CMPT 476 Lecture 24

Hamiltonian Simulation

Course experience surveys are out. Fill them out.

Last class we talked about some applications of the quantum phase estimation algorithm to quantum chemistry — this assumed access to a unitary operator $U(t)$ simulating the dynamics of an atomic system. This brings us to Feynman's original question — can a quantum computer efficiently simulate the dynamics of a quantum system, as defined by Schrödinger's equation? The short answer is yes and no as we'll discuss today, along with some methods of doing so.

(Hamiltonian simulation)

We can define the Hamiltonian simulation problem as so:

A model is given as a collection of hamiltonians that govern each interaction

Hamiltonian simulation problem

Input: A $2^n \times 2^n$ Hermitian matrix $\hat{H}$, real numbers $t, \varepsilon > 0$, and an initial state $|\psi_0\rangle \in \mathbb{C}^{2^n}$

Goal: A state $|\tilde{\psi}_t\rangle \approx e^{-i\hat{H}t}|\psi_0\rangle$ such that $|\langle\psi_t| e^{-i\hat{H}t} |\psi_0\rangle|^2 \geq 1-\varepsilon$

We want to do phase estimation of the unitary

e values in the hamiltonian are zero, estimation is difficult (why?)

Hamiltonian eigenvalues are the energy levels of the system.
If it is a fock style Hamiltonian, it is the energy in the total system.

Ground state estimation is one application of this general notion of simulation, but the basic idea is to get our computer to act like a target quantum system with Hamiltonian $\hat{H}$ so that we can observe (through evolution and measurements) properties of the quantum system. In practice, this means approximating $e^{-i\hat{H}t}$ with a poly-size circuit U to error $\approx \varepsilon$. The approximation error is chosen so that $|(\langle\psi_0|U^\dagger)(e^{-i\hat{H}t}|\psi_0\rangle)|^2 \geq 1-\varepsilon$ (the state fidelity) as in the problem statement.

So, how do we do this?

Well, first we need to understand a bit more about the construction of Hamiltonians. Most Hamiltonians arise from a model, which describes a Hamiltonian as a sum $\hat{H} = \hat{H}_1 + \hat{H}_2 + \cdots$ of simpler Hamiltonians corresponding to particular forces. Mathematically we're allowed to do this because (real-valued) linear combinations of Hermitian matrices are Hermitian.

(Sum of Hermitian matrices is Hermitian)

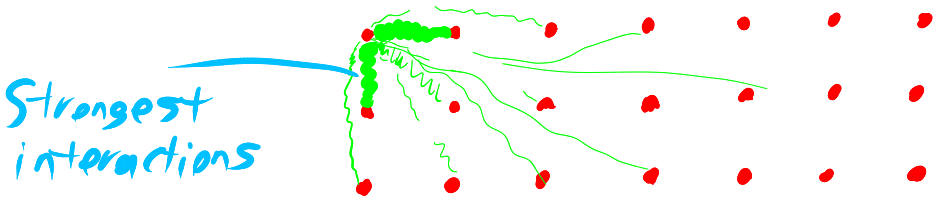
Let H_1 and H_2 be $N \times N$ Hermitian matrices.
Then for any $\alpha, \beta \in \mathbb{R}$, $\alpha H_1 + \beta H_2$ is Hermitian.

Proof

$$(\alpha H_1 + \beta H_2)^\dagger = \alpha^* H_1^\dagger + \beta^* H_2^\dagger = \alpha H_1 + \beta H_2$$ □

(The Ising model)

To illustrate how a Hamiltonian is broken down into a sum of parts, consider a lattice of interacting atomic particles



Each particle has a "spin" in 3-D space which generates a magnetic field affecting nearby particles. This interaction tapers off drastically with distance, so we can approximate the dynamics by assuming adjacent particles interact. We can hence describe the Hamiltonian of the system as a sum of Hamiltonians governing

1. Each individual particle
2. Each interaction between neighbours

Ask professor about his comment on how the addition of identity matrices will increase the density of along the diagonal.

