Building quantum applications with D-Wave's Leap

Alexander Condello

D-Wave
The Quantum Computing Company™

Before we start....

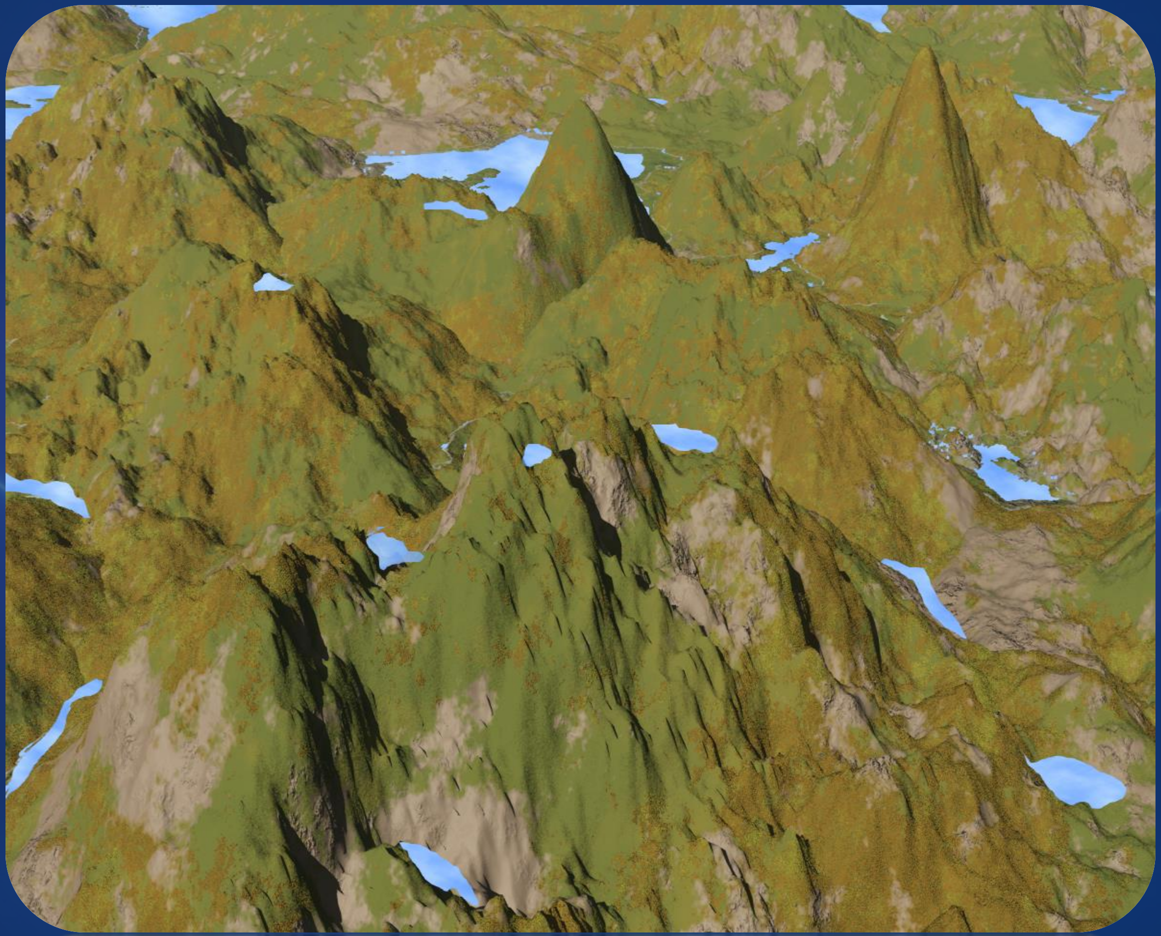
- (Almost) everything I will talk about is in open source
- Documentation is online and in LEAP
- We want feedback!
  - Community
  - Issues
  - Pull requests
- Your feedback helps us prioritize

Demo

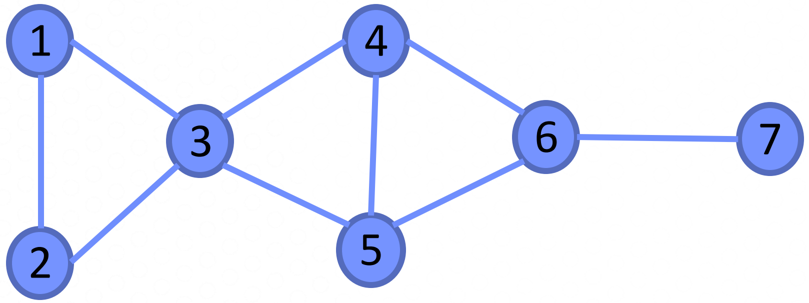
Solving Problems with BQMs

Landscape metaphor

Space of solutions defines an energy landscape and the best solution is the lowest valley

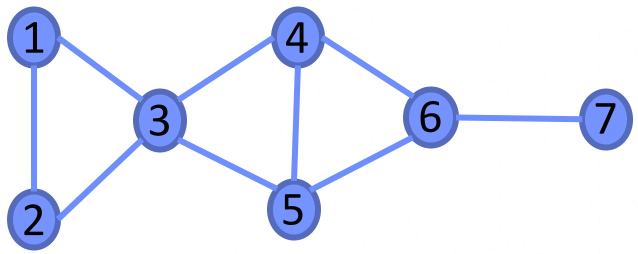


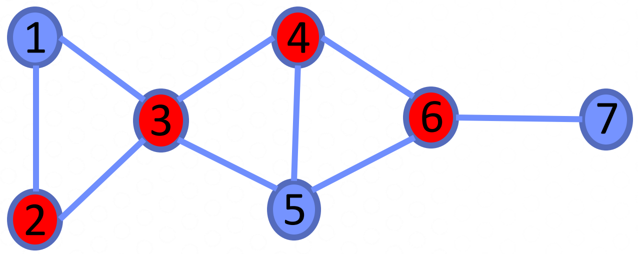
Example



- Given:
  - Network of pipelines
- What do we want:
  - A (minimum) set of junctions from which we can monitor every pipeline segment

