon in the system, this allows it to act as an arbitrary unitary gate on the encoded qubit.

To perform a measurement on the encoded qubit in the computational basis, we simply measure both modes with a photon detector and check which detector found a photon. With this, we are done with one-qubit operations.

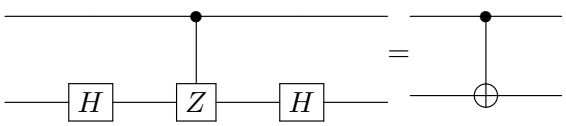
15.2.3 Getting to CNOT

We know we want to introduce some operation into our linear-optical network that will simulate the effect of a CNOT from one dual-rail qubit to another.

One thing to note is that creating a CNOT gate is equivalent to creating a different gate known as controlled Z, or CPHASE.

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

The equivalence follows from the following identity:



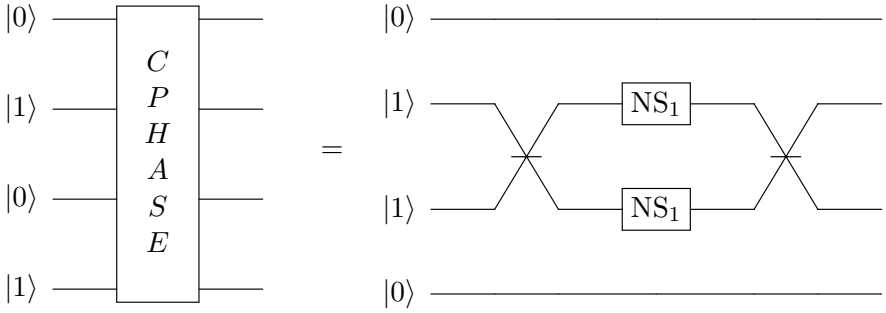
(15.1)

where the CPHASE is represented by a controlled Z gate from the top qubit onto the bottom qubit.

Therefore if we can find a way to apply a CPHASE gate, we have achieved BQP-universality. As a first step, we define an operation called NS_1 that acts as follows on a single **mode** that can contain either zero, one, or two photons:

$$\text{NS}_1(\alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle) = \alpha_0|0\rangle + \alpha_1|1\rangle - \alpha_2|2\rangle$$

NS_1 simply flips the phase if two photons occupy the mode, and does nothing otherwise. We don't care about its behavior on 3 or more photons, as that will never occur. Given the NS_1 gate, we can implement CPHASE on two encoded qubits with the following circuit:



(15.2)

If we apply two Hadamard beamsplitters with an NS_1 on each $|1\rangle$ mode in between, then we get CPHASE. Why? Let's see how it acts on each basis state.

If both qubits are in the $|0\rangle$ state, then the photons pass right through the circuit along the top and the bottom wires, and the state is unaffected. Good.

If just one of the qubits is $|1\rangle$, then one photon will pass through the beamsplitter. This creates an equal superposition state $\frac{|1\rangle|0\rangle+|0\rangle|1\rangle}{\sqrt{2}}$ or $\frac{|1\rangle|0\rangle-|0\rangle|1\rangle}{\sqrt{2}}$ depending on which mode had a photon, and these superpositions are unaffected by the NS_1 gates since neither mode can have two photons. Now, the second beamsplitter acts as the inverse of the first, therefore our final state is exactly our initial state. Good.

Finally, if both qubits are $|1\rangle$, then both photons pass through the beamsplitter. Because of the *Hong-Ou-Mandel effect*, discussed in a previous lecture, the resulting state will be

$$\frac{|2\rangle\,|0\rangle - |0\rangle\,|2\rangle}{\sqrt{2}}$$

as the photons bunch up in a single mode, and now we see that the NS_1 gates flips the phase of both branches of our superposition. The second beamsplitter again acts as the inverse of the first, giving us back our initial state but with a phase of -1 from the NS_1 gates.

Putting this all together, we see that the circuit in Equation 15.2 acts on two encoded qubits exactly as the CPHASE matrix we defined earlier acts on two qubits. Therefore, given the NS_1 gate we have CPHASE, and from there CNOT, and from there all of BQP. Fantastic!

Now what's left is just to find a linear-optical unitary transformation which can apply the NS_1 gate to a mode of our choice, so that we can actually create the circuit in Equation 15.2.

15.2.4 The Addition of Postselection

It turns out to be impossible [44] to construct the NS_1 transformation deterministically out of linear optics. So, we resort to loosening our success criteria by a little bit, by allowing our linear-optical network to use *postselected measurements*.

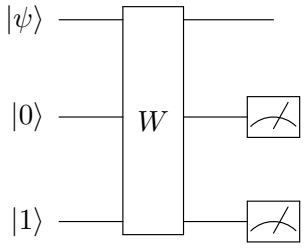
What is Postselection?

Postselection is a trick we can add to a quantum computing scheme which allows us to condition the entire computation on the result of one or more measurements, and throw away the cases where we don't get the desired result. In other words, as long as there is a nonzero chance that measuring a particular part of the system gives us an outcome we desire, we can assume that it happens and then work from there.

Now we see that what we can do about NS_1 is construct a transformation which **with nonzero probability** applies the NS_1 gate to a particular mode, but where we can measure some **other** auxiliary mode to determine whether this was successful or not. If we postselect on the success outcome we are done, and the NS_1 gate was applied to our target mode. Let's try this out.

15.2.5 Linear Optical NS_1 Gate

The idea is to create a linear optical unitary transformation W on multiple modes, some of which will be constant ancilla qubits, and one of which will be our input mode. Let's say that there will be two ancilla modes, one which constantly has zero photons and one which constantly has one photon, as shown below:



$\mathrm{NS}_1 |\psi\rangle$ given the below outcome (15.3)

$|0\rangle$ with prob. 1/4 postselected

$|1\rangle$ with prob. 1/4 postselected

Let's express the action of W on our input mode $|\psi\rangle$ parametrically:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle$$
$$W |\psi\rangle = \lambda_0\alpha_0 |0\rangle + \lambda_1\alpha_1 |1\rangle + \lambda_2\alpha_2 |2\rangle$$

What we want: $\lambda_0 = \lambda_1 = \frac{1}{2}, \lambda_2 = -\frac{1}{2}$

Why? Because up to normalization we want the zero- and one-photon basis states to be untouched, and for the two-photon state to undergo a phase flip. As for the normalization, we can see that if each λ_i has magnitude 1/2, then after the postselection succeeds with probability $(1/2)^2 = 1/4$, they will be re-normalized back to a magnitude of 1.

Now let's see how these goals concretely constrain the unitary matrix W which we want to construct.

We remember the rules for bosonic states passing through linear optical networks: for an m-by-m unitary matrix U on m modes and any basis vectors $|S\rangle = |s_1, s_2, \ldots, s_m\rangle, |T\rangle = |t_1, t_2, \ldots, t_m\rangle$ in the corresponding Fock space,

$$\langle S| \varphi(U) |T\rangle = \frac{\mathrm{Per}\ (U_{S,T})}{\sqrt{s_1! \ldots s_m! t_1! \ldots t_m!}}$$

where $U_{S,T}$ is the n-by-n submatrix of U with repeated rows and columns according to the occupation numbers $s_1 \ldots s_m$ and $t_1 \ldots t_m$, respectively.

Therefore, for our unitary transformation W,

$$\lambda_0 = \langle 0,0,1|\,\varphi(W)\,|0,0,1\rangle = \frac{\text{Per}\,\left(W_{3,3}\right)}{\sqrt{0!0!1!0!0!1!}} = W_{3,3} = \frac{1}{2}$$

$$\lambda_1 = \langle 1,0,1|\,\varphi(W)\,|1,0,1\rangle = \frac{\text{Per}\begin{pmatrix} W_{1,1} & W_{1,3} \\ W_{3,1} & W_{3,3} \end{pmatrix}}{\sqrt{1!0!1!1!0!1!}} = W_{1,1}W_{3,3} + W_{1,3}W_{3,1} = \frac{1}{2}$$

$$\lambda_2 = \langle 2,0,1|\,\varphi(W)\,|2,0,1\rangle = \frac{\text{Per}\begin{pmatrix} W_{1,1} & W_{1,1} & W_{1,3} \\ W_{1,1} & W_{1,1} & W_{1,3} \\ W_{3,1} & W_{3,1} & W_{3,3} \end{pmatrix}}{\sqrt{2!0!1!2!0!1!}} = 2W_{1,1}\lambda_1 + 2W_{1,3}W_{3,1}W_{1,1} = -\frac{1}{2}$$

In addition to these polynomial constraints, W must be unitary. We can find a suitable W as follows:

$$W = \begin{pmatrix} 1-\sqrt{2} & \sqrt{\frac{3}{\sqrt{2}}-2} & \frac{1}{2^{1/4}} \\ \sqrt{\frac{3}{\sqrt{2}}-2} & \sqrt{2}-\frac{1}{2} & \frac{1}{2}-\frac{1}{\sqrt{2}} \\ \frac{1}{2^{1/4}} & \frac{1}{2}-\frac{1}{\sqrt{2}} & \frac{1}{2} \end{pmatrix}$$

15.2.6 Recap: What exactly have we achieved?

Equipped with our unitary transformation W which can be implemented by a linear optical network with just a few beamsplitters and phaseshifters ([60]), as well as the superpower of postselecting on measurements of ancilla qubits, we have shown how to implement the NS_1 gate, and therefore universal quantum computation.

But we must be careful. The addition of postselection to our circuit makes a big difference in the conclusions that we can draw. What we have proved, for now, is that linear optical networks, under the supernatural assumption of being able to choose the outcomes of measurements, are able to simulate an arbitrary quantum circuit in polynomial time. Indeed, these networks can simulate any *postselected* quantum circuit, since we already have the postselection power for free. This result can be expressed in the following way:

$$\mathsf{PostLO} = \mathsf{PostBQP}$$

Where PostLO is the (ad-hoc) complexity class of decision problems that can be efficiently solved using a linear optical network with bounded error probability, and PostBQP is the class of decision problems that can be efficiently solved using a general quantum computer with bounded error probability and postselection on exponentially unlikely events. Aaronson proved in 2005 that PostBQP is equivalent to the very powerful classical complexity class PP [2], and therefore PostLO is also just as powerful.

This result doesn't quite satisfy our goal of showing that real-world linear optics are useful for general quantum computing, but it's a huge step in the right direction.

Lecture 16: Universal QC with Linear Optics and Adaptive Measurement

16.1 Overview

Last time, we saw that linear optics with postselection can efficiently simulate arbitrary quantum computation with postselection. That was a warmup to the "main" result of Knill, Laflamme, and Milburn [45]: a scheme that simulates arbitrary quantum computation using linear optics with adaptive measurements. Today, we will see that result. First, though, we'll use the postselected [45] theorem to give a linear-optical proof, due to Aaronson [4], that computing the permanent is #P-hard.

16.2 Computing the Permanent

Valiant [69] gave the following seminal result:

Theorem 21. *Computing the permanent is #P-hard.*

Unfortunately, his proof involves mysterious "gadgets" and is not so easy to follow. Today we'll see an alternative proof by [4] which uses linear optics, a subject that we should have pretty good intuition about by now.

This proof will be a standard reduction. Consider any Boolean circuit $C : \{0,1\}^n \to \{-1,1\}$. Boolean circuits typically output 0 or 1, but for the purpose of this proof, it's easier to let the output be -1 or 1. Now define:

$$\Delta_C := \sum_{x \in \{0,1\}^n} C(x).$$

Computing Δ_C is #P-complete since $(\Delta_C + 2^n)/2$ is the number of solutions to $C(x) = 1$. And the way we've defined Δ_C lends itself to be easily "embedded" into a quantum circuit. Specifically, let D_C be a unitary defined by $D_C |x\rangle = C(x) |x\rangle$ for all $x \in \{0,1\}^n$. Notice that D_C is diagonal.

Now, let $Q = H^{\otimes n} D_C H^{\otimes n}$ and compute:

$$\begin{aligned}\langle 0^n | Q | 0^n \rangle &= \left(\langle 0^n | H^{\otimes n} \right) D_C \left(H^{\otimes n} | 0^n \rangle \right) \\ &= \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}} \langle x | \right) D_C \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}} | x \rangle \right) \\ &= \frac{\Delta_C}{2^n}.\end{aligned}$$

That is, if we take $|0^n\rangle$, Hadamard every qubit, query D_C, Hadamard every qubit again, and then measure, the probability that we see $|0^n\rangle$ is $(\Delta_C / 2^n)^2$.

Last lecture, we saw that linear optics with postselection can simulate any quantum circuit. Let L_Q be a linear-optical network which uses Γ CNOT gates with postselection to simulate Q. If we remove postselection, then every CNOT gate has a $1/16$ chance of success, so the entire circuit has a $1/16^\Gamma$ chance of success.

Now, let's calculate the value of $\langle I | \varphi(L_Q) | I \rangle$, where $|I\rangle$ is the standard initial state. The case where every CNOT succeeds contributes an amplitude of $\langle 0^n | Q | 0^n \rangle / 16^\Gamma$, while the cases where some CNOTs fail contribute no amplitude since if a CNOT fails, then its ancilla photons were not returned to their original state.

This means:

$$\langle I | \varphi(L_Q) | I \rangle = \frac{\langle 0^n | Q | 0^n \rangle}{16^\Gamma} = \frac{\Delta_C}{2^n 16^\Gamma}.$$

Or:

$$\langle I | \varphi(L_Q) | I \rangle \, 2^n 16^\Gamma = \Delta_C.$$

As we have seen, calculating $\langle I | \varphi(L_Q) | I \rangle$ is a matter of computing a permanent, so if we have a way to calculate arbitrary permanents, we can also calculate Δ_C for arbitrary C. Thus, computing the permanent is #P-hard.

Grier and Shaeffer [35] have extended this technique to prove a few new hardness results, including:

Theorem 22. *Computing the permanent of a real orthogonal matrix is #P-hard.*

Theorem 23. *Computing the permanent of an orthogonal matrix in $\mathbb{F}_{p^4}$ for a prime $p \neq 2, 3$ is $\mathsf{Mod}_p\mathsf{P}$-hard, where $\mathsf{Mod}_p\mathsf{P}$ is an analog of #P for calculating the number of solutions mod p.*

16.3 Linear Optics with Adaptive Measurement

Our goal for this section is to explain the main result of [45]: Linear optics with adaptive measurement can be used to simulate arbitrary quantum computation. We saw last time that using the dual-rail representation makes it easy to apply one-qubit gates and perform one-qubit measurements, so the only thing we need for universality is to **implement a CNOT**.

Before we dive into linear optics, let's look at an idea we'll need from Gottesman and Chuang [34]: *gate teleportation.*

16.3.1 Gate Teleportation

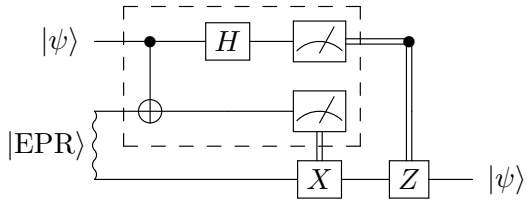
Gate teleportation is a scheme to implement universal quantum computation with a certain limited set of operations. To illustrate, let's follow [34] in proving the following theorem:

Theorem 24. *Universal quantum computation can be realized with the following:*

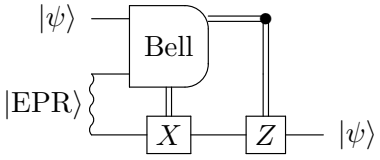
1. *One-qubit gates.*
2. *Adaptive 2-qubit measurements in the Bell basis:* $(|00\rangle \pm |11\rangle)/\sqrt{2}$ *and* $(|01\rangle \pm |10\rangle)/\sqrt{2}$.
3. *A supply of fixed entangled states meeting certain conditions. For example, a supply of GHZ states,* $(|000\rangle + |111\rangle)/\sqrt{2}$*, suffices.*

> Notice that having one-qubit gates and Bell-basis measurements is equivalent to having certain two-qubit measurements. That is, after preparing an initial state, universality can be achieved *only with measurements*!

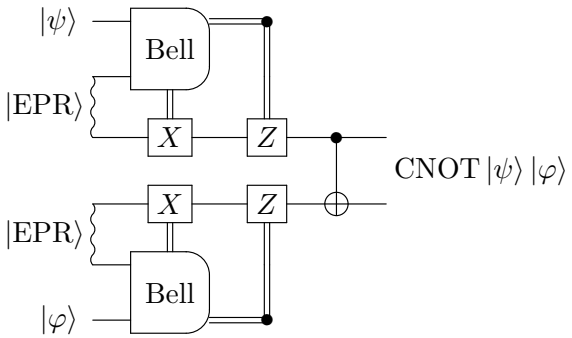
We have one-qubit gates and measurement, so we only need to implement CNOT in order to achieve universality. To do this, we use a modified version of quantum teleportation which applies a gate to qubits as they're being teleported, hence the name *gate teleportation*. First, recall standard quantum teleportation (here the double lines represent classical bits):



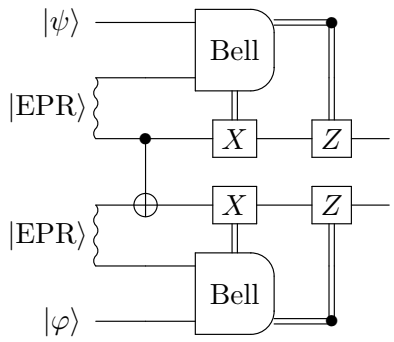
At first, it looks like we aren't allowed to do this since we have a CNOT in the protocol, but that CNOT, along with everything else in the dotted box, is actually a Bell-basis measurement:



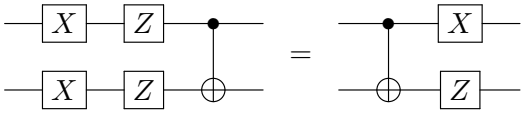
Now let's stack two of these teleportation circuits: one for the target qubit and one for the control. What we want is for the qubits to be CNOTed when they arrive. One way to do this is to just CNOT them after they arrive:



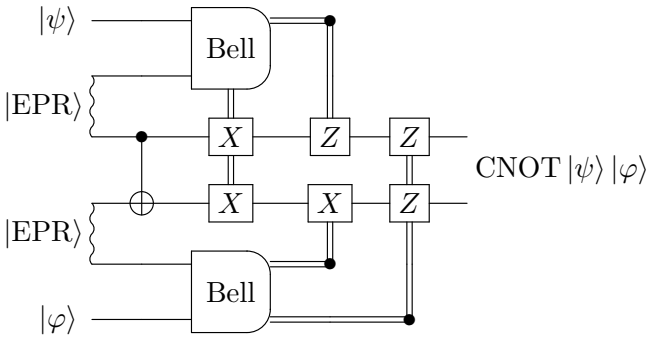
But of course we're not allowed to do that! Our goal was to implement a CNOT in the first place. Another idea is to apply the CNOT to the receiver qubits before the teleportation, like this:



Now the CNOT is just part of our entangled resource state. And if CNOT commuted with X and Z, then we'd be done. Unfortunately, it does not commute. But the next best thing is true: CNOT commutes with the Pauli group. More formally, if $\mathcal{P}_2$ is the Pauli group on 2 qubits, then $\text{CNOT}\,\mathcal{P}_2 = \mathcal{P}_2\,\text{CNOT}$. That is, for all $p \in \mathcal{P}_2$, there exists $q \in \mathcal{P}_2$ such that $\text{CNOT}\,p = q\,\text{CNOT}$. For example:



So the following circuit properly applies a CNOT and satisfies our restrictions:

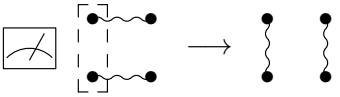


The entangled resource state we use is:

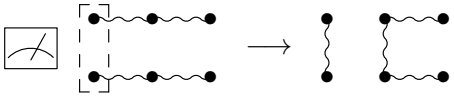
$$(I \otimes \text{CNOT} \otimes I)\frac{(|00\rangle + |11\rangle)(|00\rangle + |11\rangle)}{2} = \frac{|0000\rangle + |0011\rangle + |1110\rangle + |1101\rangle}{2}.$$

[45] showed that GHZ states can be used to construct this state.

