Internship report
L3 Parcours Spécial Physique

Quantum algorithms for machine learning applications

Gabriel Amorosetti

supervised by

Professor Elisa Ercolessi

and

PhD student Simone Tibaldi

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Dipartimento di Fisica e Astronomia
"Augusto Righi"

Acknowledgements

First, I would like to thanks Professor Elisa Ercolessi for accepting to supervise this internship, giving me the opportunity to work at the Physics department of the University of Bologna, on such an interesting subject that is quantum computation. I would also like to thanks PhD student Simone Tibaldi for his time, availability, help and kindness through this project, making me feel at ease and allowing me to learn so much about a field of study that was totally unknown for me a few weeks ago.

Many thanks to all the Erasmus, Bachelor, Master and PhD students met in the amazing city of Bologna that made this experience even more enjoyable.

I would also like to thanks all the professors from the *Parcours Spécial* Bachelor for giving us the chance to discover and experience the world of scientific research during the degree and particularly during this project in a foreign university.

Special thanks to the friends of PS that made this *Parcours Spécial* so special.

Abstract

Theoretical and experimental advances based on quantum mechanics have led to the emergence of a new scientific discipline based on these new laws of physics : quantum computing. The latter takes advantage of the principle of quantum superposition, which allows a unit of quantum information to have several values at the same time, in contrast to classical computing where the unit of information can only have one value at a time. This property makes it possible to develop quantum algorithms that solve certain types of problems much more efficiently than classical algorithms. This is why studying and researching new quantum algorithms is important : with a quantum computer, it is theoretically possible to perform tasks that a classical computer could not do in a reasonable time. The technological goal today is to achieve fairly stable quantum computers with a large number of quantum information units in order to demonstrate experimentally the advantage of this new technology. The goal of this internship is to study a particular case of quantum algorithm, based on machine learning techniques, which allows to solve combinatorial optimization problems : the quantum approximate optimization algorithm. This study, including the realization of a Python code, will lead to the implementation of this algorithm on a simulator as well as on a real IBM quantum processor.

Keywords : quantum algorithms, machine learning, Max-Cut problem, Quantum Approximate Optimization Algorithm, combinatorial optimization problems

Résumé

Les avancées théoriques et expérimentales reposant sur la mécanique quantique ont permis l'émergence d'une nouvelle discipline scientifique s'appuyant sur ces nouvelles lois de la physique : l'informatique quantique. Cette dernière tire notamment profit du principe de superposition quantique, permettant à une unité d'information quantique de posséder plusieurs valeurs en même temps, s'opposant à l'informatique classique où l'unité d'information ne peut posséder qu'une valeur à la fois. Cette propriété permet alors de développer des algorithmes quantiques résolvant certains types de problèmes de manière bien plus efficace que des algorithmes classiques. C'est donc pour cette raison qu'étudier et rechercher des nouveaux algorithmes quantiques est important : avec un ordinateur quantique, il est théoriquement possible d'effectuer des tâches qu'un ordinateur classique ne pourrait pas effectuer en un temps raisonnable. L'objectif technologique est aujourd'hui d'arriver à réaliser des ordinateurs quantiques assez stables et comportant un grand nombre d'unités d'information quantique afin de démontrer expérimentalement l'avantage de cette nouvelle technologie. L'objectif de ce stage est d'étudier un cas particulier d'algorithme quantique, reposant sur des techniques d'apprentissage automatique, qui permet de résoudre des problèmes d'optimisation combinatoire : l'algorithme quantique d'optimisation approximative. Cette étude, incluant la réalisation d'un code Python, ammènera à l'implémentation de cet algorithme sur un simulateur ainsi que sur un véritable processeur quantique d'IBM.

Mots-clés : algorithmes quantiques, apprentissage automatique, problème de coupe maximum, algorithme quantique d'optimisation approximative, problèmes d'optimisation combinatoire

Contents

1 Introduction

Quantum computation takes advantage of the laws of quantum mechanics to potentially perform better than classical computation on specific tasks. While waiting for an experimental demonstration of the superiority of quantum computers over classical computers, we might study quantum algorithms to find ones that are appropriate to solve such precise problems that classical algorithms can't solve. Combinatorial optimization problems are ones of these and are worth studying because they can be connected to physics theories. For example, the Ising model is a mathematical description of ferromagnetism that can be connected to a specific combinatorial optimization problem : the Max-Cut problem. Quantum Approximate Optimization Algorithm [2] is a type of hybrid quantum-classical algorithm created to attempt to find a solution to this particular problem. This algorithm is an application of machine learning techniques : it improves its results from itself, learning from the use of data. This report is dedicated in a first part to the study of the fundamental concepts of quantum computation, and then in a second part to the study of this quantum algorithm and its implementation on a simulator and on a real quantum computer thanks to the Qiskit [5] Python package allowing us to code quantum algorithms. To contain this report to a reasonable size, we may not present all the work done during this project, and will voluntarily skip some proofs or the interesting study of the Deutsch-Jozsa algorithm [1], the first example of quantum algorithm that performs better than any possible classical algorithm.

2 Basics of quantum computation

The binary digit, known as **bit**, is the most basic unit of information in classical computation, that can represent one of two possibles values : either **0** or **1**. Quantum computation relies on the same principle and have as basic unit of quantum information the quantum binary digit, or **qubit**.

2.1 The basic unit of quantum information : the qubit

2.1.1 Mathematical description

A qubit is two-state quantum system that **when measured**, as the bit, can only take as outcome the value either **0** or **1**. However, this quantum version of the bit is described by the laws of quantum mechanics, and can therefore, **before a measurement**, be in the state $|0\rangle$ or $|1\rangle$, but also in a **superposition** of these two states. Using the Dirac notation of quantum mechanics to describe a state, the state of a qubit can be described as a linear combination of these two possible states :

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

with $(\alpha, \beta) \in \mathbb{C}^2$ being the probability amplitudes associated to the measurement outcomes $|0\rangle$ and $|1\rangle$, verifying $|\alpha|^2+|\beta|^2 = 1$ in order to satisfy the normalisation condition, $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ forming an orthonormal basis for this vector space, also called the computational basis states. Hence, when measuring the state of a qubit, the only possible outcome are either $|0\rangle$ or $|1\rangle$, with the probability $|\alpha|^2$ or $|\beta|^2$.

If we consider a system of n qubits, the statevector of this system is the tensor product (explained in the appendices section) between the statevector of each qubit, such as $|q_1 q_2 ... q_n\rangle = |q_1\rangle \otimes |q_2\rangle \otimes ... \otimes |q_n\rangle$, with all the possible combinations of $|q_1 q_2 ... q_n\rangle$ forming the computational basis. This system can therefore be in a superposition of all these 2^n states. We can guess the interest of using a large number of qubits : when doing an operation on all these qubits, the operation is actually done on these 2^n states at the same time. This allows to perform quantum parallelism and to evaluate a function for different values simultaneously.

