Facultad de Ciencias

Study of quantum computing techniques for the resolution of optimization problems

Estudio de técnicas de computación cuántica para la resolución de problemas de optimización

Trabajo de fin de Máster
para acceder al

MÁSTER INTERUNIVERSITARIO EN CIENCIA DE DATOS

Autor : Nicolò Trevisani
Director : Diego Porras Torre
Co-director : Pablo Martínez Ruíz del Árbol

Septiembre 2020

Contents

Chapter 1

Introduction

Quantum computing is becoming a more popular topic day after day. The recent announcements by Google [1] and IBM [2] regarding the creation of novel quantum devices of unprecedented computing capabilities brought the attention of academy and companies. Still far from being a reality, algorithms designed to work on quantum machines are currently a test bench for the abilities of this technology. Particular interest resides in the possibility of facing problems considered as untreatable by classical computers. Well-known examples are combinatorial optimization problems, in which the computational cost diverges exponentially, with increasing complexity. In this work, we considered a set of *quadratic unconstrained binary optimization* (QUBO) [3] problems, more specifically *max-cut* problems. Given a non-directed graph with n vertices and m edges, the purpose of max-cut problems consists in finding the two complementary subsets of vertices such that the number of edges connecting vertices of different subsets is maximal. For our research, we considered a slightly more general formulation, in which each edge has an arbitrary weight, and the objective function to maximize is the weight of edges connecting the two vertices subsets.
Using the Ising formulation [4], it is possible to map a QUBO problem into the Hamiltonian H of a quantum system, with energy states corresponding to the objective function values. It is then possible to find the max-cut problem solution by finding the quantum state that maximizes the energy of the quantum system. To map a max-cut problem into a Hamiltonian, each vertex of the graph is associated with a quantum bit. Quantum bits are the basic pieces of quantum computers, and can be in any superposition of a fundamental state $|0\rangle$ and excited state $|1\rangle$. In our mapping, a quantum bit in the fundamental state means that the corresponding vertex is assigned to the first of the two complementary subsets, while a quantum bit in the excited state means that the vertex is in the second subset.
For this work, we used circuit-based quantum computers. They change the qubits states using quantum logic gates, that act as unitary operators. A set of qubits and quantum gates coherently arranged is called a circuit. To solve the problems, we used the *variational quantum eigensolver* [5] (VQE) algorithm and the *quantum approximate optimization algorithm* [6] (QAOA). They are hybrid quantum-classical algorithms and use parameterized quantum circuits $C(\theta)$ to generate a wave func-

tion $|\psi(\theta)\rangle$ that encodes the state of the system [7]. Employing a quantum device to evaluate the system energy state and a classical optimizer to tune the circuit parameters θ, hybrid quantum algorithms aim to find the optimal solution to the problem.
Key questions regarding quantum computing concern not only the reliability of the provided solutions but also if it can solve optimization problems faster than classical approaches. One crucial parameter of VQE and QAOA affecting both topics is the number of circuit evaluations N needed to get a reliable estimation of the system's expected value, since each circuit measurement returns only one of the possible energy states of the system. For the classical optimizer to choose the correct set of parameters θ maximizing the system energy, it is necessary to have a good knowledge of its mean value for a given parameters choice. If on one hand, large N helps the classical optimizer to precisely measure the objective function, on the other hand, it can also affect the algorithm efficiency, as the computational time is proportional to the number of circuit evaluations. To quantitatively study the effect of varying N, we designed several max-cut problems, changing the number of vertices and edges of the corresponding graphs. We then use the VQE and QAOA algorithms to solve them, observing how the choice of N affects the quality of the solution obtained, compared with the optimal solution computed through an exhaustive search. Our results indicate that for very low N, the algorithms cannot find a sensible solution, just returning random values. Once a certain threshold, only slightly affected by the problem characteristics, is overcome, the accuracy of the solution improves as the square root of N, reaching the exact solution for $N \to \infty$. All the pieces of code implemented for this work are available on github at: `https://github.com/NTrevisani/TFM`. The production of all the numerical simulations presented in this work used the open-source library Qiskit [8] developed by IBM. They are stored in Zenodo at: `http://doi.org/10.5281/zenodo.4013213`

Chapter 2

Combinatorial optimization and Quantum Mechanics

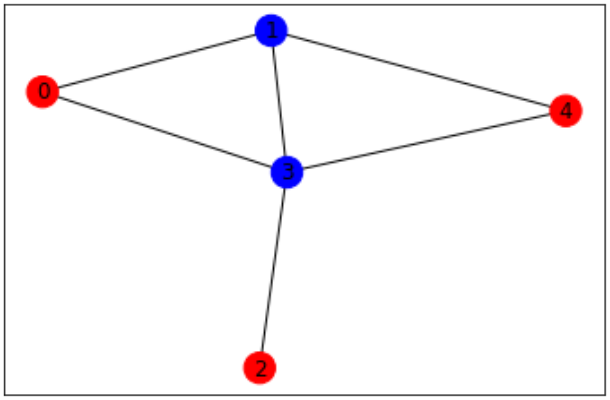
Combinatorial optimization is the task of finding the optimal solution to a problem among a finite set of possible solutions. In many cases, the set of possible solutions is so big that an exhaustive search is not viable, and it is more convenient to approximate the optimal solution. In this chapter, we will introduce quadratic unconstrained binary optimization (QUBO) [3] problems, as examples of problems requiring approximate resolution algorithms, and their parallelism with quantum systems. We will focus in particular on the max-cut problem, that we used as a benchmark in this work, presenting examples to ease the discussion.

2.1 Max-cut problem

Quadratic unconstrained binary optimization (QUBO) problems are typical examples of problems requiring combinatorial optimization. The target is to maximize an objective function:

$$F(x) = \sum_{i,j=1}^{n} x_i Q_{ij} x_j \tag{2.1}$$

where Q is a matrix defined by the problem and $x \in \{0,1\}^n$. It is easy to show that the total number of possible solutions to a QUBO problem is 2^n so that it rapidly diverges when the problem size increases.
Among the many QUBO problems, we focused on the max-cut, as it is easy to visualize, implement, and customize. To define it, let's consider a non-directed graph with n vertices and m edges. The max-cut problem aims to find the two complementary subsets of vertices such that the number of edges connecting vertices of different subsets is maximal. In a slight variation, we can consider that each edge has a weight, so that we want to maximize the sum of the weights given by edges connecting vertices of different subsets. If we define a $n \times n$ matrix, in which the entry (i, j) corresponds to the weight of the edge connecting the i and j vertices, we obtain the Q matrix associated to the max-cut problem. Each vertex is assigned a position in a string of bits and its belonging to one of the groups is determined by



0	7	0	9	0
7	0	0	7	3
0	0	0	5	0
9	7	5	0	8
0	3	0	8	0

Figure 2.1: On the left, a non-directed graph with 5 vertices and 6 edges. Each edge has been assigned a random weight, shown on the right in the matrix. Vertices of different colors belong to different subsets, according to the solution of the max-cut problem associated to the graph. The solution to this specific problem is x = {(0,1,0,1,0),(1,0,1,0,1)}, giving a cost function of 32. In general, a max-cut problem has at least two symmetric solutions, as completely exchange the vertices subsets does not change the objective function value.

the value of the bit. Figure 2.1 illustrates an example of a max-cut problem:

- on the left, the non-directed graph, with 5 vertices and 6 edges;
- on the right, the Q matrix with the edges weights;
- the solution is presented on the left, where vertices of different colors belong to different subsets.

