Quantum Annealing

NSF/DOE Quantum Science Summer School

Scott Pakin

8 June 2017

LA-UR-17-24530

Outline

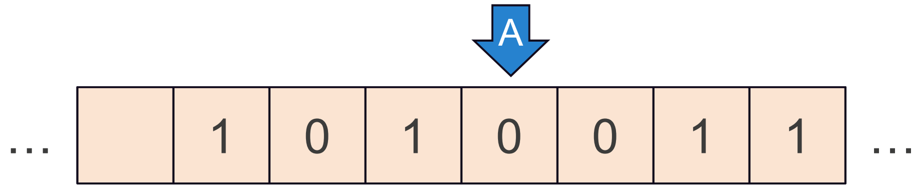
- Performance potential of quantum computing
- Quantum annealing
- Case study: D-Wave quantum annealers
- How to program a quantum annealer
- Parting thoughts

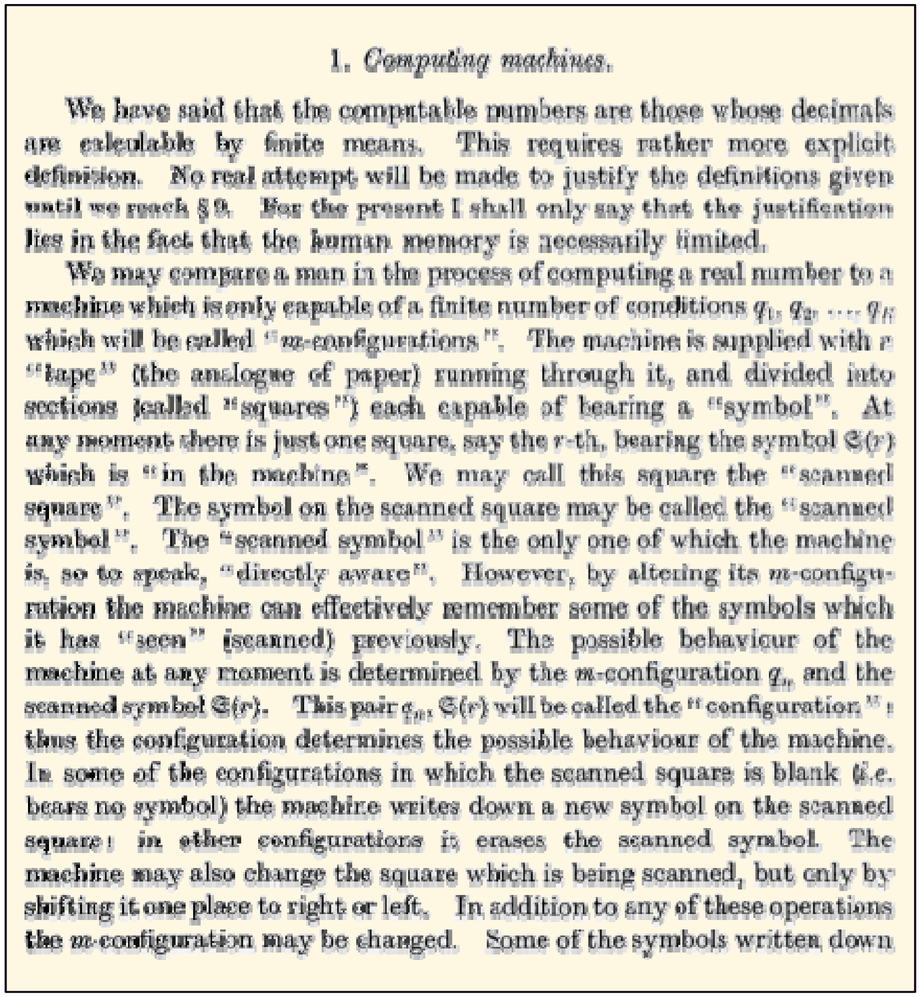
Main Topic to be Addressed

- **What problems can quantum computers solve fast?**
  - What “flavor” of quantum are we referring to?
  - What exactly is a computer?
  - What do we mean by *solve*?
  - What is considered *fast* in this context?

What is a Computer?

- **Mathematical abstraction: a Turing machine**
  - $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$
  - All states, all symbols, blank symbol, input symbols, transition function, initial state, and final states
  - All of the preceding sets are finite, but the memory ("tape") on which they operate is infinite
- **Transition function**
  - Maps {*current state*, *symbol read*} to {*new state*, *symbol to write*, left/right}
  - *Example*: "If you're in state *A* and you see a 0, then write a 1, move to the left, and enter state *B*"





1. *Computing machines.*

We have said that the computable numbers are those whose decimals are calculable by finite means. This requires rather more explicit definition. No real attempt will be made to justify the definitions given until we reach § 9. For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited.

We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions $q_1, q_2, \dots, q_R$ which will be called "*m*-configurations". The machine is supplied with a "tape" (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the *r*-th, bearing the symbol $\mathfrak{S}(r)$ which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its *m*-configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the *m*-configuration q_n and the scanned symbol $\mathfrak{S}(r)$. This pair q_n, $\mathfrak{S}(r)$ will be called the "configuration": thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (*i.e.* bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the *m*-configuration may be changed. Some of the symbols written down

A. M. Turing, "On Computable Numbers, with an Application to the *Entscheidungsproblem*". Proceedings of the London Mathematical Society, 12 November 1936.

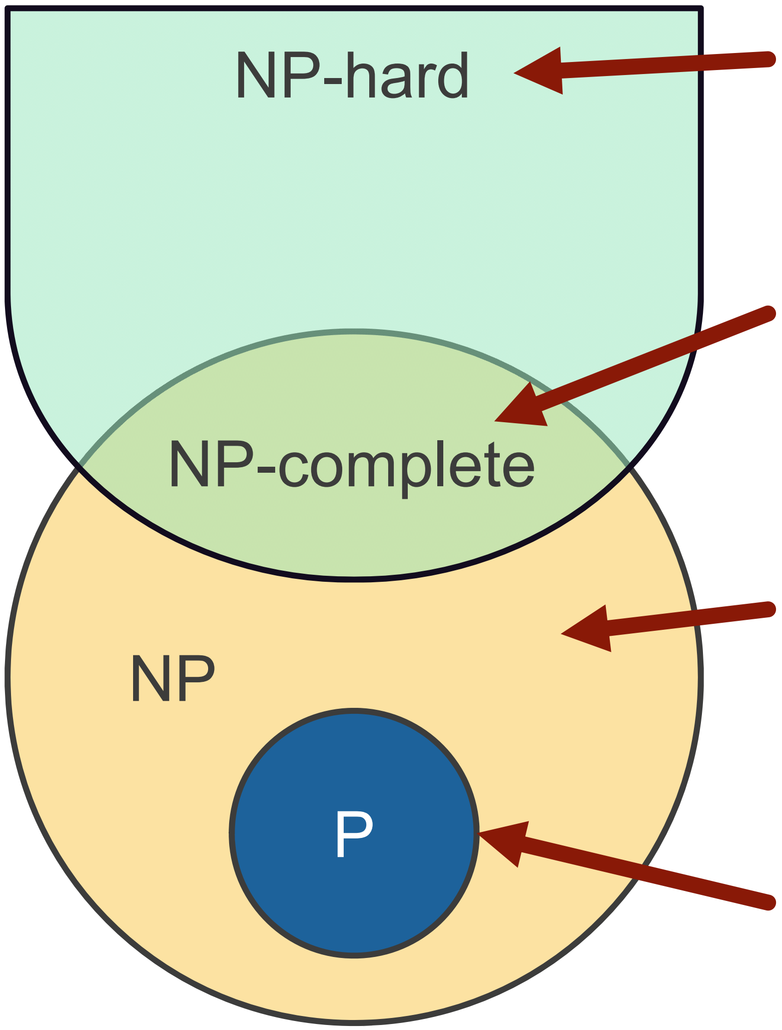
What Else is a Computer?

