Investigation of the Optimal Solution to the MaxCut Problem With Different Depth p

Je-Yu (Leo) Chou

April 2021

Abstract

Quantum computers are a new technology that may be able to bring revolutionary computational powers to certain problems. However, in addition to examining pure quantum algorithms, research on algorithms that are resistant to noise are also important. This paper provides an overview upon quantum computation and quantum approximate optimization algorithm (QAOA), and understands how different depths affect the energy predicted by both an ideal simulation as well as on a noisy simulator with T1 and T2 noise modelling off a real quantum computer. For the ideal simulation, the general trend is indeed upwards, though different layer will have differences within them. For the noisy simulator, there is not a trend to be seen, likely due to the build up of noise on each layer. However, the author did notice that the ideal simulation does calculate a higher energy than the noisy simulator across most layers. This is expected as noise within a quantum system causes states to change, thus affecting the sampling rate of bitstrings.

Contents

1 Introduction

Although computational power has increased exponentially in recent years, there are certain problems that are just very hard to solve [27, 25]. More precisely, there are problems that are easy to check but hard to find the solution to, such as some variation of the satisfiability problem (SAT Problem), which are known to be NP-hard or NP-complete [6] [1]. One such problem is the Maximum cut problem, or MAXCUT for short, where the

[1] This is assuming that P≠NP, which is the general consensus within the field

objective is to partition a graph such that the cut can have the nodes of the graph to be a maximum value (in other words, find the cut such that the edges between the two disjoint groups have the maximum value)[5]. These type of problems called combintorial optimization problem, where the number of solution increases combinatorically, which imposes a huge challenge as these problems often have profound implication in our every-day life [30]. It is important to note that, as it is an optimization algorithm, the objective is to find a solution that is *close enough* to the actual solution that maximizes the goal function. With this in mind, in 2014, Edward Farhi and Jeffrey Goldstone published the first paper on quantum approximate optimization algorithm, where they utlized the adiabatic theorem (more on section 3.2) as well as trotterization to approximate the max energy state (it is crucial to note that when first preparing the initial state $|s\rangle$, we make it so that it is at the highest energy state, thus by adiabatically evolving the schrodinger equation, the energy of the system does not change, thus getting the maximum energy value, corresponding to the approximate maximum value) [8]. In this paper, I examine how the QAOA algorithm perform with different depths, and compares the runtime and the performance in finding the maximum cut. As the graph is small, the runtime for the program should not be too long, and can be checked by running through all possible cuts and finding the optimal bitstring.

1.1 Research Question

How well does QAOA find the optimal solution to the MaxCut problem with different depths p on an IBM quantum simulator [16] [2]?

1.2 Hypothesis

Given that the primary way of running it is through an ideal simulator, I do expect an increase in p to lead to an increase in the final result; this, however, will not hold if noise is introduced as seen from literature on the noisy nature of quantum computing.

2 Background Information

2.1 Problem Statement and Formulation of the Hamiltionian

2.1.1 Problem Statement

The MAXCUT problem can be formulated both as a decision problem as well as an optimization problem, where the decision problem has been shown by Karp that it is NP-complete, while the optimization problem has been shown to be NP-hard [18]. The original decision problem as stated in Karp is as follows:

Instance Given a graph G with vertex V and edges E such that G(V,E), and the associated weight such that w(e) $\in \mathbb{Z}^+$, and e $\in$ E, positive integer K

[2] I acknowledge the use of IBM Quantum services for this work. The views expressed are those of the author, and do not reflect the official policy or position of IBM or the IBM Quantum team.

Problem Is there a partition of V into disjoint sets of V_1 and V_2 such that the sum of the weights of the edges from E that have one endpoint in V_1 and one endpoint in V_2 is at least K?
It is important here to note that the decision problem can be formulated into an optimization problem, where we find the maximum value of K.

Defining the Cost Function For any optimization problem, it is important to define a cost function such that we minimize or maximize the possible value. For the MaxCut problem, we first find the associated weight matrix, w, such that it corresponds to the weight of graph G. To find the specific weight associated with the edge of the graph, we find the matrix element for node x_i and x_j, thus w_{ij}. Further, the only way for the weight to contribute, x_i and x_j must be different, as x_i and x_j are binary variables (taking on values of 0 or 1). Thus, we get the cost function:

$$C = \sum_{i,j=1}^{n} w_{ij} x_i (1 - x_j) \tag{1}$$

2.1.2 Formulation of a QUBO instance

To formulate the Hamiltonian, we first look at constructing the cost function as a quadratic unconstrained binary optimization problem, or QUBO for short. These problems have a quadratic objective function with no specific constraint upon the variable x, where x $\in \mathbb{B}$ (x is a binary variable). Specifically, it is common to write a QUBO instance as the following:

$$f(x) = x^T Q x + c^T x \tag{2}$$

Where x is often a column vector representing the bit string, and x^T represents the conjugate transpose of the vector (transforming it to a row vector) [11]. For the MaxCut problem, we can transform it to fit the QUBO instance with the following steps. We first expand equation 1 to obtain:

$$C = \sum_{i,j=1}^{n} w_{ij} x_i - w_{ij} x_i x_j \tag{3}$$

Here, it is trivial to see that this is similar to the form as described in equation two. To further assist in simplifying and obtaining the Hamiltonian, as shown in the next section, we define two new variables, c_i and Q_{ij}, and we impose that:

$$c_i = \sum_{j=1}^{n} w_{ij} \tag{4}$$

which is the sum of the rows of the weight matrix, and

$$Q_{ij} = -w_{ij} \tag{5}$$

With this, we can thus formulate the MaxCut problem as the following QUBO instance:

$$C(x) = \sum_{i,j=1}^{n} x_i Q_{ij} x_j + \sum_{i=1}^{n} C_i x_i \tag{6}$$

2.1.3 Formulation to a Hamiltonian

As previously mentioned, we will map the cost function into a Hamiltonian, or the total energy of a quantum state. Luckily, as demonstrated through multiple papers, QUBO instances can map easily onto an Ising model, thus providing us a good computational state to work with [20]. Although we can theoretically choose any arbitrary computational basis, for the ease of measure, we will choose to use the Puali Z operator to act on quantum state $|x\rangle$, for more information on the derivation as well as motivation for the Puali Z operator, visit Appendix A. To formulate the QUBO instance into a Hamiltonian, we want the following expression to hold true:

$$H_c |x\rangle = C(x) |x\rangle \tag{7}$$

Where $H_c |x\rangle$ represents the total energy of quantum state $|x\rangle$.
The following presents the formulation of the Hamiltonian, for more information, please visit Appendix A. Z represents the Pauli Z operator.