The objective function (or cost function) for the max-cut problem is slightly different with respect to the general $F(x)$ defined in equation 2.1, and takes the form:

$$F_{max-cut}(x) = \sum_{i,j=1}^{n} x_i Q_{ij}(1 - x_j) \tag{2.2}$$

where we can see that if two vertices belong to the same subset, the contribution to the cost function is 0 (at least one of the two terms x_i or $(1 - x_j)$ is 0), while for vertices belonging to complementary subsets, the contribution is Q_{ij}.

2.2 The Ising formalism

As we want to solve QUBO problems using quantum devices, it is useful to know that there is a direct correspondence between the Q matrix of a QUBO problem and the Hamiltonian representing a quantum system whose energy states are the possible values of the objective function. This alternative formulation of optimization problems was first described by Ising [4]. Given the cost function of a QUBO

problem (see equation 2.1), the corresponding Ising Hamiltonian is given by:

$$\hat{H} = \sum_{i<j}^{n} J'_{ij}\sigma_i^z\sigma_j^z + \sum_{i=1}^{n} J''_i\sigma_i^z \tag{2.3}$$

where:

$$\begin{aligned} J'_{ij} &= -Q_{ij} \\ J''_i &= -\sum_{j=1}^{n} Q_{ij} \end{aligned} \tag{2.4}$$

and σ_i^z is the Pauli operator acting on the quantum state of the i-th vertex:

$$\sigma_i^z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}_i \tag{2.5}$$

Note the minus signs in equation 2.4: in the Ising formulation, and from this point on, the problem solution will correspond to the state giving the *minimum* energy of the system, and we will find it by minimizing the objective function.
Operatively, given the objective function (equation 2.2) of a max-cut problem, to get its corresponding Ising Hamiltonian:

- we promote x_i to the operator:
  $x_i \to \frac{1}{2}(\mathbb{1}_i + \sigma_i^z) = \frac{1}{2}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_i + \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}_i\right) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}_i$
- consequently:
  $(1 - x_i) \to \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_i - \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}_i\right) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}_i$
- for every entry of the original $n \times n$ Q matrix, we take the entry (i, j) and multiply it by the tensor product from 1 to n of 2×2 identities, with the exception of the i-th and j-th terms of the product, which are the 2×2 matrices corresponding to x_i and $(1 - x_j)$. The sum of all these products is a diagonal $2^n \times 2^n$ matrix, whose entries are the possible objective function values and corresponds to the Ising Hamiltonian associated to the problem.

The result is shown in Figure 2.2, presenting the Ising Hamiltonian corresponding to the max-cut problem defined in Figure 2.1.
If we now want to know to which vertices-subset configuration each $\hat{H}$ entry corresponds:

- we choose a possible solution (i.e. $x = (1, 0, 1, 0, 1)$);
- for each $x_i \in x$, we promote:
  $x_i = 0 \to \begin{bmatrix} 0 \\ 1 \end{bmatrix}$;
  $x_i = 1 \to \begin{bmatrix} 1 \\ 0 \end{bmatrix}$;

- we compute the tensor product $x' = \bigotimes_{i=1}^{n} x_i$, obtaining a vector of length 2^n;
- the matrix product $(x'^*)^T \hat{H} x'$ returns the desired value.

Since it is widely used in quantum mechanics and quantum computing, let's conclude this chapter introducing the Dirac bra-ket notation:

- vectors are labelled as *kets*: $x_i \to |x_i\rangle$
- vectors conjugate transpose are labelled as *bras*: $(x_i'^*)^T \to \langle x_i|$
- the matrix product previously introduced can be written as $\langle x_i| \hat{H} |x_i\rangle$ and takes the name of *expectation value*.

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	-11	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	-29	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	-24	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	-5	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	-16	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	-24	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	-19	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	-17	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	-22	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	-32	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	-21	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	-22	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	-27	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	-27	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	-16	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	-16	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	-27	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	-27	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	-22	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	-21	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	-32	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	-22	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	-17	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	-19	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	-24	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	-16	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	-5	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	-24	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	-29	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	-11	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

Figure 2.2: The Ising Hamiltonian associated to the max-cut problem defined in Figure 2.1. It is a (32×32) diagonal matrix, with entries corresponding to the possible objective function values.

Chapter 3

Quantum computing

As quantum mechanics is the underlying law governing natural processes, it has long been considered a sensible way to describe and simulate them [9]. Despite that, only thanks to the most recent technological advances, it has been possible to develop quantum machines whose computational power cannot be reached by classical computers [1, 2]. These machines, however, are not able to radically change informatics right away [10]. Instead, they can help in developing new algorithms and, used in association with classical computers, test the real power of quantum technology. In this chapter, we will present the basic concepts of quantum computing and describe the quantum algorithms we used. The primary reference for this chapter is the Qiskit Textbook [11].