- **Nondeterministic Turing machine**
  - Replace the transition *function* with a transition *relation*
  - Contradictions are allowed
  - *Example*: “If you’re in state *A* and you see a 0, then simultaneously ① write a 1, move to the left, and enter state *B;* ② write a 0, move to the right, and enter state *C*; and ③ write a 1, move to the right, and enter state *B*.”
  - At each step, an oracle suggests the best path to take (not realistic, obviously)
- **Quantum Turing machine**
  - Same 7-tuple as in the base Turing machine
  - $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$
  - *But…*set of states is a Hilbert space; alphabet is a (different) Hilbert space; blank symbol is a zero vector; transition function is a set of unitary matrices; initial state can be in a superposition of states; final state is a subspace of the Hilbert space
  - No change to input/output symbols; those stay classical

Introduction to Complexity Theory

- **What problems can a computer solve quickly?**
- **Discuss in terms of *asymptotic complexity*, not wall-clock time**
  - Ignore constants and all but the leading term
  - For input of size n, $O(n)$ can mean $3n$ seconds or $5n+2 \log n+3/n+20$ hours; it doesn't matter
  - Polynomial time, $O(n^k)$ for any k, is considered good (efficiently solvable), even if an input of size n takes $1000n^{20}$ years to complete
  - Superpolynomial time—most commonly exponential time, $O(k^n)$ for $k>1$—is considered bad (intractable), even if an input of size n completes in only 2^n femtoseconds
- **Categorize problems into complexity classes**
  - *Goal*: Determine which complexity classes are subsets or proper subsets of which other classes (i.e., representing, respectively, "no harder" or "easier" problems)
  - Approach is typically based on *reductions*: proofs that an efficient solution to a problem in one class implies an efficient solution to all problems in another class
- **Typically focus on decision problems**
  - Output is either "yes" or "no"

Venn Diagram of Common Complexity Classes



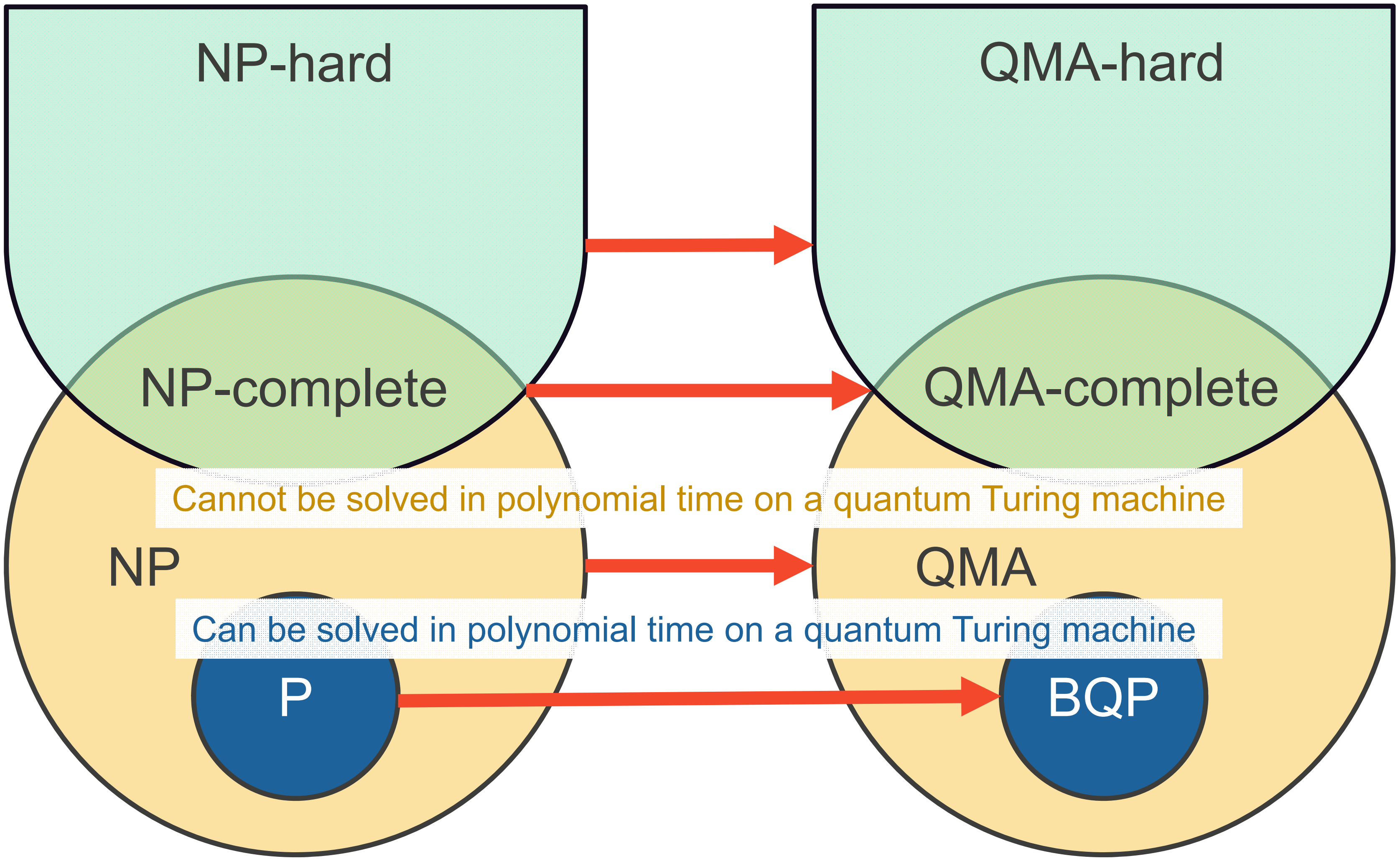
- Problems at least as hard as those in NP
- Not necessarily decision problems
- *Example*: Given a weighted graph, what is the shortest-length Hamiltonian path?

- Hardest of the problems in NP
- *Example*: Given a set of integers, is there a subset whose sum is 0?

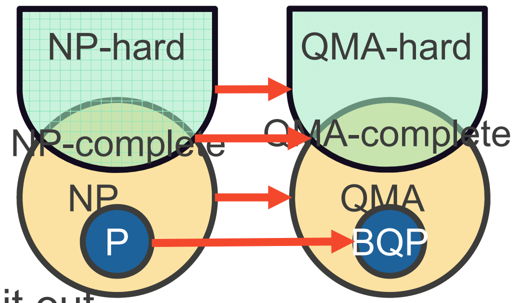
- "Hard" decision problems
- Can be solved in polynomial time on a *nondeterministic* Turing machine
- Solutions can be *verified* in polynomial time on a deterministic Turing machine
- *Example*: Does a given integer have a prime factor whose last digit is 3?

