An Introduction to Quantum Optimization Approximation Algorithm

Qingfeng Wang, Tauqir Abdullah

December 14, 2018

Contents

1 Introduction

Quantum Approximation Optimization Algorithm (QAOA) is one of the algorithms that can be implemented in the near-term quantum computer and regarded as one of the most promising algorithms to demonstrate quantum supremacy. QAOA is an approximation algorithm which means it does not deliver the 'best' result, but only the 'good enough' result, which is characterized by a lower bound of the approximation ratio.

In this report, we will introduce how QAOA works and some of explicit applications of QAOA. This report is organized in the following way. First, we gave a recent development of the method. Then, we gave the extensive explanation for how QAOA works and how it can be applied to MaxCut problem and traverse Ising filed model. Lastly, we present the conclusion and remarks.

2 Development for QAOA

QAOA was first brought by Farhi *et* al.[1] in 2014. In the paper they gave the first heuristic of QAOA and applied QAOA on MaxCut problem. Specifically, they showed the complexity of problem depends on p rather than n. If $p = 1$, for a 3-regular graph they can achieve 0.6942 times the optimal cut. At a later time, the MaxCut problem was further discussed by Wang *et* al.[2]. Farhi *et* al. also mentioned relationship between QAOA and quantum adiabatic algorithm (QAA) such they would consider QAOA as a Trotterized QAA so that for $p \to \infty$ they are equivalent.

Immediately after the first appearance of original paper, Farhi *et* al.[3] applied QAOA on another combinatorial problem Max E3LIN2. The result is not too interesting as even though it shows the capability of satisfying number of linear equations more than half, it has no significant advantage over classical algorithms. (There is a interesting story behind this, that initially this QAOA algorithm beat the best classical algorithm but immediately been fight back by a bunch of computer scientists who designed a better classical algorithm for E3LIN2.) However, this application of QAOA attracted attention Lin *et* al.,[4] who refined constraint problems in QAOA by defining the 'typicality' in QAOA notion and and showed $\mu + \Omega(1/\sqrt{D})$ fraction of constraints can be satisfied (see original paper for meaning of labels).

Later, Farhi *et* al.[5] showed the reason why QAOA is one of the methods that can help achieving so-called quantum supremacy. The idea is to show that even the lowest depth of QAOA can not be efficiently simulated using classical computer by arguing that if so then it means $P = NP$. This means QAOA is a great candidate for early demonstration of quantum supremacy.

Unlike QAA, QAOA requires a lower coherent time. This advantage get more people interested. Wecker *et* al.[6] used a variant of QAOA to solve MAX-2-SAT problem. Instead of finding expectation of operator, they wish to maximize the overlap of between output and ground state and QAOA indeed gave such a result.

The understanding of QAOA was further deepened by Yang *et* al.[7]. They called the way that QAOA applying abrupt operations on state is a 'bang-bang' (square pulse) form, and which happens to be the optimal evolution for fixed computation time. So the rigidity of QAOA is justified.

Even more interestingly, Jiang *et* al.[8] applied QAOA on unstructured search and achieved a complexity only $\sqrt{2}$ larger than the Grover's algorithm.

Taking QAOA as the leading candidate to show the power of quantum computing, Hadfield *et* al.[9, 10] showed how to implement QAOA with soft and hard constraints. Specifically, they showed how to implement on five problems, *i*.e., Max Colorable Subgraph, Max Colorable Induced Subgraph, Min Graph Coloring, Traveling Salesman Problem and Single Machine Scheduling. Notice that, the authors rephrased the acronym from Quantum Approximation Optimization Algorithm to Quantum Alternating Operator Ansatz to generalize the applicability of QAOA. Six month later, Babej *et* al. echoed this idea and used the Hadfield version of QAOA to solve lattice protein folding problem.[11]

Recently, QAOA is applied to produce non-trivial quantum states such as Greenberger-Horne-Zeilinger (GHZ) state or quantum critical ground state. Usually they would use operators such as in traverse field Ising model (TFIM). People are interested in how to implement QAOA in experiments more efficiently to obtain the ground state of TFIM. Ho *et* al. [12] shows that using QAOA, one can implement the circuit with depth $O(L)$ where L is the linear dimension of system. In addition, soon after, Ho *et* al.[13] has shown an even more astonishing result that QAOA with Long Range Interactions can prepare GHZ in $O(1)$ depth and TFIM with $> 99\%$ fidelity on 100 qubits system with merely one iteration of QAOA.

Undoubtedly, QAOA is a successful quantum heuristic can be implemented on near-term quantum computer and has drawn many people's attention. It is expected there will be more theoretical improvement of QAOA as well as experiment implementation of QAOA.

3 Problem statement

Some of the explanation might be trivial to readers but they are important for beginners to understand the reasoning behind it. For such details, they will be moved to the supplementary part to make the report concise.

Suppose we wish to maximize a objective function $C = \sum_{k=1}^{m} C_m(z)$ where $z = z_1 z_2 \cdots z_n$ where $z_k \in \{0,1\}^n$. $C_m \in \{0,1\}$ is a function that queries string z to see if the substring satisfies certain property. For example, $C_1 = 1$ if $z_3 = 0$ & $z_5 = 1$ and $C_1 = 0$ otherwise. There are 2^n possible different z and our goal is to find a z that can maximize C. Classically, we have to query each of z which requires $O(2^n)$ queries. It is still unclear how to solve this problem exactly efficiently. If an approximate solution is all we need, then there are already many classical approximate algorithms available. However, we (probably) can do better using a quantum approximate optimization algorithm.

The first step is to treat the objective function C as an operator. Since

$$C\left|z\right\rangle = \sum_{k=1}^{m} C_k(z)\left|z\right\rangle = f(z)\left|z\right\rangle,$$

the C has eigenvectors $|z\rangle$ and eigenvalues $\sum_{k=1}^{m} C_k(z) = f(z)$. The eigenvalue set $\{f(z), z \in \{0,1\}^n\}$ can be ordered from smallest to largest. The largest value $f(z)$, which we can denote $f(z')$, is the $C_{\max}$, with corresponding eigenvector $|z'\rangle$. With basis $|z\rangle$, we can construct arbitrary state as general superposition state where a_z are 2^n coefficients that are properly normalized $|\psi\rangle = \sum_{z\in\{0,1\}^n} a_z\left|z\right\rangle \text{s.t.} \sum_{z\in\{0,1\}^n} |a_z|^2 = 1$. It is easy to check that $a_z = 1$ then it corresponds to basis state $|z\rangle$. For $|\psi\rangle$ we can construct expectation value of C, that is $\langle C\rangle = \langle\psi|C|\psi\rangle$. If $|\psi\rangle = |z\rangle$, $\langle C\rangle = \langle z|C|z\rangle = f(z)$, which can be evaluated classically if $|z\rangle$ is known. Finding the $|z\rangle$ that can maximize $C(z)$ corresponding to finding a state $|\psi\rangle$ that maximize $\langle C\rangle$. See in Supplementary Material (SM A.1)

$$\langle C\rangle = \langle\psi|C|\psi\rangle = \sum_{z\in\{0,1\}^n} f(z)|a_z|^2 \leq \sum_{z\in\{0,1\}^n} f(z')|a_z|^2 = f(z')\sum_{z\in\{0,1\}^n} |a_z|^2 = f(z')$$

So $\langle C\rangle_{\max} = f(z')$ and only when $|\psi\rangle = |z'\rangle$. In other words, if we can maximize and obtain $\langle C\rangle_{\max}$, then it means we are applying $|z'\rangle$, which means if we measure the state $|\psi\rangle$ with $|z\rangle$ basis we will obtain $|z'\rangle$ with probability 1. But in general, we don't know how to find such $\langle C\rangle_{\max}$ efficiently, we can only find a $\langle C\rangle$ that is close to $\langle C\rangle_{\max}$. That is to say we can only find a $|\psi\rangle$ that is a superposition of $|z\rangle$ basis and with a probablity $|a'_z|^2$ for obtaining $|z'\rangle$ and hoping that $|a'_z|^2$ is large enough so that after we measure $|\psi\rangle$ many times we might accidentally obtain $|z'\rangle$. Later we can see that in general $|a'_z|^2$ is so small that we can only hope to find a $|z\rangle$ that makes $C(z)$ close to $C(z')$.