3.1 Quantum bits, circuits, and measurements

The quantum bit, or qubit, is the fundamental piece of a quantum computer. It is analogous to the bit in a standard computer, with the difference that while a bit can assume values 0 or 1, a qubit has two levels $|0\rangle$ and $|1\rangle$ and can be in any state which is their linear superposition. From a quantum mechanical point of view, $|0\rangle$ is the fundamental state and $|1\rangle$ the first excited state. In the canonical base, they can be expressed as:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3.1a}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3.1b}$$

As said, a qubit state $|\psi\rangle$ is the superposition of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3.2}$$

where α and β are complex coefficients that satisfy the relation:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{3.3}$$

When performing a measurement on a qubit state, the probability:

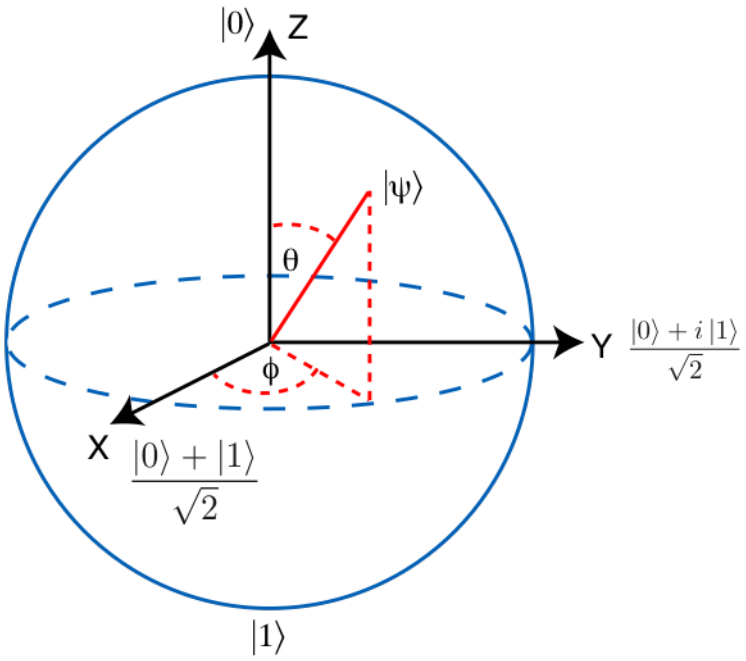


Figure 3.1: The Bloch sphere, representing a general qubit state.

- of observing it in the state $|0\rangle$ is $|\alpha|^2$;
- of observing it in the state $|1\rangle$ is $|\beta|^2$.

Due to the unitarity relation already introduced for the coefficients, the sum of the two probabilities is 1.
An alternative qubit state representation uses the probability p of measuring the qubit in the $|0\rangle$ state and the qubit phase ϕ as parameters:

$$|\psi\rangle = \sqrt{p}\,|0\rangle + e^{i\phi}\sqrt{1-p}\,|1\rangle \tag{3.4}$$

Being a probability, $0 \leq p \leq 1$, bringing us to a commonly used form to express the qubit state:

$$|\psi\rangle = \cos(\theta/2)\,|0\rangle + \sin(\theta/2)e^{i\phi}\,|1\rangle \tag{3.5}$$

where now:

- $0 \leq \theta \leq \pi$;
- $0 \leq \phi \leq 2\pi$.

With this new representation, the qubit state becomes a point on a sphere in $\mathbb{R}^3$, known as the Bloch sphere, shown in Figure 3.1. The Bloch representation is so popular since it allows defining a general operator U, able to change the qubit state from any state $|\psi\rangle$ to another state $|\psi'\rangle$:

$$|\psi'\rangle = U\,|\psi\rangle\,. \tag{3.6}$$

U has to be unitary:

$$UU^\dagger = \mathbb{1} \tag{3.7}$$

and assumes the form:

$$U(\vartheta, \varphi, \lambda) = \begin{bmatrix} \cos(\vartheta/2) & -e^{i\lambda}\sin(\vartheta/2) \\ e^{i\varphi}\sin(\vartheta/2) & e^{i(\lambda+\varphi)}\cos(\vartheta/2) \end{bmatrix} \tag{3.8}$$

Where ϑ is a rotation of the Bloch Sphere polar angle (the θ defined in Figure 3.1 and equation 3.5) and λ and φ are phases rotations that affect differently the fundamental and excited state coefficients. It is important to note here that in quantum mechanics, only phase differences have physical meaning. In other words, even if equation 3.5 has two parameters (θ and ϕ), and U has three (ϑ, φ, and λ), we are not introducing additional parameters. Instead, the coefficients phases can always be re-defined so that their difference is preserved, but the fundamental state coefficient has phase 0.
Circuit-based quantum computers change qubits state using logic gates known as *quantum gates*, and circuits are sets of quantum gates acting on one or more qubits. Quantum gates are no more than the U operator implementation, with particular parameters' choice. The most common ones and those used in this work are listed here.

Hadamard gate The Hadamard gate acts as the operator:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.9}$$

corresponding to $U(\frac{\pi}{2}, 0, \pi)$. It represents a rotation of $\frac{\pi}{2}$ around the y-axis of the Bloch Sphere, bringing a qubit from the fundamental state $|0\rangle$ to the positive superposition state $|+\rangle$:

$$H|0\rangle = |+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{3.10}$$

or from the excited state $|1\rangle$ to the negative superposition state $|-\rangle$:

$$H|1\rangle = |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{3.11}$$

X gate The X gate acts as the operator:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{3.12}$$

corresponding to $U(\pi, 0, 0)$. It represents a rotation of π around the y-axis of the Bloch Sphere, bringing a qubit from the fundamental state $|0\rangle$ to the excited state $|1\rangle$ or vice-versa.

Z gate The Z gate acts as the operator:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.13}$$

corresponding to $U(0, \pi, 0)$. It introduces a phase $\varphi = \pi$, changing the sign of the excited state coefficient.

U1 gate The U1 gate acts as the operator:

$$U1(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix} \tag{3.14}$$

corresponding to $U(0, 0, \lambda)$. It introduces a phase λ for the excited state coefficient.

U2 gate The U2 gate acts as the operator:

$$U2(\varphi, \lambda) = \begin{bmatrix} 1 & -e^{i\lambda} \\ e^{i\varphi} & e^{i(\lambda+\varphi)} \end{bmatrix} \tag{3.15}$$

corresponding to $U(\frac{\pi}{2}, \varphi, \lambda)$. It introduces different phases for the fundamental and excited state coefficients.

U3 gate The U3 gate acts as the U operator, where the three parameters ϑ, φ, and λ can be independently changed.

RX gate The RX gate acts as the operator:

$$RY(\vartheta) = \begin{bmatrix} \cos(\vartheta/2) & -i\sin(\vartheta/2) \\ -i\sin(\vartheta/2) & \cos(\vartheta/2) \end{bmatrix} \tag{3.16}$$

corresponding to $U(\vartheta, \frac{3\pi}{2}, \frac{\pi}{2})$. It rotates the qubit state around the x-axis of the Bloch Sphere.

