What’s Behind Door # n?

An Introduction to Quantum Computing

Agenda

What is Quantum Computing?

QC Basics

Types of Quantum Computers

Quantum Simulators

Practical Applications Using Quantum Computers

Short Live Quantum Computing Demo using the DWave Quantum Annealer

Who am I?

- Nick Allgood
- Working on my PhD at UMBC in Baltimore, MD
- Research interests are quantum computing, computer networks, and cybersecurity.
- Working in all aspects of technology (particularly networking) for over 13 years.

What is Quantum Computing?

In a nutshell, quantum computing is utilizing phenomena from quantum physics and using it as a model for computation.

Vastly different from classical computing (but not entirely…)

i.e: In a classical computer, a bit can only be 1 or 0. In a quantum computer, a bit can be 1 AND 0 at the same time (known as superposition).

Quantum Computing Currently

While we technically have quantum computers now (debatable), quantum computing is still very much in its infancy.

Often compared to the 1950’s of classical computing (back in the punch card days).

While IBM, Dwave, IonQ and others have come up with (debatable) quantum hardware, there is a lot of pre and post processing work that MUST be done classically.

There currently is not a formalized assembly language, but many quantum simulators have something equivalent.

Quantum Computing Basics - Maths

Quantum physics is very math heavy and as such makes use of the following disciplines of math (probably more):

- ***Linear Algebra
- Geometry
- Statistics / Probability Theory
- Calculus
- Group Theory
- Abstract Algebra
- <Insert maths I'm forgetting>

Quantum Computing Basics - Physics

Having a background in classical and quantum mechanics is helpful, but not required.

Classical mechanics are a way to describe the study of motion based on Sir Isaac Newton’s laws (Newtonian).

Quantum mechanics focuses more on the wave-particle duality and quantization of energy.

Hilbert Spaces

Created by David Hilbert, a Hilbert space is an abstract vector space of *complex numbers that generalizes a Euclidean space of n-dimensions.

** Complex numbers are an extension of real numbers to include imaginary numbers.*

In other words, one could visualize this as a multi-dimensional container.

In quantum computing, we typically stick with Hilbert spaces of 2^n to represent n-qubits. (Though qutrits and qudits exist: 3^n and 4^n).

i.e. A 2 qubit system would be represented by a 4-dimensional hilbert space.

Qubits!

A quantum bit or qubit is the quantum equivalent of a classical Shannon bit.

A classical bit is a boolean value represented by a 1 or 0 (True or False)

A qubit is a superposition of 0 and 1 at the same time where one number tells you the probability of being 0 and one tells you the probability of being a 1*.

100% chance of being 0, 0% chance of 1

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

0% chance of being 0, 100% chance of 1

In quantum mechanics, you will often see "0" being -1 and "1" being 1

Notations, notations, notations...

Dirac notation / braket notation is the standard notation used in quantum mechanics to represent a quantum state.

This notation is really used to represent vector spaces.

There are two parts to this notation, the bra and the ket.

Bra vs kets

A ket is used to represent a vector space as a column.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

A bra is used to represent a vector space as a row.

$$\langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

While it may seem like a bra and ket are the same, they can have different uses individually -- hence why they are separate.

Eigenstuff

Eigenvectors are vectors that are non-zero and only changes by a scalar factor when a linear transformation is applied to it.

The scalar that is used to change the eigenvector is known as an eigenvalue.

In quantum computing, the terms eigenvector and eigenket are mostly interchangeable.

Eigenvector typically represented by: **v**

Eigenvalue typically represented by: λ (lambda)

Qubits and Brakets

Ket's and bra's are used to represent a single qubit (though ket's are the most common).

A single qubit is nothing but a vector space, as such we can start to do some math on them. We do an tensor (Kronecker) product on two qubits and we get a larger vector space**.

$$|0\rangle \otimes \langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

The result expands the size of our quantum system exponentially, in addition provides us with the resulting quantum state of the system.

***Literature often implies and omits the tensor symbol.* $|0\rangle \langle 0| = |0\rangle \otimes \langle 0|$

Noise in Quantum Systems

Two types: Classical and Quantum.

Classical noise is typically due to known factors such as vibrations, variations thermal energy in a laser emitter, but could be also due to unknown reasons.

Quantum noise is known as decoherence which simply put is the loss of quantum behavior.

While quantum physicists are interested in studying noise, computer scientists wish to reduce the noise as much as possible (to get the most accurate result).

Logical Qubits

Physical qubits are very noisy and cause a large amount of problems with measurement.

Quantum error correction is required to attempt to identify and reduce any and all quantum noise.

Logical qubits are built from one or more physical qubits which not only provide a level of redundancy, but retain quantum information longer (longer decoherence time).

Still require error correction.

Quantum Systems and Quantum States

A quantum system “is a portion of the whole universe taken to analyze the wave-particle duality in that system [3].” A quantum state is a particular state of everything going on in that system at that present time (similar to a snapshot).

