Lecture 26, Tues April 25: Adiabatic Algorithm

At the end of the last lecture, we saw that the following problem is **NP**-hard:

Given as input an n-qubit Hamiltonian H of the special form $H = H_1 + \ldots + H_m$, with each H_i acting on at most 3 of the n qubits, estimate H's ground state energy.

This is a quantum generalization of the 3SAT problem, one that arises very naturally in condensed matter physics. Building a complicated Hamiltonian by summing local terms could also be seen as somewhat analogous to building a complicated quantum circuit by composing 1- and 2-qubit gates.

But how do you "give" someone a Hamiltonian like that?

Providing the full $2^n \times 2^n$ Hermitian matrix would be wasteful. Instead, you can simply list the local terms $H_1, \ldots, H_m$ (to some suitable precision), together with the qubits to which they're applied.

Is this problem **NP**-complete?

Since we know it's **NP**-hard, what we're asking here is whether it has polynomial-time verification. In other words, when we claim that the ground-state energy of some Hamiltonian is at most (say) 5, can we prove it by giving a short witness?

It turns out that we can---but as far as anyone knows today, only by giving a *quantum* witness! A quantum witness that works is simply the n-qubit ground state itself.

Thus, the Local Hamiltonians problem is not known to be in **NP**: it's only known to be in the quantum analogue of **NP**, which for reasons we won't go into is called **QMA** (Quantum Merlin-Arthur)

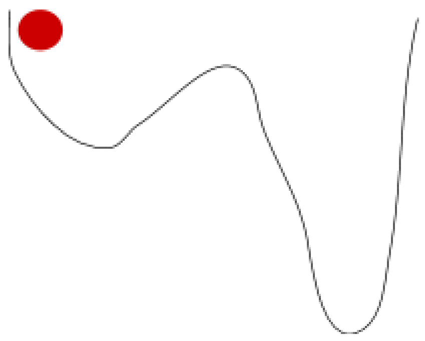
An important theorem from around 1999 says that Local Hamiltonians is actually *complete* for **QMA**, just like 3SAT is complete for **NP**.

For the specific Hamiltonian H we constructed in the last lecture---the one that encoded 3SAT---there *would* be a short classical witness, because that H was a diagonal matrix, so its ground state is always just a classical basis state. But what if H is non-diagonal, and its ground state is some complicated entangled state?

So if natural quantum systems like to settle into their ground states, and if finding the ground state is **NP**-hard, does this mean that we could use quantum systems to solve **NP**-hard problems in polynomial time "naturally"?

People talked about such questions even before the concept of quantum computing was in place.

But there's a serious problem: it's not *always* true that natural quantum systems quickly settle into their ground states. And starting from hard instances of 3SAT might produce complicated and exotic Hamiltonians, far from physicists' usual experience. Those complicated Hamiltonians might often be ones for which it's hard to reach the ground state.



In the hillside (above), will the ball get to the point that minimizes its gravitational potential energy?

Probably not anytime soon!

If we wait a million years, maybe a thunderstorm will push the ball up over the hill in the middle. But for the foreseeable future, it's much more likely for the ball to rest at the **local optimum** on the left.

In hard real-world optimization problems, you'd have a very bumpy 1000-dimensional landscape or whatever with plenty of local optima to get trapped in. You might wonder if quantum computing could help us wade through these local optima—and that certainly seems plausible. In fact, the hope that this would be true was a central starting point for today's topic:

The Adiabatic Algorithm

This is the last quantum algorithm we'll cover in this course. To this day, it remains a source of significant research and discussion.

There's a physics theorem from the 1920s called the Adiabatic Theorem:

"Adiabatic" is a term from thermodynamics; we won't really need to understand what it means in order to discuss the adiabatic algorithm.

Anyway, here's what the theorem says: suppose you prepare the ground state of some initial Hamiltonian H_i, with H_i being applied to that ground state. You then slowly and continuously change H_i into some final Hamiltonian H_f, until at the end you're applying exactly H_f. Then provided that the transition was slow enough, you'll end up at (or extremely close to) the ground state of H_f.