A natural question is whether EPR pairs can be used as our resource states. Alas, the answer is no. Given only EPR pairs, we can't make a state with more than 2 entangled qubits. The only thing we can do to possibly increase entanglement is to measure qubits of different EPR pairs, but afterwards, we are still left with two pairs of entangled qubits. In contrast, if we do the same thing with two GHZ states, we are left with an entangled pair and an entangled quadruple, and we can use the quadruple to create even larger entangled states.



Measuring EPR pairs does not lead to larger entangled states.

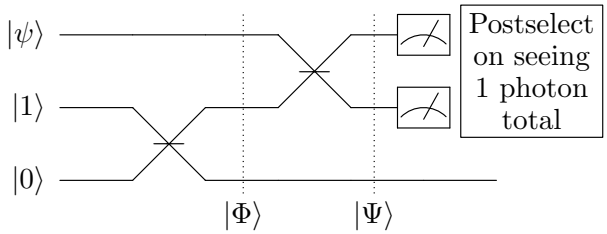


Measuring GHZ states can lead to larger entangled states.

16.3.2 Teleportation with Linear Optics

Now it is time to implement teleportation with linear optics, which, as one might imagine, is a little more complicated than teleportation with qubits.

Let $|\psi\rangle = a|0\rangle + b|1\rangle$ and consider the following linear-optical network:



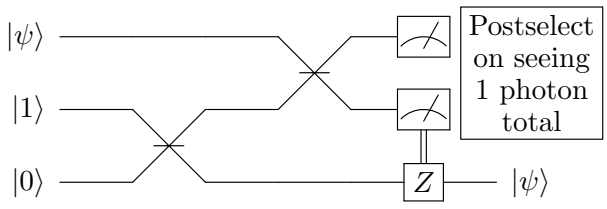
Both beamsplitters shown are Hadamards. The first Hadamard is used to construct an EPR pair, so the intermediate state $|\Phi\rangle$ is equal to:

$$\frac{(a|0\rangle + b|1\rangle)(|01\rangle + |10\rangle)}{\sqrt{2}} = \frac{a|001\rangle + a|010\rangle + b|101\rangle + b|110\rangle}{\sqrt{2}}.$$

Since the second Hadamard doesn't change the number of photons we will measure, we can apply the postselection now, yielding $|\Phi\rangle = a|010\rangle + b|101\rangle$. Now, let's apply the second Hadamard, which yields:

$$|\Psi\rangle = \frac{a|100\rangle - a|010\rangle + b|101\rangle + b|011\rangle}{\sqrt{2}}.$$

So, if we measure 10, the third mode will be in the state $a|0\rangle + b|1\rangle = |\psi\rangle$. Otherwise, we measure 01 and the third mode will be in the state $a|0\rangle - b|1\rangle = Z|\psi\rangle$. If we add in the correction Z-gate, we get:



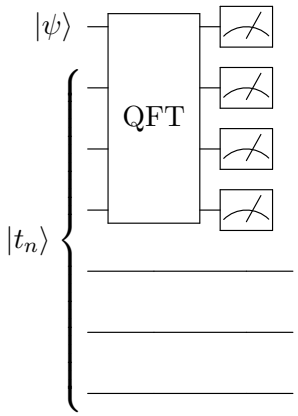
This network successfully teleports the mode, but it still uses postselection! If we remove the postselection, we get a network that succeeds with probability $1/2$, which isn't too good. Luckily for us, though, there is a generalization of this teleportation protocol that allows us to get success probabilities arbitrarily close to 1.

16.3.3 Removing Postselection

Our new teleportation protocol will use a generalized EPR pair, called $|t_n\rangle$ by [45]. $|t_n\rangle$ consists of n photons distributed over $2n$ modes like so:

$$|t_n\rangle = \frac{1}{\sqrt{n+1}} \sum_{j=0}^{n} |1^j 0^{n-j}\rangle \, |0^j 1^{n-j}\rangle \, .$$

Notice that $|t_1\rangle$ is an EPR pair. Also notice that if we measure the first n modes in a way that lets us know how many photons were in those modes, we can deduce exactly which of the last n modes have a photon in them. With that in mind, take a look at the following circuit using $|t_3\rangle$:



In place of the QFT, we could actually use any unitary whose entries all have the same magnitude. Now, suppose we observe k total photons when we measure. If $k = 0$ or $k = n + 1$, then the teleportation has failed and the state $|\psi\rangle$ has been irrecoverably damaged. Otherwise, there are two possibilities. For example, if $k = 4$, then either:

1. 0 photons came from $|\psi\rangle$ and 4 photons came from $|t_n\rangle$. This means that there is no photon in the $(n + 4)$th mode of $|t_n\rangle$.
2. 1 photon came from $|\psi\rangle$ and 3 photons came from $|t_n\rangle$. This means that there is a photon in the $(n + 4)$th mode of $|t_n\rangle$.

The first possibility contributes an amplitude of a and the second contributes an amplitude of b, possibly with a phase shift, causing the $(n + 4)$th mode to be in the state $a\,|0\rangle + be^{i\varphi}\,|1\rangle$ after the teleportation. This can be corrected with a phaseshifter, and we know which phase to apply based on our measurement outcomes. So then we end up with $|\psi\rangle$ in the $(n + 4)$th mode, as desired.

If we use $|t_n\rangle$ to teleport $|\psi\rangle$, then our probability of success is $\frac{n}{n+1}$, which doesn't *seem* good enough since that means we can only apply $\mathcal{O}(n)$ gates before a failure is more likely than a success. However, we already know that fault-tolerant quantum computing is possible, so we can boost the success probability of a logical gate arbitrarily close to 1 as long as n is a large enough constant. In fact [45] predicts that $n = 25$ will be sufficient.

We are almost done. We want to use gate teleportation, but we have yet to show how to construct our entangled resource state. Also, remember that $|t_n\rangle$ is not our resource state, $|t_n\rangle\,|t_n\rangle$ with

some CNOTs applied is (since we also need to do gate teleportation as in [34]). [45] call this state $|t'_n\rangle$, and they show how to construct it with $2^{\mathcal{O}(\sqrt{n})}$ elementary operations. That isn't too great asymptotically, but considering that we plan to keep n constant, it's good enough. And with that, we have achieved universality!

16.4 Next Time

We saw today that gate teleportation allows us to do quantum computing with fixed-size entangled resource states and two-qubit measurements. Next time, we'll look at **measurement-based quantum computing**, which achieves universality with *polynomial-sized* resource states and *one*-qubit measurements, and which comes with its own set of interesting consequences.

Lecture 17: Measurement-Based Quantum Computation

17.1 Overview

In the last lecture we wrapped up talking about the KLM scheme [45], which is a way to combine linear optics with adaptive measurements to get universal quantum computation. In the dual-rail encoding, one-qubit gates are easy to apply using beam splitters and phase shifters, but two-qubit gates, and in particular CNOT, aren't so straightforward.

Part of the solution was to use gate teleportation due to Gottesman and Chuang [34]. Given fixed entangled resource states and the ability to make measurements in the Bell basis, it's possible to simulate two-qubit gates.

The idea was to run a modified teleportation protocol where qubits come out with a CNOT having been applied to them. Because CNOT commutes with Pauli operators, it's always possible to apply Pauli operators after an initial CNOT and achieve an effect as if arbitrary Pauli operators had been applied prior to the CNOT.

While teleportation can be implemented linear-optically, it once again requires postselection, which we were trying to get rid of. The way to get around this was to use $2n$-mode resource states that we called $|t_n\rangle$, which allow teleportation with success probability $n/(n+1)$, which suffices due to quantum fault-tolerance.

In this lecture we'll see measurement-based quantum computation (MBQC), which, along with magic state distillation and KLM, is part of a recurring theme in quantum computing architecture where work is offloaded to the preparation of initial states and to measurements.

17.2 Introduction to cluster states

In 2000, Raussendorf and Briegel surprised the quantum computing community with the following theorem:

Theorem 25 ([56])**.** *There are fixed entangled "resource states" $|\psi_m\rangle$ on $m^{O(1)}$ qubits, using which one can simulate any circuit with $\leq m$ gates via one-qubit adaptive measurements only.*

The significance of this theorem is that any quantum computation can be done with all of the work of entangling done up front. The resource states can be thought of as "freeze-dried" quantum

computation.

With this approach to quantum computation, the main difficulty on the hardware side is just maintaining the coherence of these large entangled resource states.

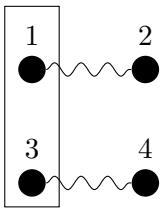
17.2.1 Entanglement in Gottesman-Chuang

Even without knowing what they are, we can reason about what the entangled resource states for MBQC would need to look like.

Recall that in the Gottesman-Chuang scheme, more powerful Bell-basis measurements were required, and in that case the resource states used were GHZ states. A natural question is whether Bell pairs in place of GHZ states would have been enough. Let's see why the answer is negative.

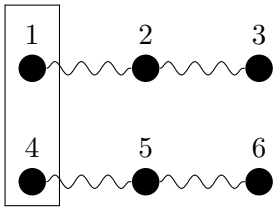
Bell pairs are on two qubits, but for universal quantum computation it's necessary to create entanglement on a large number of qubits.

Let's look at a situation where we have four qubits, where qubits 1 and 2 are a Bell pair and qubits 3 and 4 are a Bell pair.



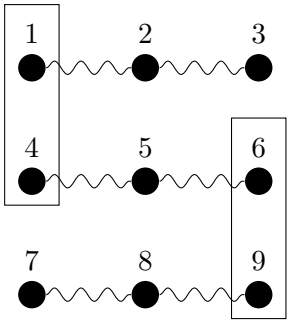
Suppose a measurement is made across qubits 1 and 3. Those qubits will collapse, possibly leaving qubits 2 and 4 in an entangled state. However, no matter how many times we repeat, we can't get a situation where more than two qubits are entangled.

In contrast, consider the situation where we have six qubits, where qubits 1–3 are in the GHZ state and so are qubits 4–6.



If we do a similar measurement on qubits 1 and 4, we can create a state where the four qubits 2, 3, 5, and 6 are entangled, up from three.

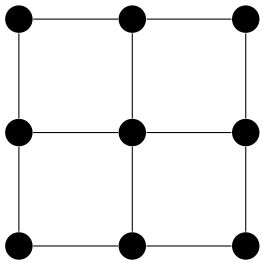
We can continue in this fashion to create larger and larger entangled states.



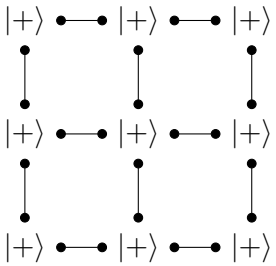
This example illustrates that if in MBQC we would like to move to only one-qubit measurements, which are incapable of creating entanglement, there will need to be some form of *global* entanglement among all of the qubits which exists from the beginning.

17.2.2 Cluster states

The resource states for MBQC are called *cluster states*. Their structure can be visualized using a 2D lattice, where the vertices represent qubits and the edges represent entangling operations which have been performed between neighboring qubits.



The example of a cluster state we'll use will be one where each qubit starts out in the $|+\rangle$ state, and a CPHASE gate, $\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix}$, is applied between every pair of adjacent qubits. Note that the order doesn't matter, as CPHASE operations are diagonal and commute with each other.

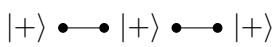


Written out in Dirac notation, this quantum state is:

$$|\psi_n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{\sum_{(i,j) \in E} x_i x_j} |x\rangle$$

where E is the set of edges.

For an explicit example of a cluster state, let's consider the 3×1 case:



Explicitly written in Dirac notation with normalization omitted, this quantum state is:

$$|000\rangle + |001\rangle + |010\rangle - |011\rangle + |100\rangle + |101\rangle - |110\rangle + |111\rangle$$

An interesting fact is that a cluster state must be 2D for universal quantum computation. We'll see in the next lecture that adaptive 1-qubit measurements on 1D states such as this can actually be efficiently simulated by classical computers.

Cluster states have desirable properties from both a practical and a theoretical viewpoint. To implement cluster states in hardware, the fact that they are 2D is convenient, as that allows them to, for example, be laid out on a silicon wafer. Mathematically, cluster states are stabilizer states, and additionally they can be generated by a circuit which is geometrically local and has constant depth.

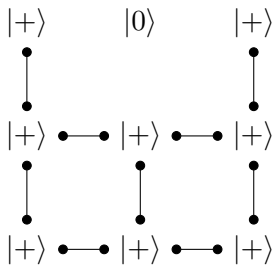
Cluster states also have significance in physics. When magnetized particles are placed in a lattice and allowed to interact only with their neighbors, the ground state of the system is something which resembles a cluster state. In this sense, cluster states represent a kind of entanglement that arises in nature.

17.3 Basic operations on cluster states

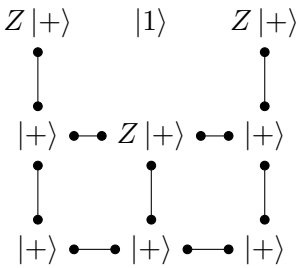
17.3.1 Destroying qubits

Now we'll see how it's possible to destroy qubits in a cluster state using measurement, in order to sculpt it into a state on a smaller number of qubits which encodes the geometry of a quantum circuit we'd like to run.

If a qubit is measured in the standard basis and the outcome is $|0\rangle$, then the remaining qubits are still in a cluster state.

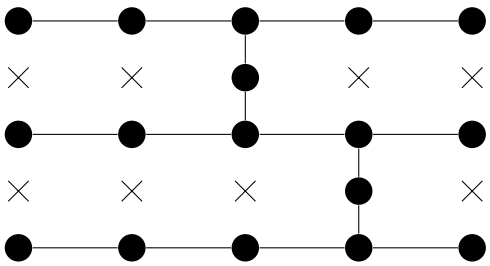


However, if the outcome is $|1\rangle$, there will be dangling -1 phases on all of the neighboring qubits, and it's necessary to apply phase gates to them in order to return the state to being a cluster state.



(Note that it's permissible to assume that one-qubit gates can be applied, because they can be folded into the final measurements which will be performed on the qubits.)

Through deleting qubits in this manner, it's possible to make patterns such as this one, which looks a lot like a quantum circuit!

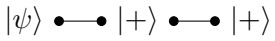


In this picture, we can imagine that each row corresponds to a qubit propagating rightward, while each vertical connection corresponds to an interaction between qubits.

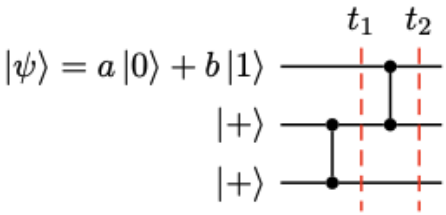
17.3.2 Moving qubits

Now we'll see how we can make a qubit move in a cluster state just by making measurements.

Imagine that we have a portion of the cluster state which looks like the following:

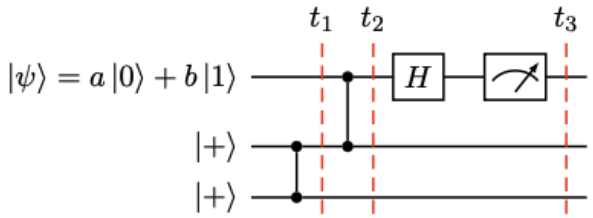


In terms of a quantum circuit, this is:



Because CPHASE gates commute, they can be written in any order, and it's convenient to use this order as it sets things up perfectly for teleportation—at time t_1, the last two qubits are in the state $|00\rangle + |01\rangle + |10\rangle - |11\rangle$, which is a Bell pair with the second qubit Hadamarded, and by time t_2, the first two qubits are entangled, which is the first step of teleportation.

We proceed by measuring the first qubit in the Hadamard basis.



At time t_2, the entire circuit is in the following state (omitting normalization throughout):

$$(a\,|000\rangle + b\,|100\rangle) + (a\,|001\rangle + b\,|101\rangle) + (a\,|010\rangle - b\,|110\rangle) + (-a\,|011\rangle + b\,|111\rangle)$$

The measurement will associate the pairs of amplitudes which are parenthesized. If the outcome is $|0\rangle$, then the state of the last two qubits at time t_3 will be:

$$(a+b)\,|00\rangle + (a+b)\,|01\rangle + (a-b)\,|10\rangle - (a-b)\,|11\rangle$$

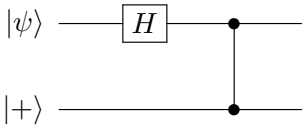
And if the outcome is $|1\rangle$, it will be:

$$(a+b)\,|00\rangle - (a+b)\,|01\rangle + (a-b)\,|10\rangle + (a-b)\,|11\rangle$$

Looking at the result if the outcome is $|0\rangle$, we see that it's equal to:

$$(a+b)\,|0\rangle\,|+\rangle + (a-b)\,|1\rangle\,|-\rangle$$

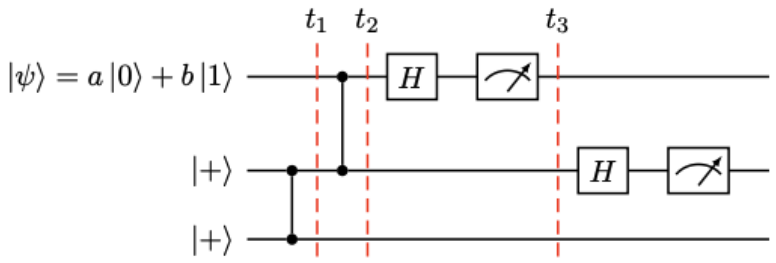
Which leaves the last two qubits as the following circuit would have:



In the case where the outcome of the measurement is $|1\rangle$, applying correction operations will lead to the same result.