A quantum computer is a quantum system and its state is a ‘snapshot’ of the all the qubit data and linear combinations being performed on that data.

You will see a lot of references to energy and amplitudes when reading about quantum computing...this is the quantum mechanics coming through...with quantum computing almost being a very thin layer on top.

Quantum Registers

Like classical computing, quantum computers have registers known as quantum registers.

These registers are made up of multiple qubits and calculations are done by manipulating the qubits within a quantum register.

Remember that qubits are represented by a vector space, so that means multiple qubits put together (tensored) make a quantum register...so we can think of a quantum register as a matrix.

Quantum Logic Gates

Similar to classical computing, quantum computing also has logic gates known as quantum logic gates or quantum gates.

Quantum gates are a foundational part of a universal quantum computer.

Represented as a unitary matrix, which is a special type of square matrix.

Many gates are single qubit gates, but there are some that operate on more than one qubit (which then causes quantum entanglement).

Common Quantum Gates

Pauli Gates

X (NOT)

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Y (Rotate Y-Axis)

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Z (Rotate Z-Axis / Phase Flip)

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Hadamard (Superposition)

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

SWAP

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

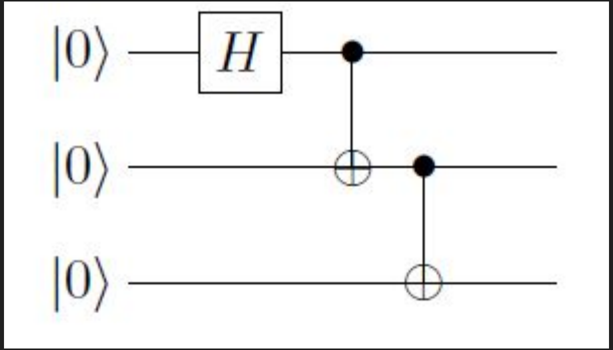
CNOT

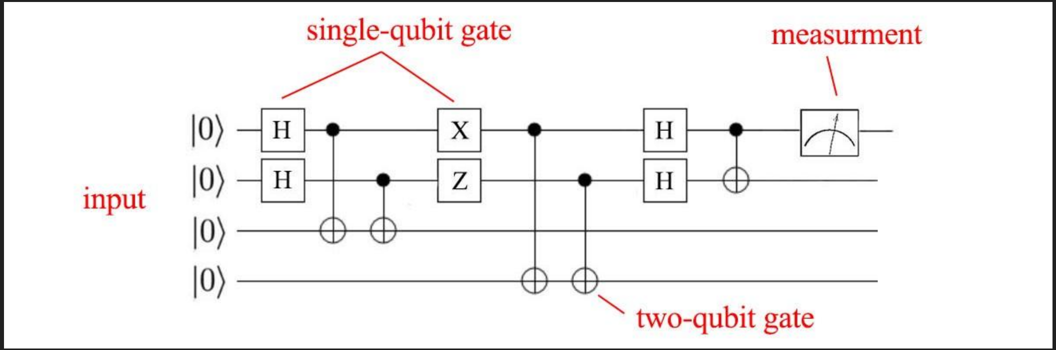
$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Quantum Circuit

Quantum gates are used to build a quantum circuit, much in the same fashion that classical logic gates are used to build a logic circuit.

Examples:





*Original image
https://medium.com/@jonathan_hui/qc-programming-with-quantum-gates-8996b667d256

Quantum Measurement

Quantum measurement is when we measure the state of a quantum system and is measured using an observable (which is represented as a matrix)

When the state of a quantum system is measured, the state “collapses” and the “return value” is a random classical bit… but that information doesn’t help us.

We also get the current state that the quantum system was in which can be transformed into a linear equation.

Quantum Measurement - Useful results

Once the measured state is returned, we also use that information to determine the probability of getting that state.

We repeat this (sampling) many times to get a system of linear equations, which can then be solved using classical computing (post-processing).

Observables are carefully crafted to ensure we get the results we want with a high degree of probability (larger than 97%).

Quantum Entanglement

Quantum entanglement is where a particle interacts with another particle and one can’t tell their quantum states apart.

In quantum computing, this is done when we have a multi-qubit gate applied to at least two qubits.

Once applied, the original state of each of the original qubits can no longer be determined.

I.e. The CNOT gate is a gate that causes entanglement on two qubits.

Quantum Oracles and Blackbox’s

A quantum oracle also known as a “black box” is any function/device that takes some input and returns output...where we do not typically know the inner workings of said function or device.

The tricky (and immensely frustrating) part for quantum computing, is when simulating a quantum algorithm, we often have to guess and implement our own oracle.

In the literature, “blackbox” is used often since..quantum physicists have no interest in the “blackbox”, they just need to know what it does and what it returns..since they will likely not have to implement the oracle.. They depend on computer scientists to do it.