- "Easy" decision problems
- Can be solved in polynomial time on a deterministic Turing machine
- *Example*: Does a given matrix have an eigenvalue equal to 1.2?

Quantum Complexity Classes


NP-hard
QMA-hard
NP-complete
QMA-complete
Cannot be solved in polynomial time on a quantum Turing machine
NP
QMA
Can be solved in polynomial time on a quantum Turing machine
P
BQP

What Do We Know?



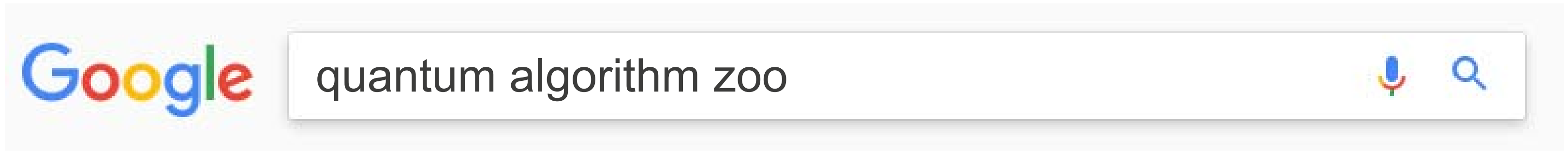
- **Short answer: Almost nothing**
- **P vs. NP**
  - We know that P ⊆ NP, but we don't know whether P = NP or P ≠ NP; conjectured that P ≠ NP
  - $1M prize from the Clay Mathematics Institute if you figure it out
- **NP-intermediate vs. NP-complete**
  - (NP-intermediate are the set of problems in NP but not in NP-complete)
  - We know that NP-intermediate ⊆ NP-complete, but we don't know if they're equal
  - *Implication*: If NP-intermediate ≠ NP-complete, then factoring (NP-intermediate) may in fact be an easy problem, but we just haven't found a good classical algorithm yet
- **P vs. BQP**
  - We know that P ⊆ BQP, but we don't know whether P = BQP or P ≠ BQP
  - *Implication*: If P = BQP, then quantum computers offer no substantial (i.e., superpolynomial) performance advantage over classical computers
- **NP-complete vs. BQP**
  - We don't know relation of BQP to NP-complete; conjectured that BQP ⊂ NP-complete
  - *Implication*: Believed that quantum computers cannot solve NP-complete problems in polynomial time

It’s Not All Doom and Gloom

- **Sure, quantum computers probably can’t solve NP-complete problems in polynomial time**
- **Still, even a polynomial-time improvement is better than nothing**
- **Grover’s algorithm**
  - Find an item in an unordered list
  - $O(n) \rightarrow O(\sqrt{n})$
- **Shor’s algorithm**
  - Factor an integer into primes (NP, but not NP-complete)
  - $O(2^{\sqrt[3]{n}}) \rightarrow O((\log n)^3)$

Aside: Quantum Algorithms (Circuit Model)

- **Key concepts**
  - *N* classical bits go in, *N* classical bits come out
  - Can operate on all 2^N possibilities in between
  - Requirement: Computation must be reversible (not a big deal in practice)
  - Main challenge: You get only one measurement; how do you know to measure the answer you're looking for?
  - High-level approach: Quantum states based on complex-valued probability *amplitudes*, not probabilities—can sum to 0 to make a possibility go away
- **Very difficult in practice**
  - Only 55 algorithms known to date

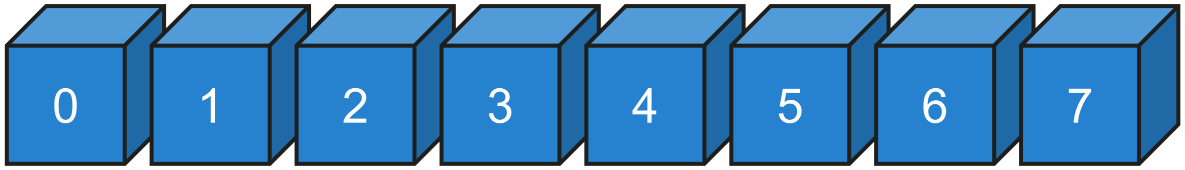


quantum algorithm zoo

  - Based on only a handful of building blocks
  - Each requires substantial cleverness; not much in the way of a standard approach

Grover's Algorithm

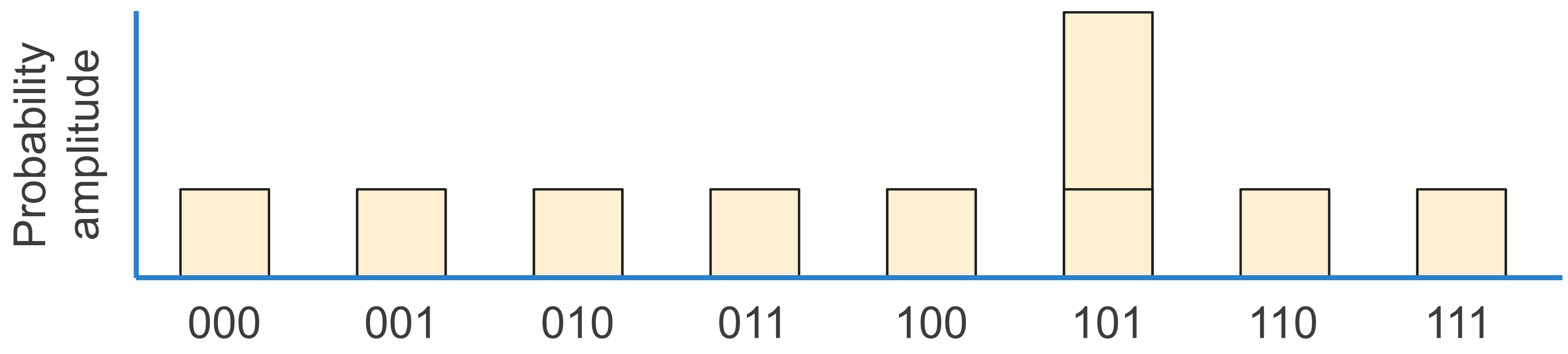
- **Which box contains the prize?**



  – Classically, must open all 8 boxes in the worst case
- **Let's see how we can use quantum effects to do better than that…**
- **Given**
  – A power-of-two number of boxes
  – A guarantee that exactly one box contains the prize
  – An operator U_ω that, given a box number $|x\rangle$, flips the probability amplitude iff the box contains the prize (i.e., $U_\omega|x\rangle = -|x\rangle$ for $x = \omega$ and $U_\omega|x\rangle = |x\rangle$ for $x \neq \omega$)
- **Define the *Grover diffusion operator* as follows**
  – $|s\rangle \equiv \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|x\rangle$ (i.e., the equal superposition of all states)
  – $U_s \equiv 2|s\rangle\langle s| - I$ (the Grover diffusion operator)

Grover’s Algorithm (cont.)

