Quantum approximate optimization algorithms and applications

Author : Giannis D. Leonidas

Supervising Professor : Demosthenes Ellinas

Examination Committee : Professor Aggelos Bletsas

Examination Committee : Professor Dimitrios Angelakis

Technical University of Crete, School of Electrical and Computer Engineer

October 15, 2021

Contents

Acknowledgments

I want to thanks my professor and mentor Prof Angelakis for his persistence and time that he gave me, not only for this thesis. I want to thanks my professor Prof Bletsas for his guidance in the estimation theory and the optimization problems. I want to express my sincerest gratitude to Benjamin Tan and Alex Dukakis from CQT Singapore for their very helpful notes and the talks that we had during my thesis. I want to thanks all my friends who were there in all the excitements and all the drama during my undergraduate years. I want to thanks my family for their unconditional support throughout the years. In the memory of my grandfather Ioannis.

Abstract

In this thesis we start by defining the basic components that bring together a quantum circuit. We explain basic gates, the concept of entanglement and why these are important for the construction of quantum algorithms. Then we proceed by analyzing two basic quantum algorithms (Deutsch-Josza and Grover's algorithms), which are the earliest in quantum computing and illustrate the notion of a quantum speed up. Next, we analyse the basic approaches for quantum optimization, including the notions of quantum annealing and adiabatic quantum computing, and analyze the first main algorithm of this thesis which is the Quantum Approximate Optimization Algorithm (QAOA). We also introduce and explain the Variational Quantum Eigensolver (VQE) and its hardware efficient version, which does not require specific gates decomposition and compare it with QAOA on the context of solving MAXCUT problems. In the final part, we analyze QAOA on a more industrial optimization setting, and solve instances of the Tail Assignment Problem for assigning planes in different routes. For this problem we test QAOA using the conventional method of minimizing the expectation value of the cost Hamiltonian and discuss the results. Finally, we also apply solve the problem by an another method based on minimizing the Gibbs objective function where we see improvements in the success probability. We analyse the inner workings of the algorithms, discuss the results and compare the various methods for different problem sizes and instances. We run our quantum algorithms in simulators, with noise and ideal ones, as well as on and prototype quantum hardware available in the cloud in IBM Q and analyze the performance for different problem size and qubit numbers.

Chapter 1

Introduction

Quantum computing uses certain properties of quantum mechanics such as quantum superposition and entanglement to perform calculations. The first model for a quantum computer was proposed by Richard Feynman in 1981, at a conference held by MIT. In 1992 David Deutsch and Richard Josza proposed the Deutsch-Josza algorithm [1]. The problem of boolean functions that this quantum algorithm solves is the first example where a quantum algorithm surpasses the classical algorithm in computational efficiency. Two years after, Peter Shor created the very well known Shor's algorithm [2] that solves the integer factorization problem. This quantum algorithm reduced the time complexity from exponential to polynomial time. The integer factorization problem is the foundation of the RSA encryption which is used vastly nowadays and Shor's algorithm solves it in polynomial time. However, Shor's algorithm require a lot of qubits in order to factor large numbers and quantum hardware nowdays comprises of up to 70-100 qubits. In 1996, Lov Grover create the Grover's algorithm for unstructured search [3] which has a quadratic speedup over the classical algorithms.

Implementation wise, in 1998 Isaac Chuang (Los Alamos National Laboratory), Neil Gershenfeld (MIT) and Mark Kubinec (University of California, Berkeley) created the first Nuclear Magnetic Resonance (NMR) Quantum Computer consisting of 2 qubits and it was used to solve the Deutsch's problem. In the past 23 years quantum computing has a very active research area and quantum computers have been realized with the recent demonstrations of quantum supremacy by Google

using 53 superconducting qubits. At this day there exists several companies that owns quantum computers , some of which are freely accessed by the public. There exists many technologies that are being used in order to create a quantum computer, such as trapped ions, superconducting qubits, quantum dots.

Except gate based quantum computers non universal quantum computers called quantum annealers. Specifically there exists quantum annealers with 5000 qubits and they are used in order to solve optimization problems which will be the focus of this thesis. Their qubits have certain topology i.e. they allow certain interactions between qubits, so they depend heavily on the problem that they solve. In top of that quantum annealing requires a lot of time in order to find the optimal solution to our problem.

Thesis outline

In this thesis we start by defining the basic components that bring together a quantum circuit. We explain basic gates, the concept of entanglement and why these are important for the construction of quantum algorithms. Then we proceed by analyzing two basic quantum algorithms (Deutsch-Josza and the Grover's algorithms), which are the entry gate to quantum computing. In chapter 3 we explain the theory behind quantum optimization and explain the main concept of this thesis which is the Quantum Approximate Optimization Algorithm (QAOA). Also we explain the Variational Quantum Eigensolver (VQE) and compare it with QAOA on the context of MAXCUT problem. In chapter 4 we analyze QAOA on a more industrial optimization problem, the Tail Assignment Problem. For this problem we test QAOA using the conventional method of minimizing the expectation value of the cost Hamiltonian. After this we test QAOA by minimizing the Gibbs objective function where we see improvements in the success probability.

Chapter 2

Introduction to quantum computing

In this chapter we will set the backbone of quantum computing. First we start by the definition of the qubit and the single and two qubit gates. Then we introduce entanglement between two qubits, a powerful property of qubits. At last we combine all of the above concepts to introduce two basic quantum algorithms, the Deutsch-Josza and the Grover's algorithm. These algorithms were among the first to provide a significant speed up over their classical counterparts and for this reason is important to understand how these algorithms work before we move to the more advanced cases of quantum optimization which is the core of this thesis later on.

2.1 What is a qubit?

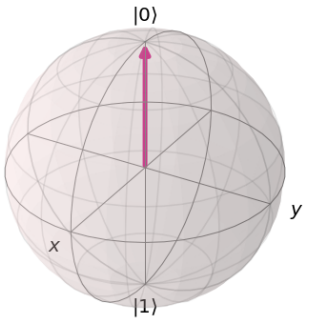
From a computer scientist's perspective, we can think of qubit as the fundamental information that is used in quantum computers just like the bit in the classical computers. With risking oversimplifying, we could say that the main difference between qubits and bits is that the latter is deterministic. That is, a bit can be either '0' or '1'. On the other hand a qubit can be in a linear combination of '0' and '1' called superposition. We should note that a qubit is not equivalent with

a bit that has some probabilities to be '0' or '1', the quantum superposition is terms of quantum amplitudes that can also destructively interfere. On other words, quantum probabilities can also cancel each other.

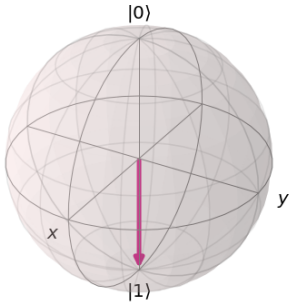
Another suitable way of thinking for qubits is the spins of the electrons. In general the spin of the electron has a direction which allows us to encode qubit states into these directions and as vectors, they can be added or substracted. Another important aspect, is the measurement. In reference of the Schrödinger's experiment with the cat in the box, as long as we don't open the box and see the condition of the cat that is inside we can say that the cat is both dead and alive or she is in a superposition of these states. The moment we open the box and see the cat, then we will have a deterministic state, either alive or dead. Exactly as the experiment above, when we measure the state of a qubit we say that it's state collapses to one state and the superposition is lost. We refer the reader to Nielsen and Chuang textbook [4] for a more detailed description of the physics, here we will stay on the basics of the necessary maths behind quantum information, which is the math of complex vector spaces. We will use bra-ket notation. In the simplest case a qubit can have two states. The set of possible states is $\{|0\rangle, |1\rangle\}$ where,

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

are called the base vectors because they represent an orthonormal base for our qubits . For a base to be orthonormal the base vectors must be orthogonal to each other (their inner product is zero) which can be shown easily, $\langle 0|1\rangle = \langle 1|0\rangle = 0$, and their magnitudes must be equal to one , $\langle 0|0\rangle = \langle 1|1\rangle = 1$.



(a) Visualization of a qubit in the state $|0\rangle$.



(b) Visualization of a qubit in the state $|1\rangle$.

Figure 2.1: Bloch sphere representation of one qubit.

A superposition in that states space can be written as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \quad \alpha, \beta \in \mathbb{C}$$

where

- $|\alpha|^2 + |\beta|^2 = 1$, $\Pr\big(|0\rangle\big) = |\alpha|^2$ and $\Pr\big(|1\rangle\big) = |\beta|^2$ are the probabilities of finding $|\psi\rangle$ in the $|0\rangle$ state and the $|1\rangle$ state respectively

To visualize the above state we need 4 real numbers, because $\alpha, \beta \in \mathbb{C}$. In the (figure) above we see the visualization of quantum states in the Bloch sphere. The Bloch sphere is a 3-parameter real (x,y,z) space which represents all the possible states-vectors for a qubit using rotations. We can get from 4 real numbers to 3 by expressing the quantum state

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

Note that the above is a valid quantum state since

$$|\cos\left(\frac{\theta}{2}\right)|^2 + |e^{i\phi}\sin\left(\frac{\theta}{2}\right)|^2 = 1$$

By further expanding the quantum state we have

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + (\cos\phi + i\sin\phi)\sin\left(\frac{\theta}{2}\right)|1\rangle$$

where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. Note that the above representation is always unique. Now we can convert this Cartesian coordinates (θ, ϕ) to spherical coordinates by using

$$x = \sin\theta\cos\phi \quad y = \sin\theta\sin\phi \quad z = \cos\theta$$

where we have assumed that radius $r = 1$ because all the quantum states that belong in the Bloch sphere have $|| \, |\psi\rangle \, ||^2 = 1$.

We can view the above (superposition) in the orthonormal base $\{|0\rangle, |1\rangle\}$ as follows

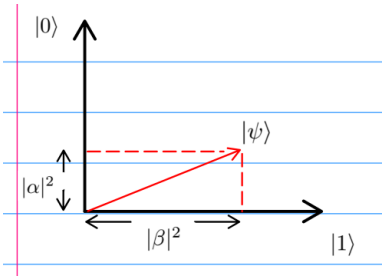


Figure 2.2: Visualization of the superposition $|\psi\rangle$ inside the orthonormal base $\{|0\rangle, |1\rangle\}$.

2.2 Single qubit gates

Now we present some of the basic tools that are needed for the manipulation of quantum states. Single qubit gates are unitary matrices that acts on a single qubit and produces a single qubit output. We start by the Pauli gates

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

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

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

Note that the above gates are unitary and have the below property

$$X^2 = Y^2 = Z^2 = -iXYZ = \mathbb{I}$$

The action of each gate is shown below

- X gate $X|0\rangle = |1\rangle \quad X|1\rangle = |0\rangle$
- Y gate $Y|0\rangle = i|1\rangle \quad Y|1\rangle = -i|0\rangle$
- Z gate $Z|0\rangle = |0\rangle \quad Z|1\rangle = -|1\rangle$

It is also very important to note that $Z|0\rangle = (-1)^0|0\rangle$ and $Z|1\rangle = (-1)^1|1\rangle$. By writting the action of the Z gate with this way we are able to get the state 0 or 1 in front of the ket like a phase. This is used in many algorithms included quantum optimization ones, and Deutsch-Josza algorithm. In general we have $Z|x\rangle = (-1)^x|x\rangle$

Some other important gates are the Hadamard gate which creates the superposition of a qubit and the phase shift gate which adds a phase to our qubit. The Hadamard gate as a unitary matrix

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

The action of the Hadamard gate onto the basis states

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

The phase shift gate

$$P(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

The action of the phase shift gate is to produce a phase shift of angle ϕ or $e^{i\phi}$ when our qubit is in the state $|1\rangle$ and do nothing if our qubit is in the state $|0\rangle$

$$P(\phi)\,|0\rangle = |0\rangle \quad P(\phi)\,|1\rangle = e^{i\phi}\,|1\rangle$$

Note here that this phase is not measurable. This means that the amplitude of $|0\rangle$,$|1\rangle$ is left unchanged. In other words the probability of measuring either $|0\rangle$ or $|1\rangle$ is the same after the appliance of the $P(\phi)$ gate. We can see that if we write

$$e^{i\phi} = \cos\phi + i\sin\phi$$

Because of the Bloch sphere representation of qubits, we must also note the rotation operators. These operators take as inputs an angle θ and rotate the qubit state around the respective axis by that angle.

Rotation around the X axis

$$R_X(\theta) \equiv e^{-i\frac{\theta}{2}X} = \cos\frac{\theta}{2}\mathbb{I} - i\sin\frac{\theta}{2}X = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_X(\theta)\,|0\rangle = \cos\frac{\theta}{2}\,|0\rangle - i\sin\frac{\theta}{2}\,|1\rangle$$

$$R_X(\theta)\,|1\rangle = \cos\frac{\theta}{2}\,|1\rangle - i\sin\frac{\theta}{2}\,|0\rangle$$

Rotation around the Y axis

$$R_Y(\theta) \equiv e^{-i\frac{\theta}{2}Y} = \cos\frac{\theta}{2}\mathbb{I} - i\sin\frac{\theta}{2}Y = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ -\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_Y(\theta)\,|0\rangle = \cos\frac{\theta}{2}\,|0\rangle + \sin\frac{\theta}{2}\,|1\rangle$$

$$R_Y(\theta)\,|1\rangle = \cos\frac{\theta}{2}\,|1\rangle - \sin\frac{\theta}{2}\,|0\rangle$$

Rotation around the Z axis

$$R_Z(\theta) \equiv e^{-i\frac{\theta}{2}Z} = \cos\frac{\theta}{2}\mathbb{I} - i\sin\frac{\theta}{2}Z = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$

$$R_Z(\theta)\,|0\rangle = \cos\frac{\theta}{2}\,|0\rangle - i\sin\frac{\theta}{2}\,|0\rangle = (\cos\frac{\theta}{2} - i\sin\frac{\theta}{2})\,|0\rangle$$

$$R_Z(\theta)\,|1\rangle = \cos\frac{\theta}{2}\,|1\rangle + i\sin\frac{\theta}{2}\,|1\rangle = (\cos\frac{\theta}{2} + i\sin\frac{\theta}{2})\,|1\rangle$$

2.3 Two qubit gates

Now that we can represent and manipulate a single qubit, we can move forward and explore the multi qubit space. The space of all the available states for 2 qubits is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ which tell us that there are four possible states for a 2 qubit system. For a 3 qubit system the available state space is $\{|000\rangle, |001\rangle, |010\rangle, \ldots, |111\rangle\}$. Each new qubit we add to our system multiply the available states by 2. That is, if we have n qubits the available states will be 2^n and the available state space will be all the different bit strings of length n : $\{\underbrace{|00\ldots0\rangle}_{n}, \ldots, |11\ldots1\rangle\}$

Let's take the system of 2 qubits, x and y, where $n = 2$. Then we will have the space $\Omega = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ .We write this space in a more compact form using the tensor product

operator. That is $\Omega = \{\underbrace{|0\rangle, |1\rangle}_{x}\} \otimes \{\underbrace{|0\rangle, |1\rangle}_{y}\}$.

No we can define the basis vectors for Ω which is a space with length equal to 4. In a strictly mathematical manner we see the elements that Ω consists of and we can do the calculations. We have

$$|u_0\rangle = |00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|u_1\rangle = |01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|u_2\rangle = |10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|u_3\rangle = |11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

From the above we see a beautiful symmetry on the statevectors spanned from Ω. This is happening because we want our base to be orthonormal i.e.

$$\langle u_i | u_j \rangle = \begin{cases} 0 & , i \neq j \\ 1 & , i = j \end{cases}$$

Now that we have our 2 qubit orthonormal base we can define a superposition in this base

$$|\psi\rangle = \alpha_0 |00\rangle + \alpha_1 |01\rangle + \alpha_2 |10\rangle + \alpha_3 |11\rangle$$

where $|\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2 = 1$ are the probabilities to find our system in the appropriate state. That is $\Pr(|11\rangle) = |\langle 00|\psi\rangle|^2 = |\alpha_3|^2$ (Born rule)

A very useful gate that takes two inputs and has two outputs is the Controlled Not gate. With respect to our basis Ω, the matrix representation of the CNOT gate is

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

This gate has two qubits as the input, one serves as the select qubit and one as the target qubit. If the select qubit is $|0\rangle$ then we do nothing to the target qubit. If the select qubit is $|1\rangle$ then we apply a X gate to the target qubit.

If we denote as $|s\rangle$ the selector qubit and $|t\rangle$ the target qubit then we see that

$$\text{CNOT}\,|s, t\rangle = |s, s \oplus t\rangle$$

where the truth table for the XOR gate is

- $0 \oplus 0 = 0$
- $0 \oplus 1 = 1$
- $1 \oplus 0 = 1$

- $1 \oplus 1 = 0$

With this in hand we can see the action of the CNOT more closely

- $\text{CNOT} \left|00\right\rangle = \left|00\right\rangle$
- $\text{CNOT} \left|01\right\rangle = \left|01\right\rangle$
- $\text{CNOT} \left|10\right\rangle = \left|11\right\rangle$
- $\text{CNOT} \left|11\right\rangle = \left|10\right\rangle$

So if we apply the CNOT gate onto the 2 qubit state $\left|\psi\right\rangle$ we will have the amplitudes of $\left|10\right\rangle$ and $\left|11\right\rangle$ swapped.

$$\text{CNOT} \left|\psi\right\rangle = \alpha_0 \left|00\right\rangle + \alpha_1 \left|01\right\rangle + \alpha_2 \left|11\right\rangle + \alpha_3 \left|10\right\rangle$$

We must also note that we can create a general control U gate, that has the same principle with the CNOT gate but it applies the U gate when the selector qubit is $\left|1\right\rangle$. The way we can construct such a gate is to say that :

Suppose U is

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$$

We want this gate to act on a state and map that state as follows

$$\left|00\right\rangle \mapsto \left|00\right\rangle \quad \left|01\right\rangle \mapsto \left|01\right\rangle \quad \left|10\right\rangle \mapsto \left|1\right\rangle \otimes U \left|0\right\rangle \quad \left|11\right\rangle \mapsto \left|1\right\rangle \otimes U \left|1\right\rangle$$

We see that

$$U \left|0\right\rangle = \begin{pmatrix} u_{00} \\ u_{10} \end{pmatrix} \quad U \left|1\right\rangle = \begin{pmatrix} u_{01} \\ u_{11} \end{pmatrix}$$

Now that we know the action of U onto it's base states we can define the Controlled U gate or CU for short as follows

$$CU = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{pmatrix}$$

where we can see that the upper left 2x2 elements denote the unitary $\mathbb{I}$ matrix which is the part that the selector do nothing to the target qubit and the lower right 2x2 elements denote the matrix U for the case that we want to apply U to our target state.

2.4 Entanglement

Einstein characterized entanglement as a spooky action at a distance. Entanglement is a property that two particles-qubits can have after they interact with each other even if they are very far apart.

To better understand the concept of entanglement, let's see how we can create entanglement between two qubits. All we need in order to create entanglement are a Hadamard and a Control Not gate.

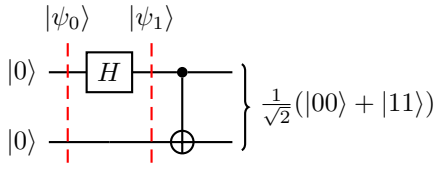


Figure 2.3: Quantum circuit for Bell's state.

We start with the state $|\psi_0\rangle = |00\rangle$. Then we apply a Hadamard to the first qubit while applying unity to the second qubit (e.g do nothing on the 2nd qubit).

$$(H \otimes \mathbb{I})\,|00\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) = |\psi_1\rangle$$

Now we apply a CNOT gate using the first qubit as control and the second as the target

$$\text{CNOT}\,|\psi_2\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Suppose that the first qubit is a random variable X and the second qubit is a random variable Y. Then we can write the output state as $\frac{1}{\sqrt{2}}(|0_X 0_Y\rangle + |1_X 1_Y\rangle)$. Now we can define the mutual information between these two random variables as the amount of information for X given that we measure Y.

We can define this mutual information as

$$I(X;Y) = H(X) - H(X|Y)$$

where

$$H(X) = -\sum_{X=x} P(X=x)\log P(X=x) = \frac{1}{2}$$

is the uncertainty for the random variable X or the entropy of X and $H(X|Y)$ is the uncertainty we have about X given that we observed Y.

Notice that at the start of the circuit these two random variables are uncorrelated, so $I(X;Y) = 0$.

For the output state we have that

i) If we measure Y to be 0, then X must be 0

ii) If we measure Y to be 1, then X must be 1

We can see that $I(X;Y) = H(X) = \frac{1}{2}$ since $H(X|Y) = 0$, because if we measure Y we are certain about the outcome of X. This means that our qubits started as uncorrelated but at the end of the circuit there exists information between them and therefore they are correlated. This tells us that we can use entanglement to share information about qubits which is useful for many quantum communication protocols in which sharing information between qubits is needed. Examples of such cases are teleportation and superdense coding. In the later, we can communicate two bits of

information by only sending one qubit physically while sharing an entangled state in advance.