To summarize, we've succeeded in moving $|\psi\rangle$ from the first qubit to the second qubit, and *additionally we've gotten a Hadamard gate for free.* It's easy to see that if $|\psi\rangle$ is moved again, it will pick up another Hadamard which will cancel the first one. And in general, moving $|\psi\rangle$ an odd number of steps will cause it to pick up a Hadamard while moving it an even number won't.

To complete the protocol, we have to measure the second qubit in the Hadamard basis and then apply any necessary correction operations on the third qubit. In our example circuit the state of the qubit at the end will be just $|\psi\rangle$, but if we were inside a larger number of qubits it would once again be encoded with CPHASEs.

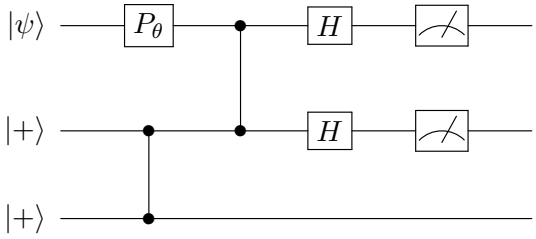


17.4 Universal quantum computation within cluster states

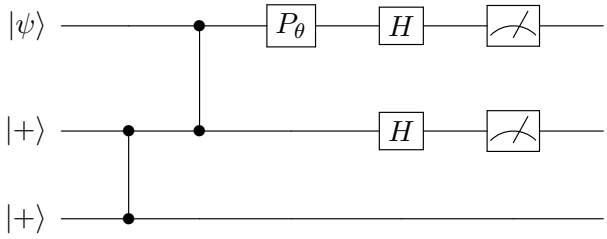
17.4.1 One-qubit gates

To do universal quantum computation, for starters we need to be able to apply any one-qubit gate. We'll see how to apply $P_\theta = \begin{pmatrix} 1 & \\ & e^{i\theta} \end{pmatrix}$, for any θ. This will be sufficient because H and P_θ generate all one-qubit gates (proof: exercise for you).

Going back to our teleportation protocol, the situation we'd like is the following, where the P_θ has been applied to $|\psi\rangle$ prior to the teleportation.



But remember, we don't have the ability to apply the P_θ on the first qubit prior to the CPHASE. However, because P_θ and CPHASE commute, this is equivalent to the following circuit:

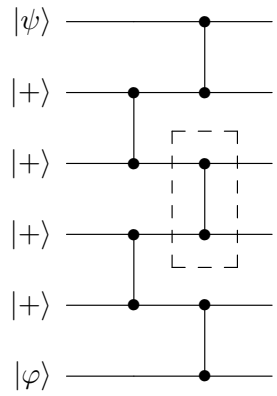


What that means is that all we have to do is to simply fold the P_θ gate into the measurement basis for the first qubit.

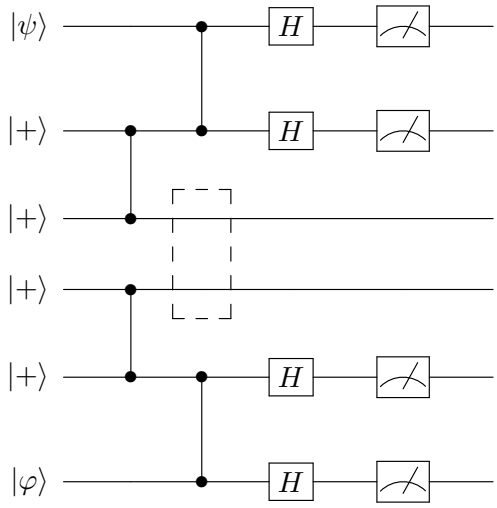
17.4.2 Two-qubit gates

The last thing remaining is to implement two-qubit operations, which we do by modifying Gottesman-Chuang gate teleportation.

We start with the following circuit, where we'd like to apply a CPHASE from $|\psi\rangle$ to $|\varphi\rangle$.

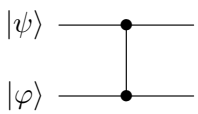


If the boxed CPHASE between the third and fourth qubits weren't present, we could move $|\psi\rangle$ to the third qubit and $|\varphi\rangle$ to the fourth qubit using the teleportation protocol:



But not only does this not reflect the actual situation where the CPHASE is in fact there at the beginning, it also doesn't do what we want, which is to output $|\psi\rangle$ and $|\varphi\rangle$ at the end with a CPHASE applied.

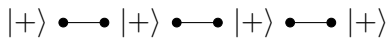
To solve both problems, we again just have to remember that CPHASE commutes with Pauli operators. By applying the appropriate correction operations when teleporting the qubits, we can move the CPHASE to the end and get a situation on the third and fourth qubits like so:



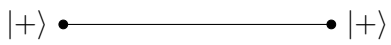
The CPHASE has been successfully implemented, but we aren't quite done, as this leaves the two qubits adjacent in the cluster state. This is a problem because they can't be moved apart again through teleportation, and the adjacency will cause undesired effects in future operations.

One thing we could do is to keep the qubits farther apart. But in order for our protocol to continue to work, we'd need some other way to get the initial CPHASE between the third and fourth qubits in the circuit.

Let's look at the example of a line of qubits in a cluster state.



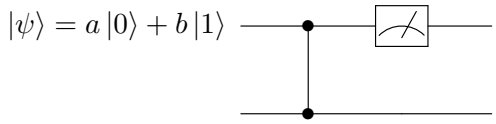
We've already seen that we can destroy qubits by measuring them in the standard basis. It turns out that if we instead measure in the Hadamard basis and apply the appropriate correction operations, we can contract paths. Going back to the picture above, if we measure the middle two qubits in the Hadamard basis and then correct, we'll get:



This allows us to get a CPHASE between two qubits even when they don't start out adjacent.

17.4.3 Measurements of final answers

In our final state, when we want to measure, each qubit may be connected with other qubits with CPHASE.



If the desired measurement is in the standard basis, then there's no issue as we'll still get $|0\rangle$ with probability $|a|^2$ and $|1\rangle$ with probability $|b|^2$. Measuring directly in other bases may not work due to the presence of the CPHASE, but we can get around this by simulating those measurements by applying more gates and then measuring in the standard basis.

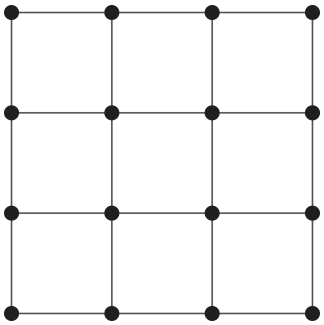
Lecture 18: Matrix Product States

18.1 Overview

In the last lecture we introduced **Measurement-Based Quantum Computing**, a paradigm by Raussendorf and Briegel which uses single qubit gates, adaptive measurement, and an entangled starting state known as a **Cluster State** to achieve universal quantum computation [57]. Crucially, two-qubit gates are *not* needed to achieve unversality.

18.1.1 Cluster States

For a cluster state on n qubits, we can think of the qubits as being arranged in a 2D lattice:



To construct the cluster state, we put all n qubits into an equal superposition of all 2^n bitstrings, and then apply a CPHASE gate, $\text{diag}(1, 1, 1, -1)$, along each edge within the lattice. This yields the state:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{\sum_{x_i, x_j \in E} x_i x_j} |x\rangle$$

However, a natural set of questions arise: Is a 1D lattice sufficient for universal quantum computation? What about a "long and skinny" 2D lattice, such as one that is $\log n \times n$?

As it turns out, neither of these are sufficient; in fact, measurement-based quantum computing becomes classically simulatable with these initial states. In this lecture, we will examine **Matrix Product States** and their connection to why these sorts of states are classically simulatable.

18.2 Matrix Product States

Definition. Let $|\psi\rangle$ be the state

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

Then $|\psi\rangle$ is a **Matrix Product State** with **Bond Dimension** D if there exist vectors $|v\rangle, |w\rangle \in \mathbb{C}^D$ and matrices $A_{1,0}, A_{1,1}, \ldots, A_{n,0}, A_{n,1} \in \mathbb{C}^{D \times D}$ such that for all $x \in \{0,1\}^n$,

$$\alpha_x = \langle v | A_{1,x_1} \ldots A_{n,x_n} |w\rangle .$$

18.2.1 Bond Dimension 1

To understand the definition better, let's consider the case when $|\psi\rangle$ has bond dimension 1. All $A_{i,b}$ will then be scalars, and the vectors $|v\rangle$ and $|w\rangle$ will also be scalars. Rewriting our state, we see that

$$|\psi\rangle = vw \sum_{x \in \{0,1\}^n} A_{1,x_1} \cdot A_{2,x_2} \ldots \cdot |x_1\rangle |x_2\rangle \ldots$$

But the term on the right here is factorable! It's simply

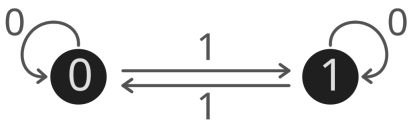
$$|\psi\rangle = vw \prod_{i=1}^{n} (A_{i,0} |0\rangle + A_{i,1} |1\rangle)$$

Thus, $|\psi\rangle$ having bond dimension 1 is equivalent to $|\psi\rangle$ being completely unentangled, a product state.

18.2.2 Higher Bond Dimensions

What about for higher bond dimensions? To grasp this concept better, we can think about it in terms of finite automata. Informally, a finite automaton is a machine with finitely many states that reads over a bitstring, one bit at a time, and transitions between states based on the input bit and current state.

To illustrate this idea, let's consider the problem of **Parity** – that is, determining the parity of the number of 1s in a bitstring. We can do this using a two-state automaton:



The two states simply correspond to the parity of bits seen so far. Initially, we start at 0. Whenever we read a 0, the parity does not change so we stay at our current state, whereas seeing a 1 flips the parity, so we switch states. The parity of the entire string is then just the state we end at.

For matrix product states, we need to modify the finite automaton model a little. Namely, we allow states to transition into possibly multiple states, assigning an "amplitude" to each transition. Each matrix $A_{i,b}$ corresponds to the state transitions to use when bit b is read at timestep i, while $|v\rangle$ and $|w\rangle$ correspond to the initial state and final state to be projected onto.

18.2.3 Bond Dimension 2^n

Using this insight, we can see that all n-qubit states have bond dimension at most 2^n. Let $|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$. We can set $|v\rangle$ to be the vector $[1, 0, 0, \ldots]^T$ and $|w\rangle$ to be the vector of amplitudes $[\alpha_{00\ldots0}, \alpha_{00\ldots1}, \ldots]^T$. Meanwhile, the $A_{i,b}$'s simply move the 1 from the initial state to the correct position; that is, $A_{i,0}$ is always just the identity function, while $A_{i,1}$ moves all states s to the state $2^i + s$.

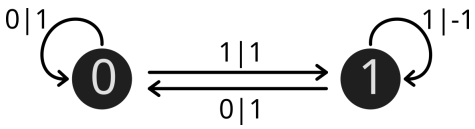
The result is that, on input x, we are left at the state whose binary label is x, which has corresponding final amplitude α_x.

18.2.4 One Dimensional Cluster States as Matrix Product States

Now, let's see how we can represent the 1D cluster state as a matrix product state, using our finite automaton perspective.

If we imagine an automaton proceeding across x from left to right, we can observe that the amplitude flips from $+1$ to -1 (and vice versa) each time we see two consecutive 1's.

This can be easily translated to a 2-state finite automaton with amplitudes. Each state represents the last bit we read in, and we multiply the amplitude by -1 only when there are two consecutive 1's:



Thus, we have

$$|v\rangle = \begin{bmatrix}1\\0\end{bmatrix} \qquad A_{i,0} = \begin{bmatrix}1 & 0\\1 & 0\end{bmatrix} \qquad A_{i,1} = \begin{bmatrix}0 & 1\\0 & -1\end{bmatrix} \qquad |w\rangle = \begin{bmatrix}1\\1\end{bmatrix}$$

And so 1D cluster states have bond dimension at most 2. Indeed the bond dimension is *exactly* 2, since there is entanglement and therefore we know the dimension cannot be 1.

18.3 Schmidt Rank and Bond Dimension

Now, we will examine the connection between **Schmidt Rank** and **Bond Dimension**.

Schmidt Rank. Recall that all pure states over two collections of qubits, A and B, can be represented in "Schmidt Form" as:

$$|\psi_{AB}\rangle = \sum_{i=1}^{\min(\dim A, \dim B)} \lambda_i |v_i\rangle |w_i\rangle ,$$

where all $|v_i\rangle \in A$ are orthogonal, and the same with all $|w_i\rangle \in B$. The number of non-zero λ_i's, which does *not* depend on the specific choice of $|v_i\rangle$'s and $|w_i\rangle$'s, is then called the **Schmidt Rank** of $|\psi\rangle$.

18.3.1 Bond Dimension and Schmidt Rank

As it turns out, the bond dimension of an n-qubit matrix product state satisifies the following:

$$\text{BondDim} |\psi\rangle = \max_{i \in [n]} \ \text{SchmidtRank} |\psi_{1\ldots i,\ i+1\ldots n}\rangle$$

where $\text{SchmidtRank} |\psi_{1\ldots i,\ i+1\ldots n}\rangle$ denotes the Schmidt rank of $|\psi\rangle$ when we partition the system into one subsystem containing qubits $1, \ldots, i$ and another containing $i+1, \ldots, n$. Note that this implicitly assumes an ordering on the qubits, but the notion of a matrix product state (and its associated bond dimension) already did the same.

Proof. First, we show that the bond dimension must be at least the Schmidt rank across any cut. Observe that we can rewrite the amplitude α_x as a sum over projection matrices $|i\rangle \langle i|$, which just sum to the identity:

$$\langle x|\psi\rangle = \alpha_x = \langle v| A_{1,x_1} \ldots A_{n,x_n} |w\rangle = \sum_{i=1}^{D} \langle v| A_{1,x_1} \ldots A_{t,x_t} |i\rangle \langle i| A_{t+1,x_{t+1}} \ldots A_{n,x_n} |w\rangle$$

We can then use the above product to define $|v_i\rangle$ and $|w_i\rangle$:

$$|v_i\rangle = \sum_{y \in \{0,1\}^t} (\langle v| A_{1,y_1} \ldots A_{t,y_t} |i\rangle) |y\rangle , \qquad |w_i\rangle = \sum_{z \in \{0,1\}^t} \left(\langle i| A_{t+1,z_{t+1}} \ldots A_{n,z_n} |w\rangle\right) |z\rangle$$

And thus we are left with a Schmidt decomposition of at most D vectors:

$$\begin{aligned} \langle x|\psi\rangle &= \sum_{i=1}^{D} \langle v| A_{1,x_1} \ldots A_{t,x_t} |i\rangle \langle i| A_{t+1,x_{t+1}} \ldots A_{n,x_n} |w\rangle \\ &= \sum_{i=1}^{D} \langle x_{1\ldots t}|v_i\rangle \langle x_{t+1\ldots n}|w_i\rangle = \langle x| \sum_{i=1}^{D} |v_i\rangle |w_i\rangle \end{aligned}$$

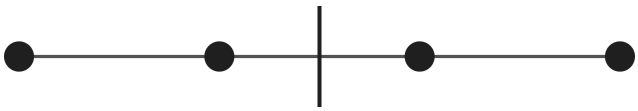
For the converse direction, we'll construct a matrix product state whose bond dimension is at most the maximum Schmidt rank. Thus, assume our state can be written as $|\psi\rangle = \sum_{i=1}^{D} |y_i\rangle |z_i\rangle$, where here and later we'll omit normalization for convenience. We can build up the matrix product state inductively, by maintaining that $\langle v| A_{1,x_1} \dots A_{t,x_t} |i\rangle = \langle x_1 \dots x_t | y_i\rangle$ for all $i \in \{1, \dots, t\}$.

To extend this property to $t+1$, imagine we measured the $(t+1)$st qubit and saw $|0\rangle$, yielding a state $|\psi_0\rangle = \sum_{i=1}^{D} |y_i'\rangle |z_i'\rangle$, with some possibly different Schmidt form between the first t qubits and the remaining $n-t$. Then all we need to do is use $A_{t+1,0}$ to represent the change of basis from $|y_i\rangle$'s to $|y_i'\rangle$'s. This gives us

$$\langle v| A_{1,x_1} \dots A_{t+1,x_{t+1}} |i\rangle = \langle x_1 \dots x_{t+1} | y_i'\rangle ,$$

as desired. We similarly construct $A_{t+1,1}$ by imagining the $(t+1)$st qubit was measured to be $|1\rangle$.

Remark. Intuitively, Schmidt rank represents how much entanglement there is between the two subsystems A and B, so the bond dimension simply represents the maximum entanglement along any "cut" of the cluster state. Earlier, we mentioned that a wide 2D lattice, such as one that is $\log n \times n$, is classically simulatable; indeed, any cut will only have $\log n$ entangled pairs crossing it, which corresponds to a Schmidt rank and bond dimension of $2^{\log n} = n^{O(1)}$. Soon, we will see how this polynomial upper-bound results in classical simulatability.



In the 1D case, the Schmidt rank across any cut is only 2, corresponding to a single EPR pair.

18.3.2 Improved General Upper Bound on Bond Dimension

Using the connection between bond dimension and Schmidt rank, we can get a tighter upper bound on the bond dimension of any n-qubit state, namely $2^{n/2}$. For consider the sizes of A and B; one of them will always contain $k \leq n/2$ qubits, so that even in the maximally entangled case only $2^k \leq 2^{n/2}$ EPR pairs can exist between the subsystems. Thus, all Schmidt ranks are at most $2^{n/2}$, so the bond dimension is also upper-bounded by $2^{n/2}$.

