A Practical Approach to Quantum Annealing

GOTO CHICAGO 2020

TRIFORK.
...think software

AGENDA

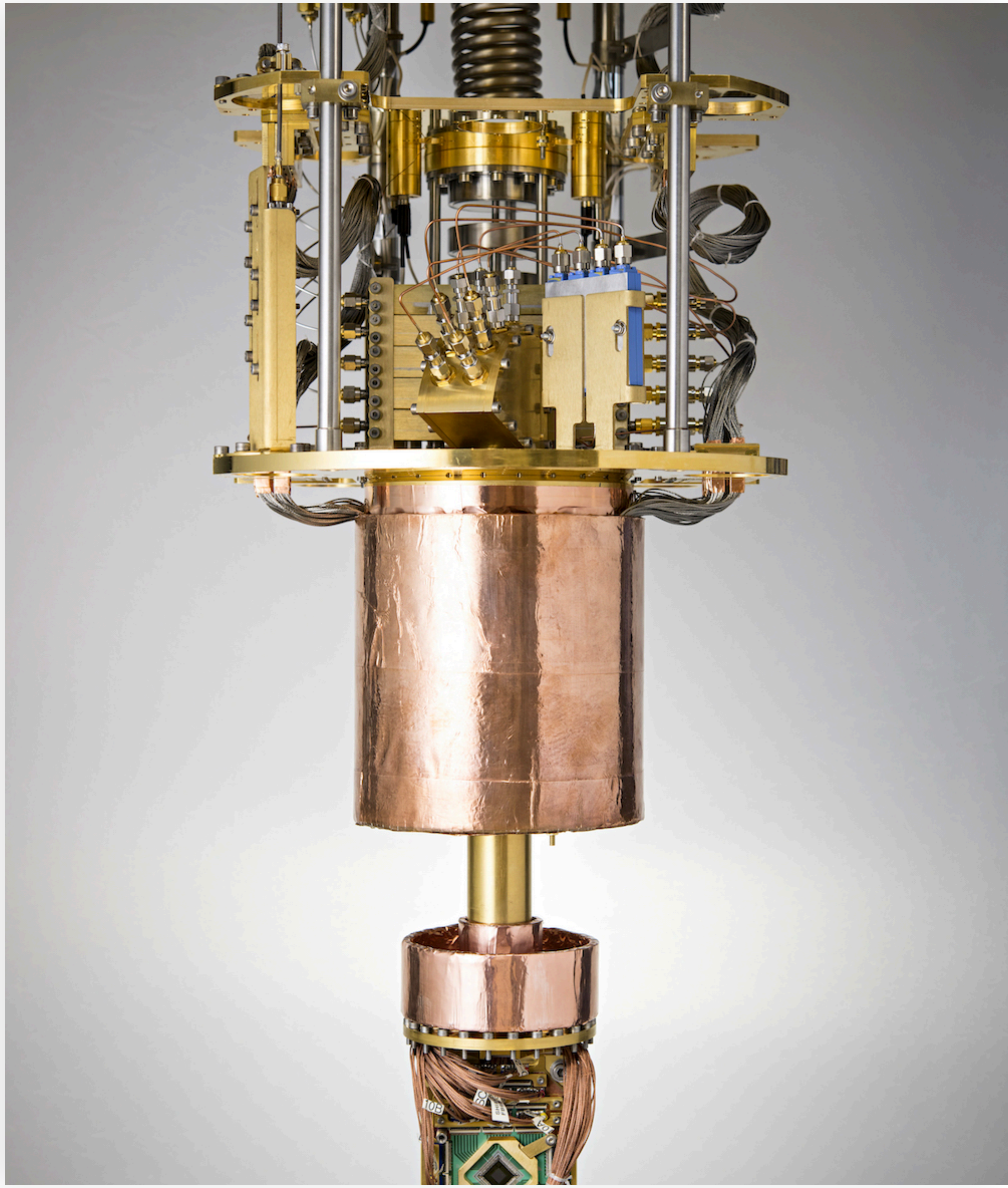
Practical Quantum Annealing - Part 1

- Quantum Annealing - Hardware and Basic Principles
- Introduction to Programming Model and API
- Constraint Satisfation Problems

