DCSE summerschool 2023

Matthias Möller
Numerical Analysis @ DIAM
Quantum Computing Lab

Quantum computing in the news

Article

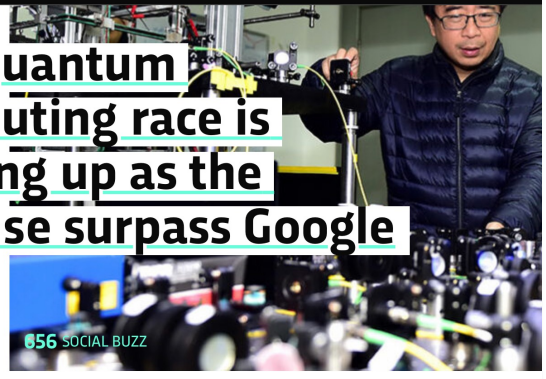
Quantum supremacy using a programmable superconducting processor

https://doi.org/10.1038/s41586-019-1666-5
Received: 22 July 2019
Accepted: 20 September 2019
Published online: 23 October 2019

Frank Arute[1], Kunal Arya[1], Ryan Babbush[1], Dave Bacon[1], Joseph C. Bardin[1,2], Rami Barends[1], Rupak Biswas[3], Sergio Boixo[1], Fernando G. S. L. Brandao[1,4], David A. Buell[1], Brian Burkett[1], Yu Chen[1], Zijun Chen[1], Ben Chiaro[5], Roberto Collins[1], William Courtney[1], Andrew Dunsworth[1], Edward Farhi[1], Brooks Foxen[1,5], Austin Fowler[1], Craig Gidney[1], Marissa Giustina[1], Rob Graff[1], Keith Guerin[1], Steve Habegger[1], Matthew P. Harrigan[1], Michael J. Hartmann[1,6], Alan Ho[1], Markus Hoffmann[1], Trent Huang[1], Travis S. Humble[7], Sergei V. Isakov[1], Evan Jeffrey[1],

TECHWIRE ASIA — Insights Latest Popular

The quantum computing race is heating up as the Chinese surpass Google



Chinese quantum computer surpasses Google's 'quantum supremacy'.

By Dashveenjit Kaur | 20 July, 2021

- China is unveiling a super-advanced 66-qubit quantum supercomputer called "Zuchongzhi"
- The Chinese team claims that it has solved a problem in just over an hour that would otherwise take the world's most powerful classical supercomputer eight years to crack.

Quantum Computing

On "Quantum Supremacy"

October 21, 2019 | Written by: Edwin Pednault, John Gunnels & Dmitri Maslov, and Jay Gambetta

Recent advances in quantum computing have resulted in two 53-qubit processors: one from our group in IBM and a device described by Google in a paper published in the journal *Nature*. In the paper, it is argued that their device reached "quantum supremacy" and that "a state-of-the-art supercomputer would require approximately 10,000 years to perform the equivalent task." *We argue that an ideal simulation of the same task can be performed on a classical system in 2.5 days and with far greater fidelity*. This is in fact a conservative, worst-case estimate, and we expect that with additional refinements the classical cost of the simulation can be further reduced.

Quantum computing in Europe

Computertechnologie

Ein Quantensprung für Deutschland?

Stand: 15.06.2021 18:25 Uhr

In Ehningen bei Stuttgart wurde Europas erster Quantencomputer eingeweiht. Der ultraschnelle Rechner der Firma IBM soll der Wirtschaft helfen, im Wettstreit mit China und den USA zu bestehen.

TECHNOLOGY NEWS MAY 11, 2021 / 10:52 AM / UPDATED 5 MONTHS AGO

Germany to support quantum computing with 2 billion euros

By Reuters Staff 2 MIN READ

TNO FOCUS AREAS COLLABORATION OUR IMPACT ABOUT TNO CAREER

THE FIRST EUROPEAN ONLINE QUANTUM COMPUTER PLATFORM

4 May 2020 • 3 min reading time

Leading universities and quantum hubs from China to America and the Netherlands are working on the development of a usable quantum computer. Within QuTech, TNO is working on innovative quantum technology in collaboration with Delft University of Technology and with some success, because a new version of the 'Quantum Inspire' quantum computing platform was launched on 20 April 2020. It is, in fact, the first European quantum computer platform that is generally accessible online.

09.04.2021 . Awards

Quantum Delta NL awarded 615 million euro from Netherlands' National Growth Fund to accelerate quantum technology

A Tale of Two Paradigms

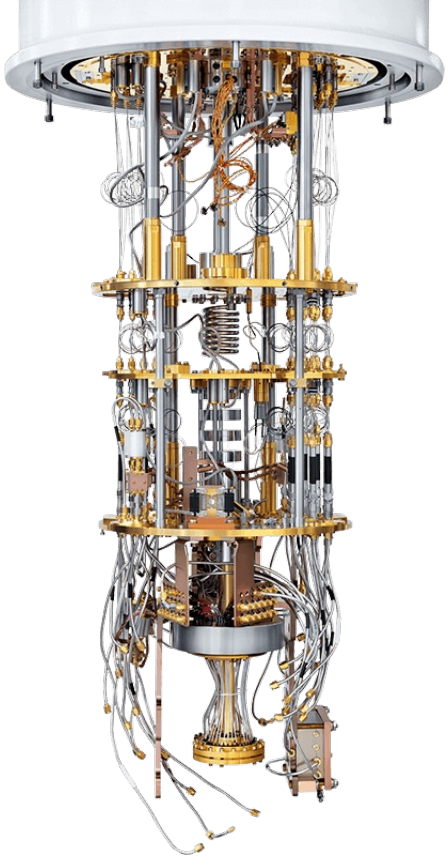
Digital QC
universal/programmable
~ 100 qubits
TRL 4-5 (TRL 9 expected 2035)
few use cases/algorithms

Analog QC
task specific (comb. optimization)
5000+ qubits
TRL 8-9
many use cases

(theoretical) speed-up

optimal solutions

Sources: https://www.fz-juelich.de/ias/jsc/EN/Research/ModellingSimulation/QIP/QTRL/_node.html
Michielsen K., FZ Jülich: Quantum Annealing for Optimization and Classification | D-Wave Qubits 2021