Assuming a few fine-print conditions that we won't go into.

So, gradual change from one ground state brings us to another ground state.

In the minds of Farhi, Goldstone, and other physicists, this suggested the following plan to solve **NP**-hard problems using Hamiltonians.

We know that $H = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ is a one-qubit Hamiltonian with $|+\rangle$ as its unique ground state.

Its eigenstates are $|+\rangle$ and $|-\rangle$.
The energy of $|+\rangle$ is 0 and the energy of $|-\rangle$ is 2, so $|+\rangle$ is the ground state.

So we can create an initial Hamiltonian H_i (where here the *i* means "initial") by applying H to each qubit individually:

$$H_i = \sum_{i=1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}_i$$

The state $|+\rangle^{\otimes n}$:

- is easy to prepare in a quantum computer

- would stay put forever if we continued to apply H_i forever.

But then, we gradually change H_i to another Hamiltonian H_f, which encodes some n-bit 3SAT instance that we'd like to solve:

Parameterizing time as $t \in [0, 1]$

$$H_i = \sum_{i=1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}_i \quad \rightsquigarrow \quad H_f = \sum_{i=1}^{m} h_i$$

In the simplest version, at any point in time t, we apply the Hamiltonian

$$H_t = (1-t)H_i + tH_f.$$

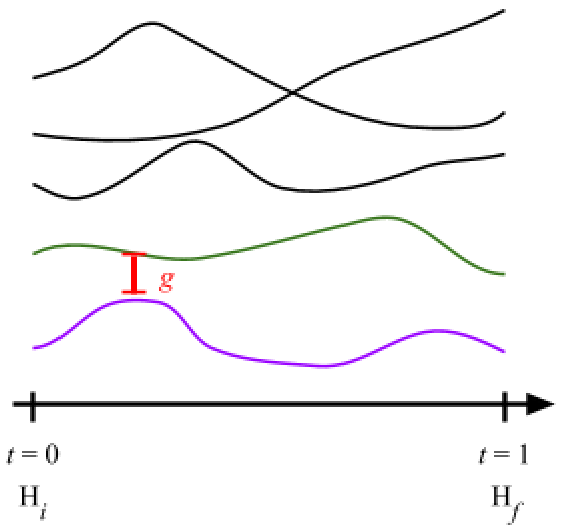
Since each H_t is a sum of terms acting on a few qubits only, we can efficiently apply these H_t's using Trotterization (discussed in the last lecture).

If everything goes according to plan, we'll end up in the ground state of H_f---in other words, the optimal solution to our 3SAT instance (!!).

So what's the catch? Well, the crux of the matter is that the transition from H_i to H_f must be sufficiently slow.

How slow?

To answer that question, let's plot the eigenvalues of H_t as a function of the time parameter t (example shown on the right).



H_t could have as many as 2^n eigenvalues (all real numbers). But we're especially interested in the smallest eigenvalue (the ground energy) and the one right above that (the first excited energy).

Our goal is to stay in the ground state throughout.

The eigenvalues, also called **energy levels**, can change continuously over time.

Sometimes two energy levels can even cross each other. Physicists call that a **level crossing**.

When they *almost* cross, it's an **avoided level crossing**.

If the two lowest energies cross each other, then we leave the ground state. Even if the two lowest energies *nearly* cross each other, there's a significant risk that we'll leave the ground state. The closer together the two lowest energies get, the slower we have to run the algorithm to avoid confusing them.