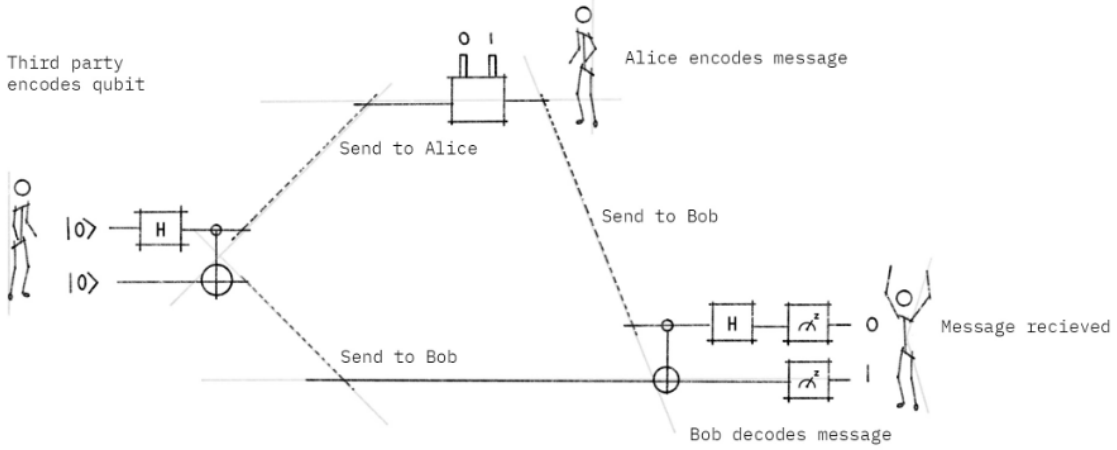


Figure 2.4: Superdense coding protocol. A third party runs the bell circuit and produces state $|\beta_{00}\rangle$ which encodes the classical bit string '00' or '11' as information. Then he sends one qubit to Alice and one qubit to Bob, wherever they are. Then Alice (the sender of the message) encodes the message and sends her qubit to Bob. At last, Bob performs a CNOT followed by a Hadamard gate and then he measures his qubits. At the end he will have the 2bit message that Alice sent.

The example we used above is based on the first of the four Bell's states. These states are maximally entangled and can be produced using the (circuit) above but varying the inputs i.e start with $|00\rangle$ for first Bell state, $|01\rangle$ for the second Bell state, $|10\rangle$ for the third Bell state and $|11\rangle$ for the fourth Bell state.

A concrete way for representing the 4 Bell states is the following

$$\beta_{xy} = \frac{1}{\sqrt{2}}(|0y\rangle + (-1)^x |1\bar{y}\rangle)$$

where $x \in \{0,1\}, \quad y \in \{0,1\}$ and $\bar{y}$ is the negation of y.

2.5 Deutsch-Josza algorithm for boolean functions

We will now exploit another interesting property of quantum computers called quantum parallelism. This is a property that come to us through the superposition of qubits. When we have a

superposition of n qubits we can instantaneously manipulate all the 2^n possible states.

We will see that quantum parallelism give us a computational advantage over classical computers. In order to see why this interference of qubits is helpful we will present the Deutsch-Josza algorithm. It was created by David Deutsch and Richard Josza in 1992. This algorithm is the first quantum algorithms that created and has proven speed up over it's classical counterpart.

2.5.1 The problem

We start with a boolean function with n inputs and a boolean output, $f : \{0,1\}^n \to \{0,1\}$. This means that f has for input a bit string of length n e.g. $\underbrace{110\ldots000}_{n}$. Assume that one of this functions is guaranteed to be constant or balanced (i.e. has an equal number of 0's and 1's). The goal of this algorithm is to determine whether the aforementioned function is constant or balanced.

A classical computer will need to evaluate at best half the inputs plus the next one in order to be certain that the function is constant or balanced. The worst case scenario for a classical computer will require $O(n)$. The Deutsch-Josza algorithm solves this problem with a single evaluation of f.

2.5.2 The oracle

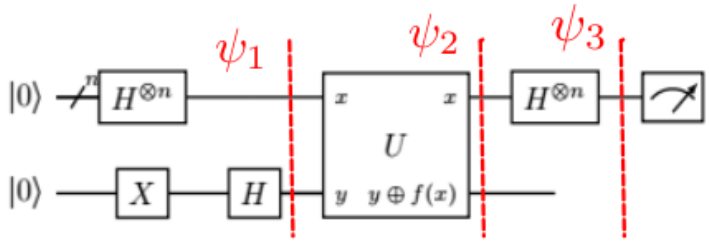


Figure 2.5: Deutsch-Josza quantum circuit.

In the figure (above) we see the quantum circuit for the Deutsch-Josza algorithm. The new gate-operator we see here is called the oracle (noted as U) and it is a $2^{n+1} \times 2^{n+1}$ matrix that acts on $n+1$ qubits. We can think of the oracle as a blackbox that we have to use in order to pass

information to all qubits simultaneously, and that information will be the value of $f(x)$ for each x. Because we want the oracle to be a valid gate for the circuit it must be represented by a Unitary matrix. We define the action of oracle to be

$$U\left|x,y\right\rangle = \left|x, y \oplus f(x)\right\rangle$$

Now if we recall the action of the CNOT gate in a state

$$CNOT\left|s,t\right\rangle = \left|s, s \oplus t\right\rangle$$

We can see that the action of oracle can be realized as a control operation where now the control is a function of our qubits rather that the qubits themselves.

Now notice that in the quantum circuit we have a NOT gate followed by a Hadamard acting on the last qubit (first two gates of the second qubit). That is

$$HX\left|0\right\rangle = H\left|1\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle) = y$$

The key here is that we start y in this state , so that

$$y \oplus f(x) = \frac{1}{\sqrt{2}}(\left|0 \oplus f(x)\right\rangle - \left|1 \oplus f(x)\right\rangle) = \frac{1}{\sqrt{2}}(-1)^{f(x)}(\left|0\right\rangle - \left|1\right\rangle)$$

which can be shown by taking $f(x) = 0$ and $f(x) = 1$.

2.5.3 The algorithm

The Deutsch-Josza algorithm consists of five steps

(i) We start with $n+1$ qubits where we mark the first n qubits as x and the last qubit as y. All qubits are initialized in the all zero state and then we apply a NOT gate to qubit y.

(ii) We apply a NOT gate to qubit y and then we apply Hadamard gates to all qubits.

(iii) We act with the oracle U on all our qubits.

(iv) We apply Hadamard gates to qubits x .

(v) We measure the qubits x.

In order to understand this algorithm we will solve the quantum circuit by executing steps i) to v).

(i) Preparation step

$$|\psi_0\rangle = \underbrace{|0\rangle^{\otimes n} |1\rangle}_{n+1}$$

(ii) Apply Hadamards to the $n+1$ qubits

$$(H^{\otimes n} \otimes H) |\psi_0\rangle = H^{\otimes n} |0\rangle^{\otimes n} \otimes H |1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) =$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} |x\rangle (|0\rangle - |1\rangle) = |\psi_1\rangle$$

(iii) Apply the oracle on $|\psi_1\rangle$

The oracle acts on a state as

$$U |x, y\rangle = \big|x, y \oplus f(x)\big\rangle$$

$$U |\psi_1\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} |x\rangle \left(\big|0 \oplus f(x)\big\rangle - \big|1 \oplus f(x)\big\rangle\right) = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle) = |\psi_2\rangle$$

(iv) Apply Hadamard gates to the first n qubits

From now we will leave the last qubit out of the total wavefunction since it won't be measured,

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle$$

$$H^{\otimes n}|\psi_2\rangle = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle = \frac{1}{2^n}\sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\cdot y}|y\rangle = |\psi_3\rangle$$

where

$$x\cdot y = x_0y_0 \oplus x_1y_1 \oplus \ldots \oplus x_{n-1}y_{n-1}$$

Note that the above bitwise product will produce a 1 output when we have odd number of $x_iy_i = 1$, otherwise it will produce 0.

(v) Measure the first n qubits

Now we want to measure the first n qubits, so let's find the probability of measuring the all 0 state $|0\rangle^{\otimes n}$ from $|\psi_3\rangle$

$$Pr(\underbrace{00\ldots000}_{n}) = |\langle 0|^{\otimes n}|\psi_3\rangle|^2 = \Big|\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\Big|^2$$

where now we only have the internal summation over the x because we projected $|\psi_3\rangle$ onto the all 0 state and because of the orthonormal basis of our n qubits we have exactly 1 term of the summation over y equal to 1, namely $\langle 00\ldots000|00\ldots000\rangle = 1, \qquad \langle 00\ldots000|00\ldots001\rangle = 0.$

Now we have two possibilities, either the function $f(x)$ is constant or balanced, so we can fix $f(x)$ and check the probability again. That is

$$Pr(00\ldots000) = \begin{cases} |\frac{1}{2^n}2^n|^2 = 1 & \text{if f(x) is constant} \\ |\frac{1}{2^n}(\frac{n}{2}-\frac{n}{2})|^2 = 0 & \text{if f(x) is balanced} \end{cases}$$

where we used that $\big|\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\big| = 2^n$ if $f(x)$ is constant and $\big|\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\big| = \frac{n}{2}-\frac{n}{2}$ if $f(x)$ is balanced.

We see that by measuring at the end of the quantum circuit we expect two outcomes. If we measure and find the all 0 state then we are certain that $f(x)$ is constant. If we measure and don't find the all 0 state then we are certain that $f(x)$ is balanced. Notice that we run the quantum

algorithm exactly 1 time and we can be certain about a property of some function $f(x)$, while the classical algorithm would need $2^n - 1$ evaluations of $f(x)$.

2.6 Grover's algorithm for database searching

2.6.1 The problem

Suppose that we have the name of a doctor but we are missing his/her number. So we want to search a database of all the doctors in the area to find the doctor with the same name and obtain his number. Suppose that this database has N elements-doctors. The classical computer will need to search each of these doctor's names and compare with the one we have. This solution will give us complexity O(N) since we would have to search all the elements of the database in the worst case scenario. Grover's algorithm give us a squared root speedup which means that it has O($\sqrt{N}$) complexity.

2.6.2 The oracle

We want to construct an algorithm which is independent of the database in work. For this reason we will assume that the total elements of the database is $N = 2^n$ and we will focus on the indices of this elements $0 \leq i \leq N - 1$, where i is the integer index of the elements. We also want a way of determining if the element in question is actually the element we are looking for. For this reason we need a function f which takes as input an index x in the range of 0 to $N - 1$. We can then say that $f(x) = 1$ iff x is a solution to the search problem and $f(x) = 0$ iff x is not a solution to the search problem.

Now that we have the function $f(x)$ which determines whether an element is a solution to our problem or not, we take it for granted that we have an oracle that does exactly this work. The next step is to figure out a way that we take the answer of this oracle, while leaving the indices unchanged for further calculations e.g the search. For this we need an extra ancilla or oracle qubit. We can then describe the oracle as a unitary operator O

$$|x\rangle |q\rangle \xrightarrow{O} |x\rangle \left|q \oplus f(x)\right\rangle$$

where $|x\rangle$ is the index register and $|q\rangle$ is the oracle qubit. We can say now that by preparing $|x\rangle |0\rangle$ and applying the oracle O, if the oracle qubit has been flipped then x is a solution to our problem.

It is more convenient for us to encode this $f(x)$ in the phase of the appropriate state. We do this because we don't have to measure to obtain $f(x)$, because if we measure a quantum state then the protocol has ended. So we can use that

$$|x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \xrightarrow{O} (-1)^{f(x)} |x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

To go one step further we can say that since the state of the second qubit has not changed before and after the appliance of the oracle O, we can for simplicity not write it in the total wavefunction. So overall we have that

$$|x\rangle \xrightarrow{O} (-1)^{f(x)} |x\rangle$$

So we have successfully mark the correct solution to our problem since we will get a negative phase when $f(x) = 1$, but keep in mind that this phase is not relevant. This means that if we measure, the probability of finding this state have not changed. Ideally we would want this probability, after some modifications, to be equal to 1 and so measure the correct solution to our problem.

2.6.3 The algorithm

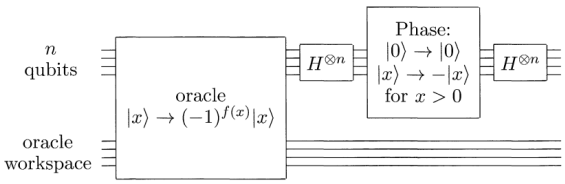


Figure 2.6: Grover operator. This is a subroutine of the main algorithm.

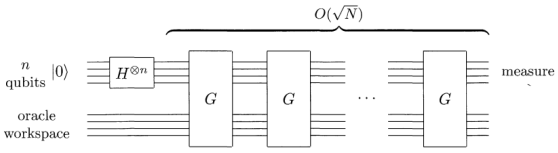


Figure 2.7: Quantum search algorithm. Here we apply the Grover operator after we have put the n qubits in superposition using the Hadamard gate in all of them.

The algorithm starts with the quantum computer in the state $|0\rangle^{\otimes n}$. Then we use the Hadamard transform to put the quantum computer in the equal superposition state

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

Then the algorithm makes use of a subroutine called Grover iteration or Grover operator which we denote as G. This subroutine consists of 4 steps

- (1) Apply the oracle O
- (2) Apply $H^{\otimes n}$
- (3) Perform a conditional phase shift with every index except 0 receiving a phase shift of -1
- (4) Apply $H^{\otimes n}$

Note that the conditional phase shift can be represented by the delta function as follows

$$|x\rangle \mapsto -(-1)^{\delta_{x0}} |x\rangle$$

where

$$\delta_{x0} = \begin{cases} 0 & , x \neq 0 \\ 1 & , x = 0 \end{cases}$$

This mapping indeed produces a minus 1 phase when the state $|x\rangle$ is not the 0 state. But when we are writing a circuit all the components must be unitaries so we must represent this conditional phase shift as a unitary. This unitary is

$$U = 2\,|0\rangle\,\langle 0| - \mathbb{I}$$

We can verify the above since

$$\textbf{if } x = 0 \rightarrow (2\,|0\rangle\,\langle 0| - \mathbb{I})\,|0\rangle = |0\rangle$$

$$\textbf{if } x \neq 0 \rightarrow (2\,|0\rangle\,\langle 0| - \mathbb{I})\,|x\rangle = -\,|x\rangle$$

So we can see that this unitary has the same effect as the conditional phase shift using the delta function. In the left (figure) we see the grover operator which consists of steps 1-4 and it is denoted by G.

We can combine steps 2,3,4 so that

$$H^{\otimes n}(2\,|0\rangle\,\langle 0| - \mathbb{I})H^{\otimes n} = 2\,|\psi\rangle\,\langle\psi| - \mathbb{I}$$

where $|\psi\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$. Thus Grover operator can be written as

$$G = (2\,|\psi\rangle\,\langle\psi| - \mathbb{I})O$$

where O is the operator for the oracle action $O\,|x\rangle \rightarrow (-1)^{f(x)}\,|x\rangle$

In the right (figure) we see the overall quantum search algorithm. By using a set of Hadamard gates and the Grover operator approximately $\sqrt{N}$ times we can find the element we are looking for.

2.6.4 Geometric visualization and complexity of Grover's algorithm

Now that we have all the components of the algorithm we can generalize the problem. Assume that we have a database with N elements and M solutions to the search problem. Let's define the

normalized states

$$|\alpha\rangle \equiv \frac{1}{\sqrt{N-M}} \sum_{x \in A} |x\rangle$$

where the sum runs over all N-M indices which are not solutions. Here A is the set of elements besides the M solutions with $|A| = N - M$

$$|\beta\rangle \equiv \frac{1}{\sqrt{M}} \sum_{x \in B} |x\rangle$$

where the sum runs over all M indices which are solutions. Here B is the set of elements that are solutions to our problem with $|B| = M$.

Then we can define the initial state of our database as

$$|\psi\rangle = \sqrt{\frac{N-M}{N}} |\alpha\rangle + \sqrt{\frac{M}{N}} |\beta\rangle$$

where we define $c = \sqrt{\frac{N-M}{N}}$ and $d = \sqrt{\frac{M}{N}}$ for convenience in the math below.

What we achieved with this reformation of our problem is that now the initial state is defined in the space spanned by $|\alpha\rangle$ and $|\beta\rangle$, the vector containing all elements besides the solutions and the vector containing the solutions to the searching problem.

Note that the oracle acting on the initial state $|\psi\rangle$ gives us

$$O(c\,|\alpha\rangle + d\,|\beta\rangle) = c\,|\alpha\rangle - d\,|\beta\rangle$$

because the oracle acts only on the vector that contains the solutions to the problem by applying a -1 phase. So after the action of the oracle on the initial state we see that only the amplitude of the solution vector has changed. In the space spanned by $|\alpha\rangle$ and $|\beta\rangle$ this is equal to a reflection of the initial state about the $|\alpha\rangle$ axis as we see below.

Similarly the remaining steps of the Grover operator which we have denoted as $2\,|\psi\rangle\,\langle\psi| - \mathbb{I}$ also

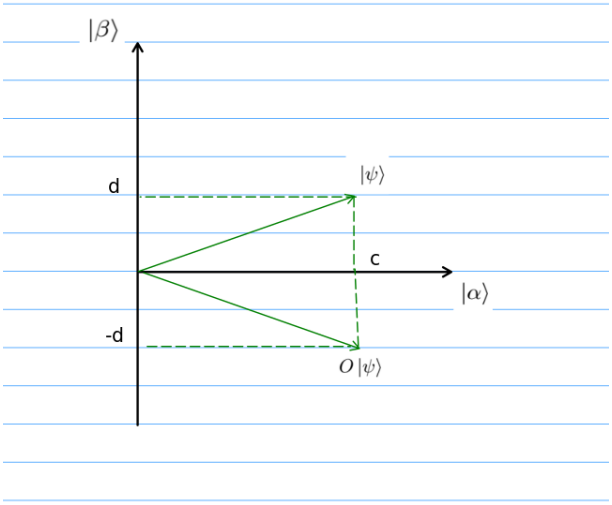


Figure 2.8: Visualization of oracle's action. We start at the state $|\psi\rangle$ and after the appliance of the oracle the state $|\psi\rangle$ has been reflected by the axis $|\alpha\rangle$.

performs a reflection. To see this let's apply this operator on a general state $\sum_k \alpha_k |k\rangle$. We have

$$(2|\psi\rangle\langle\psi| - \mathbb{I})\sum_k \alpha_k |k\rangle = \sum_k (\frac{2}{N}\sum_k |k\rangle\langle k| - \mathbb{I})\alpha_k |k\rangle = \sum_k \frac{2\alpha_k}{N}\sum_k |k\rangle - \alpha_k |k\rangle =$$

$$= \sum_k (2\sum_k \frac{\alpha_k}{N} - \alpha_k)|k\rangle \Leftrightarrow$$

$$(2|\psi\rangle\langle\psi| - \mathbb{I})\sum_k \alpha_k |k\rangle = \sum_k (2\langle\alpha\rangle - \alpha_k)|k\rangle$$

where we used that $|\psi\rangle\langle\psi| = \frac{1}{N}\sum_k |k\rangle\langle k|$ since $|\psi\rangle$ is the equal superposition of all k possible states and $\langle\alpha\rangle \equiv \sum_k \frac{\alpha_k}{N}$ is the mean value of the amplitudes of the k possible states. With this in mind we can see that the state after the action of $2|\psi\rangle\langle\psi| - \mathbb{I}$ will be in the same space spanned by $|\alpha\rangle$ and $|\beta\rangle$ but each amplitude is inverted around the state $|\psi\rangle$.

It is important to note that the result of two reflections is a rotation which tells us that if we apply the Grover operator k times the resulting state will still be in the space spanned by $|\alpha\rangle$ and $|\beta\rangle$.

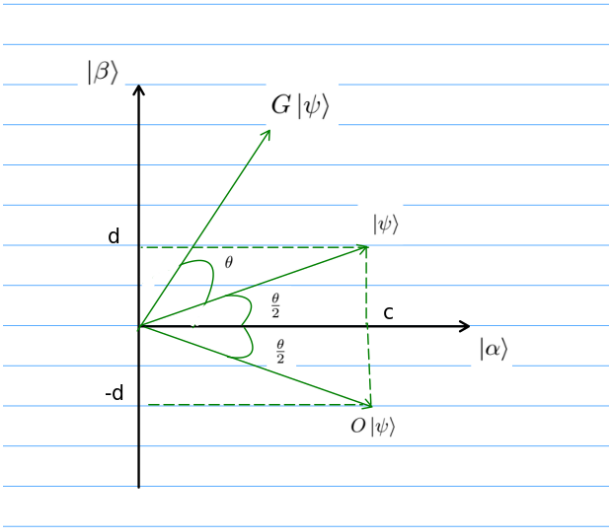


Figure 2.9: Grover operator's action. After the appliance of the oracle we are at the state $O|\psi\rangle$. Then we apply the Grover operator G and we reflect the state $O|\psi\rangle$ by the state $|\psi\rangle$. This is the key feature of the algorithm because we achieved to raise the probability of the marked element.

