Using QAOA to Solve NP-Hard Problems on NISQ Computers

New Mexico Supercomputing Challenge Final Report
Los Alamos High School, Team 29

Elijah Pelofske

April 2019

Abstract

NP-Hard problems are a class of optimization problems which can be solved in exponential time in the best case, which is a significant obstacle because problems of this type have many important applications, such as bioinformatics. Quantum computing has the potential to drastically increase the speed at which important problems, such as NP-Hard optimization problems, can be solved. Most quantum algorithms which can solve important and sufficiently large problems require fault tolerant qubits, and high qubit number and connectivity. Unfortunately the best quantum computers to date are referred to as Noisy Intermediate-Scale Quantum (NISQ) computers, and have low qubit number and connectivity. Therefore, in order to apply quantum computers to useful applications, quantum algorithms which can be used on cloud accessible NISQ devices must be implemented. In this case, the approximation hybrid quantum-classical optimization algorithm QAOA (Quantum Approximate Optimization Algorithm) will be tested because it can be implemented on currently accessible NISQ computers. As an additional comparison, an adiabatic quantum annealer device (DWave 2000Q) is compared. The viability of using NISQ computers to solve industry relevant NP-Hard problems will be investigated using QAOA in comparison to classical solvers and a DWave 2000Q device.

Contents

1 Introduction

Quantum computers offer the potential to solve many industry relevant problems in areas such as chemistry, artificial intelligence, optimization, and cryptography. All of these applications are based on quantum algorithms, which have been created over the past roughly 25 years, including the most well known of which are Shors Algorithm [17] and Grover Search Algorithm [6]. Cloud accessible Noisy Intermediate-Scale Quantum (NISQ) computers are, at present, the current state of quantum computing in creating fully scalable and fault tolerant gate based quantum computers. In general, NISQ devices cannot run most quantum algorithms on problems of any meaningful size, largely due to limited connectivity, small number of qubits, and high gate and measurement error rate. These limitations of NISQ devices lead to restrictions on embed able problem size, and shallow quantum circuit depth.

The Quantum Approximation Optimization Algorithm (QAOA) [13] is specifically designed to run in polynomial time on NISQ devices, and find optimal (and therefore possibly optimum) solutions to optimization problems. This algorithm is therefore appealing as a means to solve critical optimization problems which classically take exponential computational complexity to solve exactly.

The structure of this paper is as follows; the introduction outlines the background and details of this area of research, as well as overall project objectives. Section 2 defines and particular problem which will be considered, as well as their relevance. Section 3 defines the experimental setup and procedures. Section 4

describes the various results from the experiments, and section 5 analysis the specifics of these results, and finally we conclude with section 6.

1.1 Quantum Computing

The concept of quantum computing is based on the idea of a quantum bit (qubit). Qubits are typically small loops of superconductive material, which have been cooled to 15 millikelvin. Microwave pulses directed at specific points on these small circuits can apply quantum "gates" or unitary matrices to these qubits, which causes a change in the wavefunction of the qubit. If a qubit is not measured (i.e. it has zero decoherence), then it is in a superposition between two values (0 and 1). Once measured, that value "collapses" into one of those two. However, if we took a statistically significant series of measurements, we would begin to see the statistical distribution of the superposition of the qubit. For example we might see that 70 percent of the time the qubit is in the 1 state, and 30 percent of the time it is in the 0 state.

Next, once we add multiple qubits and multi qubit gate connectors (in this case, Controlled-NOT gate), we can begin to look at quantum entanglement. This is the idea that multiple qubits can have correlated state values - where if we measure one qubit on an imaginary fault tolerant quantum computer, after applying some specific gates, the state of the other qubits can be determined. After some gates have been applied, the final state can be measured by some classical sensors, and thus we get a binary string out. That binary string can be one of 2^N states, where N is the number of qubits on the quantum computer.

The primary properties of NISQ devices are limited qubit number and connectivity, as well as relatively high decoherence (measurements from thermal fluctuations in the environment), which means high error rates with gate and measurement operations. As more gates are applied in a given circuit, these error rates combine, which leads to shallow usable circuit depth (i.e. only a few gate operations can be performed in a single run). For all NISQ devices, multiple runs (for example up to 8192 runs) can be measured, in order to verify the results are consistent.

Due to the computational speed ups offered by quantum computers, many companies are becoming interested in investing in research and development on quantum computers. Dwave was the first to begin working on quantum computing nearly two decades ago. Many large technology companies such as Intel, Microsoft, Google, and Honeywell are also beginning to invest in these devices. Although it is not detailed in this paper, Google has many NISQ devices which are internally at. In general, each of these companies is investing in one of three types of quantum computing. The first is Adiabatic Quantum Annealing (D-Wave is the only such device in this area), the second is ion trap based quantum computing (IONQ), and the

third is the most popular because of its similarity to classical microchip fabrication -i.e. printed gate based quantum computing (similar fabrication process as D-Wave). Within that third category, there are a few outliers, for example some companies are investing in using silicon or diamond imperfections to create a 3D lattice for quantum computing. And others such as Microsoft are focusing on discovering a new material to create fault tolerant quantum computing, and in the meantime they are focusing on software development.

Many of these companies have also invested in the human interfacing with these devices, specifically the programming languages to be used. IBM created QASM, Rigetti created Quil, and Microsoft created Q # . However, only two of these many companies have allowed public access to some or all of their devices. These are IBM and Rigetti. For this research, only those cloud accessible NISQ devices will be used, as well as classical simulators. The devices these companies created both fall into the third category - chip manufactured gate based quantum computing.