- **The basic algorithm is fairly straightforward to apply:**
  - Put each of the *N* qubits in a superposition of $|0\rangle$ and $|1\rangle$
  - For $\sqrt{N}$ iterations
    - Apply U_ω to the state
    - Apply U_s to the state
- **How does that work?**
  - Gradually shifts the probability amplitude to qubit ω from all the other qubits
  - When we measure, we’ll get a result of ω with near certainty

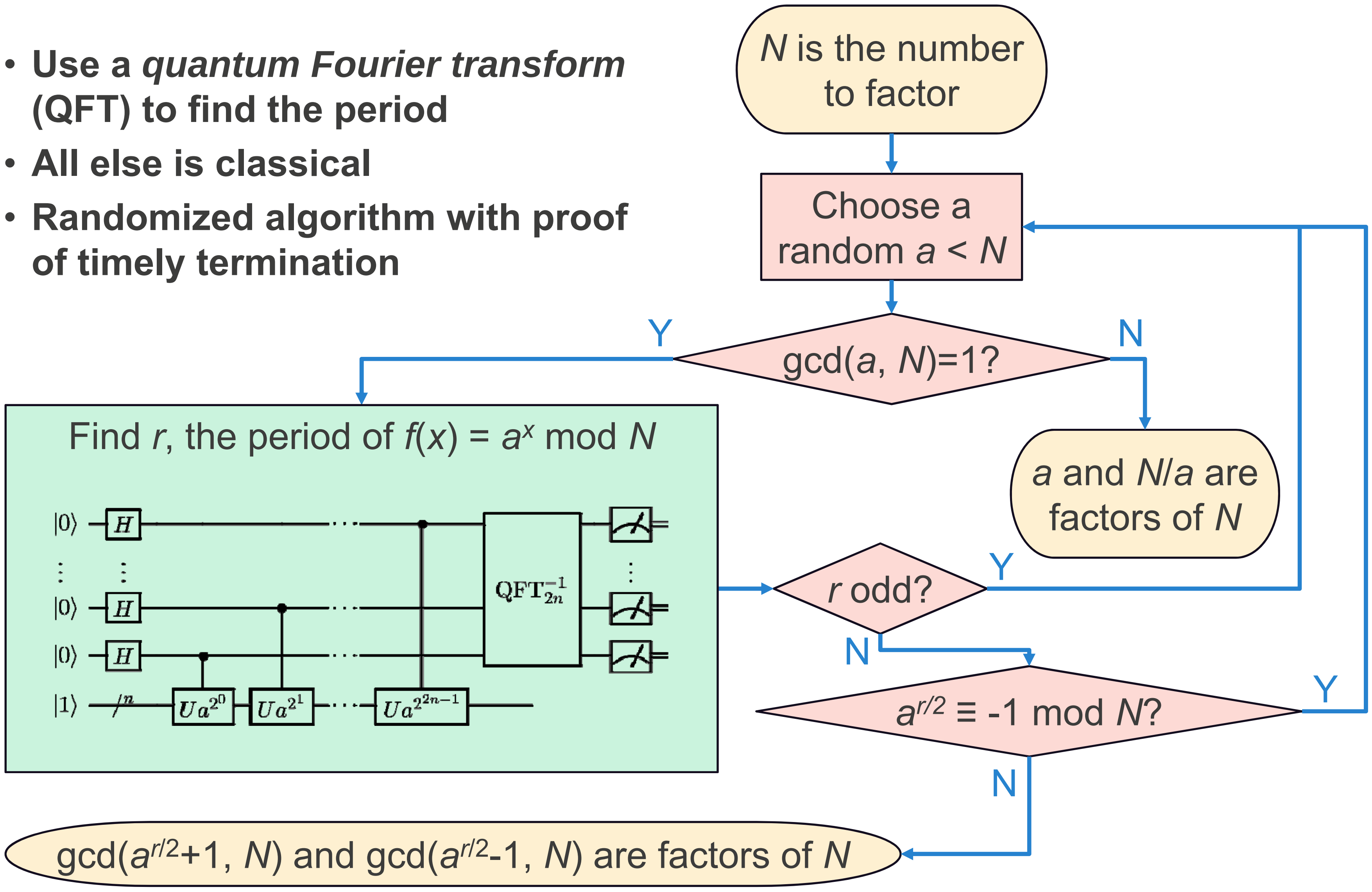


Shor’s Algorithm

- **Factor 1,274,093,332,123,426,680,869 into a product of two primes**
  - Okay, it’s 135,763,451,261×9,384,656,329
- **Observations**
  - Given that N is the product of two primes, p and q
  - Given some a that is divisible by neither p nor q
  - Then the sequence $\{a^1 \bmod N, a^2 \bmod N, a^3 \bmod N, a^4 \bmod N, a^5 \bmod N, \ldots\}$ will repeat every r elements (the sequence’s *period*)
  - As Euler discovered (~1760), r always divides $(p-1)(q-1)$
- **Example**
  - Let a be 2 and N be 15 (=3×5)
  - Then $a^x \bmod N = \{2, 4, 8, 1, 2, 4, 8, 1, 2, 4, 8, 1, 2, 4, 8, 1 \ldots\}$ so r is 4
  - Lo and behold, 4 divides (3−1) (5−1)=8
- **Approach**
  - Once we know the period, r, it’s not too hard to find N’s prime factors p and q
  - Unfortunately, finding r is extremely time-consuming…for a classical computer

Shor's Algorithm (cont.)

- Use a *quantum Fourier transform* (QFT) to find the period
- All else is classical
- Randomized algorithm with proof of timely termination



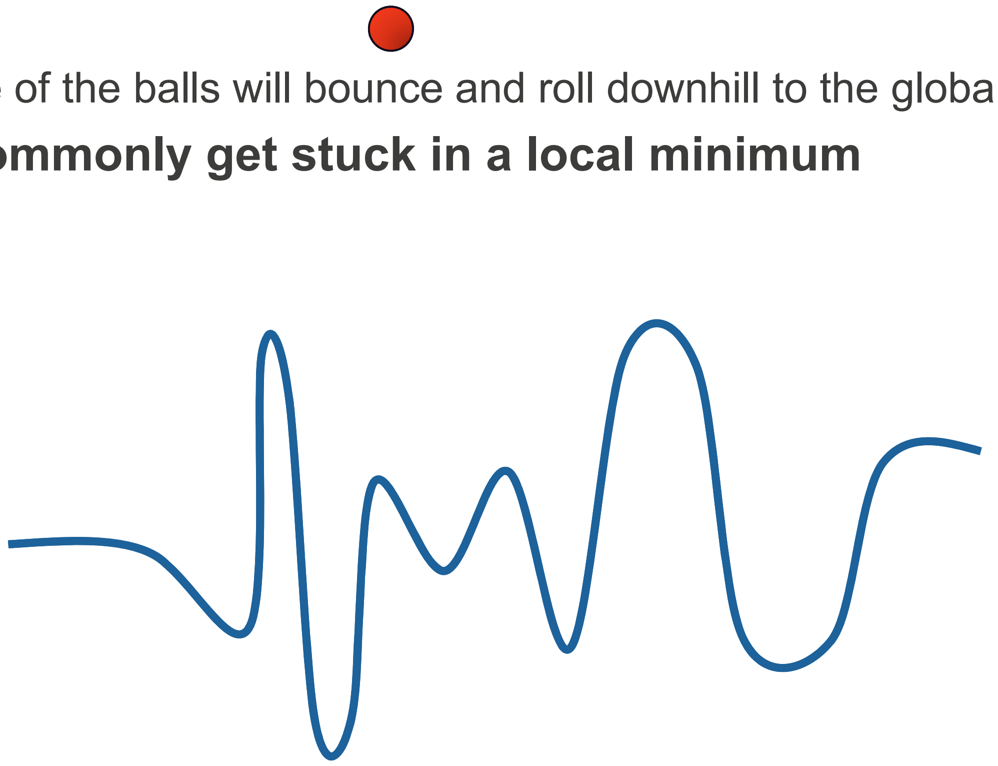
N is the number to factor
Choose a random $a < N$
$\gcd(a, N)=1$?
Y
N
a and N/a are factors of N
Find r, the period of $f(x) = a^x \bmod N$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|1\rangle$
H
n
Ua^{2^0}
Ua^{2^1}
$Ua^{2^{2n-1}}$
QFT_{2n}^{-1}
r odd?
Y
N
$a^{r/2} \equiv -1 \bmod N$?
Y
N
$\gcd(a^{r/2}+1, N)$ and $\gcd(a^{r/2}-1, N)$ are factors of N