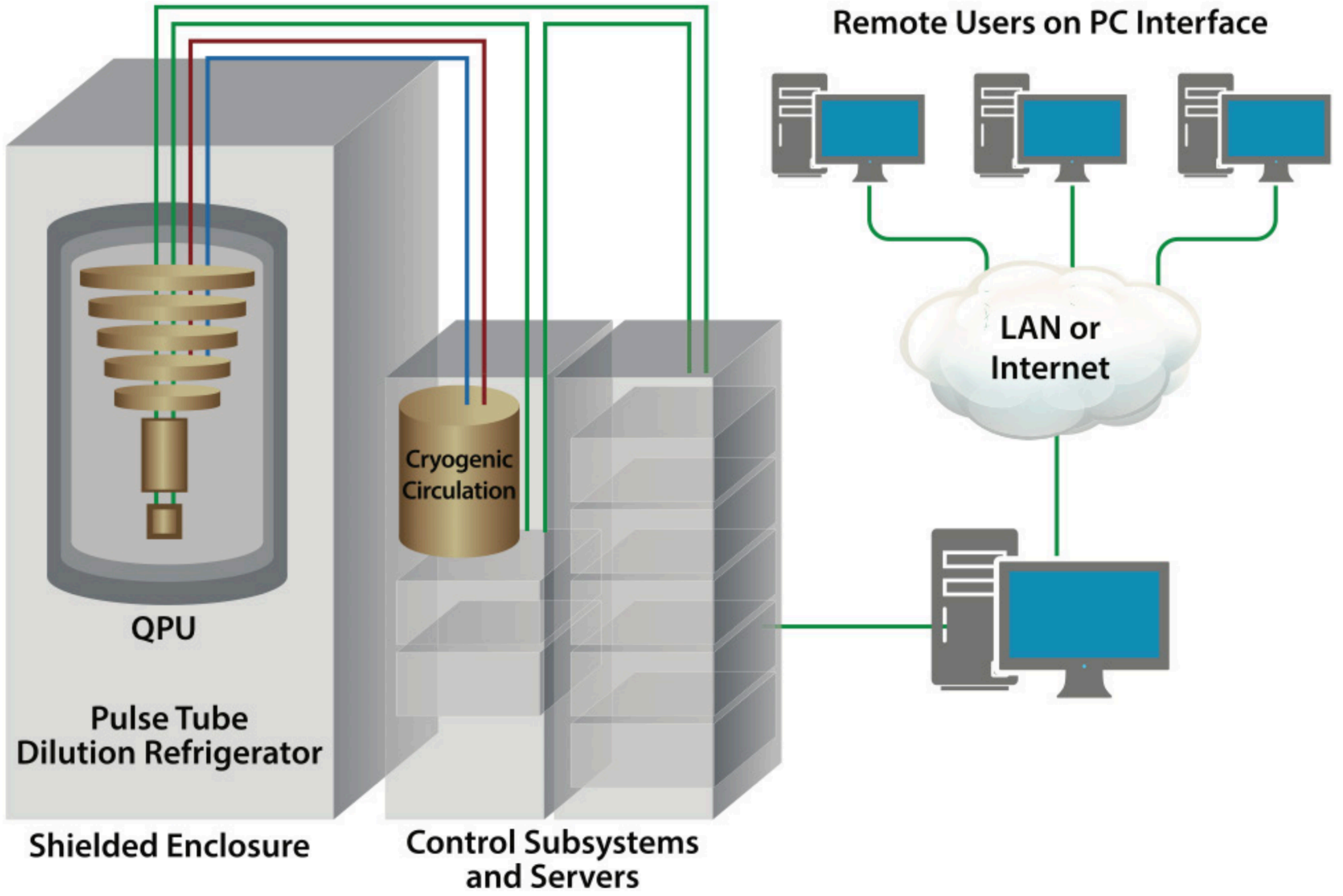
HARDWARE AND BASIC PRINCIPLES

Quantum Annealing

- Quantum annealing is a metaheuristic for finding the global minimum of a given objective function
- Finds a finite set of possible solutions using quantum fluctuation based computation
- The annealing process starts in a superposition of all possible states
- Alternative to gate-based quantum computing
  - Can be applied to a wide range of problems