Consider an example state $|\psi\rangle = |s\rangle$ which is an uniform superposition of all $|z\rangle$:

$$|s\rangle = H^{\otimes n}\left|0\right\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}\left|z\right\rangle$$

If we measure it in $|z\rangle$, each result is a single basis state $|z\rangle$, and each basis state $|z\rangle$ correspond to a $C(z)$. After we measure it in basis T times, we can obtained a distribution of basis states as well as distribution of $C(z)$ with maximum value $C(z'')$ and corresponding state $|z''\rangle$.

The average of the distribution will be

$$\langle C\rangle = \sum_{z\in\{0,1\}^n} f(z)|a_z|^2 = \frac{1}{2^n}\sum_{z\in\{0,1\}^n} f(z).$$

The probability of obtaining $|z'\rangle$ for each measurement is $\frac{1}{2^n}$ and we have

$$C(z'') \xrightarrow{T\to\text{inf}} C(z') = C_{\max}$$

which is no better than enumerating $|z\rangle$ or by random guess.

So, $|s\rangle$ state is not interesting, but we can somehow rotate $|s\rangle$ to make it closer to the $|z'\rangle$. To do this, we define two types of rotation unitary matrices $U(C,\gamma) = e^{-i\gamma C}$ and $U(B,\beta) = e^{-i\beta B}$.

3.1 Discussion about $U(C,\gamma)$

First we discuss operation $U(C,\gamma)$, which is defined:

$$U(C,\gamma) = e^{-i\gamma C} = e^{-i\gamma\sum_{\alpha=1}^{m} C_\alpha} \overset{T}{=} \Pi_{\alpha=1}^{m} e^{-i\gamma C_\alpha} + err$$

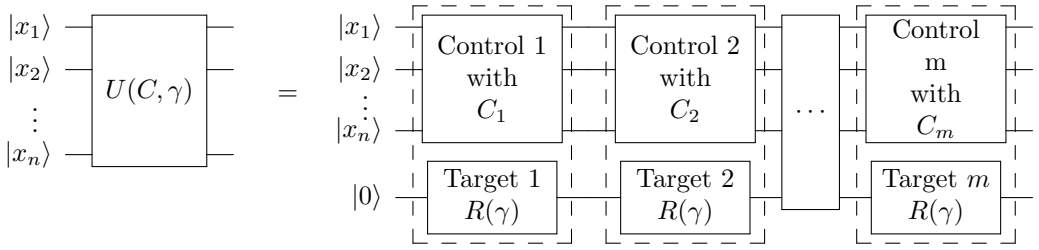
where T means Trotter decomposition and γ is an angle which we will come back later. If two C_α are commute then $err = 0$, which means the expression is exact without error. Since here we use a operator gives $[C_i, C_j] = 0$, we can write the exponential of summation as the product of exponentials . That is

$$U(C,\gamma) = e^{-i\gamma C} = \Pi_{\alpha}^{m} e^{-i\gamma C_\alpha}.$$

We can decompose $U(C,\gamma)$ into the product form because $U(C,\gamma)$ can be implemented by applying terms successively. (Detail in Supplementary A.2)

$$|\psi\rangle = \sum_{z\in\{0,1\}^n} a_z |z\rangle \xrightarrow{U(C,\gamma)} \sum_{z\in\{0,1\}^n} [\Pi_{\alpha=1}^{m} U(C_\alpha(z),\gamma)]\, a_z |z\rangle \text{, where: } \begin{cases} U(C_\alpha(z),\gamma) = 1, & \text{if } C_\alpha(z) = 0 \\ U(C_\alpha(z),\gamma) = e^{-i\gamma}, & \text{if } C_\alpha(z) = 1 \end{cases}$$

This means for a certain component $|z\rangle$, a phase $e^{-i\gamma}$ will be added in front of $|z\rangle$ for each of satisfied condition C_α. This can be easily translated into a explicit circuit using controlled-phase gates with the help of an ancilla qubit. Notice, an ancilla qubit is not always required depending on the C causes. A circuit would looks like this: See in Supplementary Material (SM A.2)



where $R(\gamma) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\gamma} \end{bmatrix}$ is a phase gate with $\gamma \in [0, 2\pi]$.

To analyze,

$$|\psi\rangle |1\rangle = |x_1 x_2 \cdots x_n\rangle |1\rangle = \sum_{z\in\{0,1\}^n} a_z |z\rangle |1\rangle .$$

For component $a_z |z\rangle |1\rangle$, if C_α is satisfied then $a_z |z\rangle \otimes R(\gamma) |1\rangle = a_z R(\gamma) |z\rangle \otimes |1\rangle$ which effectively change the phase; and if C_α is not satisfied then leave the component $|z\rangle$ alone.

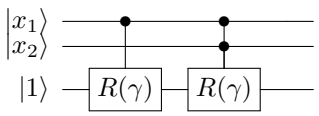
As an explicit example, suppose a state $|\psi\rangle = |x_1\rangle |x_2\rangle$, where $n = 2$ and in general $x_1 = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ and $x_2 = \beta_0 |0\rangle + \beta_1 |1\rangle$. Also we have $m = 2$ clauses:

$$\begin{cases} C_1 = 1, & \text{if } x_1 = |1\rangle \\ C_2 = 1, & \text{if } x_1 = x_2 = |1\rangle \end{cases}$$

which gives truth table $C_1(00) = 0, C_1(01) = 0, C_1(10) = 1, C_1(11) = 1$ and $C_2(00) = 0, C_2(01) = 0, C_2(10) = 0, C_2(11) = 1$. For a $U(C,\gamma)$ gate, we should expect the result: (Detail in SM A.2)

$$|x_1 x_2\rangle \xrightarrow{U(C,\gamma)} \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + R(\gamma)\alpha_1\beta_0 |10\rangle + R^2(\gamma)\alpha_1\beta_1 |11\rangle$$

Or eqivalently can construct circuit with the help of ancilla qubit $|1\rangle$ (Detail in SM A.2)



$$|x_1x_2\rangle|1\rangle \xrightarrow{\text{control}-R(\gamma)} \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + R(\gamma)\alpha_1\beta_0|10\rangle + R(\gamma)\alpha_1\beta_1|11\rangle$$
$$\xrightarrow{\text{CC}-R(\gamma)} (\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + R(\gamma)\alpha_1\beta_0|10\rangle + R^2(\gamma)\alpha_1\beta_1|11\rangle)|1\rangle$$

which is exactly the same as applying $U(C,\gamma)$. It also can be observed that if certain $|z\rangle$ satisfies $m(z)$ clauses, then it has a phase $R^{m(z)} = e^{-im(z)\gamma}$. Notice that, if we consider the control gate can be implemented in depth 1, then the depth of a single $U(C,\gamma)$ presented in this way is m. However, depth could be much larger since large control gates probably require a lot of basic gates. To sum up, by decomposing the C, we can implement gate $U(C,\gamma)$ using basis gates with depth m. It is worth noting here that different type of clauses C will likely require different type of gate implementations other than simple control gates, as mentioned in MaxCut problem.