18.4 Classical Simulatability

A famous result by Guifre Vidal shows that we can classically simulate a T-gate quantum circuit with n qubits in time $\text{poly}(n, T, D)$, provided the bond dimension across any cut *never* exceeds D [72].

This result is a general statement for all quantum circuits, but we are only concerned with its implication for measurement-based quantum computing — namely, that we can simulate 1-qubit gates and measurements on a state with bond dimension $D = \text{poly}(n)$.

To apply a unitary $U = \begin{bmatrix} U_{0,0} & U_{0,1} \\ U_{1,0} & U_{1,1} \end{bmatrix}$ to qubit t, we can directly act on $A_{t,0}$ and $A_{t,1}$. Namely, we can simply construct new matrices $A'_{t,0} = U_{0,0}A_{t,0} + U_{0,1}A_{t,1}$ and $A'_{t,1} = U_{1,0}A_{t,0} + U_{1,1}A_{t,1}$.

To simulate a measurement of the tth qubit of $|\psi\rangle$ in the $\{|0\rangle, |1\rangle\}$ basis, we could naïvely compute the probability that we measure $|0\rangle$, as follows:

$$\begin{aligned}\mathbb{P}[\text{qubit } t \text{ is } |0\rangle] &= \sum_{x\in\{0,1\}^n, x_t=0} |\langle\psi| x_1 \ldots x_{t-1}\, 0\, x_{t+1} \ldots x_n\rangle|^2 \\ &= \sum_{x\in\{0,1\}^n, x_t=0} \langle w| A^\dagger_{n,x_n} \ldots A^\dagger_{t,0} \ldots A^\dagger_{1,x_1} |v\rangle \langle v| A_{1,x_1} \ldots A_{t,0} \ldots A_{n,x_n} |w\rangle\end{aligned}$$

We can rearrange this as follows, assuming x is indexed from $2 \ldots n$:

$$\sum_{x\in\{0,1\}^{n-1}, x_t=0} \langle w| A^\dagger_{n,x_n} \ldots A^\dagger_{t,0} \ldots A^\dagger_{2,x_2} \left(\sum_{x_1\in\{0,1\}} A^\dagger_{1,x_1} |v\rangle \langle v| A_{1,x_1} \right) A_{2,x_2} \ldots A_{t,0} \ldots A_{n,x_n} |w\rangle$$

This internal sum is easy to compute, since it's just a sum over two values. But this process repeats! Once we finish summing for A_{1,x_1}, we can then apply a similar idea to A_{2,x_2}, then A_{3,x_3}, and so forth until we have fully collapsed the sum.

Finally, to reflect the changes caused by measurement, we simply zero out the other matrix; that is, if we measure qubit t to be $|0\rangle$, then we set $A_{t,1}$ to be the all-zeroes matrix, and if we measure qubit t to be $|1\rangle$, then we set $A_{t.0}$ to be the all-zeroes matrix.

Lecture 19: Blind and Authenticated Quantum Computing

19.1 Overview

In the last lecture we talked about matrix product states and their connection to the simulatability of various types of cluster states. In this lecture we introduce a method for securing quantum computation using MBQC (measurement-based quantum computing).

19.2 Background

19.2.1 Notions of Security

In cryptography, there are generally two components to the security of data—confidentiality and integrity. Confidentiality ensures that an adversary cannot learn the contents of the data, and integrity ensures that an adversary cannot tamper with the data.

These notions can very easily be translated to the context of securing delegated computations, i.e when a client outsources a computation to a remote server. For confidentiality, we want the server to at least be unable to see the data it's operating on. A stronger notion (blindness) would include that the server is unable see the computation it's doing as well. For integrity, we want to be able to verify that the server did the computation the client requested and returned the correct result.

In the classical world, these problems are (partly) solved using fully homomorphic encryption and delegated computation protocols, at least under cryptographic assumptions. However for quantum computations, we will take a very different approach discovered by Broadbent, Fitzsimons, and Kashefi [25], which we'll call the BFK protocol.

19.2.2 NP, BQP, and IP

Before we jump into the BFK protocol, let's first formalize the notion of verifiable computation. We start at NP, the class of problems for which there exists a classical polynomial-time verifier which takes in only a witness as input. One example of an NP problem is factoring[1]—given a

[1]Technically, we first need to formulate factoring as a decision problem: for example, "is the jth bit in the prime factorization a 1?"

factorization of a number N, we can multiply the factors back together and check that we get N to verify that the factorization is correct.

However, there might be problems in BQP that are not in NP. For example, there is no obvious "witness" a quantum server could produce to convince a classical verifier that it simulated a quantum system correctly. Another example is Simon's problem: determining whether a given function f is two-to-one or one-to-one. In the case where f is two-to-one, we are guaranteed that there exists an s such that if $f(x) = f(y)$ where $x \neq y$, then $y = x \oplus s$; here s is called the secret shift. A classical verifier can be convinced that f is two-to-one by being given s. But if f is one-to-one, there is no obvious way to convince a polynomial time verifier that such an s does not exist.

To break free of NP prison, we can allow interactivity between the client and server. Now instead of being limited to "single-shot" proofs (witnesses), the client and server can communicate back and forth, giving us the complexity class IP (interactive proofs). We maintain the restriction that the entire protocol must run in polynomial-time. In the classical world, the jump from single-shot to interactive proofs greatly expands the class of problems we can verify from NP all the way to PSPACE [64].

Since our server is bounded and can only perform polynomial-time quantum computations, we limit ourselves to $\mathsf{IP}_{\mathsf{BQP}}$, the subset of IP where all computation done by the server must be in BQP. From here, we'd ideally show that $\mathsf{IP}_{\mathsf{BQP}} = \mathsf{BQP}$, meaning that every quantum computation can be verified by a classical client. However, things are not so simple. The BFK protocol requires that we give our clients a little bit more power: namely, the ability to transmit unentangled qubits in arbitrary states to the server. We denote the resulting complexity class $\mathsf{IP}^{|\theta\rangle}_{\mathsf{BQP}}$. Luckily, this ability is not too unrealistic, since we already know how to transmit qubits through fiber-optic cables; for example, it does not require the verifier to have a quantum computer.

19.3 The BFK Protocol

The BFK protocol relies on a special type of cluster state called a *brickwork state* (see figure 19.1). As is typical in cluster states, each of the qubits is initialized to the $|+\rangle$ state, and then a controlled phase gate is applied for every edge. Doing measurement-based quantum computing with the brickwork state is universal. The BFK protocol then provides verifiable, blind quantum computation using MBQC with the brickwork state, meaning that its existence proves that $\mathsf{IP}^{|\theta\rangle}_{\mathsf{BQP}} = \mathsf{BQP}$. Each row of the brickwork state will represent one qubit, while each column will represent one time step. To apply a quantum circuit, we measure column by column, proceeding from left to right. CNOTs can be applied whenever there is a vertical edge, allowing interaction between two rows.

The protocol works in three stages.

1. The client initializes all the qubits to be used in the brickwork state, applying secret random phases known only to the client, and sends them to server.

2. The server initializes the brickwork state over the given qubits by applying controlled phase gates on the appropriate edges.

3. For each column, going from left to right, the client then directs the server to measure each

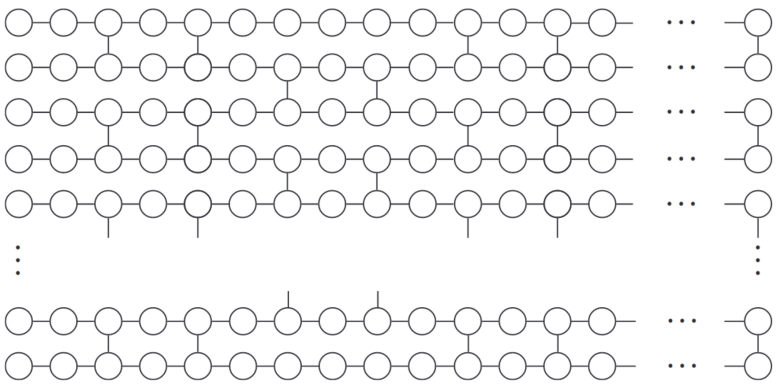


Figure 19.1: A brickwork state.

of the qubits in certain bases, and the server returns the results of those measurements. This process repeats until all qubits have been measured.

In the first phase, the client chooses a random $\theta_{x,y}$ for each qubit, where each $\theta_{x,y}$ is drawn uniformly at random and independently from the set $\{0, \pi/4, \ldots, 7\pi/4\}$. The client then initializes each qubit to

$$|\phi_{x,y}\rangle = \frac{|0\rangle + e^{i\theta_{x,y}}|1\rangle}{\sqrt{2}}.$$

Note that this can be viewed as applying a phase shift to each qubit if all the qubits started in the $|+\rangle$ state. Thus it commutes with the controlled phase gates that the server applies in stage 2.

This means that the first two steps have the same effect as starting with a standard brickwork state and then allowing the client to apply its secret phase shifts. The shifts then act as a one-time-pad, "encrypting" each of the qubits into a random state so that it looks maximally mixed to the server.

In the third phase, the client goes column by column, performing their quantum computation over the cluster state while adjusting for their random initialization of the qubits. The client only ever measures in the four orthonormal bases formed by the eight states $(|0\rangle + e^{i\theta}|1\rangle)/\sqrt{2}$ for $\theta \in \{0, \pi/4, \ldots, 7\pi/4\}$. Note that this set of measurements is still universal for quantum computation (which is why it was chosen).

However, whenever the client wants to measure in the basis formed by two qubits $|a\rangle, |b\rangle$, they randomize the order of the basis vectors they send to the server. This makes it so the server cannot extract information from the measurement result, since it doesn't know whether the client will interpret the measurement result as a 0 or 1.

Here's a sketch for why the protocol provides blind quantum computing. The randomization in the first stage hides the "true" measurement bases from the server, and the randomization in the third stage hides the true measurement results from the server.

Thus, the server just sees a collection of maximally mixed qubits, which it measures in uniformly random bases to get uniformly random outcomes, and gains literally no information about which quantum computation it's performing, other than an upper bound on its length. However, this protocol would only work for an "honest but curious" server, since we can't yet guarantee that the server is following the protocol correctly.

The solution to this problem is for the client to insert random "tripwire qubits" into the cluster state in way such that it already knows what a measurement outcome on the tripwires should be. Suppose p is the proportion of tripwires in the cluster state. Due to the blindness of the protocol, the server doesn't know which qubits are the tripwires, so if it attempts to deceive the client by changing one or more measurement results, then with probability $\geq p$ it'll be lying on one of the tripwire qubits.

Since the client already knows what the measurement outcomes on the tripwires should be, this means that the client will detect a dishonest server with probability at least p. To increase the probability of detecting dishonesty, the protocol can be rerun multiple times. Thus the BFK protocol provides blind and verifiable quantum computing for all problems in BQP, proving that $\mathsf{IP}^{|\theta\rangle}_{\mathsf{BQP}} = \mathsf{BQP}$.

Lecture 20: Authenticated QC and Quantum State Tomography

20.1 Overview

In the last lecture we discussed the Broadbent, Fitzsimons, and Kashefi (BFK) [25] blind and authenticated quantum computing protocol. The BFK protocol provides one way for a classical client to check if a quantum computer is doing the calculations it is asked to do. BFK is considered blind because the quantum computer only knows that it is applying some gates to maximally mixed states which the client provides. The client can authenticate the measurements of the quantum computer by the use of 'tripwire' qubits that end up in the wrong states with high probability if incorrect calculations are done.

In this lecture we discuss the subsequent developments in this area, such as those of Reichardt, Unger, and Vazirani (2012) [61], and Mahadev (2018) [50]. After finishing our discussion of blind and authenticated quantum computing, we begin our discussion of quantum state tomography.

20.2 Subsequent Developments in Blind and Authenticated Quantum Computing

20.2.1 Drawback of the BFK Protocol

Question: What is unsatisfying about BFK?

It requires the classical client to send quantum states to the quantum computer. This cannot be done with the classical Internet and would therefore require a 'quantum internet,' or else a classical client that was close to the quantum computer. Furthermore the client creating and sending quantum states, is hard to reconcile with the idea that the client is 'purely classical.'

BFK's Proposed Solution to a Fully Classical Client

BFK suggested a possible solution which required two entangled but non-communicating quantum servers. A classical client, Alice, communicates classically with both entangled servers, while the servers cannot communicate with each other. Furthermore, the protocol assumes the servers have a polynomial number of EPR pairs. These states are represented as $|\phi^+_{x,y}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, where

$x = 1, ..., n$ indexes the columns of the brickwork state, and $y = 1, ..., m$ indexes the rows. Alice then selects a basis for the first server to measure $|\phi^+_{x,y}\rangle$ in, with basis vectors corresponding to angles $\theta_{x,y} \in_R \{0, \pi/4, 2\pi/4, ..., 7\pi/4\}$. Server 1 then responds with the measurement outcomes, $m_{x,y}$. Due to the entanglement between the servers, the second server's states collapse. Therefore, Alice can send Server 2 a measurement scheme, where the measurement basis is defined by $\tilde{\theta}_{x,y} = \theta_{x,y} + m_{x,y}\pi$. Alice can then verify the second server's responses as in the original BFK protocol.

20.2.2 Reichardt, Unger, and Vazirani (2012)

Unfortunately, BFK were unable to prove the soundness of this protocol. A few years later, Reichardt, Unger, and Vazirani [61] re-approached the problem from a different angle.

Question: How through classical communication can a verifier, Alice, verify that she is communicating with quantum computers at all?

Alice can play the CHSH game, introduced in QIS I, with both servers. In this game, challenge bits (x and y) are sent to Servers 1 and 2 respectively. Both servers then respond with answer bits (a and b), the servers win the game iff $ab = x \oplus y$. The best classical strategy wins 75% of the time, while entangled quantum devices can win the game with probability $\cos^2(\pi/8) \approx 85.4\%$. Therefore if the servers win $> 75\%$ of the time, the only explanation is they are entangled. The key insight of RUV was that the CHSH game not only verifies that the servers are quantum, but also tells Alice about what unitary operations they are applying. In a far-reaching generalization, RUV showed how to use CHSH-type games to then force the servers to apply a specific circuit proving to Alice that they applied the right gates at each step.

RUV then concluded that $\mathsf{MIP}^*_{\mathsf{BQP}} = \mathsf{BQP}$, where $\mathsf{MIP}^*_{\mathsf{BQP}}$ denotes the class of problems that admit a multiprover interactive proof system with entangled provers with only the power of BQP.

20.2.3 Mahadev (2018)

In a breakthrough, Mahadev [50] gave the the first verification problem with fully classical communication and only one prover. The key assumption of Mahadev's protocol is that a certain cryptographic problem is hard for a quantum computer to solve. The specific problem she uses is called Learning With Errors or LWE. One can think of LWE as solving a system of linear equations with random noise; a large portion of quantum-resistant cryptography (and in particular lattice-based cryptography) is now based on the assumed hardness of this problem. Mahadev's theorem is that, if LWE is hard for quantum devices, then $\mathsf{IP}_{\mathsf{BQP}} = \mathsf{BQP}$.

Main Idea: Using LWE, the client can construct a trapdoor 2-to-1 function. A trapdoor function is a function f which is easy to compute but hard to invert without special information called the 'trapdoor' and easy to invert with the trapdoor. A 2-to-1 function has exactly two pre-images for any output, $f(x) = f(y) = c$. Alice, in this case, knows the trapdoor and how to efficiently compute f. She shares the code with the server, so the server knows how to compute f, as well. Alice keeps secret how to invert f, so only she, not the server, can feasibly do that.

Question: How can the client know the server is not classical?

Alice tells the server to repeatedly prepare quantum states of the form

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle .$$

Alice then asks the server to measure the $f(x)$ register and return the result. This leaves the first register in an equal superposition of the pre-images, $\frac{|x\rangle + |y\rangle}{\sqrt{2}}$. Alice is aware of what x and y are because she knows $f(x)$ and can invert f. Alice can then ask the server to measure the state $\frac{|x\rangle + |y\rangle}{\sqrt{2}}$ in either the standard or Hadamard bases. In the latter case, she would expect to see some random string, z, such that $z \cdot (x \oplus y) = 0 \pmod 2$. If the server fails this test then the state is not in the superposition it should be in, indicating the server performed incorrect calculations.

Problems with Mahadev's Protocol:

1. It is expensive in practice, requiring a full fault tolerant quantum computer to simulate even a single gate.
2. It is not known whether LWE is hard for a quantum computer, and we have no hope of proving this in our current state of knowledge.

20.2.4 Summary of all protocols

The table below summarizes the strengths and weaknesses of the three protocols we've discussed: BFK, RUV, and Mahadev's.

	Information Theoretic Security	One prover (Quantum Computer)	Classical Communication
BFK [25]	✓	✓	✗
RUV [61]	✓	✗	✓
Mahadev [50]	✗	✓	✓
?	✓	✓	✓

Potentially there is a protocol that is information-theoretically secure, has only one prover, and relies on purely classical communication, but if so, that solution has not yet been discovered!

20.3 Quantum State Tomography

Quantum state tomography revolves around the idea that we are classical beings trying to learn information about quantum states through measuring them. Specifically, suppose you have an unknown quantum state in nature and want to determine what it is. More precisely, imagine you have a machine that whenever you press a button it returns a copy of ρ, a mixed state. We would like to learn ρ as a density matrix.

Basic Observations:

1. We'll need more than one copy of ρ because measurements are destructive.
2. Even then, we can only hope to approximate ρ, that is to output a density matrix $\tilde{\rho}$ such that $\tilde{\rho} \approx \rho$ since we cannot learn continuous parameters exactly from a finite number of discrete measurement outcomes.

We can measure the closeness between $\tilde{\rho}$ and ρ using the **trace distance**, a standard distance metric for mixed states. The trace distance is defined as

$$||\tilde{\rho} - \rho||_{tr} := \frac{1}{2} \sum_i |\lambda_i|, \tag{20.1}$$

where λ_i are the eigenvalues of $\tilde{\rho} - \rho$. Our goal is then to output a $\tilde{\rho}$ such that $||\tilde{\rho} - \rho||_{tr} \leq \varepsilon$.