Quantum Algorithms

As the name implies, algorithms designed to run on a quantum computer (or simulated one).

Many can be exponentially faster than classical algorithms due to exploiting superposition and quantum entanglement.

Many make heavy use of the quantum fourier transformation (which is the quantum equivalent of the fourier transformation).

Some Famous Quantum Algorithms

- Deutsch-Josza (Single query of blackbox)
- Simon’s (Blackbox, foundation for Shor’s)
- Shor’s (Factoring / Eventually Breaking RSA)
- Grover’s (Linear Search)

Types of Quantum Computers

Adiabatic (Quantum Annealer): Used to find the minimum global state and is the more common of quantum computers seen today. Typically useful for optimization problems (NP-Hard) [5].

Analog (Quantum Simulator): These are used to simulate complex quantum interactions and able to detect untraceable quantum details. Most likely will start to see dramatic improvements over classical computing [5].

Universal: The “holy grail” of quantum computing that combines the full power of classical and quantum computation and will be the most general purpose quantum computer [5].

Hybrid: Combines quantum annealing and classical computing together in one machine, but only uses quantum CPU (QPU) for certain operations [6].

Measurement Based: Starts with a fixed entangled state and then a sequence of measurements [16].

Quantum Simulators (Software)

There are many quantum simulators that can be run on classical computers to simulate a quantum computer.

Many of these are free and open source.

- Qrack (Free and Open Source) [7].
- Microsoft Q# (Free and Mostly Open Source) [8].
- IBM Qiskit (Free and Open Source) [9].
- ...and loads more!

Quantum computing used today

Roswell Park Cancer Institute is using the D-Wave annealer to optimize IMRT radiation treatments [10].

Booz-Allen Hamilton is using the D-Wave annealer to optimize satellite convergence [11].

Volkswagen is using the D-Wave annealer for real-world traffic flow optimization [12].

Accenture is utilizing 1Qbit for molecule comparisons to accelerate drug discovery for neurological and neurodegenerative conditions [13].

D-Wave Annealer Demo

D-Wave allows people to sign up for free to access time on one of their annealers using their LEAP platform. Once signed up you can get an API token.

Their API is in C and Python 2.x.

Good documentation with lots of demos using Jupyter notebooks.