So if we assume that $\cos\frac{\theta}{2} = \sqrt{\frac{N-M}{N}}$ and $\sin\frac{\theta}{2} = \sqrt{\frac{M}{N}}$ then

$$|\psi\rangle = \cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle$$

Below we see the action of the Grover operator in the angle dependent state $|\psi\rangle$

As we can see above the resulting state will have an angle of $\frac{\theta}{2} + \theta = \frac{3\theta}{2}$. So after one appliance of the Grover operator we will have the state

$$G|\psi\rangle = cos\frac{3\theta}{2}|\alpha\rangle + \sin\frac{3\theta}{2}|\beta\rangle$$

which can be generalized for k applications of G.

$$G^k|\psi\rangle = \cos\Big(\frac{2k+1}{2}\theta\Big)|\alpha\rangle + \sin\Big(\frac{2k+1}{2}\theta\Big)|\beta\rangle \qquad (**)$$

The goal now is to rotate the state $|\psi\rangle$ near the state $|\beta\rangle$. We want to find the minimum

number that we are going to use the Grover operator in order to find the correct solution with high probability. We know that

$$\sin\frac{\theta}{2} \leq \frac{\theta}{2} \qquad \forall\theta \in \mathbb{R}^{+}$$

We will use the fact that $\sin\frac{\theta}{2} = \sqrt{\frac{M}{N}}$, so

$$\frac{\theta}{2} \geq \sqrt{\frac{M}{N}} \Leftrightarrow \theta \geq 2\sqrt{\frac{M}{N}}$$

The above gives us a lower bound for the angle θ. We can also obtain the maximum probability of error from the lower bound of θ.

$$\Pr(\text{error}) = \sin^2\sqrt{\frac{M}{N}}$$

To obtain a lower bound for k which is the number of Grover iterations that we need, we must see that in the ideal case we want the amplitude of $|b\rangle$ in the (**) equation to be equal to 1. So

$$k\theta = \frac{\pi}{2} \rightarrow \sin k\theta = 1 \quad \text{and} \quad \cos k\theta = 0$$

$$k\theta = \frac{\pi}{2} \Leftrightarrow k = \frac{\pi}{2\theta}$$

Now we can see that θ is on the denominator so by using the lower bound of θ we obtain an upper bound for k.

$$k = \frac{\pi}{4}\sqrt{\frac{N}{M}}$$

and therefore we can say that for a database that uses N elements and we are searching for M solutions we need to use the Grover operator $O(\sqrt{\frac{N}{M}})$ times. This gives us a quadratic speedup over the classical computer which runs in $O(\frac{N}{M})$. We can see that as $k\theta$ gets close to $\frac{\pi}{2}$ the probability of success goes to 1 because $\Pr(\text{success}) = \sin^2 k\theta$.

Chapter 3

Quantum optimization

In this section we will see how we can apply quantum algorithms to solve optimization problems. In general we have two types of quantum algorithms that are used today, the gate model algorithms and the non gate model algorithms or quantum annealers. The first runs on a quantum computer that consists of universal type gates like the Pauli matrices and is problem independent. That means that you can use the same algorithm to solve many combinatorial problems. The latter is based on the adiabatic theorem of quantum mechanics and its goal is to find a minimum energy state of a given Hamiltonian which corresponds to the minimum value of some cost function. The problem in this type of algorithms is that they are problem dependent. This dependance comes in the quantum hardware of the annealers. On top of that, the time of their execution needs to be long enough in order to find the optimal solution.

Optimization problems are the problems where we want to find the absolute best solution from a set of feasible solutions. Such problems arise in many research fields, for example in mathematics an optimization problem is to find the maximum area between two integrals, in finance an optimization problem is to maximize the income of a company while keeping the risk of firing personnel at the minimum, in physics an optimization problem is to maximize the volume of a box while knowing its dimensions but we have limited material to create it, in chemistry an optimization problem is to find the minimum energy state of a molecule.

Such problems are generally time consuming and in many cases they cannot be solved in polynomial time. For that reason approximation algorithms are introduced. These algorithms goal is to find approximations of the best solution rather than the best solution itself and it is proved that in many cases the time complexity that they use is reduced.

Optimization problems can be divided in two categories with respect to the variables that we use, if we use continuous variables then the optimization problem is called continuous and if we use discrete variables then the optimization problem is called discrete(Dimitri Bertsekas [5]). In the next section we will see a category of discrete optimization problems.

3.1 Quadratic unconstrained binary optimization

Quadratic Unconstrained Binary Optimization (QUBO) is a combinatorial optimization problem with a wide range of applications from finance to machine learning. Problems like Partition problems, Graph coloring problems, Task Allocation Problems have been translated to QUBO problems.

Let $f : \mathbb{B}^n \to \mathbb{R}$ where $\mathbb{B} = \{0, 1\}$ be a quadratic polynomial over binary variables. We have

$$f_Q(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} q_{ij} x_i x_j + \sum_{i=1}^{n} h_i x_i \tag{m.1}$$

with $x_i \in \mathbb{B}$, quadratic coefficients $q_{ij} \in \mathbb{R}$ for $1 \leq j \leq i \leq n$ and linear coefficients $h_i \in \mathbb{R}$ of the binary variables. The QUBO problem consists of finding a binary vector $x^\star$ that is minimal w.r.t. f among all other binary vectors. That is

$$x^\star = \min_{x \in \mathbb{B}^n} f(x)$$

We can also use the matrix formulation for the QUBO problem which is

$$f_Q(x) = x^T Q x$$

where $Q \in \mathbb{R}^{n \times n}$ is the symmetric $n \times n$ matrix containing the coefficients q_{ij}.

To better understand this formulation let's solve an example. Suppose you have the optimization problem

$$\text{Minimize} \quad y = -2x_1 - 3x_2 + 8x_3 + 4x_4 + 4x_1x_2 + 5x_1x_3 + 6x_2x_3 + 10x_3x_4$$

where the variables x_i are binary. We see that function y is a quadratic function with binary variables. That is we have some linear terms $-2x_1 - 3x_2 + 8x_3 + 4x_4$ and some quadratic terms $4x_1x_2 + 5x_1x_3 + 6x_2x_3 + 10x_3x_4$.

A very useful property of binary variables is that $x_i = x_i^2$, so we can write the linear terms as $-2x_1^2 - 3x_2^2 + 8x_3^2 + 4x_4^2$. We did that in order to obtain the matrix formulation of the QUBO problem

$$\text{Minimize} \quad y = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix} \begin{pmatrix} -2 & 2 & 2.5 & 0 \\ 2 & -3 & 3 & 0 \\ 2.5 & 0 & 8 & 5 \\ 0 & 3 & 5 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = x^T Q x$$

The solution of the above problem is : $y = -3$ for $x_1 = x_3 = x_4 = 0$ and $x_2 = 1$.

In general, QUBO problems represents NP-hard problems (Fred Glover et al. [6]). These problems are hard to optimize due to the fact that the quadratic function might have several local minima. This model of optimization is of great significance to the quantum computing world. That is because it has a close connection to the Ising model as we will see in the next section.

3.2 The Ising model and QUBO

Quantum computers that we have today require a bit of understanding of the underlying physics. For that reason in this section we will describe a classical model of statistical mechanics called the Classical Ising Model also used to solve QUBO problems. It is a physical model that allows us to have a lattice of spins where the neighboring pairs of spins can interact with each other. Also it

makes use of a magnetic field that allows us to control these spins.

Each spin can be though of as a magnet. If a spin has the north pole facing up we represent it with a variable $\sigma_i = 1$ and if it has the south pole facing up we represent it with $\sigma_i = -1$.

Suppose we have three magnets as shown in the figure below.

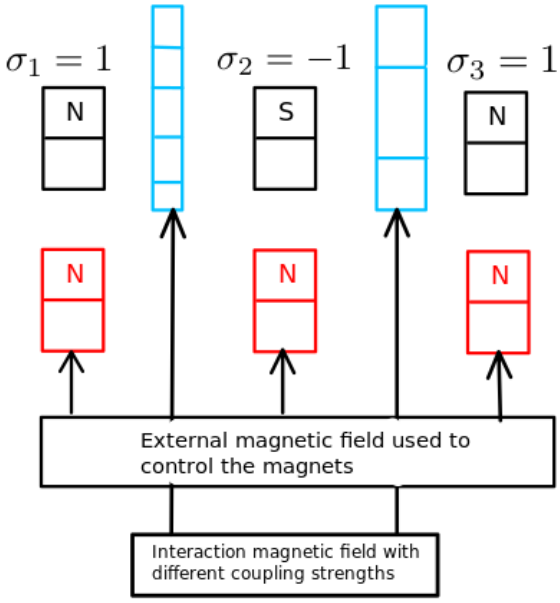


Figure 3.1: Three interacting magnets in the configuration of the Classical Ising Model

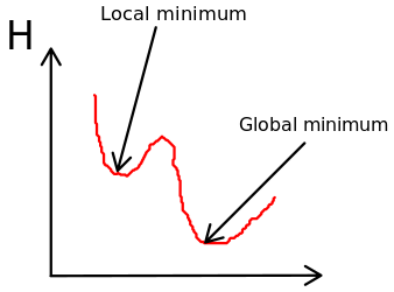
To represent the total energy or Hamiltonian of this system we write

$$H = \sigma_1\sigma_2 + \sigma_2\sigma_3 + h_1\sigma_1 + h_2\sigma_2 + h_3\sigma_3 \Leftrightarrow$$

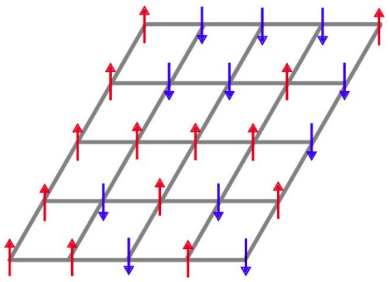
$$H = \sum_{(i,j)} J_{ij}\sigma_i\sigma_j + \sum_i h_i\sigma_i \tag{r.1}$$

where J_{ij} represents the coupling strength between spin i and j , and h_i represents the external magnetic field on spin i (see (figure)) .

In many problems that occurs in physics and in other sciences, one is interested in finding the minimum energy of a given physical system. As in the QUBO problems, the energy configuration might have local optima that will make the task of finding the true minimum energy difficult.



(a) Energy configuration of a Classical Ising Model. Here we see that there exists a local minimum that might destroy the optimization of the energy function.



(b) Configuration of the 2D Ising model on a square lattice. The arrows represent spin up (red) and spin down (blue). Picture originally created by Sascha Wald

In the statistical model we described above we have not used any quantum mechanics. Since we are interested in solving problems on a quantum computer and utilize the possibilities for quantum tunneling and quantum speed ups let's see in the next section how we can transform the Classical Ising Model into it's quantum counterpart. The quantum version of the Ising Model can be obtained with only one single change in the Ising Hamiltonian H. First, since we are talking in the quantum mechanics world, everything that we use is described by operators. Eventually this operator that we will call quantum Ising Hamiltonian, will act on some state.

The key here is to note that the variables that we have in the classical Ising Hamiltonian are the spins $\sigma_i = \pm 1$. These variables have the same values as the eigenvalues of the Z operator. Let's remember the action of the Z operator on the basis states from (Chapter 2)

$$\sigma^z |0\rangle = |0\rangle , \quad \sigma^z |1\rangle = -|1\rangle$$

With the above action we can get the values of σ_i as probability amplitudes in the states where our Hamiltonian operator will act. So the quantum version of the H from equation (r.1) is the same with the only difference being that we have σ_i^z operators instead of σ_i integers. With that being said, we have

$$H_Q = H = \sum_{(i,j)} J_{ij}\sigma_i^z\sigma_j^z + \sum_i h_i\sigma_i^z \tag{r.2}$$

as the quantum version of the Ising Hamiltonian.

Let's rewrite the QUBO model and the quantum version of the Ising model in order to compare them

$$\text{QUBO} \quad f_Q(x) = \sum_{i=1}^{n}\sum_{j=1}^{i} q_{ij}x_ix_j + \sum_{i=1}^{n} h_ix_i$$

$$\text{Quantum Ising Hamiltonian} \quad H_Q = H = \sum_{(i,j)} J_{ij}\sigma_i^z\sigma_j^z + \sum_i h_i\sigma_i^z$$

The above equations are very similar, but the first obeys laws from the classical world while the second obeys laws of the quantum world. Thankfully, there exists a mapping between these two functions that requires only two steps. In the first step we map the the binary variables $x_i \in \{0, 1\}$ to spins/magnets $s_i \in \{-1, 1\}$ using the below transformation

$$x_i = \frac{s_i + 1}{2}$$

In the second step, one must replace these spins variables with pauli Z operators. By doing so we have translated our original QUBO problem to a quantum Hamiltonian. Now we can use the quantum computer to find the lowest energy state of the quantum Hamiltonian and therefore solve the QUBO problem!

3.3 Adiabatic quantum and quantum annealers

As its name suggests, Adiabatic Quantum Optimization (AQO) is based on the Adiabatic Theorem of Quantum Mechanics.

Adiabatic Theorem of Quantum Mechanics 1 *A physical system remains in its instantaneous*

eigenstate if a given pertubation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum Max Born and Vladimir Folk 1928

This theorem tells us that if we apply some external conditions to our quantum system that change slow in time,then it will adapt its form. To understand this theorem let's see an example.

Suppose we have a Hamiltonian H_P whose ground state encodes the solution to our optimization problem and an initial Hamiltonian H_0 whose ground state is easy to prepare. In many problems $H_0 = \sum_{i=1}^{n} \sigma_x^i$ which tell us that we apply an X gate to each of our n qubits . If we prepare a quantum system to be in the ground state of H_0 and then evolve our system using a time dependent Hamiltonian

$$H(t) = (1 - \frac{t}{T})H_0 + \frac{t}{T}H_P$$

then our system will remain to its ground state at all times, which means that for $t = T$ our system will be in the ground state of H_p which is our solution. We chose the particular time dependent Hamiltonian because $H(0) = H_0$ and $H(T) = H_P$. Following these adiabatic theorem we must be very careful in how we evolve our system. That is that the time T that we will use scales with $O(g_{min}^{-2})$ where g_{min} is the minimum energy gap between the ground state and the next excited state of $H(t)$. For that reason, if g_{min} is very small then the time T will be very large and the whole process will be computationally difficult. Despite this, AQO has inspired quantum computing techniques such as the Quantum Annealing and has a close relation to the Quantum Approximate Optimization Algorithm (QAOA) which we will use later.

Quantum Annealing (QA) identifies the ground state of the time dependent Hamiltonian $H(t)$ that we had above. Although it fails to meet some of the conditions of the Adiabatic Theorem. This is useful because it relaxes the control of our quantum system e.g. the preparation of a pure quantum state that evolves under strictly adiabatic dynamics. However this relaxation allows our quantum state to mix with nearby energy eigenstates giving us a probabilistic model. For this reason after the QA has finished one needs to sample the observed outcome from a distribution of the energy eigenstates which will give us the correct solution with some probability.

In this day D-Wave has built quantum annealers that solves optimization problems by transferring the original optimization problem to a hardware connected graph (CHIMERA for example) that allows nearest neighbor interactions of qubits.

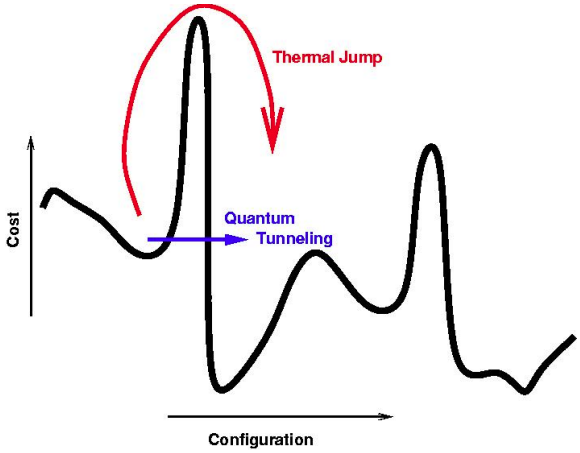


Figure 3.3: Cost-energy configuration. Using thermal jumps physical systems must go above the barrier while quantum tunneling can go through the barrier

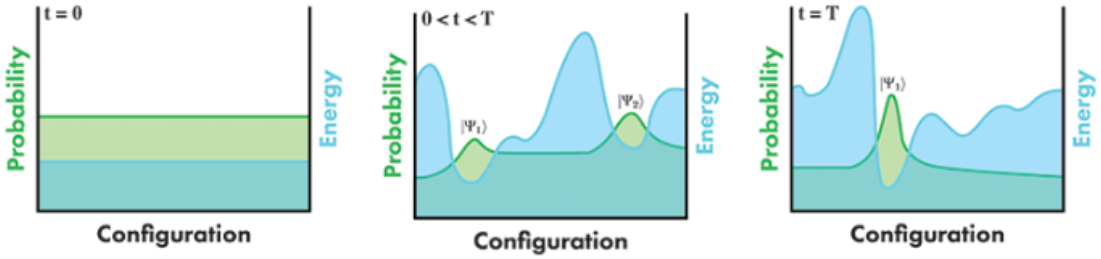


Figure 3.4: Enegy and probability distribution while running a quantum annealer, from the DWave manual handbook

In the figure (above) we see how the energy and probability distribution changes while running a quantum annealer. We see that for $t = 0$ the energy as well as the probability are equally distributed. For $0 \leq t \leq T$ we see that the energy distribution starts to show some minimums while the probability for those minimums is bigger than the other values of the energy. Finally for $t = T$ we see that energy has exactly one minimum value for which the probability is the biggest.

3.4 Variational quantum algorithms

We will now tackle the optimization problems using a hybrid quantum-classical model of computation. This model uses the quantum computer to obtain a parametrized wavefunction and then uses the classical computer to maximize/minimize the expected value of the problem Hamiltonian. It was suggested to circumvent the issue of going slow in the quantum annealers as well as implementing the Hamiltonian of the problem at hand in the available hardware. In general we create a parametrized quantum circuit which depends on some angle θ and then the classical computer varies this θ with the purpose of maximizing/minimizing $E(\theta) = \langle\psi(\theta)| H |\psi(\theta)\rangle$. Furthermore, we will analyze and compare the performance of two quantum variational algorithms for the MAXCUT problem, the Quantum Approximate Optimization Algorithm, Fahri et al. [7] and the Variational Quantum Eigensolver, Díez-Valle et al. [8] . The latter does not even require implementing the specific Hamiltonian but uses a more generic hardware and thus is also called hardware efficient, albeit it can be harder to optimize.

3.5 Solving MAXCUT with quantum computers

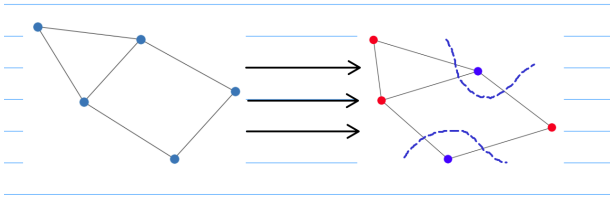


Figure 3.5: MAXCUT solution on 5 node graph

We are given an undirected graph $G = (V, E)$ where V is the set of nodes in the graph and E is the set of edges. We know that $|V|$ is the number of nodes and $|E|$ is the number of edges in the graph. The MAXCUT problem states as follows :

We want to split the set of nodes into two mutually exclusive sets while maximizing the edges that begin from one set and end to the other set.

As we can see in the figure above by splitting the nodes in the red and blue set we have achieved the maximum cut of 5 edges. The cost function to be maximized in this case is

$$C(x) = \sum_{i,j \in E} x_i + x_j - 2x_i x_j$$

where $x_i \in \{0, 1\}$. We can see that if $x_i + x_j - 2x_i x_j = 1$, then x_i and x_j belongs in different sets and if $x_i + x_j - 2x_i x_j = 0$, then x_i and x_j are both equal to 1 or 0 meaning that they belong to the same set. For this derivation we will use the two steps that we show at section (The Ising Model and QUBO).

We start by setting $x_i = \frac{s_i+1}{2}$ we get

$$C(s) = \sum_{i,j \in E} \frac{s_1 + 1}{2} + \frac{s_j + 1}{2} - 2\frac{s_i + 1}{2}\frac{s_j + 1}{2} \Leftrightarrow$$

$$C(s) = \frac{1}{2} \sum_{i,j \in E} s_i + 1 + s_j + 1 - s_i s_j - s_i - s_j - 1 \Leftrightarrow$$

$$C(s) = \frac{1}{2} \sum_{i,j \in E} 1 - s_i s_j$$

Now we replace the spin variables with pauli Z operators and obtain the quantum Ising Hamiltonian

$$C = \frac{1}{2} \sum_{(i,j) \in E} 1 - \sigma_i^z \sigma_j^z \tag{v1}$$

3.6 Quantum approximate optimization algorithm

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum algorithm that solves optimization problems. Hybrid quantum algorithms consists of a parametrized quantum circuit that depends on some variational parameters. They are called hybrids because they work alongside a classical computer whose purpose is to optimize the output of the quantum circuit with respect to the problem that we are trying to solve. In general, optimization problems require a lot

of steps in any algorithmic approach and therefore it are not easy for classical computers to solve them. For this reason approximation algorithms came to light, where their goal is to approximate the best solution rather than absolutely calculating the best solution. Solving the MAXCUT with QAOA will give the reader a good first understanding of the algorithm.

This algorithm is based on two unitaries $U(C,\gamma), \quad U(B,\beta)$ and an integer $p \geq 1$. The two unitaries corresponds to some rotations on our qubits while the integer p denotes the depth of our circuit.

We define the operator C to be the quantum Ising Hamiltonian eq. (v1) for the MAXCUT problem

$$C = \frac{1}{2} \sum_{(i,j)\in E} 1 - \sigma_i^z \sigma_j^z$$

We define the first unitary operator that depends on an angle γ as

$$U(C,\gamma) = e^{-i\gamma C} = \prod_{(i,j)\in E} e^{-i\gamma C_{(i,j)}}$$

where γ lies between 0 and 2π.

We define the operator B which is the sum of single bit σ^x operators as

$$B = \sum_{j=1}^{n} \sigma_j^x$$

where σ_j^x denotes the Pauli X gate acting on the j-th qubit.