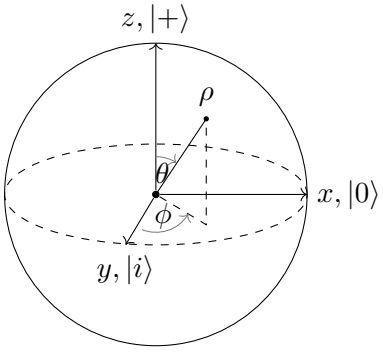
20.3.1 1-Qubit Case

To build intuition, let's start with the case where we are trying to learn just a single qubit, in state

$$\rho = \begin{pmatrix} a & b \\ b^* & c \end{pmatrix}. \tag{20.2}$$

We can use the following strategy:

1. Measure in the standard basis to determine a and c.
2. Measure in the Hadamard and $\{|+i\rangle, |-i\rangle\}$ bases to determine b.



On the Bloch Sphere, the standard basis measurement teaches us about ρ's position along the x-axis, the Hadamard basis measurement teaches us about ρ's position along the y-axis, and the $\{|+i\rangle, |-i\rangle\}$ basis measurement teaches us about ρ's position along the z-axis. In order to get additive error at most ε in each of these three measurements (and therefore in the trace distance), standard statistics tells us we need to repeat each measurement $\mathcal{O}(\frac{1}{\varepsilon^2})$ times, as usual on different copies of ρ.

20.3.2 D-dimensional Hilbert Space Case

In general, we are trying to learn a $D \times D$ Hermitian, positive semi-definite, trace-1 matrix that looks like

$$\rho = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1n} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{n1} & \rho_{n2} & \cdots & \rho_{nn} \end{pmatrix}. \tag{20.3}$$

We could think of ρ as made of $\log_2(n)$ qubits, but it will be more convenient for now to think of it as a single particle in a superposition of n modes.

In order to determine the diagonal entries of ρ, we measure ρ in the standard basis. Next we can apply a Hadamard beamsplitter between each neighboring pair of modes, so that the matrix applied looks like

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & . & . & . \\ 1 & -1 & . & . & . \\ \vdots & & \ddots & & \vdots \\ . & . & . & 1 & 1 \\ . & . & . & 1 & -1 \end{pmatrix}. \tag{20.4}$$

This allows us to learn about the corresponding 2×2 blocks of ρ. Finally, to complete our knowledge of those 2×2 blocks we measure ρ in the basis

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i & . & . & . \\ 1 & -i & . & . & . \\ \vdots & & \ddots & & \vdots \\ . & . & . & 1 & i \\ . & . & . & 1 & -i \end{pmatrix} \tag{20.5}$$

where each 2×2 block consists of the B matrix,

$$B = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}. \tag{20.6}$$

We then repeat all this with different permutations of the modes to learn ρ's other off-diagonal entries. Therefore the number of different measurement bases needed is $1 + 2\lfloor D/2 \rfloor$, which is at most $D + 1$. This gives us a so-called **tomographically complete set** of measurement bases, or a set of measurement bases for which the statistics of their outcomes suffice to reconstruct ρ. Returning to our original question, we still need to figure out how many copies of ρ we need to be within ε.

Goal: Find a $\tilde{\rho}$ such that $||\tilde{\rho} - \rho||_{tr} \leq \varepsilon$.

To achieve this, we will determine each entry of ρ to additive error $\pm\delta$. This reduces the problem to, how small does δ have to be to achieve $||\tilde{\rho} - \rho||_{tr} \leq \varepsilon$? If every entry of ρ is determined to accuracy $\pm\delta$, we know each entry of the matrix $\tilde{\rho} - \rho$ is at most δ in absolute value. From this it's

not hard to show that every eigenvalue of $\tilde{\rho} - \rho$ is at most $\delta\sqrt{D}$ in absolute value, and if there are D eigenvalues the trace distance is therefore

$$\frac{1}{2}\sum_{i=1}^{D} |\lambda_i| \leq \frac{\delta D^{3/2}}{2} \tag{20.7}$$

Solving

$$\varepsilon = \frac{\delta D^{3/2}}{2}$$

for δ, we get

$$\delta = \frac{2\varepsilon}{D^{3/2}}.$$

Finally, each entry is determined to $\pm\delta$ through $1/\delta^2$ repetitions. So the total number of copies of ρ is

$$\mathcal{O}(\frac{D}{\delta^2}) = \mathcal{O}(\frac{D^4}{\varepsilon^2})$$

since there are D different bases and we have to measure in each basis $\mathcal{O}(1/\delta^2)$ times. This gives us a first upper bound for how many copies of a quantum state are needed to do tomography. Later work saw a reduction in the upper bound, but this will be covered in the next lecture!

Lecture 21: Tomography (continued) and Learning Theory

21.1 Overview

In the last lecture, we introduced the problem of quantum state tomography: we are given samples of an unknown D-dimensional (potentially mixed) quantum state and want to estimate its density matrix ρ. We obtained an algorithm to estimate the state to trace distance at most ε using $O(D^4/\varepsilon^2)$ copies of ρ by considering tomographically complete bases and estimating the parameters of ρ in each of these bases.

In this lecture, we discuss algorithms for state tomography with improved sample complexity, first bringing the number of samples down to $O(D^3/\varepsilon^2)$ then $O(D^2/\varepsilon^2)$. We argue that, up to logarithmic factors, the latter is optimal. That is, for a D-dimensional quantum state, the sample complexity of state tomography is quadratically related to D, and this is essentially tight.

Since the cost of full quantum state tomography often puts it out of reach for even a modest number of qubits, we'll aim to relax the goal. We discuss some potential relaxations and introduce some relevant concepts from classical computational learning theory.

21.2 Efficient Tomography

The $O(D^4/\varepsilon^2)$ protocol from last lecture begs the question: can we attain better sample complexity? Furthermore, what if ρ is a pure state, or more generally of low rank?

For a while, the best known algorithm for tomography required $O(D^4/\varepsilon^2)$ samples. Then, in 2014, Kueng, Rauhut, and Terstiege improved this result to $O(D^3/\varepsilon^2)$ [46]. Then, Haah, Harrow, Ji, Wu, and Yu obtained an algorithm for quantum tomography using $O(rD\log(D)/\varepsilon^2)$ copies of ρ [37]. Independently, O'Donnell and Wright obtained an algorithm using $O(D^2/\varepsilon^2)$ copies [53]. Both algorithms take advantage of low-rankness in density matrices to obtain an $O(rD/\varepsilon^2)$ algorithm for tomography on rank r mixed states. In other words, the sample complexity of performing tomography on a pure state is linear in its dimension[1].

The high level idea of O'Donnell et. al. was to exploit representation-theoretic results to construct measurements. Essentially, they combine earlier results on representation-theoretic measurements

[1] Actually, this result was known prior to O'Donnell et. al. and Haah et. al., but the sample dependence was on the square of the rank, thus the $O(D^3/\varepsilon^2)$ upper bound for mixed states.

to produce a partition on the Hilbert space of ρ, as well as a random unitary which, when combined, provide a good bound on the Frobenius norm[2] of $\tilde{\rho} - \rho$, which then yields a bound on the standard trace norm. As the exact details are beyond the scope of this lecture (and likely the course) we refer the reader to O'Donnell and Wright 2016 [53], where a full argument is given.

21.2.1 Lower Bounds

In this section, we argue $O(D^2/\varepsilon^2)$ samples are essentially optimal, at least in the dependence on D. The argument relies on a protocol with two parties. We'll give an information-theoretic lower bound on the number of qubits required to convey to one party which of a set of mixed states was selected by the other.

Claim 26. *For every D, there exist $K = \exp(D^2)$ states $\rho_1, ..., \rho_K$ of dimension D such that for all $i \neq j$*

$$\|\rho_i - \rho_j\|_{Tr} \geq \frac{9}{10}$$

The idea is just to sample K random subspaces, each of dimension say $D/10$, and take ρ_i to be a uniform mixture over the ith subspace. A full proof, with dependence on ε, is given in [30].

Theorem 27. *At least $\Omega(D^2/\log D)$ copies of ρ are required to learn a D-dimensional state ρ to constant error in trace distance.*

Proof. Consider the following protocol. Alice and Bob fix a set of $K = \exp(D^2)$ states of dimension D as in Claim 26. Alice picks $i \in [K]$ and sends $\rho_i^{\otimes m}$ to Bob. Bob wishes to determine i. By the Holevo bound, $\Omega(\log K) = \Omega(D^2)$ qubits are needed to convey this information. Since each ρ_i consists of $\log D$ qubits, Alice sends $m \log D$ qubits to Bob, and so we must have that $m = \Omega(D^2/\log D)$. □

For a diagram of the protocol, see Figure 21.1.

21.2.2 Limitations of Tomography

Consider the following drawback of tomography: if we wish to perform complete tomography on a state of dimension $D = 2^n$, this requires $\Theta(D^2) = \Theta(2^{2n})$ operations, making it very expensive. In fact, even the state-of-the-art experiments have only been able to perform tomography on states consisting of about 10 qubits [63].

As seen in the previous section, we cannot circumvent this exponential dependence on the number of qubits, but perhaps changing the paradigm will lead to a more tractable problem. There are several changes we can make to the basic tomography question, including:

1. *Prior Knowledge:* Using clues about the structure of the state could simplify the problem: for example, given any candidate pure state $|\psi\rangle$, it is easy to check whether ρ is close to or far from $|\psi\rangle\langle\psi|$ by simply projecting onto $|\psi\rangle$.

[2] The Frobenius norm is the square root of the sum of squares of matrix entries. It can be thought of as the 2-norm of the matrix treated as a vector.

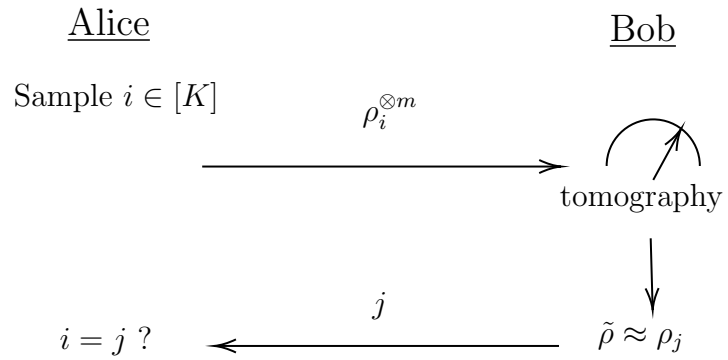


Figure 21.1: The protocol from the proof of Theorem 27. Alice samples and sends a random ρ_i and Bob uses tomography to find a $\tilde{\rho}$ and sends the j that $\tilde{\rho} \approx \rho_j$. In order for $j = i$, Alice would need to send $\Omega(\log K)$ qubits.

2. *Special States:* If we restrict the domain of states we wish to learn (to, for example, stabilizer, fermionic, or matrix product states) then we might be able to get much better tomography procedures.

3. *Relaxing the Problem:* To reconstruct ρ as a matrix is equivalent to learning its behavior in every possible measurement basis. We could instead try to learn ρ's behavior on only *some* or *most* measurements.

In the rest of this lecture, we'll concern ourselves with the third item on this list. That is, we will relax the tomography problem to approximating the behavior of ρ on "most" bases. First, though, we will need some learning theory fundamentals.

21.3 Learning Theory

In this section, we introduce the paradigm of PAC-learning and state some theorems regarding its sample complexity. These concepts will then be useful when defining a relaxed form of quantum state tomography.

21.3.1 PAC-learning

In learning theory, our goal is to deduce a function out of a universe (called a "concept class") of possible functions given a set of function evaluations (called "training examples"). For now, we consider functions whose output is boolean.

Definition 28. *A concept class is a set of functions from some domain $\mathcal{U}$ to $\{0, 1\}$.*

A basic goal here is **PAC-learning**, where PAC stands for "**P**robably **A**pproximately **C**orrect." In the PAC framework, introduced by Valiant [70], we seek a hypothesis that matches the concept on most inputs, and we also allow for the possibility that our procedure fails to do this with small probability.

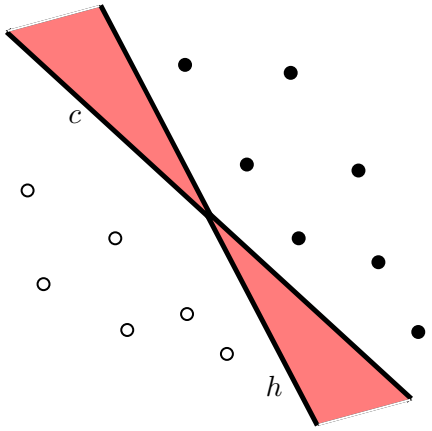


Figure 21.2: PAC-Learning: Suppose the white points are labeled 0 and the black points are labeled 1, and that c is the mystery concept (from the concept class of hyperplanes in $\mathbb{R}^2$. The hypothesis h, is not a perfect representation of c as it classifies incorrectly in the shaded region. However, as long as the probability of sampling in this shaded region is small, h satisfies the conditions of the PAC-learning problem.

Formally, fix a concept class $\mathcal{C}$ (and a concept $c \in \mathcal{C}$), some unknown distribution $\mathcal{D}$ over $\mathcal{U}$, and constants ε, δ. We are given m samples $(x_1, c(x_1)), ..., (x_m, c(x_m))$, where $x_1, ..., x_m$ are drawn independently from $\mathcal{D}$. Our goal is to find some hypothesis $h \in \mathcal{C}$ such that

$$\mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq c(x)] \leq \varepsilon$$

Furthermore, we allow for our procedure to output a "bad" hypothesis (that is, one that doesn't satisfy the above condition), so long as this happens with probability at most δ (here, the randomness comes from the sampled training data and, possibly, the randomness in the algorithm).

We may wish to quantify how many samples we need to achieve this.

Definition 29. *A concept class $\mathcal{C}$ is (m, ε, δ)-PAC learnable if there exists an algorithm $\mathcal{A}$ such that for every $c \in \mathcal{C}$ and every distribution $\mathcal{D}$, $\mathcal{A}$ receives m samples $(x_1, c(x)), ..., (x_k, c(x_k))$ (where $x_1, ..., x_k$ are sampled from $\mathcal{D}$) and, with probability at least $1 - \delta$ outputs h such that*

$$\mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq c(x)] \leq \varepsilon$$

We have the following theorem, also due to Valiant.

Theorem 30. *If $\mathcal{C}$ is finite then it is (m, ε, δ)-PAC learnable, where*

$$m = \frac{1}{\varepsilon} \log \left(\frac{|\mathcal{C}|}{\delta} \right)$$

Proof. Our algorithm will maintain a set of potential hypotheses $h \in C$ (initially starting with the entire set), eliminating any that are inconsistent with any of the m samples. Then we output any remaining hypothesis in the set. Call a hypothesis h bad if $\mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq c(x)] > \varepsilon$, and good otherwise. Clearly there's at least one good hypothesis, namely c itself. Thus our algorithm only

fails when a bad hypothesis survives. Any particular bad hypothesis survives with probability at most $(1-\varepsilon)^m$, so by the union bound, the probability that at least one bad hypothesis survives is at most $|\mathcal{C}|(1-\varepsilon)^m$. Setting $\delta = |\mathcal{C}|(1-\varepsilon)^m$ and then solving for m now yields the theorem. □

Of course, there are other considerations with PAC-learning. For instance, we might be interested in computational complexity of learning: just because a small number m of samples suffices, doesn't mean that there's an efficient algorithm to *find* a hypothesis h consistent with those samples.

The fact that $|\mathcal{C}|$ appears in this expression, however, begs a different question: what if the concept class that we wish to learn is very large or even infinite? Some very natural concept classes (such as hyperplanes in $\mathbb{R}^n$) fall under this category. In the next section, we discuss the VC-dimension, which serves to characterize the sample complexity even of infinite concept classes.

21.3.2 VC-dimension

In this section, we define the VC-dimension (short for Vapnik-Chervonenkis dimension), which in some sense characterizes the difficulty of learning a (potentially infinite) concept class.

Definition 31. *We say a concept class $\mathcal{C}$ of functions from $\mathcal{U}$ to $\{0,1\}$* **shatters** *a set $\{x_1, ..., x_k\} \subseteq \mathcal{U}$ if, for every possible assignment to $c(x_1), ..., c(x_k)$ (of which there are 2^k) there exists some $c \in \mathcal{C}$ that is consistent with that assignment.*

Then the **VC-dimension** *of a concept class $\mathcal{C}$, denoted* VCDIM($\mathcal{C}$)*, is the size of the largest set shattered by $\mathcal{C}$ (or ∞ if there is no largest set).*

As an example, let's consider the concept class $\mathcal{C}$ of halfspaces in $\mathbb{R}^2$. We claim that VCDIM($\mathcal{C}$) ≥ 3. Consider an equilateral triangle in $\mathbb{R}^2$. If the value assignments of all points are equal, we trivially have a hyperplane that matches the data (take a horizontal line above all the points). Now suppose, without loss of generality, that two of the points are labeled 1 and one is labeled 0. We can draw a line that bisects the sides of the $0-1$ edges in the triangle, which gives a separating hyperplane (see Figure 21.3). Note that $\mathcal{C}$ cannot shatter a set of 3 colinear points, but as long as $\mathcal{C}$ can shatter one set of size 3, we have that VCDIM($\mathcal{C}$) ≥ 3.

Now we argue at a high level that for any set of 4 points in $\mathbb{R}^2$, there is an assignment such that no hyperplane separates them correctly. Number the points in order of increasing counterclockwise angle from the origin, breaking ties by highest x coordinate. In order, label the points 0/1 in alternating order (say $(x_1, y_1), (x_3, y_3)$ are labeled 0 and (x_2, y_2), (x_4, y_4) are labeled 1); see Figure 21.4 for an example of this labeling. It is not hard to show that any hyperplane $\theta x + \phi y = c$ that has $\theta x_1 + \phi y_1 - c \leq 0$ and $\theta x_3 + \phi y_3 - c \leq 0$ must have one of $\theta x_2 + \phi y_2 - c \leq 0$ or $\theta x_4 + \phi y_4 - c \leq 0$, violating the assumption that $\mathcal{C}$ shatters the set.

We now give a theorem that relates the sample complexity of PAC-learning a concept class $\mathcal{C}$ to its VC-dimension:

Theorem 32 ([20])**.** *Every concept class $\mathcal{C}$ is (m, ε, δ)-PAC learnable, where*

$$m = O\left(\frac{\text{VCDIM}(\mathcal{C})}{\varepsilon} \log\left(\frac{1}{\delta\varepsilon}\right)\right)$$

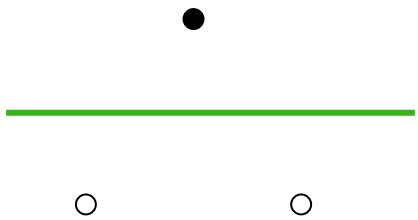


Figure 21.3: The class of hyperplanes in $\mathbb{R}^2$ shatters at least one set of size 3. As before, the white points are labeled 1 and the black points are labeled 0.