Outline

- Performance potential of quantum computing
- **Quantum annealing**
- Case study: D-Wave quantum annealers
- How to program a quantum annealer
- Parting thoughts

Simulated Annealing

- **Classical (and classic) optimization approach**
- **Find the coordinates of the minimum value in an energy landscape**
- **Conceptual approach**
  - Drop a bunch of rubber balls on the landscape, evaluating the function wherever they hit
  - Hope that one of the balls will bounce and roll downhill to the global minimum
- **Challenge: Commonly get stuck in a local minimum**



Quantum Mechanics to the Rescue

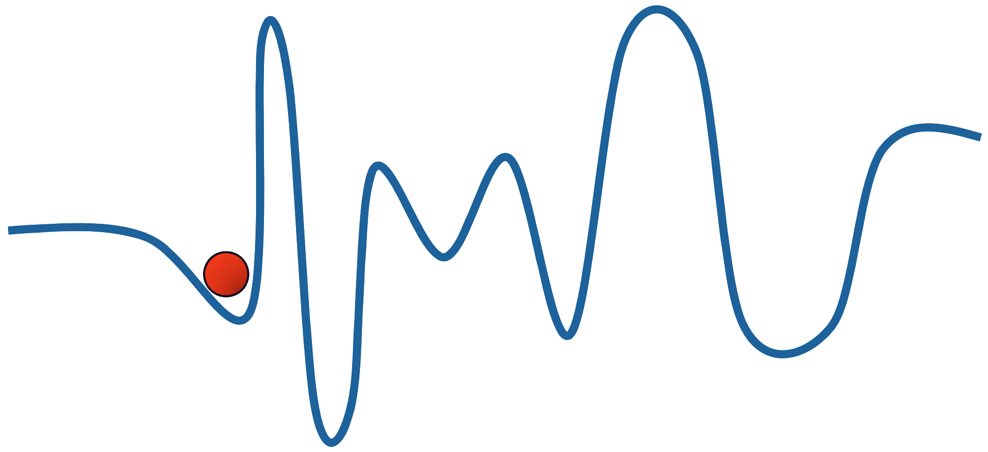
- **Consider adding a time-dependent transverse field to a 2-local Ising Hamiltonian:**

$$\mathcal{H}(t) = \overbrace{\underbrace{-\sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1} J_{i,j}\sigma_i^z\sigma_j^z}_{\text{Longitudinal interactions}} \underbrace{-\sum_{i=0}^{N-1} h_i\sigma_i^z}_{\text{Longitudinal field}}}^{\mathcal{H}_0\text{ (classical part)}} \underbrace{-\Gamma(t)\sum_{i=0}^{N-1}\sigma_i^x}_{\text{Transverse field}}$$

- **Implication of the adiabatic theorem**
  - If we gradually decrease the amplitude of the transverse field, $\Gamma(t)$, from a very large value to 0, we should drive the system into the ground state of $\mathcal{H}_0$
- **The real benefit: quantum tunneling**

Quantum Tunneling

- **Introduced by the $\Gamma(t)$ (transverse) term**
- **Enables jumping from one classical state (eigenstate of $\mathcal{H}_0$) to another**
  - Decreases likelihood of getting stuck in a local minimum
- **Unlike simulated annealing, width of energy barrier is important, but height is not**



Time Evolution

- **If purely adiabatic and sufficiently slow, the system remains in the ground state as it moves from the initial, “generic” Hamiltonian to the problem Hamiltonian**
- **D-Wave’s initial state**
  - Ground state (not degenerate): $|+\rangle|+\rangle|+\rangle \cdots |+\rangle$
  - 1st excited state ($\binom{N}{1}$-way degenerate): $|-\rangle|+\rangle|+\rangle \cdots |+\rangle$, $|+\rangle|-\rangle|+\rangle \cdots |+\rangle$, $|+\rangle|+\rangle|-\rangle \cdots |+\rangle$, … $|+\rangle|+\rangle|+\rangle \cdots |-\rangle$
  - 2nd excited state ($\binom{N}{2}$-way degenerate): $|-\rangle|-\rangle|+\rangle \cdots |+\rangle$, $|-\rangle|+\rangle|-\rangle \cdots |+\rangle$, $|+\rangle|-\rangle|-\rangle \cdots |+\rangle$, … $|+\rangle|+\rangle|+\rangle \cdots |-\rangle$
  - etc.

A Brief Aside

- **What we just saw is adiabatic quantum *optimization***
  - Optimization problem is to find the $\sigma_i^z \in \{-1, +1\}$ that minimize $\mathcal{H}_0$
- **A more powerful variation is adiabatic quantum *computing***

$$\mathcal{H}_{ZZXX} = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{i,j}\sigma_i^z\sigma_j^z + \sum_{i=0}^{N-1} h_i\sigma_i^z + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} K_{i,j}\sigma_i^x\sigma_j^x + \sum_{i=0}^{N-1} \Delta_i\sigma_i^x$$

  - "[A]diabatic quantum computation (error free) is equivalent to the quantum circuit model (error free). So adiabatic quantum computers (error free) are quantum computers (error free) in the most traditional sense."