$$H_c = \sum_{i,j=1}^{n} \frac{1}{4} Q_{ij} Z_i Z_j - \sum_{i=1}^{n} \frac{1}{2}(c_i + \sum_{j=1}^{n} Q_{ij}) Z_i + (\sum_{i,j=1}^{n} \frac{Q_{ij}}{4} + \sum_{i=1}^{n} \frac{c_i}{2}) \tag{8}$$

2.2 Quantum Computers

Quantum computers have been envisioned since the 1980s by Richard Feynman [9]. Unlike traditional computers, quantum computers harvest the power of quantum mechanics to solve problems quickly, namely superposition, entanglement, and interference [23]. Below will explore more of the quantum mechanics behind quantum computers and its mathematical formulation[3]. Below explore the construction of a quantum computers, and its mathematical formulation. In section 2.3, the basics of quantum mechanics will be used to explain the concept behind QAOA, including variational method as well as adiabatic time evolution.

2.2.1 Physics of Quantum Computers

Quantum computers, unlike classical computers, harvest the power of quantum mechanics to speed up certain calculation[4]. In particular, principles of superposition, amplitude amplification, and entanglement, are elements that are believed to be able to demonstrate quantum supremacy [19]. Below presents a brief overview of quantum mechanics, though it is by no means a comprehensive overview. For more information, please reference ref. [12].

Superposition and Amplitudes As seen from Young's double slit experiment, electrons exhibit wavelike properties. In 1926, Erwin Schrodinger proposed the Schrondinger Equation, which was enourmously successful in predicting non-relativistic particles [22]. This means that, unlike the classical world, probability is inherint within quantum mechanics, as proven by Bell's Theorem [22]. For the purpose of quantum computing, we

[3]There are a lot of different notations, however, Dirac notation will primarily be used. For more information, please visit ref. [23]

[4]It is crucial to note that experts believe only certain problems, such as the integer factorization problem, can be solved efficiently, other NP-complete problems, such as the infamous Travelling Salesman Problem, is currently not believed to be able to experience a speedup with quantum computers

can simply say that a quantum state can be represented with two orthogonal basis vectors, thus an arbitrary quantum state is a linear combination of two orthogonal states. Theorectially, any pair of orthogonal vectors could form a computational basis, however, the basis vectors are often taken as the eigenvectors of the Pauli matrices. In particular, the eigenvector of Puali Z operator is the most common used one. Thus, we can represent a quantum state as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{9}$$

Where $|\psi\rangle$ is an arbitrary quantum state; $\alpha, \beta \in \mathbb{C}$, representing probability amplitudes (implying $|\alpha|^2$ is the probability to measure state $|0\rangle$, and vice versa). As α and β are probability amplitude, it follows that $|\alpha|^2 + |\beta|^2 = 1$. Furthermore, as quantum systems can often be seen as waves, it is also possible to interfere, amplifying the desired amplitudes. Grover's algorithm is thus a perfect example of the importance of amplitude amplification [13].

Entanglement Quantum Entanglement has been dubbed 'spooky action at a distance' by Einstein, and is the subject of several paper (collectively known as EPR paradox) [7]. This is the heart of quantum computation, as it is one of the truly quantum properties that classical mechanics do not describe. Quantum entanglement describes the fact that when we measure a physical feature of one particle, the measurement can have an effect on its entangled particle (sometimes a perfect correlation). In quantum computing, bell states describes four maximally entangled states, and is often used in quantum algorithm as it is at the core of a quantum speedup. Application of quantum entanglement can be seen from superdense coding and quantum teleportation[5] [2].

2.2.2 Basics of Quantum Computation

Basis States In quantum computation, we often use the Z-axis as the basis of computation, thus we get state $|0\rangle$ and state $|1\rangle$, represented by two column vectors:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{10}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{11}$$

Measurements When measuring the probability that a state $|\psi\rangle$ will measure in state $|x\rangle$ (such as the $|0\rangle$ state), we can express the probability as the square of the absolute value of the inner product (dot product in a hilbert space) between the conjugate transpose of $|x\rangle$ and state $|\psi\rangle$, thus:

$$p(|x\rangle) = |\langle x|\psi\rangle|^2 \tag{12}$$

It is important to note that measurement in quantum mechanics will have an effect on the system. Specifically, any measurement made on a state already in superposition will cause the wavefunction to collapse and the particle will choose to be in one and only one state only.

[5]It is worth noting that quantum teleportation does not enable information to travel faster than light, as seen from the no-go theorem in physics

Bloch Sphere It is often useful to represent visually a quantum state, thus we use a bloch sphere. Below presents the process of mapping a state onto a bloch sphere

Global Phase Consider the following quantum state: $(0,\ i)$[6]. It can be seen as:

$$i \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{13}$$

To see the probability to measuring the state $|x\rangle$, we can plug in equation 12 to get the following:

$$\begin{aligned} p(|x\rangle) &= |\langle x|(i|1)\rangle| \\ &= |i \cdot \langle x|1\rangle|^2 \\ &= |\langle x|1\rangle|^2 \end{aligned} \tag{14}$$

As seen, the global phase in this case is i, and it has no effect upon the measurement. In fact, this is true for all global phase, only local phase (phase that is applied to only one state) is measureable [2].

Constructing the Bloch Sphere With the realization from above that global phase does not matter in the context of quantum computation, we can begin our construction of the bloch sphere.