We define the **Minimum Eigenvalue Gap**, ***g***, as the smallest the gap ever gets between the first excited energy and the ground energy, as we vary the time *t*.

$$\boldsymbol{g} = \min_{t \in [0,1]} \big(\underbrace{\lambda_2^{(t)}}_{\text{First excited energy}} - \underbrace{\lambda_1^{(t)}}_{\text{Ground energy}} \big)$$

In the statement of the adiabatic theorem, one of the "fine print" conditions that we didn't mention earlier is $g > 0$.

g turns out to be a crucial quantity for determining how long the adiabatic algorithm will take to solve a given problem instance. Roughly speaking: in order to ensure that we remain in the ground state throughout, as we pass the value of *t* where the gap is *g*, we need to vary *t* at a speed of only $\sim g^2$, which takes $\sim \frac{1}{g^2}$ computation steps.

So in particular: *if* we could show that *g* was always lower-bounded by $1/n^{O(1)}$ for 3SAT (or some other **NP**-complete problem), then we'd get **NP** ⊆ **BQP**: quantum computers would be able to solve all **NP** problems in polynomial time.

An early paper about this, which was published in *Science* in 2001, basically said, "Well… this looks like it could work."

Note: In reality we're approximating the Hamiltonians with discrete-time unitary transformations, so we'd end up with something *close* to the ground state of H_f, not the ground state itself. But that would still be good enough to solve our **NP**-hard problem.

So the question boils down to:
<u>What is the behavior of g, the minimum spectral gap, for the problem instances that we want to solve?</u>

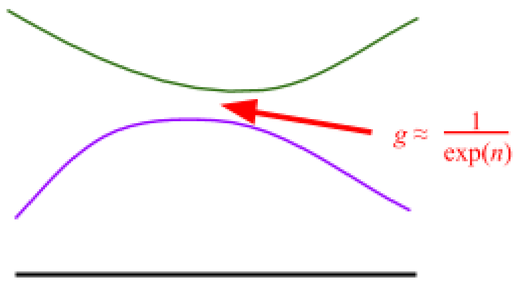
Farhi once went to an expert in condensed matter physics and said: this is the system we're looking for, do you think that *g* will decrease polynomially or exponentially as a function of *n*?

The expert says: I think it will decrease exponentially.

That's not the answer Farhi wants to hear. So Farhi asks: why? What's the physical reason?

After thinking about it some more, the expert responds: because if it only decreased polynomially, then your algorithm would work.

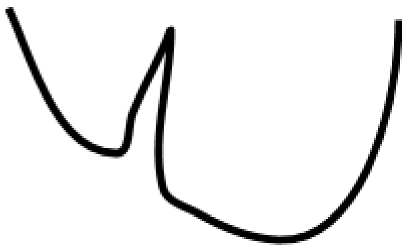
What emerged after a couple of years is that, for hard instances of 3SAT (or even 2SAT, for that matter), the ground energy and the first excited energy often *do* get exponentially close to touching. At the point where that happens--i.e., the avoided level crossing--you'd need to run the algorithm for an exponential amount of time in order to remain in the ground state, and thereby solve your SAT instance.



But the story doesn't end here. The physicists regrouped and said, "Okay, so maybe this technique doesn't *always* work, but it might still give big advantages for *some* types of optimization problems!"

By now there's been lots of research on classifying the types of solution landscapes where the adiabatic algorithm performs well and those where it performs poorly. Some encouraging results came from:

(Farhi, Goldstone, Gutmann 2002)



These authors constructed fitness landscapes that had a global minimum at the bottom of a wide basin, but also a tall thin spike blocking the way to that minimum. Starting from the far left, a classical algorithm based on Steepest Descent would get stuck forever at the base of the spike (i.e., at a local minimum), and would never reach the global minimum.

OK, but before we examine the performance of the adiabatic algorithm on this sort of landscape, shouldn't we first look at better classical algorithms?

Indeed: Simulated Annealing is a better classical optimization algorithm--one that, much like the adiabatic algorithm, will always *eventually* reach the global minimum, if you run it for long enough.

In fact, simulated annealing can be thought of as a classical counterpart to the adiabatic algorithm.

The basic idea of **Simulated Annealing** is to evaluate the fitness function around the current point and then:

- Make a move that improves fitness (Most of the time)
- Make a move that worsens fitness (Some of the time)

The probability of making a move that worsens fitness is time-dependent, and decreases as time goes on.

