References and Code

February 20, 2019

Contents

I Quantum Computing

Just some quick notes on the hardware aspects of all current quantum devices. 1st they are all in dilution refrigerators which hold them at about 15 millikelvin. 2nd, ranging form every hour to every 24 hours each device gets all of its qubits and qubit connections re calibrated. Lastly, All of these devices are Niobium or Niobium alloy electronics, typically printed on chips similar to classical electronics. The reason for this is that at 9 Kelvin, Niobium becomes a superconductor.

1 Adiabatic Quantum Annealing

The D-Wave devices use adiabatic quantum annealing based optimization, based on applying two different Hamiltonians on the physical chip in two different anneal schedules; as seen in 1. Figure 2 shows the principal behind quantum advantage that could be gained by using this adiabatic *quantum* annealing.

2 Gate Based Quantum Computing

Figure 3 shows the space in which Pauli operators change the superposition of a qubit.

II QAOA

In table 1 we list all of the classical optimization algorithm which can be used by each of the 2 code distributions used in this research.

The QAOA algorithm relies on these classical optimizes to optimize the variational parameters, the expectation values for which are measured by the quantum component of the algorithm (specifically the Variation Quantum Eigensolver (VQE)).

III QUBO Formulations

The table 2 shows the various QUBO formulations for each of the types of problems tested.

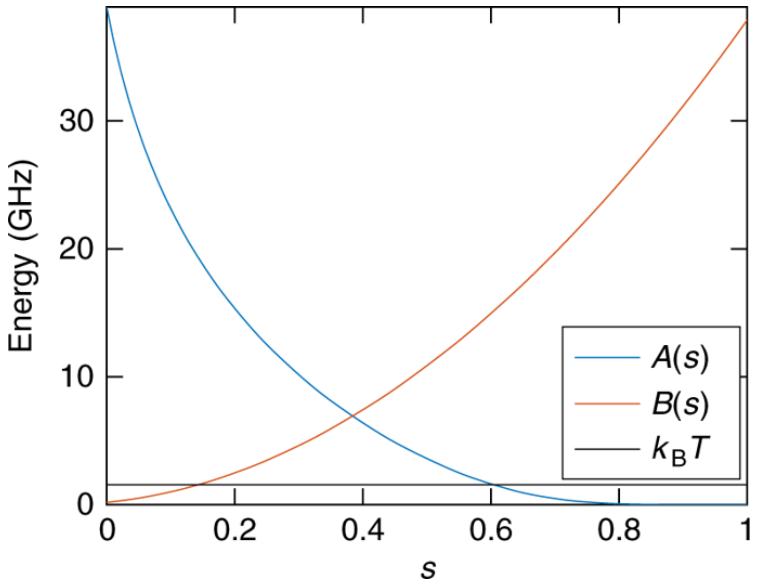

Figure 1: This diagram described the Hamiltonian application schedule which describes the adiabatic quantum annealing process

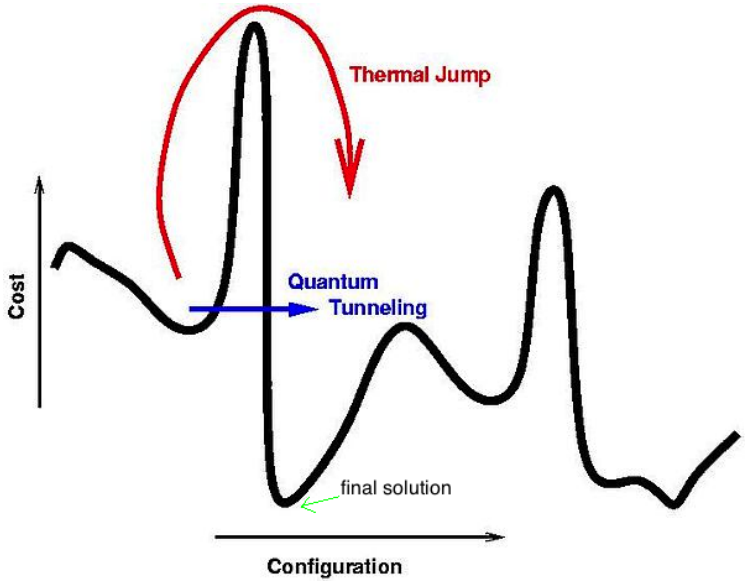

Figure 2: This diagram shows the potential quantum advantage gained by quantum adiabatic annealing when attempting to find a minimum energy level on a complex energy landscape