Example





Junctions for monitoring pipes

Goal:

Nodes that *cover* every edge

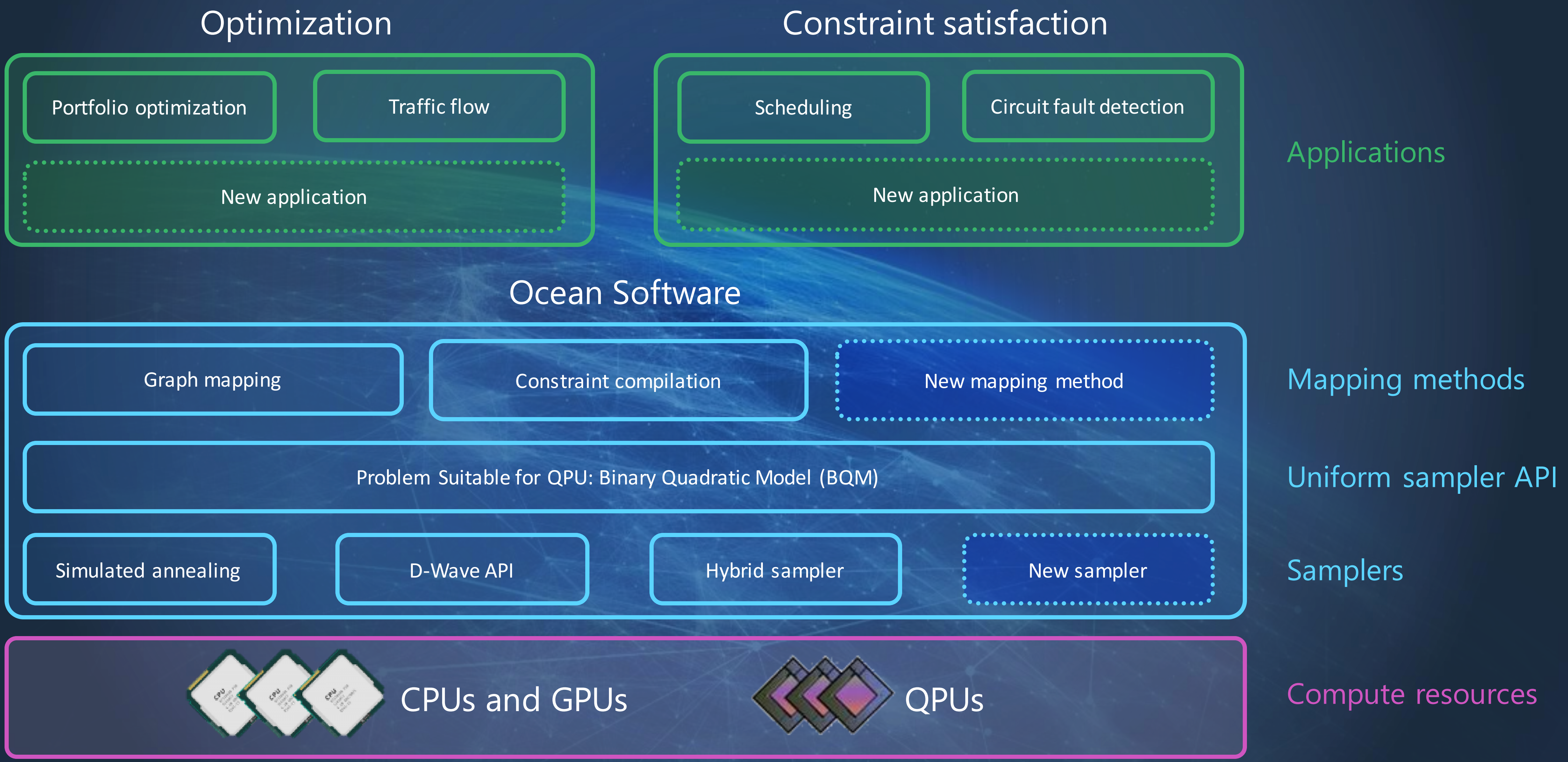
A vertex cover.

With the Ocean tools...

```python
import networkx as nx

import dwave_networkx as dnx

from dwave.system import DWaveSampler, EmbeddingComposite


sampler = EmbeddingComposite(DWaveSampler())


G = nx.Graph()

G.add_edges_from([(1,2),(1,3),(2,3),(3,4),(3,5),(4,5),(4,6),(5,6),(6,7)])


cover = dnx.min_vertex_cover(G, sampler=sampler)
```

Ocean™ software development kit

Suite of open-source Python tools on the D-Wave GitHub repository



With the Ocean tools…

```
import networkx as nx

import dwave_networkx as dnx

from dwave.system import DWaveSampler, EmbeddingComposite


sampler = EmbeddingComposite(DWaveSampler())


G = nx.Graph()

G.add_edges_from([(1,2),(1,3),(2,3),(3,4),(3,5),(4,5),(4,6),(5,6),(6,7)])


cover = dnx.min_vertex_cover(G, sampler=sampler)
```

