Intro to Quantum Computing

Part 3: Adiabatic quantum computing and quantum annealing

Olivia Di Matteo

Quantum Information Science Associate, TRIUMF

Last week

Do you have any questions about gate-model quantum computing?

This week

Today: theoretical background

- Brief introduction to adiabatic quantum computing
- Simulated and quantum annealing
- The transverse-field Ising model
- D-Wave hardware
- Optimization problems, Ising Hamiltonians, and QUBOs

Tomorrow: technical details and applications

- Minor embeddings
- Solving a simple graph theory example on the QPU
- Classifying Higgs decay signals
- Particle track reconstruction
- High-level discussion about quantum machine learning

Gate model quantum computing

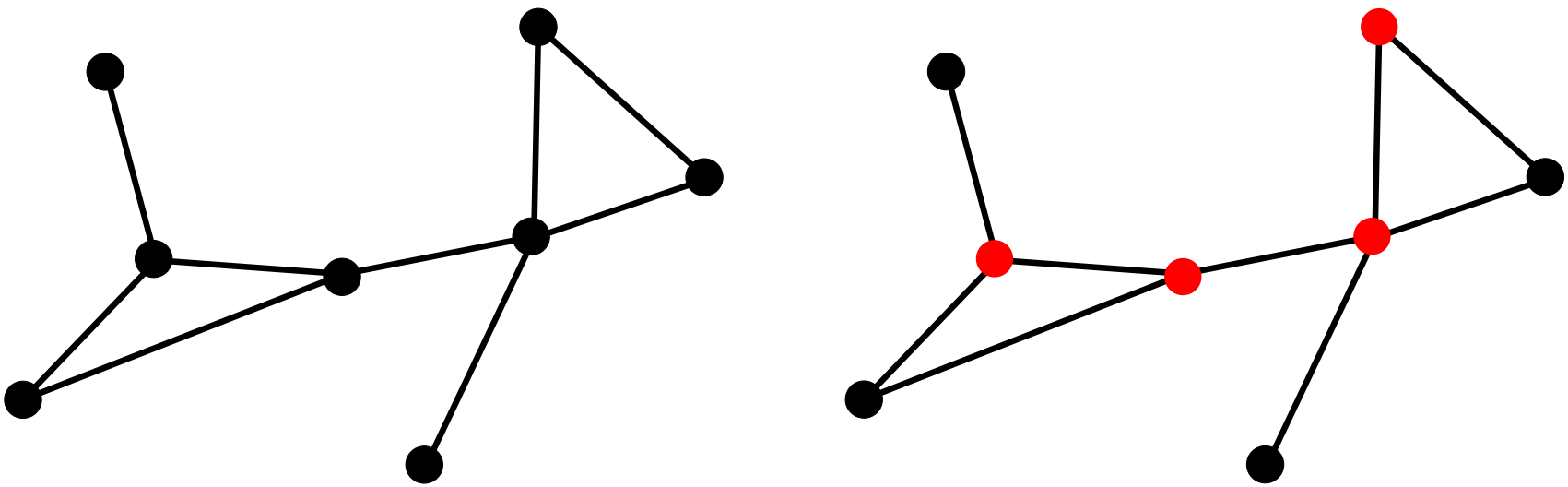
Quantum circuits provide us with a very intuitive way to think about quantum computation.

1. Prepare input state
2. Perform unitary operations
3. Measure output state

But this is not intuitive for every type of problem.

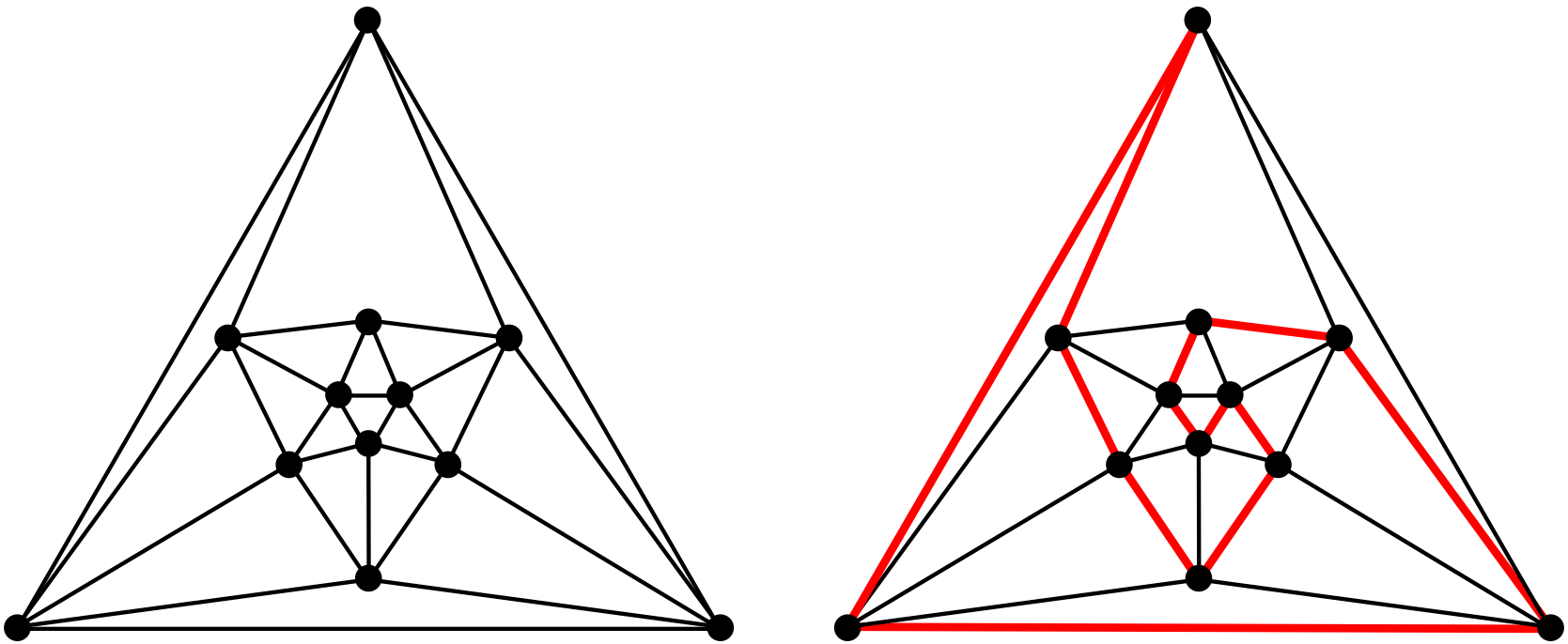
Problem: vertex cover

Given a graph $G = (V, E)$, what is the *smallest number of vertices* you can colour such that every edge in the graph is attached to at least one coloured vertex?



Problem: Hamiltonian cycles

Given a graph, can we find a path through it that visits every node *exactly once* and returns to the starting point?



(In graph theory terms, can we find a *Hamiltonian cycle*?)

Problem: financial portfolio optimization

You have the opportunity to purchase 100 units of stocks from a fixed list of assets. You know the average returns of each stock, and their covariances.

Stock	Avg. return
AAA	3.44 %
BBB	2.21 %
CCC	-0.28 %
⋮	⋮

Cov.	AAA	BBB	⋯
AAA	0.0038	0.002	⋯
BBB	0.002	-0.006	⋯
CCC	0.014	-0.0008	⋯
⋮	⋮	⋮	⋱

Suppose you're restricted to buying no more than 5 of any stock.

Which stocks, and how many of each, should you purchase, to maximize your profits?

Optimization problems and quantum computing

These are all examples of optimization problems, with varying types of constraints:

- Colour the *fewest* number of vertices
- Find a path that hits every vertex *once*
- Choose a portfolio that *maximizes profits*

Many classical optimization problems are *NP-complete*[1], and thus hard for classical computers to solve.

[1] Loosely, exponentially hard to solve, but easy to verify if a solution is valid.

Optimization problems and quantum computing

All of these problems can be solved using quantum computing!

Maybe a quantum computer can even solve them faster.

...But it's not so obvious how to formulate them in the gate model.

Instead we will look to *adiabatic quantum computing*, and specifically *quantum annealing*.

Adiabatic quantum computing (AQC)

AQC is a model that can very naturally encode and solve these kinds of *optimization problems*.

Recall that every unitary U is directly related to a Hermitian Hamiltonian H under the correspondence

$$U = e^{-iHt} \tag{1}$$

Last week we saw how the gate model can be used to *simulate* the evolution of a Hamiltonian.

Instead of simulating the Hamiltonians, AQC works with them directly to perform computations.

Hamiltonians and optimization problems?

The structure of a classical optimization problem is something like:

$$\min_{\vec{x}} \ \text{cost}(\vec{x}) \quad \text{subject to constraints}(\vec{x}) \tag{2}$$

where $\vec{x}$ is a multi-dimensional vector of parameters in the problem space.

In a physical context, optimization can be interpreted as an energy minimization problem.

Optimization	**Physical system**
$\vec{x}$	State of the system
$\text{cost}(\vec{x})$	Hamiltonian
Optimum $\vec{x^*}$	Ground state
$\text{cost}(\vec{x^*})$	Ground state energy

Adiabatic quantum computing

1. Design a Hamiltonian whose ground state represents the solution to our optimization problem
2. Prepare a system in the ground state of an easy-to-prepare driving Hamiltonian
3. Perform adiabatic evolution to transform the system from the ground state of the driving Hamiltonian to the ground state of the problem Hamiltonian, which is our solution

The adiabatic theorem

Why would we want to do this?

Theorem:

"A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum."

What we can take from this:

If we initialize a system in the lowest energy state and perturb it slowly enough, it will remain in the lowest energy state (with respect to the changed system)

Adiabatic evolution

Let H_d be a *driving Hamiltonian* whose ground state can be easily initialized in your system.

Let H_p be a *problem Hamiltonian* whose ground state represents the solution to a problem of interest.

Adiabatic evolution is expressed mathematically as the function

$$H(s) = A(s)H_d + B(s)H_p \tag{3}$$

The parameter s is representative of time; s goes from 0 to 1; $A(s)$ decreases to 0 with time and $B(s)$ increases from 0.

Adiabatic quantum computing

This leads to lots of questions, which we will work through in varying order today:

- Is this really quantum computing?
- Where does the quantum hardware come in?
- How do we express optimization problems as Hamiltonians?
- Which driving Hamiltonian should we choose?
- How slow is 'slowly enough'?

Is this really quantum computing?

Yes.

AQC and the gate model are *equivalent up to polynomial overhead*.

AQC ⇒ gate model

Straightforward: perform Hamiltonian simulation of the adiabatic evolution process.

Gate model ⇒ AQC

Non-trivial: need to design a Hamiltonian whose ground state will be the same as the output state as the one we get at the end of the circuit model computation

For more information and a proof, see: `http://arxiv.org/abs/1611.04471`

Where does the quantum hardware come in?

D-Wave (in Burnaby!) has created specialized hardware to perform *quantum annealing*. One thing their machines can do is solve a specific class of optimization problems called QUBOs.

Image credit:
`www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware`

The 2D Ising model and quantum annealing

Hamiltonian structure

Before we discuss how to convert problems to Hamiltonians, we need to first understand what kinds of Hamiltonians the D-Wave is capable of implementing.

We are going to focus on *multi-qubit* Hamiltonians.

Recall that the single-qubit Pauli operators

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

form a *basis* for the 2×2 Hermitian matrices, i.e. for 2×2 H,

$$H = c_i \mathbb{1} + c_x \sigma^x + c_y \sigma^y + c_z \sigma^z \tag{4}$$

for some coefficients c_k.