2.1.2 Bloch sphere representation

As α and β are complex numbers that are composed of real and imaginary parts, $|\psi\rangle$ is described by 4 unknown variables. However the condition $|\alpha|^2 + |\beta|^2 = 1$ allows to have an expression $|\psi\rangle$ dependant on only 3 variables. Knowing that, it can be shown that $|\psi\rangle$ can be rewritten as $|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right)$. $e^{i\gamma}$ is known as the *global phase factor*, having no observable effects and so can be ignored. We now have a statevector depending on only two variables :

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$$

This equation describes the surface of sphere, that is known as the Bloch sphere. This allows us to geometrically visualize the state of a single qubit.

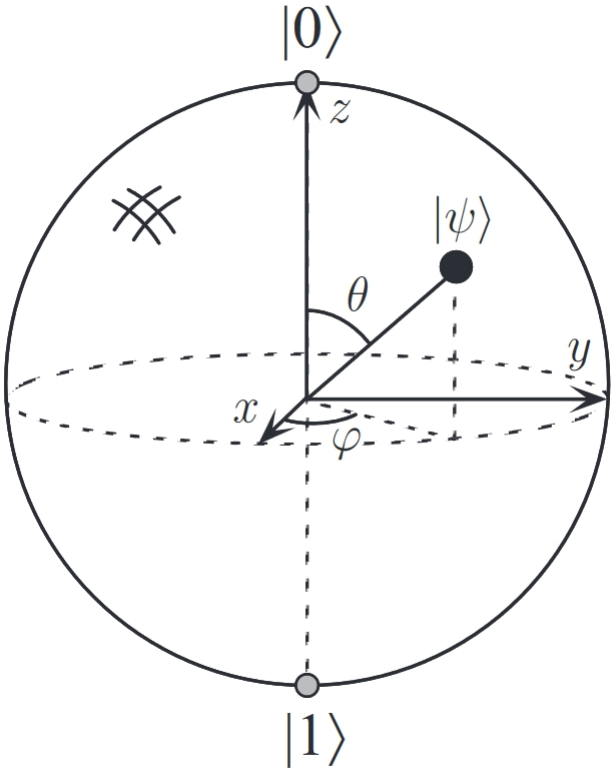

Figure 1: Bloch sphere representation of a qubit [4]

2.2 Qubit operations

The state of a classical bit can be changed through the use of logic gates, allowing us to manipulate the information in order to create algorithms. Qubits work the same way and the information they contain can be manipulated through the use of quantum logic gates. These gates are described by unitary matrices, and applying a gate G on a quantum state $|\psi\rangle$ is mathematically translated to $|\psi'\rangle = G|\psi\rangle$, where $|\psi'\rangle$ is the new quantum state after the application of the gate. All of the quantum logic gates are their own inverse, and so applying them two times will have no effects on the initial statevector. This section will only cover some of the most important quantum logic gates that will be useful for us.

The most important single qubit gates are described by the Pauli matrices :

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

The X gate is the quantum version of the NOT gate, that when applied to a qubit in the state $|0\rangle$ give the state $|1\rangle$ and vice versa. The action of the X gate can be visualized on the figure 1 as a rotation of the statevector of π around the x axis. Y and Z gate actions can be as well visualized as rotations of the statevector of π around the y and z axis.

One gate that will also be convenient for quantum computation is the *Hadamard* gate : applying it to $|0\rangle$ or $|1\rangle$ will give us the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ or $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. This gate will be useful as it produces a state in a perfect superposition of the computational basis states, allowing us to take the most of the quantum superposition principle.

Hadamard gate matrix represensation is defined as :

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

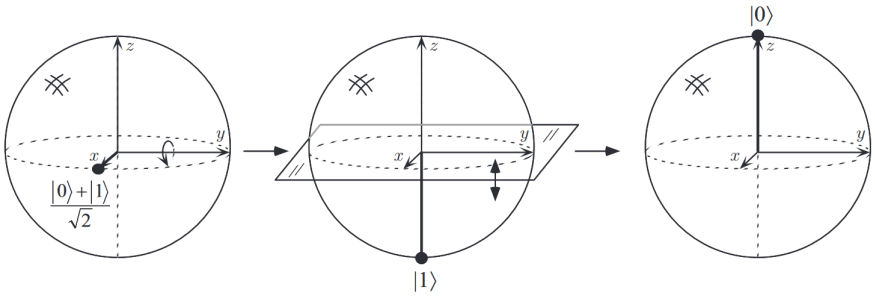

Figure 2: Visualization of the Hadamard gate on the Bloch sphere, acting on the input state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ [4]

The figure 2 helps visualize the application of the Hadamard gate as a rotation of the initial statevector of $\frac{\pi}{2}$ around the y axis, followed by a rotation of π over the x axis.

The table 3 in the appendices section sums up the actions of these single qubit gates.

Multiple qubits gates also exist and are widely used in quantum computation like the $CNOT$ or *Toffoli* gate [6] for example. However, even if studied during this project, we will not introduce them as they are not useful for the specific problem we choose to work on.

2.3 Quantum circuits

In the same way classical bits are manipulated through a classical circuit, quantum algorithms rely quantum circuits to operate qubits. Quantum circuits consist in a sequence of initializations of qubits states, gates applied to these qubits and measurements.

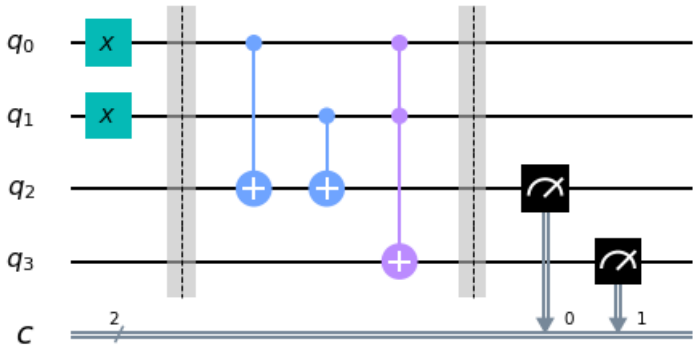

Figure 3: Example of a quantum circuit [5]

Figure 3 is an example of a very simple quantum circuit. Even when not precised, qubits are usually initialized to the state $|0\rangle$. With time going from left to right, each qubit is represented by a line, on which gates applied to the concerned qubit are represented. Most of the time, a measurement is done on each qubit at the end of the circuit, with outcome 0 or 1, that have to be stored in a classical bit represented by the double bottom line.