IV Device Connectivity

This section goes over the various hardware connectivity graphs for the NISQ devices used (Figures 5, 6, 7, 8, 9), as well as a D-Wave 2X device (Figure 4).

1 D-Wave

Ideally, these graphs would be complete; in general, the oldest generations of D-Wave devices (up until a recent announcement of graph connectivity change) follow the chimera graph architecture defined by 3 dimensions; x, y, m. m is the dimension defined by the bipartite aspect of the chimera construction: (2, m). Then x and y are the "height" and "length" of the resulting chimera graph, where each (2, m) unit cell is connected to each other unit cell by 4 connections to each direction as long as the limit of the overall dimensions for the chimera graph is not reached. Therefore, the ideal hardware graph for D-Wave 2X devices is C(12,

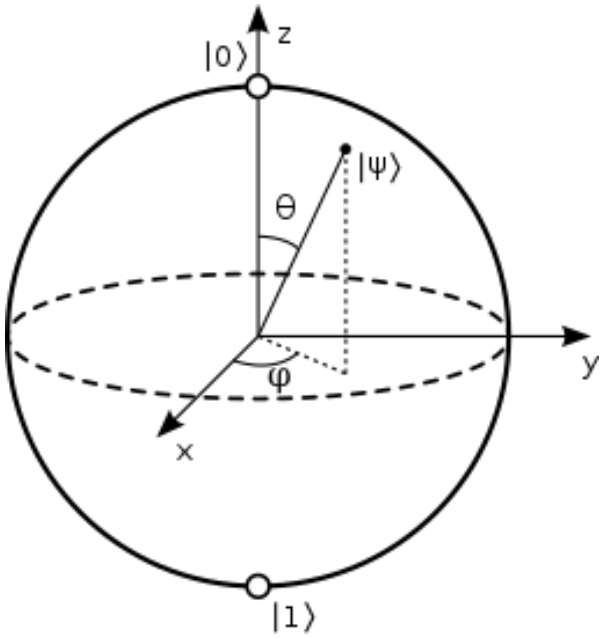

Figure 3: This is the Bloch sphere, rotation about this sphere describes the various single qubit unitary gates for Gate based quantum computation

Rigetti (Grove)	IBMQX (Qiskit-Aqua)
Nelder-Mead	Nelder-Mead
COBYLA	COBYLA
POWELL	POWELL
CG	CG
SLSQP	SLSQP
L-BFGS-B	L-BFGS-B
TNC	TNC
NEWTON-CG	SPSA
BFGS	P-BFGS

Table 1: Potential classical optimizer algorithms

NP-Hard Optimization Problem QUBO Formulations	
NP-Hard problem	QUBO formulation
Maximum Clique	$-A\sum_{i=1} x_i - B\sum_{i,j\in\overline{E}} x_i x_j$ (1)
Minimum Vertex Cover	$A\sum_{uv\in E}(1-x_u)(1-x_v) + B\sum_v x_v$ (2)
Maximum Cut	$\sum_{i,j\in E}(x_i\overline{x}_j + x_j\overline{x}_i)$ (3)