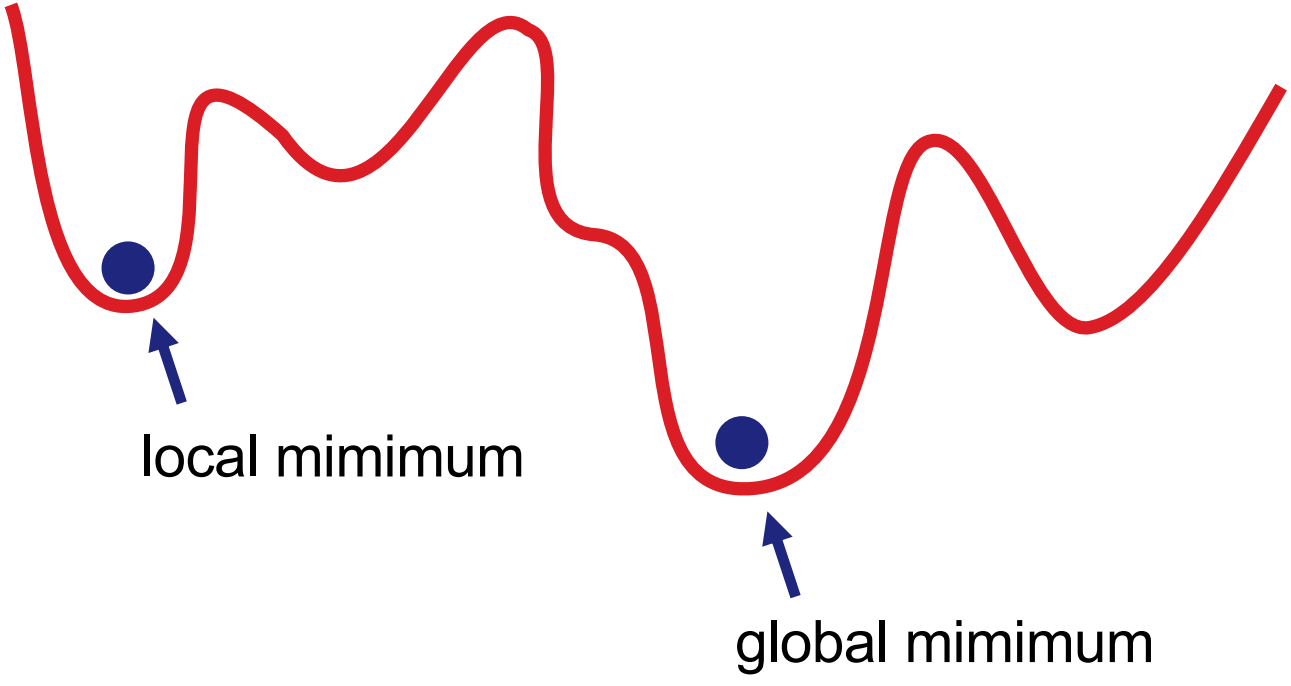


(theoretical) speed-up

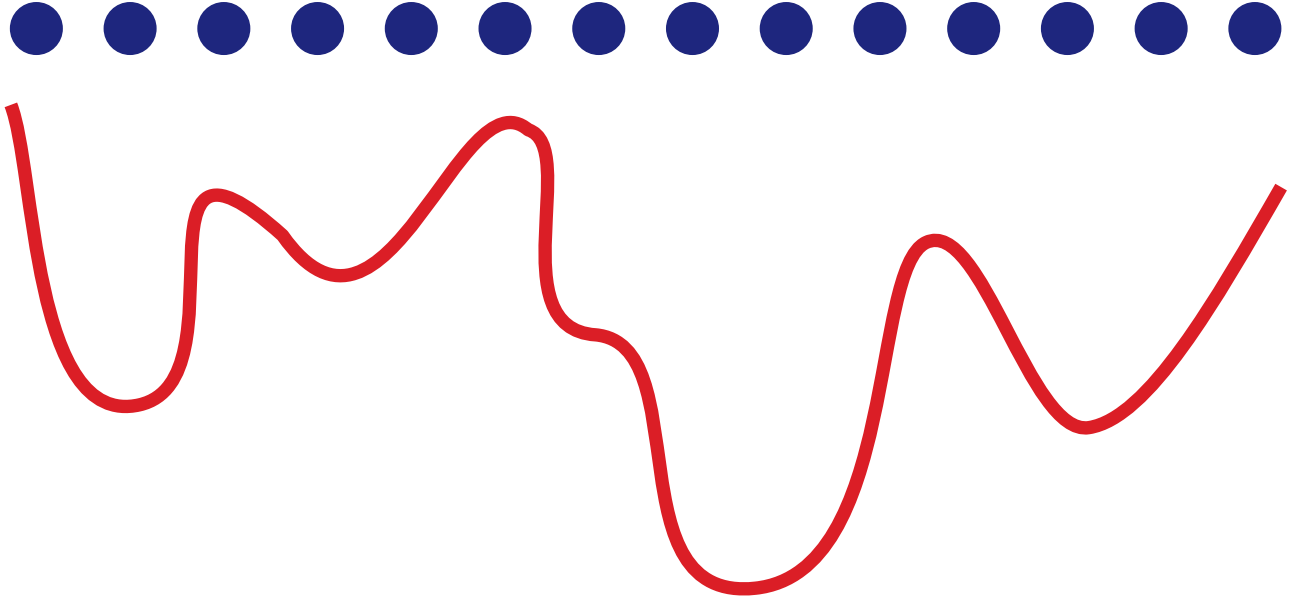
optimal solutions

Sources: https://www.fz-juelich.de/ias/jsc/EN/Research/ModellingSimulation/QIP/QTRL/_node.html
Michielsen K., FZ Jülich: Quantum Annealing for Optimization and Classification | D-Wave Qubits 2021