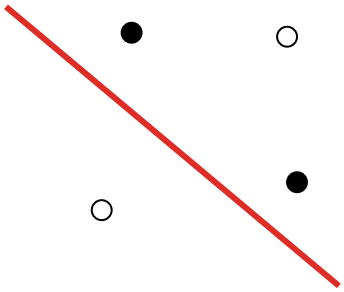


Figure 21.4: Given any set of 4 points, we can label them in such a way that any line that classifies both black points correctly, must classify at least one white point incorrectly.

We omit the proof. The high-level intuition, however, comes from the relation between compressibility and learnability: if a concept class shatters sets of high dimension, we need correspondingly many samples to separate potential hypotheses. This theorem allows us to consider the learnability of infinite concept classes. Furthermore, it is, in some sense, optimal, as no concept class of infinite VC-dimension is PAC-learnable in any finite number of queries.

In the next lecture, we will look at PAC-learning of quantum states, and use a generalization of VC-dimension to real-valued functions (called "fat-shattering-dimension") to argue about a "pretty good" version of quantum state tomography that requires exponentially fewer samples.

Lecture 22: Probably Approximately Almost Correct Quantum Tomography

22.1 Overview

In the last lecture we discussed quantum state tomography and the inherent drawback that it requires an amount of data that's exponential in the number of qubits. We derived that at least $\sim D^2$ copies of the state ρ are needed, where $D = 2^n$ is the Hilbert space dimension. This could even lead to the philosophical question of whether quantum mechanics is useful for describing physical systems, if the learning of a state could theoretically take longer than the lifetime of the universe itself. We have, however, learned of various ways around this:

- *Prior knowledge* – If the experimentalist has some idea about what the state should be, that guess can be verified in a constant number of queries as they could perform a swap test, or project onto the guess state.
- *Tomography of restricted families of states* – For example, stabilizer, fermionic states, or states of low bond dimension which have classically efficient simulations.
- *Relaxed Tomography (PAC Learning)* – Learning relevant details about the state which are useful for most purposes without hoping to recover the full density matrix, or even something close to it.

We began talking last time about Valiant's Probably Approximately Correct Learning framework[70], which has been the basic theoretical framework for machine learning for decades. Today we will finish the discussion of PAC learning and how to use it effectively in Quantum State Tomography.

22.2 Valiant's PAC Framework

The basic setup is that we have some class $\mathcal{C}$ of functions $c : \mathcal{U} \longrightarrow \{0, 1\}$ mapping some universal set of points to $\{0, 1\}$. An example being linear half-spaces, where $\mathcal{U} = \mathbb{R}^2$ is the plane and the functions f all have the form

$$f(x, y) = \begin{cases} 1 & \text{if } ax + by \geq c \\ 0 & \text{otherwise} \end{cases}$$

for some real constants a, b, c.

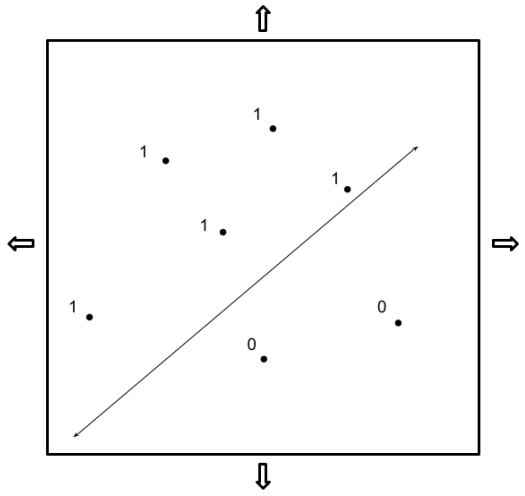


Figure 22.1: Example of Halfspace function

As the learner, we hope to discover the unknown function $f : \mathcal{U} \longrightarrow \{0, 1\}$ or something close to it. Importantly, we require that all the samples are taken independently from the same distribution $\mathcal{D}$ on which the hypothesis will be evaluated. While this requirement can be relaxed, if there are no assumptions on the distribution from which samples can be drawn, it becomes impossible to learn. In practical machine learning, the issue where a model tries to predict some behavior that it hasn't been previously trained on, but there are no samples to base a prediction on, is known as being **Out of Distribution**. So we assume the training data comes from $\mathcal{D}$ and evaluate the algorithms based on future samples also coming from $\mathcal{D}$.

More formally, we are given a set of samples

$$x_1, x_2, \ldots x_m \sim \mathcal{D}$$

and the classification of each of these points using the function $c \in \mathcal{C}$:

$$c(x_1), c(x_2), \ldots c(x_m)$$

The goal is to find a hypothesis $h \in \mathcal{C}$ that is ε-close to c with probability at least $1 - \delta$ over $x_1, x_2, \ldots x_m$:

$$\Pr_{x \sim \mathcal{D}}[h(x) = c(x)] \geq 1 - \varepsilon \quad \text{w.p. } 1 - \delta \tag{22.1}$$

22.2.1 First Theorem

The first theorem we saw is almost trivial, though still very important. It states that the necessary number of samples m is at most

$$\frac{1}{\varepsilon} \log \frac{|\mathcal{C}|}{\delta}.$$

The strategy you can use is *any* strategy that obtains an h matching c for all the training data given. If you can obtain such an h with the above number of samples, you will have a hypothesis such that equation (22.1) holds, achieving generalization. This theorem gives us a practical bound, even with exponentially many hypotheses (due to the log factor), but this falls apart when we encounter hypothesis spaces that are infinite (which is often the case in practice). In these cases the correct concept for measuring the complexity of the hypothesis class is the **VC dimension**, which we defined last time using shattering.

22.2.2 VCdim & Shattering

We say $\mathcal{C}$ shatters the set $\{x_1, x_2, \dots x_m\} \subseteq \mathcal{U}$, if all 2^k possibilities for $c(x_1), c(x_2), \dots c(x_m)$ are consistent with *some* $c \in \mathcal{C}$. In other words, being in $\mathcal{C}$ puts no constraints on the possible values of the $c(x_i)$'s. The **VC dimension** of $\mathcal{C}$, $\text{VCdim}(\mathcal{C})$, is then the size of the largest set shattered by $\mathcal{C}$. As an example, the class of linear halfspaces discussed previously has a VCdim of 3.

22.2.3 Blumer et al.

Blumer et al [20] showed that, in general,

$$m = O\left(\frac{\text{VCdim}(\mathcal{C})}{\varepsilon} \log \frac{1}{\delta\varepsilon}\right)$$

samples suffice. This linear dependence on the VCdim is tight, leaving only possible optimizations to the dependence on ε and δ. The theorem often gives a useful bound even with an infinite hypothesis space. On the other hand, it can also happen that $\text{VCdim}(\mathcal{C}) = \infty$. A trivial example is given by the class of *all functions on an infinite universe*, or more concretely *all polynomials with unbounded degree*. In [20] the authors interpreted the above bound as a connection between the compression of data from the past and the predictability of the future. Informally, if the number of samples, m, is too much less than the VCdim of what you're attempting to learn, then as far as you know, you could have fit any data to the hypothesis. But if any data could be fit to the hypothesis, the hypothesis becomes useless in making any relevant predictions. For example, given any three points you can fit some quadratic function to them, but that gives no guarantee that the function predicts over any other points. If you could fit a quadratic to 10 or 20 points, however, *that* might be useful.

This intuition had been the reigning wisdom of machine learning, but has been complicated by the ability of deep learning models to predict outcomes even though they have many, *many* more parameters than the number of points in the training set. This means that the above theorems, despite being true for all concept classes $\mathcal{C}$ (including deep learning models), fail to *explain the success* of deep learning models. One of the great tasks of this era in machine learning is to explain the success of deep learning, mathematically.

22.3 Applying the Framework to Pretty Good Quantum Tomography

In the 2007 paper by Scott Aaronson [3], we consider a quantum (mixed) state ρ of n qubits, with the dimension, D, of the Hilbert space being 2^n. We let $\mathcal{D}$ be some unknown distribution over two-outcome measurements; each measurement can be thought of as a circuit with no restrictions on complexity, where we measure some qubit in the $\{|0\rangle, |1\rangle\}$ basis, giving us two possible outcomes.
Useful Quantum Information Fact: For every such circuit there exists some Hermitian matrix E such that

$$\Pr[\rho \text{ is accepted}] = \text{Trace}(E\rho)$$

22.3.1 Following Valiant's Framework

Given: $E_1, E_2, \ldots E_i, \ldots E_m \sim \mathcal{D}$ and $\text{Tr}(E_i\rho)$ for all i
Immediately we notice that $\text{Tr}(E_i\rho)$ is not a Boolean function as in the original Valiant's framework. We can also see that we would not be given the exact real number $\text{Tr}(E_i\rho)$, but an estimate $\text{Tr}(E_i\rho) \pm \gamma$ from multiple experimental measurements.

What we want: A hypothesis state σ such that, when the measurement E is drawn from the distribution $\mathcal{D}$, there's a high probability $(1-\delta)$ that σ is γ-close to ρ:

$$\Pr_{E\sim\mathcal{D}}[|\text{Tr}(E\sigma) - \text{Tr}(E\rho)| > 2\gamma] \leq \varepsilon$$

Moreover, we'd like this to be true with probability at least $1-\delta$ over the choice of training measurements $E_1, E_2, \ldots E_i, \ldots E_m$. We'd ideally hope for ρ but as we've seen this would require data exponential in n. σ could be far from ρ in trace distance, as long as it approximately mimics ρ's behavior over the distribution $\mathcal{D}$.

22.3.2 Upper Bound on Number of Samples m

By combining results from learning theory and quantum information theory, Aaronson [3] proved the following upper bound on the number of samples m that suffice for this sort of learning:

$$m = O\left(\frac{1}{\gamma^2\delta^2}\left(\frac{n}{\gamma^2\delta^2}\log^2(\frac{1}{\gamma\delta}) + \log(\frac{1}{\delta})\right)\right) \tag{22.2}$$

The most important part here is the *linear dependence* on the number of qubits in the state.

The strategy to achieve this: Simply find any σ such that $\text{Tr}(E_i\sigma) \approx \text{Tr}(E_i\rho)$ for all i. At least one such state must exist, namely ρ itself.

We can also consider the computation time for finding such a σ. Given that the hypothesis is a $2^n \times 2^n$ matrix, we couldn't hope to have computation time less than 2^n in general. There are other barriers to polynomial-time computation: the size of the inputs (describing all E_i) could be exponential in n. Also, even in classical learning theory, with polynomial-size training data and hypotheses, learning can be cryptographically hard — e.g. breaking pseudorandom generators.

Could the complexity be exponential in D?

Fortunately not. σ is a positive semi-definite, Hermitian matrix with $(\mathrm{Tr}(\sigma) = 1)$ and it should satisfy the constraints of $\mathrm{Tr}(E_i\sigma) \approx \mathrm{Tr}(E_i\rho)$ for all i. There is a name for the computational problem of finding such a matrix: **SDP** or **Semi-Definite Programming**. It is known that **SDP** is solvable in polynomial in D time by a non-trivial algorithm.

Returning to the question of how to prove upper bound on sample complexity, one of the first complications that arises, is that $\mathrm{Tr}(E_i\rho)$ is a real-valued function as opposed to the Boolean functions in Valiant's framework. To deal with this, we'll need a generalization of VC dimension called **fat-shattering** dimension.

22.3.3 Fat-shattering

Definition: Let $\mathcal{C}$ be a class of functions mapping points $x \in \mathcal{U}$ to real values $c(x) \in [0, 1]$. We say that $\mathcal{C}$ **fat-shatters** a set $\{x_1, x_2, \ldots x_k\} \subseteq \mathcal{U}$ if for some $(\alpha_1, \alpha_2, \ldots \alpha_k) \in [0, 1]^k$, all 2^k possibilities defined as: $c(x_1) \geq \alpha_1 + \varepsilon$ **or** $c(x_1) \leq \alpha_1 - \varepsilon$; $c(x_2) \geq \alpha_2 + \varepsilon$ **or** $c(x_2) \leq \alpha_2 - \varepsilon$; ... are consistent with some $c \in \mathcal{C}$.

The ε-fat-shattering dimension of $\mathcal{C}$, or $\mathrm{fat}_\varepsilon(\mathcal{C})$, is then defined as the size of the largest ε-fat-shattered set.

This turns out to be the correct generalization of VCdim, which allows us to state the real-valued generalization of Blumer et al. Namely, suppose we are looking for a hypothesis $h \in \mathcal{C}$ such that

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}} [|h(x) - c(x)|] \leq \varepsilon$$

with high probability, at least $1 - \delta$. This is just one convenient notion chosen to represent how closely one real-valued function approximates another; others could be the 2-norm, the max-norm, and so on, which can be converted to each other with some loss.

Bartlett and Long [19] prove that

$$m = O\left(\frac{1}{\varepsilon^2}\left(\mathrm{fat}_{\varepsilon/5}(\mathcal{C}) \log^2 \frac{1}{\varepsilon} + \log \frac{1}{\delta}\right)\right)$$

samples suffice with probability $\geq 1-\delta$. This bound is polynomial in $\frac{1}{\varepsilon}$, logarithmic in $\frac{1}{\delta}$, and linear in the fat-shattering dimension. All that's left to prove the approximate quantum tomography theorem seen in equation 22.2, is to determine the fat-shattering dimension of quantum states considered as a hypothesis class.

22.3.4 Fat-shattering dimension of quantum states as a hypothesis class

Given a Hilbert space dimension D, let Q_D consist of all functions of the form $f_\rho(E) = \mathrm{Tr}(E\rho)$. These take as input a 2-outcome measurement and output a probability. There could be an enormous number of such functions that correspond to no quantum state. With no constraints, the fat-shattering dimension would be ∞. However, if we believe in quantum theory, then we have a constraint we can place on the class of functions, namely that there exists some density matrix ρ such that $f_\rho(E)$ arises as $\mathrm{Tr}(E\rho)$.

We want: To calculate $\mathbf{fat}_\varepsilon(Q_D)$.
Claim:

$$\mathbf{fat}_\varepsilon(Q_D) = \Theta\left(\frac{\log(D)}{\varepsilon^2}\right)$$

The key point of this is the logarithmic dependence on D, giving a linear dependence on the number of qubits n.

Why should the fat-shattering dimension be linear in n?

Recall that the fat-shattering dimension is the size of the largest set shattered. Suppose $\mathbf{fat}_\varepsilon(Q_D) = k$. This would mean that there exist k measurements $E_1, E_2, \ldots E_k$ such that all 2^k possibilities (meaning larger than some value plus ε or smaller than that value minus ε) for $\mathrm{Tr}(E_1\rho)$, $\mathrm{Tr}(E_2\rho)$, $\ldots \mathrm{Tr}(E_k\rho)$ are consistent with some density matrix ρ.

Suppose k is much larger than n. This would reveal itself as a quantum protocol in which one could 'stuff' $k \gg n$ bits of data into a much smaller number of qubits, such that someone could then measure the qubits and recover any one of the k bits of their choice, with large success probability.

This seems in direct tension with Holevo's Theorem! To finish the proof, though, we need a generalization of Holevo's Theorem due to Ashwin Nayak [51].

Nayak proved the following: If we want any protocol where Alice encodes Bob k bits of information into a state ρ, such that Bob can retrieve any one bit of his choice by measuring, then Alice must send $\Omega(k)$ qubits (assuming one-way communication only). In more detail, we can derive that if we want to encode data using n qubits, and have the ability to recover any of the k bits of information with bias at most ε, we require the number of bits to be

$$k = O\left(\frac{n}{\varepsilon^2}\right).$$

22.3.5 Lemons To Lemonade

We can take the theorems proved by Nayak [51], restricting the amount of data compressible into qubits (lemons), and turn them into an upper bound on the fat-shattering dimension of the class of quantum states. We find that if the fat-shattering dimension were any greater than $\frac{n}{\varepsilon^2}$, it would imply a violation of Nayak's bound. We then combine this with classical learning theory to get an algorithm for learning quantum states using a number of samples that's logarithmic in the Hilbert space dimension D (lemonade!).

Lecture 23: Finishing Tomography and Introducing Black Holes

23.1 Overview

In the last lecture we discussed the work of [3], on how to "learn" a general quantum state in the sense of Valiant's PAC model [70].

Instead of trying to learn the full density matrix ρ of an unknown state, the goal was shifted to learning ρ's behavior on most two-outcome measurements E drawn from an unknown distribution $\mathcal{D}$. This was done using a training set of measurements $E_1, \ldots, E_m$ drawn independently from $\mathcal{D}$, along with approximate values of $\mathrm{Tr}(E_i\rho)$ for each $i = 1, \ldots, m$. The main result was that in order to estimate $\mathrm{Tr}(E\rho)$ for most Es drawn from $\mathcal{D}$, one only needs the number of samples m to be linear in the number n of qubits of ρ. This gives an exponential improvement over full quantum state tomography, which requires at least $\approx 4^n$ samples.

In this lecture we introduce another framework for learning quantum states, called *Shadow Tomography*, as well as a computationally efficient method for learning stabilizer states. We'll end with a cliffhanger about the apparent problem quantum mechanics has describing black holes.

23.2 Shadow Tomography

23.2.1 The Problem

The name Shadow Tomography refers to how, rather than trying to learn the underlying density matrix ρ of some quantum state, we could instead to recover the "shadow" that ρ casts on some known two-outcome measurements $E_1, \ldots, E_M$ — namely, the acceptance probabilities $\mathrm{Tr}(E_1\rho), \ldots, \mathrm{Tr}(E_M\rho)$. The problem was first posed by Aaronson in 2016 [5].

Problem (Shadow Tomography). *Given an unknown D-dimensional quantum mixed state ρ, as well as known 2-outcome measurements $E_1, \ldots, E_m$, each of which accepts ρ with probability $Tr(E_i\rho)$ and rejects ρ with probability $1 - Tr(E_i\rho)$, output numbers $b_1, \ldots, b_M \in [0, 1]$ such that $|b_i - Tr(E_i\rho)| \leq \epsilon$ for all i, with success probability at least $1 - \delta$. Do this via a measurement of $\rho^{\otimes k}$, where $k = k(D, M, \epsilon, \delta)$ is as small as possible.*

Clearly it is possible to use full quantum state tomography to recover ρ, which would require

$k = O(D^2/\epsilon^2)$ copies of ρ. One could also solve the problem using $k = \tilde{O}(M/\epsilon^2)$ copies of ρ by simply applying each measurement E_i to many separate copies of ρ to get a good estimate b_i of the probability that E_i accepts ρ. However, neither of these options are appealing if D and M are both exponentially large.