3.2 Discussion of $U(B,\beta)$

Phase change $U(C,\gamma)$ alone does not change the probability of obtaining a certain basis since $|R^{m(z)}a_z|^2 = e^{-im(z)\gamma}e^{i(z)\gamma}|a_z|^2 = |a_z|^2$. So we have to introduce rotation operator $U(B,\beta)$. Here define B as

$$B = \sum_{j=1}^{n}\sigma_j^x = \sigma_1^x \otimes I^{\otimes(n-1)} + I_1 \otimes \sigma_2^x \otimes I^{\otimes(n-2)} + \cdots + I^{\otimes(n-1)} \otimes \sigma_2^x$$

We define $U(B,\beta) = e^{-i\beta B} = e^{-i\beta\sum_{j=1}^{n}\sigma_j^x} = \Pi_{j=1}^{n}e^{-i\beta\sigma_j^x} = \Pi_{j=1}^{n}U(B_j,\beta)$ where $U(B_j,\beta) = e^{-i\beta\sigma_j^x}$

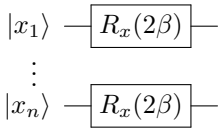
The decomposition of B is exact because

$$[\sigma_i^x,\sigma_j^x] = [\sigma_i^x \otimes I_j, I_i \otimes \sigma_j^x] = (\sigma_i^x \otimes I_j)(I_i \otimes \sigma_j^x) - (I_i \otimes \sigma_j^x)(\sigma_i^x \otimes I_j) = \sigma_i^x \otimes \sigma_j^x - \sigma_i^x \otimes \sigma_j^x = 0$$

A rotation operator $R_x(\theta) = e^{-i\frac{\theta}{2}\sigma^x}$, where $\theta \in [0,2\pi]$, can be used to construct $U(B,\beta)$ with

$$R_x(2\beta) = e^{-i\frac{2\beta}{2}\sigma^x} = U(B,\beta)$$

It is now clear that $2\beta \in [0,2\pi] \to \beta \in [0,\pi]$. This $U(B,\beta)$ can be implemented efficiently with depth 1:



3.3 Workflow

It is not guaranteed that after $|\psi'\rangle = U(B,\beta)U(C,\gamma)|\psi\rangle$ the new state $|\psi'\rangle$ is good enough (or close enough to $|z'\rangle$). A natural remedy is to applying $U(B,\beta)U(C,\gamma)$ multiple times, with different β and γ each time. Suppose we apply p times such operation, we obtained a new state which depends on the angles and defined as:

$$|\psi(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle = |\boldsymbol{\gamma},\boldsymbol{\beta}\rangle = U(B,\beta_p)U(C,\gamma_p)\cdots U(B,\beta_2)U(C,\gamma_2)U(B,\beta_1)U(C,\gamma_1)|\psi\rangle$$

where $\boldsymbol{\gamma} = (\gamma_1,\gamma_2,\cdots,\gamma_p)$ and $\boldsymbol{\beta} = (\beta_1,\beta_2,\cdots,\beta_p)$. It is not easy to determine $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ in advance and there are many possible ways to find such optimal angle with iterations and this corresponds to a $2p$-dimensional optimization. In a typical workflow:

1. Begin with a trial state $|\psi\rangle = |s\rangle$
2. (Classical computer) Initialize $2p$ parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ (determine what angles to use for gates). This could be all 0 if no better choice is implied.
3. (Quantum computer) Construct $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ using $U(B, \beta_p)U(C, \gamma_p) \cdots U(B, \beta_2)U(C, \gamma_2)U(B, \beta_1)U(C, \gamma_1) |s\rangle$ with angles determined in previous step. The circuit for individual operator is shown in previous sections. Now $|\psi\rangle_{\text{new}} = |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ is updated.
4. (Quantum computer) Measure $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ in computational basis set and obtain a value $|z\rangle$.
5. (Classical computer) Using $\sum_{k=1}^{m} C_k(z)$ to calculate $C(z)$.
6. Repeat step 1 to 4 to a number of times. It means we measure the same $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ for many times, and to obtain distribution of states $|z\rangle$. Each $|z\rangle$ corresponds a $C(z)$ which results in a distribution of $C(z)$ with largest value $C(z'')$ and average (expectation) value of C will be $\langle \boldsymbol{\gamma}, \boldsymbol{\beta}|C|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$. We only output the $C(z'')$ and it's eigenvector $|z''\rangle$ as the result of applying $2p$ parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$.
7. Select a new set of $2p$ parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, repeat step 3 to 6. Obtain a distribution of $C(z'')$ and choose the largest one as the final output. The termination condition depends on how you update next set of angles. Consider this as a $2p$-parameter function optimization problem with only function evaluation (step 3 to 6) involves quantum system. If you choose angles from a fine grid, then just run through $2p$ dimensional grid. This could become prohibitively large as the number grows $O(r^{2p})$ where r is roughly the grid resolution (number of points) for each parameter. Other ways of updating new angles including gradient descend or other classical optimization methods. If so, the algorithm will terminate with a preset convergence value.

3.4 Applying MaxCut To QAOA

It is interesting to see how different objective functions C applied in QAOA. Different C means different set of clauses. Take MaxCut problem as example. The problem can be phrased is this way. Suppose we have a connected graph, it can be determined by n vertices and edge set $\{\langle jk\rangle\}$ of size m. We can classify each vertex into one of the two categories, by setting values to $g(i) = +1$ or $g(i) = -1$, where $g(i)$ is a function to query the belonging of vertex i. A cut occurs if two vertices of an edge disagrees, or $g(i)g(j) = -1$. Our goal is to find a classification such that we have the maximum number of cuts, or disagreement of signs alone edges set $\{\langle jk\rangle\}$. Notice, the true maximum could be less than m, consider cutting a triangular as an example.