3 Quantum algorithms applied to combinatorial optimization problems

3.1 The Max-Cut problem

Combinatorial optimization problems consist on finding the optimal solution from a finite set of solutions. The Max-Cut problem is one those and relies on graphs. These graphs are made of nodes that can be assigned one of two possibles values, connected to each others by an edge. To explain the Max-Cut problem, we will use a simple example as the one on figure 4, displaying 4 nodes with 4 edges graphs.

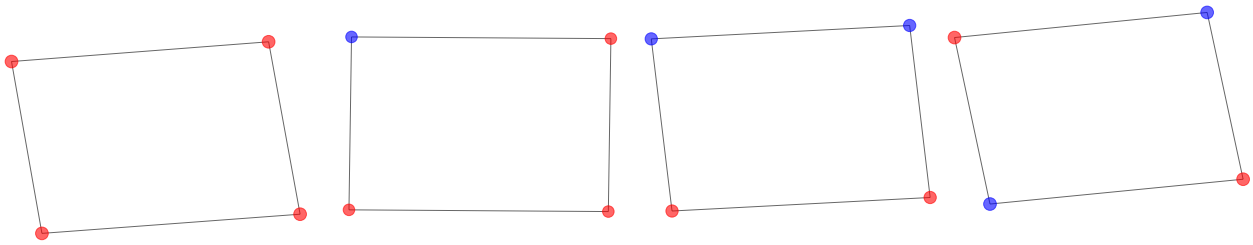

Figure 4: The Max-Cut problem [5]

Here, the nodes can be either red or blue. The goal of the Maxcut problem is to assign each node either the color red or blue, such that the number of edges that are connected by two different colored nodes is maximum. As there are 4 edges and two possible colors for the nodes, the goal is to find the best assignment from the $2^4 = 16$ that are possible. From the left to the right, we can count for each graph the number of edges that satisfy this condition : 0, 2, 2 and 4. As it is a simple graph, we can easily deduce that the rightmost graph of the figure 4, and its symmetric (reversing the blue and red nodes) are the best solutions. We can now also replace red and blue by the values 0 or 1, giving us as optimal solutions the bits strings "1010" and "0101".

A first approach to determine how much a solution is optimal is to use a cost function. Keeping our 4 nodes graph as example, we can imagine a cost function that will assign an edge either the weight $+1$ if it is connected by two nodes having the same value, or the weight -1 if it is connected by two nodes having different values, the cost function being the sum of all the weights of a graph. As the goal of the Max-Cut problem is to determine a solution having the maximum number of edges connected by two nodes having different values, we want to assign the more -1 weights as possible, meaning we want the lowest possible value for the cost function. Hence this maximization problem can become a minimization problem.

For this example, we can easily determine all the possible values for the cost function, considering all the possible combinations for a 4 nodes circular graph. These results are listed in the table 1 of the appendices section. As we found before, the graphs associated to the bits strings "1010" and "0101" have the lowest cost values and are indeed the optimal solutions of this problem.

Taking advantage of quantum computation will be useful for larger graphs : as the number of nodes n grows, the number of possible solutions 2^n grows exponentially and determining the cost value to every possible solution is not doable in a reasonable time. Studying quantum algorithms that could solve this type of problem faster than classical algorithms is one of the promising applications of quantum computers.

3.2 Quantum Approximate Optimization Algorithm

Quantum Approximate Optimization Algorithm (QAOA), introduced by *Farhi et al* [2] is a variational (meaning it depends on free parameters) quantum algorithm that could solve Max-Cut problems faster than a classical algorithm. QAOA quantum circuit uses two parameterized unitary gates $U(\beta)$ and $U(\gamma)$ that prepare a quantum state $|\psi(\beta,\gamma)\rangle$, with the number of qubits and the parametrization of the gates depending on the problem considered. The goal of QAOA is to find the best parameters that give a quantum state that encodes the optimal solution of the problem.

Each of these unitary gates depends on a Hamiltonian : the problem Hamiltonian H_P and the mixing Hamiltonian H_M, hence giving us the expressions $U(\beta) = e^{-i\beta H_P}$ and $U(\gamma) = e^{-i\gamma H_M}$. In the quantum circuit, this block of two gates can be applied p times on the starting quantum state :

$$|\psi(\boldsymbol{\beta},\gamma)\rangle = \underbrace{U(\boldsymbol{\beta})U(\gamma)\cdots U(\boldsymbol{\beta})U(\gamma)}_{p \text{ times}} |\psi_0\rangle$$

We call p the depth of the quantum circuit.

To better understand how the QAOA is built, we will focus on the example of our 4 nodes circular graph used before. Following the previous description of the QAOA, we need to find the mixing Hamiltonian H_M and the problem Hamiltonian H_P associated to the Max-Cut problem.

First, we need to find a problem Hamiltonian H_P that describes in a quantum way the cost function we introduced previously. To do this, instead of using the set $\sigma_i = \{0, 1\}$ for the possible nodes values, we can use the set $z_i = \{+1, -1\}$, where $z_i = -2\sigma_i + 1$. The bit 0 is now assigned the value +1, and the bit 1 the value -1. We shall now use a quantum description of our graph suitable for quantum computation : instead of bits, we will assign a qubit state either $|0\rangle$ or $|1\rangle$ to a node. Hence, with this quantum description considered, a $|0\rangle$ node is now assigned the value +1, and a $|1\rangle$ node the value -1. This is perfectly described in a quantum way by the action of the Z Pauli gate on a qubit, which eigenvalues in the computational basis are +1 and -1, associated to the eigenvectors $|0\rangle$ and $|1\rangle$. Therefore we can formulate our problem Hamiltonian as

$$H_P = \sum_{i,j} Z_i Z_j,$$

a sum over the edges, where Z_i and Z_j are Z Pauli gates acting respectively on the qubits i and j, these two qubits being assigned to two nodes connected by and edge. It can be shown that $Z_0 Z_1(|0\rangle \otimes |0\rangle) = +1(|0\rangle \otimes |0\rangle)$ or $Z_0 Z_1(|0\rangle \otimes |1\rangle) = -1(|0\rangle \otimes |1\rangle)$ (and it is also true if we reverse the 0s and the 1s). This short expression of H_P should in reality takes the following form if we consider for example our 4 nodes graph problem, using hence 4 qubits :

$$\begin{aligned} H_P = (Z \otimes \mathbb{1}^{\otimes 3})(\mathbb{1} \otimes Z \otimes \mathbb{1}^{\otimes 2}) + (\mathbb{1} \otimes Z \otimes \mathbb{1}^{\otimes 2})(\mathbb{1}^{\otimes 2} \otimes Z \otimes \mathbb{1}) \\ + (\mathbb{1}^{\otimes 2} \otimes Z \otimes \mathbb{1})(\mathbb{1}^{\otimes 3} \otimes Z) + (\mathbb{1}^{\otimes 3} \otimes Z)(Z \otimes \mathbb{1}^{\otimes 3}) \end{aligned}$$

H_P is a (16×16) matrix (see the full matrix in the appendices section) that can be multiplied by any 4 qubits statevector $|q_0 q_1 q_2 q_3\rangle$ of our quantum circuit. We show in the appendices section that indeed, this problem Hamiltonian is the correct quantum version of our classical cost function, giving us the eigenvalue -4 for the optimal solutions among the possibles ones associated to our 4 nodes graph problem. We built H_P such that its ground states encode the optimal solutions.