Trading off exploration for exploitation.

The name "simulated annealing" comes from a 7000-year old technology. **Annealing** is the process of making a metal stronger by heating it up and then slowly cooling it. This gives the atoms in the metal a chance to bounce around, escape from a local optimum that might be causing brittleness, and then slowly settle into a better optimum.

On the fitness landscape with the spike (pictured above), simulated annealing would *eventually* get over the spike despite how energetically unfavorable it is.

However, if the spike is tall enough, then it would take an exponential amount of time.

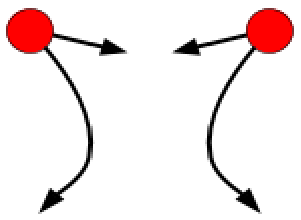
We'd be waiting for the thunderstorm, so to speak.

On the other hand, if the spike is thin enough, then Farhi et al. showed that the adiabatic algorithm can get over it in only polynomial time. It does so by exploiting a well-known quantum phenomenon called **tunneling**.

Popular articles explain tunneling by saying that a quantum particle can get through barriers that a classical particle could never get through. But it would probably be more accurate to say: "in places that a classical particle would need exponential time to get through, *sometimes* a quantum particle can get through in polynomial time."

In terms of interference, we can say, "The paths that involve the particle not going over the spike interfere destructively and cancel each other out, leaving only paths where the particle does get over it."

The phenomenon of tunneling is important in many places in physics. For one thing, it's why the sun can shine.

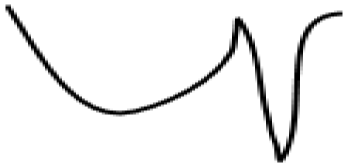


Nuclear fusion requires hydrogen atoms to get super close for them to realize that it's energetically favorable for them to fuse. The trouble is, when they're not quite so close, they also have strong repulsion (because both nuclei are positive).

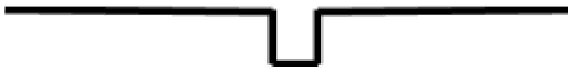
When quantum mechanics came along, it explained that while the energy barrier would prevent fusion classically, it still happens because the nuclei are able to tunnel past the barrier.

Anyway, the 2002 paper of Farhi et al. was good news for the prospects of using the adiabatic algorithm, but tunneling only helps if the spike is sufficiently thin.

Since then, we've learned more about the types of fitness landscapes for which the adiabatic algorithm is expected to help.

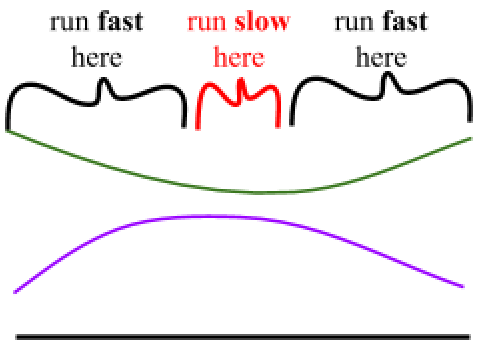


In a landscape like the one pictured on the left, simulated annealing and the adiabatic algorithm would both have trouble and would both take an exponential amount of time.



Or consider the fitness landscape on the right (we can only draw 1 dimension, but imagine that all of the 2^n solutions in an n-dimensional hypercube have equal values except for the one good solution). This would also take exponential time for both simulated annealing and the adiabatic algorithm to traverse.

We actually already know this by the BBBV Theorem---because in this case, we're effectively just querying a black box in an attempt to find a unique marked item.



It turns out that if you're clever about how you run the adiabatic algorithm, you can achieve the Grover speedup in the case above, but not anything faster.

Indeed, the BBBV Theorem tells us that $\Omega(2^{n/2})$ steps are needed, using the adiabatic algorithm or any other quantum algorithm.

What's cool is that just by knowing BBBV (without any physics), you can say that this decrease has to be exponential.