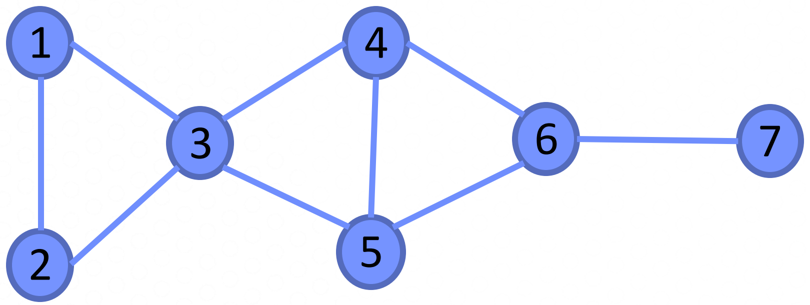
Binary Quadratic Model

$$E(v) = \sum_{i,j} v_i v_j a_{i,j} + \sum_i v_i b_i + c$$

$$a_{i,j}, b_i, c \in \mathbb{R}$$

$$v_i \in \{-1, +1\} \text{ or } v_i \in \{0, 1\}$$

Binary Quadratic Model and pipelines



$$E(v) = \alpha \sum_{v \in V} v + \sum_{u,v \in E} (1-u)(1-v)$$

Binary Variables

- Each junction either has a sensor or no

Pairwise interactions

- Every edge needs a sensor

Linear optimization

- Minimum number of sensors

Materials Properties

- Atomic magnetometer
- Solid state materials simulation
- Quantum molecular dynamics
- Quantum chemistry computation

Optimization

- Radiotherapy
- Multi-period portfolios
- Satellite placement
- Traffic flow
- Internet ad placement

200+ EARLY APPLICATIONS

- Finding Higgs Boson
- Image recognition
- Tree cover classifier
- DNA binding
- Individual cancer drugs

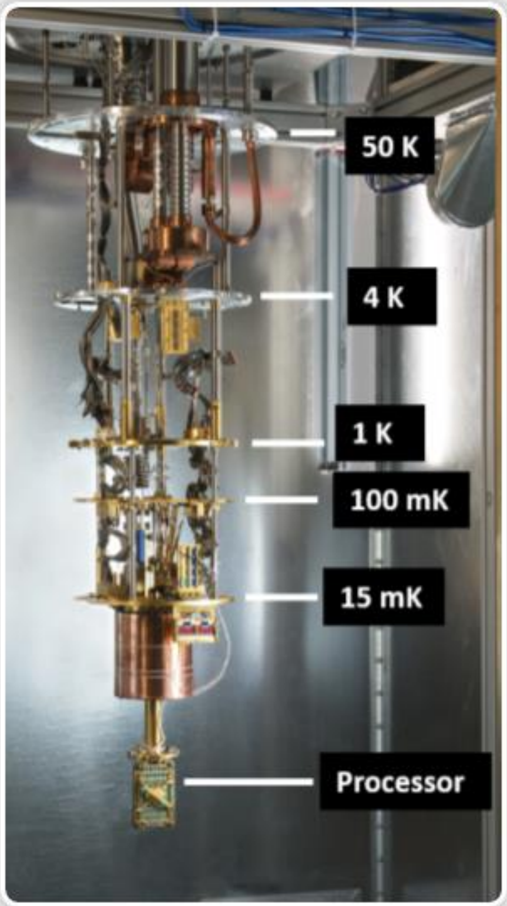
Machine Learning

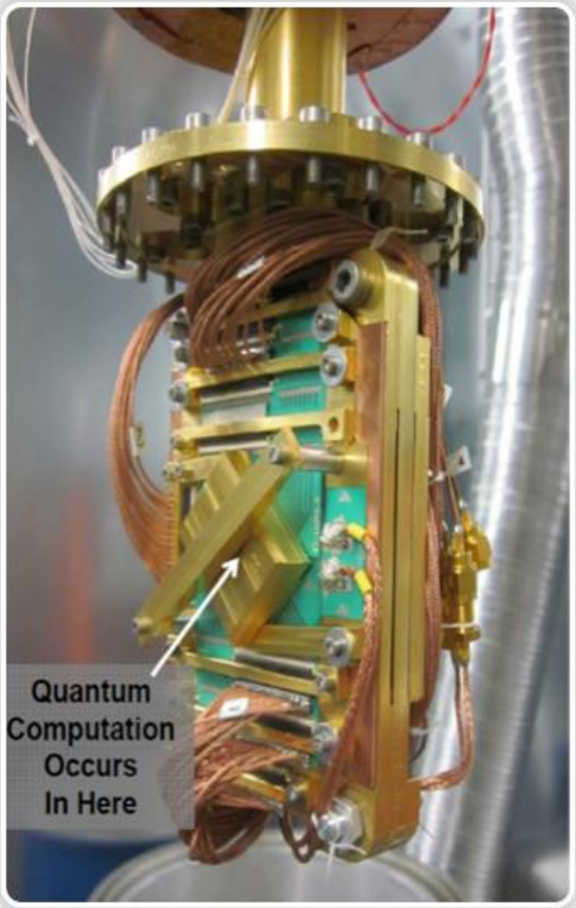
- Formation of Terrorist Networks
- Fault detection in circuits
- Facial recognition