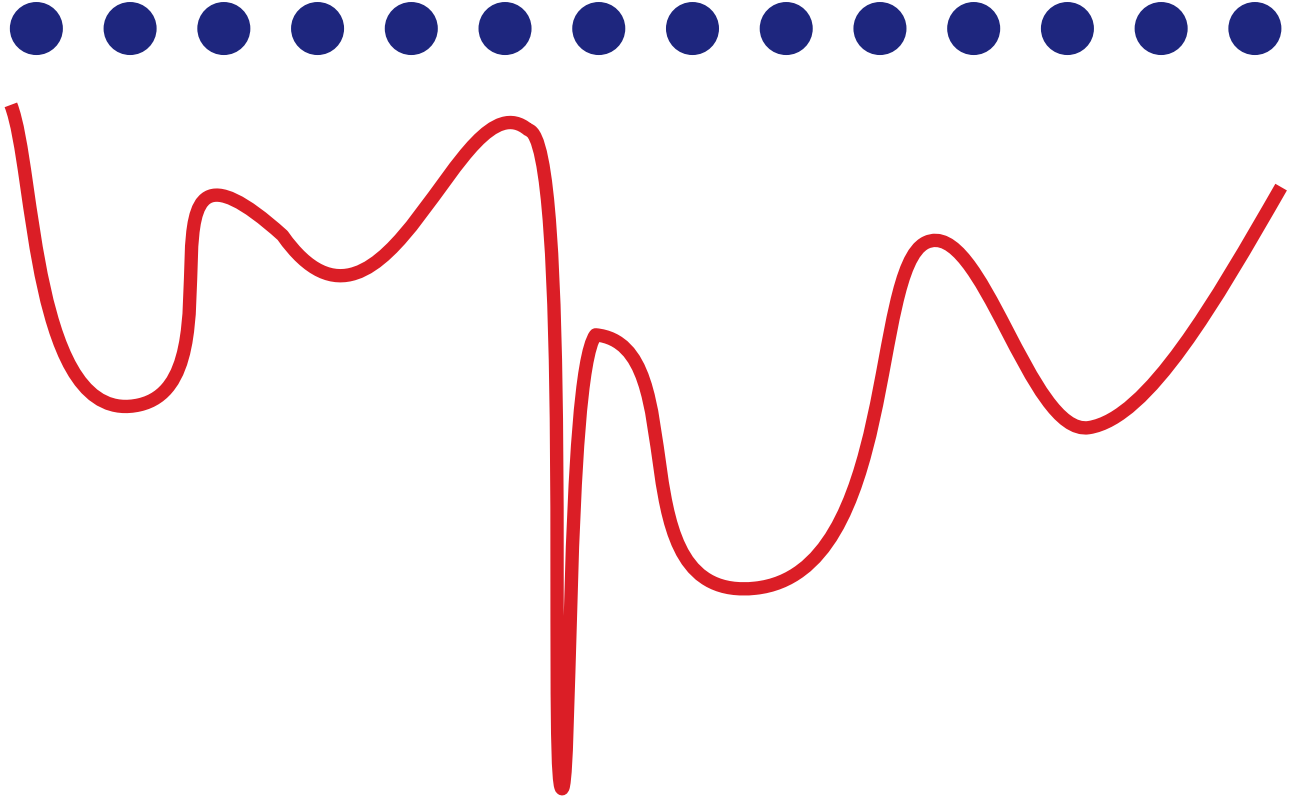
Finding optimal solution



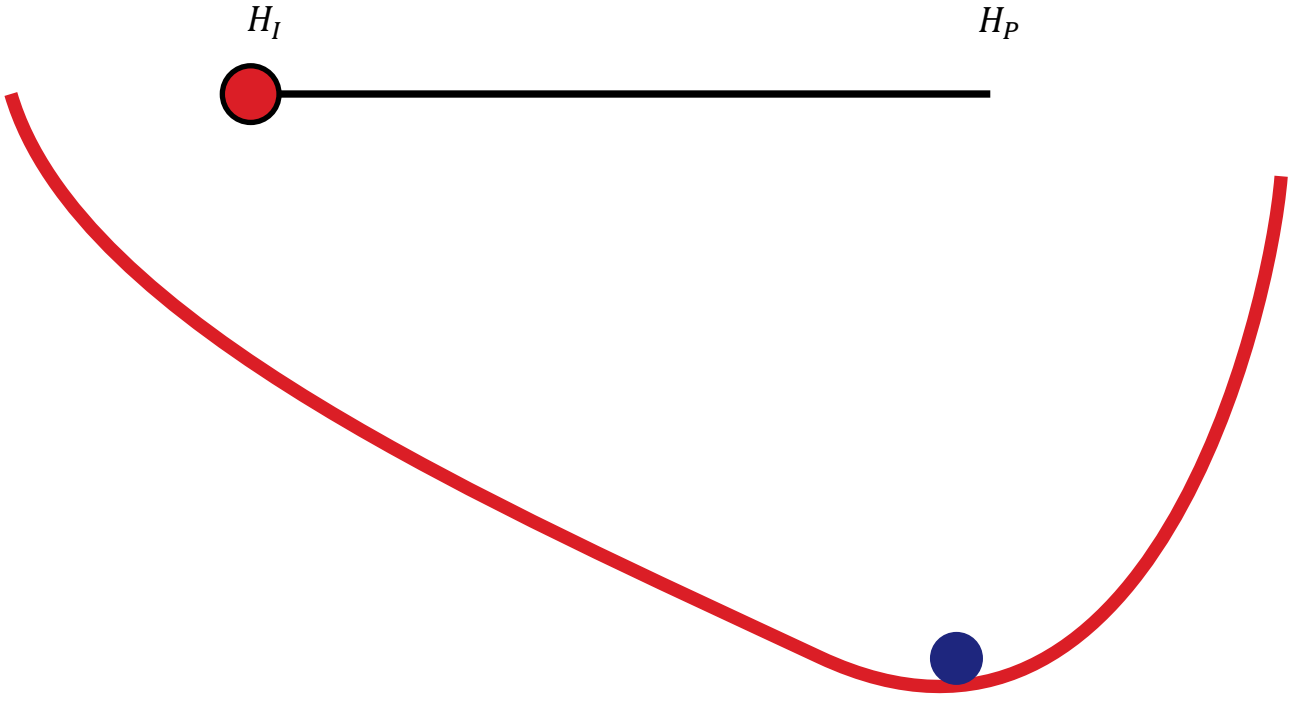
Brute force sampling



Brute force sampling

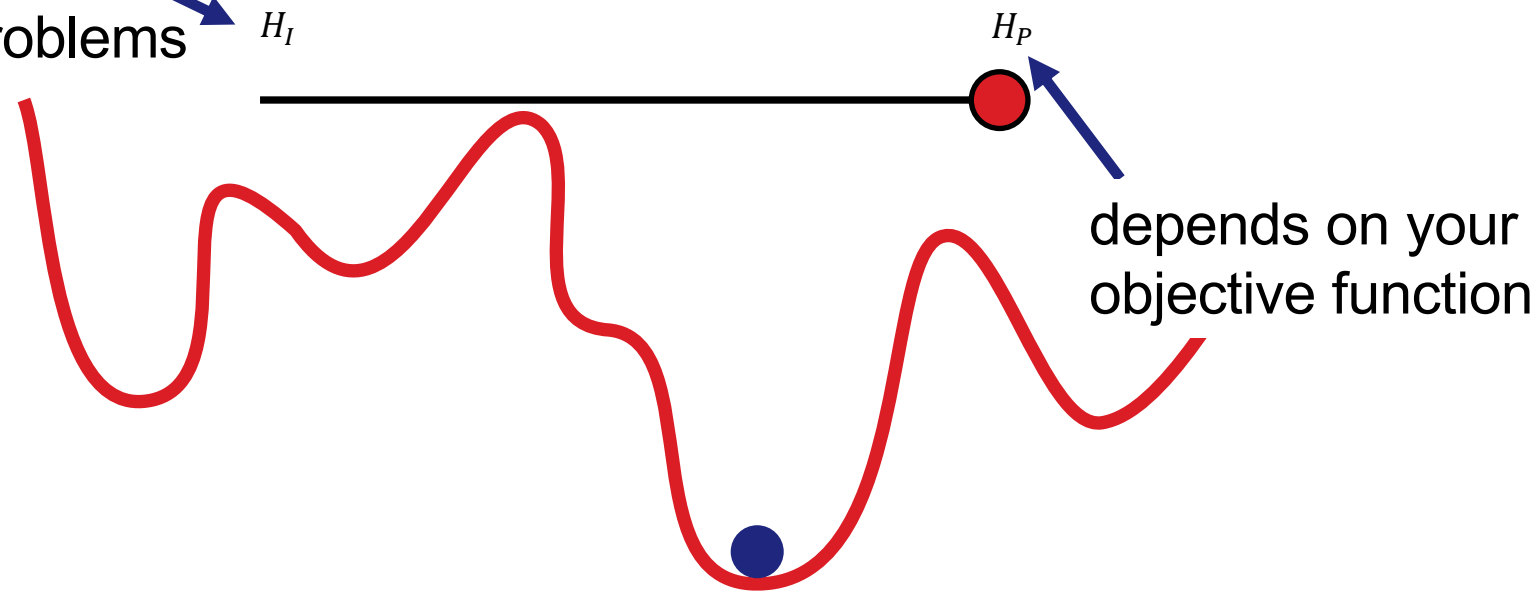


Quantum annealing

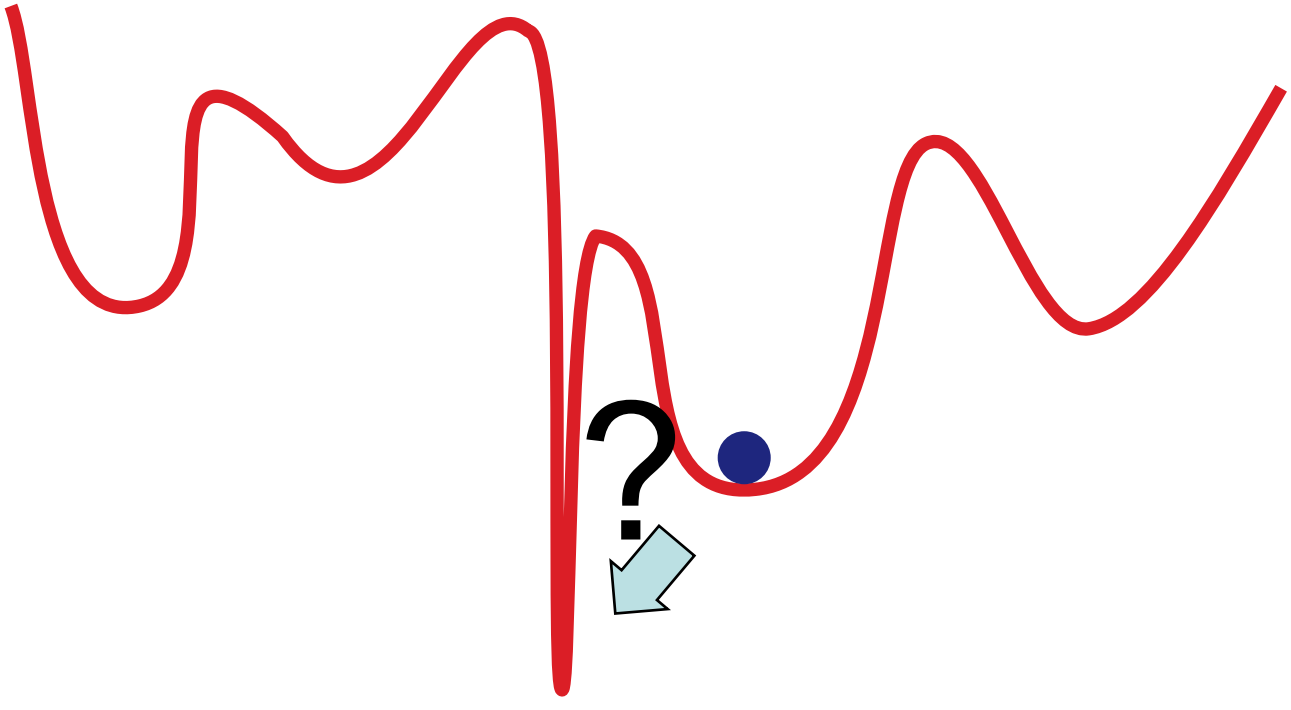


Quantum annealing

same for
all problems



Quantum tunneling



Linear algebra intermezzo – Hermitian matrices

- **Definition 1**: A matrix $A \in \mathbb{C}^{n\times n}$ is termed
  - Hermitian if $A^\dagger = A$ and
  - anti-Hermitian if $A^\dagger = -A$.

- **Theorem 2**: The eigenvalues of a Hermitian matrix are real

$$A|\psi_i\rangle = \lambda_i|\psi_i\rangle, \qquad \lambda_i \in \mathbb{R}$$

  and the eigenvectors corresponding to distinct eigenvalues are orthogonal

$$\langle\psi_i|\psi_j\rangle = \delta_{ij}.$$

Linear algebra intermezzo – Hermitian matrices

- **Theorem 3**: Hermitian matrices are diagonalizable

$$A = R\Lambda R^{-1}.$$

- **Theorem 4**: Hermitian matrices can be written in terms of their eigenbasis

$$A = \sum_{i=1}^{n} \lambda_i |\psi_i\rangle\langle\psi_i|$$

Ground-state problem

- Simulating a quantum mechanical system in equilibrium often amounts to finding the ground state of a Hamiltonian that is modelling this system.

- **Definition 5: (Ground-state problem)**: For an input Hamiltonian H, find an eigenstate $|\psi_0\rangle$ of H with the smallest eigenvalue λ_0 (read: with minimal energy). The state $|\psi_0\rangle$ is called the ground state of H and is the solution to

$$|\psi_0\rangle = \arg\min_{|\psi\rangle} \langle\psi|H|\psi\rangle$$

- Without going into further details, we use that the Hamiltonian is Hermitian.

The variational principle

- **Theorem 6**: The expectation value of a Hermitian matrix H on any state $|\psi\rangle$ is bounded from below by its smallest eigenvalue λ_0

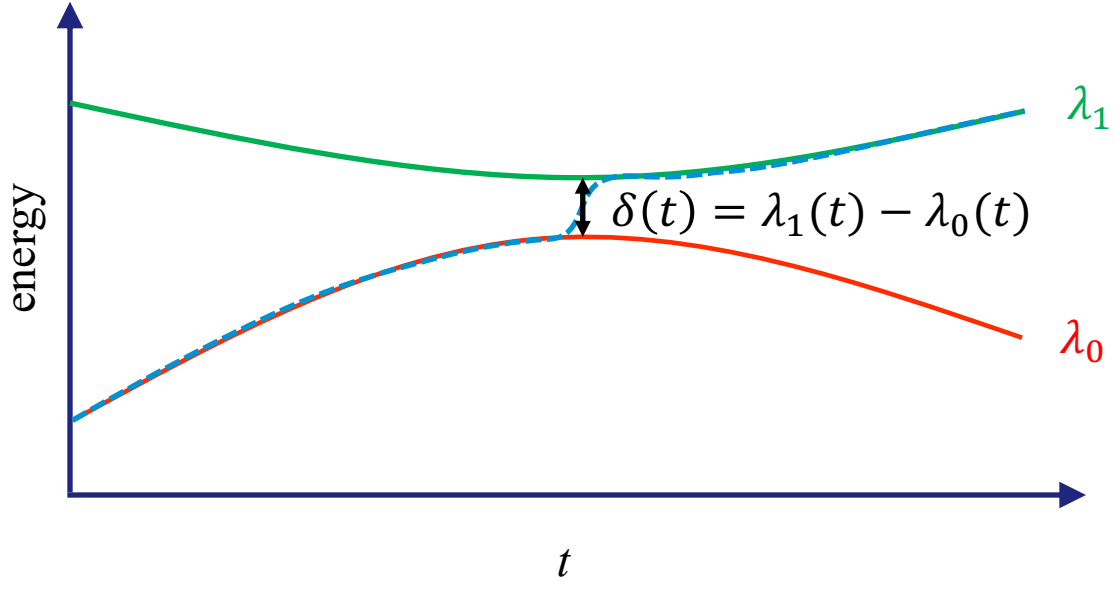
$$\begin{aligned}\langle\psi|H|\psi\rangle &= \langle\psi|\left(\sum_{i=1}^{n}\lambda_i|\psi_i\rangle\langle\psi_i|\right)|\psi\rangle \\ &= \sum_{i=1}^{n}\lambda_i\langle\psi|\psi_i\rangle\langle\psi_i|\psi\rangle \\ &= \sum_{i=1}^{n}\lambda_i \underbrace{|\langle\psi|\psi_i\rangle|^2}_{\geq 0} \geq \lambda_0\end{aligned}$$

- Equality is obtained for the ground state $|\psi_0\rangle$:

$$\langle\psi_0|H|\psi_0\rangle = \langle\psi_0|\lambda_0|\psi_0\rangle = \lambda_0\langle\psi_0|\psi_0\rangle = \lambda_0$$

Adiabatic algorithm

- Adiabatic quantum computing solves the ground-state problem by starting with a simple problem and varying it ‘smoothly’ to the ground-state problem



Adiabatic theorem

Theorem 7: Let $H(t)$ be a smooth family of Hamiltonians parametrized by $t \in [0, T]$, such that

- $H(0) = H_I$ (initial Hamiltonian),
- $H(T) = H_P$ (problem Hamiltonian) and
- $H(t)$ is gapped for all t, that is, $\delta(t) \coloneqq \lambda_1(t) - \lambda_0(t) \neq 0$.