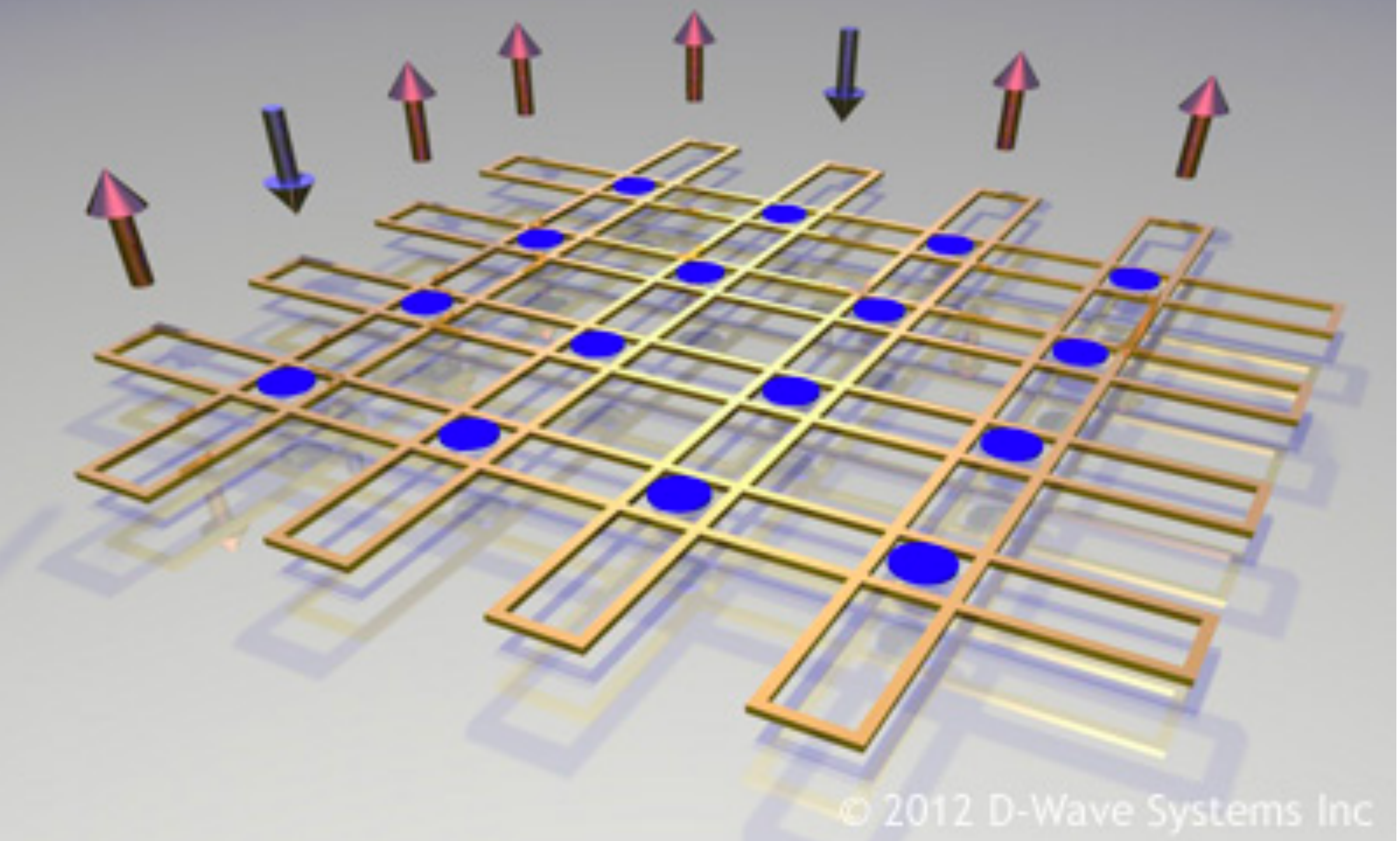



D:WAVE
2000Q
D:WAVE

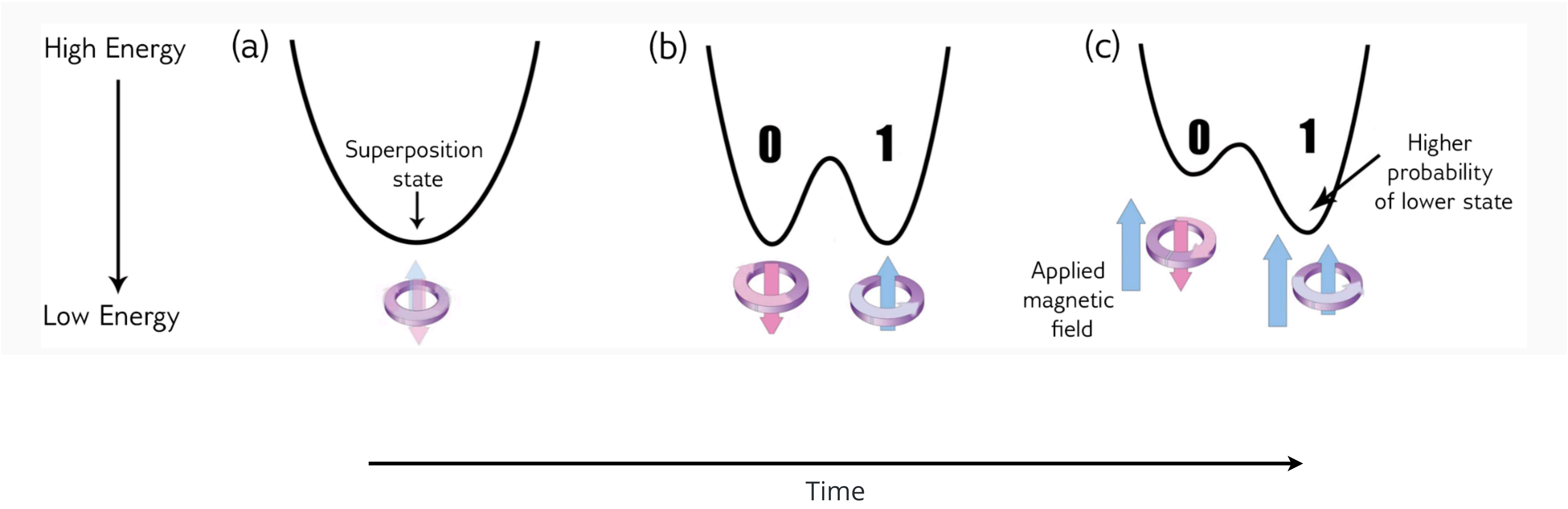


TRIFORK.
...think software

Qubit loops coupled together



Adiabatic quantum computing