Cyber Security & Fault Detection

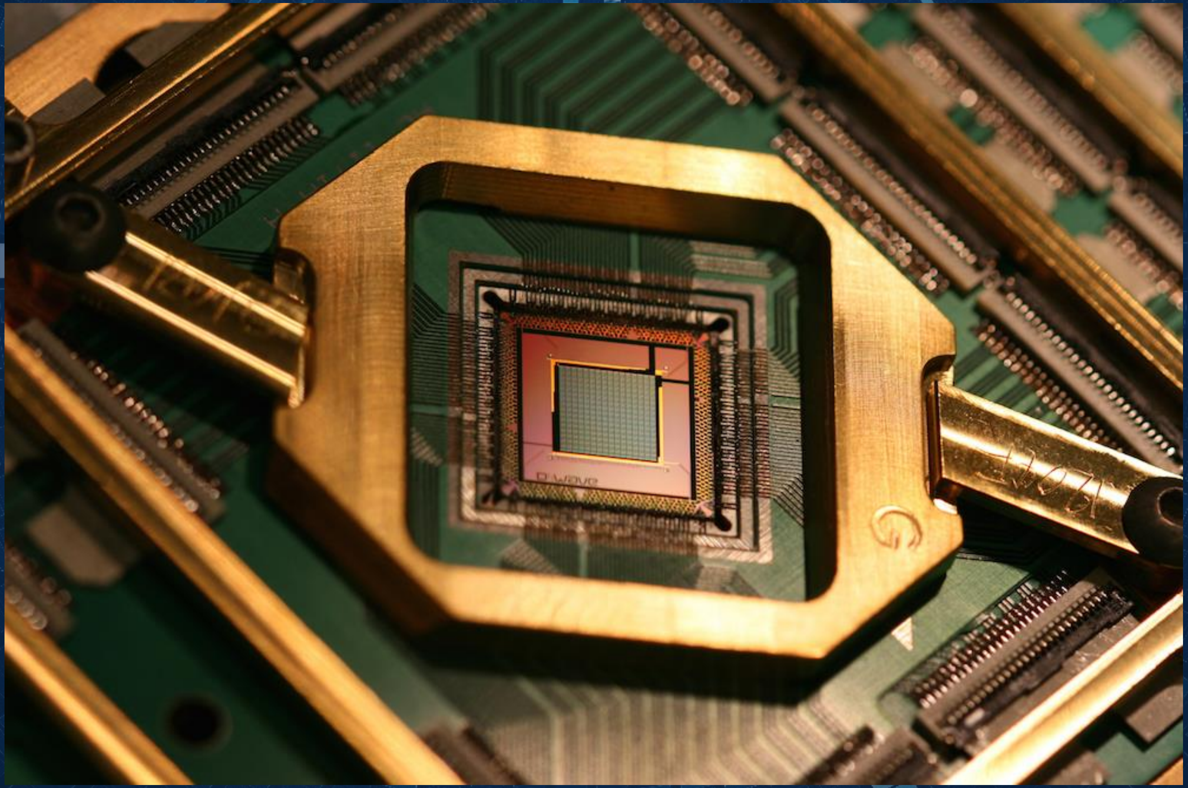
BQM to QMI

The D-Wave System





Quantum Processing Unit (QPU)



Ising Hamiltonian

$$H_{\text{ISING}} = -\frac{A(s)}{2}\Big(\sum_i \sigma_x^{(i)}\Big) + \frac{B(s)}{2}\boxed{\Big(\sum_i h_i \sigma_z^{(i)} + \sum_{i>j} J_{i,j}\sigma_z^{(i)}\sigma_z^{(j)}\Big)}$$

Quantum Machine Instruction (QMI)

Binary Quadratic Model

$$E(v) = \sum_{i,j} v_i v_j a_{i,j} + \sum_i v_i b_i + c$$

$$a_{i,j}, b_i, c \in \mathbb{R}$$

$$v_i \in \{-1, +1\} \text{ or } v_i \in \{0, 1\}$$

Quantum Machine Instruction (QMI)