As we can only measure the difference in phase between state $|0\rangle$ and $|1\rangle$, we can constrain α and β to real numbers and add a local phase to one of the terms (it is standard to add it to $|1\rangle$), thus getting the following equation:

$$|\psi\rangle = \alpha\,|0\rangle\, e^{i\phi}\beta\,|1\rangle \tag{15}$$

where $\alpha, \beta, \phi \in \mathbb{R}$. As seen before, the state must be normalized, thus:

$$\sqrt{\alpha^2 + \beta^2} = 1 \tag{16}$$

As we are trying to plot it onto a sphere, we will represent α and β with angle θ (it is standard to divide the angle by 2 as we are plotting it onto a sphere), resulting the following:

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \tag{17}$$

with $\theta, \phi \in \mathbb{R}$. We can easily verify this as $\sqrt{\sin^2 x + \cos^2 x} = 1$ for all $x \in \mathbb{R}$.

Plotting on the Bloch Sphere A bloch sphere is a sphere existing on a complex plane with radius of one, and it can visually represent a quantum state (it cannot display multiple at once). As all quantum states lie on the surface, we only need three coordinates to specify the point on the sphere, and we define them as follows:

$$\vec{r} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix} \tag{18}$$

A visual representation of the bloch sphere is shown in appendix B.

[6]Although written similar to a row vector, it should be taken as a column vector as it represents a quantum state

Logic gates in Quantum Computing Gates are essential to computation, as they provide a way to transform the state into the desired output. In quantum computation, gates simply perform a transformation on the quantum state. One crucial thing to note about quantum gates are that they are reversible, or unitary. Thus, the following must be true: $U^\dagger U = I$ where U is a unitary matrix, $U^\dagger$ is the adjoint of U (taken by transposing U then taking the complex conjugate of the elements), and I is the identity matrix [23]. Although there are a lot of gates in quantum computation, only the X gate, hadamard gate, and CNOT gates are introduced (other gates are rotational gates). For a more comprehensive overview, please reference [23].

The X gate is analagous to the NOT gate in classical computing, namely, it transforms state $|0\rangle$ to state $|1\rangle$ and vice versa. It has the following matrix representation (σ_x and σ_1 are common notation for this gate also):

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{19}$$

The hadamard gate puts a qubit into equal superposition. The state actually forms the eigenvector of the Pauli-X matrix, thus it is also a valid form of computational basis, called $|+\rangle$. The state can be represented as:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{20}$$

when applied to state $|0\rangle$ and

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{21}$$

when applied to state $|1\rangle$. The Hadamard matrix is:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{22}$$

A CNOT gate is a two qubit gate that can perform a controlled X gate, thus if the control qubit is $|0\rangle$, it will do nothing; and if the control qubit is $|1\rangle$, then the target qubit will be applied with a X gate. the CNOT gate has the following matrix (as it is a 2-qubit gate, it is a 4×4 matrix):

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{23}$$

2.3 Quantum Approximate Optimization Algorithm (QAOA)

QAOA is an algorithm proposed for the Noisy Intermediate-Scale Quantum (NISQ) era, where quantum computers are still noisy and the machines are limited in size (less than 100 qubits). Below presents the algorithms and how it works [24].

2.3.1 Variational Quantum Algorithms and the NISQ Era

QAOA is an algorithm proposed in the NISQ era, where we are unable to harvest the full power of quantum system due to noisy qubits [24]. This led to a whole class of algorithm known as variational quantum algorithms from the variational principle in quantum mechanics [4].

A Hamiltonian is a Hermitian operator that corresponds to the total energy of the system. As a Hamiltonian is a hermitian matrix, it must have real value eigenvalues. We can express a hamiltonian operator as the sum of the outer-product (tensor product) of its eigenvectors, ψ_i weighted by its eigenvalue, λ_i, expressed as the following:

$$H = \sum_{i=1}^{N} \lambda_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \tag{24}$$

To get the expectation value of the Hamiltonian of a particular state ψ, we can represent it as the following:

$$< H >_{\psi} = \langle\psi|H|\psi\rangle \tag{25}$$

We can thus substitute equation 25 into equation 24, and get the following:

$$\begin{aligned} < H >_{\psi} &= \langle\psi| \left(\sum_{i=1}^{N} \lambda_i \left|\psi_i\right\rangle \left\langle\psi_i\right|\right) |\psi\rangle \\ &= \sum_{i=1}^{N} \lambda_i \left\langle\psi|\psi_i\right\rangle \left\langle\psi|\psi_i\right\rangle \\ &= \sum_{i=1}^{N} \lambda_i \left\langle\psi|\psi_i\right\rangle^2 \end{aligned} \tag{26}$$

From this, it is clear to see that λ_i must have a positive value as the inner product between ψ and ψ_i, thus it is clear that $\lambda_i \leq \langle\psi|H|\psi\rangle$. From this, we can set some parameter that we optimize such that we can get the minimum energy (getting minimum and maximum energy can be viewed as the same in this case) [2].

As one can see, this algorithm is unlike some traiditional quantum algorithm such as Shor's Algorithm or the original Deutsch-Jozsa Algorithm where only quantum computers are used to solve the particular problem, instead, it requires preprocessing to find the best angle or classical optimization [28, 26, 4]. This is because quantum computers are still prone to errors from outside noises (thus the reason why many superconducting quantum computers are cools to near absolute zero), and the size cannot get large enough to solve complex problems (even though Shor's Algorithm promises to factor large numbers, the largest factored today on a quantum computer is 21) [21]. Variational quantum algorithms utilize the power of classical optimization with algorithms such as gradient descent and the power of quantum computers, the ability to calculate the energy of a quantum state, to solve a problem.

2.3.2 Constructing the Mixer Hamiltonian

When viewing the algorithm, it is clear to see that we apply two rotational gates. These gates encode the cost Hamiltonian and the mixer hamiltonian (H_B), defined as the

following:

$$H_B = \sum_{i=1}^{N} \sigma_i^x \tag{27}$$

where σ^x is the Puali-x operator. One thing to note is that the mixer hamiltonian does not have to be this exact form, but it is written as the sum of all Pauli-X operator for the convienience [8]. The mixer Hamiltonian is constructed this way as it does not commute with the cost Hamiltonian (for two operators that do not commute, $AB \neq BA$) [8]. To understand why a non-commuting operator must be applied, we first understand the dangers of local minima. When optimizing the energy of a certain system, it often presents with multiple local minima, and this often confuses an algorithm to not escape it, and thus unable to find the global minima. If we only apply the cost hamiltonian or with a mixer hamiltonian that commutes with the cost hamiltonian, by basic linear algebra, the two operator will only dilate the eigenvector, not change its direction. With a non-commuting operator, it allows the roation of the eigenvector, thus able to better escape a local minima.