Then, we will use as our mixing Hamiltonian

$$H_M = \sum_{i=0}^{n-1} X_i,$$

n being the number of qubits, where the X_i Pauli gate only acts on the qubit i. We use this particular mixing Hamiltonian as it is more suitable to the problem at hand [3]. The point of adding this mixing Hamiltonian to our circuit is that without it, we would not be able to get rid of some of the parameterized diagonal terms of the problem Hamiltonian during the optimization, hence having as our circuit output the same state as the input one, up to a certain phase. This mixing Hamiltonian enables the minimization process. For our considered problem, it is expressed by

$$H_M = X \otimes \mathbb{1}^{\otimes 3} + \mathbb{1} \otimes X \otimes \mathbb{1}^{\otimes 2} + \mathbb{1}^{\otimes 2} \otimes X \otimes \mathbb{1} + \mathbb{1}^{\otimes 3} \otimes X$$

The parameterized unitary gates $U(\beta)$ and $U(\gamma)$, depending on the two hamiltonians $H_P = \sum_{i,j} Z_i Z_j$ acting on two qubits and $H_M = \sum_{i=0}^{n-1} X_i$ acting on one qubit, are implemented in the Qiskit Python Package [5] by the RZZ and RX gates. They are definined as following :

$$R_{ZZ}(\beta) = \exp\left(-i\frac{\beta}{2} Z \otimes Z\right) = \begin{pmatrix} e^{-i\frac{\beta}{2}} & 0 & 0 & 0 \\ 0 & e^{i\frac{\beta}{2}} & 0 & 0 \\ 0 & 0 & e^{i\frac{\beta}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\beta}{2}} \end{pmatrix}$$

$$RX(\gamma) = \exp\left(-i\frac{\gamma}{2} X\right) = \begin{pmatrix} \cos\frac{\gamma}{2} & -i\sin\frac{\gamma}{2} \\ -i\sin\frac{\gamma}{2} & \cos\frac{\gamma}{2} \end{pmatrix}$$

We can visualize on the Bloch sphere the action of the RZZ gate on two qubits as the rotation of each qubit statevector of $\frac{\beta}{2}$ around the z axis, and the action of the RX gate on one qubit as the rotation of the qubit statevector of $\frac{\gamma}{2}$ around the x axis. These definitions follow the previous description of the two unitary gates, up to a factor $\frac{1}{2}$. This convention makes these two gates π-periodic.

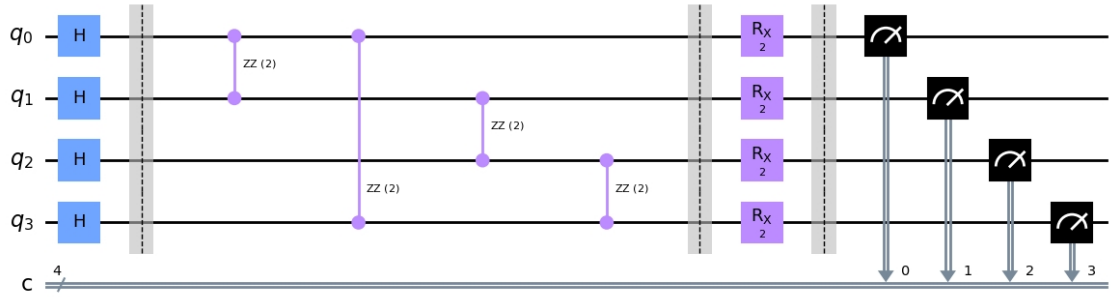

Figure 5: QAOA quantum circuit for $p = 1$, with $\beta = 2$ and $\gamma = 2$

The figure 5 represents the quantum circuit of the QAOA, considering a Max-Cut problem with a 4 nodes circular graph. To take advantage of the quantum superposition of a state, each qubit is initialized to an equal superposition of all the basis states thanks to Hadamard gates. This gives us the state $|\psi_0\rangle = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes n}$, with $n = 4$ on figure 5, which is the tensor product of all the qubits being in a perfect equal superposition of the state $|0\rangle$ and $|1\rangle$. Then, the RZZ gate is applied to each pair of qubits representing two nodes connected by and edge. Finally, the RX gate is applied to each qubit, and measurements are done on each qubit, assigning each corresponding classical bit the value 0 or 1.

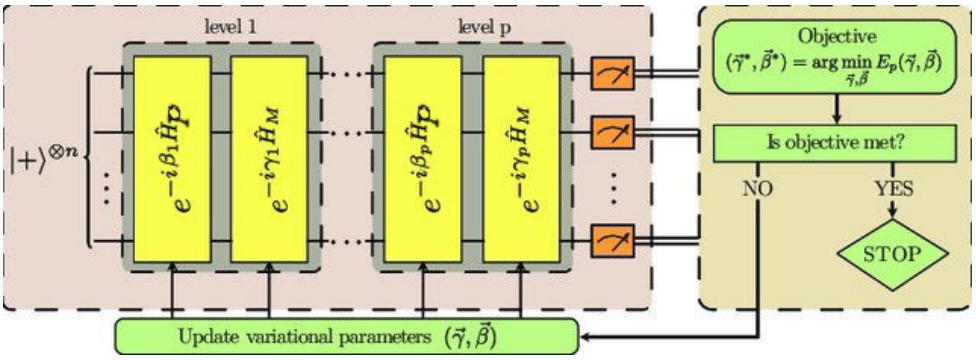

Figure 6: Schematic representation of the QAOA [7]

As described in figure 6, QAOA works following this process :

1. We choose $(\vec{\beta_0}, \vec{\gamma_0})$ initial values. We show in the appendices section the importance of this initial guess.