So our objective function becomes:

$$C = \sum_{\langle jk\rangle} C_{\langle jk\rangle}$$

where

$$C_{\langle jk\rangle} = \frac{1}{2}(-g(i)g(j) + 1) = \begin{cases} 1, & \text{if edge } \langle jk\rangle \text{ is a cut, that is, } g(i)g(j) = -1 \\ 0, & \text{if edge } \langle jk\rangle \text{ is not a cut, that is, } g(i)g(j) = 1 \end{cases}$$

This MaxCut problem can be translated in the quantum perspective in the following way. Consider n vertices as n qubits in computational basis and regard the classification of a vertex as assigning the state of that qubit, $|0\rangle$ or $|1\rangle$. At the end of day, measure the basis state $|z\rangle$ which is a tensor product of $|0\rangle$ and $|1\rangle$, giving the classification of qubits. In this way, all 2^n different vertex classifications correspond to 2^n computational basis. We still want to use the classical objective function, so we need to consider C as operator:

$$C|z\rangle = \sum_{\langle jk\rangle} C_{\langle jk\rangle}(z)|z\rangle = C(z)|z\rangle$$

where

$$C_{\langle jk\rangle}|z\rangle = \frac{1}{2}(-g_j \otimes g_k + I)|z\rangle = \begin{cases} |z\rangle, & \text{if edge } \langle jk\rangle \text{ is a cut, that is, } g_j \otimes g_k|z\rangle = -I \\ 0, & \text{if edge } \langle jk\rangle \text{ is not a cut, that is, } g_j \otimes g_k|z\rangle = I \end{cases}$$

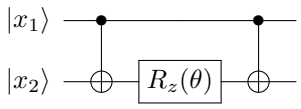
Since $g_i \otimes g_j |z\rangle = (g_i |z_i\rangle) \otimes (g_j |z_j\rangle)$, and we want it to be $-I$ if z_i and z_j are both $|0\rangle$ or $|1\rangle$ and I if z_i and z_j belong different qubits, we can choose function $g = \sigma^z$ to satisfy this property since $\sigma^z |0\rangle = +1$ and $\sigma^z |1\rangle = -1$. So $C_{\langle jk\rangle} = \frac{1}{2}(-\sigma_j^z \otimes \sigma_k^z + I)$.

Similar to previous discussion, we will construct $U(C, \gamma)$ and $U(B, \beta)$ so that

$$U(C,\gamma) = e^{-i\gamma C} = e^{-i\gamma \sum_{\langle jk\rangle} C_{\langle jk\rangle}} = \Pi_{\langle jk\rangle} e^{-i\gamma C_{\langle jk\rangle}} = \Pi_{\langle jk\rangle} U(C_{\langle jk\rangle}, \gamma)$$

$$U(C_{\langle jk\rangle}, \gamma) = e^{-i\gamma \frac{1}{2}(-\sigma_j^z \otimes \sigma_k^z + I)} = e^{-i\frac{-\gamma}{2}(\sigma_i^z \otimes \sigma_j^z)} e^{-i\frac{1}{2}\gamma I}.$$

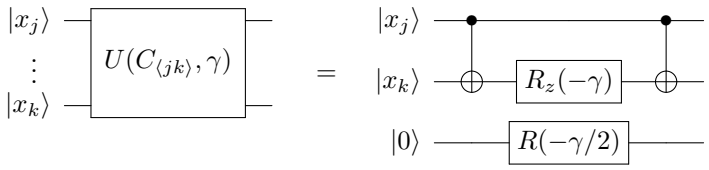
It can be verified that $\exp\left(-i\frac{\theta}{2}(\sigma_j^z \otimes \sigma_k^z)\right)$ can be implemented as:



Supplementary material A.3 shows that the circuit has the same result as applying operator directly.

As a result, we can implement $e^{-i\frac{-\gamma}{2}(\sigma_i^z \otimes \sigma_j^z)}$ using CNOT gates and $R_z(\gamma)$. The second part $e^{-i\frac{\gamma}{2}I}$ is a phase change which can be done using $R(-\gamma/2)$.

So the circuit can be implemented :



The circuit result can be obtained as:

$$\begin{aligned}
&(\alpha_0\beta_0 e^{-i\frac{-\gamma}{2}} |00\rangle + \alpha_0\beta_1 e^{i\frac{-\gamma}{2}} |01\rangle + \alpha_1\beta_0 e^{i\frac{-\gamma}{2}} |10\rangle + \alpha_1\beta_1 e^{-i\frac{-\gamma}{2}} |11\rangle) |0\rangle \\
&\xrightarrow{I^{\otimes n} \otimes R(\gamma)} (\alpha_0\beta_0 e^{-i\frac{-\gamma}{2}} |00\rangle + \alpha_0\beta_1 e^{i\frac{-\gamma}{2}} |01\rangle + \alpha_1\beta_0 e^{i\frac{-\gamma}{2}} |10\rangle + \alpha_1\beta_1 e^{-i\frac{-\gamma}{2}} |11\rangle) e^{-i\gamma I} |0\rangle \\
&= e^{-i\frac{\gamma}{2}} (\alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 e^{i\gamma} |01\rangle + \alpha_1\beta_0 e^{i\gamma} |10\rangle + \alpha_1\beta_1 |11\rangle) |0\rangle
\end{aligned}$$

As can be seen, a phase $e^{i\gamma}$ is added to $|z\rangle$ if two qubits are different (a cut), which is exactly what we want.

The operator $U(B, \beta)$ is the same and described in the previous chapter. The normal work flow will be the same as described in the previous chapter. However, some more simplifications can be performed based on the property of graphs. First, for each edge $\langle jk\rangle$ we only need to evaluate a subgraph that is p steps away from edge $\langle jk\rangle$ instead of whole graph. The idea is that we observe that not all vertices are involved in $U(B, \beta)$ and $U(C, \gamma)$, or mathematically when we write out $\langle C\rangle$ explicitly, only terms contains $\langle C\rangle$ and those no more than p steps aways are kept. It is p steps because $U(C, \gamma)$ contains terms $\sigma_j^x \sigma_k^x$ that mixed one more outer layer of qubits each step. This is a powerful statement because now complexity has transformed from n dependent to p dependent. So, if p is fixed or grows slowly with n, then this algorithm can give dramatic speed up. However, this is not always the case as for a graph, with maximum degree v, the number of qubits in the tree is given as:

$$q_{\text{tree}} = 2\frac{(v-1)^{p+1} - 1}{v - 2}$$

and it can be seen q_{tree} could still be large with large p. Does this simplify the circuit implementation? Indeed, only qubits involved in $\langle jk\rangle$ and distance less than p are required, reducing the depth of circuit, especially for $U(C, \gamma)$.

The second layer of simplification is that only some of the edges in $\langle jk\rangle$ need to be evaluated. It is based on observation that, subgraphs could be isomorphic, which means will generate the same result. As a result only need to evaluate such subgraph once and multiply the occurrence of such graph. The occurrence depend on n but can be easily calculated classically with a given graph. This does not seems as powerful as first simplification but still can give a speed boost when there are many isomorphic subgraphs. For example, in a ring model, there is only one type of subgraph of segment of $2p + 2$ connected vertices. Does this help with the implementation of circuit? Now it means we are braking down the expectation value of original C into fewer smaller terms. For ring graph, we just need to evaluate one C_{12} expectation value rather than all edges.