2.3.3 Evolution of a Quantum State

We now have all the pieces to construct the circuit. However, we still have not figure about how to encode both the cost Hamiltonian and the mixer Hamiltonian as gates yet. We first understand how to describe an evolving system using the time-dependent schrodinger's equation:

$$i\hbar \frac{\partial}{\partial t} |\Psi(t)\rangle = H |\Psi(t)\rangle \tag{28}$$

Here, i is the imaginary number, and $\hbar$ is planck's constant divided by 2π (this is sometimes omitted for natural units). This equation describes how a system's energy (denoted by the H on the right hand side of the equation) will evolve through time t. Solving the equation, we get:

$$|\Psi(t)\rangle = e^{-iHt/\hbar} |\Psi(0)\rangle \tag{29}$$

where $|\Psi(0)\rangle$ denotes the inital starting energy of the system. Seeing how we can solve the Schrodinger's Equation to see how a quantum state evolve, we can thus construct our gates. For more information on the exponentiation of matrices, visit appendix C.

2.3.4 Constructing Unitary Gates

As seen previously, to understand how a system evolve, we solve the schrodinger equation and get a solution to exponentiate the Hamiltonian with some parameter. Here, we define γ as the parameter for the cost hamiltonian and β for the mixer hamiltonian [8]. Below presents the result after exponentiating the matrix:

$$U_c(\gamma, H_C) = e^{-i\gamma H_c} = \prod_{i,j=1}^{N} R_{Z_i Z_j}(\frac{1}{4} Q_{ij}\gamma) \prod_{i=1}^{N} R_{Z_i}(\frac{1}{2}(c_i + \sum_{j=1}^{N} Q_{ij})\gamma) \tag{30}$$

Here, R_Z refers to rotational Z gate, and R_{ZZ} refer to controlled Z gates (similar to CNOT gate except applied with rotation of Z)

Below presents the result of the mixer hamiltonian:

$$U_B(\beta, H_B) = e^{-i\beta H_B} = \prod_{i=1}^{N} R_x(2\beta) \tag{31}$$

Where R_x is a rotational X gate, and this result can be seen from appendix C.

2.3.5 Algorithm

Below presents how the algorithm works, note that for most quantum system, it is initialized to be at state $|0\rangle$. We first create an equal superposition by applying a hadamard gate to every single qubit, create state $|+\rangle$. Below presents the equation:

$$|+\rangle^{\otimes n} = \sum_{x \in 0,1^n}^{n} \frac{1}{\sqrt{2^n}} |x\rangle \tag{32}$$

where $\otimes$ denotes the tensor product (also known as outer product).

We then apply $U_C(\gamma_i, H_C)$ and $U_B(\beta_i, H_B)$ alternatively for p times (p is the depth), we then measure the final state.

3 Variables

3.1 Independent Variable

Depth p Mathematically, this algorithm is guranteed to reach the optimal solution as p approaches infinity. This, however, is not done due to the complexity as well as the noise within the system. In this paper, I am interested in seeing how increasing depth p can improve our algorithm (the original paper has only test for $p = 1$).

3.2 Dependent Variable

Optimal Cut As stated before, the goal for the problem is to find the maximum cut such that the graph can be separated into two disjoint sets. By using the algorithm, it should be able to return the optimal bitstring as well as the calculate the maximum energy.

3.3 Controlled Variable

To ensure the optimal model, a simulator with Clifford noise model will be applied. One of the challenges of a quantum system is the fact that it cannot be interfered by outside noises, and a simulator best does that. The system will be calibrated, and the calibration result will be shown in the paper. Other variables that I will watch out for include the archetecture that I am using, as well as the same weight for the graph. It also worth noting that the approach is with an optimizer, instead of warm-starting the algorithm (finding the optimal value beforehand).

4 Methodology

4.1 Ethical and Environmental Concern

This experiment runs on IBM's software, thus the author has sufficient reason to believe that the IBM quantum team has examined their ethical and environmental concern and has passed their ethics board [10]. However, as a whole, I avoid using my laptop in a dangerous area and avoid extraneous runnings.

4.2 Methods

IBM's quantum simulator will be primarily used to ensure reproducible results (unlike real systems where the errors can change). To assist with this research, python library networkx will be used to create and visualize graphs [14]. Qiskit, an open-source library developed by IBM, will be used to access all necessary function to set up the problem instance as well as running on an IBM simulator [1]. A problem instance will first be created through Qiskit into a QUBO instance, then a QAOA circuit will be created. For every run, 1024 shots are sent to the simulator to understand the probability of sampling a given state. Throughout the program, a classical optimizer (cobyla) will be used to optimize the program and find the maximum cut. Finally, the graph will be plotted plot with plotly and matplotlib, both python libraries [17, 15].

4.3 MAXCUT Instance

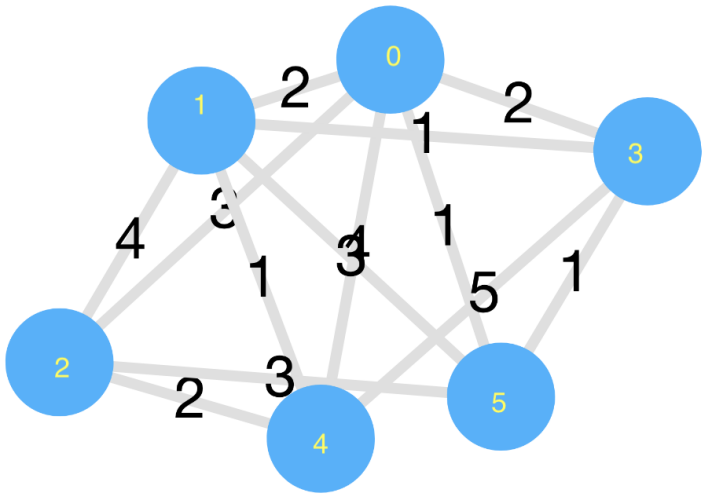


Figure 1: A picture of the MAXCUT instance, with 6 nodes and 13 edges. The weights of the edges and the node number is labelled. This graph was generated by networkx and created by the author.