— *Dave Bacon, 27Feb2007*

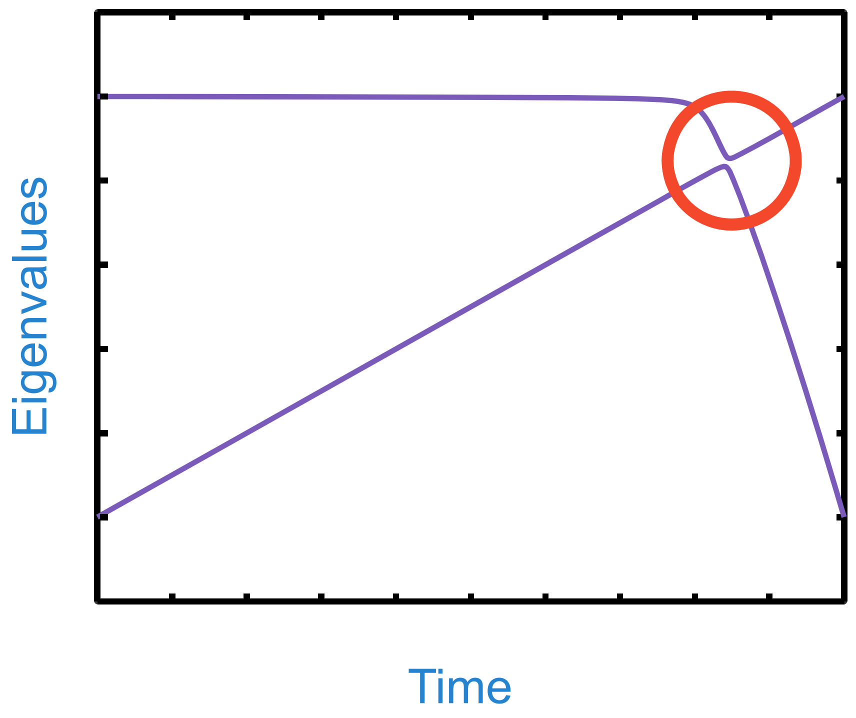
- **In this talk we'll be considering only adiabatic quantum optimization**
  - That's all that's been built to date (at least at large scale)

Annealing Time

- **From a few slides back:**
  - If we gradually decrease the amplitude of the transverse field, $\Gamma(t)$, from a very large value to 0, we should drive the system into the ground state of $\mathcal{H}_0$
- **What does "gradually" mean?**
  - (Explanation from Farhi and Gutmann)
  - $\mathcal{H}(t)$ encodes our problem
  - Want to evolve the system according to Schrödinger, $i\frac{d}{dt}|\psi\rangle = \mathcal{H}(t)|\psi\rangle$
  - Given that $\mathcal{H}(t)$ has one eigenvalue $E \neq 0$ and the rest 0, find the eigenvector $|w\rangle$ with eigenvector E
  - Assume we're given an orthonormal basis $\{|a\rangle\}$ with $a = 1, \dots, N$ and that $|w\rangle$ is one of those N basis vectors
  - Let $|s\rangle = \frac{1}{\sqrt{N}}\sum_{a=1}^{N}|a\rangle$
  - We consider the Hamiltonian $\mathcal{H} = E|w\rangle\langle w| + E|s\rangle\langle s|$ (i.e., problem + driver)
  - Let $x = \langle s|w\rangle$
  - Then, omitting a lot of math, we wind up with the probability at time t of finding the state $|w\rangle$ being $\Pr(t) = \sin^2(Ext) + x^2\cos^2(Ext)$
  - To find state $|w\rangle$ with (near) certainty we need to run for time $t_m = \frac{\pi}{2Ex}$

Determining the Annealing Time

- **Unfortunately, we don't in general know how long we need to run (i.e., we can't quickly compute t_m)**
- **Function of the minimum gap between the two smallest eigenvalues at any point during the Hamiltonian's time evolution**
- **Gap can get quite small**
- **Grover's search (right)**
  - Find an *n*-bit number such that
    $$\mathcal{H}_P|z\rangle = \begin{cases} |z\rangle & \text{if } z \neq w \\ 0 & \text{if } z = w \end{cases}$$
    for some black-box Hamiltonian $\mathcal{H}_P$
  - Here, $g_{\min} \simeq 2^{1-\frac{n}{2}}$ for n bits
  - Implication: Solution time is $O(2^n)$—no better than classical brute force



Two lowest eigenvalues for a Grover search, 12 bits

Image credit: Farhi, Goldstone, Gutmann, and Sipser (2000)

Annealing Time: Discussion

The bad

- **Very difficult to analyze an algorithm's computational complexity**
  - Need to know the gap between the ground state and first excited state, which can be costly to compute
  - In contrast, circuit-model algorithms tend to be more straightforward to analyze
- **Unknown if quantum annealing can outperform classical**
  - If gap always shrinks exponentially then no
  - (Known that in adiabatic quantum *computing* the gap shrinks polynomially)

Annealing Time: Discussion (cont.)

The good

- **Constants do matter**
  - If the gap is such that a correct answer is expected only once every million anneals, and an anneal takes 5μs, that's still only 5s to get a correct answer—may be good enough
  - On current systems, the gap scaling may be less of a problem than the number of available qubits
- **We may be able to (classically) patch the output to get to the ground state**
  - Hill climbing or other such approaches may help get quickly from a near-ground-state solution into the ground state
- **We may not even need the exact ground state**
  - For many optimization problems, "good and fast" may be preferable to "perfect but slow"

Outline

- Performance potential of quantum computing
- Quantum annealing
- Case study: D-Wave quantum annealers
- How to program a quantum annealer
- Parting thoughts

D-Wave's Hamiltonian

- **Problem Hamiltonian (longitudinal field):**

$$\mathcal{H}_P = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{i,j} \sigma_i^z \sigma_j^z + \sum_{i=0}^{N-1} h_i \sigma_i^z$$

Note:
This is a *classical* 2-local Ising Hamiltonian

  - The programmer specifies the $J_{i,j}$ and h_i, and the system solves for the σ_i^z
  - $\sigma_i^z \in \{-1, +1\}$
  - Nominally, $J_{i,j} \in \mathbb{R}$ and $h_i \in \mathbb{R}$, but the hardware limits these to a small set of distinguishable values in the ranges $J_{i,j} \in [-1, +1]$ and $h_i \in [-2, +2]$

- **Application of the time-dependent transverse field:**

$$\mathcal{H}_S(s) = \frac{\varepsilon(s)}{2} \mathcal{H}_P - \frac{\Delta(s)}{2} \sum_{i=0}^{N-1} \sigma_i^x$$

  - Programmer specifies the total annealing time, $T \in [5{,}2000]\ \mu\mathrm{s}$
  - $s = t/T$ (i.e., time normalized to [0, 1])
  - $\varepsilon(s)$ and $\Delta(s)$ are scaling parameters (not previously user-controllable but most recent hardware provides a modicum of control over the shape)