Hamiltonian structure

Similarly, the n-qubit Pauli operations are a basis for $2^n \times 2^n$ Hermitian matrices. So any multi-qubit Hamiltonian can be expressed as a sum of tensor products of Paulis:

$$H = \sum_k p_k P_k, \tag{5}$$

The Paulis will look something like this...

$$P_k = \sigma^x \otimes \sigma^z \otimes \cdots \otimes \sigma^y \otimes \mathbb{1} \tag{6}$$

To simplify notation, let σ_i^α represent σ^α, $\alpha \in \{x, y, z\}$ acting on the i'th qubit, and $\mathbb{1}$ on everything else. For example,

$$\sigma_1^x \sigma_2^y \sigma_4^y \sigma_6^z = \sigma^x \otimes \sigma^y \otimes \mathbb{1} \otimes \sigma^y \otimes \mathbb{1} \otimes \sigma^z \tag{7}$$

Hamiltonian structure

In general AQC, the following form of Hamiltonian is universal:

$$H = -\sum_i h_i \sigma_i^z - \sum_{ij} J_{ij}^z \sigma_i^z \sigma_j^z - \sum_i k_i \sigma_i^x - \sum_{ij} J_{ij}^x \sigma_i^x \sigma_j^x \qquad (8)$$

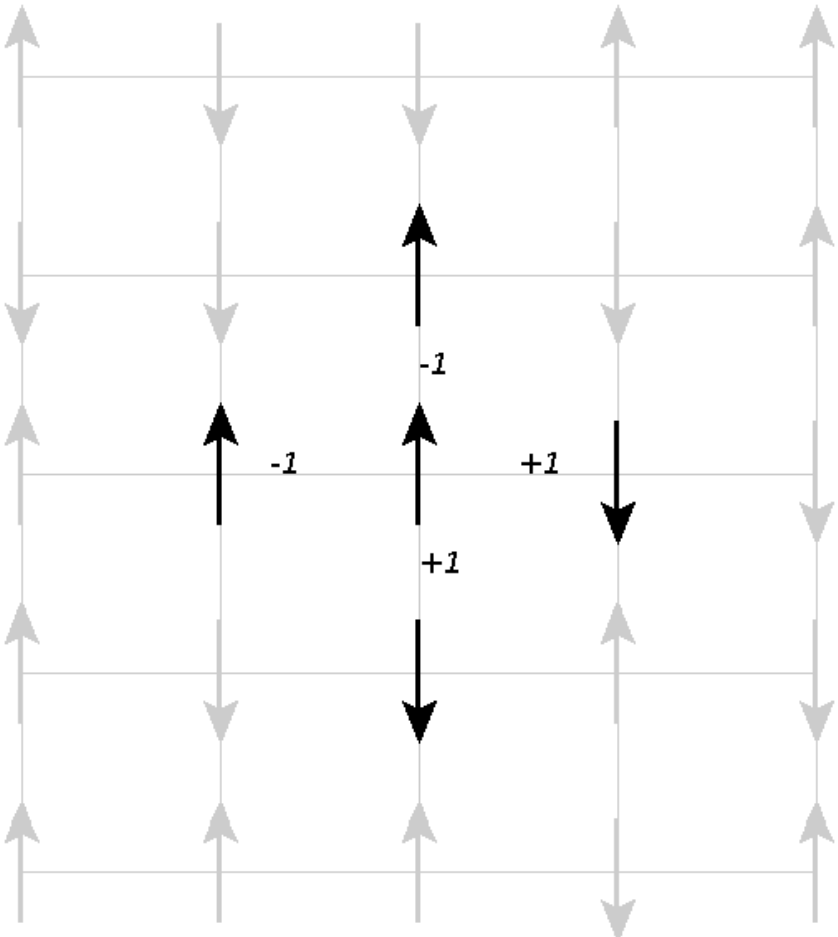
Quantum annealing on the D-Wave uses a more limited Hamiltonian:

$$H = -\sum_i h_i \sigma_i^z - \sum_{ij} J_{ij} \sigma_i^z \sigma_j^z - \sum_i k_i \sigma_i^x \qquad (9)$$

... look familiar?

Physical intuition: the 2D Ising model

Used to study phase transitions of magnetic materials.

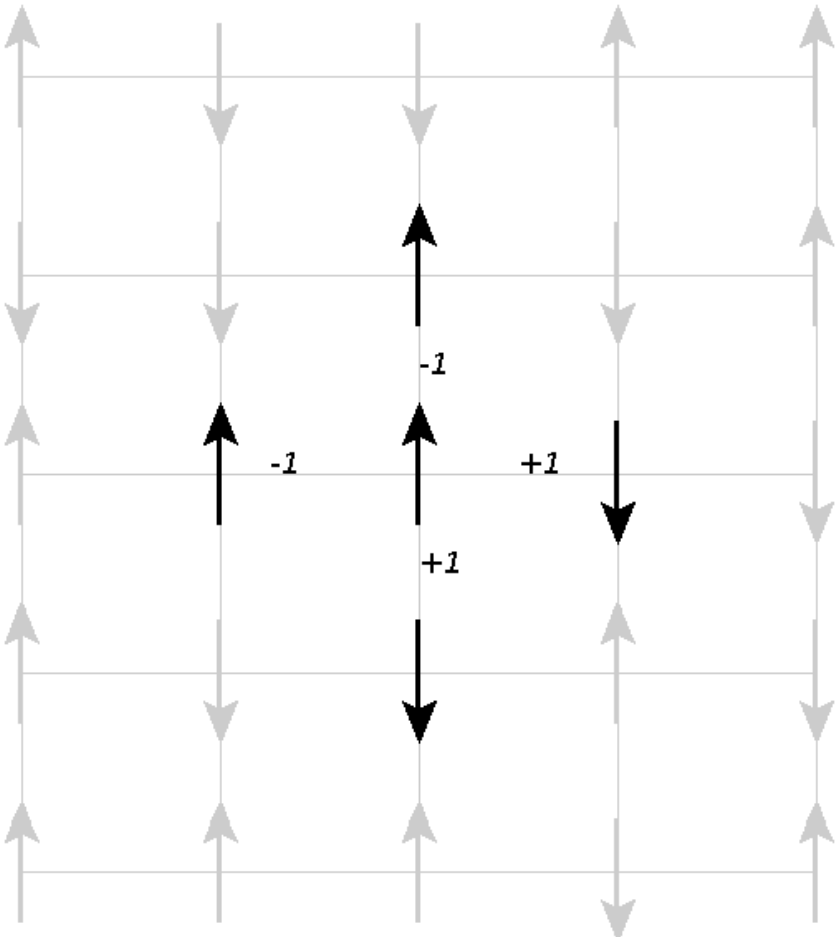


Lattice of spins with nearest-neighbour interaction. There are

- coupling terms J_{ij} to represent strength of interaction between pairs of spins i, j
- different external magnetic field h_i at each spin

Image credit: http://cp3-origins.dk/content/movies/2013-10-25-1200-arthur.pdf/images/grid3.png

Physical intuition: the 2D Ising model



The system Hamiltonian is

$$H = -\sum_{\langle i,j \rangle} J_{ij} \sigma_i^z \sigma_j^z - \sum_i h_i \sigma_i^z$$

The energy of the system is

$$E = -\sum_{\langle i,j \rangle} J_{ij} s_i s_j - \sum_i h_i s_i$$

where $s_i = \pm 1$ represent the up/down states of the spins.

What configuration of spins minimizes the total energy?

Image credit: http://cp3-origins.dk/content/movies/2013-10-25-1200-arthur.pdf/images/grid3.png

Solving the 2D Ising model

Suppose for now that $h_i = 0$ at every site. The energy of the system for a particular assignment of spins $\{s_i\}$ is given by

$$E = -\sum_{\langle i,j \rangle} J_{ij} s_i s_j \tag{10}$$

Two familiar cases...

- All $J_{ij} = 1$: lowest energy when all spins are pointing in the same direction. This is *ferromagnetism*.
- All $J_{ij} = -1$: spins will alternate in a checkerboard pattern. This is *antiferromagnetism*.

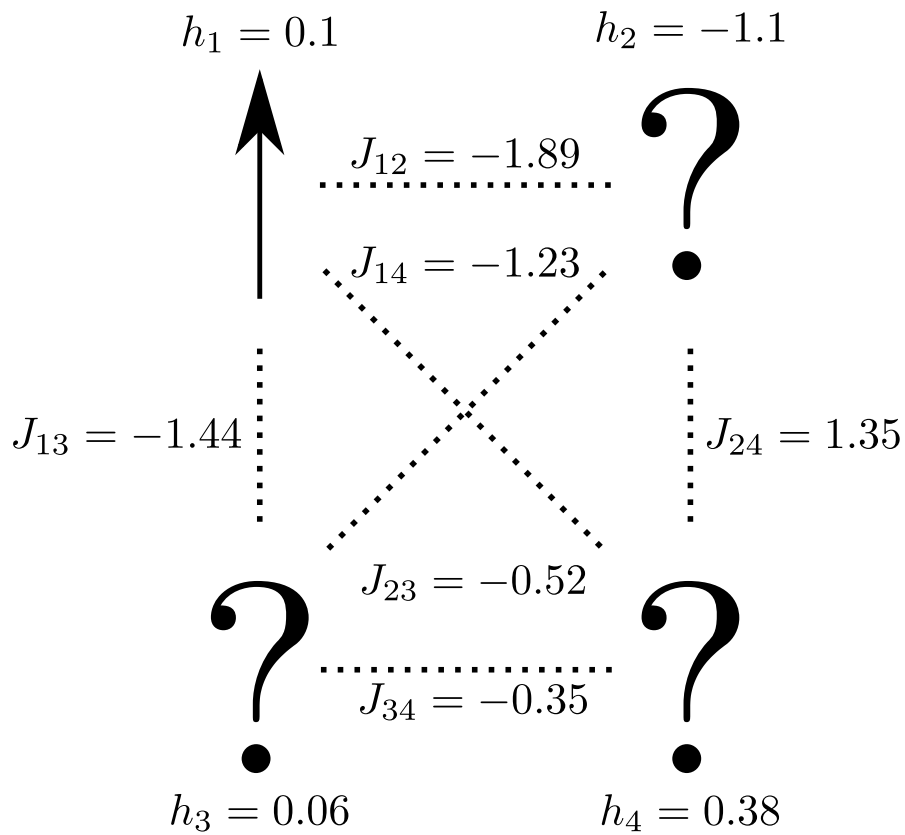
Frustration in the 2D Ising model

But what if J_{ij} take a mix of different positive and negative values?

What if we are no longer limited to nearest-neighbour interactions?

What if there is a different external field at each site?

This system is *frustrated*.

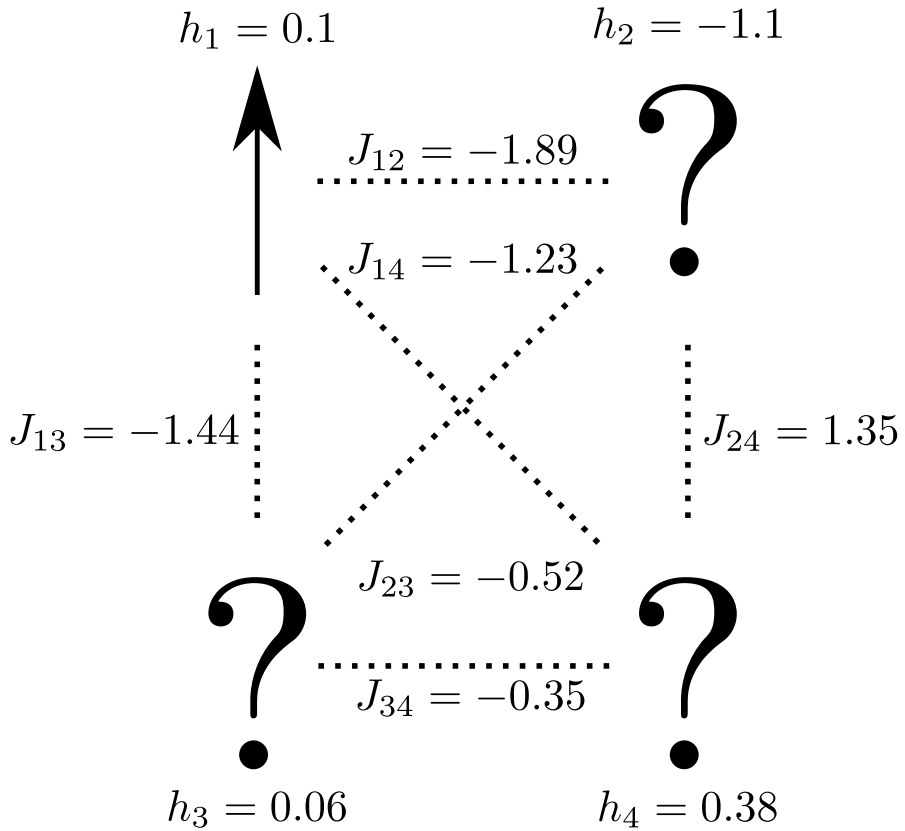


Frustration in the 2D Ising model

If there are N spins, then there are 2^N possible configurations of the system; it is not tractable to check all of these and see which one has the lowest energy!

Instead, we rely on heuristic algorithms to explore the space of possible configurations until it reaches a minimum.

A popular way of doing this is using *simulated annealing*.