1.2 Technical Details

This section is purely for background knowledge of the hardware and software used in this research, and is meant only for the reader who might not have a full background on the topic.

All modern quantum computers are inherently designed to be cloud accessible, because the interface is too inaccessible to most people, and especially the general public, because of the extreme environmental condition requirements for a quantum device to operate. Namely, a very large dilution refrigerator to cool the actual physical chip down to about 15 millikelvin, and a near perfect vacuum inside the device. All of this is an an effort to reduce noise and interactions from the atmosphere or thermal vibrations.

1.2.1 Software

At present, two companies (IBM and Rigetti) have publicly released software kits for quantum algorithm development as well as a system for distributing API keys for running quantum algorithms on NISQ devices constructed by these companies. Due to the nature of the error rates on NISQ devices, both IBM and Rigetti have also provided simulators, including an option for remote HPC simulators, which can run up to roughly 32 qubits. Additionally, these software kits offer the ability to apply noise models to these quantum computing simulators, in order to simulate the error rates which will would occur on a NISQ device. Both of these companies have also released low level programming languages which are designed specifically for constructing and running quantum programs on simulators as well as real world quantum devices. These two programming languages are QASM [11] and Quil [20]. The Github repositories [4], [2] are the largely Python

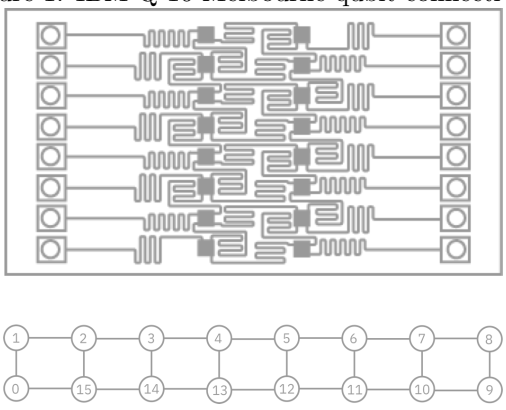
based scripts which provide a basis for quantum algorithm development. Their are algorithm development kits created by Rigetti and the IBM Quantum Experience, these are called Qiskit Aqua [3] and Grove, respectively.

Specifically for this research, personal API keys will be used on available quantum devices, IBMQX4, IBM Q 16 Melbourne, and Agave. The programming architecture for running quantum algorithms is to a large extent already implemented on these platforms. This research focuses on software which implements more and new problems on these devices using QAOA. Right now, there are examples and scripts in the respective Github repositories of some of the NP problems listed, such as maximum cut. However, this project focuses on first converting into an Ising model, which then can be converted into a Pauli operator, and then a quantum program in either QASM or Quil.

1.2.2 Hardware

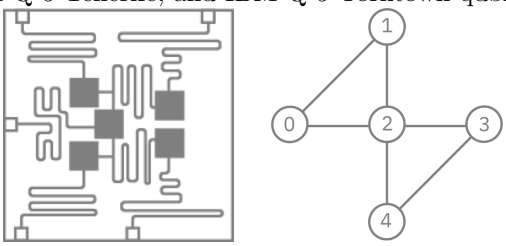
The IBM quantum experience offers a total of 4 NISQ devices for use over the cloud, 2 of which are available to anyone in the public. This research will only use those 2 which are publicly available. The names of these 2 devices are IBM Q 16 Melbourne, and IBMQX4 Tenerife. The numbers in the name represent the number of qubits. The connectivity of the qubits in these devices is not complete - full coupler maps are shown in Figure 1.2.2 and Figure 1.2.2.

Figure 1: IBM Q 16 Melbourne qubit connectivity



Every 24 hours each device is re calibrated, and for every experiment run, that calibration data is stored along with results of the experiment. the calibration data includes measured T1 and T2 times, as well as gate and readout error rates. T1 and T2 times refer to two different, but related, relaxation times of qubits. Essentially, these describe how the gate fidelity decreases over time. The IBM devices are Niobium and Aluminum Oxide (for superconductive Josephson junctions) circuits printed on silicon wafers. They are

Figure 2: IBM Q 5 Tenerife, and IBM Q 5 Yorktown qubit connectivity



fixed frequency qubits, and they have microwave oscillators which allow the readout of the qubits to occur.

Rigetti computing offers 3 NISQ devices accessible over the cloud, both of which are available to the public if they wish to do perform some research on the devices for a set period of time. the first is Agave, which has 8 qubits, and the second is called Acorn, which has 19 qubits (19 logical qubits, 20 physical qubits). The latest is called Aspen-1, and has 16 qubits. The physical qubits are joined via capacitive couplings. Both of these devices have an alternating set frequency to tunable qubit arrangement, although their connectivity differs greatly. This research does not use these devices at present however, because access is still somewhat limited.

The IBM quantum experience offers an HPC simulator for up to about 32 qubits. Both Rigetti and IBM offer basic local simulators, which can simulate up to about 28 qubits fine on a local machine, depending on that machines specifications.

1.3 QAOA

Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical algorithm which is iterative and relies on another quantum algorithm known as variational quantum eigensolver (VQE), the driver and cost hamiltonians, which are problem specific, and classical optimizer algorithms. The full details and proof for the algorithms functionality is given in [13]. QAOA is especially appealing because it runs in polynomial time.