2. We run as a simulation or an experiment on a quantum processor the QAOA quantum circuit a big number of times with these initial values. Each time, measurement on each qubit gives us either 0 or 1, giving us as a result a bits string made of 0 and 1. Running this quantum circuit several times hence gives us probabilities on all the possible measurement outcomes.

3. As these bits strings can be associated to a graph, the classical part of QAOA computes the cost of each possible outcome, in order to determine with the associated probabilities the average cost of the whole simulation or experiment.

4. The classical optimizer associated to the QAOA quantum circuit gives us new β and γ values that should give a lower average cost. We will not study the process of the optimizer as it is not the goal of this project, but we can simply say that it uses some mathematical optimization methods that compare the previous $(\vec{\beta}, \vec{\gamma})$ found and the average cost values they gave through the running of the quantum circuit, in order determine which new values should give a lower average cost. In other words, if the optimizer parameters are going in a certain direction and the average cost value is going down, the optimizer will keep trying to move these parameters in the same direction, hoping to find a minimum.

5. These new values are used to parameterize a new QAOA quantum circuit, that will be run as in step 2.

6. Then, this whole process keep going until the average cost value computed in step 3 satisfies a stop criteria, which ensures the convergence of the value, allowing a certain error margin.

Thanks to QAOA, we now have found $(\vec{\beta}*, \vec{\gamma}*)$ optimal values. When setting the QAOA circuit with $(\vec{\beta}*, \vec{\gamma}*)$, several measurements on the computational basis will give statistics on the bits strings encoding all the possible solutions of our problem. If the optimization process went correctly, the average value of the cost function associated to these results should be as low as possible, meaning that the optimal solutions of our problem are encoded by the bits strings measured with the highest frequencies. The ultimate goal of QAOA is to only measure the bits strings corresponding to the optimal solutions, and we will see that this can be done by increasing the depth p of the quantum circuit.

3.3 Results

The study of the QAOA resulted in the creation of a Python code (available in the appendices section), enabling us to run this quantum algorithm on both a simulator and an IBM quantum processor. The IBM simulator used intends to simulate the behavior of the quantum circuit on a real quantum computer. We advise the reader to take a look at the code, as it is explained as much as possible with comments and text sections and is an additional (though not necessary) resource for the good understanding of this report.

3.3.1 Simulations

p = 1

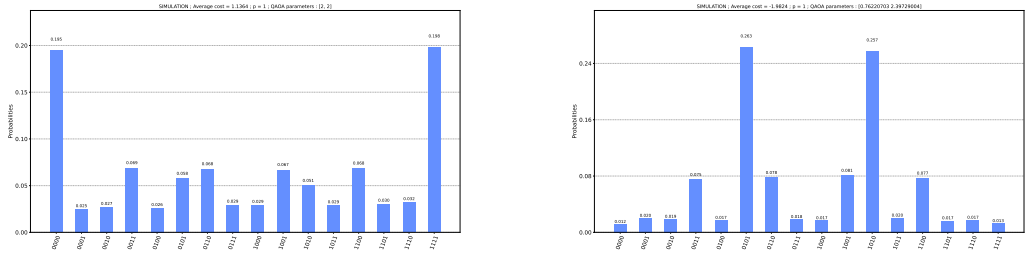

Figure 7: QAOA quantum circuit measurement statistics, for initial $(\vec{\beta_0}, \vec{\gamma_0})$ values and optimal $(\vec{\beta}*, \vec{\gamma}*)$ values $(p = 1)$

With initial $(\vec{\beta_0}, \vec{\gamma_0})$ values giving the worst possible solutions ("1111" and "0000"), the optimizer achieves to find the optimal $(\vec{\beta}*, \vec{\gamma}*)$ values, distinguishing two optimal solutions ("0101" and "1010") among all the possible ones. However, this result is not achieved with 100% certainty as bits strings encoding non optimal solutions are still present in the measurement outcomes, preventing us to have the lowest average cost value possible (-4), giving us -1.9824. Therefore we should increase the depth of our circuit to see if any improvements can be made to these results.

As the QAOA depends only on 2 parameters for $p = 1$, it is interesting to represent the average cost function for different values of β and γ, in order to see the values taken by the optimizer during its process. We can indeed notice thanks to this 2D landscape of the average cost value that QAOA with $p = 1$ is unable to reach to lowest value possible of -4.

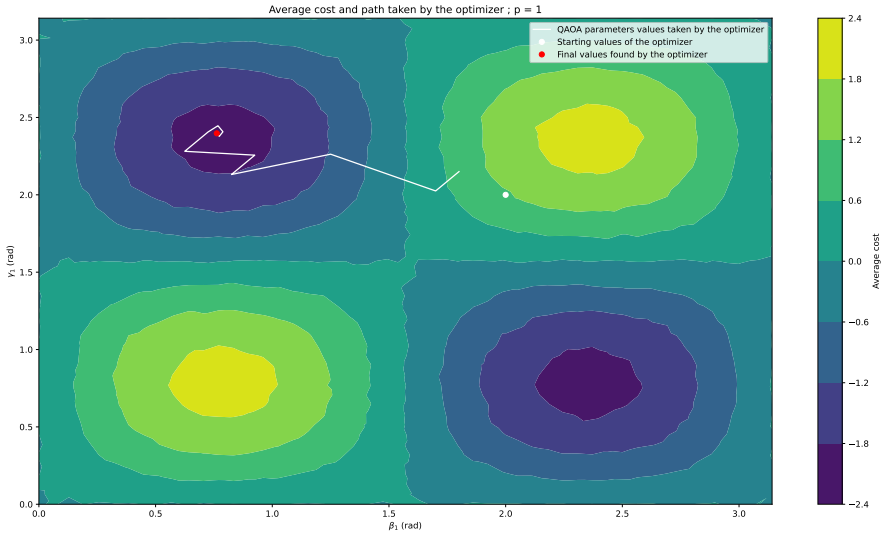

Figure 8: Representation of the average cost value depending on β_1 and γ_1, with the values taken by the optimizer ($p = 1$)

However, even if these results are not the best expected ones, noticing the convergence of the QAOA parameters values and the average cost value computed ensures that the optimizer worked in the best way possible (even if it does not ensure good optimal values found, as we can visualize in the appendix).