Just like the expert who Farhi consulted was alluding to with his wisecrack, physicists can use knowledge from quantum algorithms to learn new things about spectral gaps.

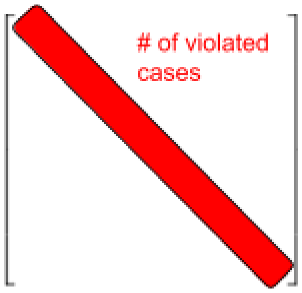
OK, here's a subtler question:

Suppose the adiabatic algorithm had worked to solve 3SAT in polynomial time. Would that have violated the BBBV Theorem?

It turns out the answer is no.

$$\begin{bmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && 1 &&&&& \\ &&& 0 &&&& \\ &&&& 1 &&& \\ &&&&& 1 && \\ &&&&&& 1 & \\ &&&&&&& 1 \end{bmatrix}$$

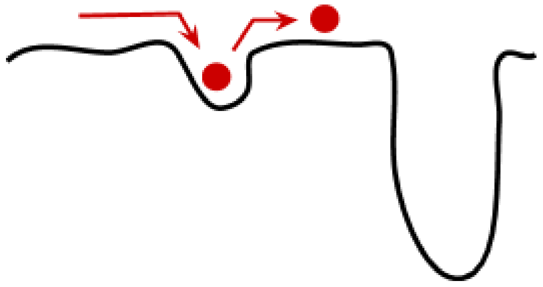
The BBBV Theorem applies only to **black-box** search--which, in the context of the adiabatic algorithm, would mean a diagonal Hamiltonian like the one on the left.



While the Hamiltonian encoding a 3SAT instance (right) is also diagonal, it contains richer information than the black box considered by BBBV. In particular, it encodes not merely whether each possible solution is satisfying or unsatisfying, but the *number* of clauses that it violates. And a 2002 paper of van Dam, Mosca, and Vazirani showed that that information, alone, is enough to reconstruct the 3SAT instance in polynomial time---and hence also to *solve* the instance in polynomial time, if we assumed (for example) that **P**=**NP**! This means that there's no hope of proving a black-box lower bound a la BBBV in this setting.

We also know classical algorithms that can solve 3SAT in less than $2^{n/2}$ time---indeed, about $O(1.3^n)$. This is another way of seeing that the BBBV Theorem can't encompass everything that it's possible to do on 3SAT.

OK, let's consider one more type of fitness landscape. A funny thing happens with landscapes like the one pictured below: simulated annealing gets into the local minimum, but the algorithm then escapes it, crosses the plateau, and reaches the global minimum in polynomial time. Meanwhile, the adiabatic algorithm just keeps returning to the local minimum, and takes exponential time to reach the global minimum!



Sometimes a classical algorithm performs better.

But there's even a further problem with establishing quantum speedups for adiabatic optimization, which is that "classical computing is a moving target." A classical computer is not limited to simulated annealing or any other specific algorithm. Thus, even if we established that the adiabatic algorithm was exponentially faster than simulated annealing for some class of "real-world" fitness landscapes, we'd still need to compare against *other* classical algorithms, which might exploit detailed knowledge about the landscapes in question. Particularly relevant here is Quantum Monte Carlo (QMC)---which, despite its name, is a quantum-inspired algorithm that runs on a classical computer, and which is widely used in many-body physics. We won't explain QMC in any detail, but will simply say that, even in the artificial cases where the adiabatic algorithm exponentially outperforms simulated annealing, recent work suggests that QMC can typically match the adiabatic algorithm's asymptotic performance classically (albeit, often with a much larger constant prefactor).

So, what about the fitness landscapes that arise in real-world, industrial applications? Do there or don't there exist any that the adiabatic algorithm can traverse exponentially faster than any classical algorithm?

The truth is, we really don't know yet. Reaching a consensus on this might require building a scalable quantum computer and then testing the algorithm out!