Now we define the second unitary operator that depends on an angle β

$$U(B,\beta) = e^{-i\beta B} = e^{-i\beta \sum_{j=1}^{n} \sigma_j^x} = \prod_{j=1}^{n} e^{-i\beta\sigma_j^x}$$

where β lies between 0 and π. This is true because

$$e^{-i\beta\sigma_j^x} = \cos(\beta)\mathbb{I} - i\sin(\beta)\sigma_j^x$$

and if we replace the angle β with $\beta + \pi$ we have

$$\cos(\beta + \pi)\mathbb{I} - i\sin(\beta + \pi)\sigma_j^x = -(\cos(\beta)\mathbb{I} - i\sin(\beta)\sigma_j^x)$$

which is the same as rotating qubit j around the x axis with an angle β, since the minus sign upfront is a global phase and thus does not affect overall state.

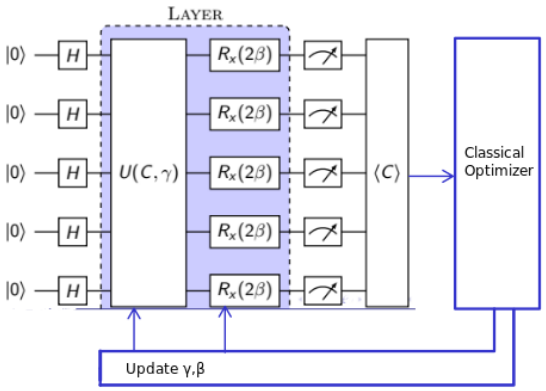


Figure 3.6: Variational loop of QAOA. For every p, we add an extra blue layer. We see that after computing the expectation value of C, we pass it as input to a classical optimizer that minimize/maximize it by running the quantum circuit and produces the best angles $\vec{\gamma}, \vec{\beta}$.

The starting state of the algorithm, as seen above, is the uniform superposition over 2^n different bit strings $|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle$.

The output of the algorithm is the state

$$\left|\psi(\vec{\gamma}, \vec{\beta})\right\rangle = U(B, \beta_p)U(C, \gamma_p)\ldots U(B, \beta_1)U(C, \gamma_1)\,|s\rangle$$

which depends solely on the angles $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_p)$, $\vec{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$.

So for each level p we apply the unitary $U(C, \gamma_p)$ followed by the unitary $U(B, \beta_p)$. This means that we have a total of $2p$ angles of the algorithm. We will also define the expectation value of the cost Hamiltonian C as

$$F_p(\vec{\gamma},\vec{\beta}) = \left\langle \psi(\vec{\gamma},\vec{\beta}) \right| C \left| \psi(\vec{\gamma},\vec{\beta}) \right\rangle$$

and let M_p be the maximum of F_p over the angles

$$M_p = \max_{\vec{\gamma},\vec{\beta}} F_p(\vec{\gamma},\vec{\beta})$$

Now let's examine the two operators $U(C,\gamma)$ and $U(B,\beta)$, and see how we can construct them using quantum gates. First we apply $U(C,\gamma)$, for $p = 1$, on an arbitrary quantum state

$$U(C,\gamma)\left|0\ldots00\right\rangle = e^{-i\gamma_1 C}\left|0\ldots00\right\rangle = e^{-i\gamma_1 \frac{1}{2}\sum_{i,j\in E}(1-\sigma_i^z\sigma_j^z)}\left|0\ldots00\right\rangle \Leftrightarrow$$

$$U(C,\gamma)\left|0\ldots00\right\rangle = \prod_{i,j\in E} e^{-i\gamma_1 \frac{1}{2}(1-\sigma_i^z\sigma_j^z)}\left|0\ldots00\right\rangle \tag{x.1}$$

A very important theorem of the linear algebra states that

$$e^A\left|u\right\rangle = e^\lambda\left|u\right\rangle, \quad \text{iff} \quad A\left|u\right\rangle = \lambda\left|u\right\rangle \tag{x.2}$$

If we look at a single term of the product at $(x.1)$ we have

$$e^{-i\gamma_1 \frac{1}{2}(1-\sigma_i^z\sigma_j^z)}\left|0\ldots00\right\rangle =$$

$$= e^{-i\gamma_1 \frac{1}{2}} e^{i\gamma_1 \frac{1}{2}\sigma_i^z\sigma_j^z}\left|0\ldots00\right\rangle \tag{x.3}$$

In general we have 4 different possibilities for that term and by using equation $(x.2)$ we get

$$e^{i\gamma_1 \frac{1}{2}\sigma_i^z\sigma_j^z}\left|00\right\rangle = e^{i\gamma_1 \frac{1}{2}1*1}\left|00\right\rangle = e^{i\gamma_1 \frac{1}{2}}\left|00\right\rangle$$

$$e^{i\gamma_1 \frac{1}{2}\sigma_i^z\sigma_j^z}\left|01\right\rangle = e^{i\gamma_1 \frac{1}{2}1*(-1)}\left|01\right\rangle = e^{-i\gamma_1 \frac{1}{2}}\left|01\right\rangle$$

$$e^{i\gamma_1 \frac{1}{2}\sigma_i^z\sigma_j^z}\left|10\right\rangle = e^{i\gamma_1 \frac{1}{2}(-1)*1}\left|10\right\rangle = e^{-i\gamma_1 \frac{1}{2}}\left|10\right\rangle$$

$$e^{i\gamma_1 \frac{1}{2}\sigma_i^z \sigma_j^z} |11\rangle = e^{i\gamma_1 \frac{1}{2}(-1)*(-1)} |11\rangle = e^{i\gamma_1 \frac{1}{2}} |11\rangle$$

So we can see that if bits i and j are the same then the coefficient of (x.3) becomes

$$e^{-i\gamma_1 \frac{1}{2}} e^{i\gamma_1 \frac{1}{2}} = 1$$

and if bits i and j are different then we have

$$e^{-i\gamma_1 \frac{1}{2}} e^{-i\gamma_1 \frac{1}{2}} = e^{-i\gamma_1}$$

Remember that the MAXCUT problem wants to split the nodes into two sets while maximizing the edges from one set to another. So if bits i and j are in the different set and there exists an edge between them, rotate the output state around the z axis by an angle of $2\gamma_1$. Note that the angle we rotate is $2\gamma_1$ because the Bloch sphere requires a half angle representation, remember that $R_Z(\theta) = e^{-i\frac{\theta}{2}\sigma^z}$ and therefore to rotate around z with an angle γ_1 we must give $2\gamma_1$ as input to R_Z.

(Below) we see the quantum circuit for $U(C, \gamma)$ (left) and for $U(B, \beta)$ (right).

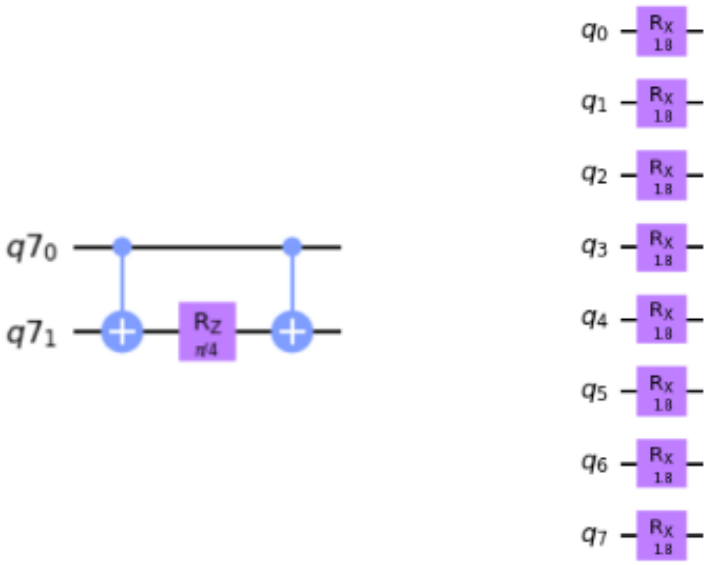


Figure 3.7: Left:Quantum circuit for one term of the operator $U(C, \frac{\pi}{8})$, the blue gate represents the CNOT gate. Right: Quantum circuit for the operator $U(B, 0.9)$.

In order to prove that this is the circuit in the figure, we start solving from left to right and apply the gate to an arbitrary state $|x, y\rangle$. So we have

$$\text{CNOT}\,|x, y\rangle = |x, x \oplus y\rangle$$

$$(\mathbb{I} \otimes R_Z)\,|x, x \oplus y\rangle = e^{-i(-1)^{x \otimes y}}\,|x, x \otimes y\rangle$$

$$\text{CNOT} e^{-i(-1)^{x \oplus y}}\,|x, x \otimes y\rangle = e^{-i(-1)^{x \otimes y}}\,|x, y\rangle$$

So we see that the quantum circuit (above) indeed produces a single term of the operator $U(C, \gamma_1)$.

For the operator $U(B, \beta_1) = e^{-i\beta_1 B} = \prod_{j=1}^{n} e^{-i\beta_1 \sigma_j^x}$, we see that it is a rotation of all the n qubits with an angle 2β around the x axis. So in terms of quantum gates we just use a $R_X(2\beta)$

from section (Two qubit gates).

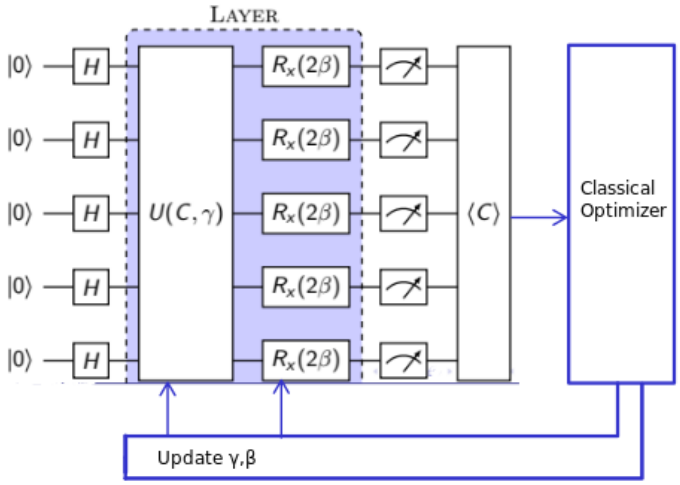


Figure 3.8: Variational loop of QAOA. For every p, we add an extra blue layer. We see that after computing the expectation value of C, we pass it as input to a classical optimizer that minimize/maximize it by running the quantum circuit and produces the best angles $\vec{\gamma}, \vec{\beta}$.

Let's go one step back and remember that QAOA is a variational algorithm. That means that a classical computer is cooperating with a quantum computer. We saw before that the output of the quantum computer is

$$\left|\psi(\vec{\gamma},\vec{\beta})\right\rangle = U(B,\beta_p)U(C,\gamma_p)\dots U(B,\beta_1)U(C,\gamma_1)\left|s\right\rangle$$

where $|s\rangle$ is the equal superposition of the 2^n states.

Now we can view this output state as a distribution of the 2^n bitstrings. For example if $n = 2$ and $p = 1$ then the output state might be something like

$$\left|\psi(\gamma,\beta)\right\rangle = \delta_0(\gamma,\beta)\left|00\right\rangle + \delta_1(\gamma,\beta)\left|01\right\rangle + \delta_2(\gamma,\beta)\left|10\right\rangle + \delta_3(\gamma,\beta)\left|11\right\rangle$$

where $|\delta_i(\gamma,\beta)|^2$ represents the probability of $\left|\psi(\gamma,\beta)\right\rangle$ to be in one of the 2^2 states and it depends solely on γ, β. Remember from section (what is a qubit?) that the sum of the probabilities

of the quantum state must be equal to one, i.e.

$$|\delta_0(\gamma,\beta)|^2 + |\delta_1(\gamma,\beta)|^2 + |\delta_2(\gamma,\beta)|^2 + |\delta_3(\gamma,\beta)|^2 = 1$$

Since this quantum state is measurable, we see that it takes values from $\{00, 01, 10, 11\}$ with some probability and these probabilities sum to one so we have the right to define the probability mass function of the quantum state. For example this (pmf) might look something like

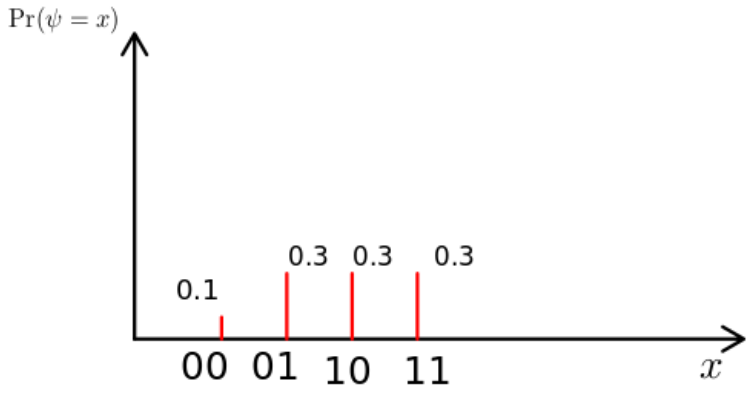


Figure 3.9: The probability mass function of a quantum state ψ consisting of 2 qubits. We have that $\Pr(|\psi\rangle = 00) = 0.1, \Pr(|\psi\rangle = 01) = \Pr(|\psi\rangle = 10) = \Pr(|\psi\rangle = 11) = 0.3$

In general we view the output of this algorithm as a distribution over all possible 2^n bit-strings. From this distribution our goal is to approximate the expected value of the cost function.

We can do that if we see that

$$F_p(\vec{\gamma},\vec{\beta}) = \left\langle \psi(\vec{\gamma},\vec{\beta}) \right| C \left| \psi(\vec{\gamma},\vec{\beta}) \right\rangle = \left\langle \psi(\vec{\gamma},\vec{\beta}) \right| \sum_{x\in\{0,1\}^n} C(x) |x\rangle \left\langle x \middle| \psi(\vec{\gamma},\vec{\beta}) \right\rangle = \sum_{x\in\{0,1\}^n} C(x) \left\langle \psi(\vec{\gamma},\vec{\beta}) \middle| x \right\rangle \left\langle x \middle| \psi(\vec{\gamma},\vec{\beta}) \right\rangle \Leftrightarrow$$

$$F_p(\vec{\gamma},\vec{\beta}) = \sum_{x\in\{0,1\}^n} C(x) | \left\langle x \middle| \psi(\vec{\gamma},\vec{\beta}) \right\rangle |^2 \tag{y.1}$$

where we have used the fact that C is diagonal in the computational basis and therefore we can

write it as

$$C = \sum_{x \in \{0,1\}} C(x) \left|x\right\rangle \left\langle x\right|$$

The above is an interesting property of diagonal matrices like C. So we saw how we can approximate the expected value of the cost function using a quantum computer. Note here that the sum in $(y.1)$ runs in $O(2^n)$ so for $n > 15$ the above method of approximation of the expected value of the cost function is not very efficient.

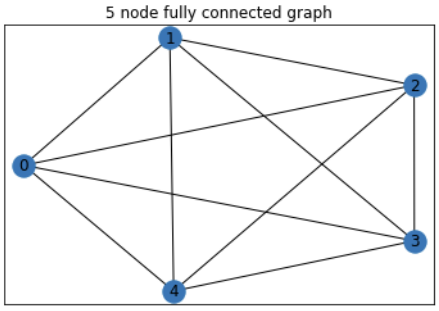
Last but not least, we will define the approximation ratio α which is a metric of how good our approximation is to the actual best value of the cost function. In the context of MAXCUT the approximation ratio is defines as

$$\alpha = \frac{F_p(\vec{\gamma}, \vec{\beta})}{C_{max}}$$

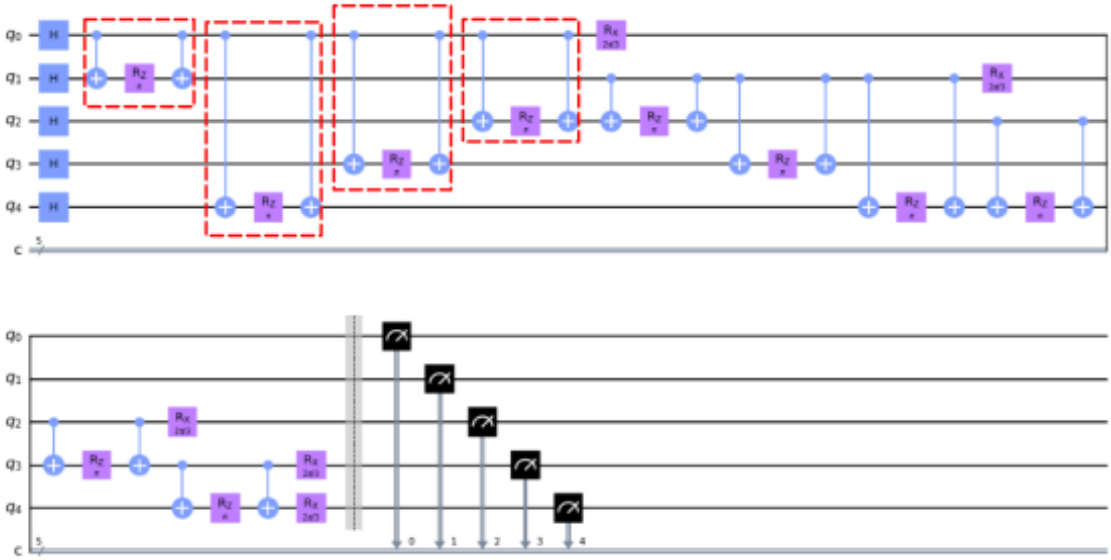
3.6.1 Using QAOA to solve the MAXCUT problem

With that being said we can now solve the MAXCUT for different graphs. We will start by solving the MAXCUT problem on a 5-node fully connected graph.

After constructing the graph, we must create the quantum circuit for QAOA. We start for $p = 1$, so we need the gates for $U(C, \gamma)$ for each edge and then a $R_X(\beta)$ for each node/qubit. We can see the quantum circuit below



(a) 5 node fully connected graph



(b) Quantum circuit for QAOA. In the red boxes we see operators $U(C, \gamma)$ for each edge from node 0 to 1, 4, 3, 2 respectively. The rotations around x can be seen for every qubit as the last gate. At the end we measure all the qubits in order to compute the weighted average.

Keep in mind that this circuit is being executed a lot of times because we want to optimize the expected value of the MAXCUT cost Hamiltonian $F_1(\gamma, \beta)$ over the angles γ, β. For this example I chose the COBYLA (Constrained Optimization By Linear Approximation) optimizer.

For this (graph) we have $C_{max} = 6$. We can see that if we split the nodes in two sets, where the one set will contain 3 nodes and the other set it will contain 2 nodes. With this in mind all the possible bit strings that are solutions to our problem are the ones with three '1' and two '0' bits

and vice versa. We visualize the solution (below)

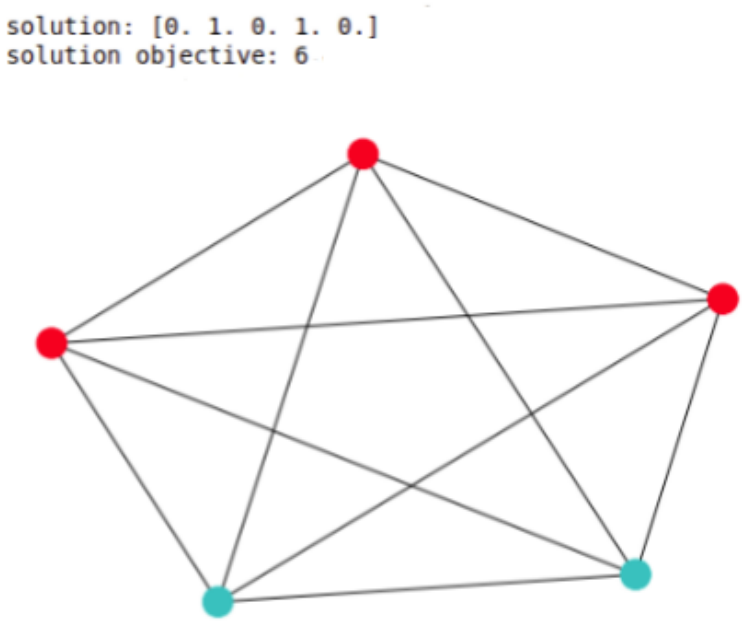
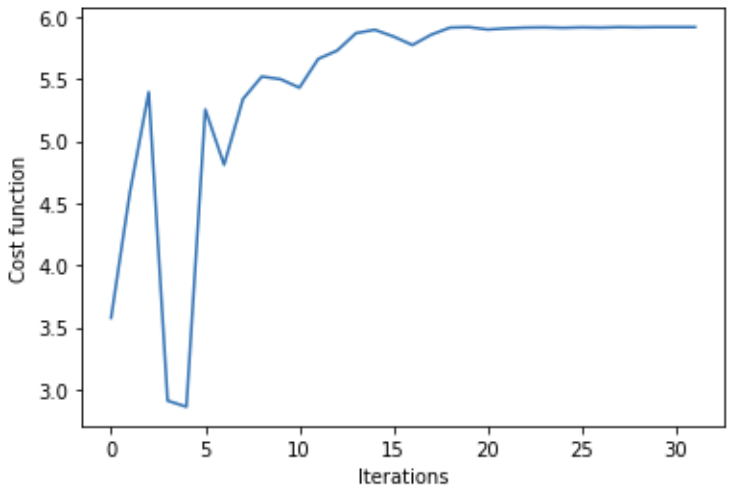


Figure 3.11: Solution for the MAXCUT in a 5 node fully connected graph. Bit string solution is '010101' with cost function value $C_{max} = 6$

The results from QAOA can be seen below. In figure (3.12) we have the results from the COBYLA optimizer for $p = 1$. First we see that $M_1(\gamma, \beta) = 5.920$ and the best angles to be $\gamma = 1.793$ and $\beta = 2.875$. We see that the optimizer ran the quantum circuit 32 times until it converges to a solution. Note that optimizers in python solve only minimization problems, so we told it to minimize the negative of the cost function which is the same as maximizing it.