$$E(v) = \sum_{i,j} v_i v_j a_{i,j} + \sum_i v_i b_i + c$$

+ Rules

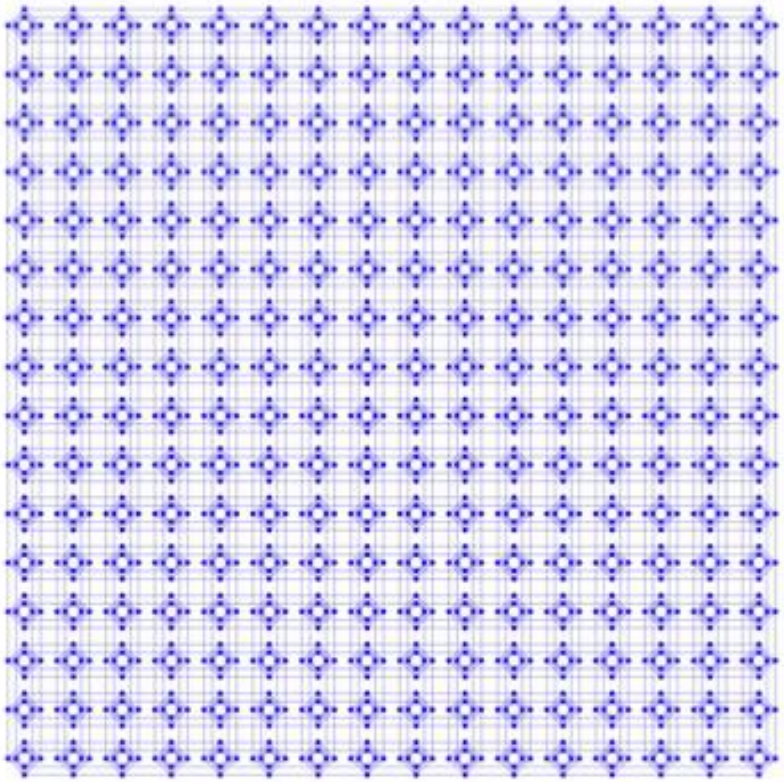
$$v_i \in \{-1, 1\}$$

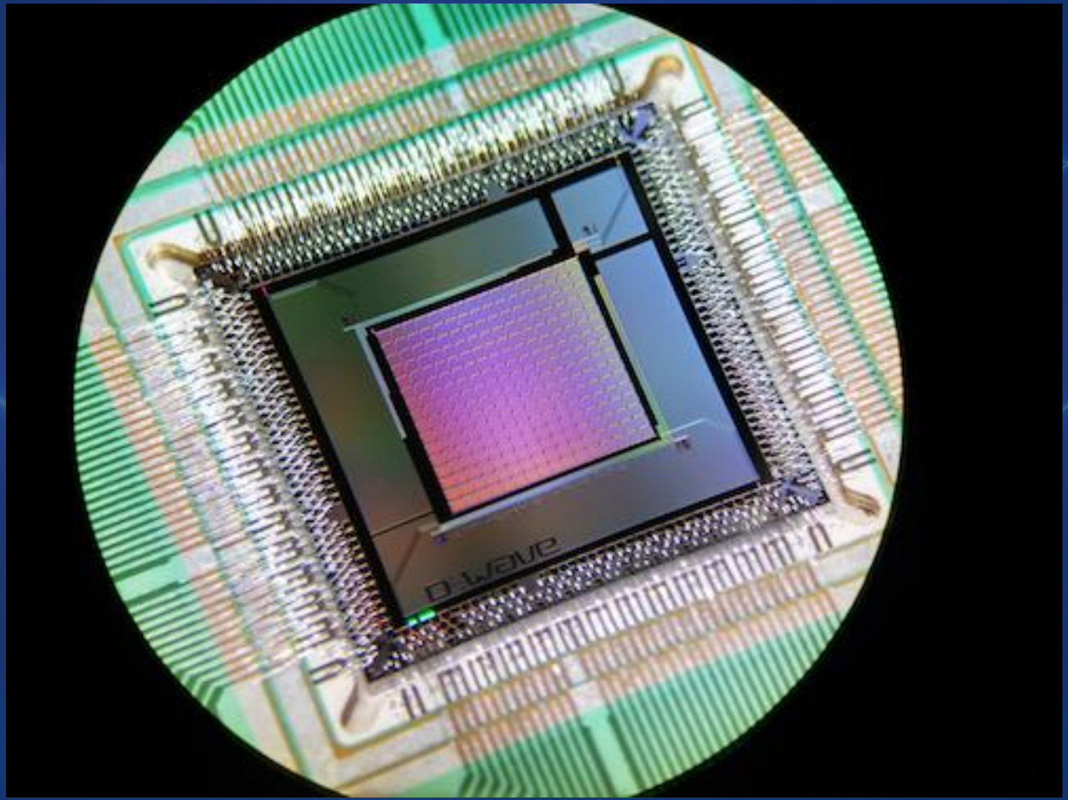
$$a_{i,j} \in [-2, 1], b_i \in [-1, 1], c \in \mathbb{R}$$

Hardware structured

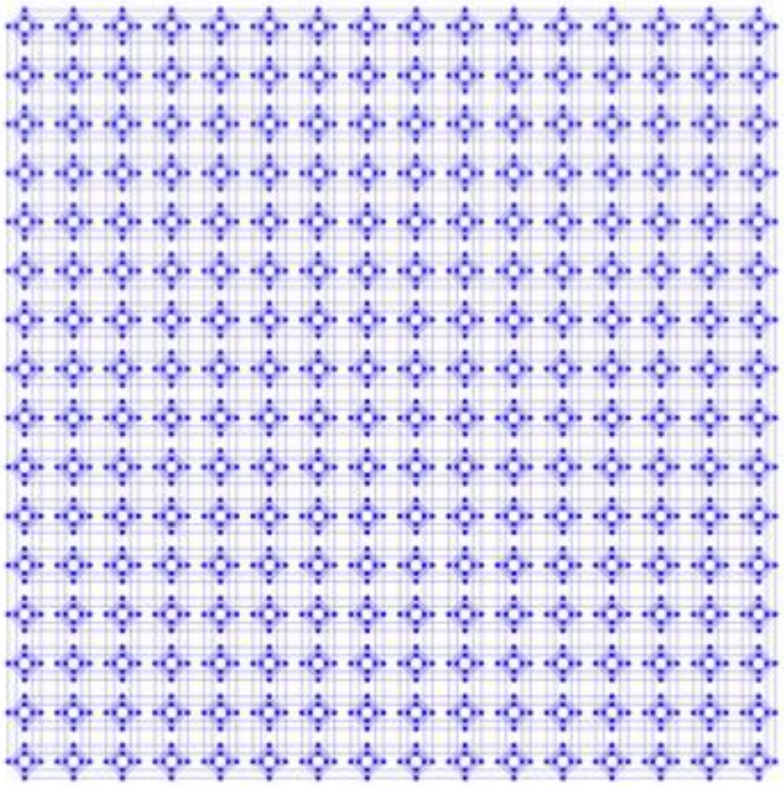
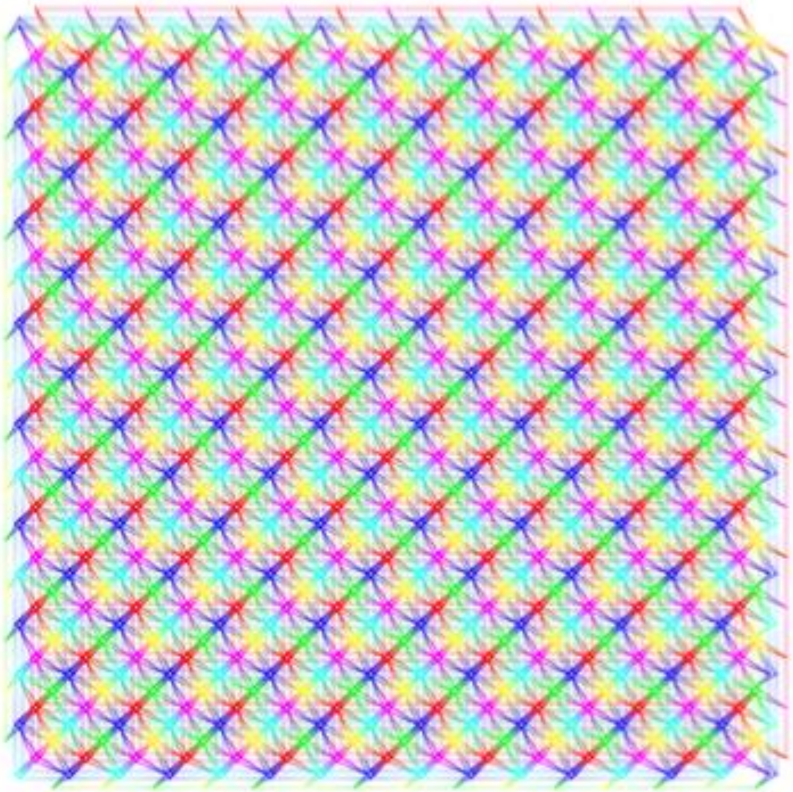
Qubit Layout