In particular,

$$\hat{H} = \sum_i \hat{H}_i + \sum_{i,j \mid i<j \text{ adjacent}} \hat{H}_{i,j}$$

Text

The Ising model further decomposes each term $\hat{H}_i$, $\hat{H}_{i,j}$ as Pauli matrices acting on particle(s) i, j, corresponding to the particle's spin. In the simplest version,

$$\hat{H}_i \propto X_i$$
$$\hat{H}_{i,j} \propto Z_i Z_j$$

$$A_i = I^{\otimes i-1} \otimes A \otimes I^{N-i}$$
(i.e. matrix A on subsystem i)

So the Ising Hamiltonian is often written as

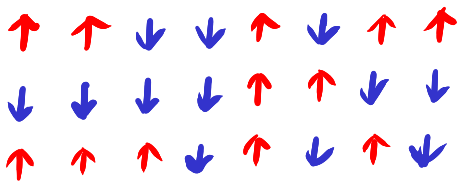
Zi is the poly-matrix acting on i? Ask professor to go over what these symbols mean

$$\hat{H} = \lambda \sum_i X_i + \sum_{i,j} J_{i,j} Z_i Z_j$$

λ: external field strength

$J_{i,j}$: coupling strength

Without the external field, we can think of the direction of each particle's magnetic field as either up or down, corresponding to the eigenvalue of Z_i and governed by $\hat{H}$.



Theorem

Computing the ground state of the Ising model is NP-hard classically.

(Adiabatic quantum computing, quantum annealers, and DWAVE)

The above can be proven by encoding the NP-Complete MAXCUT problem into the ground state of an Ising model. We won't go into the details of this reduction, but many NP-hard and generally combinatorial optimization problems can be encoded in the ground state of an Ising model. In math terms, the Ising problem is a QUBO:

We can do this where the hamiltonian of the Ising model is equivalent to QUBO.

quadratic unconstrained binary optimization problem

Any solution to a QUBO is equivalent to the solution of the ground state encoding in the Ising model

The QUBO and Ising model is at the heart of nearby QC giant D-Wave's approach.

The very rough idea is:

If you have a small spectral gap, it is easy for your system to jump from one state into another.
With a small spectral gap, the D-wave approach does not approach.
We could try to slowy move to the Hamiltonian, but that could take exponential time.

1. Program a QUBO into the system Hamiltonian
2. Begin in some known ground state (e.g. $|+\rangle|+\rangle\cdots|+\rangle$)
3. Slowly change the Hamiltonian to the encoded Hamiltonian
4. If we're lucky, we're still in the ground state, but now it's the ground state of the target Hamiltonian
5. Profit 

This approach is called quantum annealing and is motivated by the Adiabatic theorem:

We can always write the hamiltonian in terms of the Pauli matrices.

A system in a ground state will remain in a ground state even if the Hamiltonian is slowly changing over time.

In particular, if we set $\hat{H}_{initial}$ to be some Hamiltonian with an easy to prepare ground state, and $\hat{H}_{final}$ to be the encoded problem Hamiltonian, then at time t set the system Hamiltonian to

$$\hat{H}(t) = (1-t)\hat{H}_{initial} + t\hat{H}_{final}$$

the adiabatic theorem states that under ideal conditions, we will end in a ground state of $\hat{H}_{final}$. Moreover, it turns out that this is equivalent in power to the gate-model of QC.

Theorem

Adiabatic quantum computing is poly-time equivalent to gate-based quantum computation