In figure (3.13) we see the histogram of all the possible solutions to the problem. Since we have multiple solutions to our problem we see that QAOA produces them with relatively high probability. In order to produce this histogram we chose the best angles from the optimizer from (3.12) and then we ran the quantum circuit again. This time the estimated energy was a little higher $F_1(\gamma = 1.793, \beta = 2.875) = 5.949$ and we got the approximation ratio $\alpha = \frac{5.949}{6} = 0.9915$.




```
M1 = 5.920
Best angles-->
gamma = 1.793
beta = 2.875

     fun: -5.919921875
   maxcv: 0.0
 message: 'Optimization terminated successfully.
    nfev: 32
  status: 1
 success: True
       x: array([2.87520979, 1.79266232])
```

Figure 3.12: COBYLA statistics. We see here that after 32 cost function evaluations, we got $M_1(\gamma, \beta) = 5.920$. The -5.91 we see at the minimizer's results exists because we are solving the MAXCUT problem which is a maximization and python only has minimizer functions.

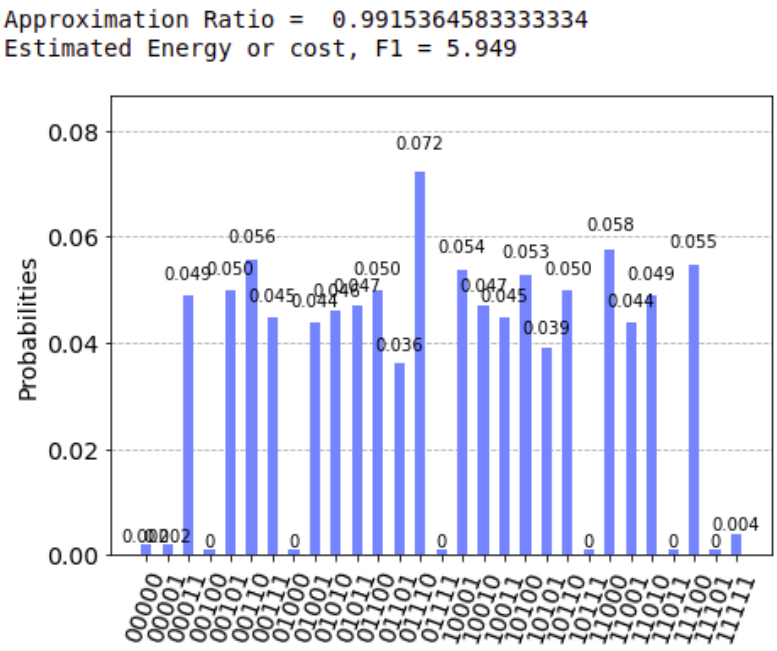


Figure 3.13: Histogram. We see here that the probabilities of the solutions to the MAXCUT problem for the 5 node graph almost sum up to 1.

Another interesting graph we plotted is the (distribution) of the cost function. For this we took 1024 measurements from the optimized quantum circuit and we found that 1016 out of 1024 measurements corresponded to the best solution. We can see the figure below

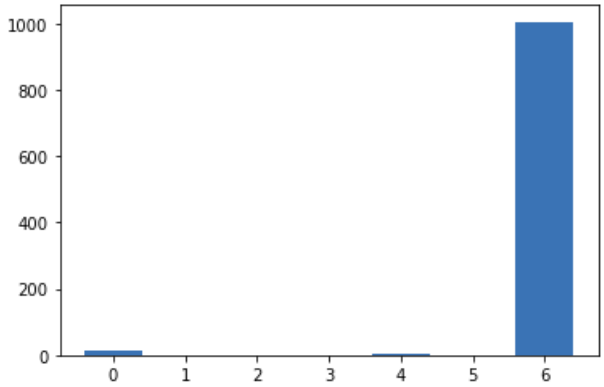


Figure 3.14: Distribution of the cost function. We see that almost all measurements that was made in the output distribution (after optimizing the circuit) produce the best cut.

To make things a little more interesting we ran the same problem in a real quantum computer, although I was not able to see or touch the quantum computer itself. This 5 qubit quantum computer is provided from IBM and we can assign it a quantum circuit to run directly from the cloud without any charge. The interesting fact here is that in a real quantum computer noise exists. With this in mind the approximation ratio dropped as well as the estimated cost function. Below we see the results

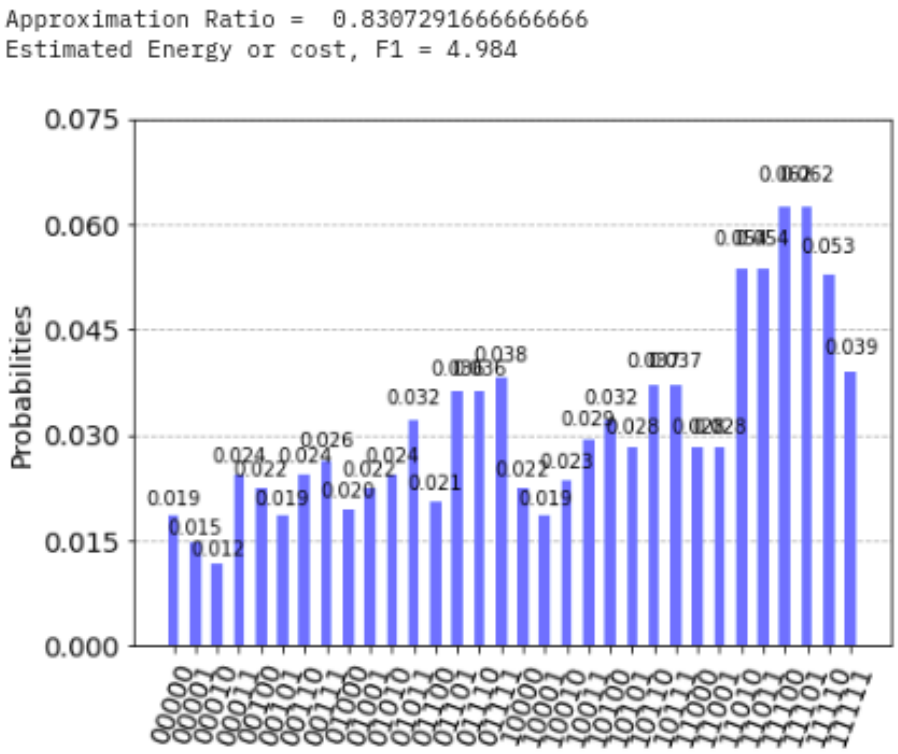
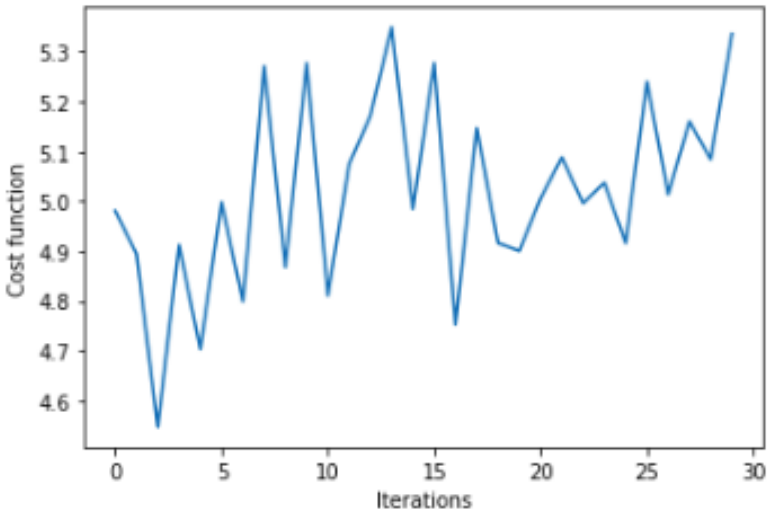


Figure 3.15: Histogram, due to the noise we see that amplitudes of bitstring that in the noisless simulation were 0, now they have probability to be sampled.

In the figure above we see that sampling from quantum computer becomes more difficult when noise is present. The approximation ratio dropped from 0.99 (noiseless simulator) to 0.83 (real quantum computer) and the estimated energy dropped from 5.949 to 4.984. With these results we can see that quantum computers are still young and have a lot of potential to grow.




```
M1 = 5.336
Best angles-->
gamma = 1.360
beta = 1.739

:      fun: -5.3359375
     maxcv: 0.0
   message: 'Optimization terminated successfully.'
      nfev: 30
    status: 1
   success: True
         x: array([1.73909282, 1.35989408])
```

Figure 3.16: COBYLA statistics. The presence of noise can be seen in the optimization of the variational circuit.

We will now test the algorithm in a 17 fully connected graph. The simulations we ran here are all noiseless since the only quantum computer that we have access consists of 5 qubits.

We start by showing the graph that we are working with which have the max cost function value of $C_{max} = 72$.

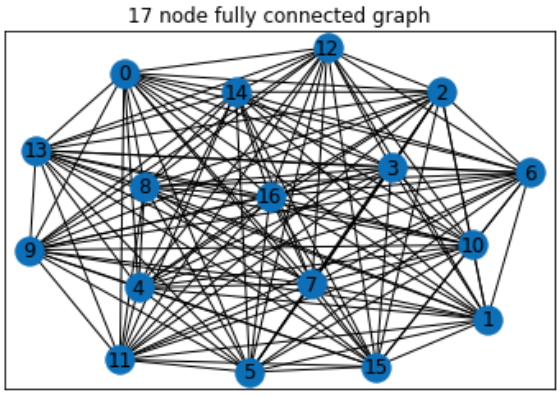


Figure 3.17: Fully connected graph with 17 nodes.

Here we will plot the approximation ratio vs the depth of the quantum circuit p. Results are shown (below) for $1 \leq p \leq 5$

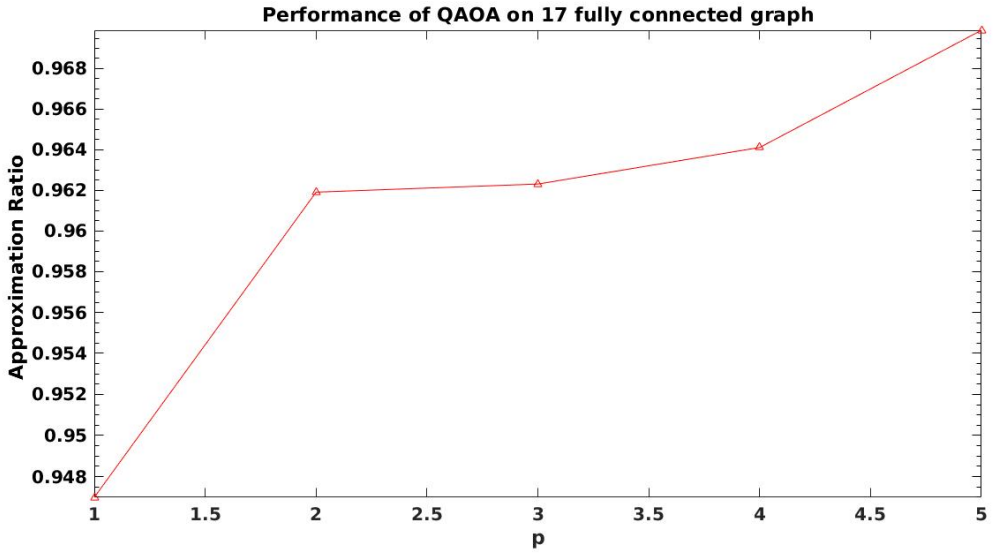


Figure 3.18: Approximation ratio vs p. We see that as p grows the approximation ratio improves.

In the figure (above) we see that as p grows the approximation ratio gets closer to 1. This experimental result corresponds to the theoretical results that states

$$\lim_{p\to\infty} M_p = \max_z C(z)$$

where $M_p = \left\langle \vec{\gamma}, \vec{\beta} \right| C \left| \vec{\gamma}, \vec{\beta} \right\rangle$ ((Appendix)).

In general we see that in the absence of noise, QAOA produces an approximation ratio of over 0.9 for $p = 1$ and it only gets better. Although in order to really experiment with this algorithm, noise must be present in all the experiments because noise exists in the quantum hardware that this algorithm is supposed to run. As a final experiment we tested the approximation ratio of QAOA on fully connected graphs of different size. In particular we tested graphs with number of nodes q = number of qubits with $6 \leq q \leq 17$. Below we see the results

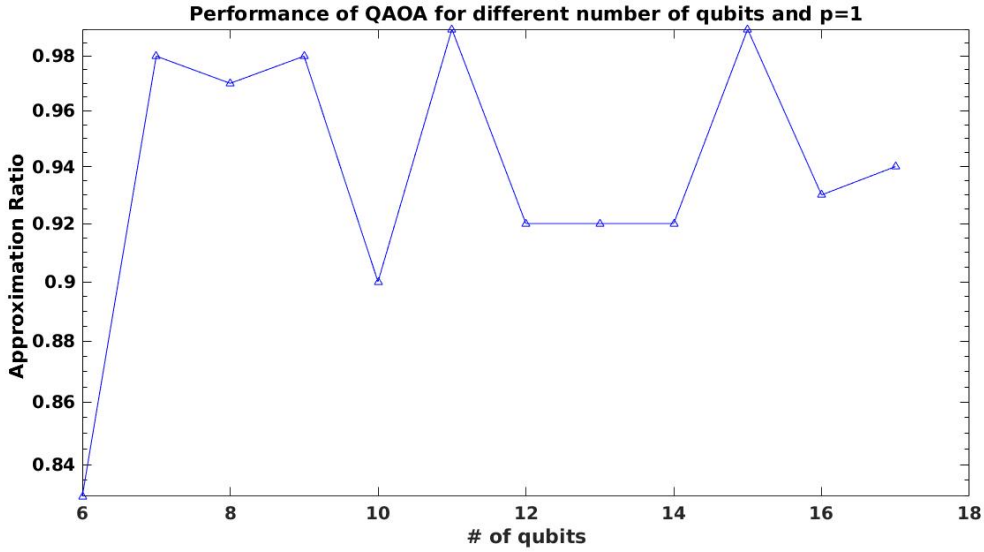


Figure 3.19: Approximation ratio vs number of qubits, for different graphs.

Overall, QAOA is a probabilistic algorithm in nature. This means that it produces a distribution of the cost function that relies on the depth of the quantum circuit p as well as the angles $\vec{\gamma}, \vec{\beta}$. MAXCUT is a combinatorial optimization problem that has been studied for many many years. In general there exists classical algorithms that outperforms QAOA like the Michel X. Goemans and David P. Williamson approximation algorithm [9]. The MAXCUT problem is also used in the

original QAOA paper (Fahri et al. [7]) to study the performance of QAOA due to the relative ease of the problem and the well defined cost function. For me personally MAXCUT was a good first introduction to QAOA.

3.7 Variational quantum eigensolver

The Variational Quantum Eigensolver (VQE) is a hybrid quantum classical algorithm. It's goal is to approximate the minimum eigenvalue λ_{min} and corresponding eigenvector $|\psi_{min}\rangle$ of a problem Hamiltonian. This has many applications from chemistry to finance. In general we can encode an optimization problem (QUBO) to a Hermitian matrix i.e. the Hamiltonian and use this algorithm to find λ_{min} which corresponds to the minimum solution of our problem.

VQE provides an estimate $\lambda_{\vec{\theta}}$ that bounds λ_{min} as

$$\lambda_{min} \leq \lambda_{\vec{\theta}} \equiv \left\langle \psi(\vec{\theta}) \right| H \left| \psi(\vec{\theta}) \right\rangle$$

In the same fashion as QAOA, VQE also uses a parametrized quantum circuit controlled by the parameter vector $\vec{\theta}$.

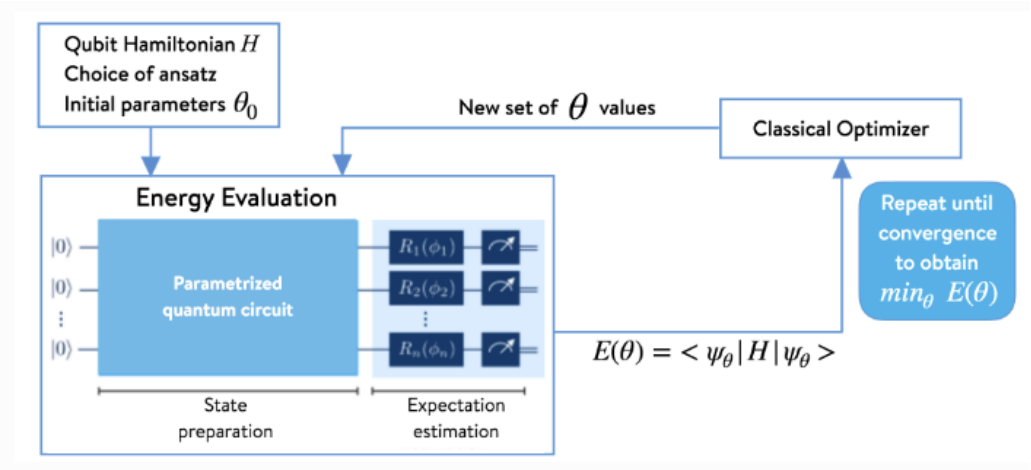


Figure 3.20: Variational loop of VQE. Here we see the quantum part of the algorithm (state preparation + expectation estimation) that cooperates with the classical part (optimizer).

Before explaining the quantum circuit let's understand how this algorithm works.

We represent the quantum circuit as an operator $U(\vec{\theta})$. We apply this operator to a starting state $|\psi\rangle$ which in the most cases is the all 0 state i.e $|\psi\rangle = |00\ldots0\rangle$. The action of the quantum circuit upon this state is represented by the equation

$$U(\vec{\theta})\,|\psi\rangle = \left|\psi(\vec{\theta})\right\rangle$$

We then use a classical optimizer to optimize $\left|\psi(\vec{\theta}\right\rangle)$ such that

$$\left\langle\psi(\vec{\theta})\right| H \left|\psi(\vec{\theta})\right\rangle \approx \lambda_{min}$$

where H is the Hamiltonian encoding our optimization problem.

The quantum circuit for VQE is hardware efficient for the NISQ era. What this means is that it uses gates that exists in quantum processors in this era. These gates are rotations around the y axis and controlled Z operations. (Below) we see the quantum circuit

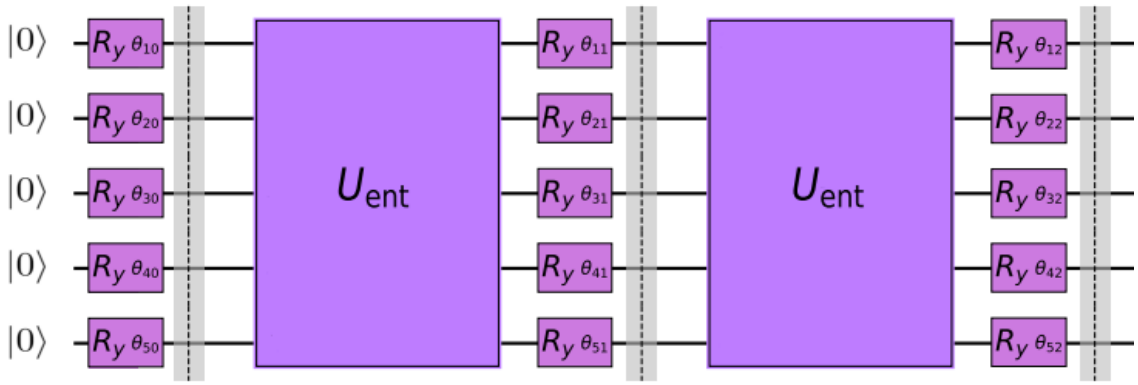


Figure 3.21: Quantum circuit for VQE with 5 qubits and 2 layers. U_{ent} is composed of controlled Z gates between bearest neighbors qubits. Note that at the very start we apply an R_Y gate to all the qubits. Using this structure we have 15 variables to optimize (angles of R_Y gates).

By using this configuration of gates for the VQE we are using linear entanglement [8]. Here we use the term linear entanglement because each Controlled Z gate is applied between nearest neighbors qubits. So we are using this circuit to create entangling states. The circuit responsible

for this is shown in the figure (above). This circuit's wavefunction can be written as

$$U_{\vec{\theta}} = \prod_{l=1}^{L} \Big[\prod_{n=1}^{N} e^{i\theta_{nl}\sigma_n^y} U_{ent} \Big] \prod_{n=1}^{N} e^{i\theta_{n0}\sigma_n^y}$$

where U_{ent} is the entangling unitary, N is the number of qubits that we have and L is the number of layers. In total we have $N(L+1)$ variables to optimize. The linear entangling gate U_{ent} can be written as

$$U_{ent} = \prod_{i=1}^{N} e^{i\frac{\pi}{4}(\mathbb{I}-\sigma_i^z)(\mathbb{I}-\sigma_{i+1}^z)}$$

where σ_i^z is the Pauli Z gate acting on qubit i. We initialize $\theta_{n0} = \frac{\pi}{4}$ in order to get the uniform superposition over all qubits. We also put an angle of $\theta_{n0} = \frac{\pi}{4}$ in the entangling gate because of the double angle representation. So for example, if we have one layer, when the above terms gets multiplied then qubit i will get rotated for $\frac{2\frac{\pi}{4}}{2} = \frac{\pi}{4}$ and we will get the equal superposition of the qubits.