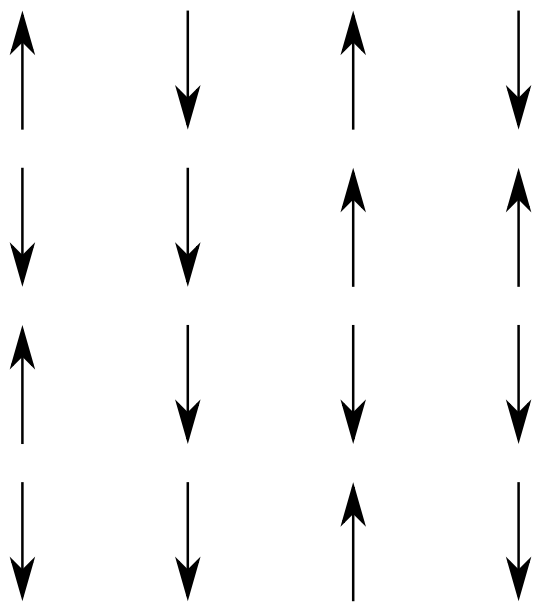


Not-unrelated tangent: simulated annealing (SA)

SA is a probabilistic algorithm. We start at a high temperature T_i, and gradually cool to a low temperature T_f, where we hope the system 'freezes' in the ground state.

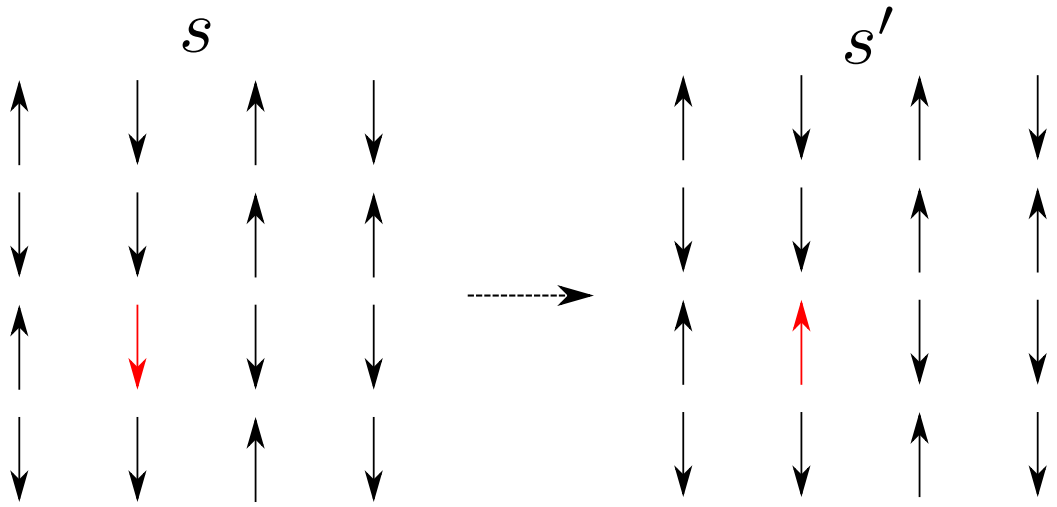
Start by choosing a random assignment of the spins at a 'high' temperature T_i:

$$s$$



Not-unrelated tangent: simulated annealing (SA)

Next, we randomly choose a spin, and look at how the energy would change if we were to flip that spin.



If $E_{s'} < E_s$, the new state is energetically favourable, so we transition to it.

If $E_{s'} > E_s$, we transition to the new state with probability

$$\Pr(s \to s') = e^{-\frac{(E_s - E_{s'})}{k_B T}} \tag{11}$$

Not-unrelated tangent: simulated annealing (SA)

This process is repeated some fixed number of times; then, we lower the temperature and repeat.

Cooling is done according to an *annealing schedule*. For example, this may be linear, e.g.

$$T(t) = (1 - t)T_i + tT_f \tag{12}$$

where t represents some fraction of a total time, and increases from 0 to 1.

Not-unrelated tangent: simulated annealing (SA)

The transition probability to unfavourable energy states is:

$$\Pr(s \to s') = e^{-\frac{(E_s - E_{s'})}{k_B T}}$$

When the temperature is higher: we are *more* likely to transition to *unfavourable* states - this allows us to broadly explore the space of configurations.

As the temperature decreases: we begin to work more and more locally, usually only making transitions when the energy of the candidate state is lower.

Demonstrations of SA

See here for an example:

`https://upload.wikimedia.org/wikipedia/commons/d/d5/Hill_Climbing_with_Simulated_Annealing.gif`

(Note this is an example for finding a maximum, and not a minimum.)

How does this look on a 2D lattice? Video time!

Not-unrelated tangent: simulated annealing (SA)

Important note: SA is probabilistic, and therefore we are *not guaranteed* to find the ground state.

However, in practice it works quite well, and you can *sample* by running it multiple times, and then choose the lowest-energy state from all your outcomes.

There are a number of tunable parameters, such as the cooling schedule and number of spin flips at each temperature.

From simulated annealing to quantum annealing

Why did I explain all this to you?

Recall earlier I defined adiabatic evolution:

$$H(s) = A(s)H_d + B(s)H_p \tag{13}$$

This is analogous to the simulated annealing temperature schedule

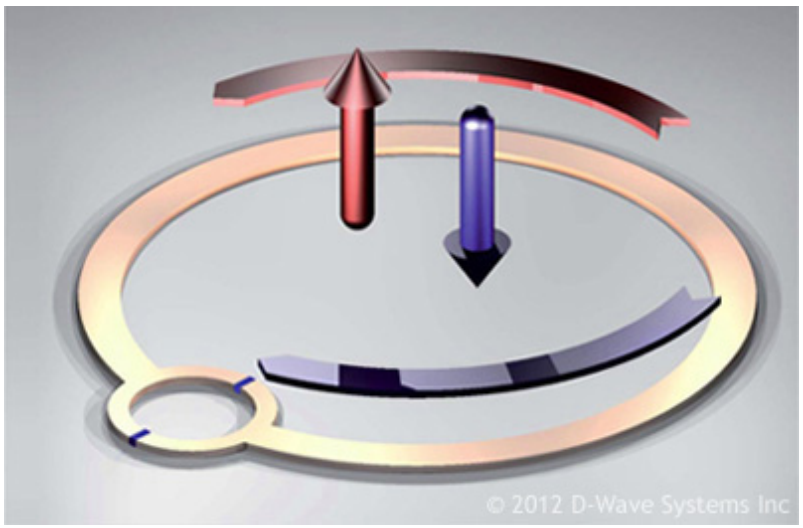
$$T(t) = (1 - t)T_i + tT_f \tag{14}$$

The D-Wave is performing a *quantum version of annealing*, adjusting the relative strengths of the Hamiltonians rather than changing a temperature.

D-Wave hardware

Superconducting qubits

D-Wave qubits are a specific type of *superconducting flux qubit* called a SQUID (Superconducting QUantum Interference Device).



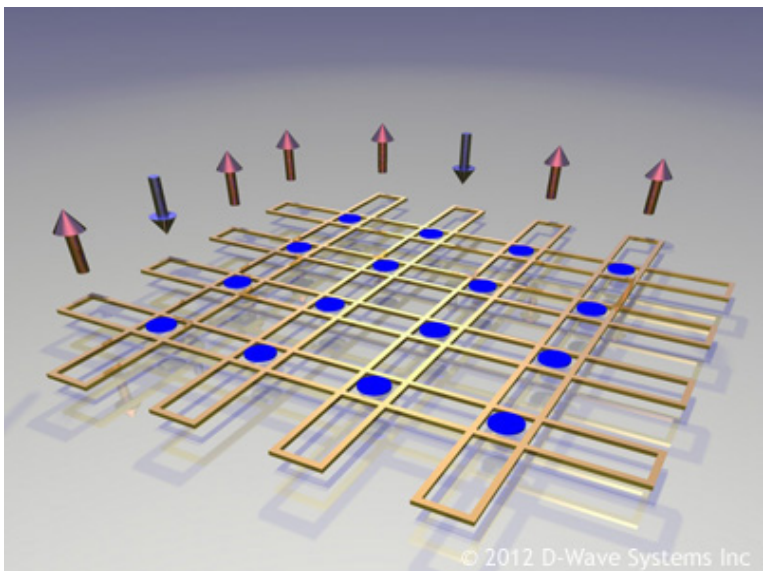
The qubit is a loop of *niobium*, which is superconducting when sufficiently cold.

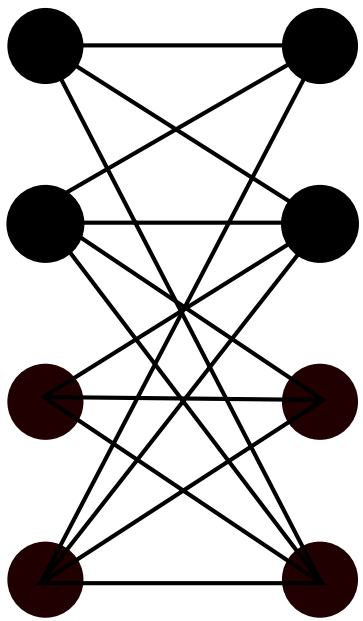
The current through the loop creates a magnetic flux inside that points either up or down - these are the two states of the qubit.

Image credit:
`https://www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware`

Coupling qubits

The processor is partitioned into 8-qubit unit cells. We can couple together qubits whose current loops overlap. A single unit cell is a complete bipartite graph $K_{4,4}$.

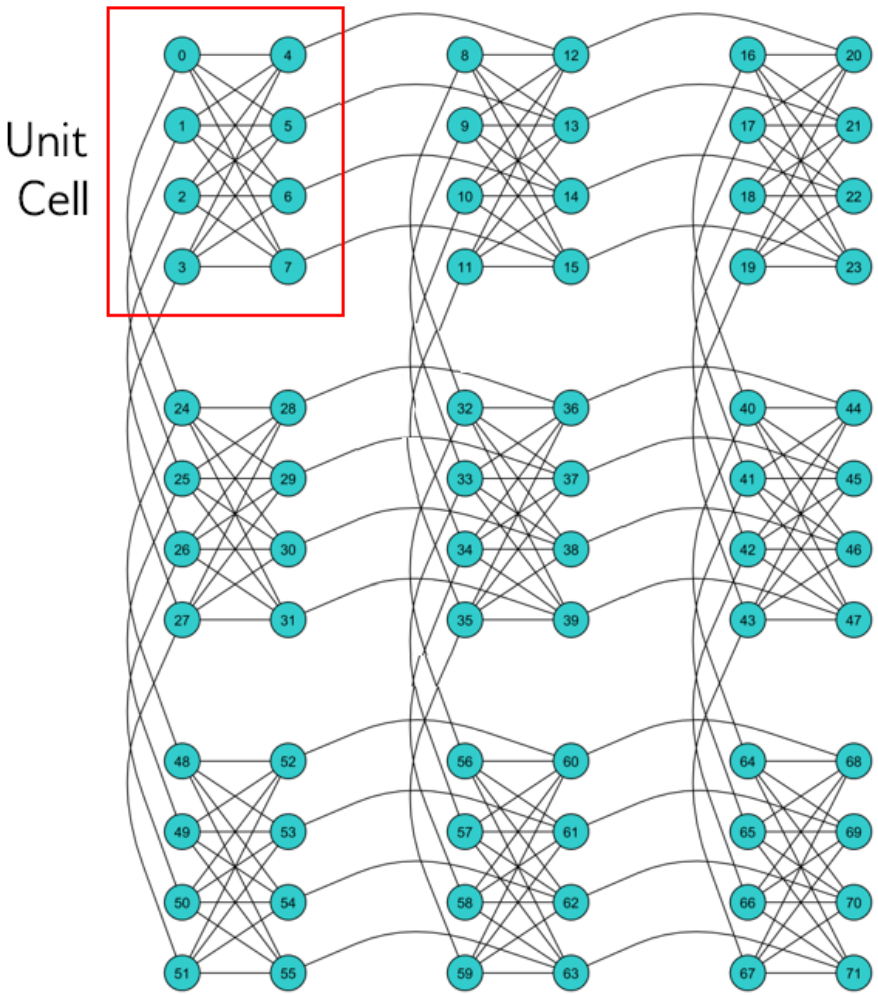




Both the coupling strengths and external biases are *tunable*.

Image credit:
`https://www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware`

Coupling qubits



The D-Wave QPU consists of an array of $K_{4,4}$ cells additional couplings between cells.

This is called the *Chimera graph*.

Each qubit is coupled with up to 6 others.

Image credit:
`https://docs.dwavesys.com/docs/latest/_images/chimera.png`

Current generation: D-Wave 2000Q

2048 qubits arranged in a Chimera graph structure.