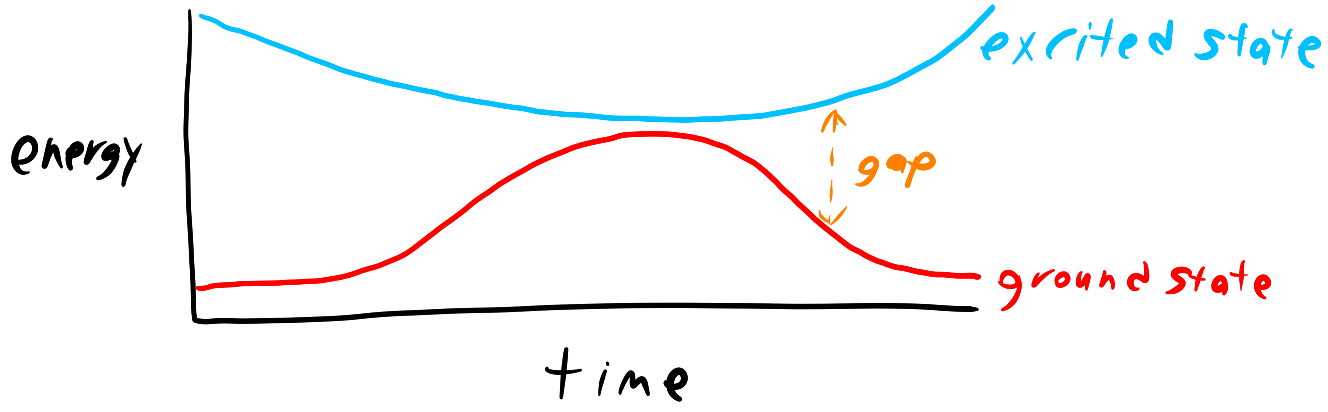
---

So, why hasn't D-Wave and quantum annealing factored large integers already?

The reasons are complicated and numerous. One of which is that quantum annealing is a heuristic approach to the adiabatic theorem. One of the reasons for this is a lack of knowledge of the energy gaps.

(Energy gaps and the adiabatic theorem)

The adiabatic theorem doesn't state that the system can't jump to an excited state, just that the probability is related to the energy gap, temperature, and speed.



If at some point in time, the gap in the smallest and next smallest eigenvalue of $\hat{H}$ is very small (as above), the system may spontaneously move to the excited state. To combat this, we can go even slower which will decrease the probability, but result in a longer (potentially exponential) runtime.

Fact

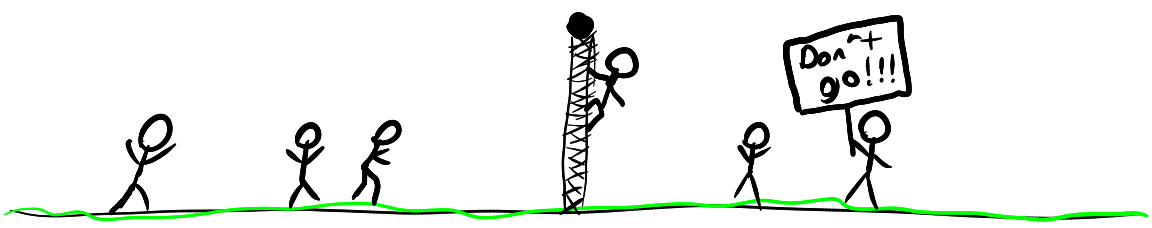
Adiabatic evolution of the Ising model is not known to solve NP-complete problems in polynomial time, due to the spectral gap problem.

(Practical limitations of quantum annealing)

Even if there is a large spectral gap, to properly implement the adiabatic theorem, we need to know the gap, and be able to faithfully apply the change slowly enough to avoid an excitation, which itself technically requires temperatures at absolute zero.

So, quantum annealing throws most formal gaurantees of the adiabatic theorem out and just tries to do it anyway and hope for the best.

On to D-Wave, while there are good reasons to believe quantum annealing in general will provide better heuristics than classical optimization approaches, it turns out that D-Wave's Hamiltonian belongs to a class for which classical methods appear to work equally well (or better). Point being, the exact nature of adiabatic QC & quantum annealing-based speed ups are still active areas of research with many people on either side of the fence.



(Back to Hamiltonian simulation)

Now that we're done with that, let's get back to simulating e^{-iHt}. Our goal is a poly-size circuit approximating e^{-iHt} to error ε. In general, this is **not possible** by virtue of the fact that **almost all n-qubit unitaries have exponential gate complexity** (just like Shannon's theorem in classical computation). However, for particular models like the **Ising** model, which are sums of a small number of **simple** terms, we can simulate them efficiently. What are these simple terms? Well, the simplest and most common terms, which also comprise the terms in the Ising model, are **Pauli operators**.