If we initialize a system in the ground state $|\psi(0)\rangle$ of H_I and let it evolve with $H(t)$ for a time $T \gg 1/\Delta_{\min}^2$ following the time-dependent Schrödinger equation

$$H(t)|\psi(t)\rangle = -i\frac{d}{dt}|\psi(t)\rangle$$

then $|\psi(T)\rangle$ will be the ground state of the problem Hamiltonian H_P.

Practical aspects

- Examples of a smooth family of Hamiltonians

$$H(t) = s(t)H_I + \big(1 - s(t)\big)H_P, \qquad 0 \le t \le T$$

$$s\colon [0,T] \to [0,1] \text{ strictly decreasing, e.g., } s(t) = 1 - t/T$$

- Rule of thumb for rate of change of Hamiltonian

$$\tau(t) \gg \int_0^1 \frac{\left\| \frac{d}{dt} H(t) \right\|}{(\delta_m)^2} ds\,, \qquad \delta_m = \min_{0 \le t \le 1} \lambda_1(t) - \lambda_0(t)$$

Example: 2-bit disagree problem – Farhi et al.

- Objective function

$$f(x) = \begin{cases} 0 & \text{if } x_1 \neq x_2 \\ 1 & \text{otherwise} \end{cases}$$

- Problem Hamiltonian

$$H_P = \frac{1 + \sigma_1^z \sigma_2^z}{2}$$

with Pauli matrix $\sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

$$H_P|00\rangle \rightarrow 1\,|00\rangle \text{ via } \frac{1 + (+1)(+1)}{2}$$

$$H_P|01\rangle \rightarrow 0\,|01\rangle \text{ via } \frac{1 + (+1)(-1)}{2}$$

$$H_P|10\rangle \rightarrow 0\,|10\rangle \text{ via } \frac{1 + (-1)(+1)}{2}$$

$$H_P|11\rangle \rightarrow 1\,|11\rangle \text{ via } \frac{1 + (-1)(-1)}{2}$$

The eigenstates and eigenvalues of H_P correspond exactly to $f(x)$

Example: 2-bit disagree problem – Farhi et al.

- Initial Hamiltonian

$$H_I = \sum_{i=1}^{2} \frac{1 - \sigma_i^x}{2}$$

with Pauli matrix $\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

- Eigenvectors

$$|x_+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad |x_-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

We obtain

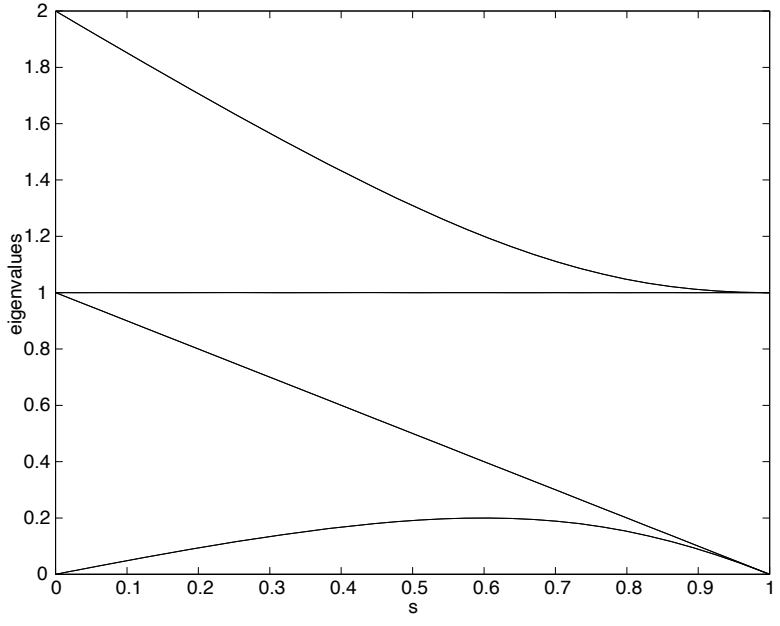
$$H_I|x_+x_+\rangle \to 0|x_+x_+\rangle$$
$$H_I|x_+x_-\rangle \to 1|x_+x_-\rangle$$
$$H_I|x_-x_+\rangle \to 1|x_-x_+\rangle$$
$$H_I|x_-x_-\rangle \to 2|x_-x_-\rangle$$

The ground state can be rewritten as

$$|x_+x_+\rangle = \tfrac{1}{2}|00\rangle + \tfrac{1}{2}|01\rangle + \tfrac{1}{2}|10\rangle + \tfrac{1}{2}|11\rangle$$

This means that each eigenstate in the standard basis is equally likely to be observed.

Example: 2-bit disagree problem – Farhi et al.



From: Farhi et al.: Quantum Computation by Adiabatic Evolution, arXiv: quant-ph/0001106v1

Ising model

- Collection of n particles arranged on the vertices of a graph $G = (V, E)$. Each particle can be in one of the two states ± 1 called **spins**.
- A spin configuration $s = s_1 \ldots s_n$ is an assignment of spin values to particles.
- The energy of a given spin configuration is defined by

$$H(s) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j$$

with **external forces** h_i and **interaction forces** J_{ij} between particles

Ising model – 2-spin example

$$H(-1,-1) = .5(-1) + (-.3)(-1) + .4 = .2$$
$$H(-1,+1) = .5(-1) + (-.3)(+1) - .4 = -1.2$$
$$H(+1,-1) = .5(+1) + (-.3)(-1) - .4 = .4$$
$$H(+1,+1) = .5(+1) + (-.3)(+1) + .4 = .6$$

The optimal solution with smallest energy value equal to -1.2 is $s = (-1,+1)$.

Ising Hamiltonian

- Given a collection of n qubits arranged on the vertices of graph $G = (V, E)$.
- Let the Ising Hamiltonian be given by

$$H = \sum_{i \in V} h_i \sigma_i^z + \sum_{(i,j) \in E} J_{ij} \sigma_i^z \sigma_j^z$$

- Recall that

$$\sigma^z |0\rangle = +1|0\rangle = (-1)^0 |0\rangle$$
$$\sigma^z |1\rangle = -1|1\rangle = (-1)^1 |1\rangle$$

- Let us apply H to the quantum state $|\psi\rangle = |\psi_1 \psi_2 \ldots \psi_n\rangle$

Ising Hamiltonian

- Ground state problem

$$\langle\psi|H|\psi\rangle = \sum_{i\in V} h_i(-1)^{\psi_i} + \sum_{(i,j)\in E} J_{ij}(-1)^{\psi_i}(-1)^{\psi_j}$$

$$= \sum_{i\in V} h_i s_i + \sum_{(i,j)\in E} J_{ij} s_i s_j\,, \qquad s_i \coloneqq (-1)^{\psi_i} \in \{-1,1\}$$

From Ising model to QUBO formulation

- Given the weights of an Ising model, the problem can be easily converted into a Quadratic Unconstrained Binary Optimization (QUBO) problem

$$f(x) = \sum_{i,j} Q_{ij} x_i x_j$$

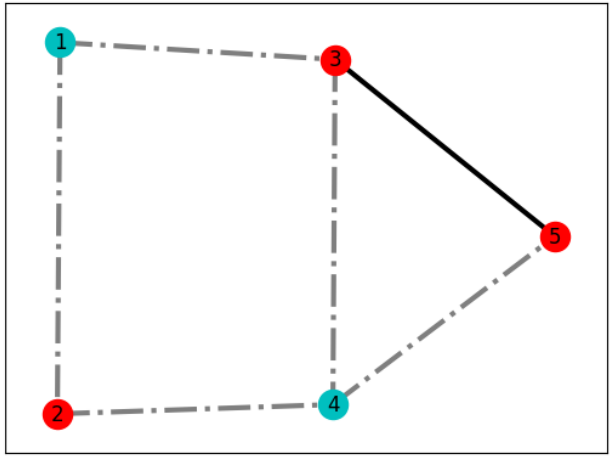
with binary variables $x_i \epsilon \{0,1\}$ by letting $s_i = 1 - 2x_i$ and adjusting the weights of the symmetric matrix Q_{ij} accordingly.

- It is also possible to derive QUBO formulations directly.

Example: maximum-cut problem

- **Problem description**: Given a graph $G = (V, E)$ the goal is to partition the graph's vertices V into two disjoint sets S and T, such that the number of edges between S and T is as large as possible. Finding such a partitioning is known as the **maximum-cut problem**.