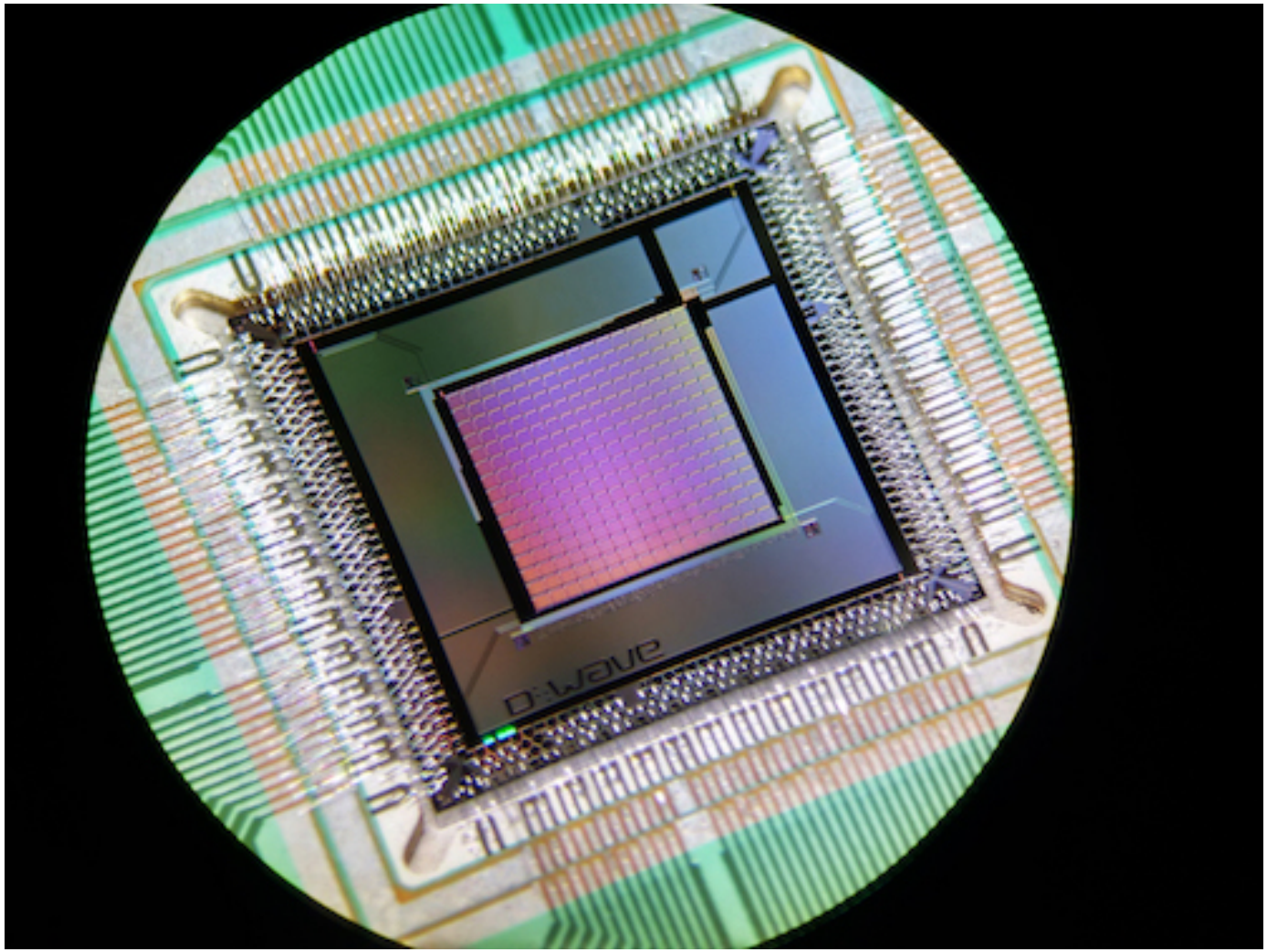


Image credit:
`https://www.dwavesys.com/resources/media-resources`

Hamiltonians for quantum annealing

Consider again:

$$H(s) = A(s)H_d + B(s)H_p \tag{15}$$

H_p is the problem Hamiltonian, and it is an Ising Hamiltonian corresponding to our problem:

$$H_p = -\sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z - \sum_i h_i\sigma_i^z \tag{16}$$

H_d is the *driving Hamiltonian*, and it is the initial state. What should we choose?

Hamiltonians for quantum annealing

Our Hamiltonians **must** be chosen so that they do not commute:

$$[H_d, H_p] \neq 0 \tag{17}$$

Why??

A couple reasons...

If they commute,

$$[H_d, H_p] = 0 \tag{18}$$

then H_d and H_p have the *same eigenstates*, with *shifted energy values*.

Let $|\psi\rangle$ be the ground state of H_d,

$$H_d|\psi\rangle = E_g|\psi\rangle \tag{19}$$

This state is also an eigenstate of H_p, but it is an excited state:

$$H_p|\psi\rangle = E_e|\psi\rangle \tag{20}$$

As we evolve...

$$\begin{aligned} H(s)|\psi\rangle &= A(s)H_d|\psi\rangle + B(s)H_p|\psi\rangle \\ &= A(s)E_g|\psi\rangle + B(s)E_e|\psi\rangle \\ &= (A(s)E_g + B(s)E_e)|\psi\rangle \end{aligned}$$

We never actually leave the eigenstate $|\psi\rangle$! But at the end of the calculation when $s = 1$, $A(s)$ goes to 0, and we're left with

$$H(1)|\psi\rangle \approx B(1)H_p|\psi\rangle = B(1)E_e|\psi\rangle \tag{21}$$

which recall is an excited state of the Hamiltonian, and not the ground state.

Furthermore, recall the adiabatic theorem...

Theorem:

"A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum."

Furthermore, recall the adiabatic theorem...

Theorem:

"A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and **if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum.**"

Hamiltonians for quantum annealing

There *must* be a non-zero gap between the energies of the ground and excited states of

- The driving Hamiltonian
- The problem Hamiltonian
- Every possible Hamiltonian you encounter during the course of your evolution

Addition of a non-commuting H_d to H_p ensures that such zero gaps, or *level crossings*, are avoided.

The driving Hamiltonian we choose is

$$H_d = -\sum_i \sigma_i^x \tag{22}$$

This does not commute with any Ising Hamiltonian, and its ground state is the *uniform superposition*, which is a quantum state.

This state is nice and easy to prepare in hardware - just apply a constant magnetic field in the x direction!

In quantum annealing, all qubits begin in a uniform superposition...

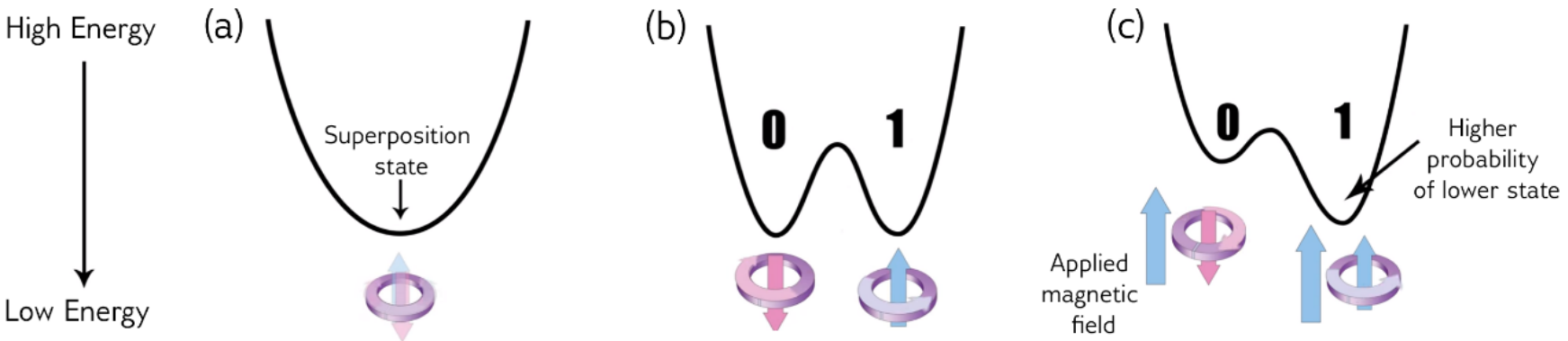


Figure 2.2: Energy diagram changes over time as the quantum annealing process runs and a bias is applied.

External magnetic fields and couplings of H_p are physically applied to each qubit according to

$$H(s) = A(s)H_d + B(s)H_p \tag{23}$$

Image credit: `https://docs.dwavesys.com/docs/latest/_downloads/09-1076A-T_GettingStarted.pdf`

Applied fields tilt the potential well, changing the relative probabilities of the qubit being measured as 0 or 1.

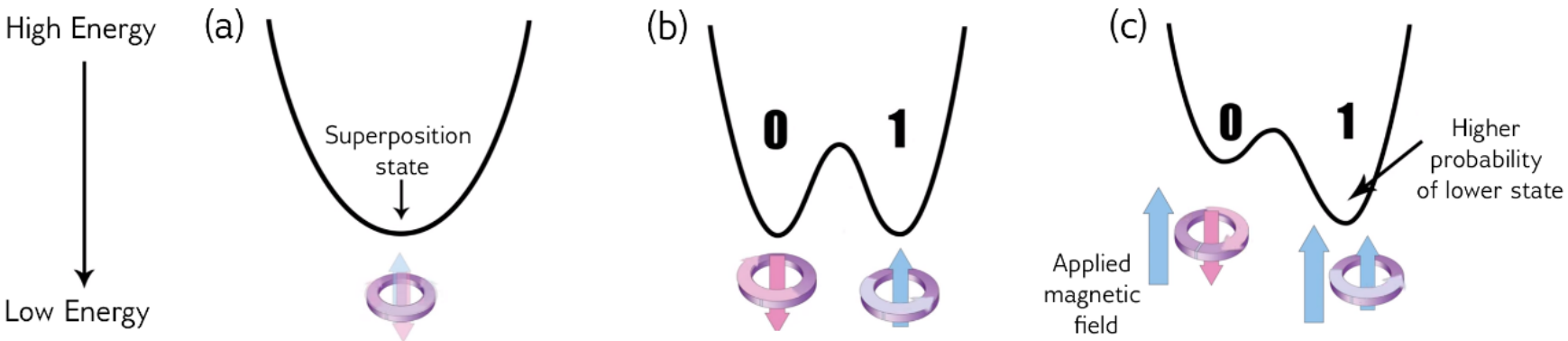


Figure 2.2: Energy diagram changes over time as the quantum annealing process runs and a bias is applied.

At the end of the annealing process, we'll be left in a 'classical' state - just some sequence of 0s and 1s (since the eigenstates of σ^z are the computational basis states).

Image credit: `https://docs.dwavesys.com/docs/latest/_downloads/09-1076A-T_GettingStarted.pdf`

How slow is 'sufficiently slow'?

The annealing speed ultimately depends on the energy gap Δ, i.e. the difference between the ground state and first excited state energies:

$$\Delta(H) = E_e(H) - E_g(H) \tag{24}$$

Small gaps are bad - they make it way easier for the state to be lifted from the ground state by noise (e.g. thermal noise).

How slow is 'sufficiently slow'?