D-Wave’s Annealing Schedule

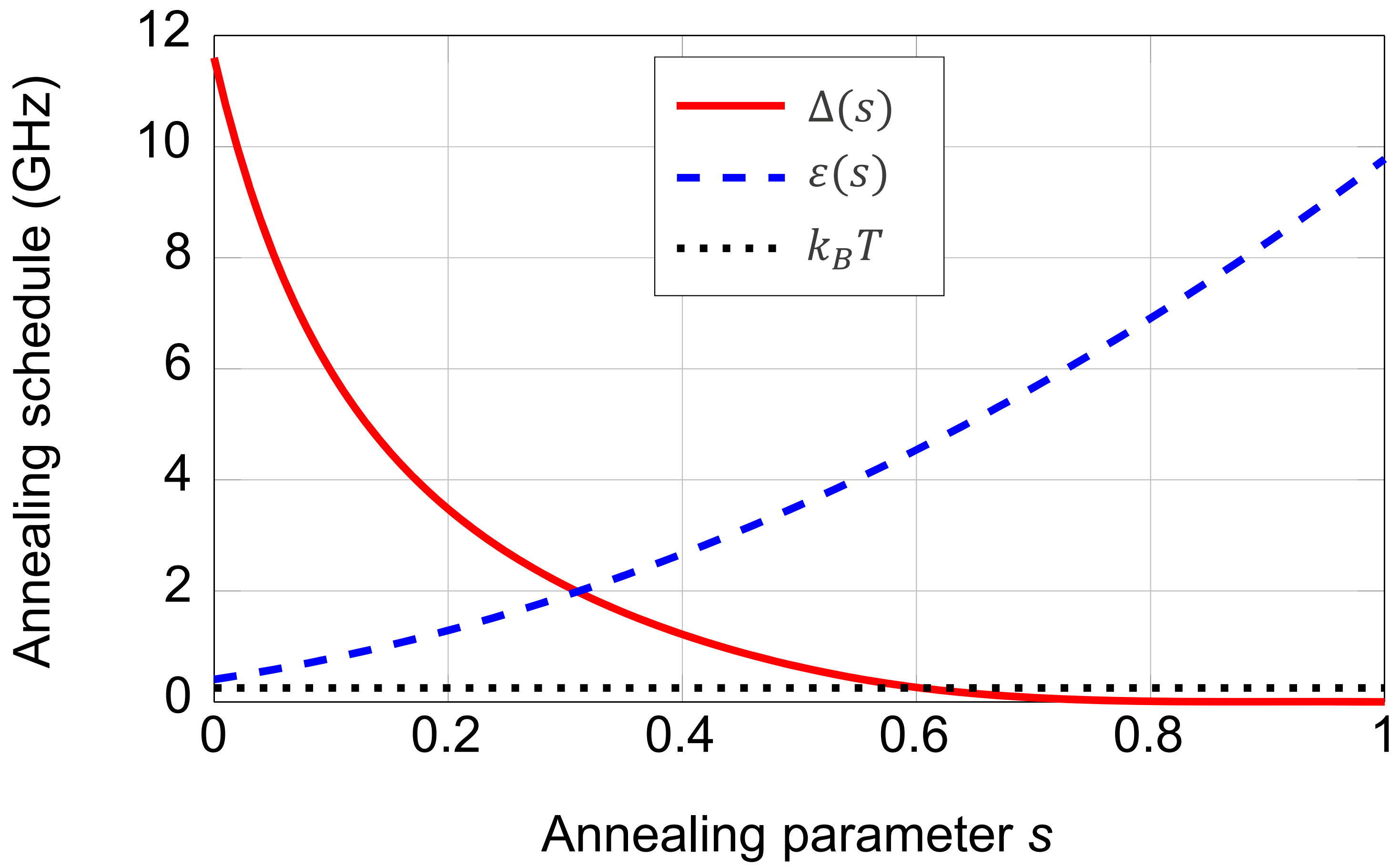
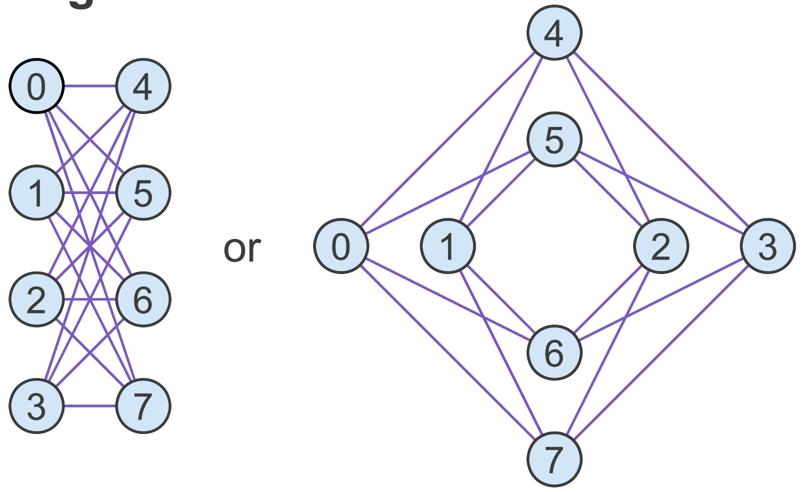


Image credit: King, Hoskinson, Lanting, Andriyash, and Amin (2016)

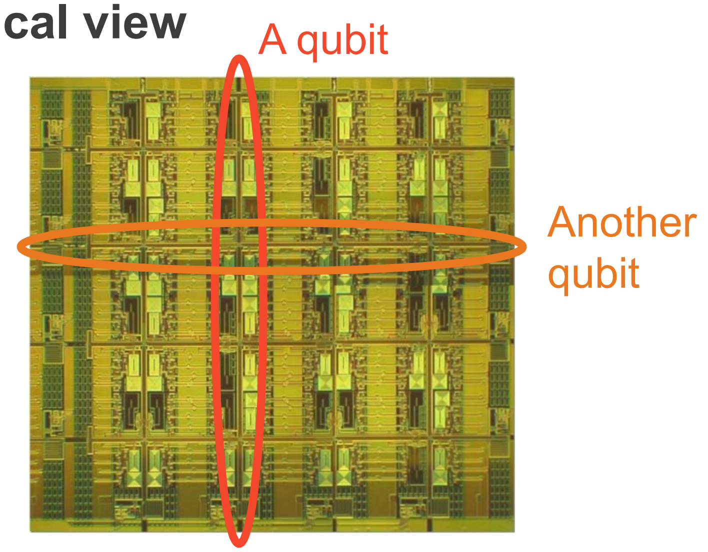
Building Block: The Unit Cell

- **Logical topology**
  - 8 qubits arranged in a bipartite graph
- **Physical implementation**
  - Based on rf-SQUIDs
  - Flux qubits are long loops of superconducting wire interrupted by a set of Josephson junctions (weak links in superconductivity)
  - "Supercurrent" of Cooper pairs of electrons, condensed to a superconducting condensate, flows through the wires
  - Large ensemble of these pairs behaves as a *single* quantum state with net positive or net negative flux
  - …or a superposition of the two (with tunneling)
  - Entanglement introduced at qubit intersections