RY gate The RY gate acts as the operator:

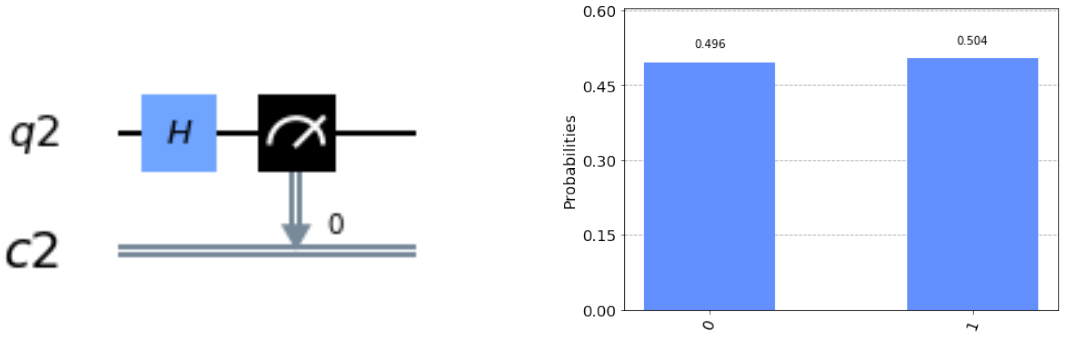
$$RY(\vartheta) = \begin{bmatrix} \cos(\vartheta/2) & -\sin(\vartheta/2) \\ \sin(\vartheta/2) & \cos(\vartheta/2) \end{bmatrix} \tag{3.17}$$

corresponding to $U(\vartheta, 0, 0)$. It rotates the qubit state around the y-axis of the Bloch Sphere.

CNOT gate The CNOT gate is the basic 2-qubits gate. Given a target qubit and control qubits, it applies an X gate to the target qubit if and only if the control qubit is in the $|1\rangle$ state. It acts on the tensor product of the control and target qubits states as the operator:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.18}$$

This gate is essential to create *entangled* states between two qubits. Entanglement is a fundamental property of quantum mechanics, with no classical analogy. A set of quantum particles (or qubits, in our case) which are put and kept in a coherent (entangled) state can behave in a strongly correlated way, even if they are too far



(a) (b)

Figure 3.2: A simple, 1-qubit circuit, with just one Hadamard gate and the corresponding measurement frequencies.

apart to let any classical local theory explaining that. In other words, the elements of an entangled quantum system do not behave independently. Instead, two entangled qubits will give correlated results when measured. In more mathematical terms, the state of an entangled system cannot be factorized as the product of single qubits states. Using the CNOT gate, any other controlled gate can be built. To avoid overloading the discussion, we will skip listing them all.

Circuit measurement

Given the probabilistic nature of quantum mechanics, if we want to know the state of a circuit of one or more qubits, we have to measure it several times. The state of a qubit is the superposition of the fundamental and excited states, but when we measure it, we can find it either in the $|0\rangle$ or in the $|1\rangle$ state, with a probability equal to the square module of the corresponding coefficient. As an example, in Figure 3.2, a simple circuit, made by one Hadamard gate acting on a single qubit, is shown together with the relative frequencies obtained measuring it 1024 times. As expected, the relative frequencies are very close to $\frac{1}{2}$, meaning that $\alpha = \beta = \frac{1}{\sqrt{2}}$. Performing more measurements would have reduced the difference between the expected and observed frequencies, but on the other hand, it would have taken more time. As we already mentioned and will see in more detail later, this is a crucial aspect to estimate the efficiency of quantum algorithms.

3.2 Quantum algorithms

The circuit-based quantum computers currently available belong to the *noisy intermediate-scale quantum* era [10]. The best available qubits today [12, 13] are noisy, meaning that quantum gates have a non-negligible probability of producing the wrong output. This probability is of the order of per-mill, meaning that circuits must contain a limited number of gates to give reliable results. While error-correction techniques exist, they require more qubits than available in intermediate-scale devices. Hybrid quantum-classical algorithms fix this situation. They use the

ability of quantum computers to efficiently measure the state of a circuit, working together with a classical computer to iteratively tune the circuit parameters and minimize the objective function. Thanks to this continuous interaction, the algorithms use small quantum circuits, and possible errors are less relevant.
In this work, we considered the variational quantum eigensolver (VQE) [5] algorithm and the quantum approximate optimization algorithm (QAOA) [6]. In both cases, the classical optimizer was the Constrained Optimization BY Linear Approximation (COBYLA) algorithm [14]. It applies linear approximations of the cost function, sampling it to estimate its gradient when it is not known a priori.

3.2.1 Variational quantum eigensolver

To understand the idea behind VQE, we start by introducing the concepts of eigenvalue and eigenvector (or eigenstate), defined through the following relation:

$$H\left|\psi_i\right\rangle = \lambda_i \left|\psi_i\right\rangle \tag{3.19}$$

where H is the matrix describing the Hamiltonian, λ_i is an eigenvalue, and $|\psi_i\rangle$ is an eigenvector. In our case, H is the Ising Hamiltonian associated to the problem, the eigenvalues its diagonal entries (and correspond to the possible cost function values), and the eigenstates are the vectors representing the possible states of our system:

$$\left|\psi_i\right\rangle = \{\left|00000\right\rangle, \left|00001\right\rangle, \left|00010\right\rangle, \left|00011\right\rangle, ...\}. \tag{3.20}$$

Here, each $|\psi_i\rangle$ is the tensor product of the vertices states, as explained in section 2.2. In the same way as to each possible solution to a QUBO problem corresponds one objective function value, to each eigenstate correponds one eigenvalue. So, if we express an optimization problem in terms of a quantum system, its solution will be the eigenstate associated with the minimum eigenvalue. In general, any other state or superposition of states:

$$\left|\psi\right\rangle = \sum_i^N \alpha_i \left|\psi_i\right\rangle \tag{3.21}$$

will give an expected value larger than the minimum eigenvalue:

$$\begin{aligned} \lambda_{min} &= \langle\psi_{min}|H|\psi_{min}\rangle \leq \langle\psi|H|\psi\rangle = \\ &= \left(\sum_i^N \left\langle\psi\right|_i \alpha_i\right) H \left(\sum_i^N \alpha_i \left|\psi_i\right\rangle\right) = \\ &= \sum_i^N |\alpha_i|^2 \langle\psi_i|H|\psi_i\rangle = \sum_i^N |\alpha_i|^2 \lambda_i \end{aligned} \tag{3.22}$$

The VQE algorithm starts with a reasonable ansatz for $|\psi\rangle$ and iteratively changes its eigenstates composition, using the classical optimizer to find the combination that minimizes the cost function.

Ansatz selection

In general, a valid ansatz has to fulfill two requisites:

- cover the largest possible number of states → many parameters;
- have a small number of gates, to avoid errors and ease the convergence → few parameters.