(Simulation of Pauli terms)

An **n-qubit Pauli** P is a tensor product of Pauli matrices on each qubit

$$P = P_1 \otimes P_2 \otimes \cdots \otimes P_n$$

where $P_i \in \{I, X, Y, Z\}$. Recall that for diagonal Λ

Any Hermitian matrix can be written as a sum of Pauli matrices.

$$\Lambda = \sum_i \lambda_i |i\rangle\langle i| = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}$$

we defined the matrix exponential as

$$e^{i\Lambda} = \sum_i e^{i(\lambda_i)} |i\rangle\langle i| = \begin{bmatrix} e^{i\lambda_1} & & \\ & \ddots & \\ & & e^{i\lambda_N} \end{bmatrix}$$

Then for any Hermitian (or normal) operator A, by the spectral theorem

$$A = U \Lambda U^\dagger$$

Assuming that our computer can do z axis rotations, we can use the z gates.
We need to diagonalize X and Y instead of using an X axis and Y axis rotation.

where Λ is diagonal and U is unitary, so

We want to diagonalize to simplify things when taking the tensor product of Pauli matrices.

$$e^{iA} = U e^{i\Lambda} U^\dagger$$

p = X tensor Y
= (H tensor SH)(Z tensor Z)(H tensor H S-dagger)
We would only need to implement the time evolution of (Z tensor Z)
e^{-iZ tensor Z} =

Noting that Z is diagonal, and

$$X = HZH$$

$$Y = SHZHS^\dagger$$

we can diagonalize any Pauli $P_1 \otimes \cdots \otimes P_n$ as

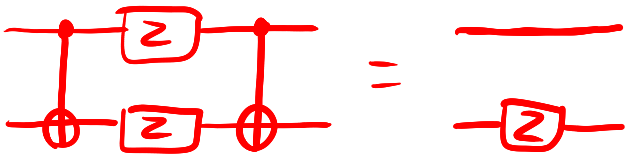
$$P_1 \otimes \cdots \otimes P_n = (U_1 \otimes \cdots \otimes U_n) \Lambda (U_1^\dagger \otimes \cdots \otimes U_n^\dagger)$$

where in particular Λ is a tensor product of Z and I Paulis. So how do we simulate $e^{-i\Lambda t}$? We might hope that $e^{i(\Lambda_1 \otimes \Lambda_2)} = e^{i\Lambda_1} \otimes e^{i\Lambda_2}$, but this doesn't work since

e^A tensor-prod e^B has an eigen value of e^{l1} * e^{l2}, l1, l2 the eigenvalues

$$e^{i\lambda_1} e^{i\lambda} = e^{i(\lambda_1 + \lambda_2)} \neq e^{i\lambda_1 \lambda_2}$$

We can however note that



So using CNOT gates we can see that

$$\Lambda = U(I \otimes I \otimes \cdots \otimes I \otimes Z) U^\dagger$$

It suffices to observe now that

$$e^{-i(I \otimes \cdots \otimes I \otimes Z)t} = I \otimes \cdots \otimes I \otimes e^{-iZt}$$

Ex. Simulate $e^{-i(X\otimes Y\otimes I\otimes Z)t}$

We first diagonalize $X\otimes Y\otimes I\otimes Z$ by diagonalizing X and Y:

$$X\otimes Y\otimes I\otimes Z$$
$$= (H\otimes SH\otimes I\otimes I)(Z\otimes Z\otimes I\otimes Z)(H\otimes HS^\dagger\otimes I\otimes I)$$

Then we map $Z\otimes Z\otimes I\otimes Z$ to $I\otimes I\otimes I\otimes Z$ with CNOT gates

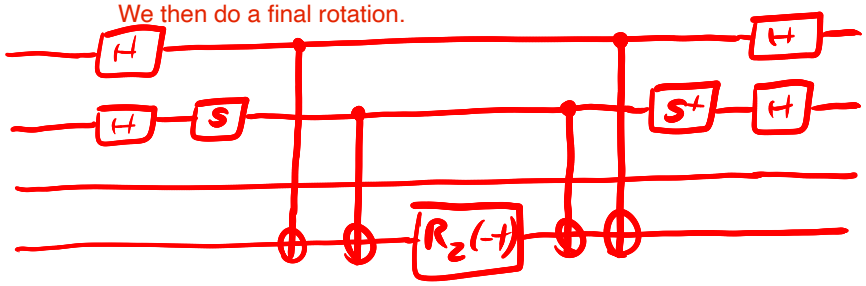
$$(Z\otimes Z\otimes I\otimes Z)$$
$$= CNOT_{1,4}\, CNOT_{2,4}\,(I\otimes I\otimes I\otimes Z)\, CNOT_{2,4}\, CNOT_{1,4}$$