- **Remark**: For general graphs G, the max-cut problem is NP-hard



NetworkX – network analysis in Python

- Create an undirected graph

```
import networkx as nx

G = nx.Graph()
G.add_edges_from([(1,2),(1,3),(2,4),(3,4),(3,5),(4,5)])
```

- Draw graph

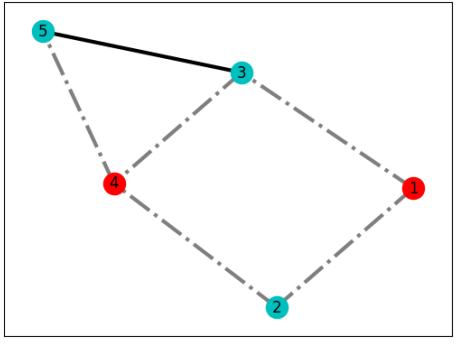
```
import matplotlib.pyplot as plt

nx.draw_networkx(G, with_labels=True)
plt.show()
```

Example: maximum-cut problem

- `python maximum_cut.py`

```
-----------------------------------------------------------------
          Set 0          Set 1    Energy        Cut Size
-----------------------------------------------------------------
         [1, 4]      [2, 3, 5]     -5.0             5
         [2, 3]      [1, 4, 5]     -5.0             5
      [1, 4, 5]         [2, 3]     -5.0             5
      [2, 3, 5]         [1, 4]     -5.0             5
```

```
Your plot is saved to maxcut_plot.png
```



Formulating QUBO problems

- **Recap**: Quadratic Unconstrained Binary Optimization (QUBO) problem

$$\min_{x \in \{0,1\}^n} f(x), \qquad f(x) = \sum_{i,j} Q_{ij} x_i x_j, \qquad Q_{ij} \in \mathbb{R}$$

- **What we have to do**
  1. *Define binary variables x_i. Keep in mind that the QA will just return a bitstring; we need to give a problem-specific meaning to it.*
  2. *Define QUBO matrix Q such that the minimum QUBO solution is an optimal solution to the original problem.*

The maximum-cut QUBO problem

- A possible approach

$$f(x) = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} \begin{bmatrix} -1 & 2 & 2 & & \\ 2 & -1 & & 2 & \\ 2 & & -1 & 2 & 2 \\ & 2 & 2 & -1 & 2 \\ & & 2 & 2 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

with the following meaning of the binary variables

- $x_i = 0 :\Leftrightarrow i$-th node is in partition S
- $x_i = 1 :\Leftrightarrow i$-th node is in partition T

The maximum-cut QUBO problem

- Let us consider the following case

$$\begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}-1 & 2 & 2 & & \\ 2 & -1 & & 2 & \\ 2 & & -1 & 2 & 2 \\ & 2 & 2 & -1 & 2 \\ & & 2 & 2 & -1\end{bmatrix}\cdot\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix} = -1$$

- Here we get a negative reward which is good, because we minimize
- This applies to any bitstring with a single bit set to one and all others zero

The maximum-cut QUBO problem

- Let us consider the following case

$$\begin{bmatrix}1 & 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}-1 & 2 & 2 & & \\ 2 & -1 & & 2 & \\ 2 & & -1 & 2 & 2 \\ & 2 & 2 & -1 & 2 \\ & & 2 & 2 & -1\end{bmatrix}\cdot\begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} = -1+2+2-1 = 2$$

- Here we get a negative reward of '-1' twice but also have to pay twice by two positive costs of '2' ending up in a higher overall value
- The positive costs make bitstrings with nodes connected by an edge being in the same partition unfavorable. But do they guarantee valid cuts?

The maximum-cut QUBO problem

- Let us consider the following case

$$\begin{bmatrix}1 & 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}-1 & 2 & 2 & & \\ 2 & -1 & & 2 & \\ 2 & & -1 & 2 & 2 \\ & 2 & 2 & -1 & 2 \\ & & 2 & 2 & -1\end{bmatrix}\cdot\begin{bmatrix}1\\0\\0\\1\\0\end{bmatrix} = -1 - 1 = -2$$

- Here we get two negative reward of '-1' and no positive costs which is optimal. At the same time this is a valid (and optimal) partitioning.

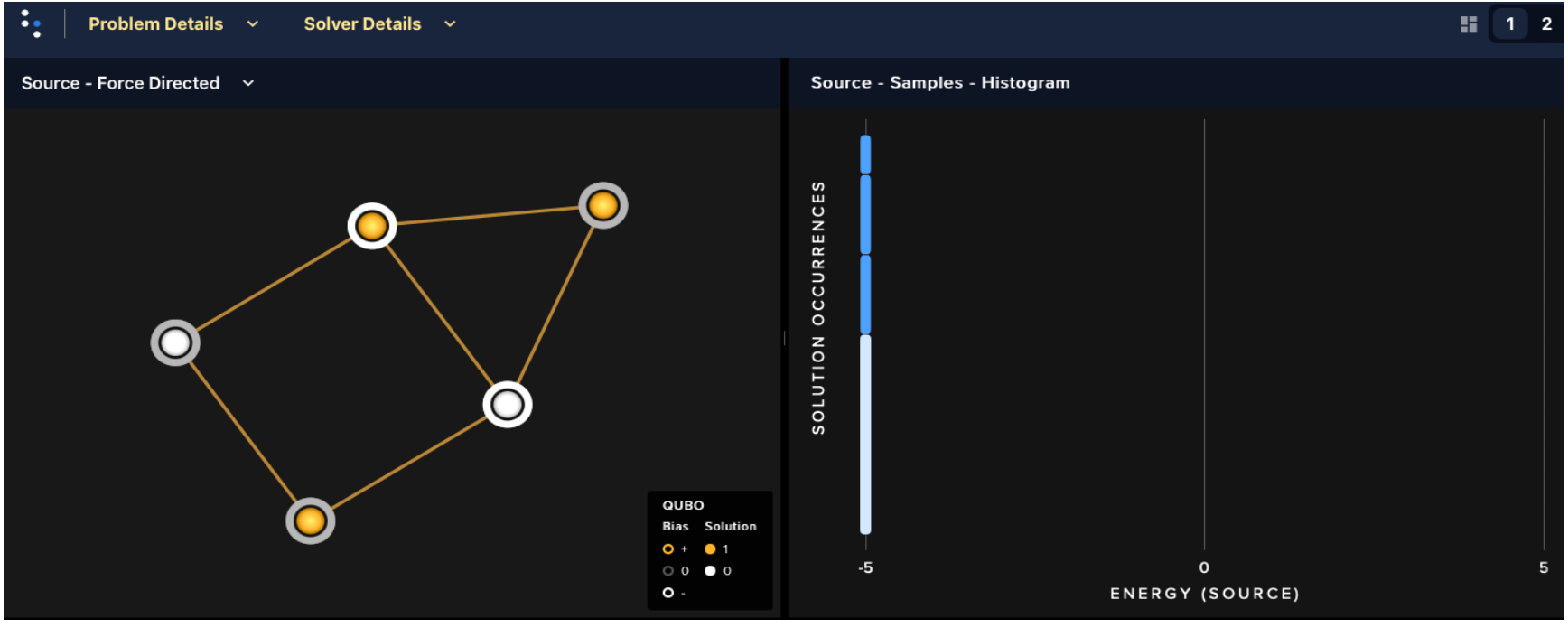
Preparing the QUBO matrix as a dictionary

```
from collections import defaultdict

# Initialize our Q matrix
Q = defaultdict(int)

# Update Q matrix for every edge in the graph
for i, j in G.edges:
    Q[(i,i)]+= -1
    Q[(j,j)]+= -1
    Q[(i,j)]+= 2
```

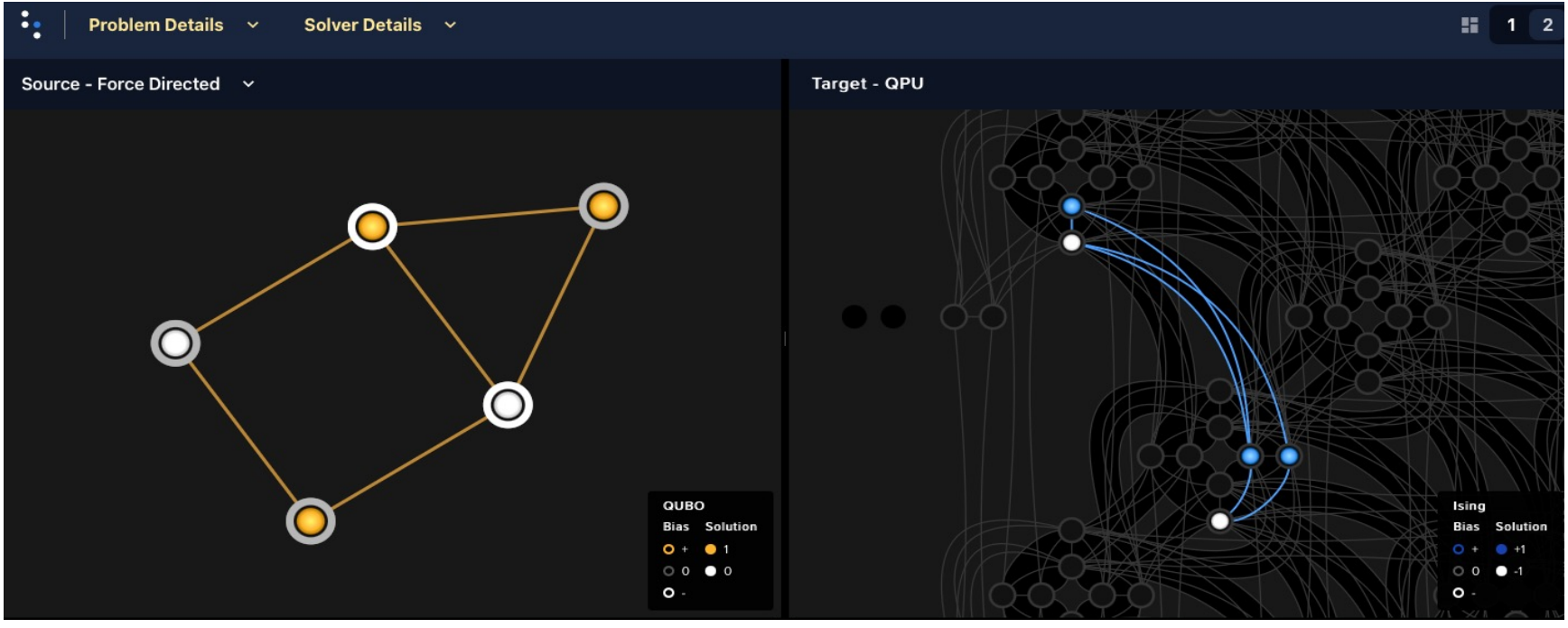
Running the QUBO problem on DWaveSampler

```
from dwave.system.samplers import DWaveSampler
from dwave.system.composites import EmbeddingComposite
import dwave.inspector

sampler = EmbeddingComposite(DWaveSampler())
response = sampler.sample_qubo(Q,
                               chain_strength=8,
                               num_reads=10,
                               label='Example - Maximum Cut')

dwave.inspector.show(response)
```