This research focuses on using QAOA to solve NP-Hard problems. 21 of the most famous NP-Complete problems are described in [18]. These NP problems can be formulated as Ising (electronic structure) or QUBO (Quadratic Unconstrained Binary Optimization) models, and these formulations can be translated into driver and cost hamiltonians represented as Pauli operators, which means that these problems can be at least approximately solved by QAOA. In this case, the NP problems will be converted into Ising or QUBO models, the standard hamiltonian formulation of a problem is shown in equation 1. The difference between QUBO and Ising formulations in terms of variables output is that Ising variables will have an output value

of either -1 or 1, and QUBO variables will have an output value of either 0 or 1. The primary tunable parameter for QAOA, besides the constructed hamiltonians, is the number of times that the cost and driver hamiltonian is applied sequentially. In a generally consistent trend, the accuracy of the output increases as the number of steps increases.

1.4 D-Wave

Although not the focus of this research, the D-Wave devices are a specific type of quantum computer, called an adiabatic quantum annealer, which solves only optimization problems.

These devices are quantum annealing devices and are constructed and sold by D-wave [1], [9], [19]. In relation to optimization problems applied to D-wave systems, there is a massive amount of research in this area [7], [5], [14], [21], [12], [8]. Later on in this article, we briefly mention a comparison between a D-Wave 2000Q device and QAOA. In general however, for smaller problems, D-Wave is substantially better than such approximate algorithms, at least right now.

1.5 Background and Related Research

Studying hybrid quantum-classical algorithm is an increasing subject of interest. For example, [22] went into an in depth analysis of applying QAOA to the NP-Hard problem of Maximum Cut, specifically using the algorithm as an exact solver by applying it until the optimum was reached. Additionally, [15] and [10] are similar studies of QAOA, specifically looking into the future usage of QAOA on newer hardware topologies. Such research into the future applications and required performance measurements for hybrid algorithms is one of most viable means to demonstrate some quantum advantage.

2 Problem

Their are many non-deterministic polynomial problems which have many important applications in increasing efficiency in industries ranging from pharmaceuticals, bioinformatics, manufacturing, traffic analysis, and massive data analysis relating to social media on the internet. The problem is that algorithms which find exact solutions to these NP (nondeterministic polynomial time) problems are not efficient, i.e. they do not run in polynomial time - instead they typically run in exponential time in the worst case.

From [16] we know that all of these problems are reducible in a specific manner, i.e. most NP-Complete problems can be reduced into different and simpler NP-Complete problems. This research focuses specifically

on the QAOA algorithm, as it seems to have the most potential to have immediate real world impacts as NISQ devices are continuing to be improved and expanded. As NISQ devices continue to improve, quantum optimization algorithms such as QAOA have the ability to begin having real world impacts faster than other quantum algorithms such as Shors algorithm, which requires fault tolerance, large number of qubits, high circuit depth, and high qubit connectivity.

2.1 NP-Hard Problems Tested

All NP problems in this research can be represented in terms of the mathematical objects called graphs. Graphs are mathematical objects defined by a collection of vertices, which are arbitrarily connected by a series of edges. These nodes and edges can have a numerical value assigned to them, these values are called weights. For the purpose of this research, only weighted versions will be considered, because it is trivial to use the weighted version as an unweighted version - i.e. the weight simply being 1.

2.1.1 Maximum Clique

Let G be a simple undirected graph defined as $G = (V, E)$, where each edge is not weighted. A clique is defined as a complete subgraph of G (i.e. every node in that subgraph is connected to every other node in that subgraph). A maximum clique is therefore the largest clique in G.

A weighted version of both maximum clique is identical to the versions described above, except that each vertex has a weight associated with it, and the value that is maximized is the sum of the weights of the vertices in the respective subsets generated (i.e. either an independent set or a clique), instead of just the number of vertices.

The decision version (NP-Complete Version) of the Maximum Clique problem asks the question of whether there is a clique of size N in a given graph. If the NP-Complete version is solved, the NP-Hard problem can be solved as well. The decision version of Maximum Clique is known to be NP-Complete, and the more general case described above (including the weighted version) are also known to be NP-Hard. Classically, heuristics have been developed to solve these problems more efficiently than an enumeration approach. However, none of these run in polynomial time. There are however approximate algorithms which run very fast, such as the minimal independent set approximation heuristic provided by the Python module Networkx.

2.1.2 Maximum Cut

Let G be a simple undirected graph defined as $G = (V, E)$ where each edge has a weight associated with it. A cut, C, is any nontrivial subset of V, and the cut weight is the sum of the weights of all of the edges crossing the cut. A maximum cut is therefore a cut of G with maximum weight. Determining the maximum cut of a graph is NP-Hard [18]. The decision version of maximum cut is answering the question of does a cut of weight N exist for a graph. This decision version is known to be NP-Complete.