Finally

$$e^{-i(I\otimes I\otimes I\otimes Z)t} = I\otimes I\otimes I\otimes e^{-iZt}$$
$$= I\otimes I\otimes I\otimes R_z(-t)$$

As a circuit,

need to cancel out the Z on the first qubit, so we do a CNOT to cancel out the z on the first and second qubit. We then do a final rotation.



Aside

A quicker route to get to this point is via the Gottesman-Knill theorem, which states among other things that the Normalizer of the Pauli group is generated by H, S, and CNOT gates...

whatever you say man

(Simulation of sums of Hamiltonians)

So, we can efficiently simulate Pauli terms, but what do we do with a Hamiltonian which (like many) is a sum of Pauli terms? Well, our first guess might be to take

$$e^{A+B} = e^A e^B$$

If we have matrices (which do not commute), we need to use the trotter formula.

If $A, B \in \mathbb{R}$ we know this from grade school math. However, this fails for matrices (and in general in any algebra where multiplication is noncommutative, in that $AB \neq BA$). To understand why, we need the real definition of matrix exponentials.

(Matrix exponential)

Let T be an operator on a Hilbert space. Then

$$e^T = \sum_{m=0}^{\infty} \frac{1}{m!} T^m$$

Note that this is exactly the Taylor series of e^x.

(Properties of matrix exponentials)

From the formal definition, we can observe that

1. $e^{\sum_i \lambda_i |i\rangle\langle i|} = \sum_i e^{\lambda_i} |i\rangle\langle i|$
2. $e^{A+B} \neq e^A e^B$ if $AB \neq BA$
3. $e^{A+B} = e^A e^B$ if $AB = BA$

We say that matrices A & B commute if $AB = BA$.

(Trotterization)

Given a Hamiltonian $\hat{H} = \hat{H}_1 + \hat{H}_2$, if H_1 & H_2 commute we can simulate $\hat{H}$ by simulating H_2 then H_1 by the above properties. That is,

$$e^{-i\hat{H}t} = e^{-i\hat{H}_1 t}\, e^{-i\hat{H}_2 t}$$

Is all hope lost if H_1 & H_2 don't commute?

NO, because mathematicians like to study things that don't commute ☺. One such mathematician was the Canadian Hale Trotter, who showed the following.

(Trotter formula)

Let A and B be square complex matrices. Then

$$e^{A+B} = \lim_{n\to\infty} \left(e^{A/n} e^{B/n}\right)^n$$

The Trotter formula tells us that

$$e^{-i(\hat{H}_1+\hat{H}_2)t} \approx e^{-i\hat{H}_1 t/\varepsilon} e^{-i\hat{H}_2 t/\varepsilon} e^{-i\hat{H}_1 t/\varepsilon} e^{-i\hat{H}_2 t/\varepsilon} \cdots e^{-i\hat{H}_1 t/\varepsilon} e^{-i\hat{H}_2 t/\varepsilon}$$

As this formula goes n times.

We want large n, but we can also think of n as 1/epsilon, so we want small epsilon.

for some small ε. How small does ε need to be? Well, we state without proof that the approximation error in the above is $O(\frac{1}{\varepsilon})$. Moreover, if

$$\hat{H} = \hat{H}_1 + \hat{H}_2 + \cdots + \hat{H}_k$$

Then it can be shown that

$$e^{-i\hat{H}t} = \left(e^{-iH_1 t/\varepsilon} e^{-iH_2 t/\varepsilon} \cdots e^{-iH_k t/\varepsilon}\right)^{\varepsilon} + O\left(\frac{1}{\varepsilon}\right)$$