D-Wave 2000Q Chimera Architecture

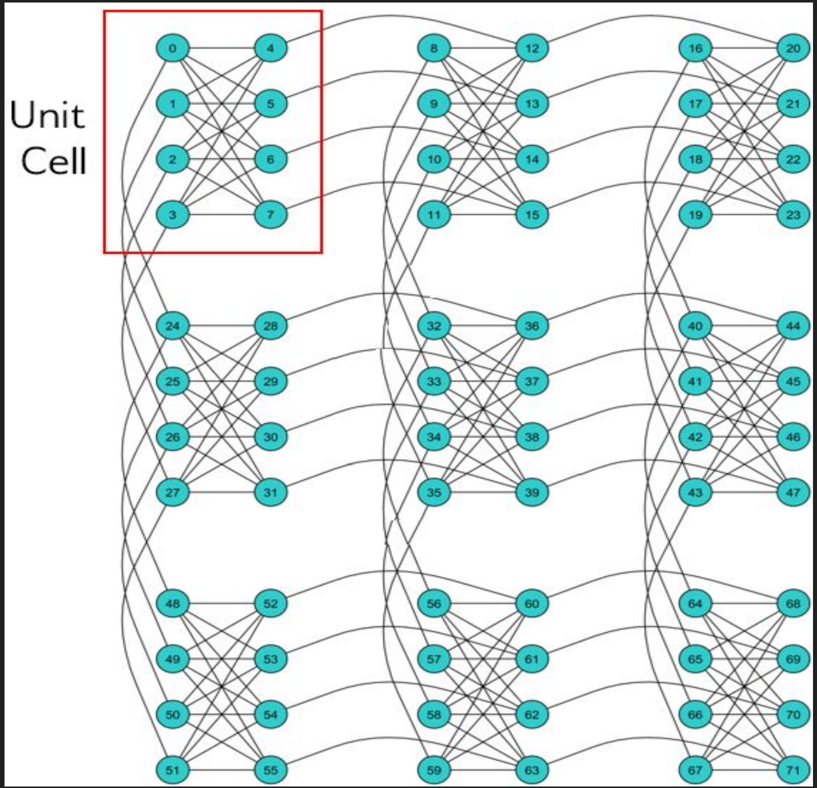


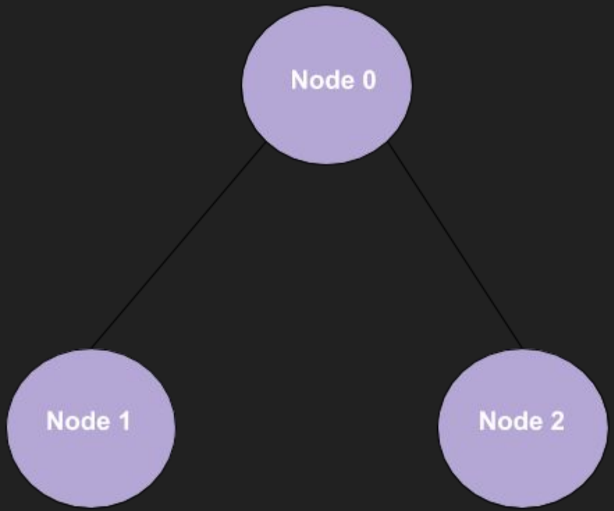
Image courtesy of https://docs.dwavesys.com/docs/latest/c_gs_4.html

Steps To Run On D-Wave (Manual Embedding)

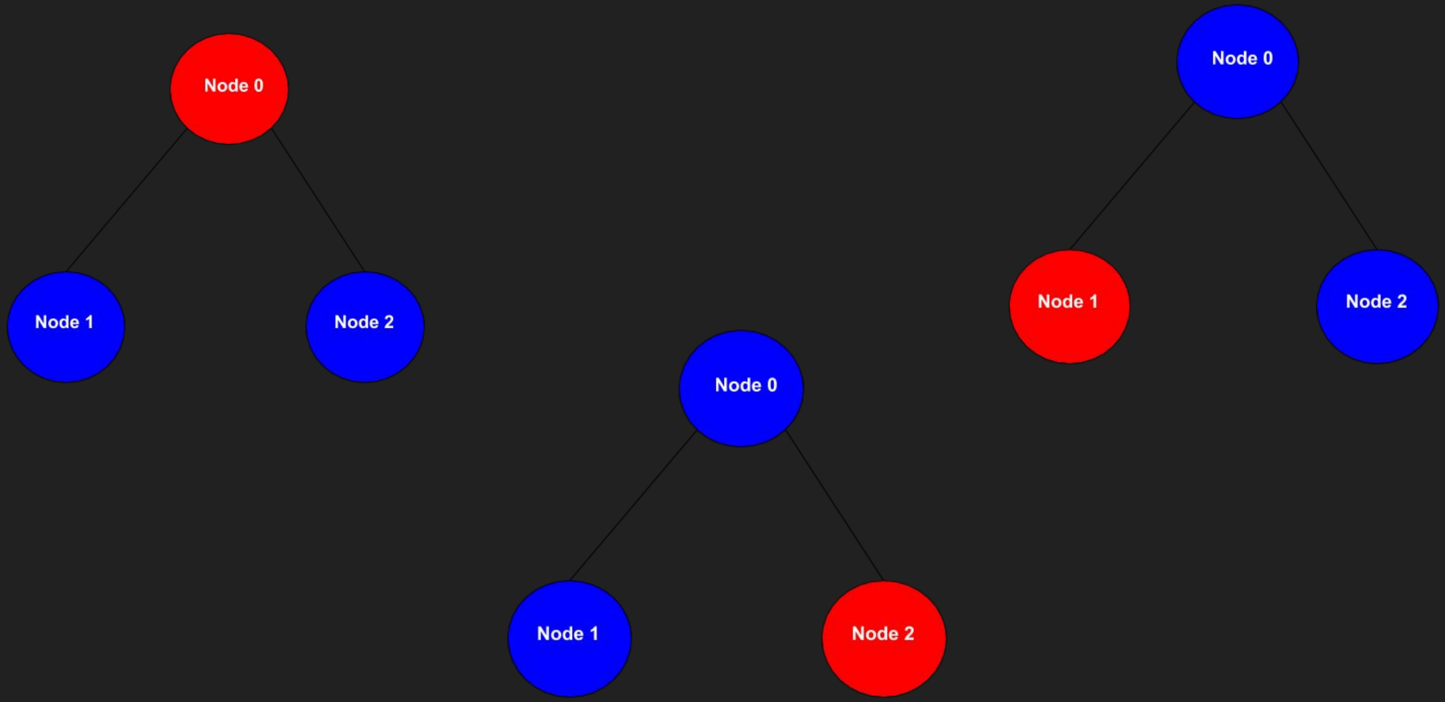
- Formulate problem to fit on D-Wave architecture
- Carefully formulate your Hamiltonian operator.
- Embed/Encode graph into D-Wave system
- Choose the result type and amount of samples
- Submit to D-Wave solver
- Get raw results
- Decode Results
- Print results

Problem Statement

We will map a 3-node graph onto the D-Wave 2000Q and assume all nodes will start out as a superposition of colors **red** and **blue**. We wish to come up with a solution to get only one node as color **red** and the rest as color **blue.**



Three Possible Solutions



Problem Details

The D-Wave annealer minimizes the Ising Hamiltonian operator that is provided by the user.