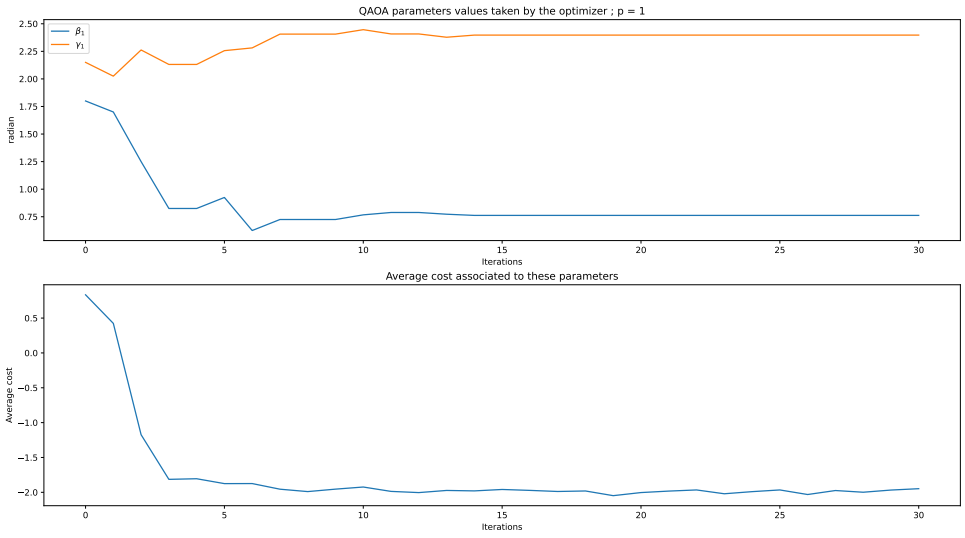

Figure 9: Convergence of the QAOA parameters and the average cost value associated, quantum circuit run on a simulator with $p = 1$

p = 2

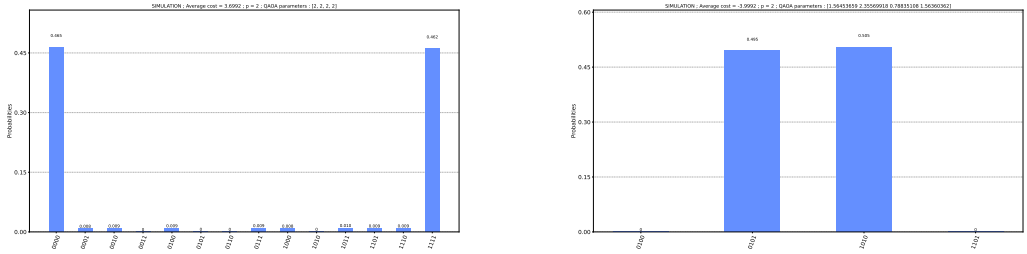

Figure 10: QAOA quantum circuit measurement statistics, for initial $(\vec{\beta_0}, \vec{\gamma_0})$ values and optimal $(\vec{\beta^*}, \vec{\gamma^*})$ values $(p = 2)$

Indeed, $p = 2$ QAOA is showing very good results as we now have, thanks to the optimal values found by the classical optimizer, an average cost value of -3.9992. The measurement outcomes enable us to clearly determine the two optimal solutions of our problem encoded by the bits strings 0101 and 1010, being certain that no other one is possibly an optimal solution.

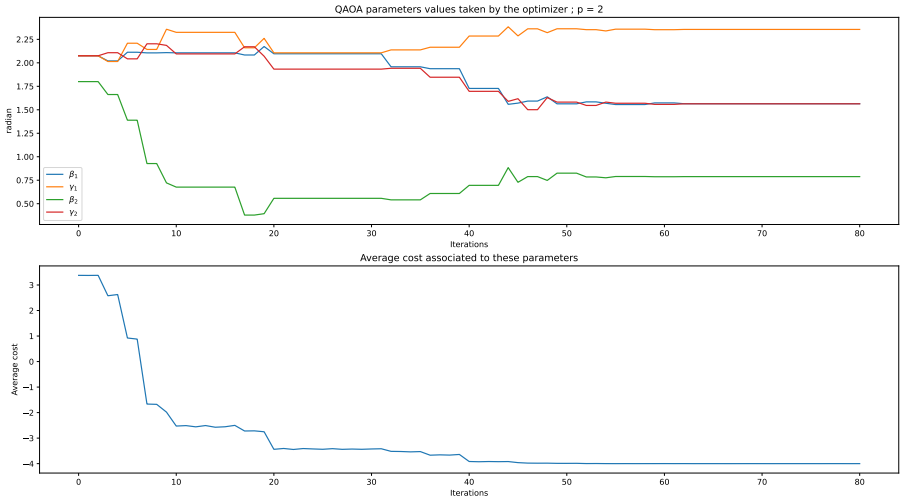

Figure 11: Convergence of the QAOA parameters and the average cost value associated, quantum circuit run on a simulator with $p = 2$

p = 3

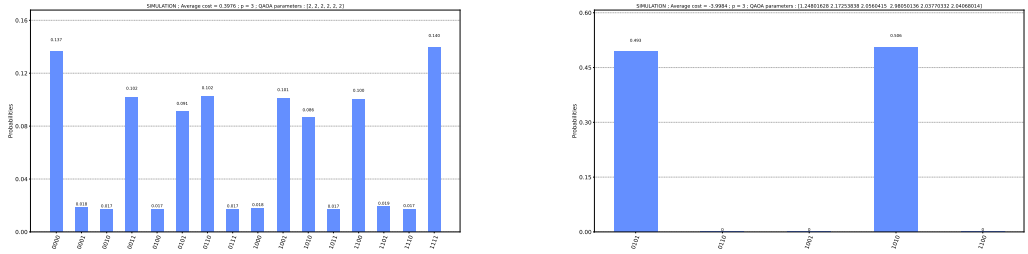

Figure 12: QAOA quantum circuit measurement statistics, for initial $(\vec{\beta_0}, \vec{\gamma_0})$ values and optimal $(\vec{\beta^*}, \vec{\gamma^*})$ values $(p = 3)$

Increasing again the depth of our circuit to $p = 3$ confirms another time that we found the optimal solutions to our problem. Indeed we almost reach the lowest cost function value again with -3.9984.