4 Implementing Traverse Ising model and final project

In the final project we wish to know if we can find intrinsic pattern to the solution angles so that we can make more informed guess on the initialized input angles. If such pattern can be found, we can feed the quantum computer with those guessed initial values and reduce the number iteration and the chance of being trapped in local minimum. The model we used here is traverse filed Ising model (TFIM), which is one of the most wildly used model in quantum simulation. It originates from the spin-spin interactions. For n aligned 1D interacting spin chain, the Hamiltonian for TFIM is:

$$H = \sum_{j=1}^{n} J_j \sigma_j^z \sigma_{j+1}^z + \sum_{j=1}^{n} h_j \sigma_j^x$$

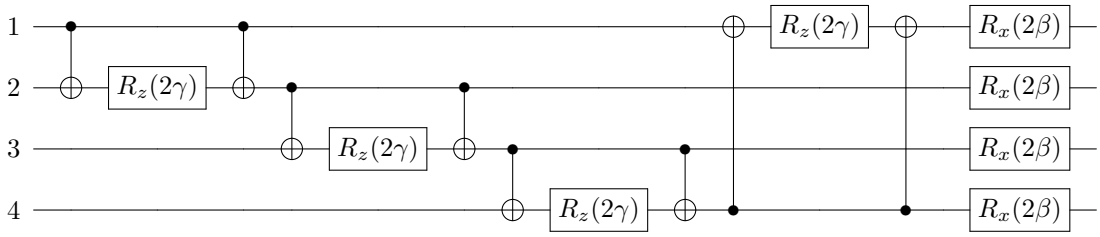
where $\sigma_i^z \sigma_j^z$ denotes the interaction between adjacent spins with strength J_j and h_j denotes the external magnetic field. Here we consider $J_j = 1$ and $h_j = 1$. In addition we have boundary condition $\sigma_{n+1}^z = \sigma_1^z$ so that the last term in first summation is $\sigma_n^z \sigma_1^z$. As described before, we can implement this in QAOA with $C = H$:

$$\begin{aligned} e^{-i\gamma C} &= e^{-i\gamma H} = e^{-i\gamma(\sum_{j=1}^{n} \sigma_j^z \sigma_{j+1}^z + \sum_{j=1}^{n} \sigma_j^x)} \\ &= e^{-i\gamma \sum_{j=1}^{n} \sigma_j^x} e^{-i\gamma \sum_{j=1}^{n} \sigma_j^z \sigma_{j+1}^z} \end{aligned}$$

For B, we still have the same σ^x operators as before. Notice that, we can simplify the expression before implement the circuit:

$$\begin{aligned} U(\beta)U(\gamma) = e^{-i\beta B} e^{-i\gamma C} &= \left(e^{-i\beta \sum_{j=1}^{n} \sigma_j^x}\right)\left(e^{-i\gamma \sum_{j=1}^{n} \sigma_j^x} e^{-i\gamma \sum_{j=1}^{n} \sigma_j^z \sigma_{j+1}^z}\right) \\ &= e^{-i(\beta+\gamma) \sum_{j=1}^{n} \sigma_j^x} e^{-i\gamma \sum_{j=1}^{n} \sigma_j^z \sigma_{j+1}^z} \\ &= e^{-i\beta' \sum_{j=1}^{n} \sigma_j^x} e^{-i\gamma \sum_{j=1}^{n} \sigma_j^z \sigma_{j+1}^z} \end{aligned}$$

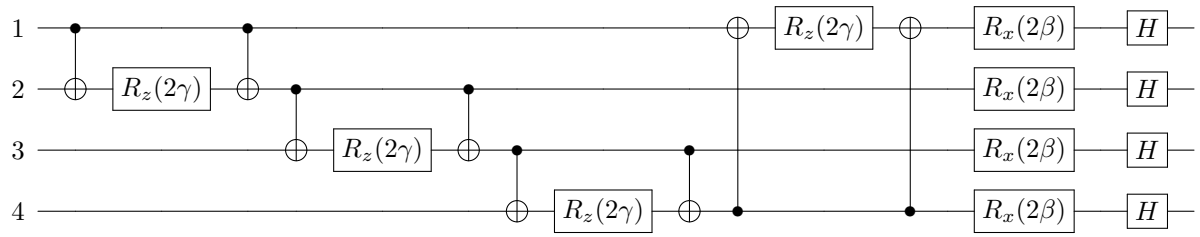
so that we can consider $\beta' = \beta + \gamma$ as our new β. The sample QAOA circuit with $n = 4, p = 1$ will look like:



The result processing contains two parts.

$$\begin{aligned} \langle H \rangle &= \langle \psi | \sum_{j=1}^{n} \sigma_j^z \sigma_{j+1}^z + \sum_{j=1}^{n} \sigma_j^x | \psi \rangle \\ &= \langle \psi | \sum_{j=1}^{n} \sigma_j^z \sigma_{j+1}^z | \psi \rangle + \langle \psi | \sum_{j=1}^{n} \sigma_j^z \sigma_{j+1}^z | \psi \rangle \\ &= \langle H_1 \rangle + \langle H_2 \rangle \end{aligned}$$

where the final state $|\psi\rangle = \sum_i c_i |z\rangle$. The first term on the right can be calculated directly based on the distribution of $|z\rangle$ basis since final state is an eigenstate of Z matrices. The second term requires more work since $|\psi\rangle$ is not an eigenstate of Pauli-X matrices. In order to get the second term, we have to measure the result with the change of basis from Z to X by applying H gate before measurement, as shown below:



An example of post-processing of $\langle\psi|H|\psi\rangle$ for a 2-qubit system is given in appendix A.4.

We used the Regetti Forest code those circuits and use Python to make pre-processing and post-processing. We tested p from 1 to 10 and from 4 to 9 qubits. However we did not go above 9 qubits due to the limitation of our computer capability. The result is not very promising, as the optimized angles are distributed with no strong correlation. It is not surprised that such correlation is difficult to find. The main reason is we have too few data points so that generalizing to higher number of qubits seems unlikely. Figures in appendix A.5. In the future, we could use some quantum cloud services since large companies have much higher simulation capabilities.

5 Conclusion and remarks

With this extensive overview of QAOA above, is any of it useful? Is Quantum Approximate Optimization Algorithm better than using a classical algorithm? Currently the answer is unknown. The MAX-CUT graph problem described above can be viewed as a special case of a constrained satisfaction problem (CSP). A CSP is defined by n variables and a collection of constraints. MAX-CUT can be viewed as a specific case of a MAX-2XOR. After The QAOA solution was more efficient than any classical algorithm at the time. In direct response to the quantum algorithm, a classical algorithm was design that out performs the quantum algorithm for general MAX-kXOR. In response to the classical algorithm, another quantum algorithm was designed that improved upon the previous result. However the improved QAOA does not beat the classical algorithm. This leaves the open question of if there exists a quantum algorithm that can beat a classical algorithm for the Mac-kXOR CSP problem. Although the QAOA applied to a CSP may not beat classical algorithms, it does show quantum supremacy [5]. Quantum Supremacy suggests that QAOA is able to solve a problem that a classical computer cannot. In other words, if one can simulate QAOA classically then P=NP.

Lastly, the QAOA implementation subject to the limitation of gate fidelity. In current noisy quantum computer the number of layers of p will mostly likely result in even worse accuracy since the the gate error negates the benefit of higher p. However, that does not rule out that QAOA can be used in noisy intermediate-scale quantum computer to demonstrate quantum supremacy in the near future.

References

[1] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv Prepr. arXiv1411.4028*, pages 1–16, 2014.

[2] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Phys. Rev. A*, 97(2):1–13, 2018.

[3] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. *arXiv Prepr. arXiv1411.4028*, pages 1–13, 2014.

[4] Cedric Yen-yu Lin, Yechao Zhu, Quantum Information, Computer Science, and For Theoretical Physics. Performance of QAOA on Typical Instances of Constraint Satisfaction Problems with Bounded Degree. *arXiv:1601.01744*, 2016.