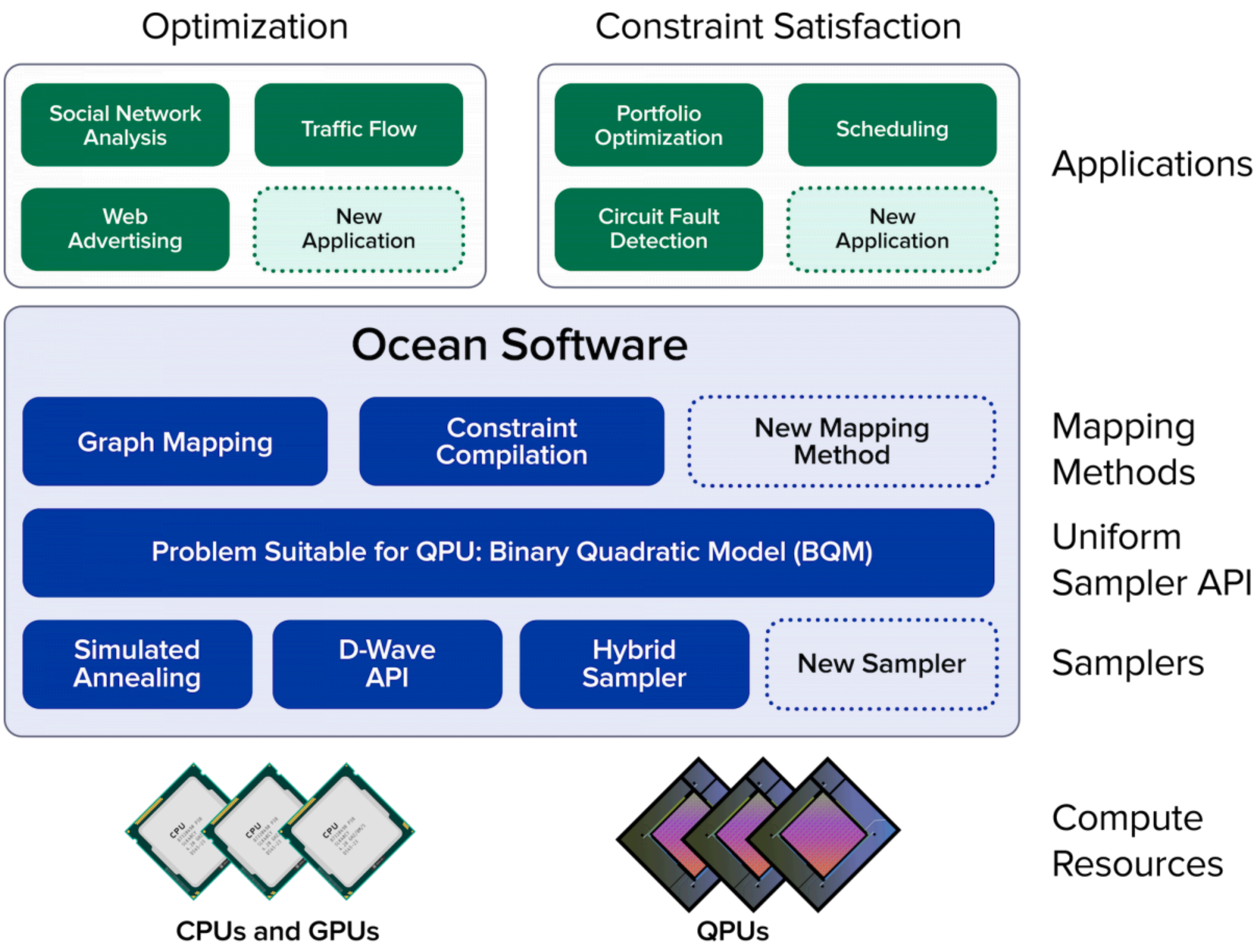


Source: https://docs.dwavesys.com/docs/latest/c_gs_2.html#

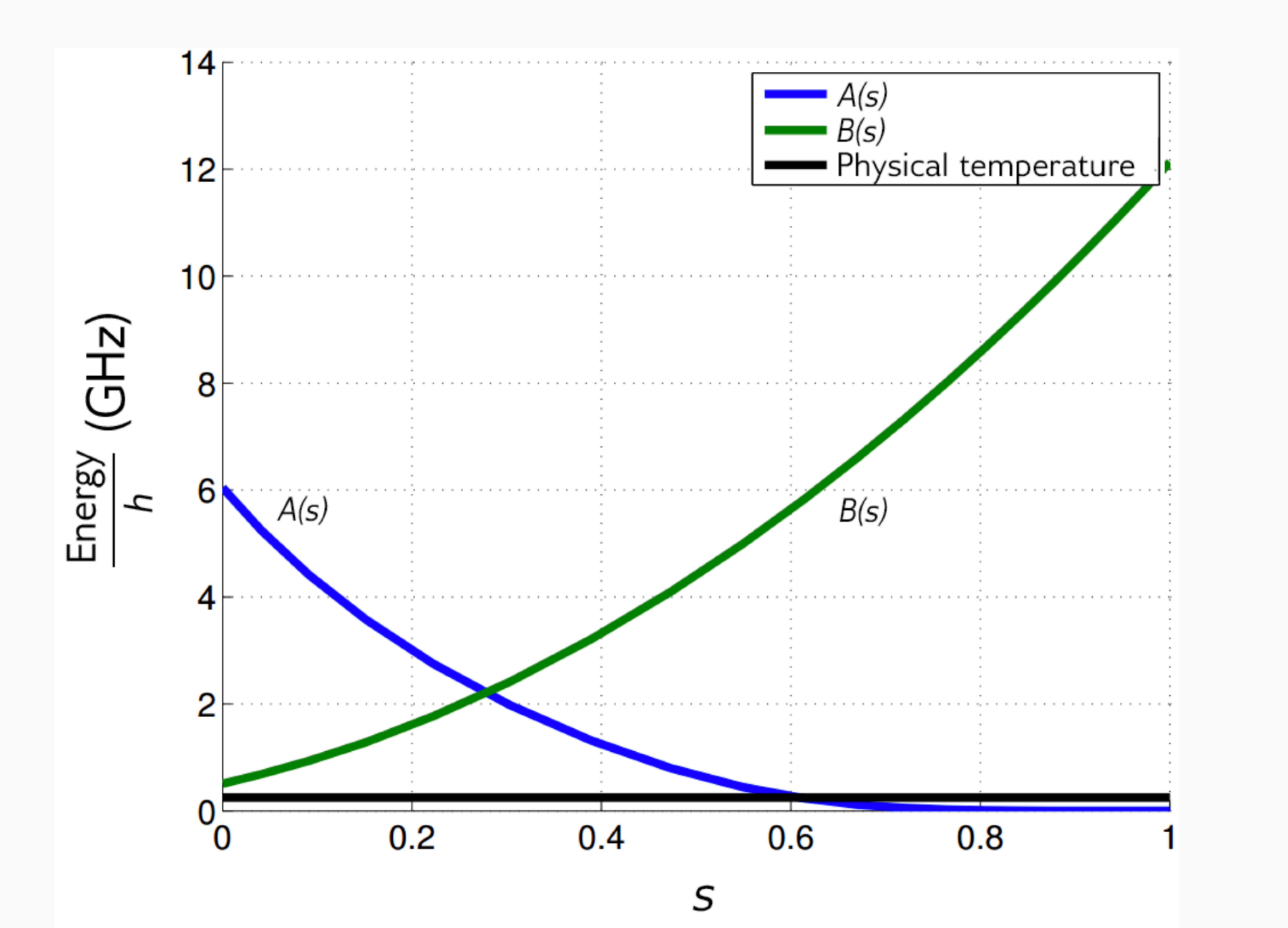
Objective function for optimisation - bias