We are then bound by a speed limit of roughly

$$\text{Speed limit} \propto \frac{1}{\min(\Delta)^2} \tag{25}$$

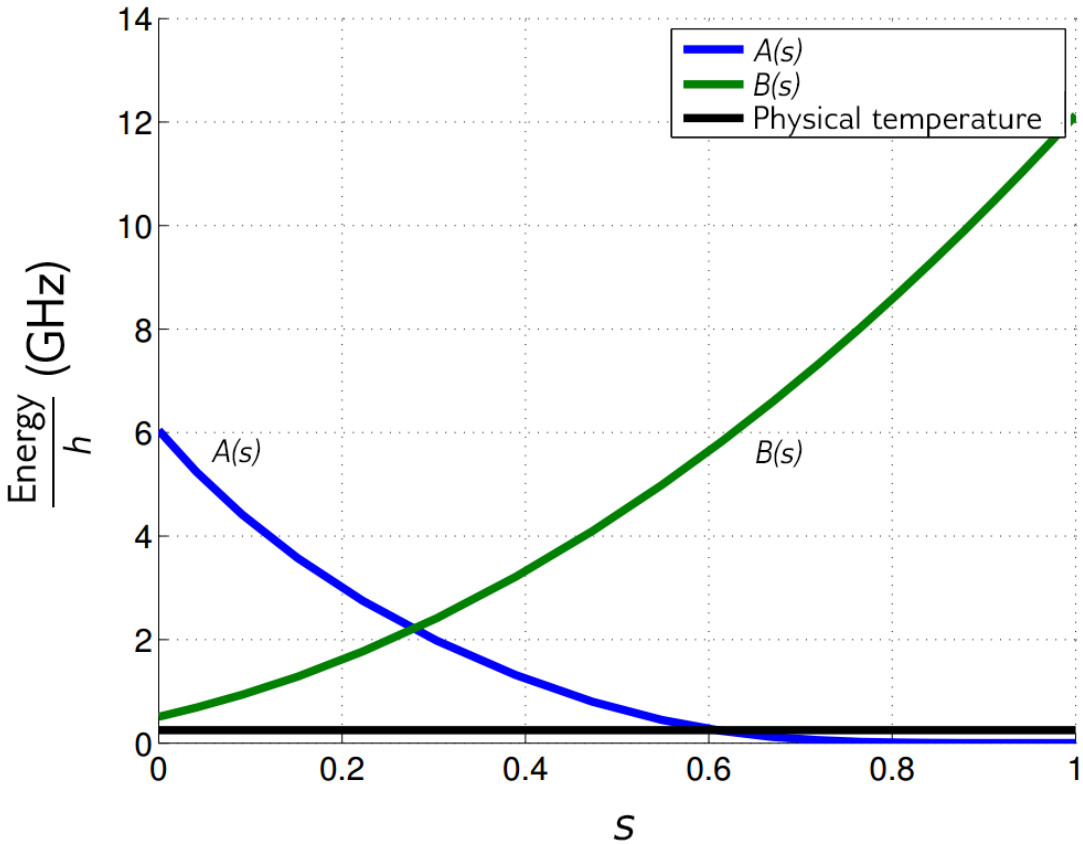
where $\min(\Delta)$ represents the minimum gap over all the Hamiltonians passed through during the annealing process.

However, computing the gap is itself a hard problem. Since we don't know the gaps, what we do instead is *anneal many, many times*, so that we will at some point encounter the ground state with high probability.

For a nice, graphical explanation, see: `https://youtu.be/tnikftltqE0`

D-Wave annealing schedules

In hardware the annealing process takes 20μs. The schedule looks something like this:



(This was for the last-gen D-Wave 2X which had $\sim$ 1000 qubits.)

Image credit: `https://docs.dwavesys.com/docs/latest/c_qpu_0.html`

D-Wave limitations

Like gate-model machines, D-Wave machines have some limitations:

- Thermal noise; must be cooled to $\sim$ 15mK like other contemporary superconducting quantum hardware
- Noise due to impurities in fabrication process (e.g. low-frequency flux noise due to adsorption of oxygen on the surface)
- Qubit coherence times are on the order of ns

D-Wave limitations

- Coupling coefficients have limited ranges; on the D-Wave 2000Q, $h_i \in [-2.0, 2.0]$, $J_{ij} \in [-1.0, 1.0]$. If values are outside that range, the problem must be rescaled, but due to noise we will lose 'resolution' with the smaller numbers
- Like simulated annealing, we are not guaranteed to find the solution, so we have to sample multiple times (default is 1000)
- As mentioned earlier, the type of Hamiltonian we can implement is limited
- Qubit connectivity is limited (more on this tomorrow)

Performance of the D-Wave

There has historically been a *lot* of arguing in the QC community over whether or not the D-Wave is a quantum device.

This has settled down a bit over the last few years.

I take an open standpoint on this, and will make a couple key points backed by the literature.

Q: Is it doing something quantum?, part I: entanglement

A: **Yes.**

Entanglement has been demonstrated between *two- and eight-qubit subsystems* during the QA procedure.

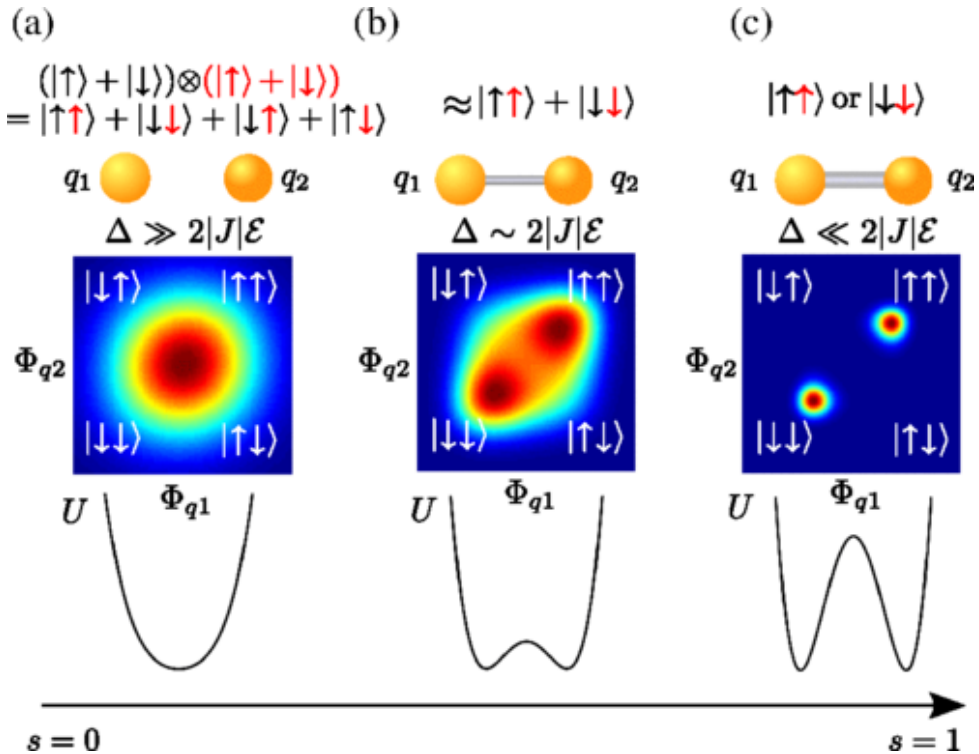


Image credit: `https://journals.aps.org/prx/abstract/10.1103/PhysRevX.4.021041`

Q: Is there a quantum speedup?

A: **Unclear.**

A key reason that we think there should be a speedup is that the system can quantum-mechanically *tunnel* through high-energy barriers during the annealing process.

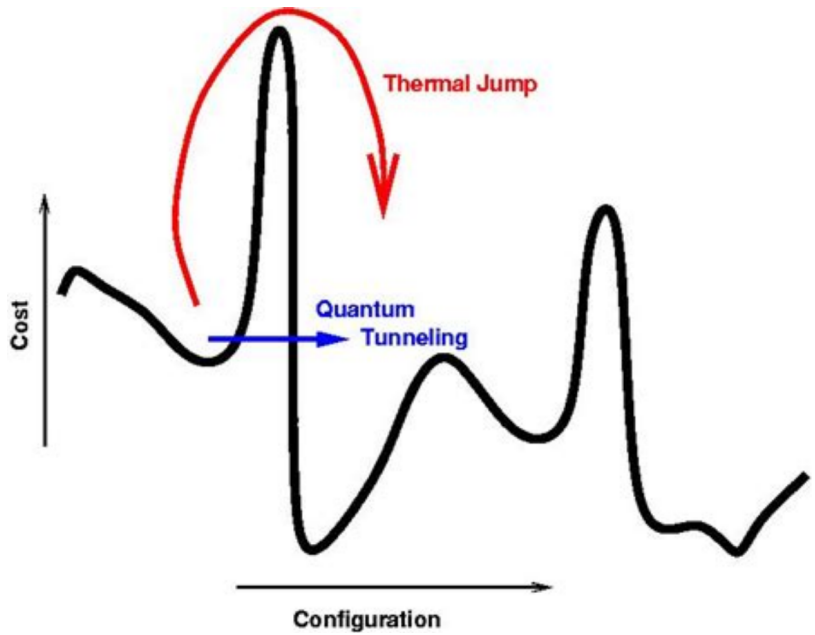
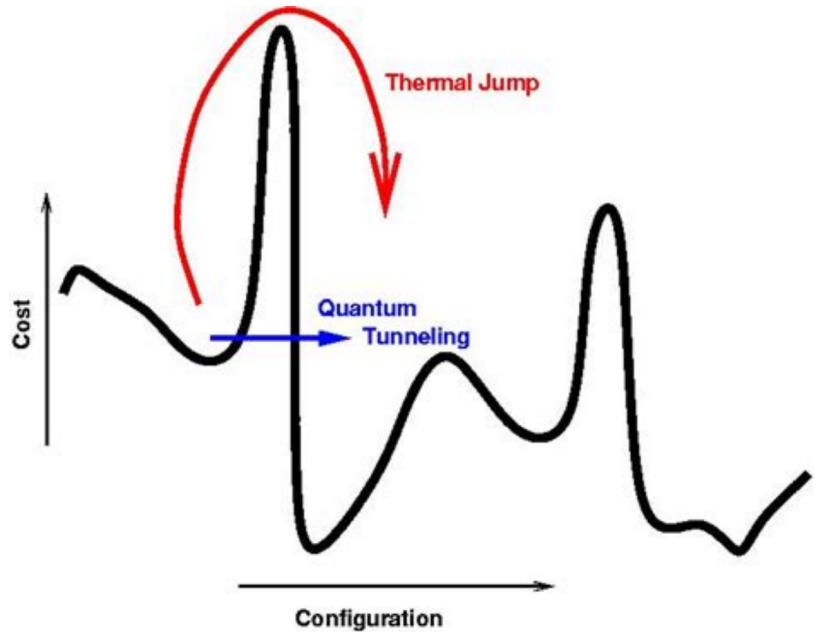


Image credit: `https://commons.wikimedia.org/wiki/File:Quant-annl.jpg`

Q: Is there a quantum speedup?

In simulated annealing, these kinds of jumps occur during the initial high-temperature stage when we are more likely to transition to unfavourable states.



Analogously, the probability of tunnelling is related to the strength of the driving Hamiltonian H_d; when $A(s)$ is large, it is more likely for tunnelling to occur.

Image credit: `https://commons.wikimedia.org/wiki/File:Quant-annl.jpg`

Q: Is it doing something quantum, part II: tunnelling

A. **Yes**.

There is experimental evidence for tunnelling.

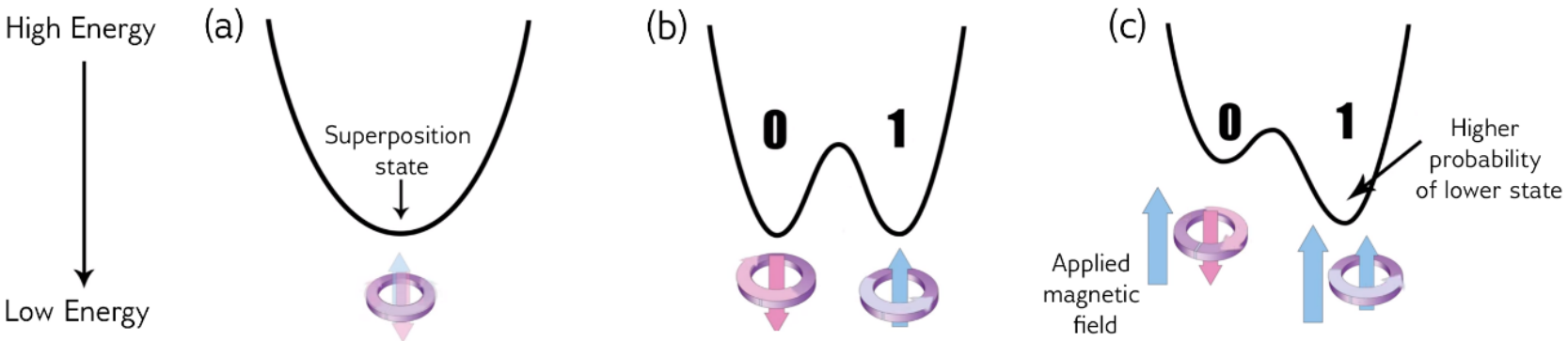


Figure 2.2: Energy diagram changes over time as the quantum annealing process runs and a bias is applied.

Main idea: At a certain point in the annealing process, the barriers are so high that transitions stop and the system freezes into a classical state. This is called the *freeze-out* time.

Image credit: `https://docs.dwavesys.com/docs/latest/_downloads/09-1076A-T_GettingStarted.pdf`

Q: Is it doing something quantum, part II: tunnelling

For transitions due to thermal excitation, this point should be *temperature dependent*, i.e. for a higher temperature, thermal fluctuations continue longer. But for transitions due to tunnelling, the temperature shouldn't matter!