[5] Edward Farhi. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. *arXiv Prepr. arXiv1602.07674*, pages 1–22, 2016.

[6] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. Training a quantum optimizer. *Phys. Rev. A*, 94(2):1–10, 2016.

[7] Zhi Cheng Yang, Armin Rahmani, Alireza Shabani, Hartmut Neven, and Claudio Chamon. Optimizing variational quantum algorithms using pontryagin's minimum principle. *Phys. Rev. X*, 7(2):1–9, 2017.

[8] Zhang Jiang, Eleanor G. Rieffel, and Zhihui Wang. Near-optimal quantum circuit for Grover's unstructured search using a transverse field. *Phys. Rev. A*, 95(6):1–9, 2017.

[9] Stuart Hadfield, Zhihui Wang, Eleanor G Rieffel, Bryan O'Gorman, Davide Venturelli, and Rupak Biswas. *Qantum Approximate Optimization with Hard and Soft Constraints*. Number November. 2017.

[10] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *arXiv Prepr. arXiv1709.03489*, pages 1–46, 2017.

[11] Tomas Babej, Mark Fingerhuth, and Christopher Ing. A quantum alternating operator ansatz with hard and soft constraints for lattice protein folding. pages 1–10, 2018.

[12] Wen Wei Ho and Timothy H. Hsieh. Efficient unitary preparation of non-trivial quantum states. *arXiv Prepr. arXiv1803.00026*, pages 1–9, 2018.

[13] Wen Wei Ho, Cheryne Jonay, and Timothy H. Hsieh. Ultrafast State Preparation via the Quantum Approximate Optimization Algorithm with Long Range Interactions. *arXiv Prepr. arXiv1810.04817*, pages 1–9, 2018.

A Supplementary material

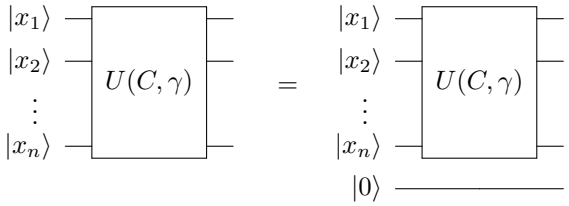
A.1 Meaning of maximizing $\langle A \rangle$

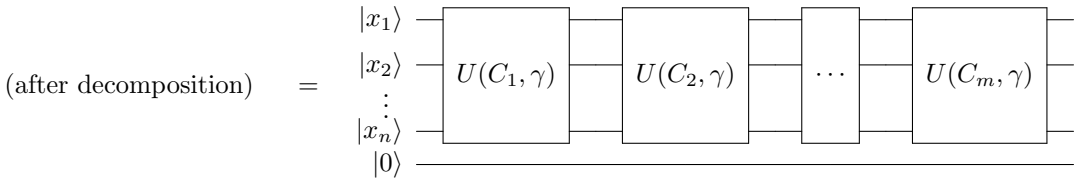
$$\begin{aligned}
\langle C \rangle = \langle \psi | C | \psi \rangle &= \left(\sum_{z \in \{0,1\}^n} a_z^* \langle z | \right) C \left(\sum_{z \in \{0,1\}^n} a_z | z \rangle \right) \\
&= \left(\sum_{z \in \{0,1\}^n} a_z^* \langle z | \right) \left(\sum_{z \in \{0,1\}^n} a_z C | z \rangle \right) \\
&= \left(\sum_{z \in \{0,1\}^n} a_z^* \langle z | \right) \left(\sum_{z \in \{0,1\}^n} a_z f(z) | z \rangle \right) \\
&= \sum_{z \in \{0,1\}^n} f(z) |a_z|^2 \leq \sum_{z \in \{0,1\}^n} f(z') |a_z|^2 \\
&= f(z') \sum_{z \in \{0,1\}^n} |a_z|^2 = f(z')
\end{aligned}$$

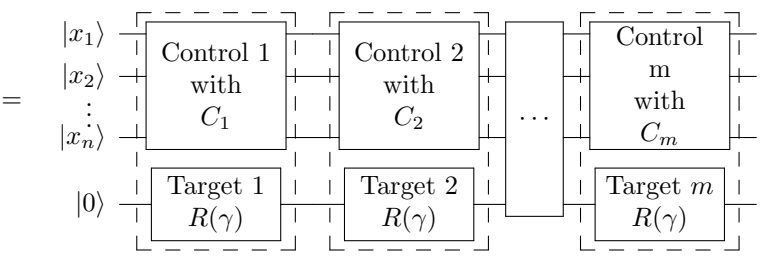
A.2 Implementing $U(C, \gamma)$

To show how we can implement $U(C, \gamma)$ successively, we begin with a general superposition state $|\psi\rangle = \sum_{z \in \{0,1\}^n} a_z |z\rangle$

$$\begin{aligned}
|\psi\rangle = \sum_{z \in \{0,1\}^n} a_z |z\rangle &\xrightarrow{U(C,\gamma)} \sum_{z \in \{0,1\}^n} e^{-i\gamma C} a_z |z\rangle = \sum_{z \in \{0,1\}^n} e^{-i\gamma \sum_{\alpha=1}^m C_\alpha(z)} a_z |z\rangle \\
&\xrightarrow{\text{decompose}} \sum_{z \in \{0,1\}^n} \left(\Pi_{\alpha=1}^m e^{-i\gamma C_\alpha(z)} \right) a_z |z\rangle = \sum_{z \in \{0,1\}^n} \left[\Pi_{\alpha=1}^m U(C_\alpha(z), \gamma) \right] a_z |z\rangle
\end{aligned}$$



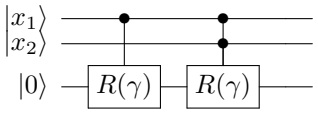




For a $U(C,\gamma)$ gate, we should expect the result :

$$\begin{aligned}
|x_1x_2\rangle &\xrightarrow{U(C,\gamma)} e^{-i\gamma C_2}e^{-i\gamma C_1}|x_1x_2\rangle = e^{-i\gamma C_2}e^{-i\gamma C_1}(\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle) \\
&= e^{-i\gamma C_2(00)}e^{-i\gamma C_1(00)}\alpha_0\beta_0|00\rangle + e^{-i\gamma C_2(01)}e^{-i\gamma C_1(01)}\alpha_0\beta_1|01\rangle \\
&\quad + e^{-i\gamma C_2(10)}e^{-i\gamma C_1(10)}\alpha_1\beta_0|10\rangle + e^{-i\gamma C_2(11)}e^{-i\gamma C_1(11)}\alpha_1\beta_1|11\rangle \\
&= e^{-i\gamma 0}e^{-i\gamma 0}\alpha_0\beta_0|00\rangle + e^{-i\gamma 0}e^{-i\gamma 0}\alpha_0\beta_1|01\rangle \\
&\quad + e^{-i\gamma 0}e^{-i\gamma 1}\alpha_1\beta_0|10\rangle + e^{-i\gamma 1}e^{-i\gamma 1}\alpha_1\beta_1|11\rangle \\
&= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + e^{-i\gamma}\alpha_1\beta_0|10\rangle + e^{-2i\gamma}\alpha_1\beta_1|11\rangle \\
&= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + R(\gamma)\alpha_1\beta_0|10\rangle + R^2(\gamma)\alpha_1\beta_1|11\rangle
\end{aligned}$$