Which brings us to our final point about the adiabatic algorithm. We've talked about implementing the adiabatic algorithm on a conventional, gate-based quantum computer, by using Trotterization to approximate Hamiltonians by discrete sequences of gates. But you might complain that that approach seems roundabout. Since the underlying physics that describes our qubits is based on Hamiltonians anyway, why not just directly map the adiabatic algorithm onto the continuous-time evolution of the qubits, and skip the Trotterization part?

Indeed, the adiabatic algorithm could be seen not only as an algorithm, but also as a proposal for the physical implementation of quantum computers. An important 2004 result of Aharonov et al. says that adiabatic quantum computers would be universal for quantum computation: that is, able to solve all **BQP** problems efficiently.

There's a venture-capital-backed startup called D-Wave that's been building special-purpose devices to implement a noisy approximation to the adiabatic algorithm (called quantum annealing), using physical Hamiltonians themselves. D-Wave's latest model has about 2000 superconducting qubits. You can encode an optimization problem of your choice onto their chip, by choosing the interaction Hamiltonian for each pair of neighboring qubits.

D-Wave was all over the press because they actually sold a few of their machines, to companies like Google and Lockheed Martin, and were notorious for claiming that quantum computing is "already useful in practice." They were even on the cover of *Time* magazine!
Professor Aaronson has been to D-Wave's headquarters.
Funnily enough, their machine is *literally* a room-sized black box (most of the hardware inside the box is devoted to cooling; the actual qubits are on a chip no larger than an ordinary computer chip).

So what's the verdict? Experimental data shows that the D-Wave device is indeed able to solve optimization problems encoded in its special format, at a speed that's often competitive with the best known classical algorithms. Unfortunately, results over the past five years do not clearly show any quantum *speedup* over the best classical algorithms.

Why not?
Roughly speaking, there are three main possible causes for the lack of speedup on D-Wave's current devices. As far as we know today, the truth might be any combination of them.

1) *Inherent limitations of the adiabatic algorithm.*
Even if we had a perfect quantum computer, running at absolute zero, the minimum spectral gaps might simply be exponential small for almost all interesting optimization problems---in which case, the adiabatic algorithm could only provide limited speedups, even in theory.

2) *Limitations in the quality of D-Wave's qubits*.
Even if the minimum spectral gap were inverse-polynomial, it still probably wouldn't be *constant*: that is, it would presumably shrink to zero as a function of the input size. And this is a problem for the following reason. It turns out that, if the *temperature* of the qubits exceeds the minimum spectral gap, then we'll typically see level crossings even if we run the adiabatic algorithm at the "right" speed.
The D-Wave qubits are cooled to about 10 milliKelvin. By normal standards, that sounds extremely cold, but it might not be cold *enough* to avoid level crossings on instances of interesting sizes! What one really wants here is absolute zero---but of course, the cost would increase enormously as one approached that, eventually becoming prohibitive.

3) *Stoquastic Hamiltonians.*
A Hamiltonian H is called stoquastic if all its off-diagonal terms are real and non-positive. One can show that, if H is stoquastic, then H has a ground state involving nonnegative real amplitudes only. It turns out that, for reasons stemming from that fact, stoquastic Hamiltonians are often much easier than arbitrary Hamiltonians to simulate on a classical computer. For example, the QMC algorithm mentioned earlier tends to work extremely well for approximating the ground states of stoquastic Hamiltonians, and less well for non-stoquastic Hamiltonians.
Unfortunately, D-Wave's hardware is currently limited to applying stoquastic Hamiltonians only. Thus, even if quantum annealing were able to yield an exponential speedup at a fixed nonzero temperature (like 10 milliKelvin), it might be that it could only do so with non-stoquastic Hamiltonians.

This sounds pretty bad! Given, especially, the temperature issue, why is anyone optimistic that quantum computing could scale even in principle? We'll see why in the next lecture, when we explore the basics of quantum error-correction: a technique that D-Wave is not currently using, but that many researchers expect will ultimately be necessary for scalable QC.