- **Logical view**

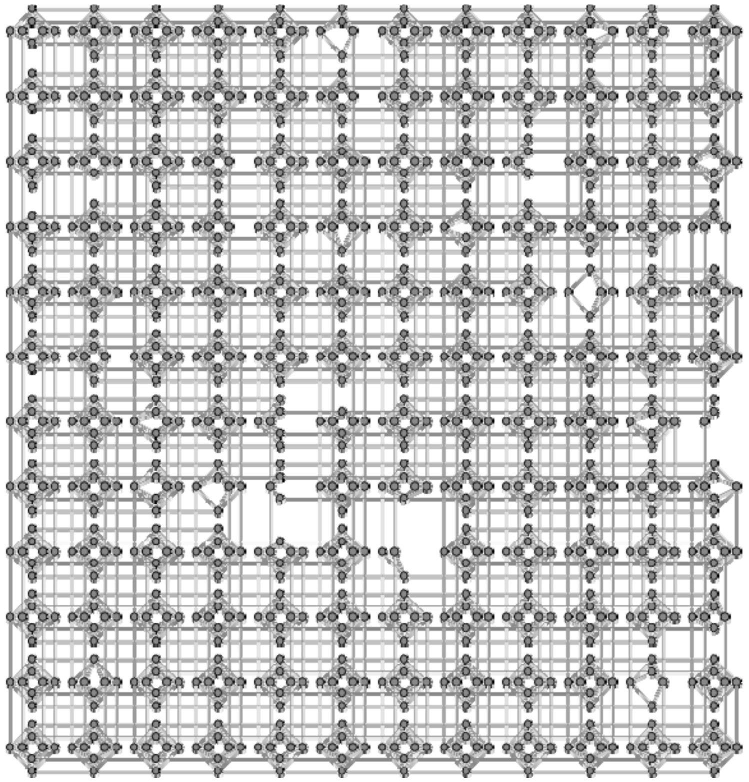


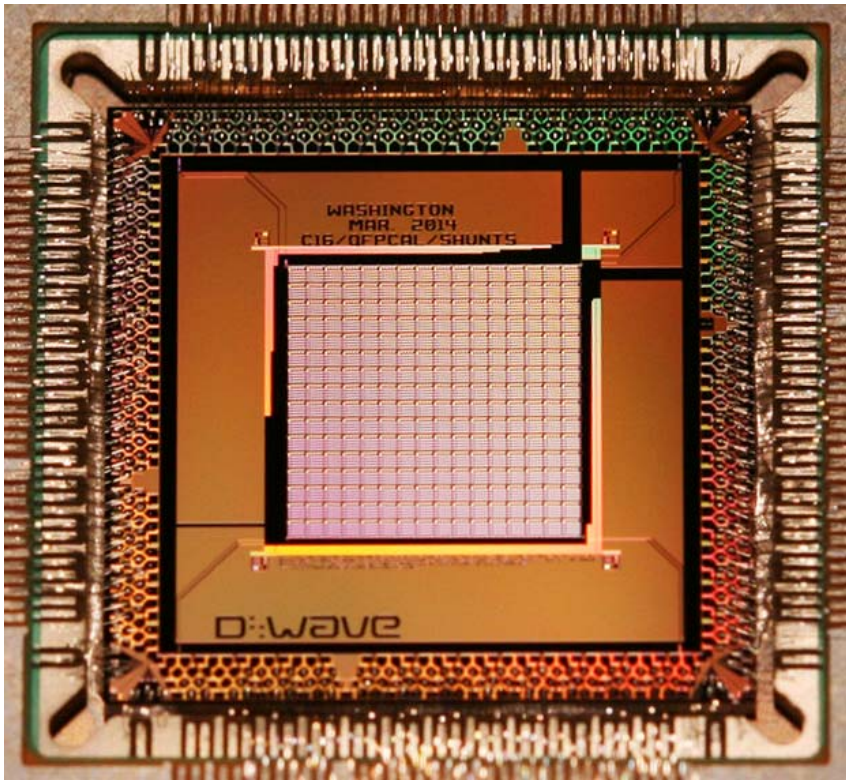
- **Physical view**



A Complete Chip

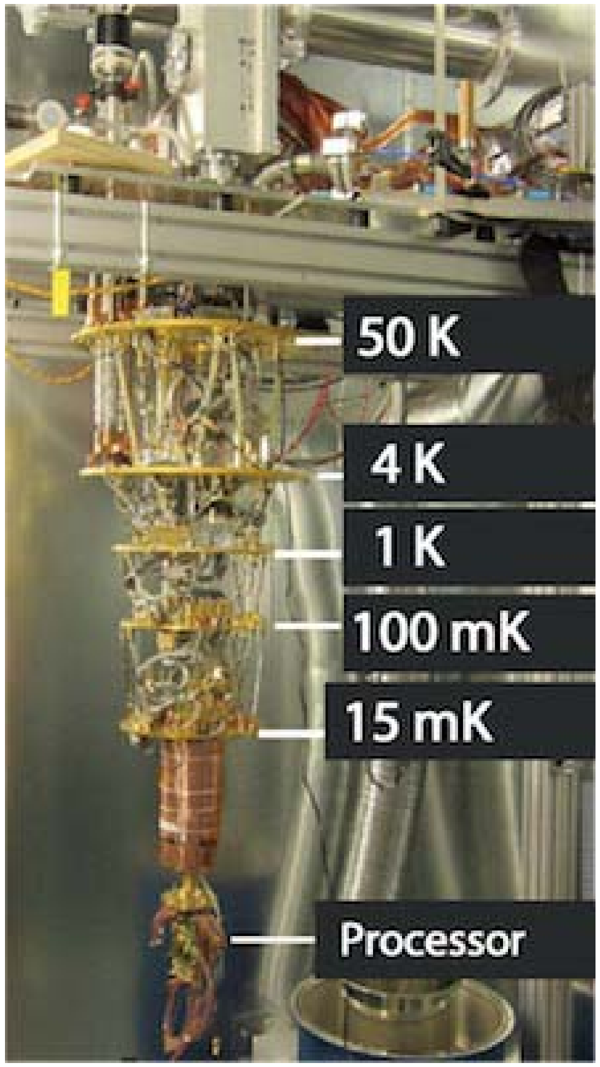
- **Logical view**
  - "Chimera graph": 16×16 unit-cell grid
  - Qubits 0–3 couple to north/south neighbors; 4–7 to east/west
  - Inevitably incomplete
- **Physical view**
  - Chip is about the size of a small fingernail
  - Can even make out unit cells with the naked eye





Cooling

- **Chip must be kept extremely cold for the macroscopic circuit to behave like a two-level (qubit) system**
  - *Much* below the superconducting transition temperature (9000 mK for niobium)
- **Dilution refrigerator**
- **Nominally runs at 15 mK**
- **LANL's D-Wave 2X happens to run at 10.45 mK**
  - That's 0.01°C above absolute zero
  - For comparison, interstellar space is far warmer: 2700 mK



What You Actually See

- **A big, black box**
  - 10’×10’×12’ (3m×3m×3.7m)
  - Mostly empty space
  - Radiation shielding, dilution refrigerator, chip + enclosure, cabling, tubing
  - LANL also had to add a concrete slab underneath to reduce vibration
- **Support logic**
  - Nondescript classical computers
  - Send/receive network requests, communicate with the chip, etc.

Deviation from the Theoretical Model

- **No all-to-all connectivity**
  - Each qubit can be directly coupled to at most 6 other qubits
  - Many qubits and couplers are absent (in an irregular, installation-specific pattern)
- **Not running at absolute zero**
- **Not running in a perfect vacuum**
- **No error correction**
- **We can therefore think of our Hamiltonian as being**

$$\mathcal{H}_S(s) = \frac{\varepsilon(s)}{2}\left(\sum_{\langle i,j\rangle} J_{i,j}\sigma_i^z\sigma_j^z + \sum_{\langle i\rangle} h_i\sigma_i^z\right) - \frac{\Delta(s)}{2}\sum_{\langle i\rangle} h_i\sigma_i^x + \mathcal{H}_?(s)$$

- **in which $\mathcal{H}_?(s)$ encapsulates the interaction with the environment**
  - That is, all the things we don't know and can't practically measure
  - Nonlinear and varies from run to run
- **Also, it takes time to set up a problem and get the results back**
  - *Before*: reset + programming + post-programming thermalization
  - *After*: readout
  - Currently, these dominate the annealing time by many orders of magnitude