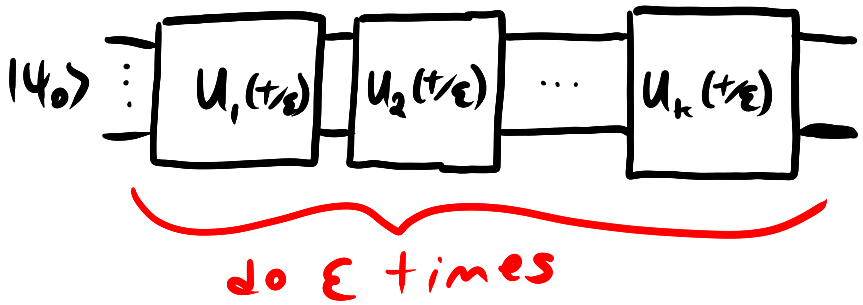
Approximating $e^{-i\hat{H}t}$ in this way is often called Trotterizing the Hamiltonian.

(Hamiltonian simulation by Trotterization)

Given a Hamiltonian $\hat{H}$ which is a sum of k easily exponentiated terms (like Paulis), we can simulate $\hat{H}$ to error $1/\varepsilon$ as

$$\left(e^{-i\hat{H}_1 t/\varepsilon} e^{-i\hat{H}_2 t/\varepsilon} \cdots e^{-i\hat{H}_k t/\varepsilon}\right)^{\varepsilon}$$

Letting $U_i(t/\varepsilon) = e^{-i\hat{H}_i t/\varepsilon}$ we can draw this as



In particular, we can perform the simulation in

$$\text{poly}(n, k, \varepsilon) \text{ time}$$

(k-local Hamiltonians)

So we can efficiently simulate a Hamiltonian with (a small number of) Pauli terms. However, many Hamiltonians, like those used in quantum chemistry, are not expressed directly in terms of Paulis. For instance, the molecular Hamiltonian for the H_2 molecule can be written as

$$\hat{H} = \sum_{\alpha,\beta} t_{\alpha,\beta}\, a_\alpha^\dagger a_\beta + \sum_{\alpha,\beta,\delta,\delta} u_{\alpha\beta\delta\delta}\, a_\alpha^\dagger a_\delta^\dagger a_\delta a_\beta$$

where $a^\dagger, a$ are something called creation and annihilation operators on a (Fermionic) Fock space.

What is typically done is to map such a Hamiltonian to a sum of Paulis, which is possible since Paulis, as we saw previously, form a basis for linear operators on $\mathbb{C}^2$.

Aside: For a molecular Hamiltonian like the H_2 Hamiltonian above, we're also mapping Fock space to "qubit" or spin 1/2 space. This can be done by, among other mappings, the (inverse) Jordan-Wigner transform.

onian where each sub hamiltonian acts non-trivially on at most k-qubits, then
e^{-i(hat H)t} can be implemented in poly(n, m, k epsilon) time.

It can be observed that if the Hamiltonian only involves interactions between at most k particles — called a k-local Hamiltonian — then the mapping only needs poly(k) Paulis for each term.

Issue. Ising model is NP-hard, which is a subcase of the k-local hamiltonian.
Llyod theorem does not imply we can solve NP hard problems in polynomial time.
Division of BQP and QMA is if we want to compute an eigenvalue (BQP hard) and getting the ground state energy (QMA)

Ex. The Ising Hamiltonian

$$\hat{H} = \lambda \sum_i X_i + \sum_{i,j} J_{i,j} Z_i Z_j$$

is 2-local since each term acts non-trivially on at most 2 qubits (only adjacent qubits interact).

Theorem (-ish) (Lloyd, 1996)

For any fixed k, a k-local Hamiltonian on n qubits can be simulated in time polynomial in n (and t and ε) on a universal quantum computer.

(Implications on complexity theory)

An astute reader may be wondering:

> what the heck? we can encode MAX CUT which is NP-complete in the ground state of a 2-local Hamiltonian. Doesn't this imply quantum computers can efficiently solve NP?

Well, no. The problem appears to be approximating the ground state well enough for phase estimation to work. For this reason, we have a very special problem in quantum complexity theory: the

k-local Hamiltonian problem

k-local Hamiltonian problem

input: a k-local Hamiltonian $\hat{H}$

goal: find the smallest eigenvalue λ of $\hat{H}$

Without going into details, in quantum complexity theory we can define two natural complexity classes which form the quantum analogue of P vs NP:

BQP (Bounded-error quantum polynomial time) (quantum easy)

and

QMA (Quantum Merlin-Arthur) (quantum hard)

The distinction between finding an eigenvalue and the smallest eigenvalue appears to distinguish BQP from QMA (assuming BQP ≠ QMA) due to the following theorems.