Source: https://www.dwavesys.com/quantum-computing

Practical Quantum Annealing - Part 1

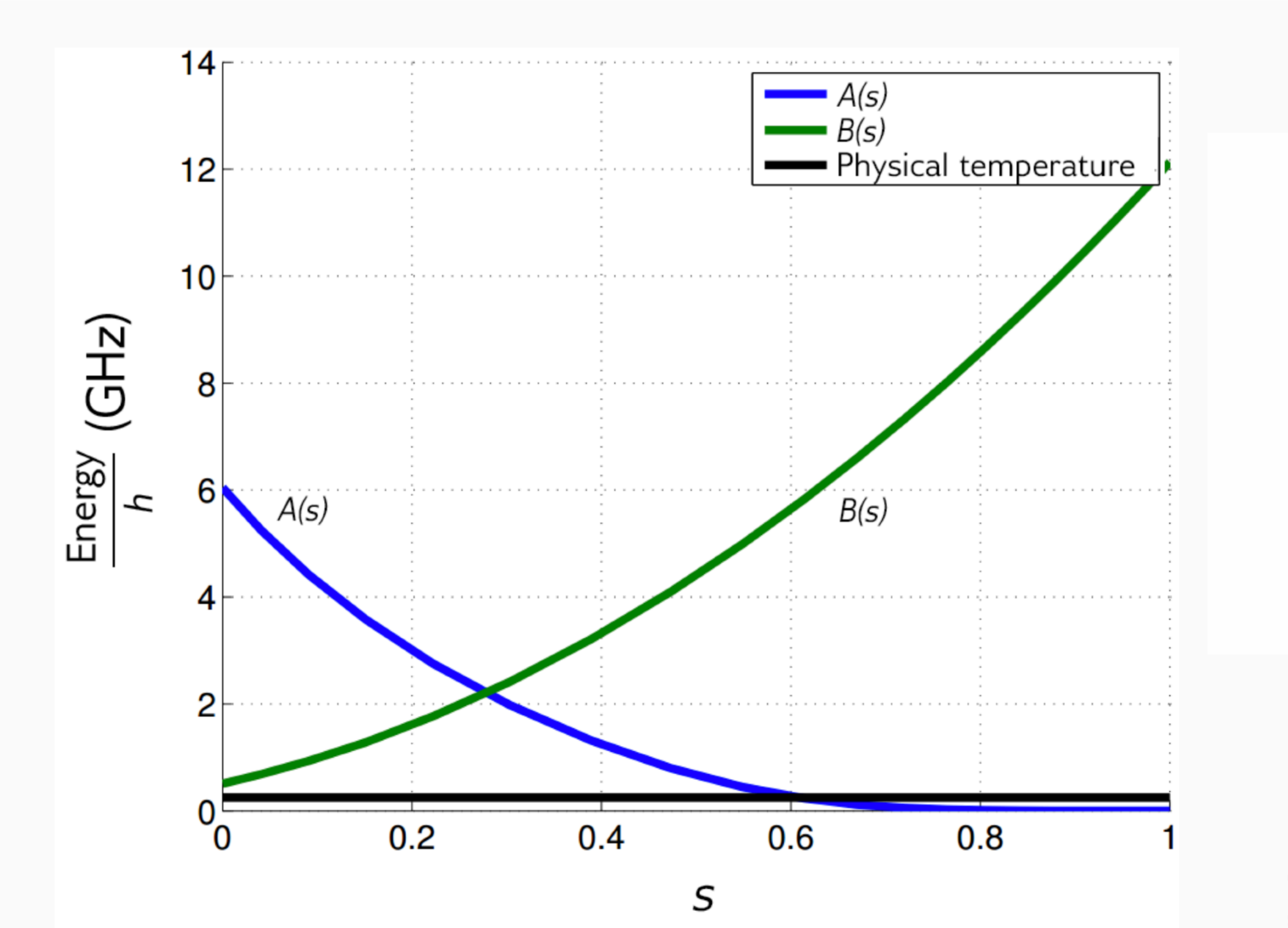

Optimization
Constraint Satisfaction
Social Network Analysis
Traffic Flow
Web Advertising
New Application
Portfolio Optimization
Scheduling
Circuit Fault Detection
New Application
Applications
Ocean Software
Graph Mapping
Constraint Compilation
New Mapping Method
Mapping Methods
Problem Suitable for QPU: Binary Quadratic Model (BQM)
Uniform Sampler API
Simulated Annealing
D-Wave API
Hybrid Sampler
New Sampler
Samplers
CPUs and GPUs
QPUs
Compute Resources

Adiabatic quantum computing



$$\mathcal{H}_{ising} = \underbrace{-\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right)}_{\text{Final Hamiltonian}}$$

Adiabatic quantum computing



Objective function for optimisation - bias

$$\mathbf{E}_{ising}(s) = \sum_{i=1}^{N} h_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{i,j} s_i s_j$$

OR

$$\mathbf{E}_{qubo}(a_i, b_{i,j}; q_i) = \sum_{i} a_i q_i + \sum_{i<j} b_{i,j} q_i q_j \, .$$

BINARY QUADRATIC MODEL

Binary Quadratic Model (BQM)