23.2.2 Solutions

But it turns out to be possible to do better than the naïve algorithms above. The first published algorithm with better bounds was due to Aaronson [6] which used $k = \tilde{O}\left(\frac{\log^4 M \log D}{\epsilon^4}\right)$ copies of ρ. After a talk Aaronson was approached by Rothblum, who works in differential privacy, and together they came up with a procedure which solves the Shadow Tomography problem using $k = \tilde{O}\left(\frac{\log^2 M \log^2 D}{\epsilon^8}\right)$ copies of ρ — an improvement if M is much larger than D — by integrating known techniques in differential privacy [13].

The current best known Shadow Tomography algorithm is due to Badescu and O'Donnell; it uses $k = \tilde{O}\left(\frac{\log^2 M \log D}{\epsilon^4}\right)$ copies of ρ [18].

23.2.3 Potential Issues

A question one might ask about Shadow Tomography is, what if the observables $E_1, \ldots, E_m$ are chosen to be tomographically complete? In this case, estimating each $\mathrm{Tr}(E_i\rho)$ would allow you to completely reconstruct ρ, and in doing so it would seemingly use fewer samples than are required by Holevo's theorem. The catch is that, in order to reconstruct ρ the estimations b_i would need to be very accurate, requiring $\epsilon \ll 1/D$, causing the sample complexity to become polynomial in D.

As a separate issue, while this work has given impressive bounds on the sample complexity of quantum tomography, it says nothing about the computational complexity of the task, which is just as important in practice. In fact, the computational complexity of these algorithms is generally quite bad — polynomial in D and thus exponential in the number of qubits n. This raises the question of whether there is some subset of quantum states and measurements for which Shadow Tomography is computationally efficient as well.

23.3 Learning Stabilizer States

Restricting quantum computation to stabilizer circuits allows for speedups in classical simulation, so stabilizer states would seem to be a good candidate for computationally efficient learning. Rocchetto [62] gave a polynomial-time algorithm to PAC-learn stabilizers states, with respect to any distribution $\mathcal{D}$ over 2-outcome stabilizer measurements, in the original sense of Aaronson's 2006 paper [3]. But we can also consider a conceptually simpler problem: exactly learning an unknown n-qubit stabilizer state $|\psi\rangle$, given arbitrary n-qubit measurements on copies of $|\psi\rangle$. Here is a solution to that problem due to Aaronson.

Let $|\psi\rangle$ be an n-qubit stabilizer state and start with k copies, $|\psi\rangle^{\otimes k}$. In contrast to the previ-

ous methods, our goal will be to determine **exactly** what $|\psi\rangle$ is. The Gottesman-Knill theorem says that $|\psi\rangle$ is uniquely determined by its stabilizer group, so to learn $|\psi\rangle$ it suffices to learn $\text{Stab}(|\psi\rangle)$.

To start, we could repeatedly measure all n qubits of $|\psi\rangle$ in the 0/1 basis, each time noting the result. We know that the span of the X-matrix of $|\psi\rangle$ gives all possible outcomes of such a measurement, so if a measurement falls outside of the span of our previous measurements, we learn a new row of the X matrix. After a certain number of measurements, none of which give a new row of X, we can be confident that the span of X, and thus X itself up to row operations, has been found. We could then repeat the process but measure in the $+/-$ basis to get the Z-matrix up to row operations. The trouble is that it would remain unclear how the rows of X and Z line up.

To fix this, instead of just measuring in these two bases, consider measuring in a *random* stabilizer basis. In other words, let C be a uniformly random stabilizer circuit; then repeatedly measure $C|\psi\rangle$ in the standard basis. As long as the X-matrix generated by this process is not full-rank, a linear equation of $|\psi\rangle$ is learned which yields an element of $\text{Stab}(|\psi\rangle)$. Continue doing this with random stabilizer circuits C until the stabilizer group is learned, at which point $|\psi\rangle$ has been learned.

The only problem with the above algorithm is that it fails when $C|\psi\rangle$ has a full-rank X-matrix. However, because $|\psi\rangle$ is a stabilizer state and C is a uniformly random stabilizer circuit, $C|\psi\rangle$ is a uniformly random stabilizer state, regardless of the distribution over $|\psi\rangle$. Therefore the probability that $C|\psi\rangle$ has a full-rank X-matrix is the probability that a random stabilizer state has a full-rank X-matrix, which can be calculated with relative ease by counting.

First recall that the total number of n-qubit stabilizer tableaus is

$$\#\text{Total} = (4^n - 1)\left(\frac{4^n}{2} - 2\right)\left(\frac{4^n}{4} - 4\right)\cdots\left(\frac{4^n}{2^{n-1}} - 2^{n-1}\right)$$

(This is greater than the number of stabilizer states, since we're counting tableaus as distinct even if they're equivalent under row operations.) Meanwhile, the total number of tableaus with full-rank X-matrix is

$$\#\text{Full Rank} = (4^n - 2^n)\left(\frac{4^n}{2} - 2^n\right)\left(\frac{4^n}{4} - 2^n\right)\cdots\left(\frac{4^n}{2^{n-1}} - 2^n\right)$$

To see this, note that the first row can't start with n 0s. Then after picking the first row, the second row must commute with and be linearly independent from the first row. Being linearly independent from the first row will multiply the 2^n excluded choices by 2, and then requiring that it commutes divides by 2, so there are $\frac{4^n}{2} - 2^n$ choices for the second row. And so on. Taking the ratio of these two values and taking the limit as $n \to \infty$ will give

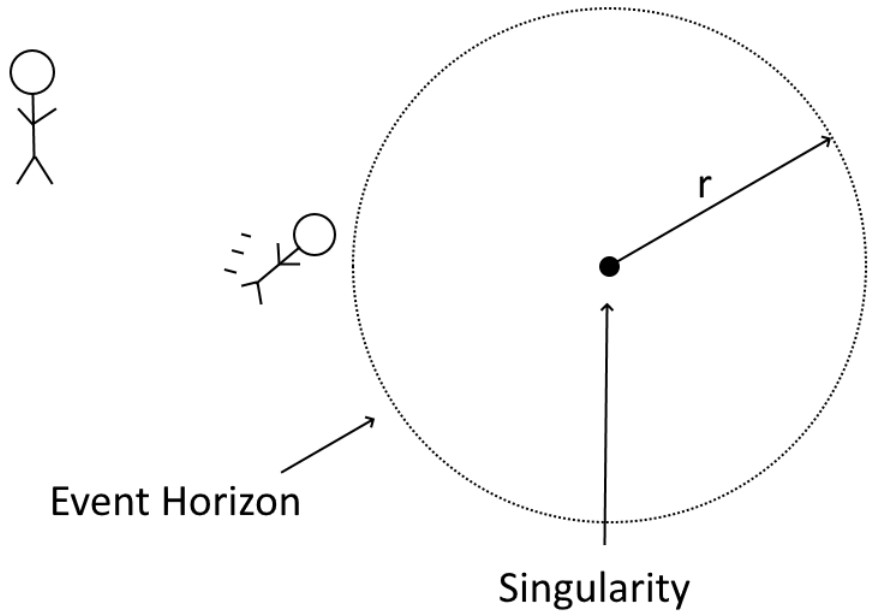
$$\mathbb{P}(\text{fail}) = \lim_{n\to\infty} \frac{\#\text{Full Rank}}{\#\text{Total}} \approx 0.42$$

So the probability of success is $\approx 58\%$ which is bounded away from 0. Therefore, after $O(1)$ attempts, with high probability a new element of $\text{Stab}(|\psi\rangle)$ is learned.

This procedure uses $O(n)$ copies of $|\psi\rangle$ per generator learned, so it requires $O(n^2)$ copies of $|\psi\rangle$ total or $O(n^3)$ qubits as $|\psi\rangle$ is an n-qubit state. Furthermore the only computation is n-dimensional linear algebra so the algorithm is also computationally efficient.

23.4 Black Holes

When a star collapses, if it is massive enough it will become so dense that, once close enough, nothing can escape it, not even light. It becomes what's known as a black hole.



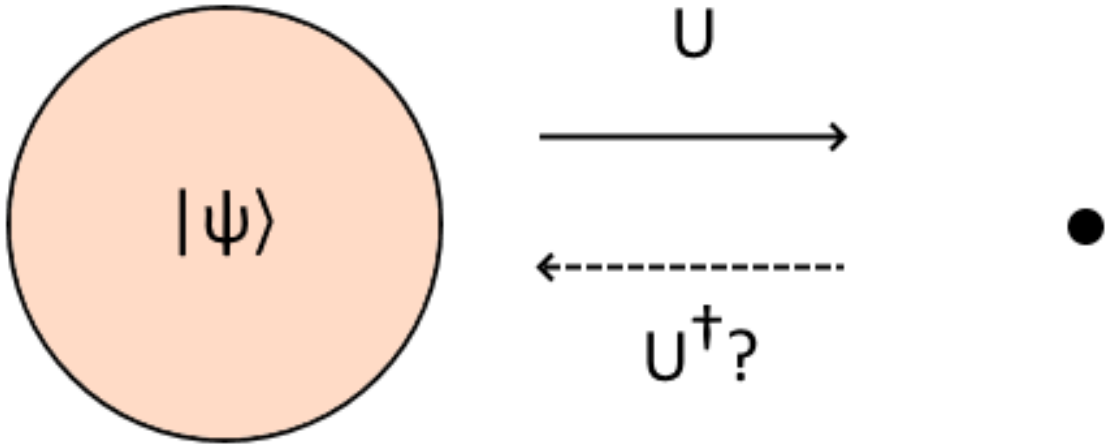
The point of no return is known as the *event horizon*. Anything which passes it will never escape, including any unfortunate observers who happen to fall in. However, other than this terrifying fact, there is nothing particularly special about the event horizon — it's just a region in space like any other. If they haven't been killed yet by the immense gravity, the observer might not notice that they had passed the event horizon. Any external observer, though, would never see anything pass the event horizon, as light cannot escape the black hole. Rather the object crossing the event horizon would appear to slow to a standstill as it got closer to the event horizon because light is affected by the gravitational pull.

23.4.1 Hawking's Information Problem

A famous question, posed by Hawking in the 1970s, asks what happens to all the information that's eaten by a black hole. Since no information can be retrieved from a black hole, it seems like any that passes the event horizon would just be destroyed, but that conflicts with the usual understanding of quantum mechanics, according to which all evolution (with the possible exception of measurement) must be unitary, and hence reversible. Directly related is that general relativity breaks down when trying to describe black holes, presumably to be replaced by a quantum theory of gravity. However any such theory would need to grapple with Hawking's information problem.

Say at a macroscopic scale a star has quantum state $|\psi\rangle$, then collapses and forms a black hole. If all transformations are unitary, then there must have been some unitary U that describes such a scenario. However, if no information can come out of a black hole, then there could be no inverse

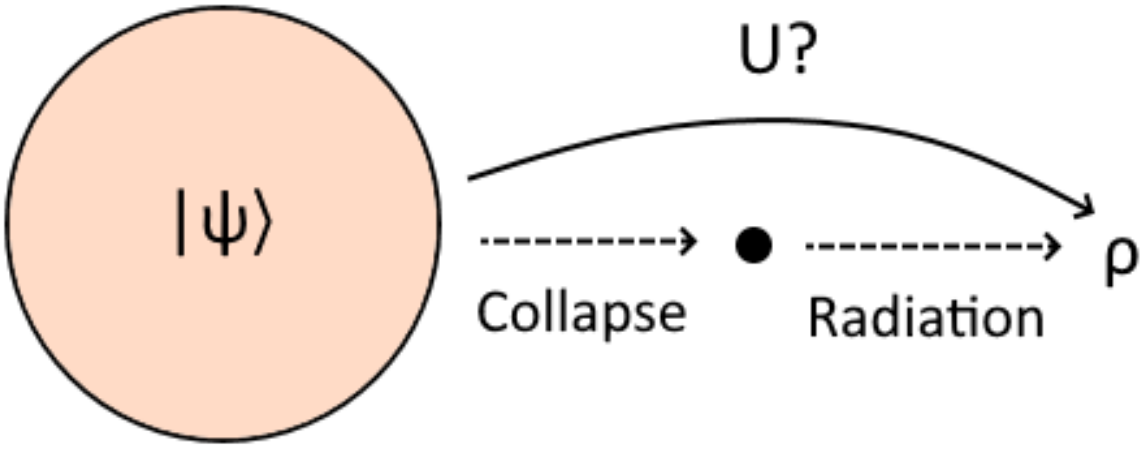
$U^\dagger$, which contradicts unitarity.



One possibility is that the state $|\psi\rangle$ somehow goes into a "baby universe" created inside the black hole; however this idea fell out of favor due to technical problems. Hawking made a decisive advance when he applied quantum field theory to the region just outside the event horizon, which led to the prediction that the black hole would emit radiation, dubbed Hawking radiation. As the energy has to come from somewhere, the theory predicted that the black hole would eventually evaporate. Just to give you a sense of scale, letting r be the radius of the event horizion, we have:

Mass of a black hole $\propto r$	
Entropy $\propto r^2$	($\approx 10^{69}$ bits/m^2)
Evaporation time $\propto r^3$	($\approx 10^{67}$-10^{100} years for astrophysical black holes)

Many hoped that perhaps the initial state $|\psi\rangle$ would somehow leak out in some scrambled form in the radiation. However, Hawking's calculation predicted that the radiation would be thermal. In this case, all that could be recovered from it is a mixed state ρ, which is impossible to get by unitarily transforming a pure state.



What can explain this?

Lecture 24: Black Holes and the Firewall Paradox

24.1 Overview

In the last lecture we discussed shadow tomography: learning the behavior of an unknown quantum state ρ on a long list of two-outcome measurements. Surprisingly, we can learn the behavior of an arbitrary n-qubit state on all circuits of some fixed polynomial-size (say, n^2) that either reject or accept the state, using only a polynomial number of copies of the state. In this case, we are not fully reconstructing the state only because we do not learn its behavior on measurements with huge circuits.

We also discussed how you can fully reconstruct an n-qubit stabilizer state given a bunch of copies of it. This can be done by applying random stabilizer circuits to various copies of the state and measuring in the computational basis, thereby learning more and more generators of the stabilizer group. This deviates from the PAC-learning framework of quantum states in several ways: we let ourselves choose the measurements, the measurements were not two-outcome (2^n outcomes instead), and we learn the state exactly (given that it is a stabilizer state of course).

Last lecture, we also briefly touched on black holes; in this lecture we will be expanding on them and focusing on the firewall paradox and Harlow and Hayden's proposed resolution of the paradox based on quantum computational complexity [39].

24.2 Black Holes

Black holes slowly evaporate and radiate photons, known as *Hawking radiation*; an observer outside a black hole would see this. This is demonstrated using quantum field theory which describes the region just outside the event horizon. The evaporation time is proportional to r^3 where r is the *Schwarzschild radius* or the radius of the event horizon. This is perhaps the slowest process known in physics. After the rest of the universe dies out, black holes will still be evaporating for more than a googol years.

The existence of Hawking radiation lets us pose the problem of information loss: what happens to information dropped into a black hole? Suppose we have a large star which we can describe with some pure state $|\psi\rangle$. The star then collapses to a black hole and we wait as the black hole evaporates into Hawking radiation. Imagine that we are to collect all the Hawking radiation, put it into a quantum computer, and look at its quantum state. The problem is that the quantum field

theory that predicted the existence of this Hawking radiation describes it as being in a thermal mixed state ρ, however this directly breaks unitarity in quantum mechanics. There is no unitary that transforms a pure state into a mixed state.

One could ask if ρ is a result of measurement, but then where did the measurement happen and what decohered the state? One could also ask whether information is a physical quantity. However, the problem of information loss doesn't actually depend on knowing what information is. In principle, this is an empirical question: do you end up with a pure state or not? It may take over a googol years to complete empirical tests, but it is still an empirical question. You can think of information as a physical quantity or non-physical quantity, but the question of whether or not you can reconstruct the initial state from Hawking radiation still stands.

This question has major consequences for fundamental questions in physics. If the answer is no, then it means the laws of quantum mechanics are not universally true as they would not hold for black holes (or at least the Schrödinger equation breaks down).

Quantum theories of gravity try to bridge this breakdown in physics. When Hawking calculated black hole evaporation, he only used quantum field theory and did not treat spacetime itself as quantum mechanical.

Thus, we can imagine that the final mixed state ρ is really just an artifact of ignoring quantum gravitational degrees of freedom. As an example of this, suppose we burn an encyclopedia. In principle, through time-reversible laws of microscopic physics, all of the information from the encyclopedia is still in the smoke and ash and thus is recoverable. But in practice, we have no technology that could actually recover the information from the burned encyclopedia. Hence, we would have to describe the state of the smoke and ash with some kind of thermal description which would correspond to a mixed state, something analogous to ρ. So, the here is that black hole evaporation is just a "fancier" version of burning an encyclopedia. The information is in degrees of freedom that existing physics does not account for; hopefully a future theory of physics will account for it.

24.3 Xeroxing Paradox

In the 80s, people began talking about the following paradox. Imagine we have a black hole and outside the black hole is Alice. Bob, being not as wise as Alice, jumps into the black hole past the event horizon. Now imagine that a qubit $|\psi\rangle$ gets thrown into the black hole as well. If black hole evaporation truly is unitary then from Alice's perspective, this qubit should eventually come out of the black hole, maybe in some extremely scrambled form, but still in the quantum state of the Hawking radiation. By waiting a long enough time, we should be able to collect all the Hawking radiation emanating from the black hole and recover the quantum state $|\psi\rangle$ by processing the radiation in a quantum computer. This is in direct contrast with what Bob sees. He sees a version of reality where $|\psi\rangle$ is going in towards the singularity, with no possibility of it ever escaping the event horizon. So who is right? Alice or Bob? If they are both right, then it would appear as if $|\psi\rangle$ has been cloned, which blatantly violates the no-cloning theorem.

In the 90s, Susskind and 't Hooft proposed black hole complementarity [66]. The idea behind this is that both Alice and Bob are right, but only as long as no one combines their two perspectives. As long as neither of them see the qubit in both situations, there is no actual manifestation of cloning a quantum state.