Table 2: QUBO's for each NP-Hard problem tested

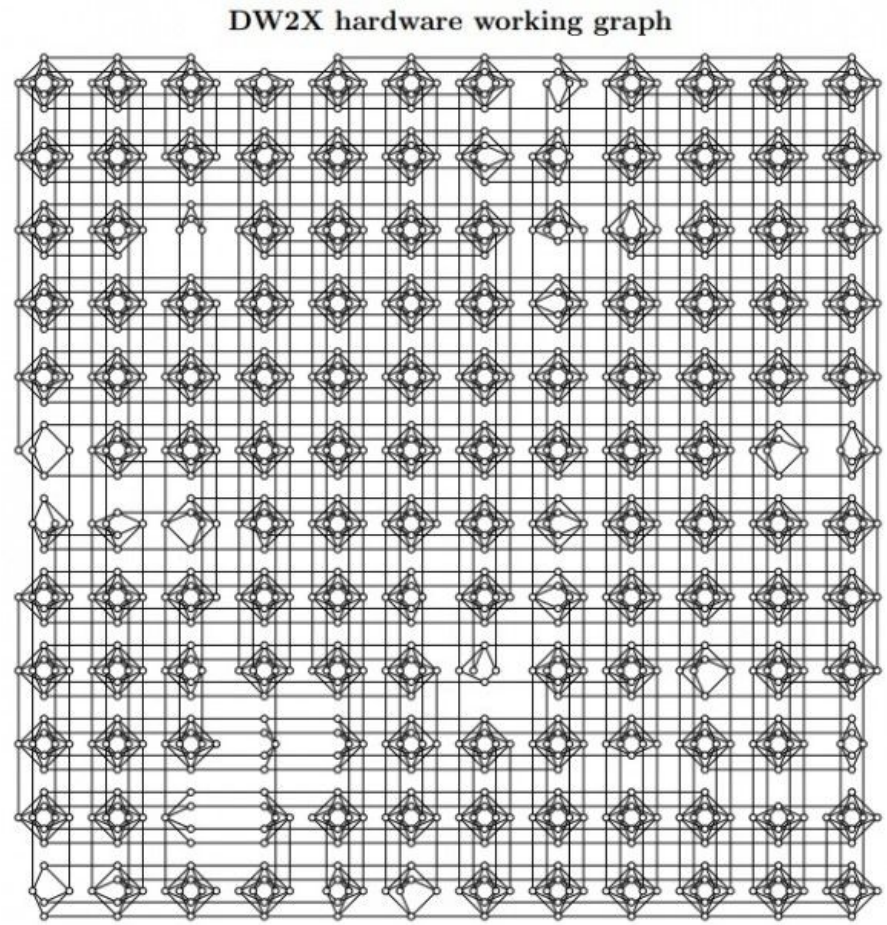

Figure 4: Above is the connectivity map for the D-Wave 2X at LANL.

12, 4). The above connectivity map is obviously missing some edges and nodes; this is due to manufacturing and field defects in the hardware. The device used in this research has less hardware defects, but is also a larger chimera graph construction (16, 16, 4), which is the updated properties of the D-Wave 2000Q. The specific device being used is called Dwave-2000Q-2-1, which has 2038/2048 of the nodes, as well as 8923 edges out of 82398 edges.

2 IBM Quantum Experience NISQ Devices

These two devices are actually used in this research project, even though they are in a somewhat limited capacity due to nature of the networked connection to these devices.

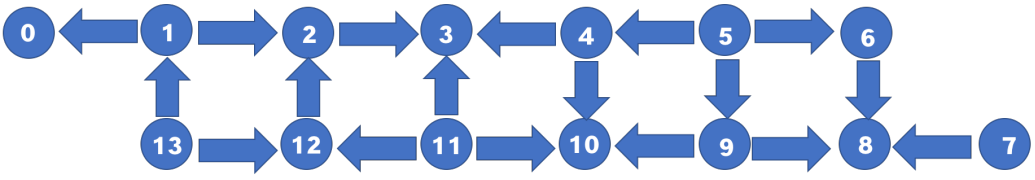

Figure 5: The CNOT connectivity map for the device melbourne publically available on the IBM Quantum Experience

3 Rigetti Computing NISQ Devices

Currently, only the device Aspen is online, and only available to a limited pool of users. Therefore, even though the programming infrastructure is available to use these devices, for the current state of this research, it is not feasible.