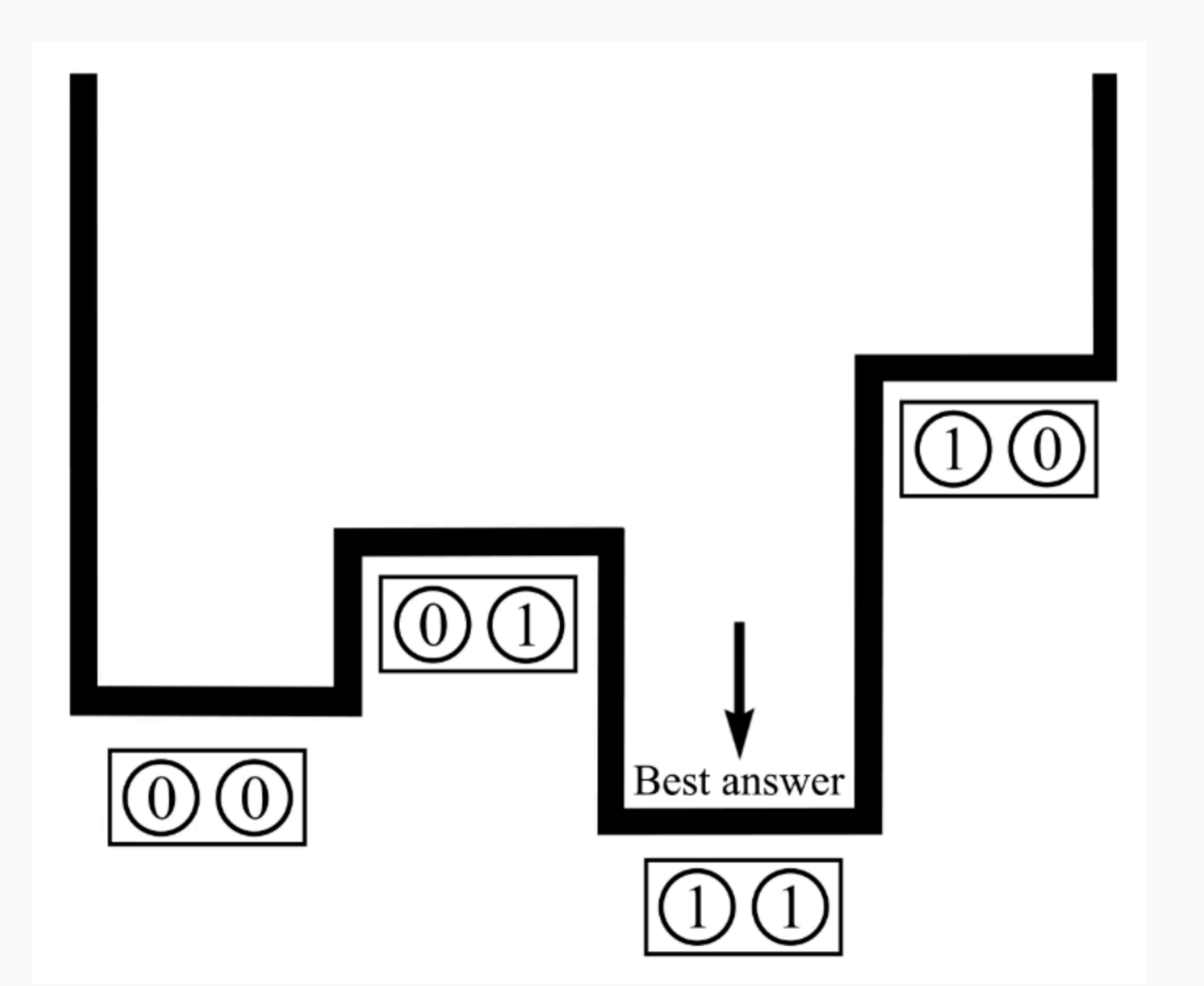
```
import dimod

bqm = dimod.BinaryQuadraticModel(
    {0: 1, 1: -1, 2: .5},                # Linear
    {(0, 1): .5, (1, 2): 1.5},           # Quadratic
    1.4,                                 # Offset
    dimod.Vartype.SPIN)                  # SPIN/BINARY
```

Embedding and sampling

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

bqm = …

sampler = EmbeddingComposite(DWaveSampler())

response = sampler.sample(bqm, num_reads=500)
```

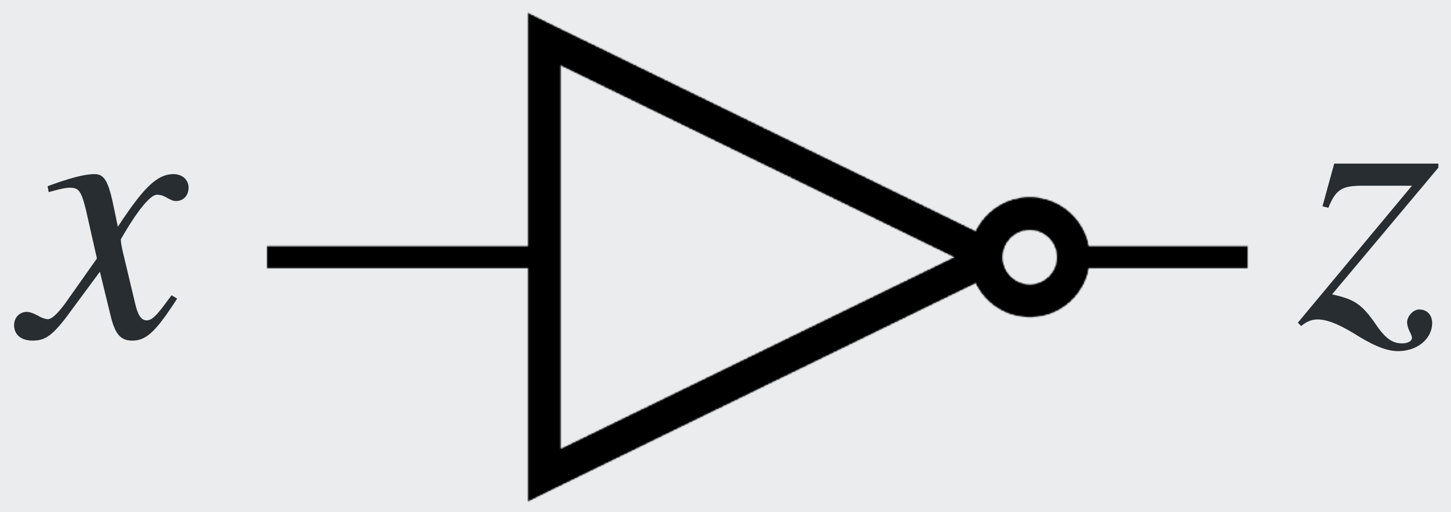
Objective functions

- Formulate an objective function
  - Punish wrong solutions (high energy)
  - Reward correct solutions (low energy)
- Express function in terms of QUBO or Ising model
- Embed and sample



PROGRAMMING MODEL

Example: Classical NOT-gate



$$\neg x = z$$

x	z	Valid?
0	1	Yes
1	0	Yes
0	0	No
1	1	No

TRUTH TABLE

PROGRAMMING MODEL

Example: Classical NOT-gate

PENALTY FUNCTION

$$2xz - x - z + 1$$

$$2 \times 0 \times 1 - 0 - 1 + 1 = 0$$
$$2 \times 1 \times 0 - 1 - 0 + 1 = 0$$
$$2 \times 0 \times 0 - 0 - 0 + 1 = 1$$
$$2 \times 1 \times 1 - 1 - 1 + 1 = 1$$

x	z	Valid?	Penalty
0	1	Yes	0
1	0	Yes	0
0	0	No	1
1	1	No	1

TRUTH TABLE

PROGRAMMING MODEL

Example: Classical NOT-gate - QUBO

PENALTY FUNCTION

$$2\mathrm{xz} - x - z + 1$$

GENERIC QUBO

$$q_{11}x_1 + q_{22}x_2 + q_{12}x_1x_2$$

PENALTY FUNCTION AS QUBO

$$-x_1 - x_2 + 2x_1x_2$$

GENERIC MATRIX

$$\begin{bmatrix} q_{11} & q_{12} \\ 0 & q_{22} \end{bmatrix}$$

PENALTY FUNCTION

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

TRIFORK.
...think software

Example: Classical NOT-gate