Theorem

The k-local Hamiltonian problem is

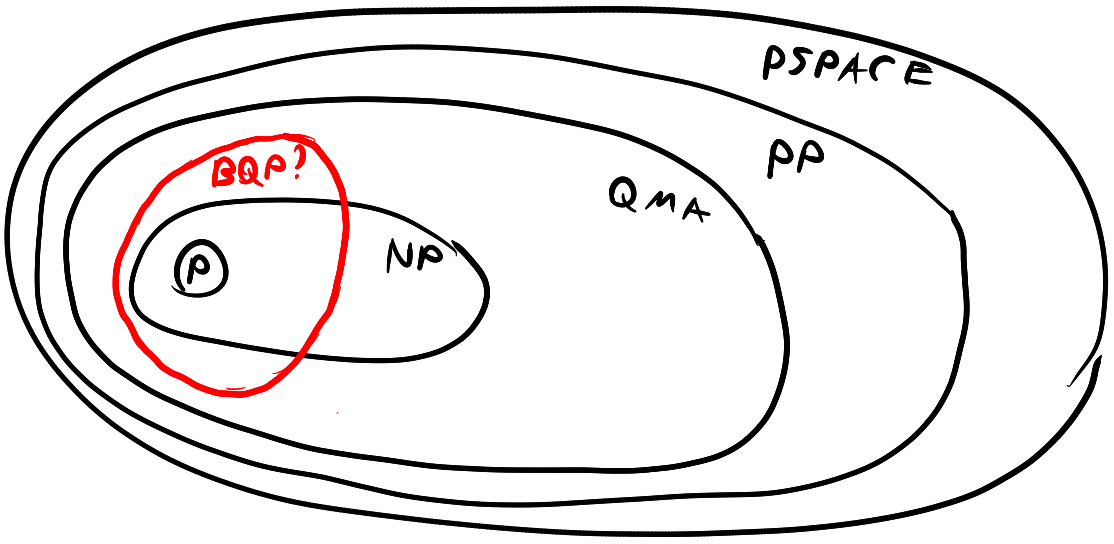
QMA-complete

Theorem

Sampling from the eigenspectrum of a k-local Hamiltonian is

BQP-complete

As with P vs. NP, the BQP vs. QMA question is still open to the best of my knowledge (a purported proof of BQP ≠ QMA exists but it seems there may be flaws). Here is the picture as we currently know it:



(Other simulation algorithms)

As the killer app for quantum computers, much work has gone into improving Lloyd's universal simulation algorithm. Broadly speaking we can divide the many existing algorithms into those using Product formulas which approximate e^{A+B} using products of e^A and e^B like the Trotter formula or (higher-order) Suzuki formulas, and those using Linear combinations of unitaries instead.

The simplest algorithm in the latter case is based on truncation of the Taylor series of e^A.

(Truncated Taylor series simulation)

Recall that the Taylor series definition of e^A is

$$e^A = \sum_{m=0}^{\infty} \frac{1}{m!} A^m$$

Since the denominator $m!$ grows extremely fast, the contributions of high-order terms become increasingly irrelevant. As such, we can truncate the Taylor series at some maximum order K, giving a finite approximation

$$e^A \approx \sum_{m=0}^{K} \frac{1}{m!} A^m$$

Without going into detail about the choice of K, the key observation is that if $\hat{H}$ is a sum of Paulis

$$\hat{H} = a_1 P_1 + a_2 P_2 + \cdots + a_k P_k$$

then

$$e^{-i\hat{H}t} \approx \sum_{m=0}^{K} \frac{t^m}{m!} (-i)^m \hat{H}^m = \alpha_1 U_1 + \alpha_2 U_2 + \cdots + \alpha_L U_L$$

is a linear combination of unitary matrices

$$U_1, \ldots, U_L$$

how we know how to implement products of unitaries — what about sums?