3 Experimental Procedures

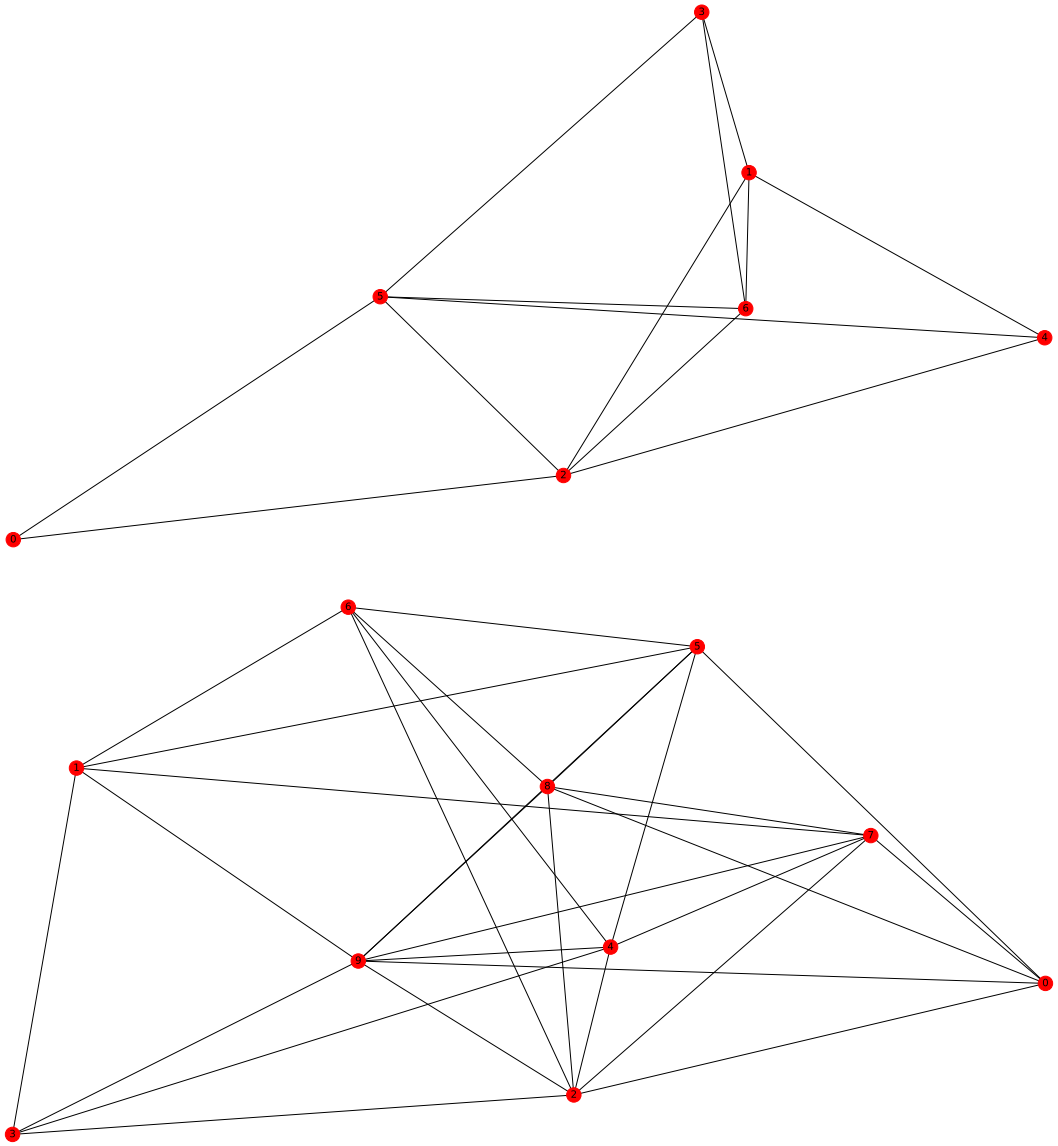
Table 3 shows the QUBO formulations for the two NP-Hard problems which will be solved by QAOA, each of these formulations generally take the form of equation 1.

$$H = \sum_{i \in V} J_{ij}\sigma_i\sigma_j - \mu \sum_j h_j\sigma_j \tag{1}$$

NP-Hard Problem Hamiltonian Formulations	
NP-Hard problem	Hamiltonian formulation
Maximum Clique Nodes	$-A\sum_{i=1} x_i - B \sum_{i,j \in \overline{E}} x_i x_j$ (2)
Maximum Cut	$\sum_{i,j \in E} (x_i \overline{x}_j + x_j \overline{x}_i)$ (3)

Because of the limited size of problems that can be run on current operating NISQ devices, as well as on current available simulators, only small problems are tested. The problem instances (which will be in the form of graphs) will be implemented on the two local simulators available from both Rigetti and IBM (including a noise model test), the real NISQ devices available from IBM. However, only averaged the qasm simulator will be used for large sample sizes involving averaging results. The purpose of these tests is to analyze the viability of NISQ devices - the NISQ device tests can be compared to the simulated noise models and the ideal simulators. Additionally, it is helpful to be able to differentiate between errors due to noise and the inherent nature of QAOA - where in some cases it only finds optimum solutions.

Figure 3: Gnp fixed test graphs

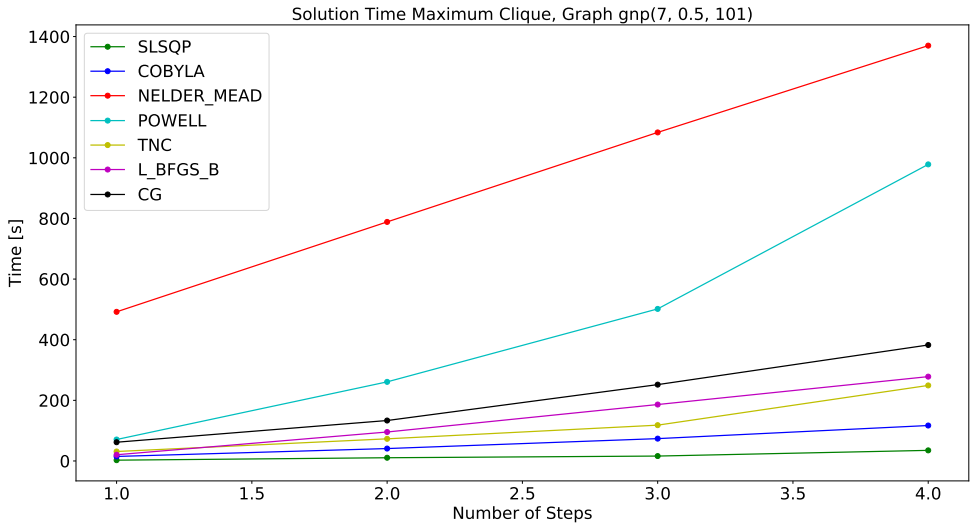


$$Approximation - Ratio = \frac{Average - Experimental - Solution - Size}{Optimum - Solution - Size} \tag{4}$$