2000Q

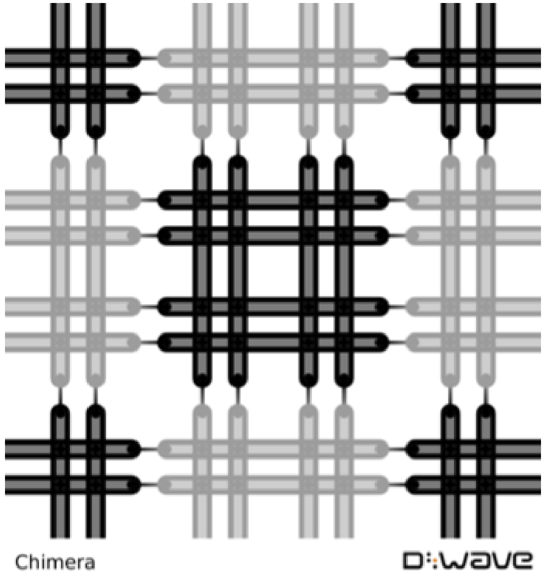
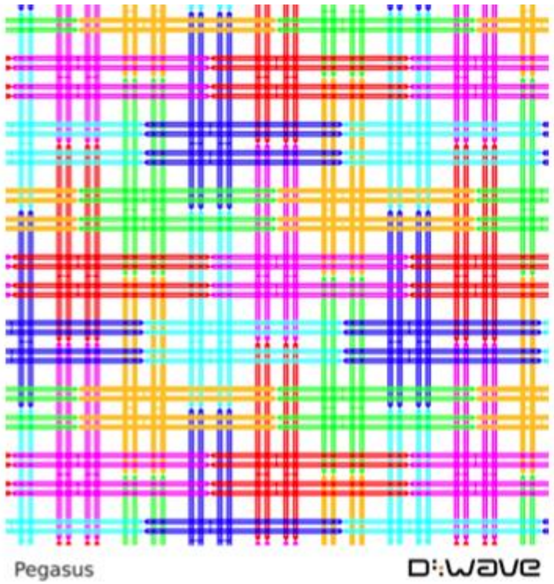




Working Graphs

2000Q		Advantage
		
2000	Qubits	5000
6000	Couplers	40000

A Closer Look...

2000Q		Advantage
		
6	Average Degree	15

Hybrid Algorithm Development

D:WAVE
The Quantum Computing Company™

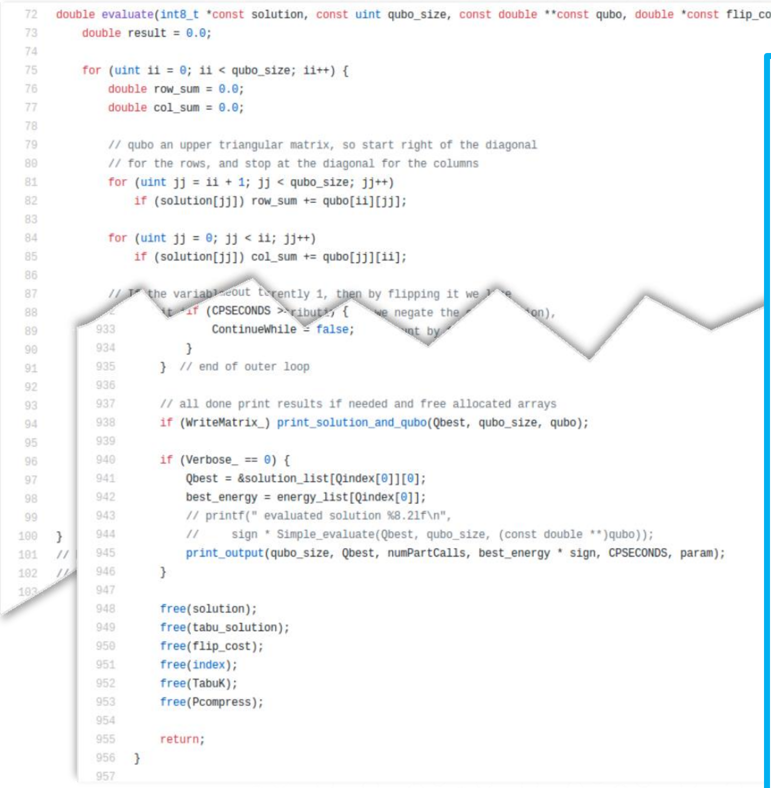
dwave-hybrid

- Hybrid Asynchronous Decomposition Sampler framework
  - Minimal, Python, solver/sampler-building framework, built atop Ocean tools
  - Leverages **quantum** and **classical** resources
  - Independent parts are executed **concurrently**
  - Problems are **broken into pieces** that fit the compute resources
  - Uses sample sets (probabilistic approach)