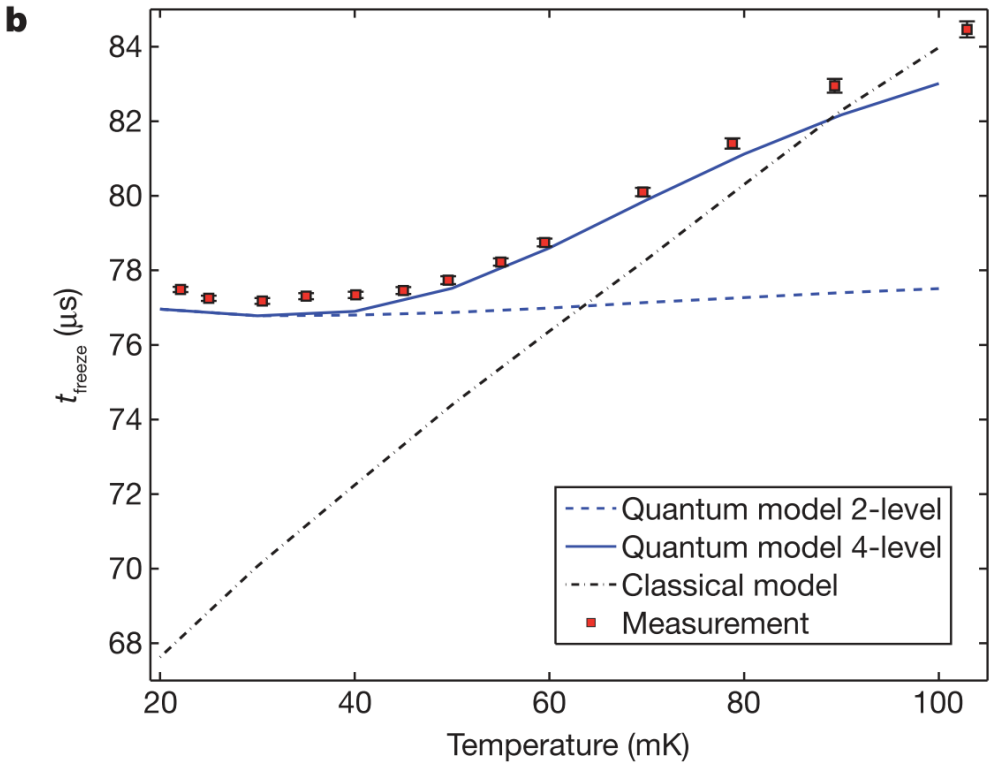


Image credit: `https://www.nature.com/articles/nature10012?error=cookies_not_supported&code=6f528d67-9632-42b7-b594-90d76b376f2e`

Q: Is there a quantum speedup?

There are certain classes of Hamiltonians that have been specially designed to

- Promote tunnelling, using high and narrow barriers [2]
- Map well to the D-Wave hardware graph[3]

for which there have been speedups demonstrated.

But it is *unclear* whether there is a quantum speedup *in general*, nor is there a general parametrization for the types of problems where this may be the case.

[2] `https://arxiv.org/pdf/1512.02206.pdf`

[3] `https://www.cs.amherst.edu/~ccmcgeoch/cf14-mcgeoch.pdf`

Q: Is there a quantum speedup?

Example: Results of a 2015 Google study between SA, D-Wave 2X, and quantum Monte Carlo methods; problems were designed to have high, narrow barriers.

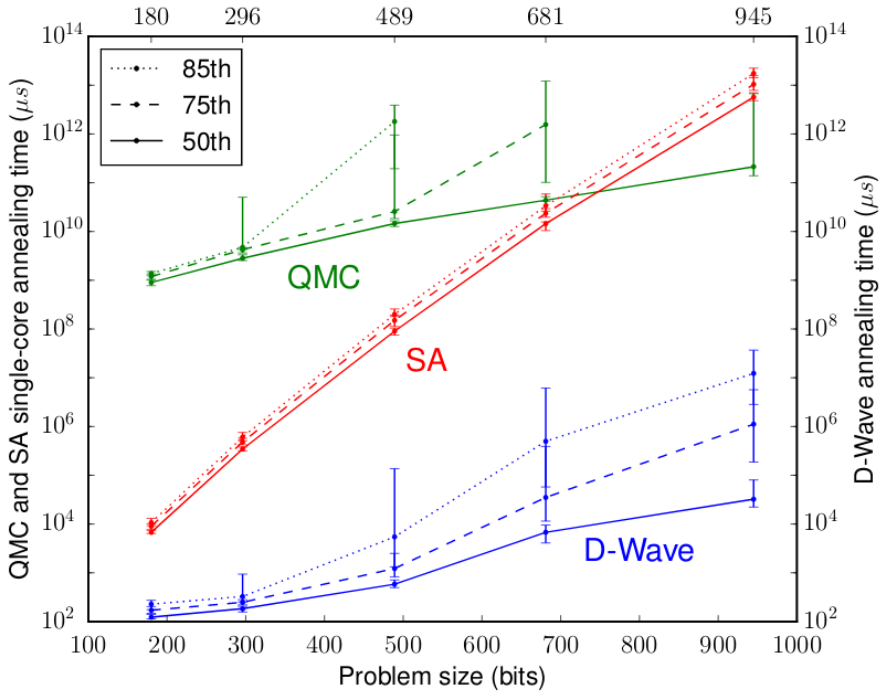


FIG. 4. Time to find the optimal solution with 99% probability for different problem sizes. We compare Simulated

Image credit: see the original paper `https://arxiv.org/pdf/1512.02206.pdf`

Q: Is there a quantum speedup?

Recall that the eigenstates of all H_p are computational basis states. Such Hamiltonians, which can be written in a basis such that all off-diagonal entries are ≤ 0, are termed *stoquastic*; their ground states can be expressed as classical probability distributions (i.e. all real, positive amplitudes).

It is thought that quantum annealers may demonstrate a quantum speed-up for problem Hamiltonians that are *non-stoquastic*, i.e. have positive off-diagonal entries (e.g. some $\sigma_i^x \sigma_j^x$ terms).

Implementing such interaction terms is technologically challenging, but there has been some recent progress in this area.

For more on this, see:

- `https://www.nature.com/articles/s41534-017-0037-z`
- `https://arxiv.org/abs/1903.06139`

Q: Does it work?

A: **Yes.**

They can find the ground states of the classes of optimization problems they are meant to, with high probability.

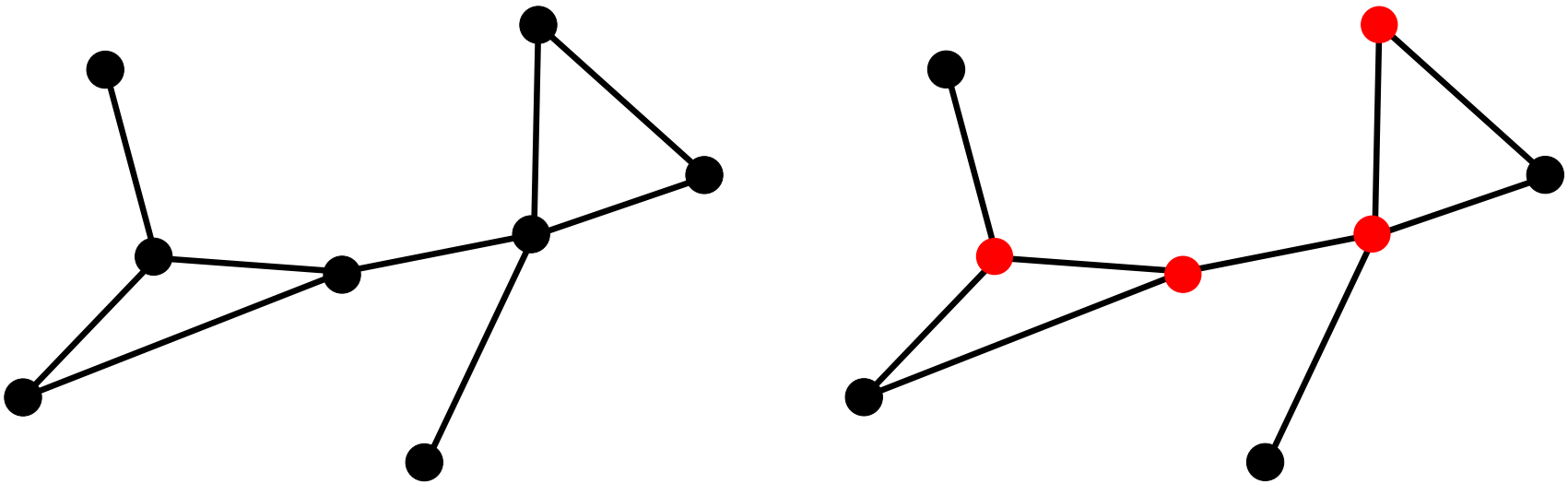
Recently, they are also showing promise for some machine learning applications (more on this tomorrow!).

Let's now look in detail at these suitable problems...

QUBOs and Ising Hamiltonians

Running example: vertex cover

Recall the vertex cover problem I showed you earlier.



How does this map to an Ising Hamiltonian?

Rather than jumping straight to Ising, we're first going to express this as something called a **QUBO**.

QUBOs

QUBO stands for:

- **Q**uadratic
- **U**nconstrained
- **B**inary
- **O**ptimization

Solving a QUBO involves finding a vector $\vec{x}$ of binary variables that minimizes or maximizes some cost function.

One way to write this is in matrix form:

$$\min_{\vec{x}} \vec{x}^T \mathbf{Q} \vec{x} + \vec{r}^T \vec{x} \tag{26}$$

where $\mathbf{Q}$ is a square matrix and $\vec{r}$ is a vector, populated with coefficients specific to the problem at hand.

QUBO and Ising model equivalence

We can rewrite this though by expanding out the matrix sums. Suppose $\vec{x}$ is a vector of length N. Then

$$\vec{x}^T \mathbf{Q} \vec{x} + \vec{r}^T \vec{x} = \sum_{i=1}^{N} \sum_{j=1}^{N} Q_{ij} x_i x_j + \sum_{i=1}^{N} r_i x_i, \quad x_i \in \{0, 1\}$$

Recall the form of the Ising Hamiltonian...

$$H = -\sum_{i<j} J_{ij} s_i s_j - \sum_i h_i s_i, \quad s_i \in \{+1, -1\} \tag{27}$$

These look quite similar!

QUBO and Ising model equivalence

We can transform between QUBOs and Ising Hamiltonians by making a simple change of variables:

$$x_i = \frac{1}{2}(s_i + 1) \tag{28}$$

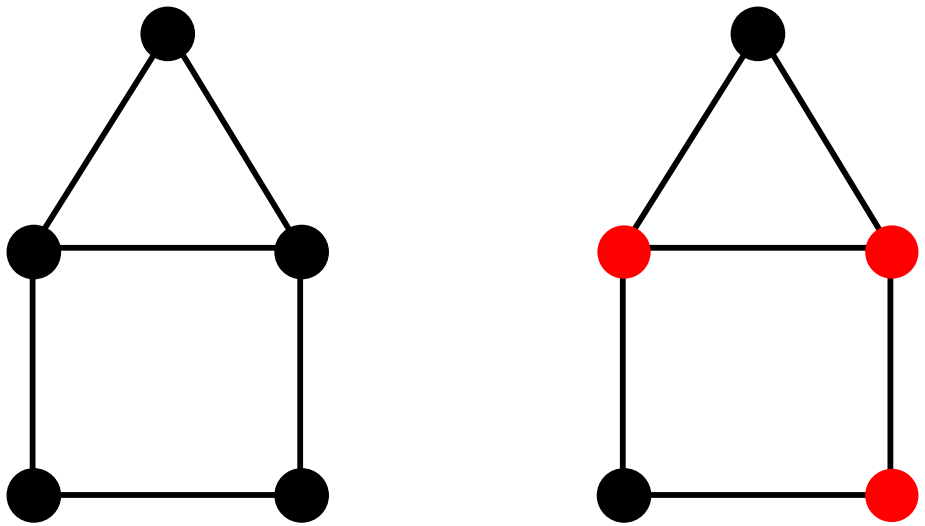
The spins with $s_i = -1$ get sent to $x_i = 0$, and spins with $s_i = 1$ to $x_i = 1$.

With this correspondence, we can run all kinds of QUBO problems on D-Wave hardware. Their libraries will even perform a lot of the conversion steps forward, so we don't have to worry about it!

For many problems, designing the QUBO in the first place is the hard part.