Our original Hamiltonian is: $H = (node_0 * node_1) + (node_0 * node_2)$

We only want the first solution, so we need a bias value, which changes our Hamiltonian: $H = node_0 + (node_0 * node_1) + (node_0 * node_2)$

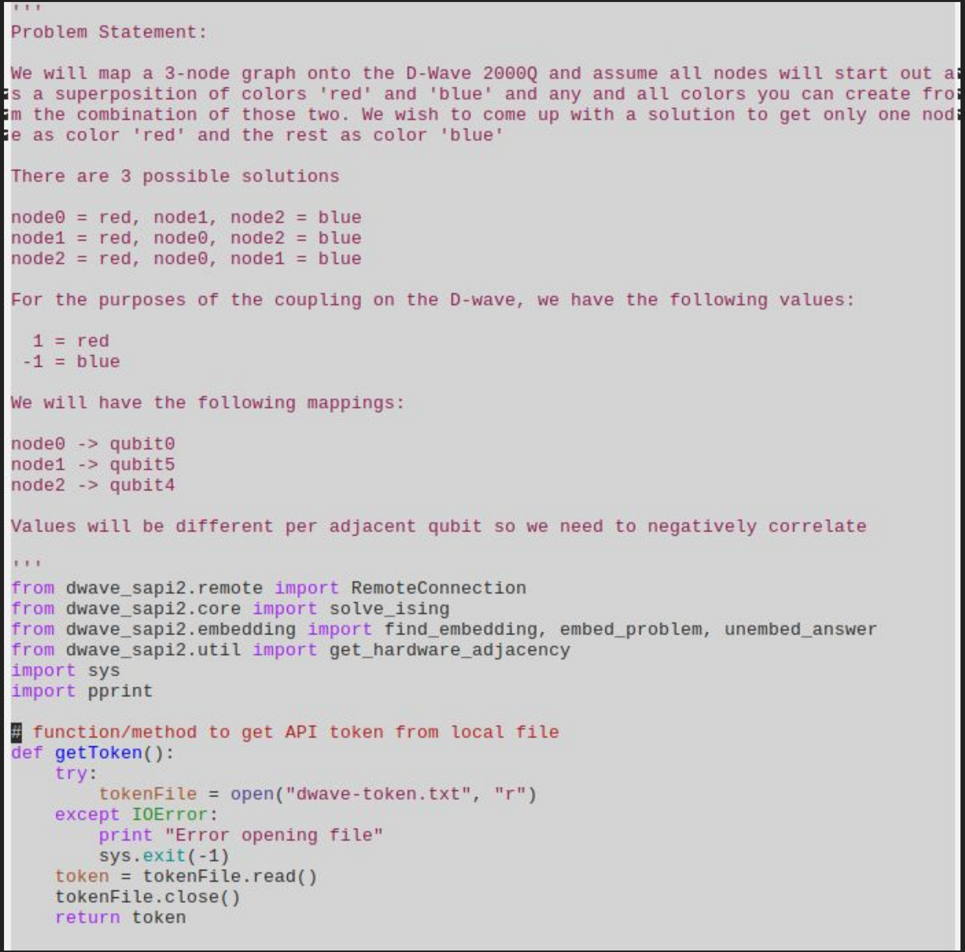
We will use ‘1’ to be **red** and ‘-1’ to be **blue**

Manually Mapping the Problem

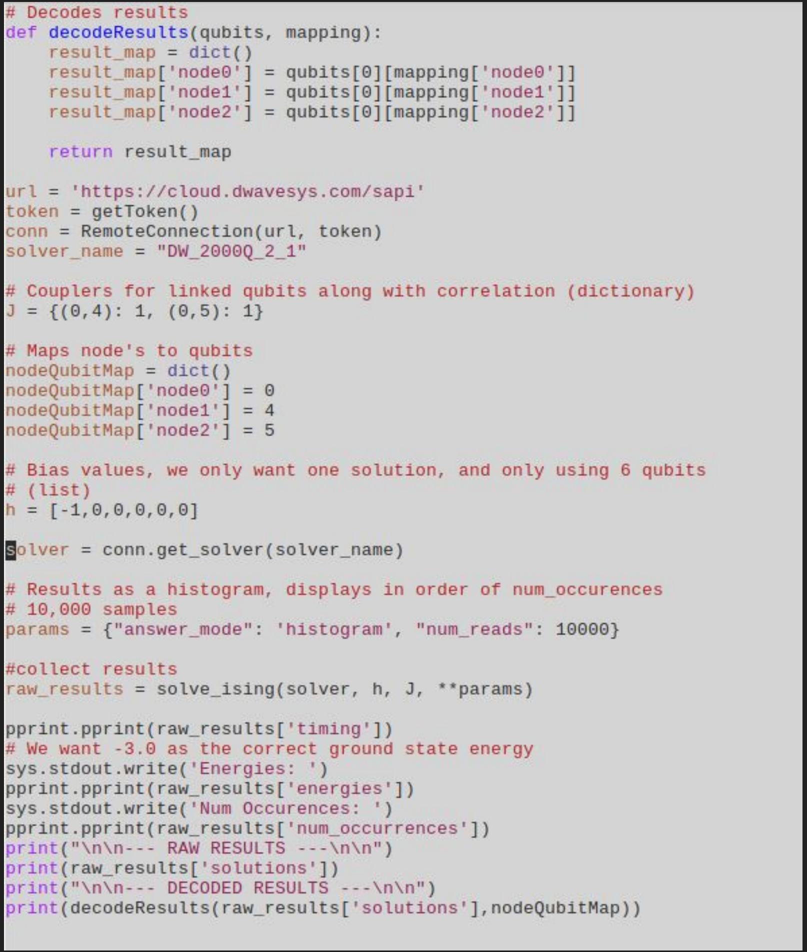
D-Wave supports manually mapping the problem directly to qubits or to utilize their API to automatically embed and map the problem (This is typically what you would want to do, especially for larger data).