4.4 Classically Solving the Problem

4.4.1 Overview

This is a very small problem instance, which means a brute force method is available. By repeatedly calculating all possible bitstrings (a total of 64 possibilities), we get Figure 2, where the x axis represents the bitstring value, and the y axis is the cut value. Notice that bitstrings 00000 and 111111 have cut values of zero as both configuration will leave the graph intact, without a cut. The maximum cut is with bitstring 010011 and 101100 (which are cut into the same two subsets) at 23. The sequence of the number is given in Figure 1.

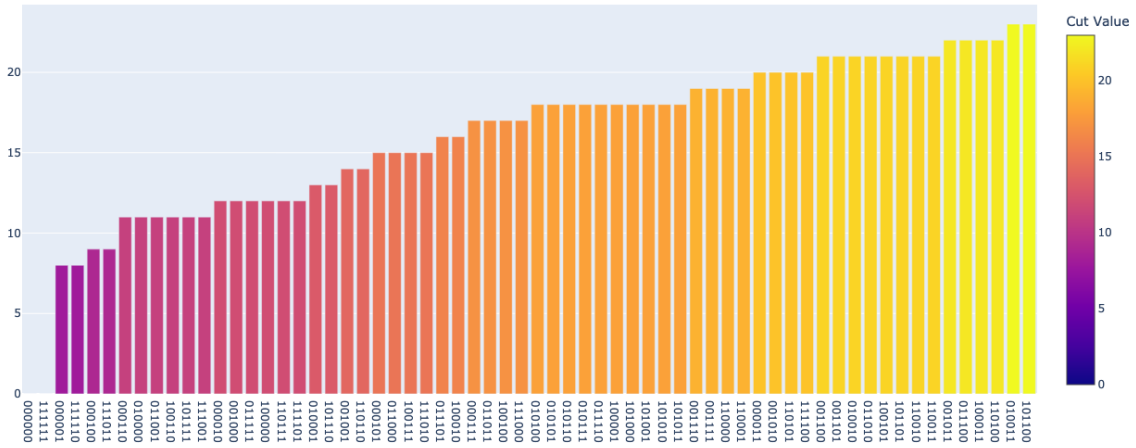


Figure 2: Bar plot graph of all possible bitstrings with a heatmap on the left. Created by the author with plotly

5 Data

5.1 Raw Data

This experiment was done through IBM's quantum simulators with multiple runs, thus there is no 'raw data'.

5.2 Obtaining Processed Data

Some data are processed through IBM, such as the native result that was ran on the simulator that includes cut value and probability. Other data, such as the average cut value (also known as average energy) was the sum of all possible bitstrings with their cut value multiplied by the probability of measuring.

5.3 Running on Regular Simulator

We first present the result on a regular simulator:

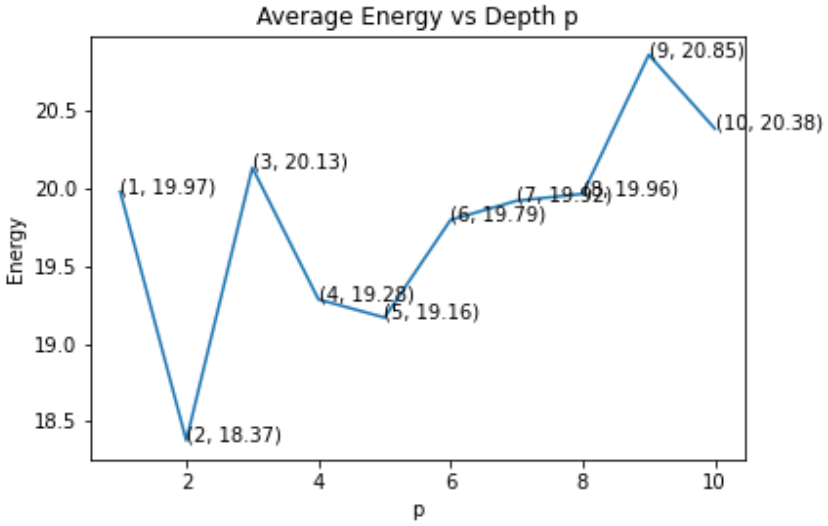


Figure 3: Graph of average cut value vs depth of p. Energy here represents the cut value, thus has no units, and p also doesn't have a unit

5.4 Running on a Noisy Simulator

Unfortunetaly, I do not have access to systems with more than five qubits, thus the noisy simulator was used to simulate the thermal errors (T1 and T2 time). T1 and T2 and sampled through a normal distribution. Both T1 and T2 are obtained from ibmq_nairobi, a quantum computer with seven qubits, 32 quantum volume with a Falcon r5.11H processor. The result was obtained on August 30th, 2021 at 4:28am.

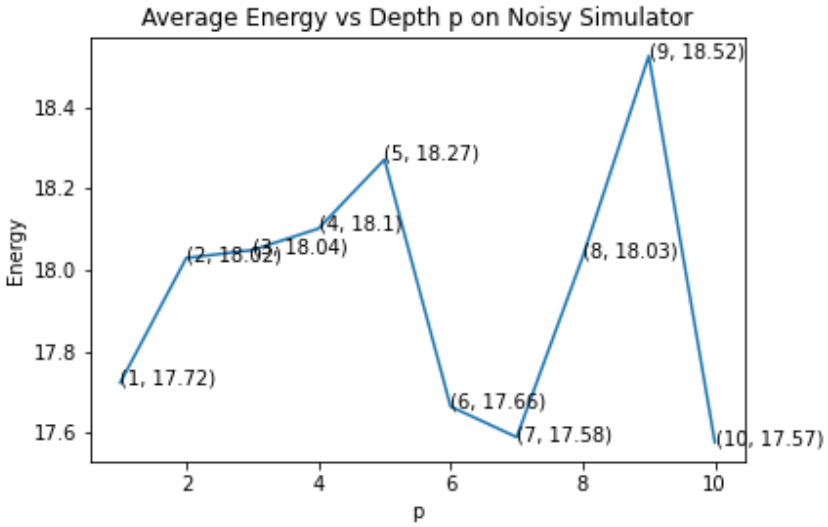


Figure 4: Graph of average cut value vs depth of p on a noisy simulator with T1 and T2 time from ibmq_nairobi. Energy here is also the cut value, thus no unit available. Depth is represented by p, with no units

6 Results

Both graphs are graphs of energy vs depth, with energy truncated to 4 significant figures. There is no error bars present as the quantum samples 1024 times, determines the probability, and return the result. The objective is to maximize the energy, such that the bitstring with the highest cut value can also have a high probability of being sampled. Below discusses the similarity and differences between the two.