A QUBO for vertex cover

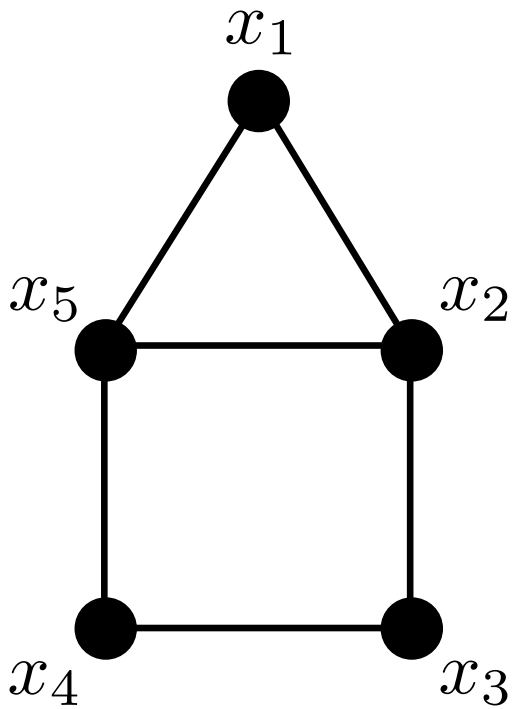
Let's start small, and revisit the problem of vertex cover.



How can we design a cost function over binary variables that will tell us the optimal set of vertices to colour?

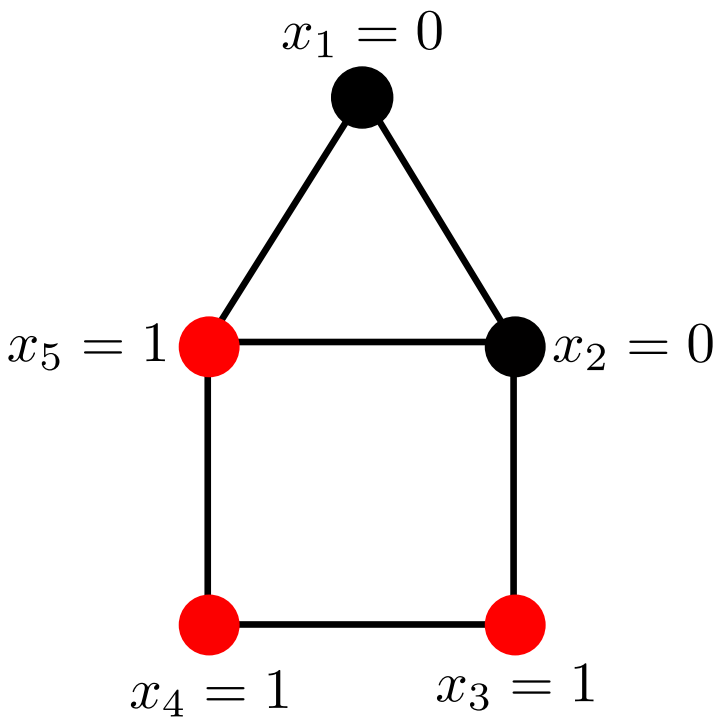
A QUBO for vertex cover

Whether or not a vertex is coloured is a *binary variable*.



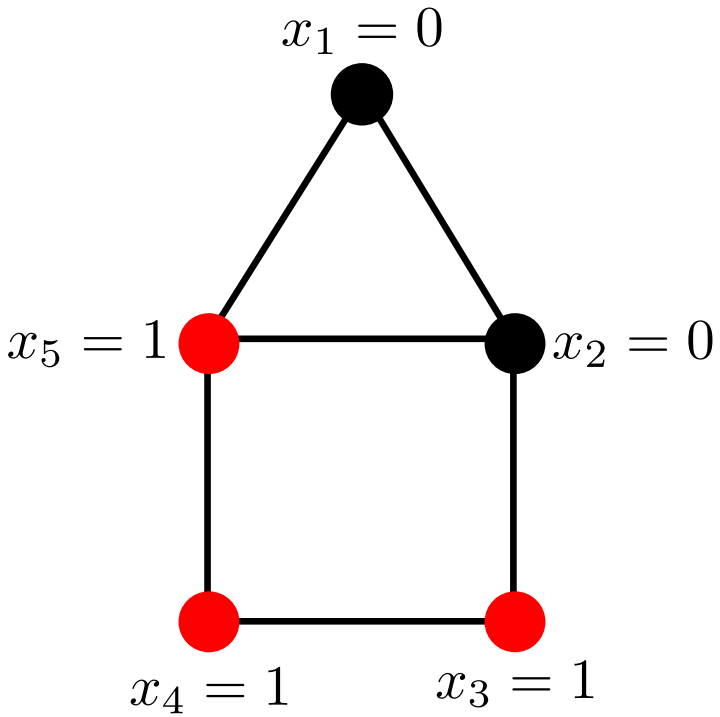
A QUBO for vertex cover

Let's assign coloured vertices to have value 1, and un-coloured 0.



Now that we have our variables, how do we come up with a minimizable cost function that represents the problem?

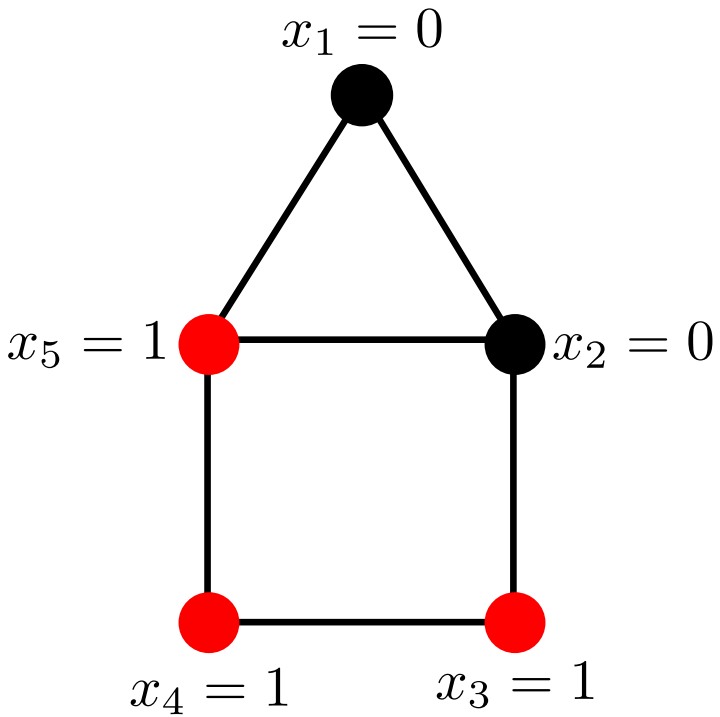
A QUBO for vertex cover



We need every edge to be attached to a coloured vertex, so we want to come up with a cost function that penalizes edges that are not attached, while favouring ones that are.

Intuitively, let's find some function of two vertices that is 0 if the colouring is valid, and 1 if it is not.

A QUBO for vertex cover



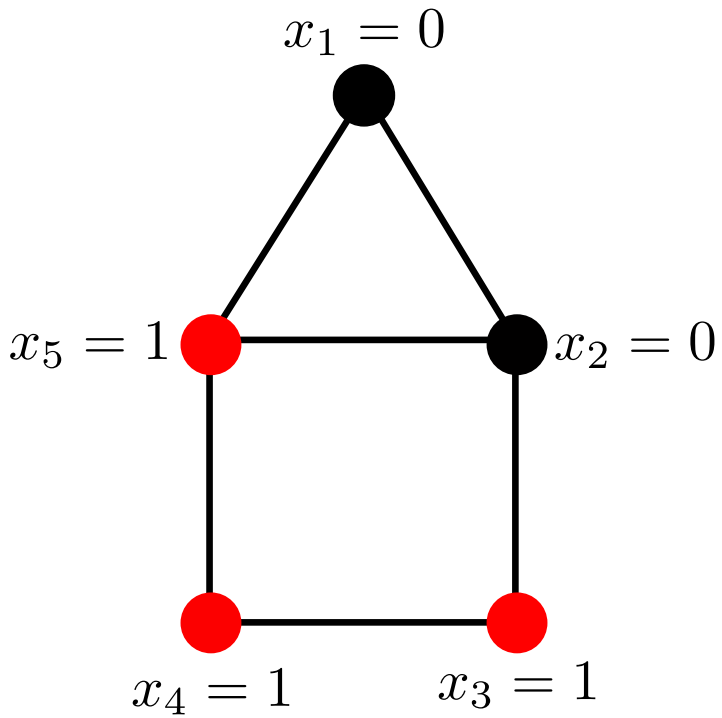
Consider for each edge ij the function

$$f(x_i, x_j) = (1 - x_i)(1 - x_j) \quad (29)$$

The possible values are:

$$f(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j = 1 \\ 0 & \text{if } x_i = 1 \text{ or } x_j = 1 \\ 1 & \text{if } x_i = x_j = 0 \end{cases}$$

A QUBO for vertex cover



Then in an optimal colouring,

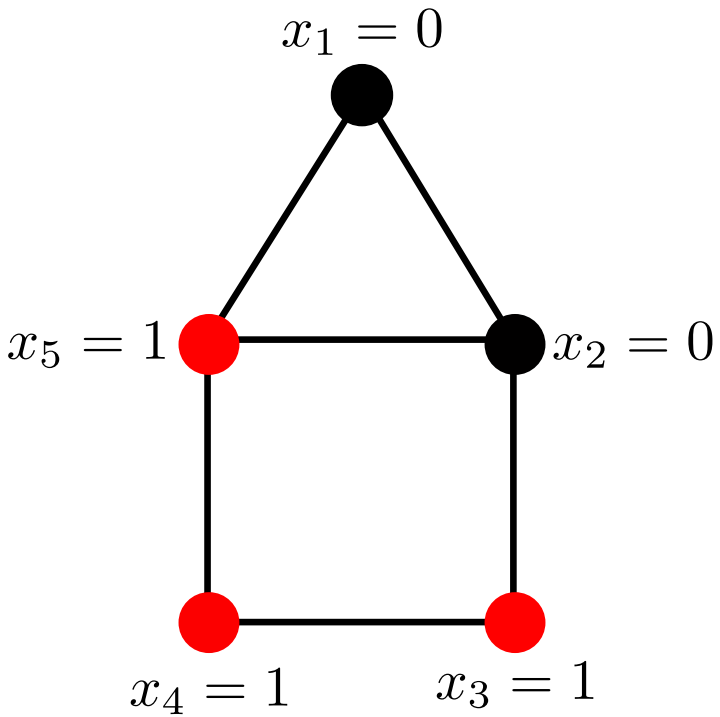
$$f(x_i, x_j) = (1 - x_i)(1 - x_j) = 0 \tag{30}$$

for all edges ij.

In QUBO form, we can write our cost function as

$$\min_{\vec{x}} \sum_{ij \in E} (1 - x_i)(1 - x_j) \tag{31}$$

A QUBO for vertex cover



However recall that we also want to colour the fewest vertices. We should increase the cost if too many variables are 1 to make those colourings unfavourable.

We can add to our QUBO

$$\sum_i x_i \tag{32}$$

which counts the number of coloured vertices.

A QUBO for vertex cover

The full QUBO for vertex cover then looks like this:

$$\min_{\vec{x}} \left(\sum_{ij \in E} (1 - x_i)(1 - x_j) + \sum_i x_i \right) \tag{33}$$

However in practice, we want to adjust slightly the ratio between the two terms using a penalty factor:

$$\min_{\vec{x}} \left(\sum_{ij \in E} (1 - x_i)(1 - x_j) + \gamma \sum_i x_i \right), \quad \gamma < 1 \tag{34}$$

Adding a $\gamma < 1$ factor to the second term ensures that the first term, i.e. assigning a valid colouring, is prioritized, while the particular number of vertices we use is secondary.

A QUBO for financial portfolio optimization

Earlier I alluded to the fact that it is often not obvious how to formulate an arbitrary problem as a QUBO.

Now I'll show you a different type of case requiring additional steps: integer optimization problems.

Tomorrow I will show you the particle physics applications which approach things in still different ways.

A QUBO for financial portfolio optimization

You have the opportunity to purchase 100 units of stocks from a fixed list of assets, and you want to maximize your profits. You know the average returns of each stock, and their covariances.

Stock	Avg. return
AAA	3.44 %
BBB	2.21 %
CCC	-0.28 %
⋮	⋮

Cov.	AAA	BBB	⋯
AAA	0.0038	0.002	⋯
BBB	0.002	-0.006	⋯
CCC	0.014	-0.0008	⋯
⋮	⋮	⋮	⋱

Suppose you're restricted to buying no more than 5 of any stock.