Or eqivalently can construct circuit with the help of ancilla qubit is equivalent to



If we use the second explicit circuit, we have

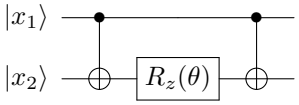
$$\begin{aligned}
|x_1x_2\rangle|1\rangle &= (\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle)|1\rangle \\
&\xrightarrow{\text{control}-R(\gamma)} \alpha_0\beta_0|00\rangle|1\rangle + \alpha_0\beta_1|01\rangle|1\rangle + \alpha_1\beta_0|10\rangle \otimes R(\gamma)|1\rangle + \alpha_1\beta_1|11\rangle \otimes R(\gamma)|1\rangle \\
&= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + R(\gamma)\alpha_1\beta_0|10\rangle + R(\gamma)\alpha_1\beta_1|11\rangle \\
&\xrightarrow{\text{CC}-R(\gamma)} \alpha_0\beta_0|00\rangle|1\rangle + \alpha_0\beta_1|01\rangle|1\rangle + R(\gamma)\alpha_1\beta_0|10\rangle \otimes |1\rangle + R(\gamma)\alpha_1\beta_1|11\rangle \otimes R(\gamma)|1\rangle \\
&= \alpha_0\beta_0|00\rangle|1\rangle + \alpha_0\beta_1|01\rangle|1\rangle + R(\gamma)\alpha_1\beta_0|10\rangle|1\rangle + R^2(\gamma)\alpha_1\beta_1|11\rangle \otimes |1\rangle \\
&= (\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + R(\gamma)\alpha_1\beta_0|10\rangle + R^2(\gamma)\alpha_1\beta_1|11\rangle)|1\rangle
\end{aligned}$$

A.3 Implementing MAXCUT

Consider applying operator $U(C,\gamma)$ directly:

$$\begin{aligned}
|x_1x_2\rangle &= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle \\
&\xrightarrow{e^{-i\frac{\theta}{2}(\sigma_1^z\otimes\sigma_2^z)}} \alpha_0\beta_0e^{-i\frac{\theta}{2}(\sigma_1^z\otimes\sigma_2^z)}|00\rangle + \alpha_0\beta_1e^{-i\frac{\theta}{2}(\sigma_1^z\otimes\sigma_2^z)}|01\rangle + \alpha_1\beta_0e^{-i\frac{\theta}{2}(\sigma_1^z\otimes\sigma_2^z)}|10\rangle + \alpha_1\beta_1e^{-i\frac{\theta}{2}(\sigma_1^z\otimes\sigma_2^z)}|11\rangle \\
&= \alpha_0\beta_0e^{-i\frac{\theta}{2}}|00\rangle + \alpha_0\beta_1e^{i\frac{\theta}{2}}|01\rangle + \alpha_1\beta_0e^{i\frac{\theta}{2}}|10\rangle + \alpha_1\beta_1e^{-i\frac{\theta}{2}}|11\rangle
\end{aligned}$$

It can be verified that $\exp\left(-i\frac{\theta}{2}(\sigma_j^z\otimes\sigma_k^z)\right)$ can be implemented as:



Consider going through the circuit:

$$\begin{aligned}
|x_1x_2\rangle &= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle \\
&\xrightarrow{CNOT} \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|11\rangle + \alpha_1\beta_1|10\rangle \\
&\xrightarrow{I\otimes R_z(\theta)} \alpha_0\beta_0e^{-i\frac{\theta}{2}\sigma_2^z}|00\rangle + \alpha_0\beta_1e^{-i\frac{\theta}{2}\sigma_2^z}|01\rangle + \alpha_1\beta_0e^{-i\frac{\theta}{2}\sigma_2^z}|11\rangle + \alpha_1\beta_1e^{-i\frac{\theta}{2}\sigma_2^z}|10\rangle \\
&= \alpha_0\beta_0e^{-i\frac{\theta}{2}}|00\rangle + \alpha_0\beta_1e^{i\frac{\theta}{2}}|01\rangle + \alpha_1\beta_0e^{i\frac{\theta}{2}}|11\rangle + \alpha_1\beta_1e^{-i\frac{\theta}{2}}|10\rangle \\
&\xrightarrow{CNOT} \alpha_0\beta_0e^{-i\frac{\theta}{2}}|00\rangle + \alpha_0\beta_1e^{i\frac{\theta}{2}}|01\rangle + \alpha_1\beta_0e^{i\frac{\theta}{2}}|10\rangle + \alpha_1\beta_1e^{-i\frac{\theta}{2}}|11\rangle
\end{aligned}$$

A.4 An example of $\langle\psi|H|\psi\rangle$ for a 2-qubit system

Suppose a general final state $|\psi\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$, where coefficients $\{a_{00}, a_{01}, a_{10}, a_{01}\}$ are given as the probability distribution from the circuit result. Also we have $< H >=< H_1 > + < H_2 >$.

$$\begin{aligned}
\langle\psi|H_1|\psi\rangle &= \langle\psi|\sum_{j=0}^{1}\sigma_0^z\sigma_1^z|\psi\rangle \\
&= (a_{00}\langle 00| + a_{01}\langle 01| + a_{10}\langle 10| + a_{11}\langle 11|)(\sigma_0^z\sigma_1^z + \sigma_1^z\sigma_0^z)(a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle) \\
&= (a_{00}\langle 00| + a_{01}\langle 01| + a_{10}\langle 10| + a_{11}\langle 11|)\sigma_0^z\sigma_1^z(a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle) \\
&\quad + (a_{00}\langle 00| + a_{01}\langle 01| + a_{10}\langle 10| + a_{11}\langle 11|)\sigma_1^z\sigma_0^z(a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle) \\
&= (a_{00}\langle 00| + a_{01}\langle 01| + a_{10}\langle 10| + a_{11}\langle 11|)(a_{00}|00\rangle - a_{01}|01\rangle - a_{10}|10\rangle + a_{11}|11\rangle) \\
&\quad + (a_{00}\langle 00| + a_{01}\langle 01| + a_{10}\langle 10| + a_{11}\langle 11|)(a_{00}|00\rangle - a_{01}|01\rangle - a_{10}|10\rangle + a_{11}|11\rangle) \\
&= 2(a_{00}^2 - a_{01}^2 - a_{10}^2 + a_{01}^2)
\end{aligned}$$

For second term, we have to rerun the circuit again with change of basis, so we will apply H gate to the end of circuit. The new state we have is

$$\begin{aligned}
|\psi\rangle &\xrightarrow{H} (a_{00} + a_{01} + a_{10} + a_{11})|00\rangle + (a_{00} - a_{01} + a_{10} - a_{11})|01\rangle + (a_{00} + a_{01} - a_{10} - a_{11})|10\rangle + (a_{00} - a_{01} - a_{10} + a_{11})|11\rangle \\
&= b_{00}|00\rangle + b_{01}|01\rangle + b_{10}|10\rangle + b_{11}|11\rangle
\end{aligned}$$