We will do the following Node to Qubit mapping:

Node	**Qubit**
node0	qubit0
node1	qubit4
node2	qubit5



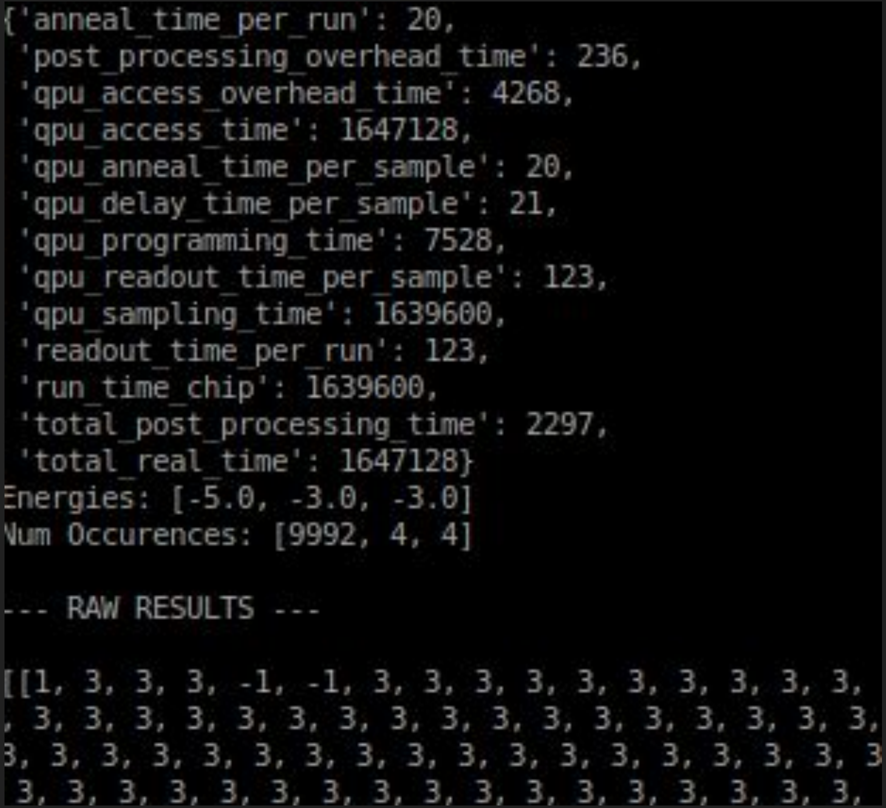
```python
'''
Problem Statement:

We will map a 3-node graph onto the D-Wave 2000Q and assume all nodes will start out a
s a superposition of colors 'red' and 'blue' and any and all colors you can create fro
m the combination of those two. We wish to come up with a solution to get only one nod
e as color 'red' and the rest as color 'blue'

There are 3 possible solutions

node0 = red, node1, node2 = blue
node1 = red, node0, node2 = blue
node2 = red, node0, node1 = blue

For the purposes of the coupling on the D-wave, we have the following values:

  1 = red
 -1 = blue

We will have the following mappings:

node0 -> qubit0
node1 -> qubit5
node2 -> qubit4

Values will be different per adjacent qubit so we need to negatively correlate

'''
from dwave_sapi2.remote import RemoteConnection
from dwave_sapi2.core import solve_ising
from dwave_sapi2.embedding import find_embedding, embed_problem, unembed_answer
from dwave_sapi2.util import get_hardware_adjacency
import sys
import pprint

# function/method to get API token from local file
def getToken():
    try:
        tokenFile = open("dwave-token.txt", "r")
    except IOError:
        print "Error opening file"
        sys.exit(-1)
    token = tokenFile.read()
    tokenFile.close()
    return token
```



```python
# Decodes results
def decodeResults(qubits, mapping):
    result_map = dict()
    result_map['node0'] = qubits[0][mapping['node0']]
    result_map['node1'] = qubits[0][mapping['node1']]
    result_map['node2'] = qubits[0][mapping['node2']]

    return result_map

url = 'https://cloud.dwavesys.com/sapi'
token = getToken()
conn = RemoteConnection(url, token)
solver_name = "DW_2000Q_2_1"

# Couplers for linked qubits along with correlation (dictionary)
J = {(0,4): 1, (0,5): 1}

# Maps node's to qubits
nodeQubitMap = dict()
nodeQubitMap['node0'] = 0
nodeQubitMap['node1'] = 4
nodeQubitMap['node2'] = 5

# Bias values, we only want one solution, and only using 6 qubits
# (list)
h = [-1,0,0,0,0,0]

solver = conn.get_solver(solver_name)

# Results as a histogram, displays in order of num_occurences
# 10,000 samples
params = {"answer_mode": 'histogram', "num_reads": 10000}

#collect results
raw_results = solve_ising(solver, h, J, **params)

pprint.pprint(raw_results['timing'])
# We want -3.0 as the correct ground state energy
sys.stdout.write('Energies: ')
pprint.pprint(raw_results['energies'])
sys.stdout.write('Num Occurences: ')
pprint.pprint(raw_results['num_occurrences'])
print("\n\n--- RAW RESULTS ---\n\n")
print(raw_results['solutions'])
print("\n\n--- DECODED RESULTS ---\n\n")
print(decodeResults(raw_results['solutions'],nodeQubitMap))
```