The accuracy of the results from running the QAOA algorithm will be determined using the approximation ratio metric, defined in equation 4.

The problem graphs tested are shown in figure 3, and were generated using the Python library Networkx, and fixing the random seed as 101 for consistent graph usage.

Figure 4: Maximum Clique Optimizer Run Time, averaged over 20 runs



4 Results

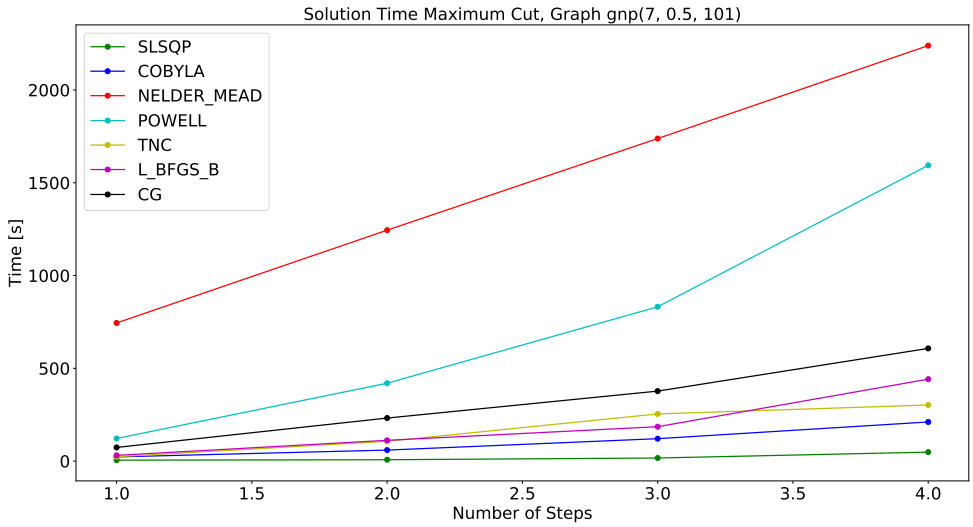
All of the code and code documentation can be found at the authors github repository for this project; `https://github.com/epelofske/quantum_optimization`. Further results are also stored in their raw form as well as in various graphs in the the project repository. The results in the repository include various unaveraged runs from using the Rigetti Quantum Virtual Machine (QVM), as well as results from a D-Wave 2000Q, accessed through the D-Wave Leap program. However, these results are not included in this paper because they are unaveraged, and the D-Wave results do not show anything novel at this comparatively small problem size. The results presented in this paper are averaged over at least 20 repetitions, and use the $C++$ QASM simulator as the classical backend, with the exception of the real world NISQ computers.

Figures 4 and 4 show the run time when using QAOA for several classical optimizers. The best optimizer from that data, SLSQP, was then used for the rest of the accuracy tests because computation time is an important metric for determining the viability of using QAOA. It should be noted that the classical simulation time is included in these graphs, but this does not change the relative measures of run time when comparing these classical optimizers.

Figures 4, 4, 4, 4 show the approximation ratio from both random guesses and using QAOA. The random approximation ratio was averaged over 10000 runs, and is a constant for each graph. The QAOA data was averaged over 100 runs. Once again, figure 3 shows the exact graph which were used for these experiments.

Figures 4 and 4 are unaveraged runs for the problem of maximum cut. These graphs are unique for this

Figure 5: Maximum Cut Optimizer Run Time, averaged over 20 runs



research because this is data from the two physical NISQ devices available from IBM. The problem which was being applied to each of these was simply the undirected version of the architecture defined in 1.2.2 and 1.2.2.

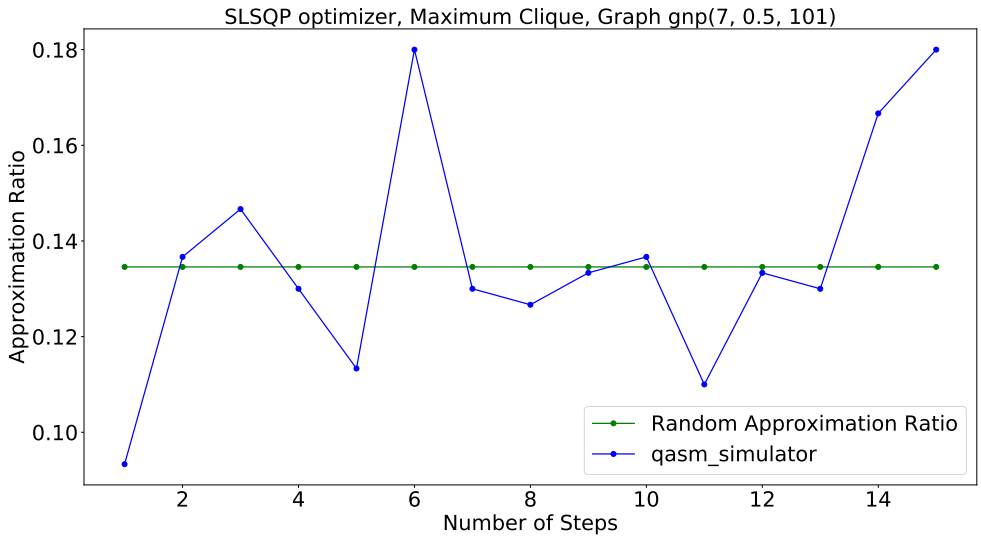
Finally, if the reader is interested in more specifics of the operation of the QAOA algorithm, the Github link provided includes some example quantum machine assembly language in the form of both Quil and QASM. Additionally, all back end software used in this research is open source and linked in the provided Github link.