3.8 Comparison of QAOA and VQE

In this section we compare the performance of QAOA and VQE for the MAXCUT problem. First we extract the theoretical complexities for both algorithms, with respect to the number of gates and the number of variables they use. Then we run the algorithms for different complete graphs so that we can see the relation between the approximation ratio and the number of nodes or qubits. At last we plot the number of cost function evaluations (circuit runs) in the different MAXCUT problems that we solved. Since both algorithms are variational, the time complexity that they have depends heavily on the classical optimization part.

More particular, for complete graphs only, the total number of gates that QAOA uses scales with $O(pn^2)$ while VQE uses $O(pn)$ gates. Although the number of variational parameters for QAOA is $2p$ while for VQE it is $n(p+1)$, where n is the total number of qubits and p is the number of layers in both algorithms. The number of gates and the number of variational variables for the two

algorithms can be seen below.

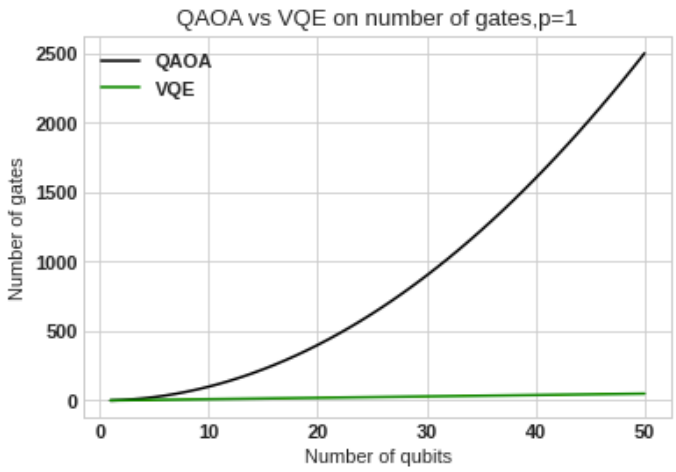


Figure 3.22: Number of gates between QAOA and VQE for $p = 1$ (one layer of gates)

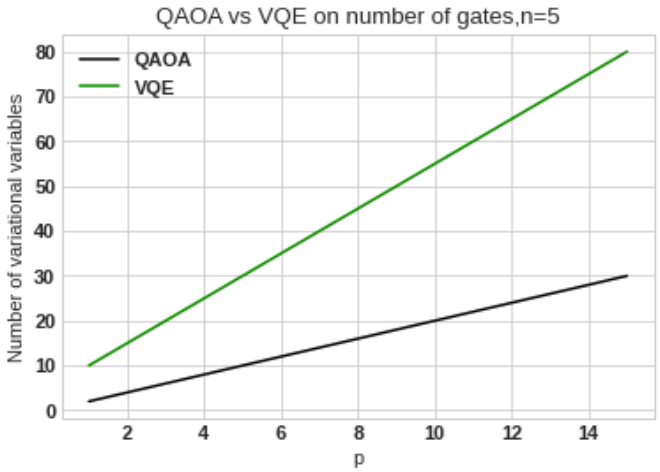


Figure 3.23: Number of variational variables between QAOA and VQE for 5 qubits. Keep in mind that the only dependence on the number of qubits comes from VQE.

For example, suppose we want to solve the MAXCUT problem for a 20 qubit fully connected graph using only one layer of gates. QAOA would need 400 gates with only 2 variational variables. VQE on the other hand would need 20 gates with 40 variables to be optimized. With these in mind we can see a tradeoff between number of gates and number of variational parameters in these

algorithm.

With that being said, below we see the performance of both algorithms in the context of MAX-CUT. More particular in the (figure) below we see the approximation ratio of the algorithms vs the number of qubits. Both algorithms were tested for one layer of gates and we found that VQE produces a better approximation ratio. The biggest difference in the approximation ratios comes for the 6 node graph and it is of the order of 0.2. In general, QAOA has an average approximation ratio of 0.943 while VQE has the average of 0.99977. So we can see that in average VQE performs better, but the difference is small.

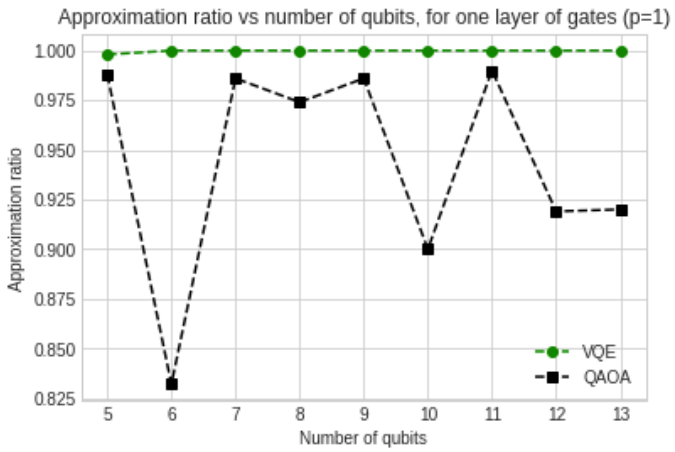


Figure 3.24: Approximation ratio vs number of qubits, for $p = 1$. Although both algorithms perform with an approximation ratio almost over 0.9, VQE has better accuracy.

(Below) lies the essence of this section. We see the cost function/quantum circuit evaluations for both algorithms. These evaluations affect the time complexity for both algorithms. QAOA has an average of 30.88 evaluations while VQE has an average of 434.33. Keep in mind that for these experiments we set the maximum evaluations of the classical optimizer at 500. So we see that after 6 qubits VQE always reaches the maximum evaluations.

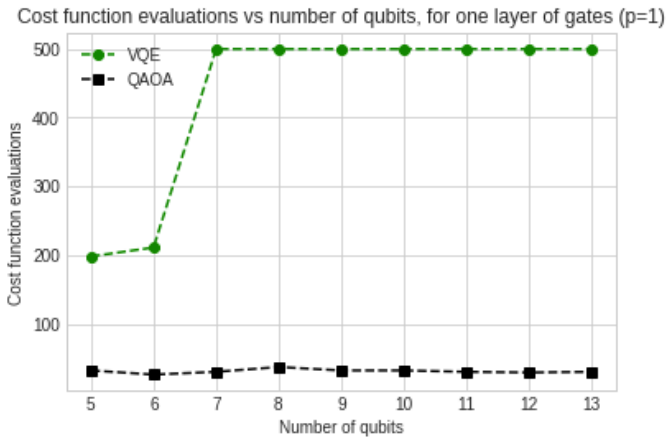


Figure 3.25: Cost function evaluations vs number of qubits, for $p = 1$. Maximum number of evaluation was set to 500. Even for the 5-node complete graph, VQE requires 198 cost function evaluations while QAOA needs only 32.

Both algorithms promise great things to the combinatorial optimization theory. Researchers like Ioannis Kolotouros and Petros Wallden [10] have showed that by changing the conventional cost function to a cost function that comes from the Conditional Value at Risk (CVaR), better results can be achieved both in performance and time complexity. Also research is being conducted in how we can reduce the number of two qubit gates in QAOA (Li Li et al. [11]) or how we can use an encoding scheme to reduce the n number of qubits to $\log_2 n + 1$ qubits (Benjamin Tan Angelakis Dimitris et al. [12]). The latter work allows for industrial level problems to be tackled with near term quantum processors.

Chapter 4

Applications of quantum optimization algorithms in real world scenarios: the tail assignment problem

4.1 Tail assignment problem

In this chapter we will focus on applying QAOA in optimization problems that exists in the community, one such problem is the Tail Assignment Problem (TAP). This problem comes from the airplane industries and it is the task of assigning individual air crafts to a given set of flights minimizing the overall cost. Because the problem here is very generic and generic optimization problems have many constraints we will focus on the decision version of the problem.

Decision optimization problems can be answered with a yes or no, so we produce an answer to the problem and we ask, is this a good answer? if the answer is good we accept it otherwise we

don't. In general we are interested in QUBO problems so we are looking to reduce bigger problems to QUBO problems. That is exactly what we are going to do with the TAP. The decision version of the TAP problem is called the Exact Cover Problem which is NP complete (Richard M. Karp [13]) .

The Exact Cover Problem is stated as follows : Consider a set $U = \{1, \ldots, n\}$ and subsets $V_i \subseteq U (i = 1, \ldots, m)$ such that $U = \cup_i V_i$. The question we ask is : Is there a subset called R that contains sets V_i, such that the elements of R are disjoint sets and the union of elements of R is U? In this day there exists a classical algorithm called algorithm X that solves this problem in $O(mn)$ (proposed by Donald Kuth [14]).

First let's state the problem that we are going to solve. This problem includes 80 flights which we declare in a set

$$F = \{0, 1, \ldots, 79\}$$

How we are going to name these flights have no meaning to the problem itself. We will also define 8 routes in the set

$$R = \{0, 1 \ldots, 7\}$$

Each route 0, 1, 2, ... contains a random number of flights from F. The strict definition of a route is a continuation of individual flights. The question we ask here is : In what way we split the elements of R such that we create a new set Y whose elements are disjoint and their union is F? By answering the previous question we solve the Exact Cover Problem and then we assign each element of Y to a different aircraft and solve the TAP. In particular we assign binary variables to the elements of Y (1 if we include the route in Y and 0 otherwise). We do that in order to have a bit string that represent the solution to our problem i.e. bit string '00001111' means that we assign routes 0, 1, 2, 3 to 4 different air crafts and we have covered all the flights.

The problem that we will solve is shown in the (table) below

Routes	Flights
0	0,1,...,19
1	20,21,...,39
2	40,41,...,59
3	60,61,...,79
4	6,20,40,60
5	4,41,21,61
6	5,22,62
7	2,43,63

Figure 4.1: Table of the flights that exist in each route. The solution to our problem is the bit string '00001111'.

I chose this problem because the solution is easy to find i.e. $R_0 \cup R_1 \cup R_2 \cup R_3 = F$ and $R_0 \cap R_1 \cap R_2 \cap R_3 = \emptyset$.

From this table we can get the problem graph. In the problem graph we represent each route with a node and edges exists between routes/nodes that share a flight.

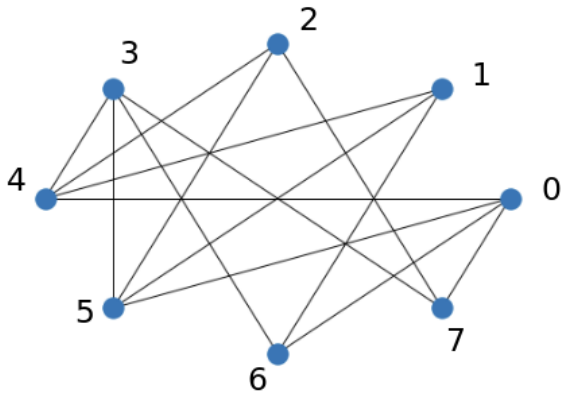


Figure 4.2: 8 route, 80 flight graph. Routes are represented with number 0 to 7.

Now it's time to move to the mathematics behind the problem. First we need to define the cost function to be minimized (as showed by Pontus Vikstål et al. [15]). That is

$$C(\vec{x}) = \sum_{f=1}^{|F|}(1 - \sum_{r=1}^{|R|}\alpha_{fr}x_r)^2 \tag{z.1}$$

Where $\alpha_{fr} = 1$ iff flight f exists in route r and $x_r = \{0, 1\}$ is the r-th bit of solution vector $\vec{x}$.

This cost function has a minimum value of 0. It is easy to check that

$$C(00001111) = 0$$

The squared term of the cost function makes the minimum value to be 0. To see that we must check when the squared term is equal to 0, that is when

$$\sum_{r=1}^{|R|} \alpha_{f_r} x_r = 1$$

or when each flight exists in only one route. If one flight exists in more that one route then the cost function rises.

4.2 Minimizing the expected value of the quantum ising hamiltonian

In order to solve this problem using a quantum computer we must map the cost function (z.1) to a quantum Ising Hamiltonian that looks like

$$H_C = \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z + \sum_{i=1}^{n} h_i \sigma_i^z \tag{z.2}$$

We start with the cost function from $(z.1)$ and we do the mapping as suggested in section (The ising model and QUBO). That is we first map the binary variables $x_r \in \{0, 1\}$ to spins $s_r \in \{-1, 1\}$ with $x_r = \frac{s_r+1}{2}$. So equation $(z.1)$ becomes

$$C(\vec{s}) = \sum_{f=1}^{|F|} (1 - \sum_{r=1}^{|R|} \alpha_{fr} \frac{s_r + 1}{2})^2$$

Now we develop the squared term and we have

$$C(\vec{s}) = \frac{1}{4} \sum_{f=1}^{|F|} \sum_{r=1}^{|R|} \sum_{r'=1}^{|R|} \alpha_{f_r} \alpha_{f_{r'}} s_r s_{r'}$$

$$+\frac{1}{2}\sum_{f=1}^{|F|}\sum_{r=1}^{|R|}\alpha_{f_r}s_r\Big(\sum_{r'=1}^{|R|}\alpha_{f_{r'}}-2\Big)$$

$$+\frac{1}{4}\sum_{f=1}^{|F|}\Big(\sum_{r=1}^{|R|}\alpha_{f_r}-2\Big)^2$$

In order to get $C(\vec{s})$ in the form of (z.2) we define

$$J_{rr'}=\frac{1}{2}\sum_{f=1}^{|F|}\alpha_{fr}\alpha_{fr'} \tag{z.3}$$

$$h_r=\frac{1}{2}\sum_{f=1}^{|F|}\alpha_{fr}\Big(\sum_{r'=1}^{|R|}\alpha_{fr'}-2\Big) \tag{z.4}$$

$$\text{const}=\frac{1}{4}\sum_{f=1}^{|F|}\Big(\sum_{r=1}^{|R|}\alpha_{fr}-2\Big)^2 \tag{z.5}$$

where (z.3) represents the interaction terms of the quantum Ising Hamiltonian between routes r and r'. It gives a penalty to the cost function if routes r and r' share a flight. Equation (z.4) represents the linear terms of the quantum Ising Hamiltonian for each route r. This term gives a penalty to the cost function if a flight belongs to two or more routes and rewards the cost function if the flight belongs only in one route. Lastly equation (z.5) is the constant term of the quantum Ising Hamiltonian which we can exclude for the optimization part.

So the cost function that we will use in this problem is the quantum Ising Hamiltonian

$$H_C=\frac{1}{2}\sum_{r=1}^{|R|}\sum_{r'=1}^{|R|}J_{rr'}\sigma_r^z\sigma_r'^z+\sum_{i=1}^{n}h_i\sigma_i^z \tag{z.6}$$

At this point we have the closed mathematical form of the quantum Ising Hamiltonian which represents our cost function. So, we have achieved to encode the solution of the Exact Cover Problem into the quantum Ising Hamiltonian. More particular we will have

$$H_C\,|x_{sol}\rangle=0\,|x_{sol}\rangle=0$$

where $x_{sol} =' 00001111'$ is the solution to our problem.

The next step is to create the quantum circuit for QAOA. In order to do this we must find the operator $U(H_C, \gamma)$. We can directly compute this operator if we directly compute the H_C matrix. To compute H_C all we need is the interaction term matrix J and the linear term matrix h. We will start by calculating J from (z.3) for each edge in the problem graph. We have

$$J = \begin{pmatrix} 10 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 10 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0 & 10 & 0 & 0.5 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 10 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 & 2 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0 & 2 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0.5 & 0 & 0 & 1.5 & 0 \\ 0.5 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 1.5 \end{pmatrix}$$

where for example $J_{04} = 0.5$ because route 0 and route 4 have 1 flight in common and $J_{00} = 10$ because route 0 has 20 elements in total.

We calculate the linear terms matrix from (z.4) and we have

$$h = \begin{pmatrix} -8 & -8.5 & -8.5 & -8 & 0 & 0 & 0 & 0 \end{pmatrix}$$

where $h_0 = -8$ because from the 20 elements of route 0, 4 elements exists in exactly 2 routes and 16 elements exists in 1 routes so $h_0 = \frac{1}{2}(4 * 0 + 16 * (-1)) = -8$.

Now we calculate H_C using equation (z.6)

$$H_C = 47 + \frac{1}{2}(Z_0Z_4 + Z_0Z_5 + Z_0Z_6 + Z_0Z_7 + Z_1Z_4 + Z_1Z_5 + Z_1Z_6 + Z_2Z_4 + Z_2Z_5 + Z_2Z_7$$

$$+Z_3Z_4 + Z_3Z_5 + Z_3Z_6 + Z_3Z_7) - 8Z_0 - 8.5Z_1 - 8.5Z_2 - 8Z_3 \tag{z.7}$$

where we have grouped together terms like $0.5Z_0Z_4$ and $0.5Z_4Z_0$ since the contribution of edge $(0,4)$ is the same with $(4,0)$.

Now we can compute the operator $U(H_C,\gamma) = e^{-i\gamma H_C}$

$$U(H_C,\gamma) = e^{-i47\gamma}e^{-i\frac{\gamma}{2}Z_0Z_4}e^{-i\frac{\gamma}{2}Z_0Z_5}e^{-i\frac{\gamma}{2}Z_0Z_6}e^{-i\frac{\gamma}{2}Z_0Z_7}e^{-i\frac{\gamma}{2}Z_1Z_4}e^{-i\frac{\gamma}{2}Z_1Z_5}e^{-i\frac{\gamma}{2}Z_1Z_6}$$

$$e^{-i\frac{\gamma}{2}Z_2Z_4}e^{-i\frac{\gamma}{2}Z_2Z_5}e^{-i\frac{\gamma}{2}Z_2Z_7}e^{-i\frac{\gamma}{2}Z_3Z_4}e^{-i\frac{\gamma}{2}Z_3Z_5}e^{-i\frac{\gamma}{2}Z_3Z_6}e^{-i\frac{\gamma}{2}Z_3Z_7}e^{i8\gamma Z_0}e^{i8.5\gamma Z_1}e^{i8.5\gamma Z_2}e^{i8\gamma Z_3} \quad \text{(z.8)}$$

Here we can exclude $e^{-i47\gamma}$ since it is a global phase and thus not measurable. What we have achieved by writting this product of exponents is that we know a-priori the gates and the coefficients that we are going to use in our quantum circuit. Operator $U(B,\beta)$ is still a rotation around the x axis on all the qubits.

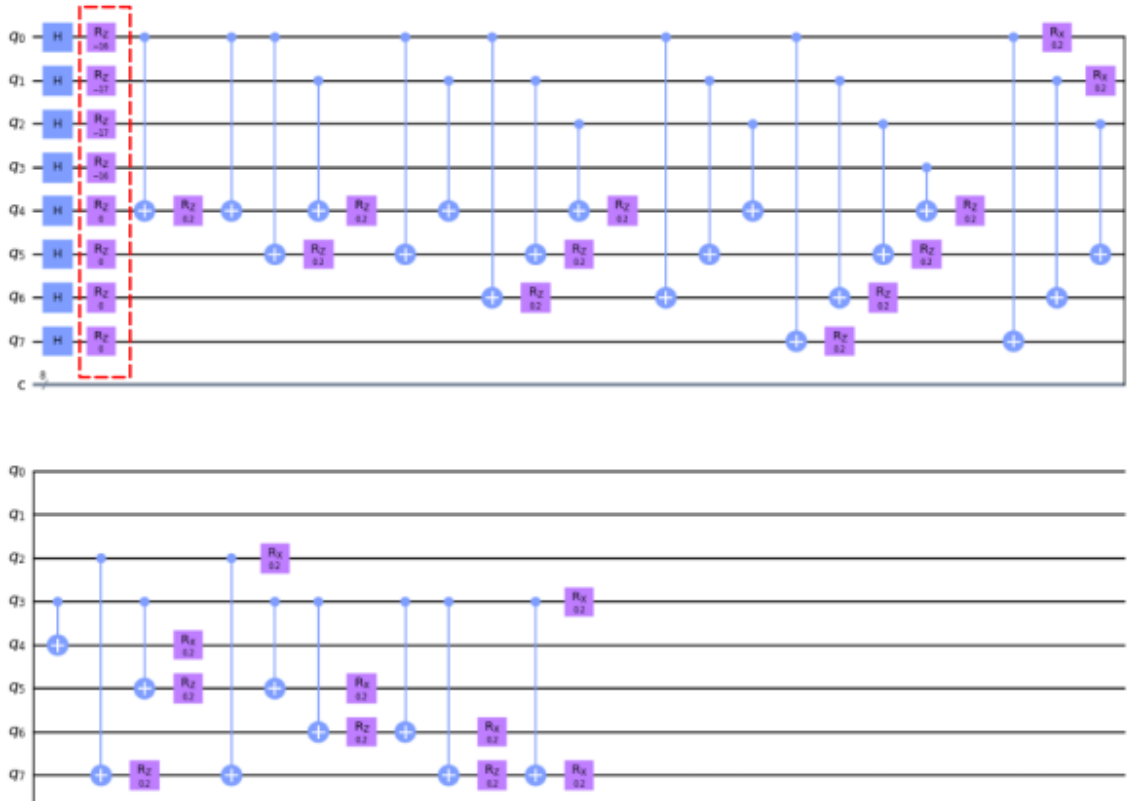


Figure 4.3: Quantum circuit for the 8 route, 80 flights problem. In contrast with the quantum circuit used for the MAXCUT problem, here we see the linear coefficients in the red dashed box.