6.1 Similarity

Both graphs presents a very jagged line, indicating that there is no equation that can adaquetly describe what the result is. This is not surprising as QAOA does not come with a performance gurantee.

6.2 Difference

6.2.1 Perfect Simulator

Running the experiment on the perfect simulator yielded a more predictable result. Although the graph is still very jagged with occasional dips, overall, it has an upward trend, especially from a depth of two to a depth of 9. This is expected as there is no noise, thus it fits equation 8 as described by Farhi in ref. [8]. However, it must be noted that this sort of system only guratees to find the correct solution given infinite p, thus when p is finite, there is no performance gurantee. It is also worth noting that as this is an inherintly probabalistic algorithm, the result may vary. Lastly, it is worth comparing it to figure 4. It can be seen that the general cut value is higher than the noisy simulator, also with the higher p performing better in general than lower p (except for when p=1).

6.2.2 Noisy Simulator

Figure 4 is the resultant graph on a noisy simulator simulating ibmq_nairobi of their T1 and T2 times. It is worth noting that this is not a perfect replica of the actual system, as I did not add readout error, and the basis gates are a bit different. With p=1 to p=5, due to the small size and low depth, the fidelity did not have as much of a role, thus having a general increase in cut value. However, for p greater than 5, it is almost impossible to guess the cut value. In addition to reason presented before, it is also worth noting that as depth increases, the error on each qubit and their gate also increases, making the algorithm less stable. Finally, we can also see that the general cut value found is often less than value found with the perfect simulator.

7 Conclusion and Further Research

7.1 Conclusion

Overall, despite the jagged graphs, my hypothesis was correct. For figure 3, when p=1 is taken out, the overall trend of the graph is upwards, indicating that the average cut value is higher as depth increases. When looking at figure 4, we can see that the cut value isn't as straightforward due to the compounding of errors within the system. Yet with both of these, the benefit of this algorithm remains to be seen as runtime increases

significantly, as shown in appendix D. However, it is worth noting that the graph shows a linear growth with runtime, which, if an increase in p can indeed lead to a greater cut, then the linearity in the runtime is very good. Further, the highest cut value is valued at 20.85 for p=9 in the ideal simulator, and 18.52 at p=9 for the noisy simulator, these two values are 90.7% and 80.5% for the noisy simulator, which are close to the classical limit (assuming unique games conjecture) [29]. However, it is important to keep in mind that this is a very small instance of a given problem, and whether this proves to be a useful algorithm remains to be seen.

7.2 Further Research

QAOA brings together the power of classical computing and quantum computing, and has the potential to provide a framework for combinatorial problems, especially as noise decreases and the number of qubits increases, we may be able to sample a large graph instance with multiple layers. However, the development of quantum computer are still very much at its infancy, and it is not expected to reach beyond 100 qubits within the next few years [24]. There has been several proposed ways to increase the efficiency and optmization of the algorithm, including warmstarting the algorithm, instead of simply using $|+\rangle$, which mimicks semidefinite programming in classical computing and can bring a performance gurantee on the algorithm [3]. It will be interesting in seeing the different ways of warm starting the algorithm and whether it can bring an even better performance gurantee than classical computing. Furthermore, conditional values at risk (CVaR) has been shown to provide a better convergence and solution to the algorithm, and it will interesting in seeing how changing the CVaR value (how much value should be considered) will change the final solution and how quickly it can converge, and which is the optimum value. Finally, a different mixer hamiltonian can be proposed and understand how the mixer hamiltonian can affect the energy landscape and whether a different hamiltonian rather than simply the sum of all pauli-X operators can indeed lead to an improvement.

8 Evaluation

8.1 Strength

Overall, this was an intensive research with rigorous mathematics and physics theorems to support it. By simulating it on a simulator, I can have a near-perfect quantum system which can better see whether an increase in depth can result in a better value. Further, with 1024 shots per run, and many runs per depth, the sample size is large enough to get a good probability in the system and understand which bitstring is more likely to be sampled. Further, with a small enough instance, multiple approaches (both classically and through a quantum computer) are available, giving a better idea of how the algorithm performed. Finally, with the noise model being modelled off of real-world value, it gives further insight on how a real quantum computer might perform.

8.2 Weakness

One of the major weakness within this investigation is the lack of control. As all the computational result were obtained thanks to IBM's hardware, there was little that I could control. This also leads to the lack of transparency as data were processed before

sending it over to me. Additionally, with no available way of adding a seeding value, this experiment's result can change due to the probablistic nature of the algorithm, though the general value is consistent. The lack of access to a real quantum device was also a big weakness, as many things just could not be simulated on a noise model, such as the readout error and the specific time and error rate for each gate. To improve this research, a seeding value should be provided and used to ensure reproducibility, and accessing a quantum system could be more indicative of how a real world system can perform.

8.3 Scope

The overall scope of this research shed some light upon the current status of quantum computer and how useful it will be to off load optimization task to quantum computers. Further, this was just one of many ways to approach this problem, and attempt to improve it with methods as described above.

A Derivation of the Problem Hamiltonian (equation8)

Below presents a derivation of the problem Hamilton. Recall Equation 1 and 6. As discussed in section 2.2, the quantum analogue of 0 and 1 is replaced by states $|0\rangle$ and $|1\rangle$. This is the eigenstate of the Puali Z (σ_z) matrix, which has eigenvalues of 1 and -1. When applied to both states, we yield the following result through basic matrix multiplication.