Among the several possibilities, we chose the RY-ansatz. To change the single qubits states and consequently the eigenstate composition of the quantum state, it uses only rotations around the Bloch sphere y-axis. Operatively, it consists of:

1. putting all the qubit in the $|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ state through rotations of $\frac{\pi}{2}$ around the y-axis of the Bloch sphere. This is done using RY-gates and ensures that the initial choice of $|\psi\rangle$ is the superposition of all the possible eigenstates with the same coefficient. The rotation angles start in $\frac{\pi}{2}$ but can be modified during the optimization;
2. connecting different qubits, introducing entanglement among them. In our implementation, we connected nearest neighbor qubits, using controlled Z-gates;
3. introducing for each qubit an additional rotation around the y-axis of its Bloch sphere. As in step 1., RY-gates perform the rotations, but in this case the initial rotation angle is 0. The angles are free parameters to be adjusted during the optimization process;
4. repeating steps 2. and 3. for a chosen number of times, keeping in mind that more layers mean more parameters for the optimizer and more gates. In our implementation, we selected a *circuit depth* of 2, in agreement with reference [7].

An example of a VQE circuit with 5 qubits and a depth of 2 layers and prepared in the RY initial state is show in Figure 3.3. The result of measuring the circuit 1024 times are shown in Figure 3.4. As the circuit corresponds to a quantum state where all the eigenstates $|\psi_i\rangle$ are present with the same coefficient α_i, they all appear with similar frequencies.

3.2.2 Quantum Approximate Optimization Algorithm

The QAOA algorithm aims to reproduce Adiabatic Quantum Computation [15] (AQC) on a circuit-based quantum device. AQC is based on a different quantum computing paradigm, not built on circuits and gates, and needs two Hamiltonians to solve an optimization problem [16]: the driver Hamiltonian (H_D) and the problem Hamiltonian (H_P). The driver Hamiltonian has a ground state easy to prepare and is used as the starting point. The problem Hamiltonian encodes the characteristics of the problem we want to solve and its ground state (usually unknown) is the solution. The problem Hamiltonian thus corresponds to the Ising Hamiltonian of the

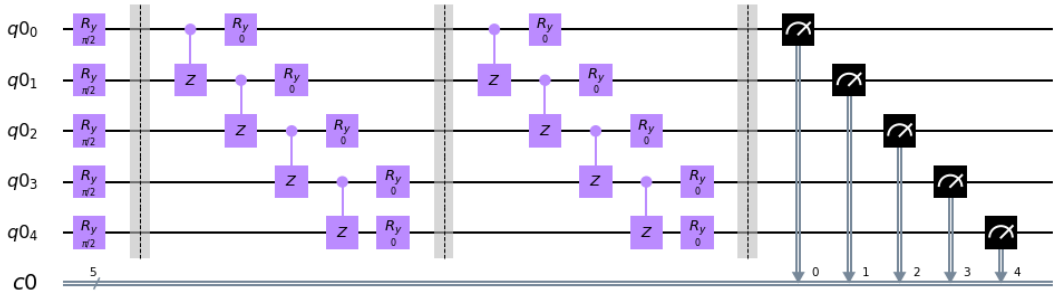


Figure 3.3: A VQE circuit with 5 qubits and a depth of 2 layers. The first RY rotations are set to $\theta = \frac{\pi}{2}$ for all qubits, while the additional gates are set to $\theta = 0$. All the angles can be freely modified during the optimization.

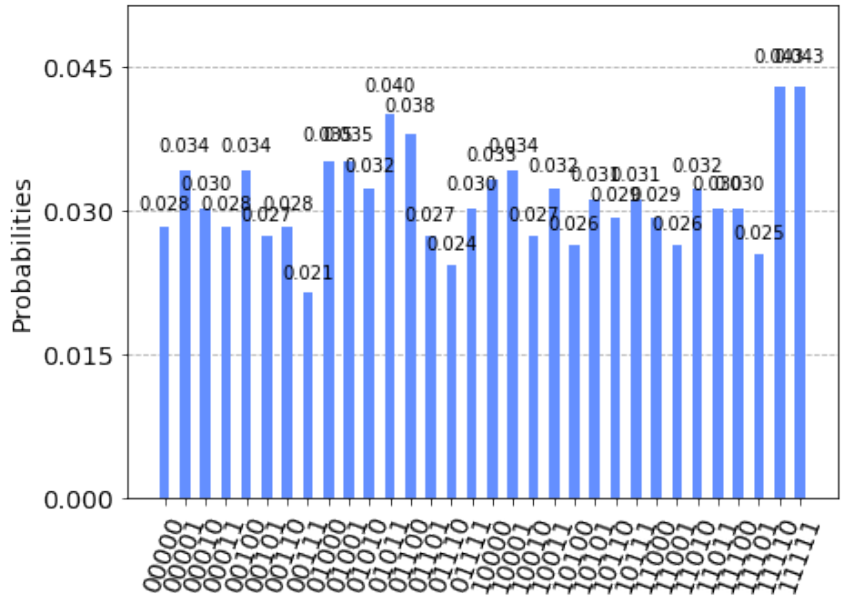


Figure 3.4: Result of measuring 1024 times the circuit in Figure 3.3. As expected, all the eigenstates appear with a similar frequency.

problem. As the Adiabatic Theorem [17] states, if we start from the ground states of a Hamiltonian (in our case H_D) and we operate a slow enough transition to another Hamiltonian (in our case H_P), we remain in the ground state, which is the solution we are looking for. More formally, we can summarize the transition from H_D to H_P by introducing a smooth function of time $s(t)$ so that:

$$\begin{cases} H(t) = (1 - s(t))H_D + s(t)H_P \\ s(0) = 0 \\ s(T) = 1 \end{cases} \tag{3.23}$$

where T is large enough for the Adiabatic Theorem to hold. The H(t) time evolution is given by the following operator:

$$U(t) = \tau \exp\left(\frac{-i}{\hbar}\int_0^T H(T)dT\right) \tag{3.24}$$

The integral in 3.24 cannot be implemented as a quantum circuit. Instead, we can use the Trotterization technique [18] to discretize it into intervals of Δt small enough so that the Hamiltonian is approximately constant over each interval. Using $U(b, a)$ to represent the time evolution from time a to time b:

$$\begin{aligned} U(T) = U(T,0) &= U(T, T-\Delta t)U(T-\Delta t, T-2\Delta t)...U(\Delta t, 0) = \\ &= \prod_{j=1}^{p} U(j\Delta t, (j-1)\Delta t) \approx \\ &\approx \prod_{j=1}^{p} e^{-iH(j\Delta t)\Delta t} \end{aligned} \tag{3.25}$$

where Δt is chosen as a multiple of $\hbar$ and the approximation improves for larger p or, equivalently, smaller Δt. Finally, we can use the approximation:

$$e^{i(A+B)x} = e^{iAx}e^{iBx} + O(x^2) \tag{3.26}$$

and plug-in the Hamiltonian:

$$H(j\Delta t) = (1 - s(j\Delta t))H_D + s(j\Delta t)H_P \tag{3.27}$$

to get an expression for the time-evolution operator that can be implemented as a quantum circuit:

$$U(T) = U(T,0) \approx \prod_{j=1}^{p} \exp\left(-i(1 - s(j\Delta t))H_D\Delta t\right) \exp\left(-is(j\Delta t)H_P\Delta t\right). \tag{3.28}$$