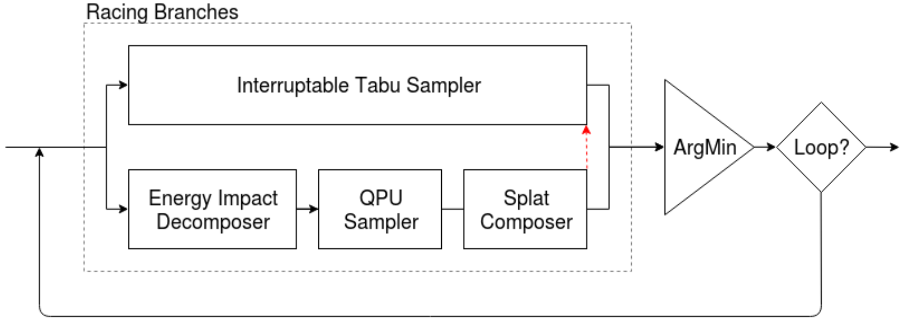
Motivation

Algorithm 1 Partitioning algorithm implemented by `qbsolv`

```
Input: QUBO instance
# best_energy is the lowest value found to date
# best_solution is the solution bit vector corresponding to the lowest value so far
# index is the indices of the bits in the solution, sorted from
#     most to least impact on value

# Get initial estimate of minimum value and backbone
solution ← random 0/1 vector
(best_energy, best_solution) ← TabuSearch(QUBO, solutio
index ← OrderByImpact(QUBO, best_solution)
passCount ← 0
solution ← best_solution
while passCount < numRepeats do
    change ← false
    for i = 0; i < fraction * Size(QUBO); i += subQUBOSize
        # select subQUBO with other variables clamped
        sub_index ← i : i+subQUBOSize−1
        subQUBO ← Clamp(QUBO, solution, index[sub_index])
        (sub_energy, sub_solution) ← DWaveSearch(subQUB
        # project onto full solution
        if (solution[sub_index] ≠ sub_solution) then
            solution[sub_index] ← sub_solution
            change ← true
        end if
    end for
    if not change then
        Randomize(solution[0 : i − 1])
    end if
    (energy, solution) ← TabuSearch(QUBO, solution)
    if energy < best_energy then
        best_energy ← energy
        best_solution ← solution
        passCount ← 0
    else
        passCount ++
    end if
    index ← OrderByImpact(QUBO, solution)
end while
Output: best_energy, best_solution
```



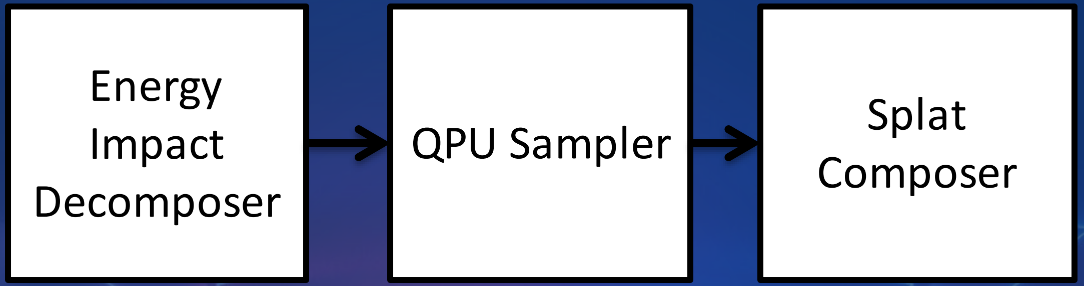

```
Loop(RacingBranches(
    InterruptableTabuSampler(),
    EnergyImpactDecomposer(size=50)
    | QPUSubproblemAutoEmbeddingSampler()
    | SplatComposer()
) | ArgMin())
```



Decomposition

Example

- Find a sub-problem with a high energy impact and solve that on the QPU

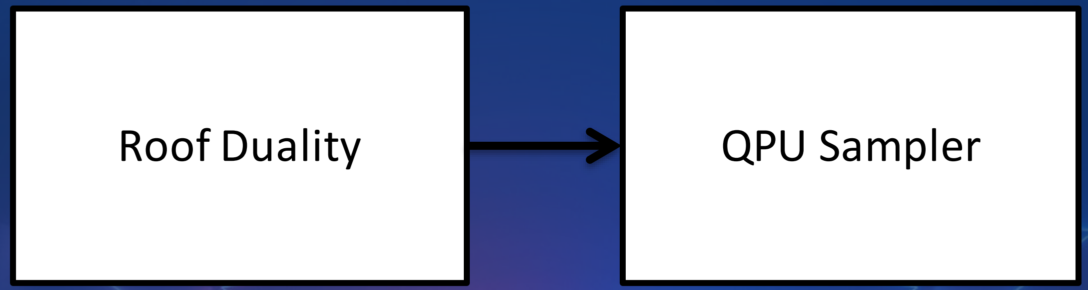



```
EnergyImpactDecomposer(size=50)
| QPUSubproblemAutoEmbeddingSampler()
| SplatComposer()
```

Pre-processing

Example

- Use roof duality to determine and fix the assignments of some variables in polynomial time




```
RoofDualityDecomposer()
| QPUSubproblemAutoEmbeddingSampler()
| SplatComposer()
```

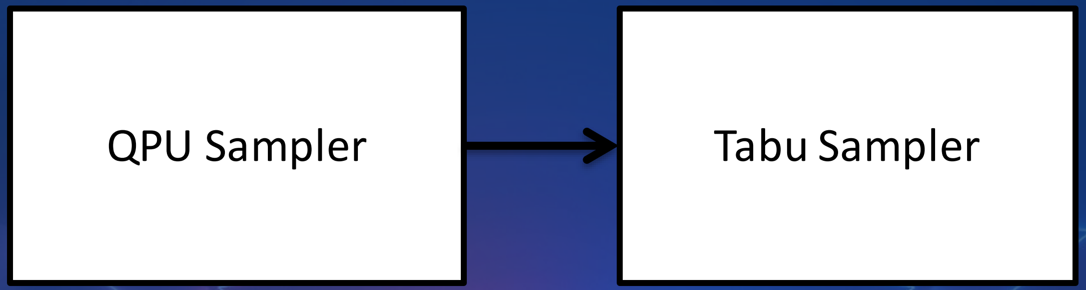
Boros, E., P.L. Hammer, G. Tavares. Preprocessing of Unconstrained Quadratic Binary Optimization. Rutcor Research Report 10-2006, April, 2006

Boros, E. P.L. Hammer. Pseudo-Boolean optimization. Discrete Applied Mathematics 123, 2002, pp. 155-225