Dwave inspectore – view 1

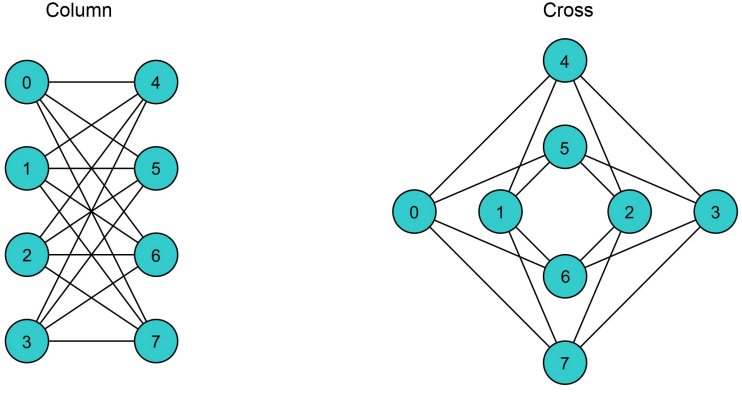


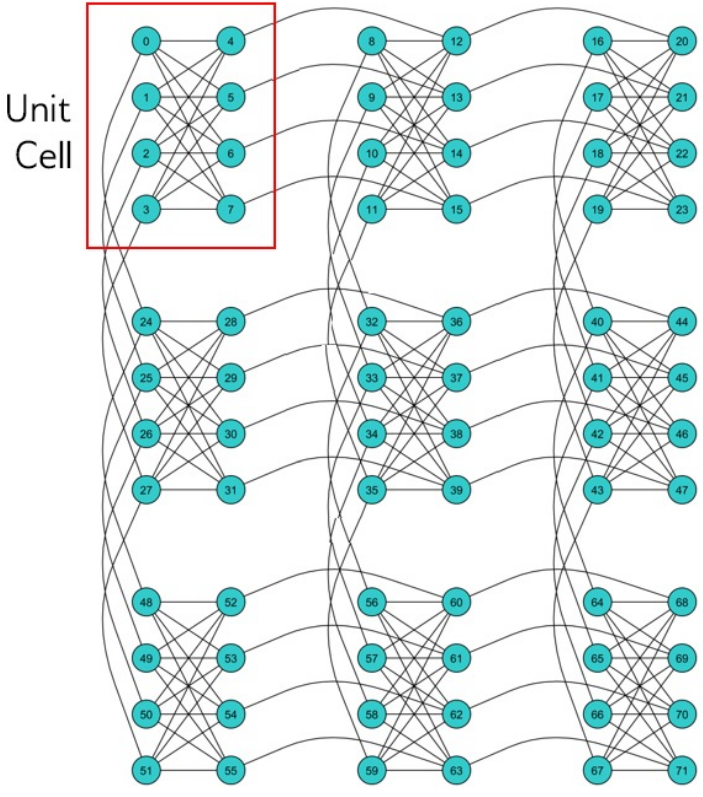
Dwave inspectore – view 2



Minor embedding

- The hardware graph consists of many small clusters of **densely connected qubits** which are connected to qubits from other clusters **partially via couplers**

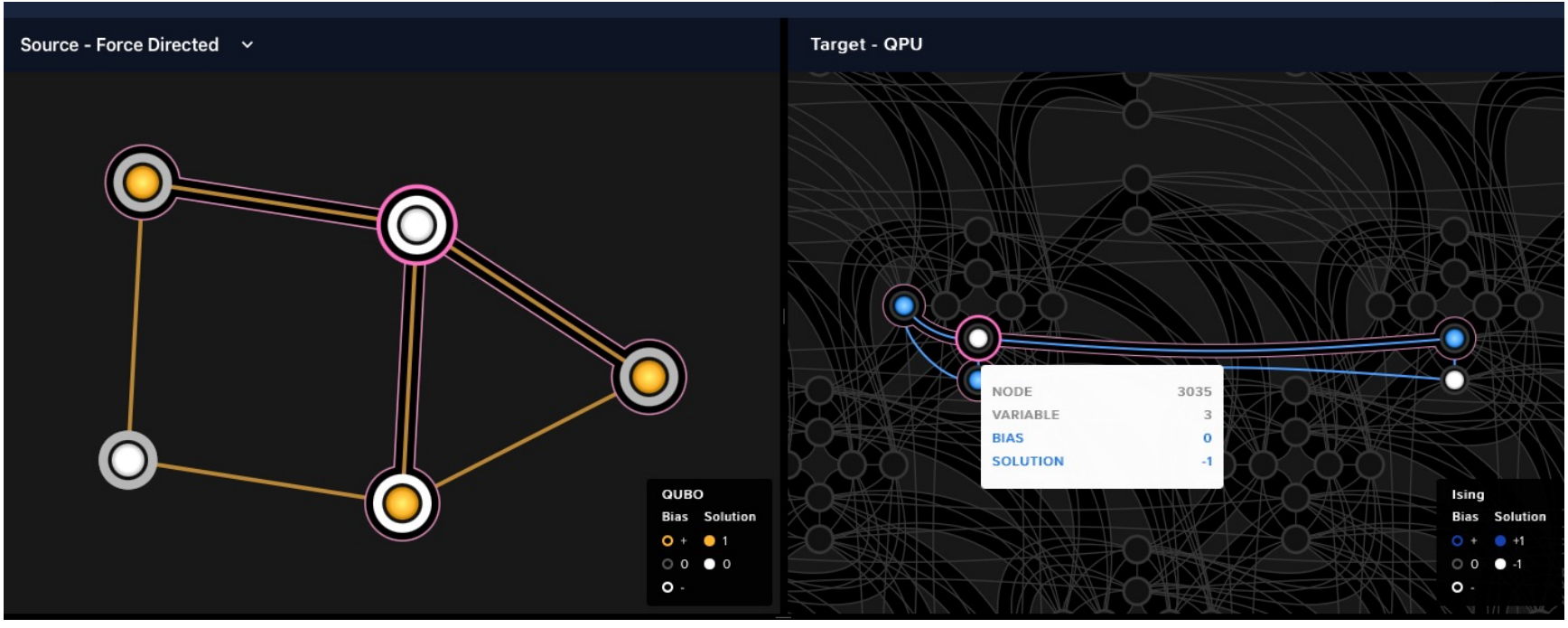




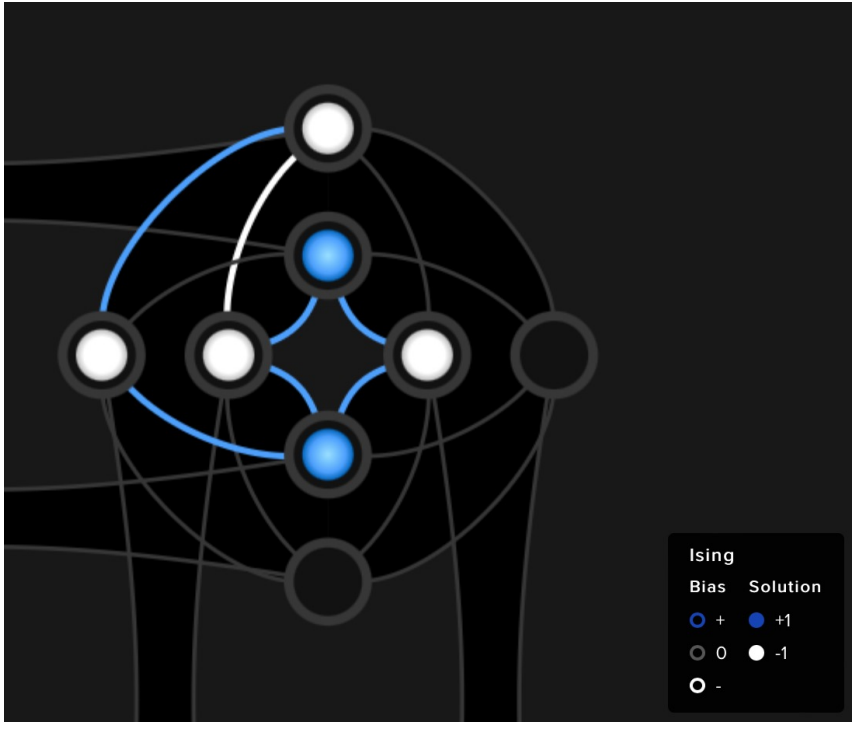
Minor embedding using minorminer

```
from dwave.system.composites import FixedEmbeddingComposite
import minorminor

sampler = DWaveSampler()
hwGraph = sampler.to_networkx_graph()

# find a valid mapping from problem to hardware graph, e.g.
# {1: [3036], 2: [3051], 3: [3035], 4: [3050], 5: [1080]}
embedding = minorminer.find_embedding(Q, hwGraph)

fixedSampler = FixedEmbeddingComposite(sampler, embedding)
response     = fixedSampler.sample_qubo(…)
```