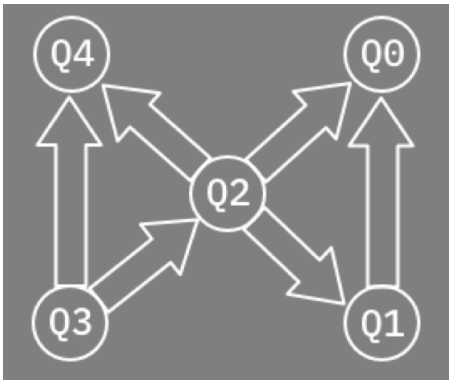

Figure 6: The CNOT connectivity map for the ibmqx4 Tenerife device publically available on the IBM Quantum Experience

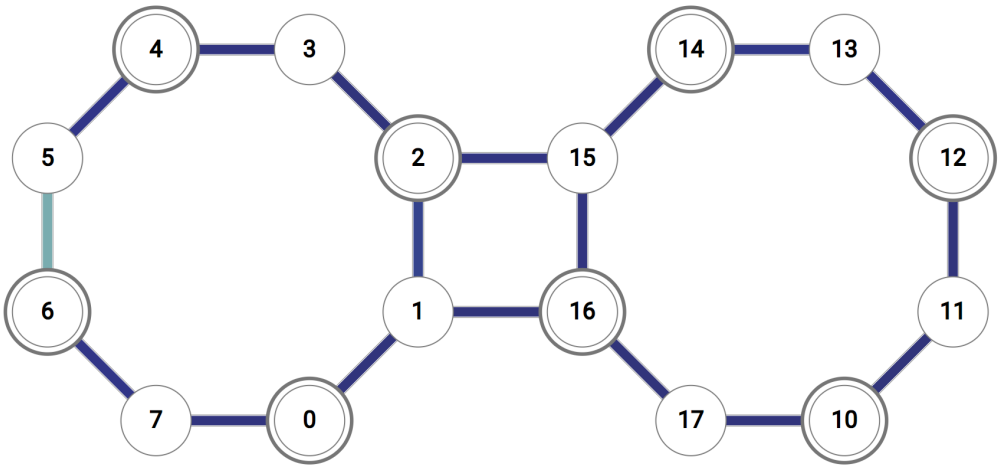

Figure 7: The CNOT connectivity map for the Aspen device

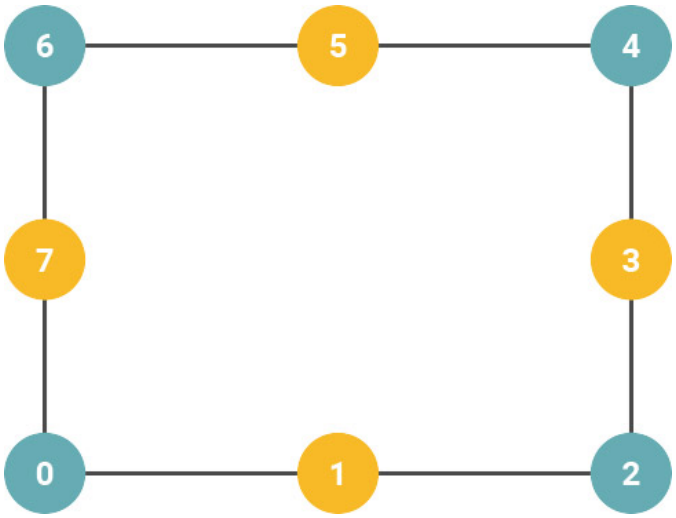

Figure 8: The CNOT connectivity map for the Agave device

V Code

The full documentation and repository for all aspects of the project, including data graphing scripts can be found at my public Github repository `https://github.com/epelofske/quantum_optimization`. The code relies on the python modules from both