Keep in mind that here we do not measure at the end of the quantum circuit. We did that in order to get the statevector of the quantum circuit. The statevector is a vector of dimensions $2^n \times 1$ that holds the coefficients of each state. For example if we have 2 qubits and the output state is $|\psi\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$, then the statevector would be

$$sv = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

From this statevetor we can get the probability of each state if we get the square of each element.

Before analyzing the results from QAOA,first we see the optimization strategy that we used for this particular problem.

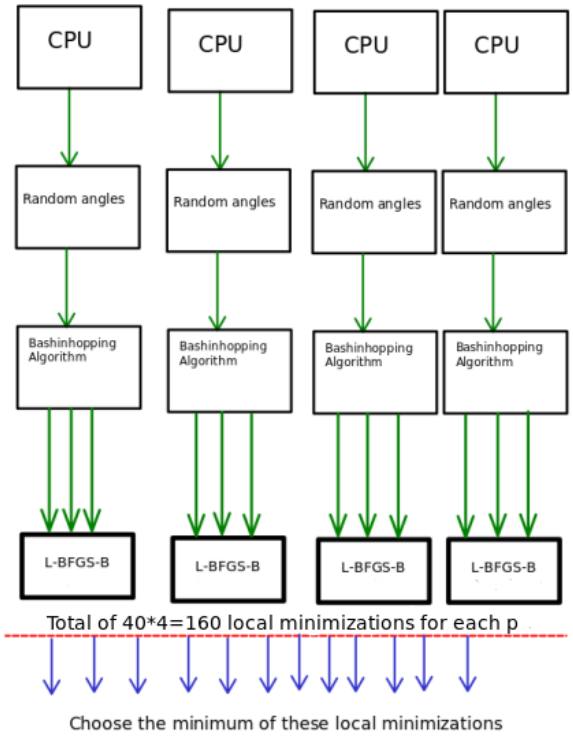


Figure 4.4: Parallel optimization of the objective function for the TAP at hand.

For this problem we used parallel optimization. In particular we ran the Bashinhopping algorithm from each CPU with a total of 4 CPUs. The Bashinhopping algorithm is a stochastic algorithm which is designed to find the global minimum. Each CPU gives the algorithm one random starting point i.e $\vec{\gamma}, \vec{\beta}$ and then the Bashinhopping algorithm runs a local optimizer 40 times. After each local optimizer have finished it runs the Metropolis criterion in order to accept or reject the optimized value. As a local optimizer we used the L-BFGS-B. At the end all 4 CPUs have finished and we choose the minimum value. So in total we are doing 160 local minimizations for each p. For $1 \leq p \leq 3$ this algorithm always converged to the global minimum, but for $p \geq 4$ the algorithm did not always converged.

We start by running QAOA and we minimize the expected value of the quantum Ising Hamiltonian. We calculate this expected value using the equation (y.1)

$$\langle H_C \rangle = F_p(\vec{\gamma}, \vec{\beta}) = \sum_{x=0}^{2^n-1} P(x)C(x)$$

where $P(x)$ is the probability of state $|x\rangle$ which we get from the statevector and $C(x)$ is the cost of state x which we can get from H_C.

With this in mind we have the below figure for $1 \leq p \leq 7$.

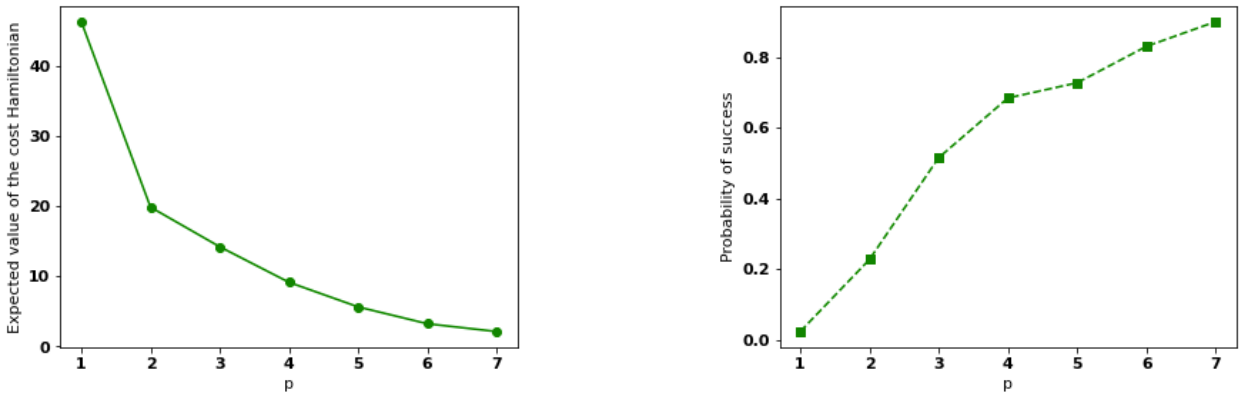


Figure 4.5: Results from QAOA for the 8 route 80 flights graph.

In the left figure we have plotted the expected value of H_C. The goal here is to get $\langle H_C \rangle = 0$. The smallest value was found at $p = 7$ with $\langle H_C \rangle = 2.13$. In the right figure we have plotted the success probability. Since we have a single solution to our problem we expect the probability of the solution to grow as the $\langle H_C \rangle$ goes to 0. Indeed we see that as $\langle H_C \rangle$ goes to 0 the probability of success raises.

Although this probabilities reffer to single shot measurements of the distribution of states after running QAOA for each p. We can also run QAOA for p=3 and obtain the distribution of possible solutions with $Pr('00001111') = 0.515$. Now we can measure this distribution m times and still obtain the correct solution. In particular the probability of having observed the solution at least once after m measurements is

$$1 - (1 - Pr('00001111'))^m \tag{z.9}$$

To get this equation we define that each measurement is independent of the others and $(1 - Pr('00001111'))^m$ is the probability to measure the other states in the m measurements. We want to have at worst $1 - \epsilon$ probability to measure the correct solution. So

$$1 - (1 - Pr('00001111'))^m < 1 - \epsilon \Leftrightarrow$$

$$(1 - Pr('00001111'))^m > \epsilon \Leftrightarrow$$

$$\log\Big((1 - Pr('00001111'))^m\Big) > \log \epsilon \Leftrightarrow$$

$$m > \frac{\log \epsilon}{\log\big(1 - Pr('00001111')\big)}$$

So if we set $\epsilon = 0.01$ we get the probability to measure the correct solution to be at worst 0.99 after measuring the quantum circuit $m = 7$ times. Overall we would have to optimize the quantum circuit for $p = 3$, then obtain the variational parameters $\vec{\gamma}, \vec{\beta}$, prepare the quantum circuit with the aforementioned variational parameters and measure 7 times. If we follow this procedure we will measure the correct solution at least once with probability 0.99.

Overall, if we compare these results with the ones from the MAXCUT problem we see that for $p = 1$ the expectation value is not even close to 0. This shows us the difficulty of the Exact Cover Problem.

4.3 Minimizing the Gibbs objective function

Another approach to this problem is to modify the function that is going to be optimized (Li Li et al. [11]).

The objective function that we will use now is

$$f(\eta) = -\log\left\langle e^{-\eta H_C}\right\rangle \tag{z.10}$$

We see that the above objective function has a minimum at $f(\eta) = 0$ because $H_C = 0$ will give us $e^{-\eta H_C} = 1$ and then $f(\eta) = -\log 1 = 0$. So our goal remains the same, try and get $f(\eta)$ as close to 0 as possible.

Here η is a hyperparameter which we define it's value with respect to the problem we want to solve. Essentially $f(\eta)$ is the cumulant generating function of the energy H_C. The cumulant generating function is the log of the moment generating function of H_C.To see how η affects the Gibbs objective function we must write the Taylor expansion of $f(\eta)$,

$$f(\eta) = \langle H_C\rangle\,\eta - \sigma^2\frac{\eta^2}{2} + \kappa_3\frac{\eta^3}{6} + \dots \tag{z.11}$$

where σ^2 is the variance of H_C. We see from (z.11) that for small η, namely $\eta < 1$, all the terms after $\langle H_C\rangle\,\eta$ contains $\eta^2, \eta^3, \dots$ that eventually goes to 0. So if we choose to minimize $f(\eta)$ for small η it will be the same as minimizing $\langle H_C\rangle$ since only the first term of $f(\eta)$ will survive. For bigger η the other cumulants affect the objective function.

This objective function can serve two purposes: (i) we can use $\eta < 1$ and minimize the expectation value of the quantum Ising Hamiltonian $\langle H_C\rangle$ and (ii) we can use $\eta \geq 1$ and maximize probability of finding the ground state ($H_C = 0$). Note here that the quantum circuit as well as equations (z.3) to (z.6) that we calculated in the analysis before stays the same in this analysis.

First let's see how the Gibbs objective function bounds the probability of finding the ground state. Suppose

$$P(X \leq \alpha) = P(e^{-tX} \geq e^{-t\alpha})$$

where t is a parameter, X is a random variable and α is a constant. Note here that e^{-tX} is a non-negative function of a random variable and therefore we can apply the Markov's inequality.

$$P(e^{-tX} \geq e^{-t\alpha}) \leq \frac{E[e^{-tX}]}{e^{-t\alpha}}$$

By applying the Markov's inequality we obtained the so called generic Chernoff bound for the random variable X. Now if we replace $x = H_C$,$\alpha = E_0$ and $t = \eta$ we have

$$P(H_C \leq E_0) \leq \frac{E[e^{-\eta H_C}]}{e^{-\eta E_0}} \Leftrightarrow$$

$$P(H_C \leq E_0) \leq e^{\eta E_0} \left\langle e^{-\eta H_C} \right\rangle \Leftrightarrow$$

$$P(H_C \leq E_0) \leq \left\langle e^{-\eta (H_C - E_0)} \right\rangle \tag{z.12}$$

We have proved that we can bound the probability of low energy from above. So if we choose an η that minimizes the right side of $(z.12)$ we will get the maximum of the left side, since we have the minus sign in the exponent of the right side.

To better understand the Gibbs objective function we will use several different values of η, namely $\eta = 500, \quad 250, \quad 100, \quad 50, \quad 1, \quad 0.1, \quad 0.01$.

(Below) we give the plot for $f(\eta)$ for different values of p. Here we see that smaller values of the Gibbs function are obtained for smaller η. In combination with the (figure) of the expected value of the cost Hamiltonian, we see that smaller $f(\eta)$ corresponds to smaller expected value.

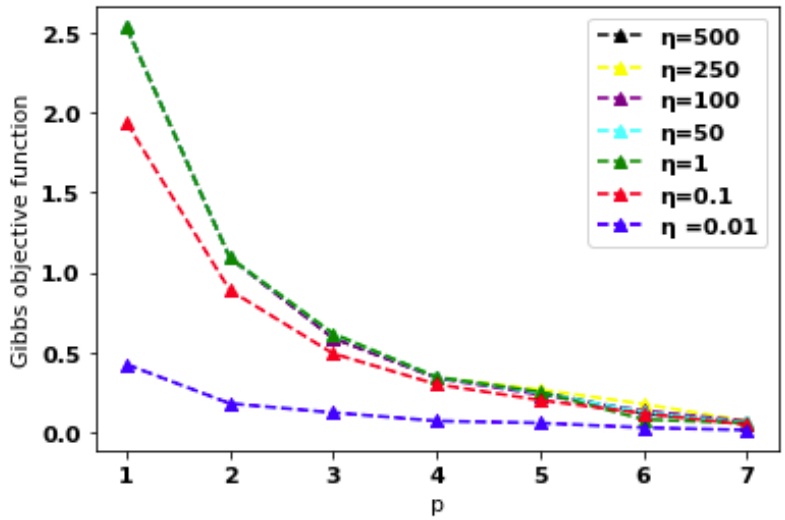


Figure 4.6: Gibbs objective function $f(\eta)$.

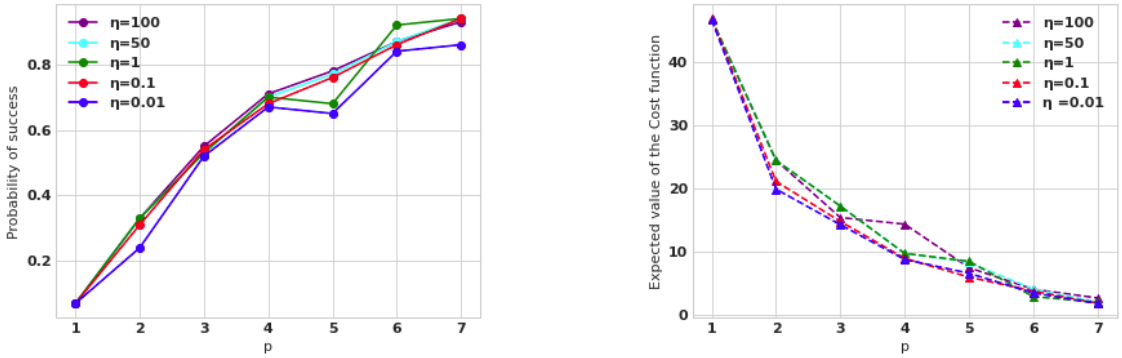


Figure 4.7: Right : Expected value of the cost Hamiltonian, $\langle H_C \rangle$. Left : Probability of success $|\left\langle 00001111 \middle| \psi_p(\vec{\gamma}, \vec{\beta} \right\rangle|^2$.

In the (figure) above it is worth noticing that the minimum expectation value comes from $\eta = 0.01$. Also for $\eta = 0.01$ we see the minimum probability of success to be obtained. This is happening because small expectation value does not necessarily mean high probability of success. For example we can find a state with small expected value but this state might be a superposition of some states and therefore the probability that we measure the correct solution out of this state could be small.

We also see that for $\eta \geq 1$ the probability of success is at the highest value for several p.

(Below) we compare the results from (Minimizing the expected value of the quantum Ising Hamiltonian) with the results we got from minimizing the Gibbs ocjective function.

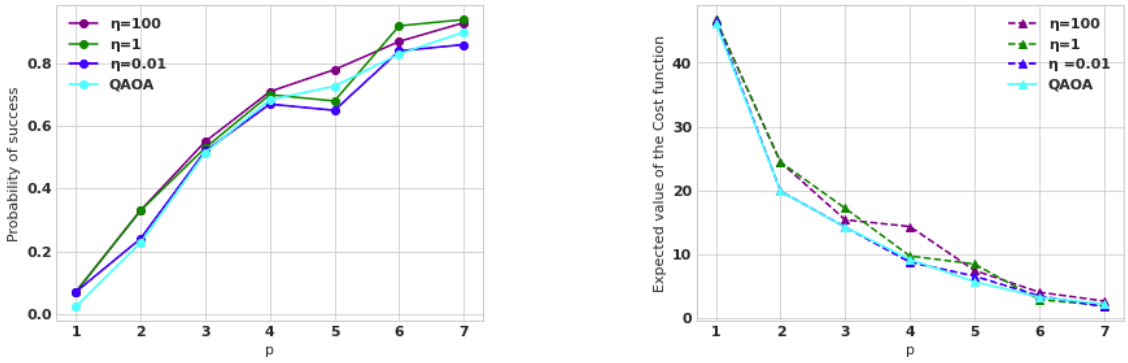


Figure 4.8: Comparison of the Gibbs method and the conventional QAOA method which we mark as QAOA in the figure. Right : Expected value of the cost Hamiltonian, $\langle H_C \rangle$. Left : Probability of success $|\left\langle 00001111 \middle| \psi_p(\vec{\gamma}, \vec{\beta}) \right\rangle|^2$.

Overall we see that for $\eta \geq 1$ we have bigger probability of success than the QAOA method starting even for $p = 2$, which is why we tried this method in the first place. Also we see that the expectation value in this case is smaller for the QAOA method for up to $p = 3$ but for $p \geq 6$ the difference is very small. With this in mind we can see that the Gibbs objective function is more flexible than the conventional QAOA method.

In the original paper from Pontus Vikstål et al. [15] for the TAP problem, they mention this method for future work. The reason why I tested this method is that the objective function is the same as in the previous section for the TAP, we just take the log of the expected value of the exponent of the cost Hamiltonian. Indeed we see better results with respect to the probability of success.

Chapter 5

Conclusion and future work

In this thesis we focused on solving optimization problems using quantum algorithms. In particular we analyze the Quantum Approximate Optimization Algorithm (QAOA) and the Variational Quantum Eigensolver (VQE) for the MAXCUT problem in complete graphs. In addition we analyze the performance of QAOA in an instance of the Tail Assignment Problem called the Exact Cover Problem. We test the algorithm using the conventional method of minimizing the expected value of the cost Hamiltonian and also we use the method of minimizing the Gibbs objective function in an 8 route, 80 flights graph.

The results we got from the conventional method of QAOA coincides with the ones found at Pontus Vikstål et al. [15]. In the conclusion of that paper, they mention the Gibbs objective function and that results from this objective function will be better with respect to the success probability. In the results we got the success probability was higher than the conventional method of QAOA ((figure)). Although we tested the Gibbs objective function in one graph, further testing must be done.

5.1 Future work

In the Tail Assignment Problem, the most computationally heavy task was to optimize the variational parameters $\vec{\gamma}, \vec{\beta}$. Even for a small number of qubits (n=8) and parallel optimization using 4 CPUs, the above task took hours for $p > 5$. There exist ways that we can reduce this time. For example in Pontus Vikstål et al. [15] they have found patterns in the variational parameters that helped to speed up the optimization time. Although patterns might exist in particular optimization problems, in the TAP it allowed them to run the quantum circuit for instances of 8,15,25 qubits and for $1 \leq p \leq 20$, so I think it is worth exploring. Another perspective is to change the objective function. In this thesis I used the Gibbs objective function but there exist also a CVaR objective function (Panagiotis Kl. Barkoutsos et al. [16]) and an evolving CVaR objective function (Ioannis Kolotouros and Petros Wallden [10]) that produces better results for VQE and QAOA than the conventional methods. Also work has been done by Benjamin Tan and Dimitris Angelakis et al. [12] at Angelakis Research Group, where they cut down the number of qubits for VQE from n to $log_2 n + 1$ allowing for real world problems to be considered. 10.000 variables for example require of the order of few dozen qubits at the minimal enconding.

Also it is worth exploring these quantum algorithms in the context of other optimization problems including finance and route optimization, as I listed in the (Appendix). The most industrial one is the Vehicle Routing Problem for which QAOA have been applied to by Stuart Hardwood et al. [17].In my future MSc studies, I would like to explore more of these problems with a focus to reduce the qubit requirements so near term quantum processors can be used to attack real world problems and build a systematic approach extending the work by Tan et al. [12].

Chapter 6

Appendix

6.1 Classical optimization problems and solutions

6.1.1 0/1 Knapsack problem

Imagine a scenario where you want to go camping with your friends and before you leave you want to prepare a knapsack or backpack to get with you. Ideally you want to have all the items that will give you comfort on your camping but you also don't want to be overweight since you will probably walk a lot! So the optimization problem that is formed here is the following, you want to maximize the items in your backpack in order for you to be comfortable in the camping with the constraint that these items will not weight more than you can carry.

To solve this problem we must first make a few definitions from Professor John Guttag lectures [18]. We will define an item to be inserted in the backpack as a tuple described by two variables

$$\text{Item} =< \text{value,weight} >$$

where the value is a positive number that represents the importance of the item on your trip, for example water should be of high importance while a hairdryer should be of low importance. The weight variable is a positive number which represents the weight of the item. We will also represent

the maximum weight that you can carry as a global integer

W as the max weight of the backpack

It is also good to have all the available items in a vector so

I as the vector of available items

We will also define a vector V which

V[i] = 1 if I[i] has been taken , V[i] = 0 if I[i] has not been taken

Now we can define the objective function that we want to maximize as

$$\sum_i V[i]I[i].\text{value}$$

where $I[i]$.value is the value component of the item. Essentially the objective function tells us that we want to maximize the value of the items we take. To see this more clearly note that if we don't take the item i then $V[i] = 0$, so the sum above contains only items that we put in the backpack.

The constraint of this problem is that we want our total backpack weight to be less or equal to the weight W that we can carry.

$$\sum_i V[i]I[i].\text{weight} \leq W$$

So to solve this problem we must look at all the possible V vectors that are created. So if we have n items then we have 2^n different binary vectors V. With this in mind we can see that for $n = 50$ the algorithm will take years to finish. Although by using dynamic programming for this algorithm reduces its complexity to O(nW),polynomial time with respect to the max weight.

6.1.2 Supply chain problem

Supply chain optimization problems occurs in large corporations where hundreds of facilities (distributors,plants,suppliers) are involved. The goals of corporate are to provide customers with the products that they need in the most efficient way possible. In order to do so there are a lot of decisions that must be made from the corporate as discussed by Arthur F. Veinott, Jr. at [19]. Some of these decisions are how much, when and where to produce the product; how much and when to ship from one facility to another; how much,when and where to store the product; how much,when and where to charge for products. Another very important factor that supply chain must account for is uncertainty.

First we will start by formulating the supply chain optimization problem for an n-period without uncertainty. In this problem we have the number of products for the i-th period as x_i, the number of products in storage for the i-th period as y_i and the given demand for the product in the i-th period as s_i. We also define the cost of producing k products as $c_i(k)$ and the cost of storing k products as $h_i(k)$. With this in mind the cost function we want to minimize is

$$C(x, y) = \sum_{i=1}^{n} c_i(x_i) + h_i(y_i)$$

, and is read as the sum over n periods of the cost of producing x_i products plus the cost of storing y_i products.