```
Q = {('x', 'x'): -1, ('x', 'z'): 2, ('z', 'x'): 0, ('z', 'z'): -1}
response = sampler.sample_qubo(Q, num_reads=5000)
for d in response.data(['sample', 'energy', ‘num_occurrences']):
  print(d.sample, "Energy: ", d.energy, "Occurrences: “, d.num_occurrences)
```

OUTPUT

```
{'x': 0, 'z': 1} Energy:  -1.0 Occurrences:  2062
{'x': 1, 'z': 0} Energy:  -1.0 Occurrences:  2937
{'x': 1, 'z': 1} Energy:  0.0 Occurrences:  1
```

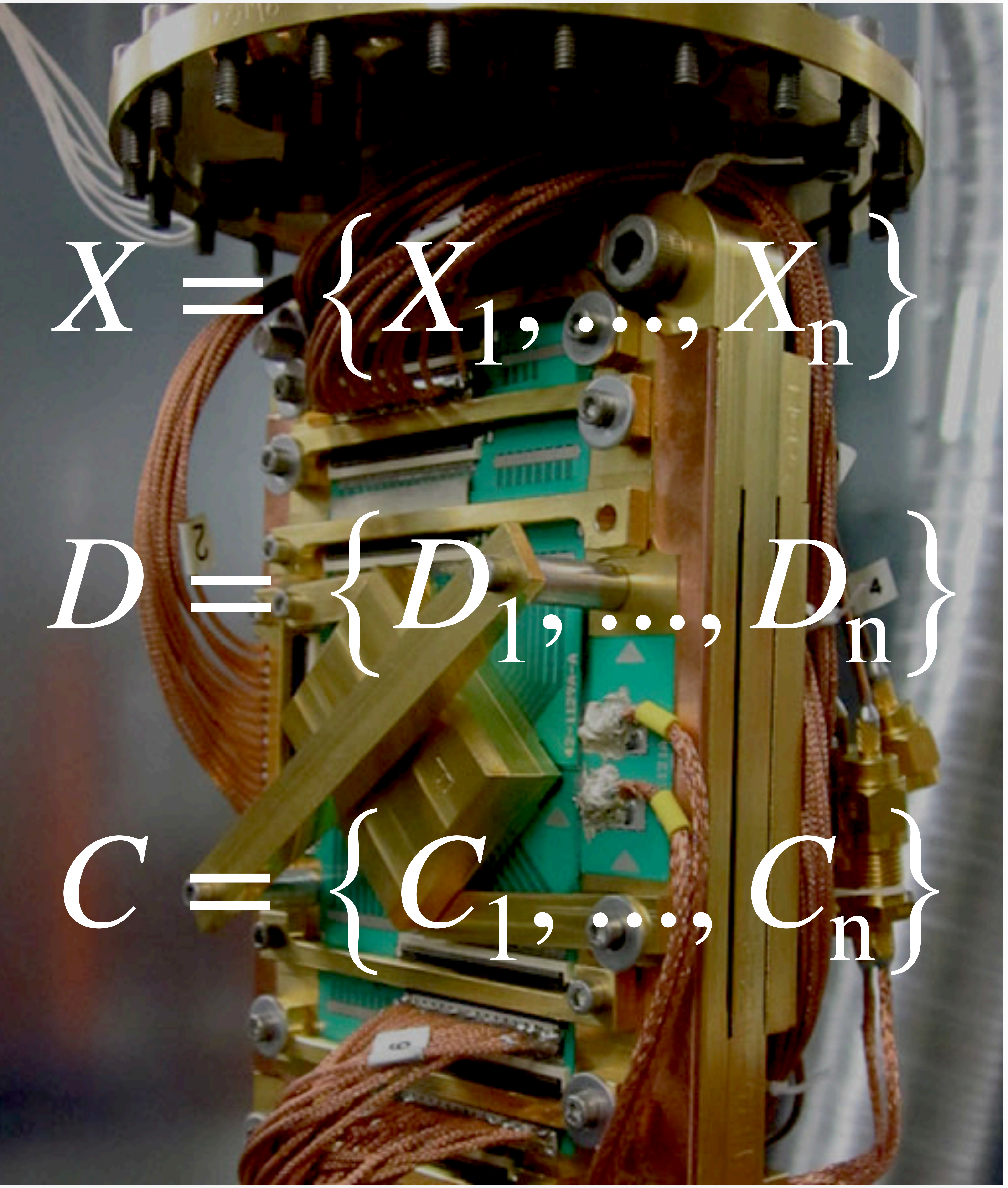
AGENDA

Practical Quantum Annealing - Part 1

Quantum Annealing - Hardware and Basic Principles

Introduction to Programming Model and API

Constraint Satisfation Problems



CONSTRAINT SATISFACTION PROBLEMS

Binary CSP

- Problems formulated as a set of binary constraints
- A constraint is defined as a set of variables and
  - A set of valid configurations or
  - A function returning true or false based on
- Constraints are stitched together to form one BQM which can be solved by by the Quantum Processor

BINARY CSP

dwavebinarycsp

CONSTRAINT 1 $a = b$

CONSTRAINT 2 $b \neq c$