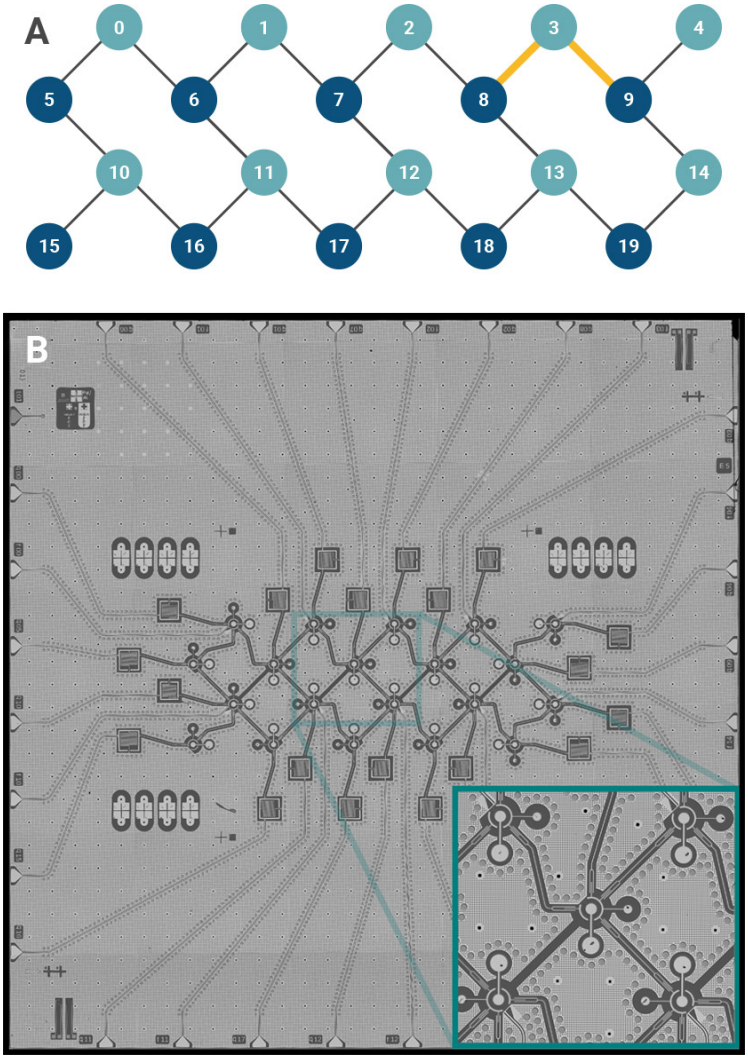

Figure 9: The CNOT connectivity map for the Acorn device (Top), as well as an image of the physical device (Bottom)

IBM and Rigetti, the open source projects for which can also be found on Github (Qiskit-Terra: `https://github.com/Qiskit/qiskit`, Qiskit-Aqua: `https://github.com/Qiskit/qiskit-aqua`, PyQuil: `https://github.com/rigetti/pyquil`, Grove: `https://github.`. A short example of one specific piece of Python code which was used in this research is provided, although the specific functions called in it will not likely be well understood unless the source code is analyzed.

Lastly, at the end of the QAOA, a resulting series of gates is the result, or more specifically the result from running that algorithm. The specific input to QAOA varies what this end result code is, and the number of steps (trotterization order) we repeat the algorithm also affects its length. Attached is code resulting from one of test graphs in this research in the form of Quil, as well as the equivalent QASM code, specifically showing the differences in the code when the number of steps is varied. Specifically, this was generated using the Gnp(7, 0.5, 101) random graph generator, and the classical optimizer for the classical portion of QAOA SLSQP was used. And this was done on the QVM, with no noise models. All of the attached code, including the Python, Quil, and QASM code, is attached at the very end of this document, after all of the figures.