The constraint that follow this problem is

$$x_i + y_{i-1} - y_i = s_i, \quad i = 1, \ldots, n$$

, which means that we want the demand s_i to be equal to the x_i products we produce plus the y_{i-1} products which have been stored in the previous period minus the y_i products we have stored in the current period.

For the problem to be complete we must also assume that all the above discrete variables are non negative.

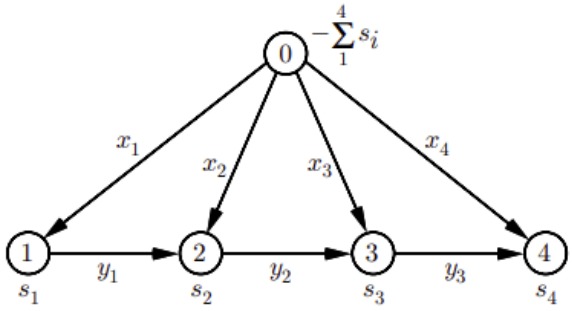


Figure 6.1: Minimum cost network flow problem.The edges (0,1),(0,2),(0,3),(0,4) are called the production edges and represent the production at period i. The edges (1,2),(2,3),(3,4) are called storage edges and represent the products in store between the specified periods.

Such linear cost problems can be easily mapped to a network flow problem as seen in figure (above). The solution here is to find a minimum cost chain or directed path on the graph from node 0 to i while satisfying the demand at i by shipping along that chain.

By using dynamic programming we can define the minimum cost at period i as

$$C_i = \min(c_i, h_{i-1} + C_{i-1}), \quad i = 1, \ldots, n$$

This tells us that for each period i we choose the smallest between two options : i) production cost c_i is sufficient for the demand at period i, production edge is sufficient or ii) storage cost h_{i-1} and total cost C_{i-1} at previous period, storage edge $(i-1, i)$ and previous node are sufficient. This recursive method allow us to split the bigger problem of finding the minimum cost into sub problems for which we find the minimum cost.

The complexity of this algorithm is $O(n)$ because at worst we have to calculate n different C_i. Each calculation consists of one addition and one comparison.

6.1.3 Portfolio optimization problem

Suppose you are an investor with some initial capital in your disposal. The problem is that you want to invest your capital in a number of different securities so that you have certain return with

minimum risk involved. There are two main types of risks involved with the above problem; (1) the risk of having less return as expected and (2) the risk of a not well-diversified portfolio which comes from not having enough different types of securities in the portfolio.

A very compact portfolio optimization model was introduced by Harry Markowitz in a 1952 essay called Modern Portfolio Theory (MTP) or mean-variance analysis. Before explaining the model we must first define the variables that we will work with (Hassan Rahnama [20]).

Let x_i represent the proportion of the total capital invested in security i. It is fair to assume that this x_i can be represented by a random variable. With this in mind we denote μ_i as the expected return of x_i and σ_i^2 the covariance for each security $i = 1, \ldots, n$. We can now define the portfolio which is a vector of x_i and we will denote as $\bar{x} = (x_1, x_2, \ldots, x_n)$. So we can define the total expected return and variance of the portfolio as follows

$$E[\bar{x}] = x_1\mu_1 + x_2\mu_2 + \cdots + x_n\mu_n = \mu^T\bar{x}$$

$$\text{Var}[\bar{x}] = \sum_{i,j} \rho_{ij}\sigma_i\sigma_j x_i x_j = \bar{x}^T\Sigma\bar{x}$$

where $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ is the vector of the expected values, ρ_{ij} is the correlation coefficient between securities i and j ($\rho_{ii} = 1$) and $\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$ is the covariance matrix of the portfolio.

The model introduced by Harry Markowitz can be formulated in 3 ways. First, we minimize the variance or risk of the portfolio while bounding the expected return of the portfolio. Second, we maximize the expected return of the portfolio while bounding the risk or variance of the portfolio. Lastly, we combine the previous two and also using a new variable q which is called risk tolerance.

• Minimization the portfolio variance

Here the cost function to be minimized is

$$\min_{\bar{x}} \quad \bar{x}^T\Sigma\bar{x}$$

In order to solve the optimization problem we need to bound the expected return. For this

reason we use R, so suppose we want our expected return to be greater of equal than R. Therefore

$$E[\bar{x}] = \mu^T \bar{x} \geq R$$

Two constraints that goes along with our problem are

$$\sum_n x_i = 1$$

which means that the proportions of the total capital should sum to 1 .Also we must assume that $x_i \geq 0 \quad \forall i$.

• Maximization of the total expected outcome

Here the cost function to be maximized is the expected outcome of our portfolio

$$\max_{\bar{x}} \mu^T \bar{x}$$

while bounding the variance of the porfolio by a constant σ^2

$$\bar{x}^T \Sigma \bar{x} \leq \sigma^2$$

The two extra constrains from above also apply in this case.

• Combination of the above

The cost function to be minimized here is

$$\min_{\bar{x}} \quad \bar{x}^T \Sigma \bar{x} - q\mu^T \bar{x}$$

where $q \in [0, \infty)$ is the risk tolerance and it is assumed known for the problem. We see that for $q = 0$ we get the estimator which minimized the risk (1st bullet) and as q grows we get a mixture of the problem.

Also in this model we have the two extra constraints

$$\sum_n x_i = 1$$

and $x_i \geq 0 \quad \forall i$.

In order to solve the above models we can either make use of non linear programming or swap the inequality constraints with equality constraints and solve using the lagrance multipliers.

6.1.4 Vehicle routing problem

The Vehicle Routing problem (VRP) is defined on a graph $G = (V, A)$ where $V = \{u_1, u_2, \ldots, u_n\}$ is a set of vertices and $A = \{(u_i, u_j) : i \neq j, u_i, u_j \in V\}$. We will call vertex v_1 the "depot" which means the starting node of our m vehicles. The rest of the nodes can be cites or customers. The VRP is the problem of finding m vehicle routes of minimum total cost,where the total cost can be travel costs or travel time or distance,starting and ending at the depot, such that each remaining vertex is visited exactly once by one vehicle and satisfying some constraints.

The most common type of constraints are : time window constraints when each vertex-costumer must be visited within a specified time interval, duration constraints when the total length of a route must be less or equal than some constant L, capacity constraints when the vehicle for example has capacity Q then the "goods" that the vehicle picked up on a route must be less or equal than Q.

This problem can be thought of as m different Travelling Salesperson problem (TSP). For the TSP prolem there exists many algorithms and APIs in this day capable of producing good solutions. The best known algorithm for the VRP is called the Taburoute algorithm of Gendreau,Hertz and Laporte (1994). Recently Stuart Harwood et al. [17] tested the QAOA on the VRP and had some promising results.

There exists many different formulations for the VRP that they differ with respect to the constraints. We will consider the VRP problem with time windows constraints. Here the decisions to be made are whether routes are travelled or not.

We will define a route as a sequence of nodes $(v_1, v_2, \ldots, v_P)$ for some route-specific positive integer P.Note that a route must begin and end at the depot so $v_1 = v_p = d$.

For each feasible route the sum of the product delivered/picked up must be physically possible :$0 \leq Q_0 + \sum_{j=2}^{p} q_{v_j} \leq Q$, where Q_0 is the initial cargo of the vehicle, Q is the max cargo that the vehicle can carry and q_{v_j} is the cargo that a vehicle received at node v_j.

We associate each node v_j with a time window $[\alpha_{v_j}, \beta_{v_j}]$. We can let $T_{v1} = 0$ be the starting time of the depot node, then the arrival time at node v_{p+1} is given by $T_{v_{p+1}} = \max\{\alpha_{v_{p+1}}, T_{v_p} + t_{v_p,v_{p+1}}\}$ where $T_{v_p} + t_{v_p,v_{p+1}}$ is the starting time at node v_p plus the time between nodes v_p and v_{p+1}. Although a vehicle cannot arrive after the time window has closed i.e. $T_{v_p} \leq \beta_{v_p}$.

We define the set of routes to be R. If route $r \in R$ has node sequence $(v_1, v_2, \ldots, v_P)$, then it has cost $c_r = \sum_{p=1}^{P-1} c_{v_p,v_{p+1}}$.

Finally, we define $\delta_{v,r}$ to be equal to 1 iff route r contains node v and 0 otherwise.

We also have the binary variable x_r which is equal to 1 iff a vehicle travels route r and 0 otherwise. So we have the objective function to be minimized

$$\sum_{r \in R} c_r x_r$$

and the constraints to be

$$\sum_{r \in R} \delta_{v,r} x_r = 1 \quad \forall v \in V$$

6.2 The cost function as an eigenvalue of the cost hamiltonian matrix

In this section we will prove how we can get the value of the cost function of the MAXCUT problem as an eigenvalue of the cost Hamiltonian matrix. We have

$$C(s) = \frac{1}{2} \sum_{(i,j) \in E} (1 - s_i s_j)$$

where $s \in \{-1, 1\}^n$, $G = (V, E)$ is a graph with $n = |V|$ the number of nodes in the graph and E is the set of edges. We can construct the cost Hamiltonian for the MAXCUT if we replace the s_i variables with σ_i^z (pauli Z-matrix acting on qubit i). So we have the following cost Hamiltonian

$$C = \frac{1}{2} \sum_{(i,j)\in E} (\mathbb{I} - \sigma_i^z \sigma_j^z)$$

What we want from the cost Hamiltonian is to get the cost of each of the possible 2^n bit-strings as an eigenvalue of the cost Hamiltonian matrix. We can see this happening by applying the cost Hamiltonian on an arbitrary bit-string

$$C\,|x\rangle = \frac{1}{2} \sum_{(i,j)\in E} (\mathbb{I} - \sigma_i^z \sigma_j^z)\,|x\rangle \Leftrightarrow$$

$$C\,|x\rangle = \frac{1}{2} \sum_{(i,j)\in E} |x\rangle - \sigma_i^z \sigma_j^z\,|x\rangle \Leftrightarrow$$

$$C\,|x\rangle = \frac{1}{2} \sum_{(i,j)\in E} |x\rangle - (-1)^{x_i}(-1)^{x_j}\,|x\rangle \Leftrightarrow$$

$$C\,|x\rangle = \frac{1}{2} \sum_{(i,j)\in E} (1 - (-1)^{x_i}(-1)^{x_j})\,|x\rangle \Leftrightarrow$$

$$C\,|x\rangle = C(x)\,|x\rangle$$

where we have used $\sigma_i^z\,|x_i\rangle = (-1)^{x_i}\,|x_i\rangle$ and that $x_i = 0 \rightarrow (-1)^{x_i} = 1 = s_i, \quad x_i = 1 \rightarrow (-1)^{x_i} = -1 = s_i$ so that we can write the cost function that depends on $x \in \{0, 1\}^n$ as $C(x) = \frac{1}{2}\sum_{i,j\in E}(1 - (-1)^{x_i}(-1)^{x_j})$.

So we proved that after applying C on an arbitrary state we can get the bit-string's cost function as an eigenvalue.

6.3 Classical pre-processing on bounded degree graphs

Now we will see a way to compute the expectation value of the cost Hamiltonian on a classical computer. Note that this computation occurs when we consider the MAXCUT problem for graphs with bounded degree. First we write the cost Hamiltonian as a sum of local terms,

$$C = \sum_{(j,k)} C_{(j,k)}$$

where (j,k) refers to an edge from the total edge set of the graph and $C_{(j,k)} = \frac{1}{2}(\mathbb{I} - \sigma_j^z \sigma_k^z)$

For $p = 1$ we have the expected value of the cost Hamiltonian as

$$F_1(\gamma,\beta) = \left\langle \psi(\gamma,\beta) \right| C \left| \psi(\gamma,\beta) \right\rangle$$

Due to the linearity of the expectation value

$$E[\sum_{(j,k)} C_{(j,k)}] = \sum_{(j,k)} E[C_{(j,k)}]$$

We can write the expected value of the cost Hamiltonian as the sum of some local expected values

$$F_1(\gamma,\beta) = \sum_{(j,k)} \langle s| U^\dagger(C,\gamma) U^\dagger(B,\beta) C_{(j,k)} U(B,\beta) U(C,\gamma) |s\rangle$$

where we have expanded $\left|\psi(\gamma,\beta)\right\rangle = U(B,\beta)U(C,\gamma)\left|s\right\rangle$ and it's complex conjugate $\left\langle\psi(\gamma,\beta)\right| = \langle s| U^\dagger(C,\gamma)U^\dagger(B,\beta)$ with $|s\rangle = |+\rangle_1 |+\rangle_2 \ldots |+\rangle_n$ to be the equal superposition of n qubits.

In this moment we must remember that the goal of this algorithm, in the MAXCUT problem, is to maximize the above expected value. By expressing this value into the sum of local terms our goal is to divide the bigger problem (computing the expected value of the cost Hamiltonian) to smaller problems (compute local terms of the expected value of the cost Hamiltonian). By doing so we can restrict $F_1(\gamma,\beta)$ to depend on nodes that exist on some sub graphs instead of all the nodes on the graph. This is more computationally efficient and we can do it before we even run QAOA

on a quantum computer since it depends only on sub graphs.

Before explaining any further, let's first see what lead the authors to search for this kind of solution to the problem [7]. We analyze a single term of the sum of $F_1(\gamma, \beta)$. That is

$$U^\dagger(C,\gamma)U^\dagger(B,\beta)C_{(j,k)}U(B,\beta)U(C,\gamma) \tag{y.2}$$

where I deliberately left out $|s\rangle$ and $\langle s|$ since the analysis will be the same if we include them or not.

Let's write the operator $U(B,\beta) = e^{-i\beta\sum_{j=1}^{n}\sigma_j^x}$ in $(y.2)$

$$U^\dagger(C,\gamma)\underbrace{e^{i\beta\sum_{j=1}^{n}\sigma_j^x}C_{(j,k)}e^{-i\beta\sum_{j=1}^{n}\sigma_j^x}}U(C,\gamma)$$

Focus on the under brace of the above equation. Note that terms from the right exponent that do not involve qubits j and k commute through $C_{(j,k)}$ and cancel out. This happens because $C_{(j,k)}$ only acts on qubits j and k leaving the other unchanged while $U(B,\beta)$ acts on all qubits. So we can write the above equation as

$$U^\dagger(C,\gamma)\underbrace{e^{i\beta(\sigma_j^x+\sigma_k^x)}C_{(j,k)}e^{-i\beta(\sigma_j^x+\sigma_k^x)}}U(C,\gamma)$$

Factors of $U(C,\gamma)$ which does not involve qubits j and k will also commute through and cancel out. We see that each term of the expected value of the cost Hamiltonian depends on edge (j,k) and edges adjacent to this one (for $p = 1$). For a bounded degree graph that means that these terms are independent of the graph size which give us the right to efficiently compute these terms on a classical computer. For every p that we add, edges that contribute to the total expected value are at most p steps away from the edge we are considering.

For example let's consider graphs with bounded degree 3. For any graph there exists 3 types of sub graphs that are shown in the (figure) below

With this in mind we can define the expected value of the cost Hamiltonian with respect to these sub graph types. We define

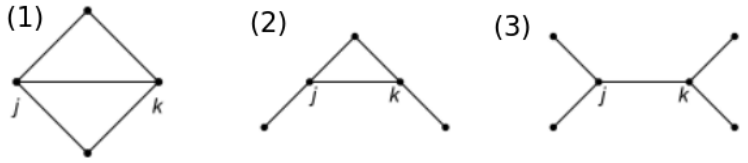


Figure 6.2: (1)Nodes j and k have 3 connected edges,(2)Nodes j and k have 2 connected edges,(3)Nodes j and k have 1 connected edge

$$C_G = \sum_{(l,l')\in G} C_{(l,l')}$$

which is the cost for a sub graph G. We will also define operators $U(C,\gamma), U(B,\beta)$ for a sub graph G as

$$U(C_G,\gamma) = e^{-i\gamma C_G}$$

and

$$U(B_G,\beta) = e^{-i\beta B_G}$$

where $B_G = \sum_{j\in G} \sigma_j^x$. At last we define the state $|s,G\rangle$ to be the superposition of the qubits that are in the sub graph G,

$$|s,G\rangle = \prod_{l\in G} |+\rangle_l$$

Now we can write the expectation value of each edge (j,k) as the expectation value on the sub graph. Each edge (j,k) is associated with a sub graph $g(j,k)$ with a local expectation value with respect to the sub graph depending state. We define this local expectation value as

$$f_g(\vec{\gamma},\vec{\beta}) = \left\langle s,g(j,k)\right| U^\dagger(C_{g(j,k)},\gamma_1)\dots U^\dagger(B_{g(j,k)},\beta_p) C_{(j,k)} U(B_{g(j,k)},\beta_p)\dots U(C_{g(j,k)},\gamma_1)\left|s,g(j,k)\right\rangle$$

Recall the expectation value of the cost Hamiltonian

$$F_p(\vec{\gamma},\vec{\beta}) = \sum_{(j,k)} \langle s| U^\dagger(C,\gamma_1)\dots U^\dagger(B,\beta_p) C_{(j,k)} U(B,\beta_p)\dots U(C,\gamma_1) |s\rangle$$

If two different terms of the sum produce sub graphs that are isomorphic then their contribution to the total expected value will be the same. This is due to the fact that isomorphic graphs are edge preserving so they will have the same cost. With this in mind we can count the available sub graphs that occur in the problem graph (for the 3-regular graphs we only have 3 different kind of sub graphs), compute the sub graph dependent expectation value of the cost and add them to obtain the total expectation value. We can write the above in the following equation

$$F_p(\vec{\gamma},\vec{\beta}) = \sum_g w_g f_g(\vec{\gamma},\vec{\beta})$$

where w_g is the number of occurrences of sub graph g in the original graph.

We see that $f_g(\vec{\gamma},\vec{\beta})$ depends only on the qubits/nodes of the sub graph g. The maximum number of qubits in a sub graph comes when the sub graph is a tree, independent of the number of nodes and clauses in the problem graph. Using this notation we can also read w_g directly from the problem graph and compute $F_p(\vec{\gamma},\vec{\beta})$ in a classical computer with matrices and vectors whose dimensions are independent of the problem size. The only dependence on the problem size exists in the weights w_g.

6.4 Connection of QAOA and the adiabatic quantum algorithm

Here we note the connection of QAOA to the Adiabatic Quantum Algorithm (Edward Fahri et al. [21]). This algorithm is designed to find the best solution if the time T that we run the system is long enough. This creates a continuous path that depends on time T from the starting state to the optimal state. QAOA can produce a Trotterized approximation to this path by splitting

the continuous path into smaller discrete ones. This happens by applying $U(C,\gamma), U(B,\beta)$ on the starting state for each p, where the sum of the angles is the total run time T. This suggests that $\lim_{p\to\infty} M_p = \max_z C(z)$. Although due to its approximation nature we don't necessarily need to make p large to get a good solution.

Bibliography

[1] David Deutsch and 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.

[2] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332.

[3] Lov K Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[4] Michael A Nielsen and Isaac L Chuang. "Quantum computation and quantum information". In: *Phys. Today* 54.2 (2001), p. 60.

[5] Dimitri Bertsekas. *Network optimization: continuous and discrete models*. Athena Scientific, 1998.

[6] Fred Glover, Gary Kochenberger, and Yu Du. "A tutorial on formulating and using QUBO models". In: *arXiv preprint arXiv:1811.11538* (2018).

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

[8] Pablo Dıez-Valle, Diego Porras, and Juan José Garcıa-Ripoll. "Quantum variational optimization: the role of entanglement and problem hardness". In: *arXiv preprint arXiv:2103.14479* (2021).

[9] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145.

[10] Ioannis Kolotouros and Petros Wallden. "An evolving objective function for improved variational quantum optimisation". In: *arXiv preprint arXiv:2105.11766* (2021).

[11] Li Li et al. "Quantum optimization with a novel gibbs objective function and ansatz architecture search". In: *Physical Review Research* 2.2 (2020), p. 023074.

[12] Benjamin Tan et al. "Qubit-efficient encoding schemes for binary optimisation problems". In: *Quantum* 5 (2021), p. 454.

[13] Richard M Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[14] Donald Knuth. *Algorithm X*. URL: `https://en.wikipedia.org/wiki/Knuth%27s_Algorithm_X`.

[15] Pontus Vikstål et al. "Applying the quantum approximate optimization algorithm to the tail-assignment problem". In: *Physical Review Applied* 14.3 (2020), p. 034009.

[16] Panagiotis Kl Barkoutsos et al. "Improving variational quantum optimization using cvar". In: *Quantum* 4 (2020), p. 256.

[17] Stuart Harwood et al. "Formulating and solving routing problems on quantum computers". In: *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–17.

[18] John Guttag. *6.00 SC Introduction to Computer Science and Programming*. 2011. URL: `https://www.youtube.com/watch?v=BRjwkgQct28&t=2494s&ab_channel=MITOpenCourseWare`.

[19] A Veinott. "Lectures in supply-chain optimization". In: *Management Science and Engineering* 361 (2005).

[20] Hassan Rahnama. "A Portfolio Optimization Model". PhD thesis. Ecole Polytechnique, Montreal (Canada), 2016.

[21] Edward Farhi et al. “Quantum computation by adiabatic evolution”. In: *arXiv preprint quant-ph/0001106* (2000).