(Linear combination of unitaries)

Suppose we wanted to implement $U + V$ — how could we do it? If $U = I$, this looks a bit like a unitary we've already seen:

$$C\text{-}V = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes V$$

The controlled-U_2 gate applies I if the first qubit is in the $|0\rangle$ state, and V if it's in the $|1\rangle$ state. What if it's in the $|+\rangle$ state?

$$C\text{-}V\,|+\rangle|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle \otimes I|\psi\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes V|\psi\rangle$$

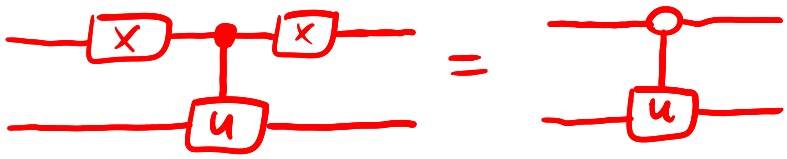
Now, intuitively we've applied a superposition of I and V, but the superposition is mediated by the first qubit with which the target is entangled and is morally in the superimposed state $|+\rangle$. What happens then if we try to un-superposition the first qubit with a Hadamard?

$$(H \otimes I)\left[\frac{1}{\sqrt{2}}|0\rangle \otimes I|\psi\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes V|\psi\rangle\right]$$

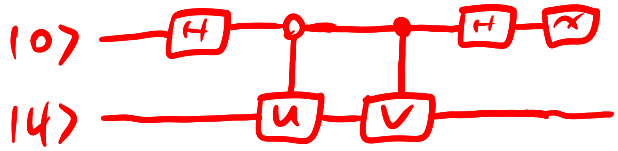
$$= \frac{1}{2}\left[|0\rangle \otimes I|\psi\rangle + |1\rangle \otimes I|\psi\rangle + |1\rangle \otimes V|\psi\rangle - |1\rangle \otimes V|\psi\rangle\right]$$

$$= \frac{1}{2}\left[|0\rangle \otimes (I+V)|\psi\rangle + |1\rangle \otimes (I-V)|\psi\rangle\right]$$

While we don't exactly have $(I+V)|4\rangle$, if we measure the first qubit, with some probability we'll get $(I+V)|4\rangle$. The probability is related to the norm of $I+V$, as it is not necessarily unitary, but we'll ignore these technicalities for now. At this point it suffices to note that



implements a Controlled-U where U is applied if and only if the first qubit is in the state $|0\rangle$ (called a negative controlled-gate). Using both controlled gates together, the circuit

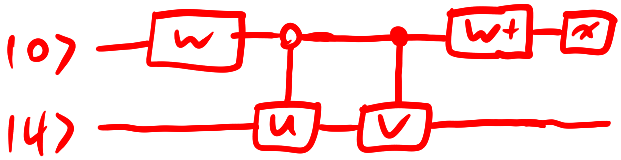


implements the transformation $U+V$ on $|4\rangle$ if the measurement result is 0.

More generally, if we want $\alpha U + \beta V$ where $|\alpha|^2+|\beta|^2=1$, we can do the same but with the first qubit in the weighted superposition $\alpha|0\rangle+\beta|1\rangle$. Note that the following unitary

$$W=\begin{bmatrix}\alpha & -\beta^* \\ \beta & \alpha^*\end{bmatrix}$$

suffices to map $|0\rangle \longmapsto \alpha|0\rangle+\beta|1\rangle$, and so our resulting circuit is



For a linear combination of k unitaries, it turns out that we can do the same thing (efficiently) with a $\log_2 k$ qubit register holding the k coefficients and a quantum multiplexer deciding which unitary to apply. These are usually called the PREPARE and SELECT operators:

$$\text{PREPARE}|0\rangle^{\otimes \log_2 k} \propto \sum_i \alpha_i |i\rangle$$

$$\text{SELECT}|i\rangle|\psi\rangle = |i\rangle(U_i|\psi\rangle)$$

This is very high-level, because we haven't even discussed the elephant in the room:

This only works with probability proportional to the norm of $\sum_i \alpha_i U_i$!

To bump up the probability sufficiently high, we need a quantum algorithm called amplitude amplification, which brings us to our next major quantum algorithm:

Grover's Search Algorithm...