```
import dwavebinarycsp
import operator

csp=dwavebinarycsp.ConstraintSatisfactionProblem(‘BINARY')
csp.add_constraint(operator.eq, ['a', 'b'])
csp.add_constraint(operator.ne, ['b', 'c'])
result = csp.check({'a': 1, 'b': 1, 'c': 0}) # True
```

BINARY CAP

Converting constraints to model

```
..
csp.add_constraint(operator.eq, ['a', ‘b'])
csp.add_constraint(operator.ne, ['b', 'c'])

bqm = dwavebinarycsp.stitch(csp) # model

print(bqm)
BinaryQuadraticModel({a: 1.9999999998686426,
b: -1.7323851242423416e-09, c:
-2.0000000004188005}, {('a', 'b'):
-3.9999999991051225, ('b', 'c'):
3.9999999999105169}, 2.0000000001284974,
'BINARY')
```

$$\begin{bmatrix} 2 & -4 & 0 \\ 0 & -1.73 & 4 \\ 0 & 0 & -2 \end{bmatrix}$$

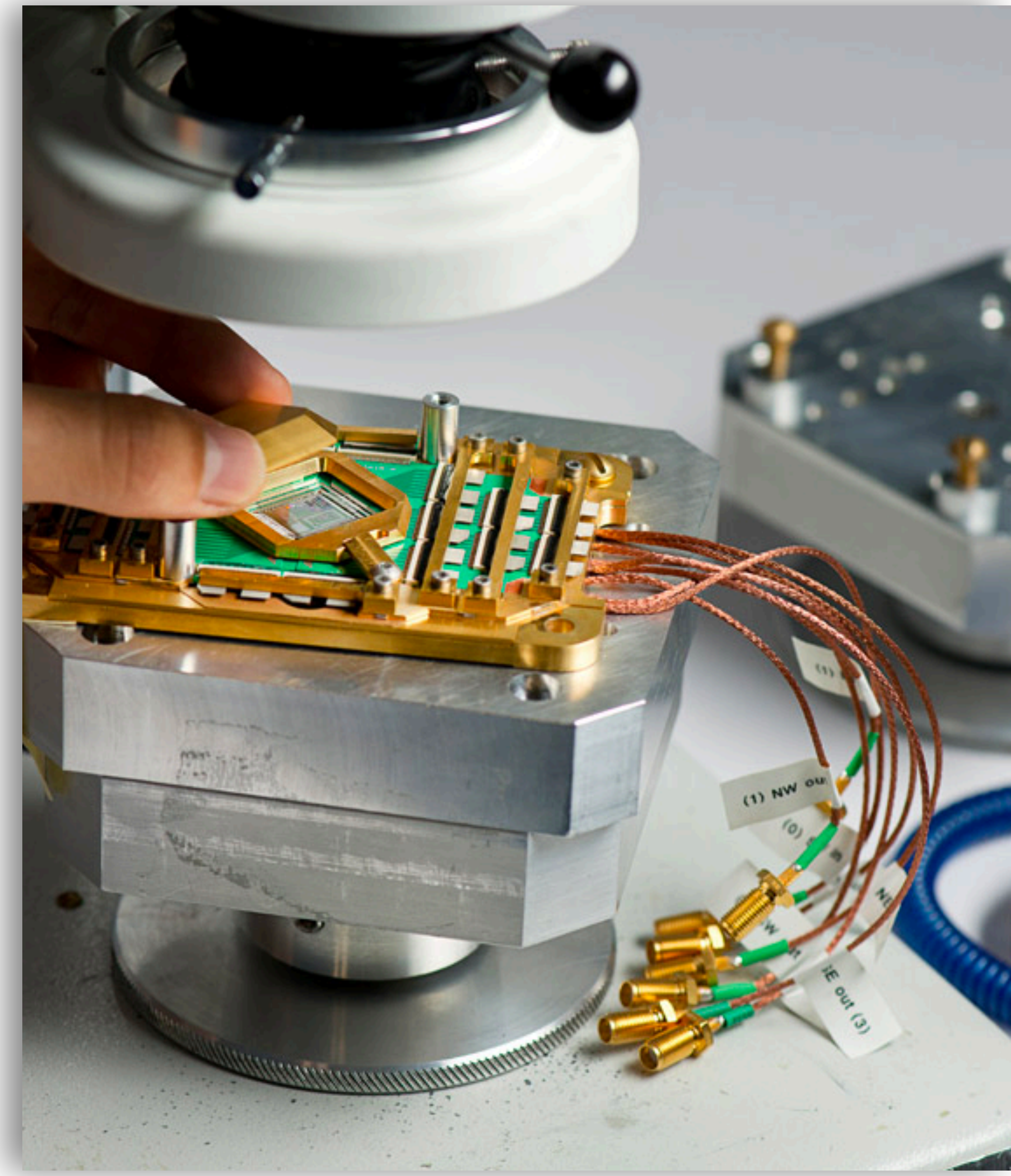
QUBO REPRESENTATION

TRIFORK.
...think software

NEXT STEPS

Resources

- GOTO Videos
  - “Fueling the Quantum Application Era with the Cloud - Murray Thom”
    - https://youtu.be/nn5xTQVoxbY
  - “Quantum Computing - Jessica Pointing”
    - https://youtu.be/d2pGGNQ63GQ
- Take the Leap !
  - https://www.dwavesys.com/take-leap
- GOTO Masterclasses
  - https://gotocph.com/2020/pages/offseasonmasterclasses



COMING NEXT

Practical Quantum Annealing - Part 2

Constraint Satisfation Problems Revisited

Networks and Network Algorithms

Divide and Conquer - An Introduction

Thank you

NIELS STHEN HANSEN
NSH@TRIFORK.COM

TRIFORK.
...think software