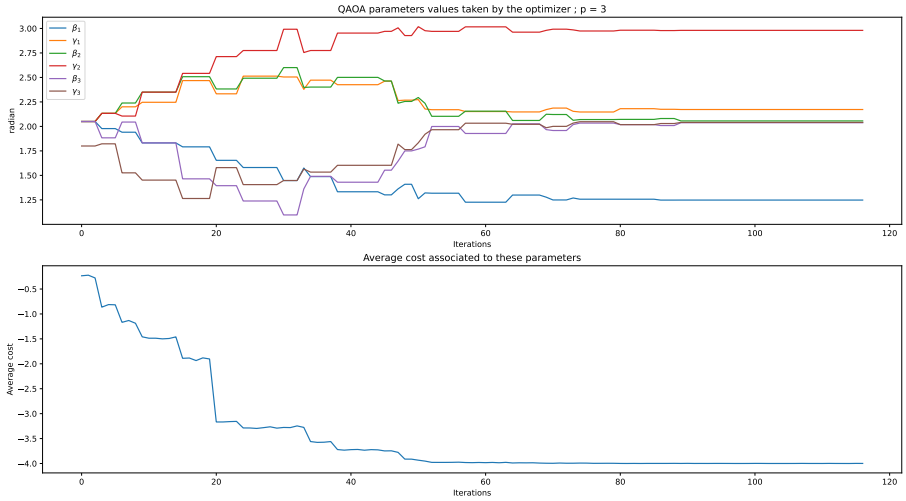

Figure 13: Convergence of the QAOA parameters and the average cost value associated, quantum circuit run on a simulator with $p = 3$

3.3.2 QAOA optimal parameters on a real quantum computer

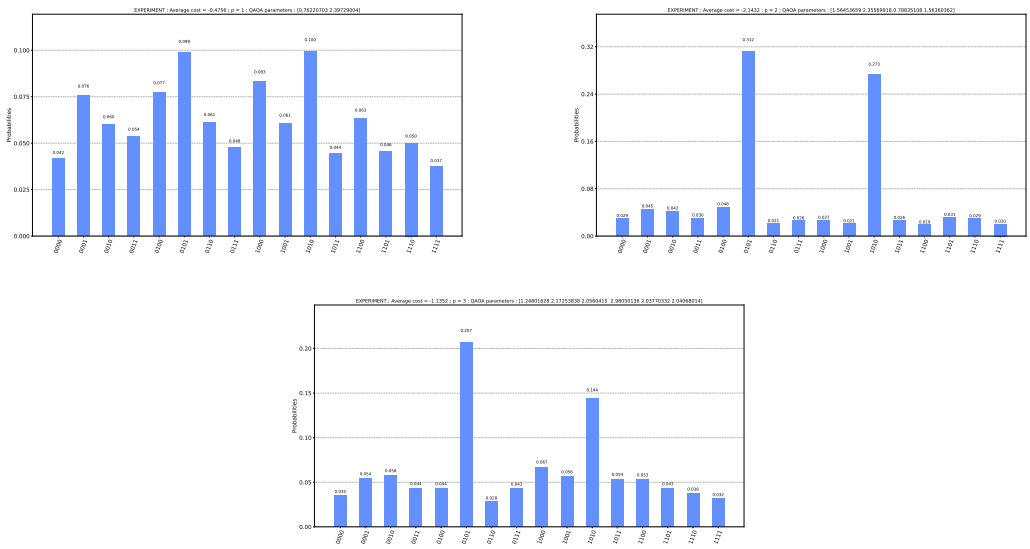

Figure 14: QAOA quantum circuit measurement statistics, for $p = 1$, $p = 2$, $p = 3$ (with optimal $(\vec{\beta}^*, \vec{\gamma}^*)$ values)

Running the QAOA on a real quantum computer with the optimal parameters values found by the optimizer gives us noticeably different results than on the simulator. As for $p = 1$, even if we saw with the simulation that we are not able to clearly confirm an optimal solution, we are now not able to guess at all which bits string could be a good candidate to represent such an optimal solution. Indeed, the average cost value of -0.475 gives no information on how optimal or not are these possible solutions.

By increasing the depth with $p = 2$, we can be more confident with the results, as we indeed notice two solutions standing out. However this is not confirmed with a total certainty, as the average cost value of -2.1432 is still far from -4.

One could have expected better results with a $p = 3$ QAOA, but this is not the case as the average cost value increased compared to the one of $p = 2$ QAOA.

We may wonder why does a simulator performed better than a quantum computer. Actually, quantum computers are subject to a lot of experimental constraints that prevent them from performing as best as they possibly could. Qubits can be perturbed by quantum noise due to their interactions with the environment, the technological default of their implementation or the fact that qubits are not working at the absolute $0K$ temperature.

4 Conclusion

By allowing variational parameters in the Quantum Approximate Optimization Algorithm, we showed that it was able to find the best possible solutions of a simple combinatorial optimization problem. This way, we were able to verify, as expected from the work of *Farhi et al* [2] that increasing the depth of the quantum circuit ensures more accurate results and hence trustworthy solutions. In our case, a depth of $p = 2$ is sufficient to find the optimal solutions of a 4 nodes graph Max-Cut problem. However, as quantum computers already show confusing results for such a simple graph compared to simulators, we might wait for better stability of these ones in order to solve more difficult combinatorial optimization problems, like the Max-Cut ones associated to larger and more complex graphs.

References

[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 (Dec. 1992), pp. 553–558. ISSN: 0962-8444, 2053-9177. DOI: 10.1098/rspa.1992.0167. URL: https://royalsocietypublishing.org/doi/10.1098/rspa.1992.0167.

[2] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. DOI: 10.48550/ARXIV.1411.4028. URL: https://arxiv.org/abs/1411.4028.

[3] Stuart Hadfield et al. "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz". In: *Algorithms* 12.2 (Feb. 2019), p. 34. DOI: 10.3390/a12020034. URL: https://doi.org/10.3390%2Fa12020034.

[4] Nielsen, Michael A. and Chuang, Isaac L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge: Cambridge University Press, 2010. URL: https://doi.org/10.1017/CBO9780511976667.

[5] *Qiskit : An Open-source Framework for Quantum Computing*. 2021. URL: https://qiskit.org/.

[6] Tommaso Toffoli. "Reversible computing". In: *Automata, Languages and Programming*. Ed. by Jaco de Bakker and Jan van Leeuwen. Springer Berlin Heidelberg, 1980, pp. 632–644. ISBN: 978-3-540-39346-7.

[7] Pontus Vikstål et al. "Applying the Quantum Approximate Optimization Algorithm to the Tail-Assignment Problem". In: *Physical Review Applied* 14.3 (Sept. 2020). DOI: 10.1103/physrevapplied.14.034009. URL: https://doi.org/10.1103%2Fphysrevapplied.14.034009.

List of Figures

List of Tables

Appendices

Python code

All the results presented in the report were made thanks to a Python code created during this project. This code is able to perform QAOA applied to the Max-Cut problem on a simulator or a real quantum computer.
As a Python notebook can't be displayed in a good way in a report, the whole code is available here : **gitlab.com/AmorosettiG/bachelor-thesis-l3-ps**.
We advise the reader to take a look at it, and even to download it and run it locally, as it is explained as much as possible with comments and text sections (the code may require installations of additional Python packages).