We can further simplify the notation by writing the time evolution operator in terms of two sets of free parameters $\beta = \{\beta_1, \beta_2, ..., \beta_p\}$ and $\gamma = \{\gamma_1, \gamma_2, ..., \gamma_p\}$:

$$U(T) \approx \prod_{j=1}^{p} e^{-i\beta_i H_D} e^{-i\gamma_i H_P} \tag{3.29}$$

Now that we know the parametric expression of the time-evolution operator, we can select an initial state and apply U(T) to go to the solution state. To translate it into a quantum circuit:

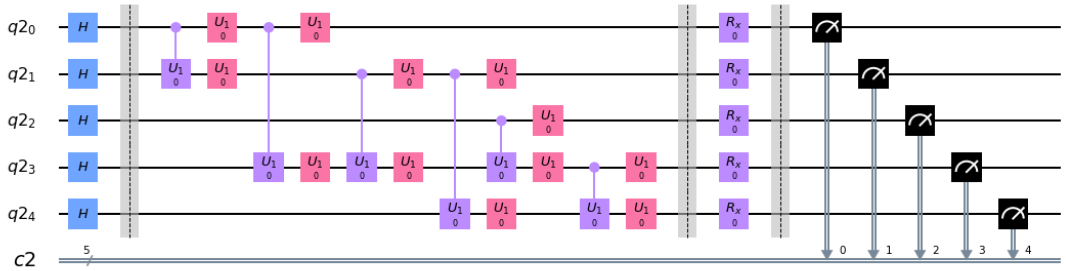


Figure 3.5: A QAOA circuit with 5 qubits and a depth of 1 layer. The circuit is based on the max-cut problem defined in Figure 2.1. Notice that the circuit has only two free parameters, β and γ. Unlike the VQE circuit of Figure 3.3, the Hadamard gates do not present freely adjustable rotation angles.

1. define the initial state, by putting all the qubits in the $|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ state. Unlike the VQE case, the initial rotation angles are fixed and cannot be adjusted during the optimization;

2. introduce the H_P rotation gates: for each qubits pair (i, j), if the $Q_{i,j}$ entry is non-null, insert a controlled U1 gate with rotation angle -2$\gamma_k \cdot Q_{i,j}$ connecting the two qubits and a U1 gate with rotation angle $\gamma_k \cdot Q_{i,j}$ for each of the two qubits;

3. introduce the H_D rotation gates: add an RX gate with rotation angle $2 \cdot \beta_k$ to each qubit;

4. repeat steps 2. and 3. p times, considering that each additional layer introduces two new parameters (γ_k, β_k) and several new gates. In our implementation, we selected a circuit depth of 2, in agreement with reference [7].

An example of a QAOA circuit based on the problem defined in Figure 2.1, with 5 qubit and depth 1, is shown in Figure 3.5. The γ and β angles are initially set to 0, so in the corresponding quantum state, all the eigenstates $|\psi_i\rangle$ appear with the same coefficient α_i. The result of measuring the circuit 1024 times, shown in Figure 3.6, confirms that all the eigenstates have been sampled with similar frequencies. To obtain the correct angle values needed to get the optimal solution, we operated as in the VQE case, measuring the cost function associated to a certain parameter choice and tune them using a classical optimizer.

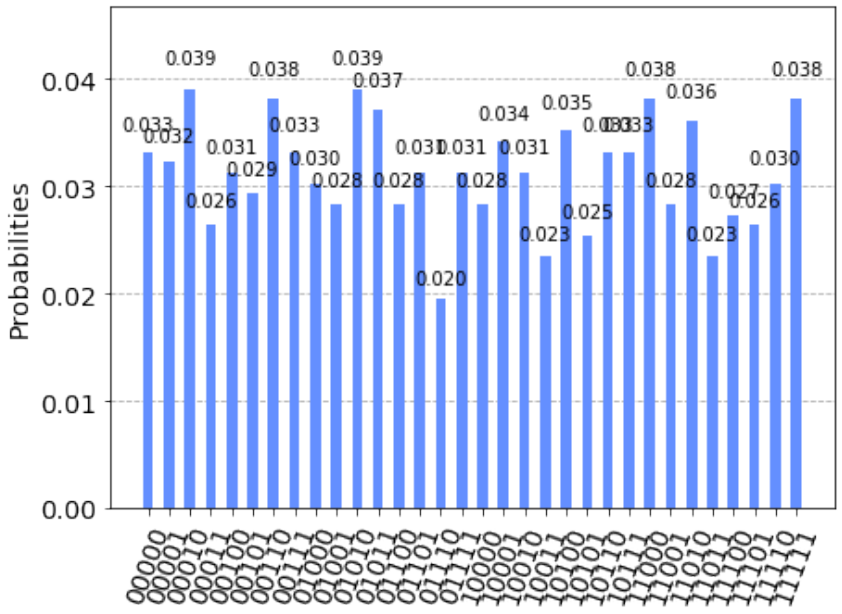


Figure 3.6: Result of measuring 1024 times the circuit in Figure 3.5. As expected, all the eigenstates appear with a similar frequency

Chapter 4

Circuit measurement studies

Now that we presented the problem we want to solve in chapter 2 and the algorithms that we used in chapter 3, we can focus on the main purpose of this work. In this chapter, we will discuss the details of our research regarding the effect of changing the number N of circuit measurements, on the convergence of quantum algorithms. We will list the studies we performed, present the results, and explain our interpretation.