5 Analysis

There are several important things which are observed in Figures 4 and 4. The first is that when considering the viability of an algorithm, run time is an important consideration, and in this case we find that SLSQP is the most efficient optimization algorithm. The other thing to note is that we can clearly see the polynomial complexity behavior of the algorithm, particularly because at these small problem sizes, the classical run time of simulating quantum systems is negligible.

Figures 4 and 4 show that overall we need faster communication time to the NISQ computers for better averaging, and it is important that we implement simulation noise models of real world NISQ devices to determine a more concise viability threshold. These figures as well as the approximation ratio figures show that overall accuracy, as well as run time, can be impacted by the type and complexity of the QUBO for the

Figure 6: Maximum Clique



particular problem we are considering.

6 Conclusion

From the analysis of the results, we conclude the following regarding using QAOA to extend the capacity to solve such optimization problem such as NP-Hard problems.

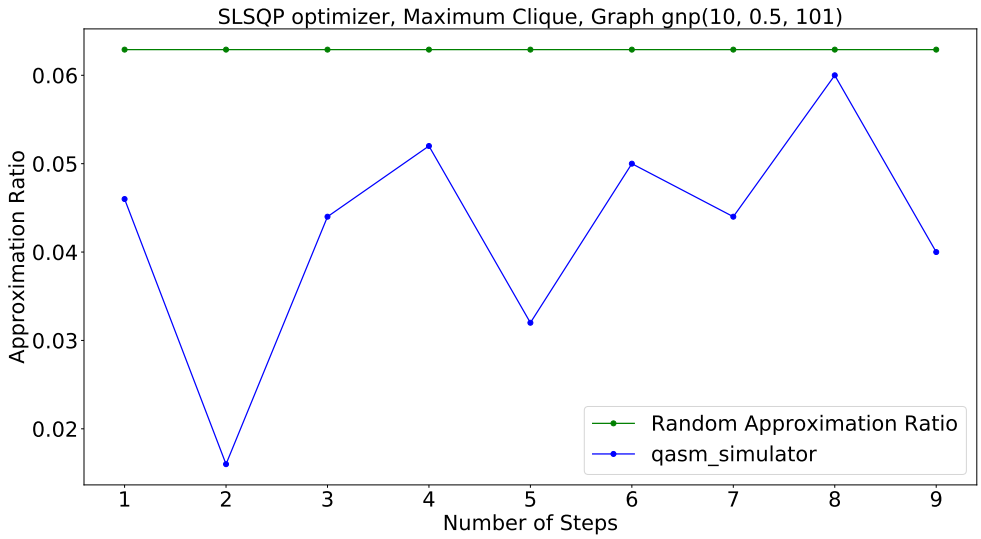
1. The network connectivity time is a significant current limitation - speeding up this communication time, such as the user specific Quantum Virtual Machine from Rigetti is a desirable property for such NISQ devices and the associated applications.

2. The complexity of the QUBO models for these optimization problems heavily affect the run time as well as accuracy for found solution sizes for the QAOA algorithm.

3. Currently, the D-Wave quantum annealers perform much better in comparison to QAOA, particularly for small problems. Further research using larger simulations of NISQ devices and, as the manufacturing improves, larger physical NISQ devices.

4. The performance of the classical optimization algorithm used in the QAOA algorithm is critical for the overall performance of the algorithm. Of all of the algorithms tested, the algorithm SLSQP (Sequential Least Squares Programming) was found to be the most efficient.