, and also $\sigma^x \xrightarrow{H} \sigma^z$, so we have:

$$\begin{aligned}
\langle H_2\rangle &= (b_{00}\langle 00| + b_{01}\langle 01| + b_{10}\langle 10| + b_{11}\langle 11|)(\sigma_0^z + \sigma_1^z)(b_{00}|00\rangle + b_{01}|01\rangle + b_{10}|10\rangle + b_{11}|11\rangle) \\
&= (b_{00}\langle 00| + b_{01}\langle 01| + b_{10}\langle 10| + b_{11}\langle 11|)(\sigma_0^z)(b_{00}|00\rangle + b_{01}|01\rangle + b_{10}|10\rangle + b_{11}|11\rangle) \\
&\quad + (b_{00}\langle 00| + b_{01}\langle 01| + b_{10}\langle 10| + b_{11}\langle 11|)(\sigma_1^z)(b_{00}|00\rangle + b_{01}|01\rangle + b_{10}|10\rangle + b_{11}|11\rangle) \\
&= b_{00}^2 + b_{01}^2 - b_{10}^2 - b_{11}^2 + b_{00}^2 - b_{01}^2 + b_{10}^2 - b_{11}^2 \\
&= 2b_{00}^2 - 2b_{11}^2 \\
&= 2(a_{00}a_{01} + a_{00}a_{10} + a_{11}a_{01} + a_{11}a_{10})
\end{aligned}$$

This is the same result if we calculate directly without change of basis:

$$\begin{aligned}
\langle H_2\rangle &= (a_{00}\langle 00| + a_{01}\langle 01| + a_{10}\langle 10| + a_{11}\langle 11|)(\sigma_0^x + \sigma_1^x)(a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle) \\
&= 2(a_{00}a_{01} + a_{00}a_{10} + a_{11}a_{01} + a_{11}a_{10})
\end{aligned}$$

It seems to imply that if we know the probability of state $|\psi\rangle_z$ in z basis then we can simply calculate the probability in $|\psi\rangle_x$. This is not true since knowing the $|a_{00}|^2$ is not the same as knowing a_{00} where the sign (phase) is implicitly still unknown. As a result, we still have to run the circuit with the change of basis when measuring the Hamiltonian in corresponding basis.

A.5 Traverse Ising model Figures

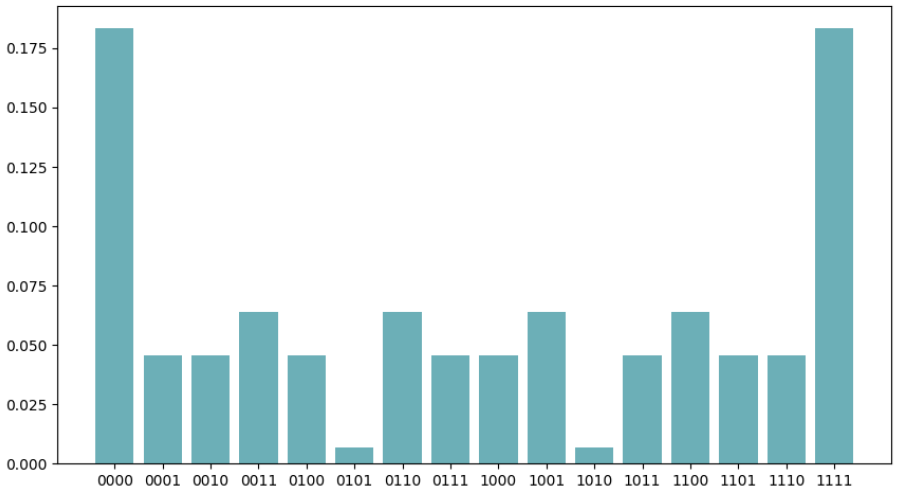


Figure 1: Traverse Ising with p = 1

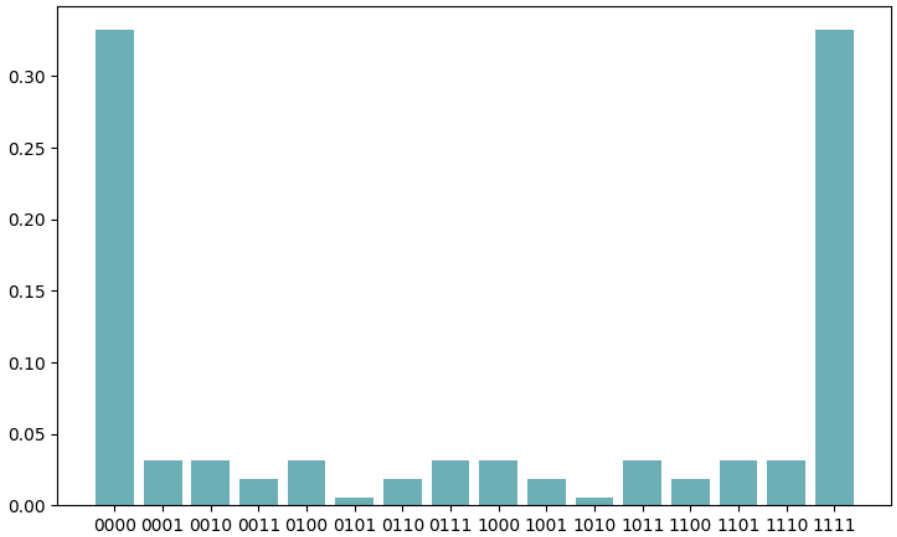


Figure 2: Traverse Ising with p = 2

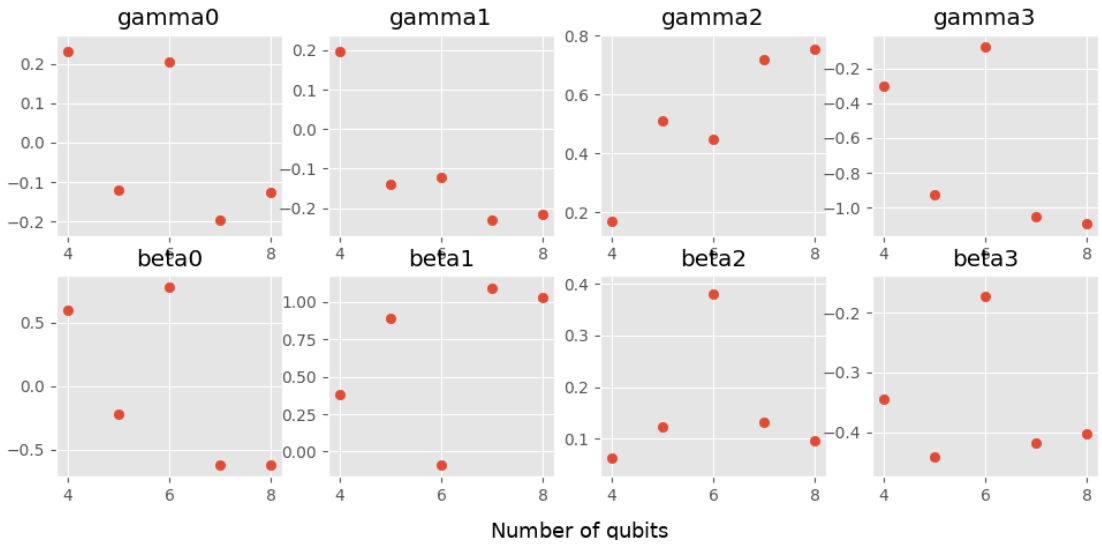


Figure 3: Final gamma and beta for various number of qubits with p = 4