4.1 Parameters scan

In this research, both the target problem and the optimization algorithms have parameters that we adjusted to broaden the validity of our investigation.
To obtain a study as complete as possible, we inspected several different max-cut problems, obtained changing the number of vertices and the number of vertices connections. We first of all wanted to determine the behavior of the two algorithms when facing max-cut problems of different size, namely associated to graphs of different number of vertices. As the maximum number of edges for a non-directed graph of n vertices is:

$$m_{max} = \frac{n(n-1)}{2}, \tag{4.1}$$

to avoid any dependence on the number of edges, the number of vertices connections was fixed to $m_{max}/2$. Additionally, the vertices connected by edges were randomly selected, as the edges weights. The complete parameters scan for this study is show in Table 4.1. In the same Table, the mean cost function (the mean of the Ising Hamiltonian eigenvalues) for each problem is listed, together with the minimum cost function, corresponding to the minimum eigenvalue. These quantities were obtained by an exhaustive search of all the possible cost function values. In a second phase, we focused the attention on the graphs complexity, once fixed the size. In this case, we selected graphs with 10 vertices, changing the edges number. A summary of the scan is presented in Table 4.2. In both cases, and for both algorithms, we used the same set of number of circuit measurements (also called shots) during the circuit parameter optimization:

$$\text{shots} = [1, 2, 4, 8, 12, 16, 24, 32, 64, 96, 128, 192, 256, 512] \tag{4.2}$$

vertices	edges	mean cost function	min cost function
10	22	-61.5	-92
11	27	-68.5	-105
12	33	-85.5	-135
13	39	-102	-157
16	60	-171	-249
18	76	-215.5	-309

Table 4.1: Parameter scan for the problem-size study. Together with the number of vertices and edges, the mean cost function and the minimum cost function of each problem are shown.

vertices	edges	mean cost function	min cost function
10	10	-28.5	-55
10	15	-42.5	-75
10	20	-55	-88
10	22	-61.5	-92
10	25	-65.5	-99
10	30	-77	-122
10	35	-90.5	-122
10	40	-106.5	-143
10	45	-124	-158

Table 4.2: Parameter scan for the graph-complexity study. Together with the number of vertices and edges, the mean cost function and the minimum cost function of each problem are shown.

For each shot value, the optimization is repeated 100 times and, independently of the measurements used during the optimization, the final circuit configuration is evaluated 128 times.

4.2 Results

Several quantities can serve to estimate the performances of the algorithms. To evaluate the algorithm ability to reach the optimal solution, we focused our attention on the final cost function value and the fraction of solutions containing the eigenstate of minimum energy. In particular, we considered the difference between the final cost function and the eigenvalues mean value, as a function of the $1/\sqrt{\text{shots}}$. As the number of total circuit evaluations is the key-value to measure the algorithm rapidity, it was the reference quantity we considered for this purpose. In the following, we will list the results of the optimizations for the two algorithms and the problem-size and graph-complexity studies.

4.2.1 VQE results

The results of the problem-size and graph-complexity studies obtained using the VQE algorithm are presented in this section.

Problem-size study

The VQE performance results for the problem-size study are illustrated in Figure 4.1. Each point represents the mean value of 100 optimizations and their standard deviation, and darker colors refer to problems related to larger graphs. In the left plot (Figure 4.1a), it can be seen that when the number of shots is small, the algorithm does not manage to properly converge, returning almost random results. Their mean is simply the ansatz cost function value, with a dispersion that decreases when the number of shots gets larger. Once a certain threshold is overcome (at around $1/\sqrt{\text{shots}} = 0.2$, corresponding to 25 shots), the final cost function starts to decrease linearly with $1/\sqrt{\text{shots}}$, reaching the optimal solution for shots $\to \infty$. It is interesting to note that the linear behavior for a large number of shots is independent of the problem size. Only for the largest shots value, the points start to diverge. Most probably, the optimization solution is getting close to the true solution and start to saturate. This can be better understood by looking at the right plot (Figure 4.1b), where the y-axis is re-scaled so that -1 corresponds to the optimal solution, independently of the problem size. It is clear the smallest problems are already very close to the true solution and the cost function cannot decrease much more. In the bottom plot (Figure 4.1c), we show the fraction of solutions containing the optimal solution, as a function of the number of circuit measurements. In this case, the result is a bit surprising, as we would expect an increasing trend when using more shots, which is not so evident. According to reference [7], it can be explained considering that the mean cost function value is not the best possible figure of merit to perform an optimization. Let's consider, for example, the results of two optimizations:

1. $|\psi_1\rangle = \dfrac{|\psi_{min}\rangle + |\psi_{max}\rangle}{\sqrt{2}}$
2. $|\psi_2\rangle = |\psi_{middle}\rangle$

where:

- $\langle\psi_{min}| \hat{H} |\psi_{min}\rangle = \lambda_{min} = -100$
- $\langle\psi_{max}| \hat{H} |\psi_{max}\rangle = \lambda_{max} = 0$
- $\langle\psi_2| \hat{H} |\psi_2\rangle = \langle\psi_{middle}| \hat{H} |\psi_{middle}\rangle = \lambda_{middle} = -90$

In other words, the mean cost function of $|\psi_1\rangle$ is -50, while the mean cost function of $|\psi_2\rangle$ is -90. This means that the algorithm would prefer $|\psi_2\rangle$ instead of $|\psi_1\rangle$, even if $|\psi_2\rangle$ does not contain the optimal solution and $|\psi_1\rangle$ does. In reference [7], an alternative to the mean cost function is presented, but we are not considering it for the moment. In Figure 4.2, we present the time-of-convergence results. In the left plot (Figure 4.2a), it can be seen that the number of iterations with the classical optimizer is larger when the problem size increases. On the other hand, this quantity only slightly depends on the number of shots. As the total number of circuit evaluation is the product between the iterations with the classical optimizer and the number of evaluations for each iteration, we show it on the right plot (Figure 4.2b). As expected, there is an almost linear dependence between the total circuit evaluations and the number of shots, with a coefficient given by the average number of iteration with the classical optimizer.

Graph-complexity study

The VQE performance results for the problem-size study are illustrated in Figure 4.3. They look very similar to the previous ones, and in particular Figure 4.3a shows that in all cases the algorithm manages to *step away* from the mean cost function value of approximately the same quantity for a given number of shots, independently of the problem. On the other hand, Figure 4.3b shows that the more complex the problem, the wider is the difference between the mean cost function and the optimal cost function, so that the convergence to the optimal value is slower for problems associated with graphs with more edges. In both cases, the typical behavior showing a threshold in the number of shots for the algorithm to start converging is still observed. Also Figure 4.3c is similar to Figure 4.1c, showing that the fraction of solutions containing the optimal solution generally grows with more shots, but can significantly oscillate. Moving to Figure 4.4a, we can see that the number of classical optimizer iterations does not depend on the graph complexity, and only moderately on the number of shots. Again, this translates in the linear dependence between the total circuit evaluations and the number of shots, as Figure 4.4b shows.

4.2.2 QAOA results

The results of the problem-size and graph-complexity studies obtained using the QAOA algorithm are presented in this section.