Figure 7: Maximum Clique

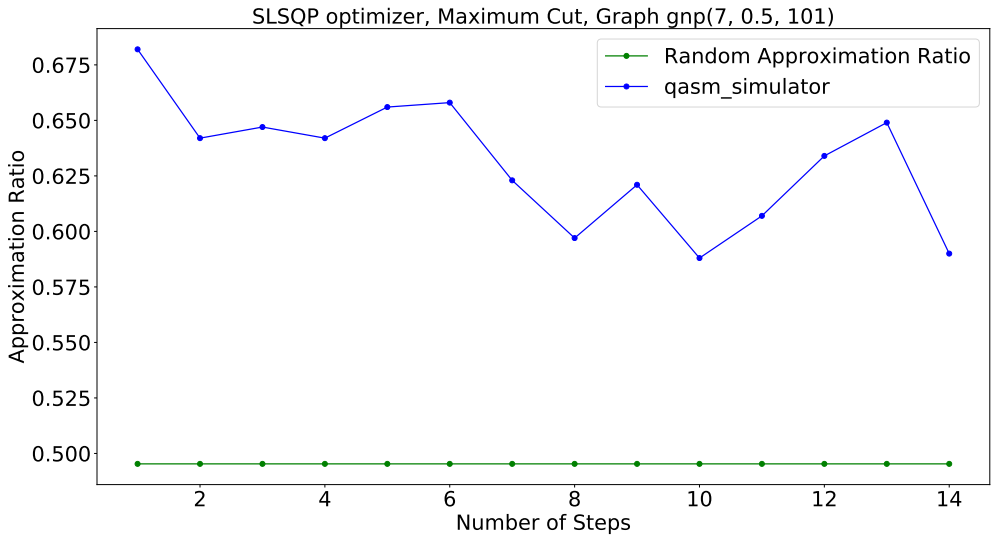


5. The trotterization order of the QAOA algorithm is critical for the overall performance of the algorithm. As a general trend, accuracy increases as the number of steps applied increases. However, a high number of steps is required for some types of optimization problems. For example, maximum cut is the easiest to demonstrate an improvement over random guessing, even at sort circuit depth. However, for more constrained problems, such as maximum clique, a larger circuit depth is required for increased accuracy.

6. The QUBO or Ising formulation of the problem is critical when considering what NISQ device to use, and could be a very specific avenue to demonstrate quantum advantage in the future. For example, the Maximum Clique QUBO formulation tells us that the only quadratic interacting terms come from $\overline{G}$ (the complement of the problem graph). This means that an especially dense problem graph could be solved on very sparse NISQ architecture. This is important because in general it seems that sparse device connectivity is the most viable device construction for the near term, and higher problem graph density is more computationally intensive for classical solvers.

7. In order to make meaningful use of QAOA, not only do we need larger NISQ computers, we also need significantly reduced error rates. Even for a comparatively short circuit depth algorithm, such error rates are a problem. Consequently, even for other quantum algorithms, these hardware improvements are necessary. Even so, this research and several others have demonstrated the hypothetical capabilities of QAOA.

Figure 8: Maximum Cut



8. The IBM and Rigetti software packages include converters for many NP-Hard problems, and any optimization problem which can be represented in the particular Hamiltonian formulation of QUBO or Ising, and therefore QAOA can be implemented on physical NISQ computers using these software platforms.

For future research directions, we propose the following specific areas:

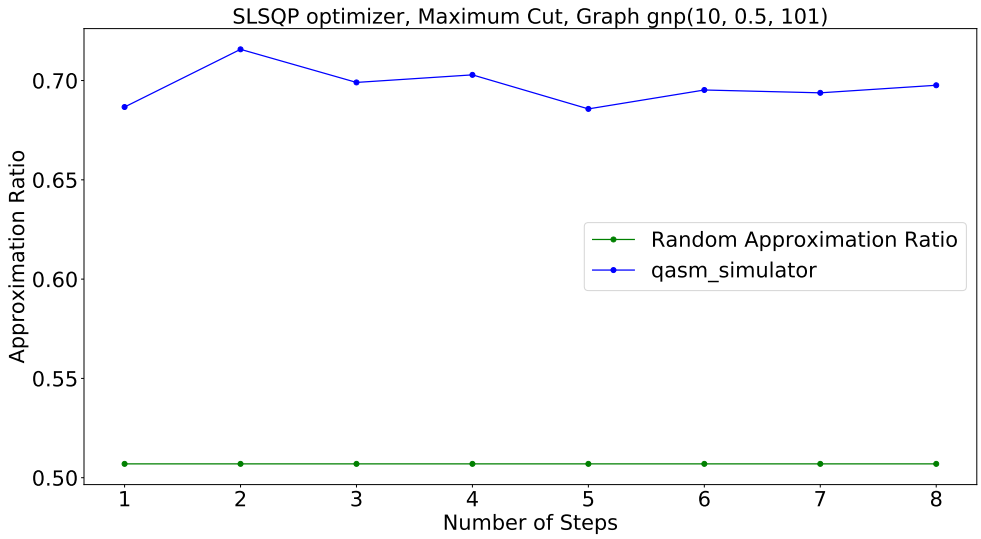
1. Testing more NP-Hard problems, such as TSP, Minimum Vertex Cover, Hamiltonian Cycle, Chromatic Number, and Maximum Independent Set.

2. Potentially paralyzing and increasing the sample size for the averaging process as well as increasing the problem sizes.

3. Applying QAOA on larger quantum computers, including NISQ computers released through IONQ, Rigetti, and future IBM devices.

4. Implementing noise models for various real world NISQ devices, and simulating higher qubit connectivity on classical simulators, thus providing an insight into the performance of higher connectivity NISQ architectures.

Figure 9: Maximum Cut



7 Acknowledgements

We acknowledge use of the IBM Q for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team.

We also thank the Slack communities of Rigetti-Forest and Qiskit, and the D-Wave Leap program.

References

[1] Quantum Computing for the Real World Today. Technical report.