**Code inspired by Mark Fingerhuth [14] and also available on my Github [15].*

Raw Results



```
{'anneal_time_per_run': 20,
 'post_processing_overhead_time': 236,
 'qpu_access_overhead_time': 4268,
 'qpu_access_time': 1647128,
 'qpu_anneal_time_per_sample': 20,
 'qpu_delay_time_per_sample': 21,
 'qpu_programming_time': 7528,
 'qpu_readout_time_per_sample': 123,
 'qpu_sampling_time': 1639600,
 'readout_time_per_run': 123,
 'run_time_chip': 1639600,
 'total_post_processing_time': 2297,
 'total_real_time': 1647128}
Energies: [-5.0, -3.0, -3.0]
Num Occurences: [9992, 4, 4]

--- RAW RESULTS ---

[[1, 3, 3, 3, -1, -1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3
 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
```

Getting Useful Results

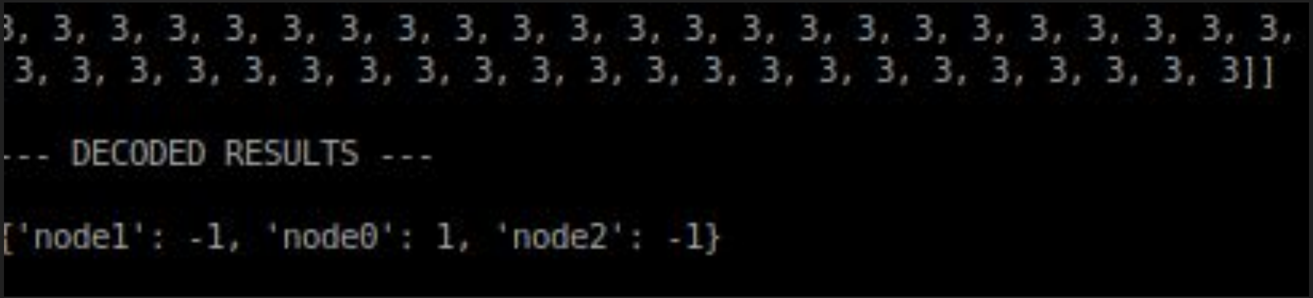
We currently have a large list which has the values of all qubits. We're only using the first 6 qubits so we can take the following snippet from the list:

[1, 3, 3, 3, -1, -1, 3, 3, …]

To get the results we are looking for, we need to reverse the mappings we did earlier.

Qubit	Node
qubit0	node0
qubit4	node1
qubit5	node2

Decoded Results



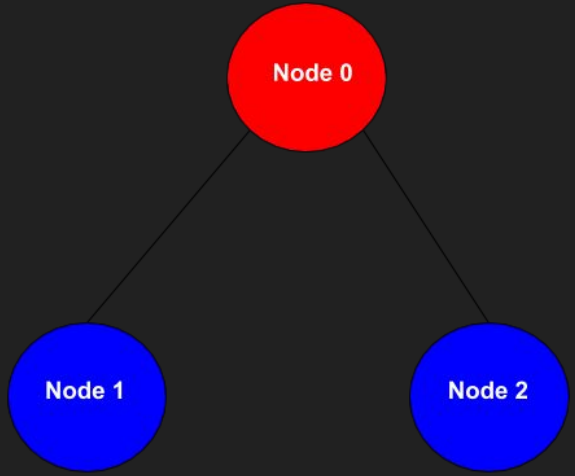
```
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]]

--- DECODED RESULTS ---

{'node1': -1, 'node0': 1, 'node2': -1}
```

Final Results

The value of ‘3’ is the default value for an unused/unmeasured qubit.