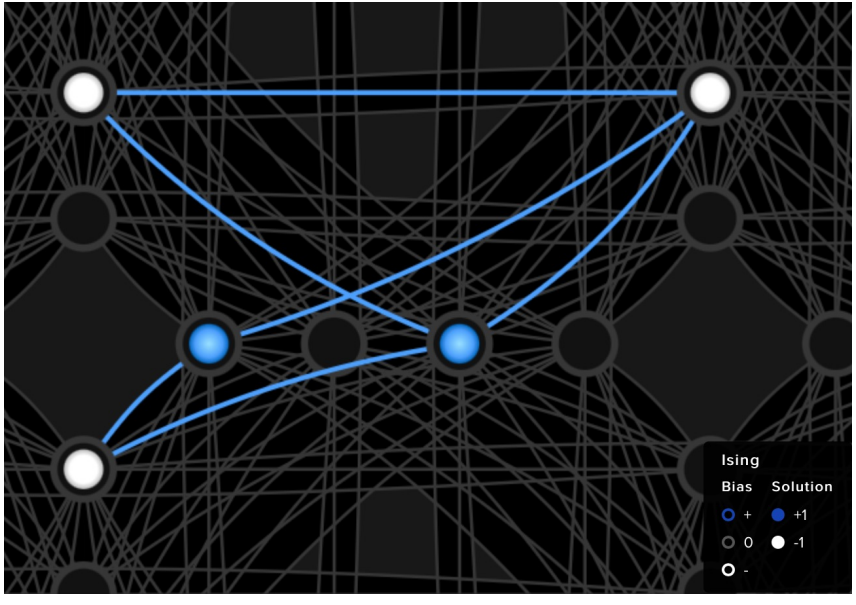
Dwave inspectore – view 2



Different DWaveSamplers have different topologies



DWave 2000Q



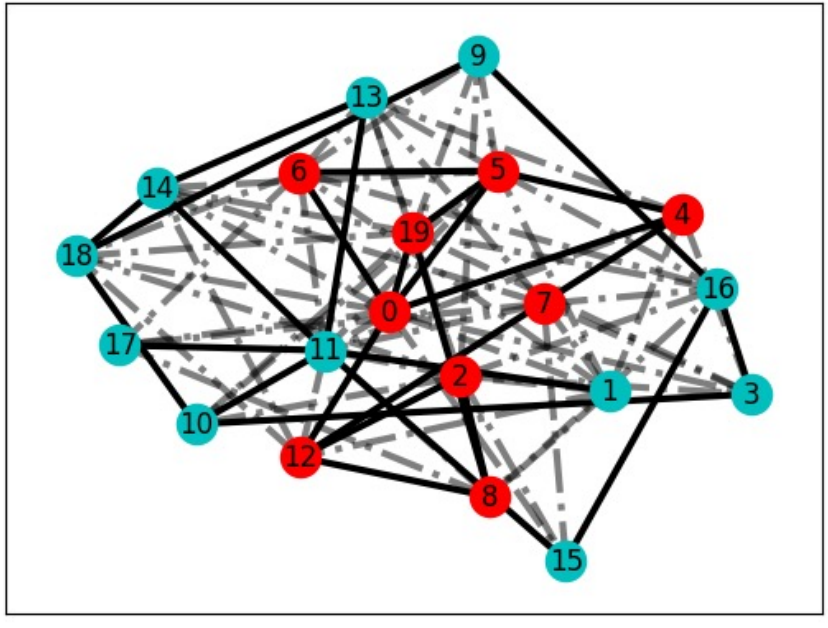
DWave Advantage2 (prototype)

Choosing a specific DWaveSampler

```
sampler = DWaveSampler(region='…',solver='…')
```

region	solver
eu-central-1	Advantage_system5.3
na-west-1	Advantage_system6.1
	Advantage_system4.1
	Advantage2_prototype1.1
	DW_2000Q_6

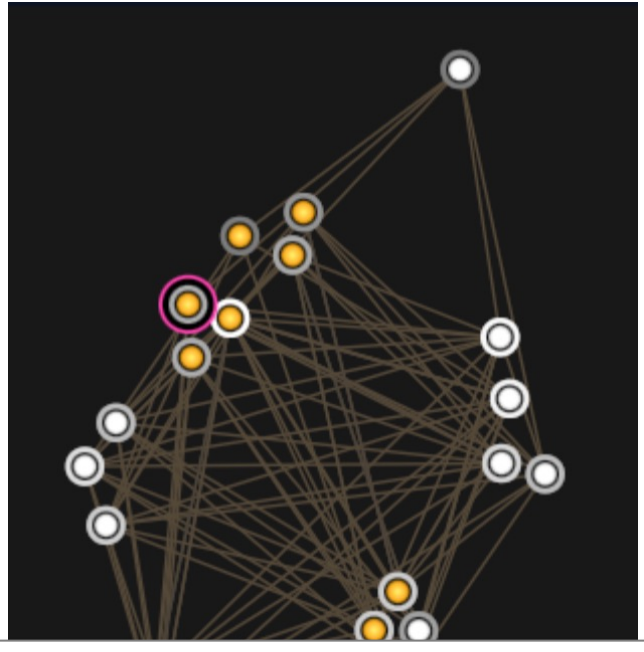
What if the problem graph cannot be embedded?

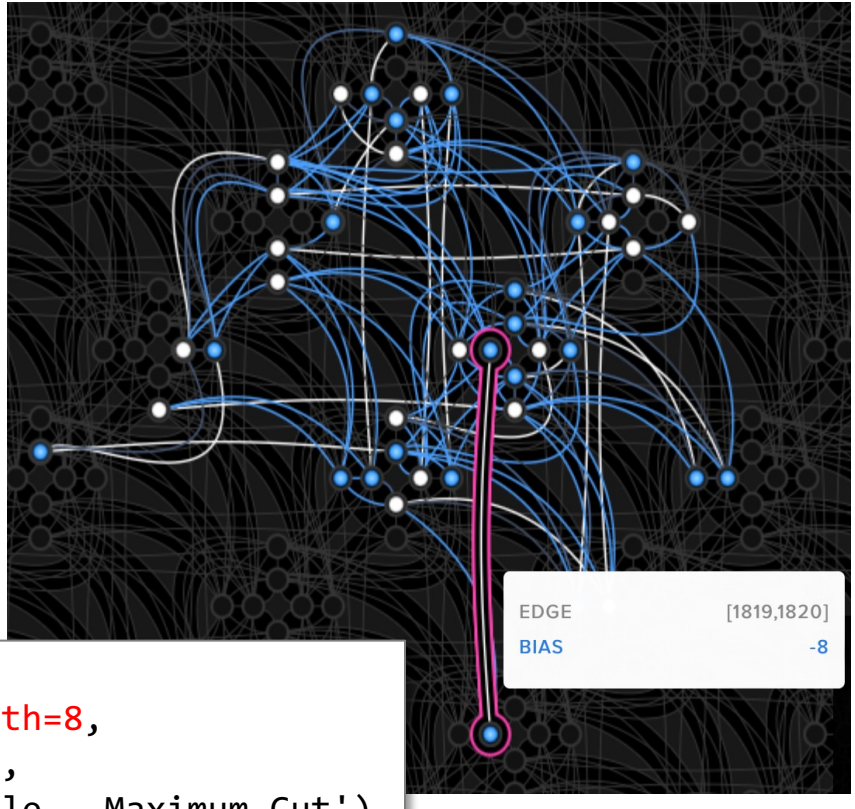


maximum_cut_large.py

```
# Create graph with 20 nodes
# and probability of 50% of
# any two nodes to be
# connected by an edge
G = nx.gnp_random_graph(20, 0.5)
```

Duplication of variables and weak coupling by chains






```
response = sampler.sample_qubo(Q,
                               chain_strength=8,
                               num_reads=10,
                               label='Example - Maximum Cut')
```

Quiz

$$f(x) = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} \begin{bmatrix} -1 & 2 & 2 & & \\ 2 & -1 & & 2 & \\ 2 & & -1 & 2 & 2 \\ & 2 & 2 & -1 & 2 \\ & & 2 & 2 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

1. How would the above QUBO problem change if x_1 would be duplicated and x_1 would be connected to x_2 and x_1' would be connected to x_3?
2. What is a good value for the chain strength between x_1 and x_1'? Why?

Quiz

$$f(x) = [x_1 \quad x_1' \quad x_2 \quad x_3 \quad x_4 \quad x_5] \begin{bmatrix} -1 & -s & & 2 & & \\ -s & -1 & 2 & & & \\ & 2 & -1 & & 2 & \\ 2 & & & -1 & 2 & 2 \\ & & 2 & 2 & -1 & 2 \\ & & & 2 & 2 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_1' \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

- All weights in a given QUBO are scaled to [-1,+1] before execution on the QA. If the chain strength is larger than the largest absolute value in the QUBO, chain strengths are set to one, and the QUBO weights are proportionally smaller. If $s \to \infty$ the pair (x_1, x_1') would act like a separate entity not interacting with any other node of the graph. We would thus not solve the original problem anymore.