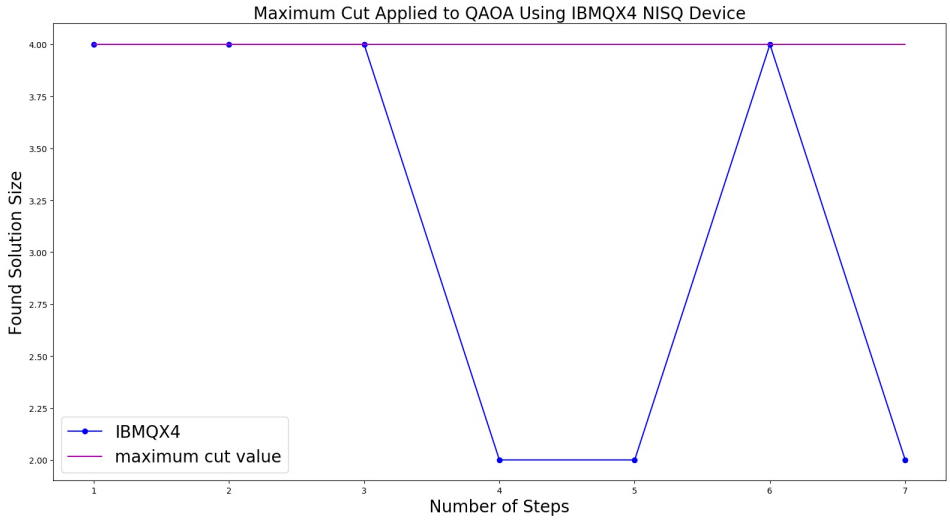
[2] *Pyquil*, 2018 October. A Python library for quantum programming using Quil.

[3] *Qiskit Aqua*, 2018 October. Aqua provides a library and tools to build applications for Noisy Intermediate-Scale Quantum (NISQ) computers.

[4] *Qiskit Terra*, 2018 October. Terra provides the foundations for Qiskit. It allows the user to write quantum circuits easily, and takes care of the constraints of real hardware.

[5] Evgeny Andriyash, Zhengbing Bian, Fabian Chudak, Marshall Drew-Brook, Andrew D King, William G Macready, and Aidan Roy. Boosting integer factoring performance via quantum annealing oosets TECHNICAL REPORT. Technical report.

Figure 10: IBMQX4 NISQ device



[6] Eva Borbely. Grover search algorithm. Technical report.

[7] Guillaume Chapuis, Georg Hahn, and Guillaume Rizk. Finding Maximum Cliques on the D-Wave Quantum Annealer. Technical report.

[8] Carleton Coffrin, Harsha Nagarajan, and Russell Bent. Ising Processing Units: Potential and Challenges for Discrete Optimization. Technical report.

[9] Steve Conway, Earl C ; Joseph, and Robert Sorensen. The Promise of Quantum Computing Strengths and Limitations of Classic Supercomputers. Technical report, 2016.

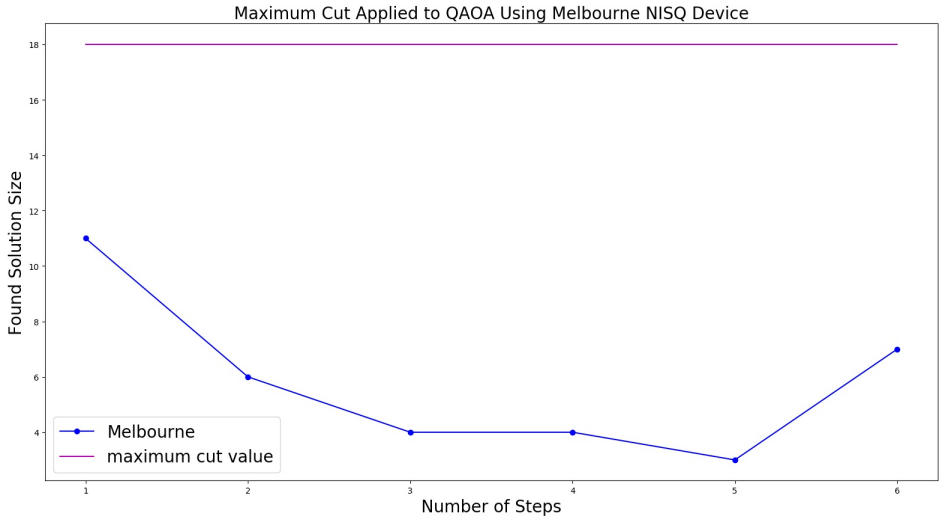
[10] Gavin E Crooks. Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem. pages 15–17, 2018.

[11] Andrew W. Cross, Lev S. Bishop, John A. Smolin, and Jay M. Gambetta. Open Quantum Assembly Language. 7 2017.

[12] Hristo Djidjev, Guillaume Chapuis, Georg Hahn, and Guillaume Rizk. Efficient Combinatorial Optimization Using Quantum Annealing. Technical report, 2016.

[13] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. 11 2014.

Figure 11: Melbourne NISQ Device



[14] Phil Goddard, Susan Mniszewski, Florian Neukart, Scott Pakin, and Steve Reinhardt. How Will Early Quantum Computing Benefit Computational Methods? Technical report.

[15] G G Guerreschi and A Y Matsuura. QAOA for Max-Cut requires hundreds of qubits for quantum speed-up. pages 1–14, 2018.

[16] Richard M. Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. 2010.

[17] Samuel J Lomonaco. A LECTURE ON SHOR'S QUANTUM FACTORING ALGORITHM VERSION 1.1. Technical report, 2000.

[18] Andrew Lucas. Ising formulations of many NP problems. 2 2013.

[19] Technology Overview. The D-Wave 2000Q$^{\mathrm{TM}}$ Quantum Computer. Technical report.

[20] Robert S Smith, Michael J Curtis, and William J Zeng. A Practical Quantum Instruction Set Architecture. Technical report.

[21] Hayato Ushijima-Mwesigwa, Christian F A Negre, and Susan M Mniszewski. Graph Partitioning using Quantum Annealing on the D-Wave System. Technical report.

[22] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Physical Review A*, 2018.