Post-processing

Example

- Use the QPU to seed another classical algorithm



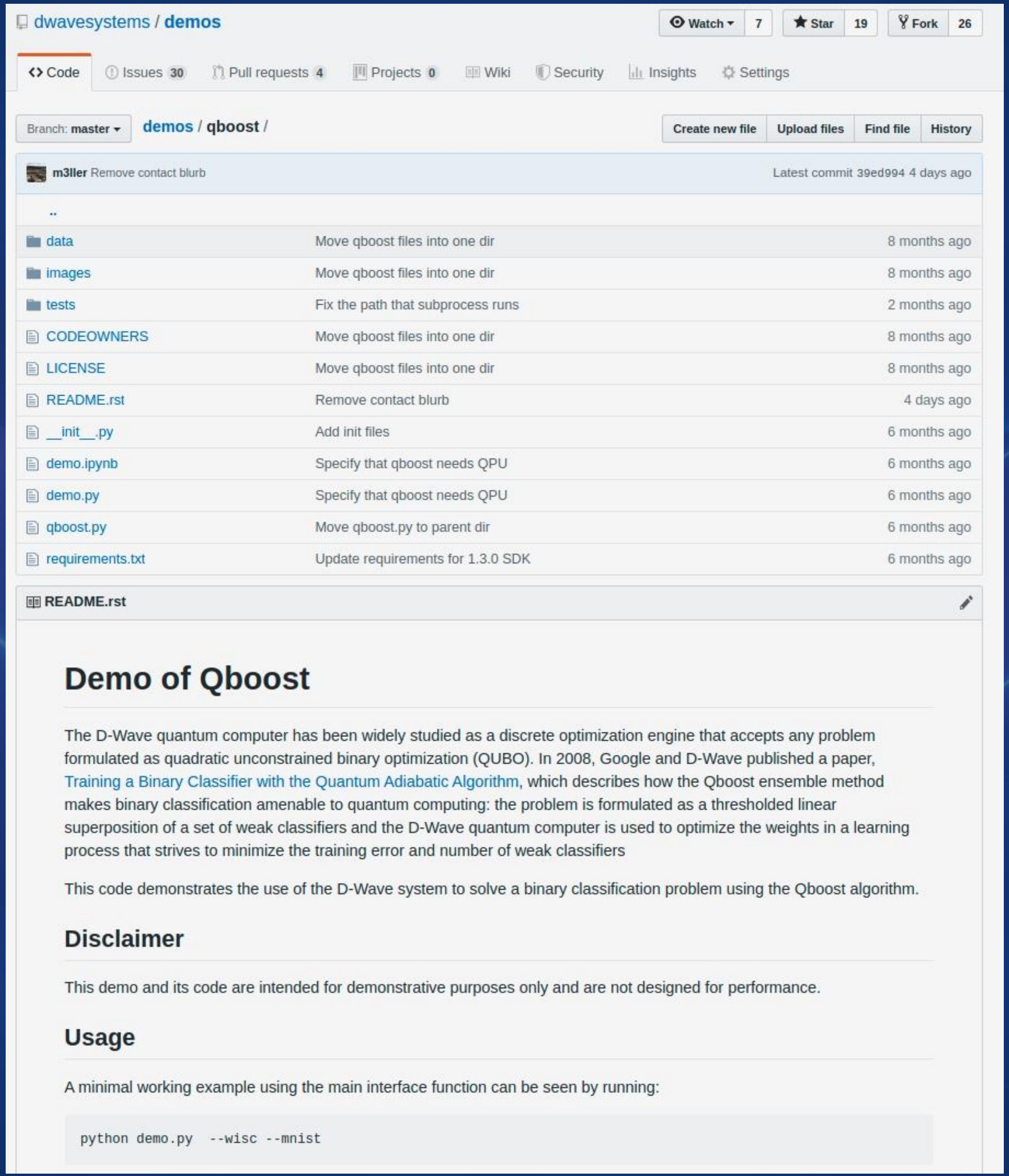
QPUSubproblemAutoEmbeddingSampler()
| TabuSubproblemSampler()

Meta-algorithms

Example

- Use a quantum computer to build a strong classifier from a selection of weak classifiers
- Co-developed with Google to train image classifiers for cars

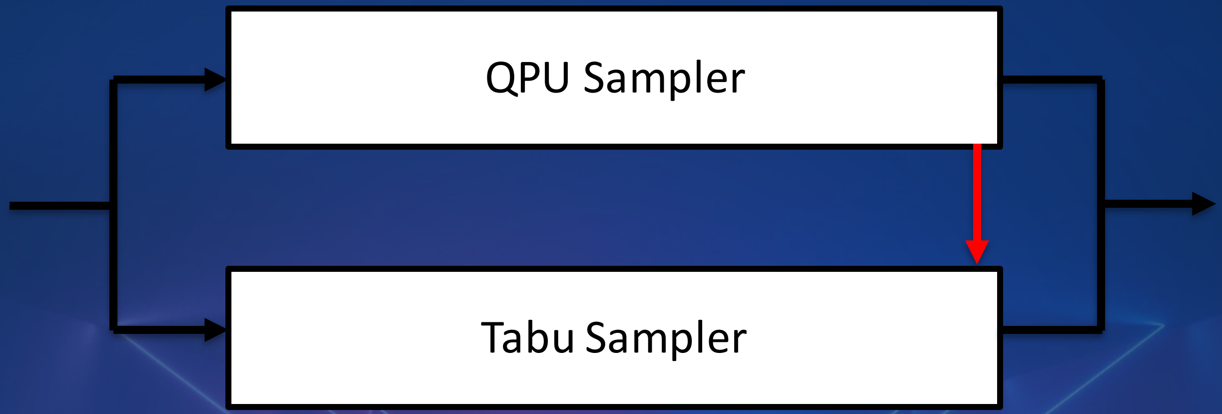
Neven, Denchev, Rose, Macready. Training a Large Scale Classifier with the Quantum Adiabatic Algorithm. https://arxiv.org/abs/0912.0779, Dec 2009



Racing

Example

- Run tabu while waiting on the result from the QPU




```
RacingBranches(
   InterruptableTabuSampler(),
   EnergyImpactDecomposer(size=50)
   | QPUSubproblemAutoEmbeddingSampler()
   | SplatComposer()
) | ArgMin())
```