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{33}$$

and

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = (-1)\,|1\rangle \tag{34}$$

From this, one can see that the Puali Z operator will do nothing to state 0, and make the state negative (times it by negative 1) if the state is one, we can thus write the Pauli Z operator as follows:

$$Z_i\,|x\rangle = (-1)^{x_i} = 1 - 2x_i \tag{35}$$

Solving for x_i, we get the following:

$$\frac{I - 2Z_i}{2}\,|x\rangle = x_i \tag{36}$$

Where I represents the identity matrix (matrix equivilant of 1).
After obtaining equation 12, we can apply it to equation 6, getting the following:

$$H_c\,|x\rangle = \sum_{i,j=1}^{n} Q_{ij}(\frac{I - Z_i}{2})(\frac{I - Z_j}{2}) + \sum_{i=1}^{n} c_i(\frac{1 - Z_i}{2}) \tag{37}$$

By rearranging term, we can obtain equation 6.

B Visual Representation of the Bloch Sphere

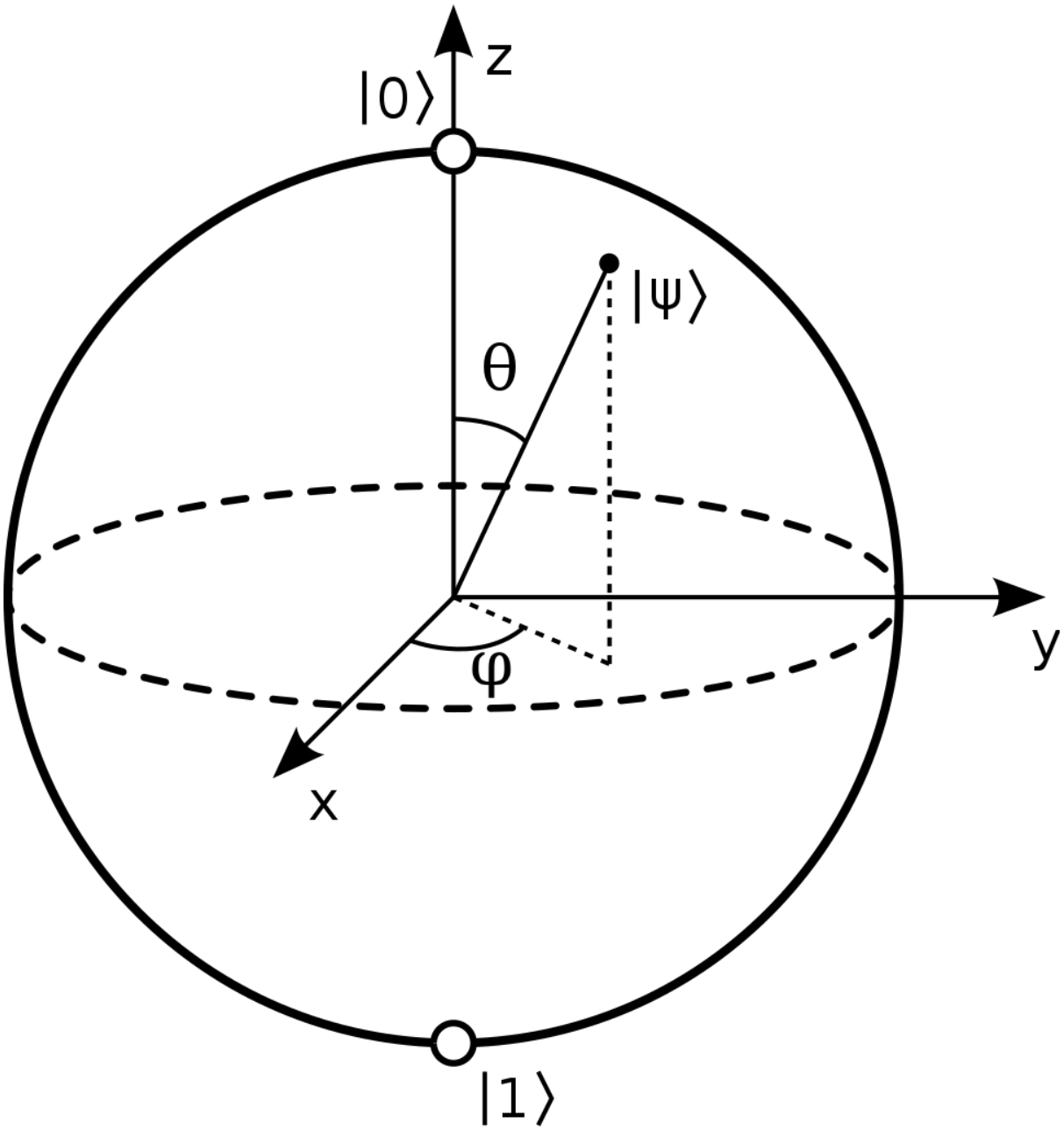


Figure 5: The Bloch sphere, a geometric representation of a two-level quantum system, illustrated by Smite-Meister, distributed under CC BY-SA 3.0 license

C Exponentiation of a Matrix

As seen from solving the schrodinger equation, we get an exponentiation of a matrix, H. This section is dedicated to the understanding of it.

We first understand Taylor series, where a function can be expressed as an infinite sum of polynomials:

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{38}$$

Specifically, if we center the function around zero, we get the Maclaurin series:

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}x^k \tag{39}$$

An application of this is seen in the Maclaurin series of e^x, where e is the natural number:

$$e^x = \sum_{k=0}^{\infty} \frac{1}{k!}x^k \tag{40}$$

We can further extend this for M, where M is a matrix. Below presents the exponentiated version of the Pauli Matrices.