How on Earth do you turn this into a QUBO to perform stock selection?

I will show you one such formulation...

A QUBO for financial portfolio optimization

Suppose we have M different stocks, AAA, $BBB, \ldots, MMM$.

Let $\vec{x}$ represent the number of units of each stock that we will purchase:

$$\vec{x} = (x_1, x_2, \ldots, x_M), \quad x_i \in \mathbb{Z}, \ 0 \leq x_i \leq 5 \tag{35}$$

We are supposing here that we are restricted to *integer amounts* of stocks - apparently this is sometimes a purchasing constraint in financial applications.

A QUBO for financial portfolio optimization

We know the *average returns* of each stock:

Stock	Avg. return
AAA	$r_1 = 3.44\ \%$
BBB	$r_2 = 2.21\ \%$
CCC	$r_3 =$-0.28 %
$\vdots$	$\vdots$

The expected value of return on our investment allocation $\vec{x}$ is

$$\text{exp. return} = \vec{r}^T \vec{x} = \sum_{i=1}^{M} r_i x_i \tag{36}$$

A QUBO for financial portfolio optimization

What's to stop us from just buying the maximum amount of each of the stocks with the highest returns?

Typically want to look at *relationships* between the assets and diversify the portfolio. We thus add a *risk* term related to the covariances Q_{ij} of the assets:

$$\text{risk} = \lambda \vec{x}^T \mathbf{Q} \vec{x} = \lambda \sum_{i,j=1}^{M} Q_{ij} x_i x_j \tag{37}$$

where λ is a risk aversion coefficient.

If two assets are highly correlated and we invest a lot in both, if one goes south then things will be doubly bad.

A QUBO for financial portfolio optimization

For our cost function, we would like to *maximize* return, and *minimize* risk:

$$\max_{\vec{x}} \vec{r}^T \vec{x} - \lambda \vec{x}^T \mathbf{Q} \vec{x} \tag{38}$$

Equivalently, we can flip signs and minimize the negative returns:

$$\min_{\vec{x}} \lambda \vec{x}^T \mathbf{Q} \vec{x} - \vec{r}^T \vec{x} \tag{39}$$

This is starting to look like a QUBO...

There are still some things we need to add though.

A QUBO for financial portfolio optimization

We are limited to buying a total of 100 units of stocks, so we need to add a penalty term that will increase the energy of solutions that don't satisfy this. We want

$$\sum_{i=1}^{M} x_i = 100 \tag{40}$$

We'll add to the cost function:

$$\gamma \left(\sum_{i=1}^{M} x_i - 100 \right)^2 \tag{41}$$

This is 0 when the constraint is satisfied, and positive when it is violated in either direction. γ represents a penalty factor on how badly we want the constraint to be satisfied.

A QUBO for financial portfolio optimization

So our total cost function is:

$$\min_{\vec{x}} \quad \lambda \sum_{i=1}^{M} \sum_{j=1}^{M} Q_{ij} x_i x_j - \sum_{i=1}^{M} r_i x_i + \gamma \left(\sum_{i=1}^{M} x_i - 100 \right)^2 \tag{42}$$

Wait a minute... the 'B' in QUBO means we need *binary* variables x_i, but here these are integers!

A QUBO for financial portfolio optimization

When working with integer problems, we need to make a *change of variables* from integers to binary. We call this an *integer encoding*. We will expand each x_i as:

$$x_i = \sum_{j=1}^{d_i} c_j^{(i)} y_j^{(i)}, \quad y \in \{0, 1\} \tag{43}$$

The c_j^i here are expansion coefficients that we are free to choose.

Performing this expansion will increase the size of the problem (and take up more space on the D-Wave chip).

A QUBO for financial portfolio optimization

Suppose each x_i ranges from 0 to 5.

We can choose all $c_j^i = 1$, a *unary* encoding:

$$x_i = y_1^{(i)} + y_2^{(i)} + y_3^{(i)} + y_4^{(i)} + y_5^{(i)} \tag{44}$$

Each x_i will require 5 binary variables.

A QUBO for financial portfolio optimization

We can choose all c_j^i to be powers of 2, a *binary* encoding:

$$x_i = 1 \cdot y_1^{(i)} + 2 \cdot y_2^{(i)} + 4 \cdot y_3^{(i)} \tag{45}$$

Now each x_i will require 3 binary variables.

Or we can choose something entirely different. People have developed heuristic techniques to choose encodings. For example, may we choose:

$$x_i = 1 \cdot y_1^{(i)} + 1 \cdot y_2^{(i)} + 1 \cdot y_3^{(i)} + 2 \cdot y_4^{(i)} \tag{46}$$

A QUBO for financial portfolio optimization

For simplicity, let's choose the unary encoding,

$$x_i = \sum_{j=1}^{5} y_j^{(i)} \tag{47}$$

Then to construct the QUBO, we need to take the cost function

$$\min_{\vec{x}} \quad \lambda \sum_{i=1}^{M} \sum_{j=1}^{M} Q_{ij} x_i x_j - \sum_{i=1}^{M} r_i x_i + \gamma \left(\sum_{i=1}^{M} x_i - 100 \right)^2 \tag{48}$$

and replace all the x_i with the sums over ys.

A QUBO for financial portfolio optimization

Feel free to work it out yourself, the answer looks something like:

$$\min_{\vec{y}} \sum_{i,j=1}^{M} \sum_{k,\ell=1}^{5} (\lambda Q_{ij} + \gamma) y_k^{(i)} y_\ell^{(j)} - \sum_{i=1}^{M} \sum_{k=1}^{5} (r_i + 200\gamma) y_k^{(i)} \tag{49}$$

where I have removed the constant 10000γ since it is a constant shift in the energy.

To get the Ising Hamiltonian, we would then perform a change of variables of each y to the spin domain,

$$y_j^{(i)} = \frac{1}{2}\left(s_j^{(i)} + 1\right) \tag{50}$$

Note: people have actually implemented this kind of problem, albeit at a very small scale: `https://ieeexplore.ieee.org/document/7482755`.

Overhead of problem conversion

That was really ugly.

The main points I wanted to make here are:

- A wide variety of problems can be turned into QUBOs
- Formulation of the QUBO is not always intuitive and requires some creativity
- The need to convert between domains, e.g. integer to binary, leads to a large overhead in the number of problem variables

The latter point is one contributor to the fact that even though a D-Wave 2000Q chip has 2048 qubits, we (very likely) cannot solve a problem over 2048 variables.

Summary

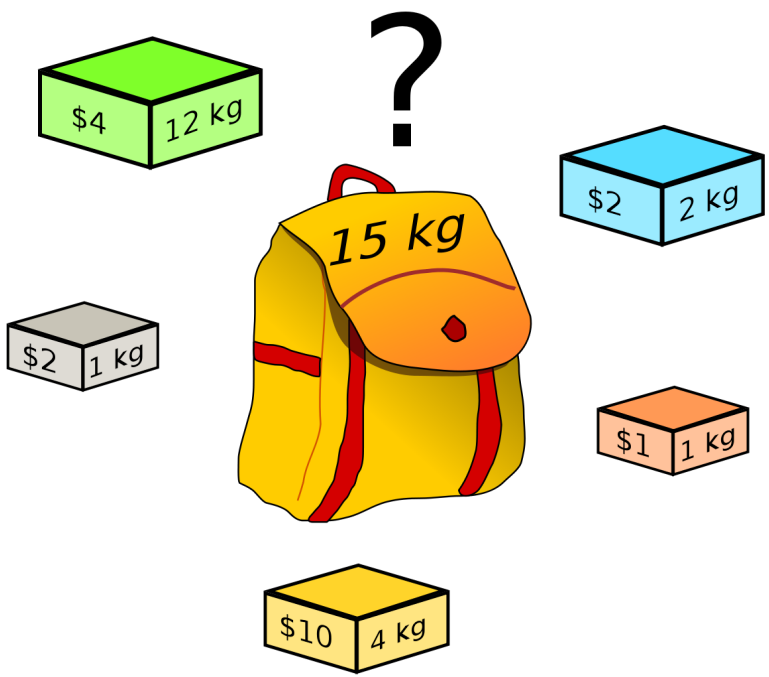
We discussed:

- The ideas behind adiabatic quantum computing
- Simulated annealing and quantum annealing
- D-Wave hardware
- Formulation of example problems as QUBOs

Tomorrow: I will solve the vertex cover problem *live* on the D-Wave!

We will also discuss two HEP case studies, and discuss the idea of quantum machine learning.

Homework: The Knapsack Problem



You have a collection of N items.

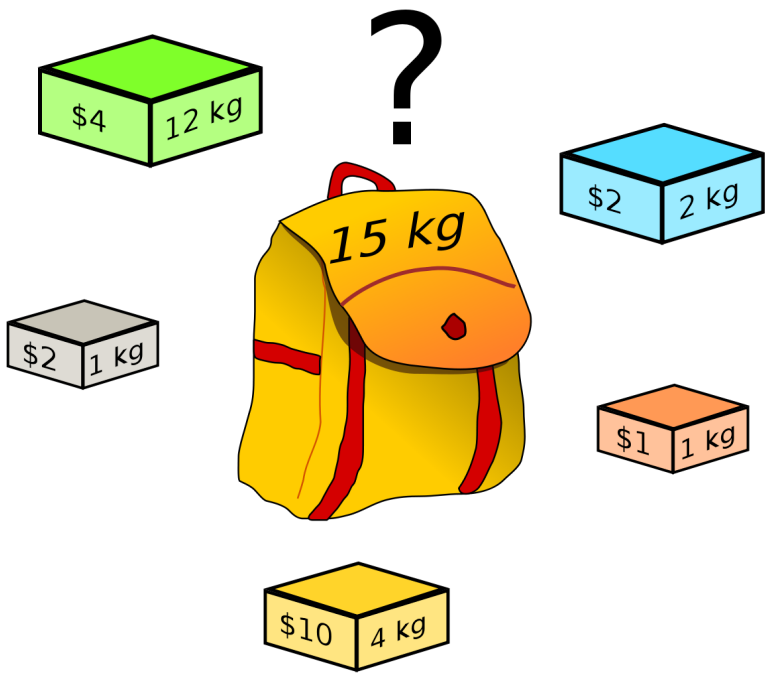
Each item has value c_i, and weight w_i.

We want to choose a set of items to put in the knapsack that

- Maximizes the total value of the contents
- Weights as close as possible, but no more than an upper limit W

Image credit: https://commons.wikimedia.org/wiki/File:Knapsack.svg

Homework: The Knapsack Problem



Your job is to construct a QUBO that represents this optimization problem. Think about:

- What your binary variables should represent, and how many there are
- The structure of your cost function
- How to ensure we get close to or exactly the optimal W
- How to penalize overweight solutions

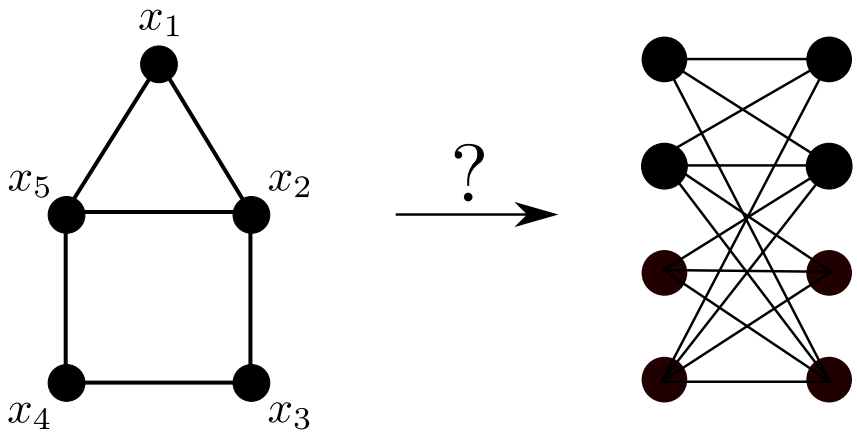
Hint: there are 3 components to this QUBO.

Image credit: https://commons.wikimedia.org/wiki/File:Knapsack.svg

‘Homework’: Vertex cover and the chimera graph

Recall the QUBO for vertex cover we designed earlier in the lecture.

Write it out in full for the example graph. If you’d like, convert it to an Ising Hamiltonian - what are the coupling coefficients and external fields?



This graph has only 5 vertices; a D-Wave Chimera unit cell has 8. Can you assign the vertices to qubits in a way that satisfies the necessary couplings?