After applying the reverse of our mapping, we have the following: [1,-1, -1] which will correlate to:

Node	Color	Measured Value
node0	red	1
node1	blue	-1
node2	blue	-1



Questions ?

Special Thanks!

Special thanks to the following who reviewed this presentation and provided valuable input.

- Mr. Ajinkya Borle (UMBC - PhD Candidate)
- Dr. Samuel Lomonaco (UMBC - Professor of Computer Science)
- Mr. Daniel Strano (Qrack Creator and Lead Developer)
- Mr. Benn Bollay (Qrack Contributor)

Thank You!

Always up for discussions can during/after Fosscon.

If you missed me, you can email me at nick.allgood@gmail.com

Slides and code will be uploaded to my Github
https://github.com/nallg00d/fosscon2019

References

[1] D-Wave Inc. *D-Wave Leap.* https://cloud.dwavesys.com/leap/signup/

[2] Rieffel, Eleanor Polak, Wolfgang. *Quantum Computing A Gentle Introduction.* The MIT Press Cambridge, MA. 2011 Massachusetts Institute of Technology. ISBN 978-0-262-01506-6.

[3] Nielsen, Michael A. Chuang, Isaac L. *Quantum Computation and Quantum Information.* 2010 Cambridge University Press, New York. ISBN 978-1-107-00217-3

[4] Matzke, Douglas J. *Quantum Computing Using Geometric Algebra*. Published 2002. Accessed Aug. 5, 2019. http://www.matzkefamily.net/doug/papers/PHD/phd_matzke_final.pdf

[5] Desjardins, Jeff. *The Three Types of Quantum Computers and Their Uses*. Visual Capitalist. Published March 14, 2016. Accessed August 7, 2019. https://www.visualcapitalist.com/three-types-quantum-computers/

References (cont.)

[6] D-Wave Inc. *D-Wave Announces Quantum Hybrid Strategy and General Availability of D-Wave Hybrid.* Published June 26, 2019. Accessed August 7, 2019. https://www.dwavesys.com/press-releases/d-wave-announces-quantum-hybrid-strategy-and-general-availability-d-wave-hybrid

[7] Strano, Daniel. Bollay, Benn. V*M6502q and Qrack*. https://github.com/vm6502q/qrack https://qrack.readthedocs.io/en/latest/

[8] Microsoft Research. *The Q# Programming Language.* https://docs.microsoft.com/en-us/quantum/language/?view=qsharp-preview

[9] IBM Research Quiskit Community. *Qisket*. https://qiskit.org/

[10] D-Wave Inc. *Quantum Annealing Applied to Optimization Problems in Radiation Medicine.* https://www.dwavesys.com/sites/default/files/Radiotheraphy-Optimization-Roswell-Park_0.pdf

References (cont.)

[11] D-Wave Inc. Booz-Allen Hamilton. *Heterogeneous Quantum Computing For Satellite Communication.* https://www.dwavesys.com/sites/default/files/QuantumForSatellitesQubits-4.pdf

[12] D-Wave Inc. Volkswagen. *Quantum Computing at Volkswagen: Traffic Flow Optimization using the D-Wave Quantum Annealer.* https://www.dwavesys.com/sites/default/files/VW.pdf

[13] D-Wave Inc. Accenture. *Quantum Computing Advance Drug Discovery. Accenture.* https://www.accenture.com/us-en/success-biogen-quantum-computing-advance-drug-discovery

[14] Fingerhuth, Mark. *How Do You Write a Simple Program for a D-Wave Device.* https://quantumcomputing.stackexchange.com/questions/1451/how-do-you-write-a-simple-program-for-a-d-wave-device

[15] Allgood, Nick. Github: https://github.com/nallg00d/fosscon2019

[16] Josza, Richard. *An Introduction to Measurement Based Quantum Computing.* Department of Computer Science. Bristol, BS8 1UB UK. https://arXiv:quant-ph/0508124v2 . Published Sept 20, 2005. Accessed August 13, 2019.

[17] Allgood, Nick. *fosscon-maths-2019.* https://www.overleaf.com/project/5d48cd21b448c478cc66e647

[18] D-Wave Inc. *Getting Started With The D-Wave System.* Published October 13, 2018. Accessed August 16, 2019. https://docs.dwavesys.com/docs/latest/_downloads/09-1076A-T_GettingStarted.pdf