Recall:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{41}$$

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{42}$$

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{43}$$

When we exponentiate these matrices with parameter θ, we get the following rotational gates:

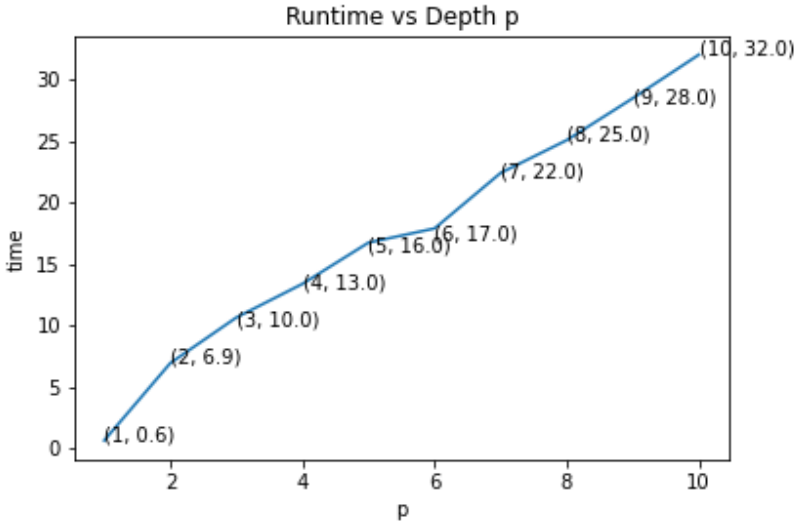
$$\begin{aligned} e^{-i\theta\sigma_x} &= R_x(2\theta) \\ &= \begin{bmatrix} \cos(\theta) & -i\sin(\theta) \\ -i\sin(\theta) & \cos(\theta) \end{bmatrix} \end{aligned} \tag{44}$$

$$\begin{aligned} e^{-i\theta\sigma_y} &= R_y(2\theta) \\ &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \end{aligned} \tag{45}$$

$$\begin{aligned} e^{-i\theta\sigma_z} &= R_z(2\theta) \\ &= \begin{bmatrix} e^{-i\theta} & 0 \\ 0 & e^{i\theta} \end{bmatrix} \end{aligned} \tag{46}$$

D Runtime of Program

Below presents the runtime of the program as depth increases:



References

[1] Héctor Abraham et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2019. DOI: 10.5281/zenodo.2562110.

[2] Abraham Asfaw et al. *Learn Quantum Computation Using Qiskit*. 2020. URL: http://community.qiskit.org/textbook.

[3] Lennart Bittel and Martin Kliesch. *Training variational quantum algorithms is NP-hard – even for logarithmically many qubits and free fermionic systems*. 2021. arXiv: 2101.07267 [quant-ph].

[4] M. Cerezo et al. *Variational Quantum Algorithms*. 2020. arXiv: 2012.09265 [quant-ph].

[5] Clayton W. Commander. "Maximum cut problem, MAX-CUTMaximum Cut Problem, MAX-CUT". In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Boston, MA: Springer US, 2009, pp. 1991–1999. ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0_358. URL: https://doi.org/10.1007/978-0-387-74759-0_358.

[6] Stephen Cook. "The P versus NP problem". In: *Clay Mathematical Institute; The Millennium Prize Problem*. 2000.

[7] A. Einstein, B. Podolsky, and N. Rosen. "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?" In: *Physical Review* 47.10 (1935), pp. 777–780. DOI: 10.1103/physrev.47.777.

[8] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm". In: *arXiv preprint arXiv:1411.4028* (2014).

[9] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6-7 (1982), pp. 467–488. DOI: 10.1007/bf02650179.

[10] Patrick Gibbons. *Notre Dame, IBM launch Tech Ethics Lab to tackle the ethical implications of technology*. June 30, 2020. URL: https://news.nd.edu/news/notre-dame-ibm-launch-tech-ethics-lab-to-tackle-the-ethical-implications-of-technology/.

[11] Fred Glover, Gary Kochenberger, and Yu Du. *A Tutorial on Formulating and Using QUBO Models*. 2019. arXiv: 1811.11538 [cs.DS].

[12] David Griffiths. *Introduction to Quantum Mechanics*. Cambridge, United Kingdom: Cambridge University Press, 2017.

[13] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866. URL: https://doi.org/10.1145/237814.237866.

[14] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[15] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[16] *IBM Quantum Experience*. 2021. URL: quantum-computing.ibm.com.

[17] Plotly Technologies Inc. *Collaborative data science*. 2015. URL: https://plot.ly.

[18] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations* (1972), pp. 85–103. DOI: 10.1007/978-1-4684-2001-2\{_}9.

[19] Amit Katwala. *Quantum computing and quantum supremacy, explained*. Mar. 2020. URL: https://www.wired.co.uk/article/quantum-computing-explained#:%7E:text=Right%20now%2C%20the%20best%20quantum,re%20powerful%2C%20but%20not%20reliable..

[20] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* 2 (2014). DOI: 10.3389/fphy.2014.00005.

[21] Enrique Martín-López et al. "Experimental realization of Shor's quantum factoring algorithm using qubit recycling". In: *Nature Photonics* 6.11 (2012), pp. 773–776. DOI: 10.1038/nphoton.2012.259.

[22] D. Morin. "Chapter 10 Introduction to quantum mechanics". In: 2010.

[23] Michael Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 1st ed. Cambridge University Press, 2011.

[24] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79. DOI: 10.22331/q-2018-08-06-79.

[25] R. Pruim and I. Wegener. *Complexity Theory: Exploring the Limits of Efficient Algorithms*. Berlin Heidelberg: Springer, 2005. ISBN: 9783540210450. URL: https://books.google.com/books?id=1fo7_KoFUPsC.

[26] David Deutsch Richard Jozsa. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558. DOI: 10.1098/rspa.1992.0167.

[27] R.R. Schaller. "Moore's law: past, present and future". In: *IEEE Spectrum* 34.6 (1997), pp. 52–59. DOI: 10.1109/6.591665.

[28] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/s0097539795293172.

[29] Luca Trevisan. "On Khot's unique games conjecture". In: *Bulletin of the American Mathematical Society* 49.1 (2012), pp. 91–111. DOI: 10.1090/s0273-0979-2011-01361-1.

[30] G. Yu. *Industrial Applications of Combinatorial Optimization*. Applied Optimization. Springer US, 1998. ISBN: 9780792350736. URL: https://books.google.com.tw/books?id=GuYeAQAAIAAJ.