Variable chains

- Variables can be duplicated more than once and chained together, e.g.,

- The longer the chains the more difficult it is to ensure consistency among all variables, i.e. $x_1 = x_1' = x_1'' = x_1'''$. The DWave inspector warns you about too long chains (but does not offer any other help to overcome this)

Minor embedding is NP-hard

- The minorminer (https://github.com/dwavesystems/minorminer) implements a heuristic strategy from arxiv:1406.2741 to find a good minor embedding
- Minor embedding can take a long time and the algorithm can also fail
- **Tip**: store your embeddings with Python's pickel (see next slide)
- The minorminer package offers to specialized embeddings
  - `minorminer.busclique.find_clique_embedding()`
  - `minorminer.layout.find_embedding()`

Storing minor embedding with pickel

```
import pickle; from hashlib import md5

hwGraphHash = md5(str(sorted(hwGraph.edges())).encode('utf-8')+
                  str(sorted(hwGraph.nodes())).encode('utf-8')).digest().hex()
QHash = md5(str(sorted(Q.keys())).encode('utf-8')).digest().hex()


# try to find pre-computed embedding of QUBO on hardware graph
embedding = None
try:
    file = open('.embedding-'+str(QHash)+'-'+str(hwGraphHash), 'rb')
    embedding = pickle.load(file)
    file.close()
except:
    embedding = minorminer.find_embedding(Q, hwGraph)
    file = open('.embedding-'+str(QHash)+'-'+str(hwGraphHash), 'wb')
    pickle.dump(embedding, file)
    file.close()
```

Constrained optimization intermezzo

- Minimization problem with equality constraints

$$\min_x f(x)$$
$$\text{s.t. } g(x) = 1$$

- Lagrange multiplier approach (penalty term)

$$\min_x f(x) + \lambda P(x), \qquad P(x) = (g(x) - 1)^2, \qquad 0 \ll \lambda < \infty$$

Constrained optimization intermezzo

- Minimization problem with inequality constraints

$$\min_x f(x)$$
$$\text{s.t. } g(x) < 1$$

- Slack variable approach

$$\min_x f(x)$$
$$\text{s.t. } g(x) + s = 1$$
$$s > 0$$

QUBO with constraints

- Equivalence of binary variables

$$x_i = x_i^2, \qquad x_i \in \{0,1\}$$

- Enforce binary value 0 or 1

$$x_i = 0 \quad \rightarrow \quad \lambda(x_i)^2$$
$$x_i = 1 \quad \rightarrow \quad \lambda(x_i - 1)^2 = \lambda\left(x_i^2 - 2x_i - 1\right) = \gamma\left(-x_i^2 - 1\right) \quad \rightarrow \quad -\lambda x_i^2$$

The constant term can be dropped as it is added unconditionally and only reduces the cost function by an additive term. Don't forget to include it in the post-processing if you need the exact value of your loss function.

QUBO with constraints

- Enforce 'k out of n' binary variables to be selected

$$\lambda\left(\sum_{i=1}^{n} x_i - k\right)^2$$

- **Example**: $x_1 + x_2 = 1 \quad \rightarrow \quad \lambda(2x_1x_2 - x_1^2 - x_2^2)$

  $(x_1 + x_2 - 1)^2 = x_1^2 + 2x_1x_2 + x_2^2 - 2x_1 - 2x_2 + 1 = 2x_1x_2 - x_1^2 - x_2^2 + 1$

  Again, the constant term can be dropped for the QUBO.

QUBO with linear equality constraints

- Enforce m equality constraints on n binary variables

$$\boldsymbol{Cx} = \boldsymbol{c}, \qquad \boldsymbol{C} \in \mathbb{R}^{m \times n}, \qquad \boldsymbol{c} \in \mathbb{R}^{m}$$

- Penalty term

$$P(\boldsymbol{x}) = \boldsymbol{\Lambda} \|\boldsymbol{Cx} - \boldsymbol{c}\|^2 = \sum_{k=1}^{m} \lambda_k (\langle \boldsymbol{C}_k, \boldsymbol{x} \rangle - c_k)^2 , \qquad \lambda_k > 0$$

QUBO with linear inequality constraints

- Enforce m inequality constraints on n binary variables

$$\boldsymbol{Dx} \leq \boldsymbol{d}, \qquad \boldsymbol{D} \in \mathbb{R}^{m \times n}, \qquad \boldsymbol{d} \in \mathbb{R}^{m}$$

- Slack variables

$$\boldsymbol{Dx} + \boldsymbol{s} = \boldsymbol{d}, \qquad \boldsymbol{s} > 0$$

- Penalty term

$$P(\boldsymbol{x}, \boldsymbol{s}) = \sum_{k=1}^{m} \lambda_k (\langle \boldsymbol{D}_k, \boldsymbol{x} \rangle + s_k - d_k)^2 , \qquad \lambda_k > 0$$

QUBO with linear inequality constraints

- Encode non-binary slack variable $s_k > 0$ through **binary encoding**

$$s_k = \langle \mathbf{2}, \boldsymbol{a}_k \rangle = 2^0 a_{k,1} + 2^1 a_{k,2} + \cdots + 2^{M-1} a_{k,M}, \qquad a_{k,i} \in \{0,1\}$$

- Penalty term

$$P(\boldsymbol{x}, \boldsymbol{a}) = \sum_{k=1}^{m} \lambda_k (\langle \boldsymbol{D}_k, \boldsymbol{x} \rangle - d_k + \langle \mathbf{2}, \boldsymbol{a}_k \rangle)^2 , \qquad \lambda_k > 0$$

First strategy for deriving QUBO matrices

Truth-table approach

1. Tabulate all the problem's possible states in a truth table
2. Penalize undesirable states with higher energies
3. Set QUBO coefficients that result in the selected energies for each state

- **Example**: enforce $x_2 = \neg x_1$

state	x_1	x_2	$x_2 = \neg x_1$	penalty
1	0	0	no	p
2	0	1	yes	p-1
3	1	0	yes	p-1
4	1	1	no	p

$$f(x_1, x_2) = -x_1 - x_2 + 2x_1x_2 = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} -1 & 2 \\ & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{aligned} a_1 0 + a_2 0 + b_{12} 00 &= p \\ a_1 0 + a_2 1 + b_{12} 01 &= p - 1 \\ a_1 1 + a_2 0 + b_{12} 10 &= p - 1 \\ a_1 1 + a_2 1 + b_{12} 11 &= p \end{aligned} \qquad \begin{aligned} 0 &= p \\ a_2 &= p - 1 \\ a_1 &= p - 1 \\ a_1 + a_2 + b_{12} &= p \end{aligned}$$

Elementary Boolean operations

constraint	penalty term
$z = \neg x_1$	$-x_1 - z + 2x_1 z$
$z = x_1 \vee x_2$	$x_1 x_2 + (x_1 + x_2)(1 - 2z) + z$
$z = x_1 \wedge x_2$	$x_1 x_2 - 2(x_1 + x_2)z + 3z$
$z = x_1 \oplus x_2$	$2x_1 x_2 - 2(x_1 + x_2)z + 4(x_1 + x_2)a$ $+4za + x_1 + x_2 + z + 4a$ with auxiliary variable a

Truth tables, Boolean operations and rules for their simplification can be found in the book by Harris and Harris: Digital Design and Computer Architecture.

Higher-degree polynomial reduction by substitution

- Let us assume the interaction of 3 binary variables

$$f(x_1, x_2, x_3) = x_1 x_2 x_3$$

- This can be transformed into a QUBO formulation by
  1. introducing an **auxiliary variable** $z = x_1 x_2$
  2. express constraint as

$$P(x_1, x_2; z) = x_1 x_2 - 2(x_1 + x_2)z + 3z$$

  3. replace triplet interaction by

$$x_1 x_2 x_3 = \min_z \{z x_3 + \lambda P(x_1, x_2; z)\}, \qquad \lambda > 1$$

Higher-degree polynomial reduction by substitution ($\lambda = 2$)

		$P(x_1, x_2; z)$		$zx_3 + \lambda P$		
x_1, x_2, x_3	$x_1 x_2 x_3$	$z = x_1 x_2$	$z \neq x_1 x_2$	$z = x_1 x_2$	$z \neq x_1 x_2$	$\min_z \{zx_3 + \lambda P\}$
0,0,0	0	0	3	0	6	0
0,0,1	0	0	3	0	7	0
0,1,0	0	0	1	0	2	0
0,1,1	0	0	1	0	3	0
1,0,0	0	0	1	0	2	0
1,0,1	0	0	1	0	3	0
1,1,0	0	0	1	0	2	0
1,1,1	1	0	1	1	2	1

Higher-degree polynomial reduction by substitution

- Let us assume the interaction of 4 binary variables

$$f(x_1, x_2, x_3, x_4) = x_1 x_2 x_3 x_4$$

- Introduce 2 auxiliary variables

$$z_1 = x_1 x_2, \qquad z_2 = x_3 x_4$$

- Replace original expression by

$$x_1 x_2 x_3 x_4 = \min_{z_1, z_2} \lambda\big(P_1(x_1, x_2; z_1) + P_2(x_3, x_4; z_2)\big)$$

Higher-degree polynomial reduction by minimum selection

- Substitution rules for $ax_1x_2x_3$
- if $a < 0$:

$$ax_1x_2x_3 = az(x_1 + x_2 + x_3 - 2)$$

- if $a > 0$:

$$ax_1x_2x_3 = a\{z(x_1 + x_2 + x_3 - 1) + (x_1x_2 + x_1x_3 + x_2x_3) - (x_1 + x_2 + x_3) + 1\}$$

- Consider the case $a < 0$:
  - $(x_1 + x_2 + x_3 - 2) > 0 \Leftrightarrow x_1 = x_2 = x_3 = 1$, $z = 1$ preserves negative value
  - $(x_1 + x_2 + x_3 - 2) \leq 0$ otherwise, $z = 0$ preserves negative value

Higher-degree polynomial reduction

- Collection of precipice

  Nike Dattani: *Quadratization in Discrete Optimization and Quantum Mechanics*, arXiv:1901.04405, 2019

- Common practices

  https://docs.dwavesys.com/docs/latest/handbook_reformulating.html

Summary

- AQC is ready to use today(!) and technology matures with every year
- AQC is suited for solving combinatorial optimization problems
- Main steps to solve a problem with quantum annealing
  1. Derive QUBO/Ising formulation of the problem at hand
  2. Find minor embedding of the problem graph onto the hardware graph
  3. Interpret/post-process bitstring outcome of the quantum annealer
  4. Scale problem size by hybridization to quantum-classical computation

Famous last words

- Quantum annealing is exciting and one is tempted to apply it to all types of combinatorial optimization problems just because its possible and easy

- Think twice before you use a quantum annealer! Not every problem is NP-hard and not every NP-hard problem is really hard to solve. Often classical meta-heuristic algorithms give sufficiently good approximations in practice.

- Spin-glass problems – combinatorial optimization problems with multi-modal objective functions – are commonly believed to be candidates where QA outperforms classical meta-heuristic algorithms by far.

Have fun in the coding session!