Tensor product

The tensor product between two matrices can be defined as :

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \otimes \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} a_{1,1}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} & a_{1,2}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \\ a_{2,1}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} & a_{2,2}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \otimes \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & a_{1,2}b_{2,1} & a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} \\ a_{2,1}b_{2,1} & a_{2,1}b_{2,2} & a_{2,2}b_{2,1} & a_{2,2}b_{2,2} \end{bmatrix}$$

Costs of all the possible 4 nodes circular graphs

Bits string	Cost of the associated graph
0000	+4
0001	0
0010	0
0011	0
0100	0
0101	-4
0110	0
0111	0
1000	0
1001	0
1010	-4
1011	0
1100	0
1101	0
1110	0
1111	+4

Table 1: Costs of all the possible 4 nodes circular graphs

Matrix representation of the problem Hamiltonian

$$H_P = \begin{bmatrix} +4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +4 \end{bmatrix}$$

Eigenvectors and eigenvalues of H_P

Statevector (16×1 vector)	Eigenvalue associated to H_P (16×16 matrix)
$\|0000\rangle$	+4
$\|0001\rangle$	0
$\|0010\rangle$	0
$\|0011\rangle$	0
$\|0100\rangle$	0
$\|0101\rangle$	-4
$\|0110\rangle$	0
$\|0111\rangle$	0
$\|1000\rangle$	0
$\|1001\rangle$	0
$\|1010\rangle$	-4
$\|1011\rangle$	0
$\|1100\rangle$	0
$\|1101\rangle$	0
$\|1110\rangle$	0
$\|1111\rangle$	+4

Table 2: Eigenvectors and eigenvalues of H_P

Actions of the usual single qubit gates

Initial state	Single qubit gate applied	Output state
$\alpha\|0\rangle + \beta\|1\rangle$	X	$\beta\|0\rangle + \alpha\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	Y	$-i\beta\|0\rangle + i\alpha\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	Z	$\alpha\|0\rangle - \beta\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	H	$\alpha\frac{\|0\rangle + \|1\rangle}{\sqrt{2}} + \beta\frac{\|0\rangle + \|1\rangle}{\sqrt{2}}$

Table 3: Actions of the usual single qubit gates

Importance of initial values for the optimizer

Thanks to the representation of the average cost value ($p = 1$), we can visualize the effect of the choice of initial values for the QAOA. Indeed, explaining it with simple words, we can see on the figure 15 that the optimizer is stuck behind a maximum, and will have better results with going to lower values instead going to higher ones, and will only be able to find a local minimum.

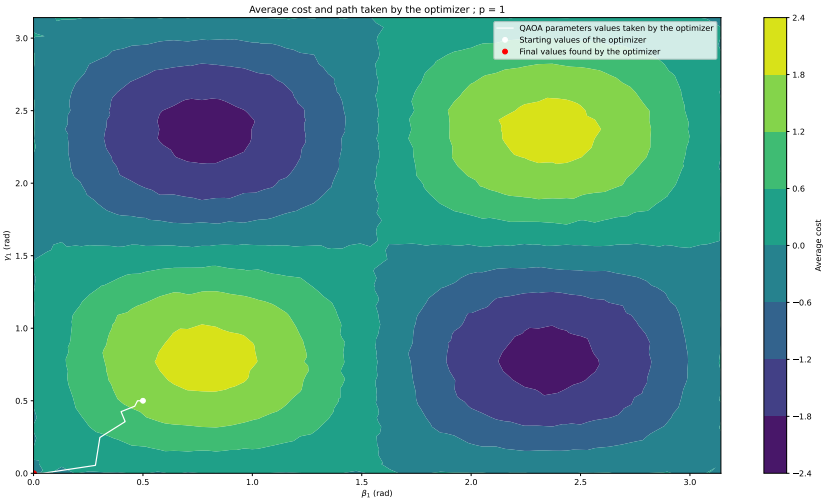

Figure 15: Representation of the average cost value depending on β_1 and γ_1, with the values taken by the optimizer ($p = 1$)

The QAOA results obtained with these initial values confirm this, as there is not any solution standing out.

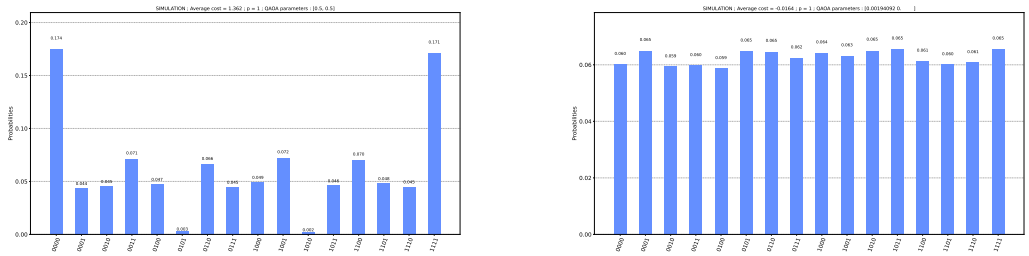

Figure 16: QAOA quantum circuit measurement statistics, for initial $(\vec{\beta_0}, \vec{\gamma_0})$ values and optimal $(\vec{\beta^*}, \vec{\gamma^*})$ values ($p = 1$)

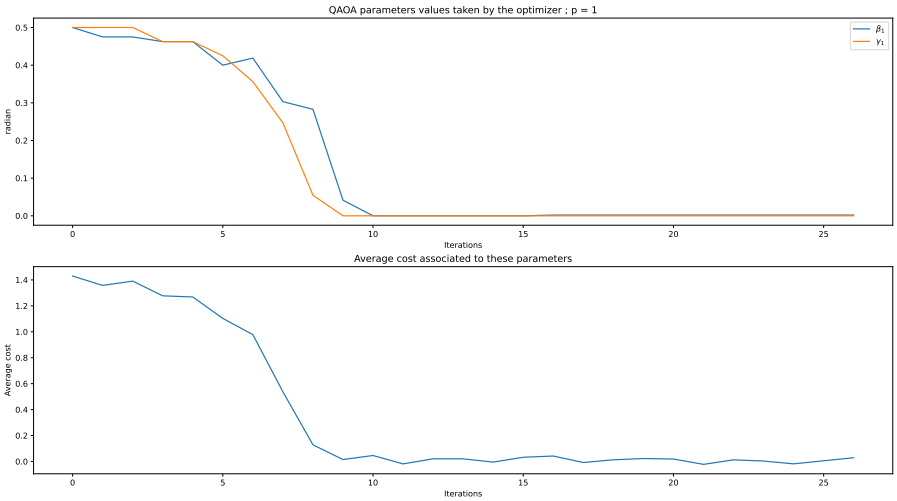

Figure 17: Convergence of the QAOA parameters and the average cost value associated, quantum circuit run on a simulator with $p = 1$