Now imagine a new scenario where Alice again throws $|\psi\rangle$ into the black hole. She waits however long it takes for $|\psi\rangle$ to be recoverable from the Hawking radiation (in practice, as we will see, typically about half of the black hole's evaporation time). Next, Alice jumps into the black hole and looks for the copy of $|\psi\rangle$. Alice should be able to see both copies of $|\psi\rangle$, which then would manifest a violation of the no-cloning theorem. Fortunately, this turned out to not actually be an issue. Because it takes so long for $|\psi\rangle$ to radiate out of the black hole, the infalling copy of $|\psi\rangle$ should have hit the singularity long before Alice collects all the Hawking radiation. Hence, when Alice jumps in she cannot actually find the other copy. This is the "complementarity" philosophy applied to black holes.

24.4 Firewall Paradox

In 2012, Almheiri, Marolf, Polchinski, and Sully (AMPS) proposed the *firewall paradox*, which brought the conceptual problems with black hole complementarity into much sharper relief [15]. They offered an experiment that a single observer could do (disregarding the limitations of current technology of course!), yet which yields an apparent conflict with accepted principles of black hole physics. The setup involves three assumptions about the description of black holes that when put together give an "absurd" consequence:

1. *No drama:* As Bob crosses the event horizon, he should not notice anything different or special. This is the prediction of classical general relativity: in Bob's frame the event horizon should look like the usual spacetime vacuum. This has the following extremely important consequence. One of the central points of quantum field theory is that there are large amounts of entanglement in just the vacuum state. The amount of entanglement rapidly falls off with distance. As a crude model of this, imagine the universe as a giant lattice of qubits and the laws of physics as some sort of local Hamiltonian whose ground state looks a lot like a cluster state. Now, for Bob to perceive normal spacetime as he crosses the event horizon means that he perceives all the entanglement of the vacuum state. In particular, some of this entanglement straddles the event horizon. Incidentally, this helps explain why Hawking radiation looks like a mixed state initially. Even assuming that the Hawking radiation is a unitary process, locally it can certainly look like it is in a mixed state as the radiation is entangled with modes still inside the black hole.

2. *Unitary evaporation:* We have already discussed this one. We assume that some unitary U maps the infalling state to the state of the Hawking radiation.

3. *Black holes are "fast scramblers":* Considering Alice who is still outside the event horizon, the whole formation and evaporation of the black hole to her looks like some big unitary transformation acting on a collection of qubits. What kind of unitary transformation is it? Whatever it is, we assume it must be very complicated and that it "scrambles" up the qubits very quickly. We can think of the black hole as a quantum circuit with a whole bunch of gates all over the place—a polynomial-size circuit, although here "polynomial" could correspond to a googol years. Now we can ask: what is the pure state that comes out after the black hole has fully evaporated? Certainly, this state is not Haar-random, not a generic state, and we know this as this state was generated by a polynomial-size circuit meaning it has to have low complexity. The conjecture now is that the state is *pseudorandom*, meaning it acts as a

random state for most physical purposes. For example if you look at the reduced state of any small number of qubits, it will just be exponentially close to the maximally mixed state.

We are now set up for the firewall paradox. Assuming you know the complete state of all the infalling matter into the black hole, let the black hole collapse, collect all the Hawking radiation that comes out, route it into a quantum computer, and wait long enough for the black hole to mostly evaporate but not completely. What do we mean by mostly? Assuming unitary evaporation and fast scrambling, how long will we have to wait until we see any nontrivial patterns in the Hawking radiation that comes out (such as entanglement between the qubits)?

We can convert this into a quantum information type problem. By the fast scrambling hypothesis, we can approximate $U|\psi\rangle$ as an n-qubit Haar-random state. Suppose that we were to look at the reduced density matrix of the first qubit only. This would be close to the maximally mixed state. Looking at the reduced density matrix of the first two qubits, we would again see the maximally mixed state and so on for the first k qubits—right up, we claim, until $k = \frac{n}{2}$ (consider $|\psi\rangle$ to be split into two sections $|\psi\rangle = \sum_i |v_i\rangle |w_i\rangle$, where $|v_i\rangle$ is the first k qubits and $|w_i\rangle$ is the remaining $n-k$ ignoring normalization). If $k < \frac{n}{2}$, then $2^k < 2^{n-k}$, meaning that the $|v_i\rangle$'s "fill up their Hilbert space" and the state of the first k qubits looks maximally mixed up to some small correction. If $k > \frac{n}{2}$ by contrast, $|v_i\rangle$ is larger than $|w_i\rangle$, and now we cannot fill up the Hilbert space with $2^{n-k} < 2^k$ pure states meaning the state of the first k qubits cannot be maximally mixed. This halfway point of black hole evaporation is called the *Page time* after the physicist Don Page. The Page time is when, information-theoretically, you should be able to see correlations in the Hawking radiation coming out of the black hole.

Now, we collect the Hawking radiation and wait until the Page time. Then we apply some suitable unitary to all of this radiation in such a way that we see the near-horizon mode is entangled with all past Hawking radiation. By acting unitarily on only the old Hawking radiation we can distill an EPR pair with the near-horizon modes. We then jump into the black hole (we just cannot help our curiosity!) and now we have a problem as the near-horizon mode is now *also* entangled with a mode inside the black hole. This violates the monogamy of entanglement, and even sets up a scenario where the observer sees the violation. What does it mean to "see" a violation of monogamy of entanglement? A qubit that is maximally entangled with two different qubits is not possible, meaning that we cannot be maximally entangled with the interior mode. Hence, we do not see a smooth progression of spacetime as we cross and we are no longer in the realm where quantum field theory accurately describes what is happening. In fact, at the event horizon one would see a wall of radiation which would be at the Planck energy (giving the firewall paradox its name) and there is in some sense no interior to the black hole. Spacetime ends at the event horizon.

Do we reject unitary evaporation of black holes? Does this imply we were wrong about black holes—that they do not have interiors at all, and instead we just have firewalls? How can we get out of this?

24.5 "Resolution" of Firewall Paradox?

A year later in 2013, Harlow and Hayden made the radical proposal that the resolution of the firewall paradox has something to do with quantum circuit complexity [39]. Going back to the AMPS experiment, after the Page time there is some unitary that you can apply to the old Hawking

radiation that puts it into an EPR pair with the near-horizon modes. But the argument for this relied on counting the number of dimensions of the Hilbert space—a purely information-theoretic argument. If we tried to extract an actual quantum circuit from the abstract dimension-counting argument, that circuit would have about $4^k \geq 2^n$ gates, as we would simply be synthesizing an otherwise structureless $2^k \times 2^k$ unitary transformation.

Let us formulate this as a computational problem:

Harlow-Hayden Decoding Problem: Given as input a description of a quantum circuit C (corresponding to the action of the black hole), which maps some simple initial state, say $|0\rangle^{\otimes n}$ to $|\phi_{RBH}\rangle$, consisting of three sets of qubits where R is the old Hawking radiation, B is the near-horizon mode (consisting of a single qubit), and H is whatever qubits are still inside the black hole. We are promised that there exists a unitary U acting only on R, which puts B and the last qubit of R into an EPR pair. Apply such a U to R.

Conjecture 33. *In the worst case, Harlow-Hayden decoding requires a quantum circuit of size exponential in n.*

In terms of astrophysical scales, this suggests that the decoding problem would take $2^{10^{70}}$ years, rather than the "mere" 10^{70} years we were considering before!

Harlow and Hayden's main result was as follows.

Theorem 34. *If the Harlow-Hayden decoding problem is always easy, then all collision-resistant hash functions can be broken in quantum polynomial time (i.e.,* BQP*).*

Aaronson [1] showed that, in the black-box setting, quantum computers *cannot* break all collision-resistant hash functions in polynomial times, which gives some evidence that Conjecture 33 is true. Later, Aaronson [5] gave stronger evidence for Conjecture 33, by showing that it follows from the existence of injective one-way functions that quantum computers cannot invert.

If Conjecture 33 holds, the implication is that we would have no hope of applying U before the black hole evaporated away, which means you can never jump into the black hole to violate monogamy of entanglement. This suggests that the smoothness of spacetime is protected by an "armor" of computational hardness. Problem solved! (Maybe.)

Bibliography

[1] Scott Aaronson. Quantum lower bound for the collision problem. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 635–642, 2002. 150

[2] Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461(2063):3473–3482, 2005. 96

[3] Scott Aaronson. The learnability of quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3089–3114, 2007. 138, 141, 142

[4] Scott Aaronson. A linear-optical proof that the permanent is# p-hard. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467(2136):3393–3405, 2011. 97

[5] Scott Aaronson. The complexity of quantum states and transformations: from quantum money to black holes. *arXiv preprint arXiv:1607.05256*, 2016. 141, 150

[6] Scott Aaronson. Shadow tomography of quantum states. *SIAM Journal on Computing*, 49(5):STOC18–368, 2019. 142

[7] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 333–342, 2011. 76, 80, 81, 82, 83, 90

[8] Scott Aaronson and Daniel J Brod. Bosonsampling with lost photons. *Physical Review A*, 93(1):012335, 2016. 88

[9] Scott Aaronson and Lijie Chen. Complexity-theoretic foundations of quantum supremacy experiments. *arXiv preprint arXiv:1612.05903*, 2016. 91

[10] Scott Aaronson and Daniel Gottesman. Improved Simulation of Stabilizer Circuits. *Physical Review A*, 70(5), Nov 2004. 28, 31

[11] Scott Aaronson and Daniel Gottesman. Improved Simulation of Stabilizer Circuits. *Phys. Rev. A*, 70:052328, 2004. quant-ph/0406196. 34, 35, 38

[12] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004. 40, 41, 42, 46

[13] Scott Aaronson and Guy N Rothblum. Gentle measurement of quantum states and differential privacy. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 322–333, 2019. 142

[14] Dorit Aharonov, Wim Van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, 50(4):755–787, 2008. 19

[15] Ahmed Almheiri, Donald Marolf, Joseph Polchinski, and James Sully. Black holes: complementarity or firewalls? *Journal of High Energy Physics*, 2013(2):1–20, 2013. 148

[16] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. 91

[17] Yosi Atia and Dorit Aharonov. How the high-energy part of the spectrum affects the adiabatic computation gap. *arXiv preprint arXiv:1906.02581*, 2019. 22

[18] Costin Bădescu and Ryan O'Donnell. Improved quantum data analysis. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1398–1411, 2021. 142

[19] Peter L Bartlett and Philip M Long. Prediction, learning, uniform convergence, and scale-sensitive dimensions. *Journal of Computer and System Sciences*, 56(2):174–190, 1998. 139

[20] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989. 133, 137

[21] Adam Bouland and Scott Aaronson. Any beamsplitter generates universal quantum linear optics. In *Electron. Colloquium Comput. Complex.*, volume 20, page 147. Citeseer, 2013. 55

[22] Sergey Bravyi, David P Divincenzo, Roberto I Oliveira, and Barbara M Terhal. The complexity of stoquastic local hamiltonian problems. *arXiv preprint quant-ph/0606140*, 2006. 20

[23] Sergey Bravyi and David Gosset. Improved classical simulation of quantum circuits dominated by clifford gates. *Physical review letters*, 116(25):250501, 2016. 46

[24] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(2):022316, 2005. 43

[25] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 517–526. IEEE, 2009. 119, 123, 125

[26] Daniel J Brod, Ernesto F Galvão, Andrea Crespi, Roberto Osellame, Nicolò Spagnolo, and Fabio Sciarrino. Photonic implementation of boson sampling: a review. *Advanced Photonics*, 1(3):034001, 2019. 90

[27] Andrew M Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 59–68, 2003. 17

[28] Bryan Eastin and Emanuel Knill. Restrictions on Transversal Encoded Quantum Gate Sets. *Phys. Rev. Lett.*, 102:110502, 2009. arXiv:0811.4262. 37

[29] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. *Physical review letters*, 102(11):110502, 2009. 42

[30] Steven T Flammia, David Gross, Yi-Kai Liu, and Jens Eisert. Quantum tomography via compressed sensing: error bounds, sample complexity and efficient estimators. *New Journal of Physics*, 14(9):095022, sep 2012. 130

[31] András Gilyén and Umesh Vazirani. (sub)exponential advantage of adiabatic quantum computation with no sign problem, 2020. 17, 19

[32] Roy J Glauber. The quantum theory of optical coherence. *Physical Review*, 130(6):2529, 1963. 54

[33] Daniel Gottesman. The heisenberg representation of quantum computers. *arXiv preprint quant-ph/9807006*, 1998. 25, 28, 33

[34] Daniel Gottesman and Isaac L Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390–393, 1999. 98, 99, 103, 104

[35] Daniel Grier and Luke Schaeffer. New hardness results for the permanent using linear optics. *arXiv preprint arXiv:1610.04670*, 2016. 98

[36] Leonid Gurvits. On the complexity of mixed discriminants and related problems. In *International Symposium on Mathematical Foundations of Computer Science*, pages 447–458. Springer, 2005. 73

[37] Jeongwan Haah, Aram W. Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu. Sample-optimal tomography of quantum states. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 913–925, New York, NY, USA, 2016. Association for Computing Machinery. 129

[38] Craig S Hamilton, Regina Kruse, Linda Sansoni, Sonja Barkhofen, Christine Silberhorn, and Igor Jex. Gaussian boson sampling. *Physical review letters*, 119(17):170501, 2017. 88

[39] Daniel Harlow and Patrick Hayden. Quantum computation vs. firewalls. *Journal of High Energy Physics*, 2013(6):1–56, 2013. 146, 149

[40] Matthew B. Hastings. The power of adiabatic quantum computation with no sign problem. *Quantum*, 5:597, Dec 2021. 17, 19

[41] Ulrich Hertrampf, Steffen Reith, and Heribert Vollmer. A Note on Closure Properties of Logspace MOD Classes. *Information Processing Letters*, 75(3):91–93, 2000. 35

[42] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4):671–697, 2004. 79

[43] E Knill, R Laflamme, and G Milburn. Efficient linear optics quantum computation. *arXiv preprint quant-ph/0006088*, 2000. 91, 92

[44] Emanuel Knill. Bounds on the probability of success of postselected nonlinear sign shifts implemented with linear optics. *Physical Review A*, 68(6):064303, 2003. 92, 94

[45] Emanuel Knill, Raymond Laflamme, and Gerald J Milburn. A scheme for efficient quantum computation with linear optics. *nature*, 409(6816):46–52, 2001. 97, 98, 100, 102, 103, 104

[46] Richard Kueng, Holger Rauhut, and Ulrich Terstiege. Low rank matrix recovery from rank one measurements. *Applied and Computational Harmonic Analysis*, 42(1):88–116, 2017. 129

[47] Richard Lipton. New directions in testing. *Distributed computing and cryptography*, 2:191–202, 1991. 82

[48] Benjamin Lovitz and Vincent Steffan. New techniques for bounding stabilizer rank. *Quantum*, 6:692, 2022. 47

[49] Austin P Lund, Anthony Laing, Saleh Rahimi-Keshari, Terry Rudolph, Jeremy L O'Brien, and Timothy C Ralph. Boson sampling from a gaussian state. *Physical review letters*, 113(10):100502, 2014. 87

[50] Urmila Mahadev. Classical verification of quantum computations. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 259–267. IEEE, 2018. 123, 124, 125

[51] Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 369–376. IEEE, 1999. 140

[52] Sepehr Nezami. Permanent of random matrices from representation theory: moments, numerics, concentration, and comments on hardness of boson-sampling. *arXiv preprint arXiv:2104.06423*, 2021. 83

[53] Ryan O'Donnell and John Wright. Efficient quantum tomography. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 899–912, New York, NY, USA, 2016. Association for Computing Machinery. 129, 130

[54] Shir Peleg, Amir Shpilka, and Ben Lee Volk. Lower bounds on stabilizer rank. *Quantum*, 6:652, 2022. 47

[55] Hammam Qassim, Hakop Pashayan, and David Gosset. Improved upper bounds on the stabilizer rank of magic states. *Quantum*, 5:606, 2021. 46, 47

[56] Robert Raussendorf and Hans J Briegel. Quantum computing via measurements only. *arXiv preprint quant-ph/0010033*, 2000. 104

[57] Robert Raussendorf and Hans J Briegel. A one-way quantum computer. *Physical review letters*, 86(22):5188, 2001. 113

[58] Ran Raz and Avishay Tal. Oracle separation of bqp and ph. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pages 13–23, 2019. 20

[59] Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. 47

[60] Michael Reck, Anton Zeilinger, Herbert J Bernstein, and Philip Bertani. Experimental realization of any discrete unitary operator. *Physical review letters*, 73(1):58, 1994. 55, 93, 96

[61] Ben W Reichardt, Falk Unger, and Umesh Vazirani. A classical leash for a quantum system: Command of quantum systems via rigidity of chsh games. *arXiv preprint arXiv:1209.0448*, 2012. 123, 124, 125

[62] Andrea Rocchetto. Stabiliser states are efficiently pac-learnable. *arXiv preprint arXiv:1705.00345*, 2017. 142

[63] Andrea Rocchetto, Scott Aaronson, Simone Severini, Gonzalo Carvacho, Davide Poderini, Iris Agresti, Marco Bentivegna, and Fabio Sciarrino. Experimental learning of quantum states. *Science Advances*, 5(3):eaau1946, 2019. 130

[64] Adi Shamir. Ip= pspace. *Journal of the ACM (JACM)*, 39(4):869–877, 1992. 120

[65] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 330–335, 1983. 77

[66] Leonard Susskind, Larus Thorlacius, and John Uglum. The stretched horizon and black hole complementarity. *Physical Review D*, 48(8):3743, 1993. 147

[67] Barbara M Terhal and David P DiVincenzo. Classical simulation of noninteracting-fermion quantum circuits. *Physical Review A*, 65(3):032325, 2002. 62

[68] Seinosuke Toda. Pp is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991. 77

[69] Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979. 57, 71, 79, 97

[70] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 131, 135, 141

[71] Leslie G Valiant. Quantum circuits that can be simulated classically in polynomial time. *SIAM Journal on Computing*, 31(4):1229–1254, 2002. 62

[72] Guifré Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical review letters*, 91(14):147902, 2003. 117

[73] Benjamin Villalonga, Murphy Yuezhen Niu, Li Li, Hartmut Neven, John C Platt, Vadim N Smelyanskiy, and Sergio Boixo. Efficient approximation of experimental gaussian boson sampling. *arXiv preprint arXiv:2109.11525*, 2021. 91

[74] Steven Weinberg. *Lectures on quantum mechanics*. Cambridge University Press, 2015. 57

[75] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, et al. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020. 88, 90