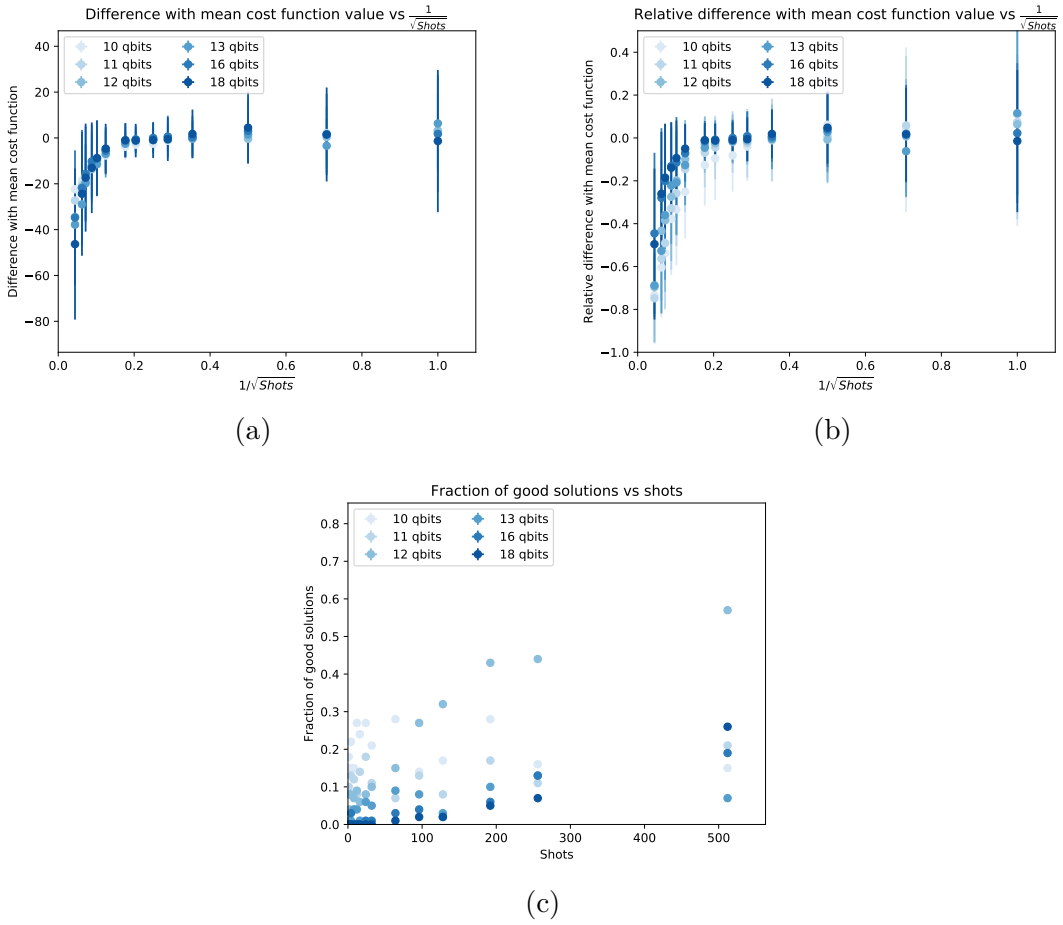
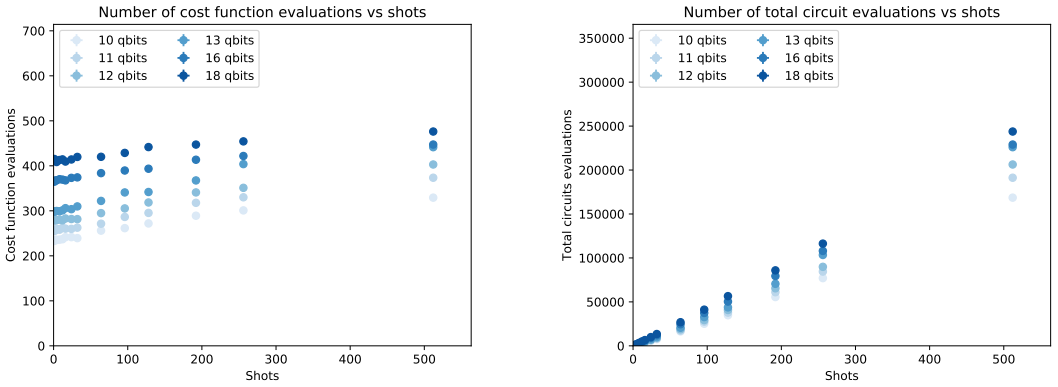


Figure 4.1: VQE algorithm performance results for the problem-size study. On the left, difference between the mean cost function (see Table 4.1) and the solution cost function as a function of $\frac{1}{\sqrt{\text{shots}}}$. On the right, the same quantity is re-scaled so that y-axis origin is the mean cost function and y-axis = -1 corresponds to the optimal solution (equivalent to the minimum cost function). In the bottom plot, the fraction of solutions including the optimal solution as a function of the number of shots.



(a) (b)

Figure 4.2: VQE time-of-convergence results for the problem-size study. On the left, number of iterations with the classical optimizer as a function of the number shots. On the right, the same quantity multiplied by the number of shots, corresponding to the total number of circuit evaluations.

Problem-size study

The QAOA performance results for the problem-size study are illustrated in Figure 4.5. Unlike the VQE case, the convergence is pretty poor and the algorithm does not seem to properly choose the optimization parameters. What we observe here is similar to what presented in reference [7], particularly for problems involving 10 or more qubits. Even if they do not show plots corresponding to Figure 4.5a and 4.5b, and their results for the fraction of solutions containing the optimal solution are not completely comparable with Figure 4.5c, they observe significantly worse results with respect to VQE, suggesting that the optimization is not taking place. They explain the observation with the small circuit depth: with just 2 trotterization steps, the classical optimizer has only 4 parameters, significantly less than the $(\text{depth}+1)\times \text{n}_{\text{qubits}}$ used by VQE. They also suggest that deeper circuits would introduce non-negligible errors so that using them is not an option. We tried to explore this possibility, by solving the 10-qubits, 22-edges problem using circuits depth of 6 and 10. The results are presented in Figure 4.6 and show no improvements. In Figure 4.7, the time-of convergence results are shown. As in the VQE case, there is only a limited dependence on the number of shots. The results are also independent of the problem size, as the number of free parameters is constant.

Graph-complexity study

We show, for completeness, the results of the graph-complexity study for the QAOA algorithm, in Figure 4.8 and Figure 4.9. They are similar to the results of the previous section and show that the algorithm is not properly converging towards the optimal solution.

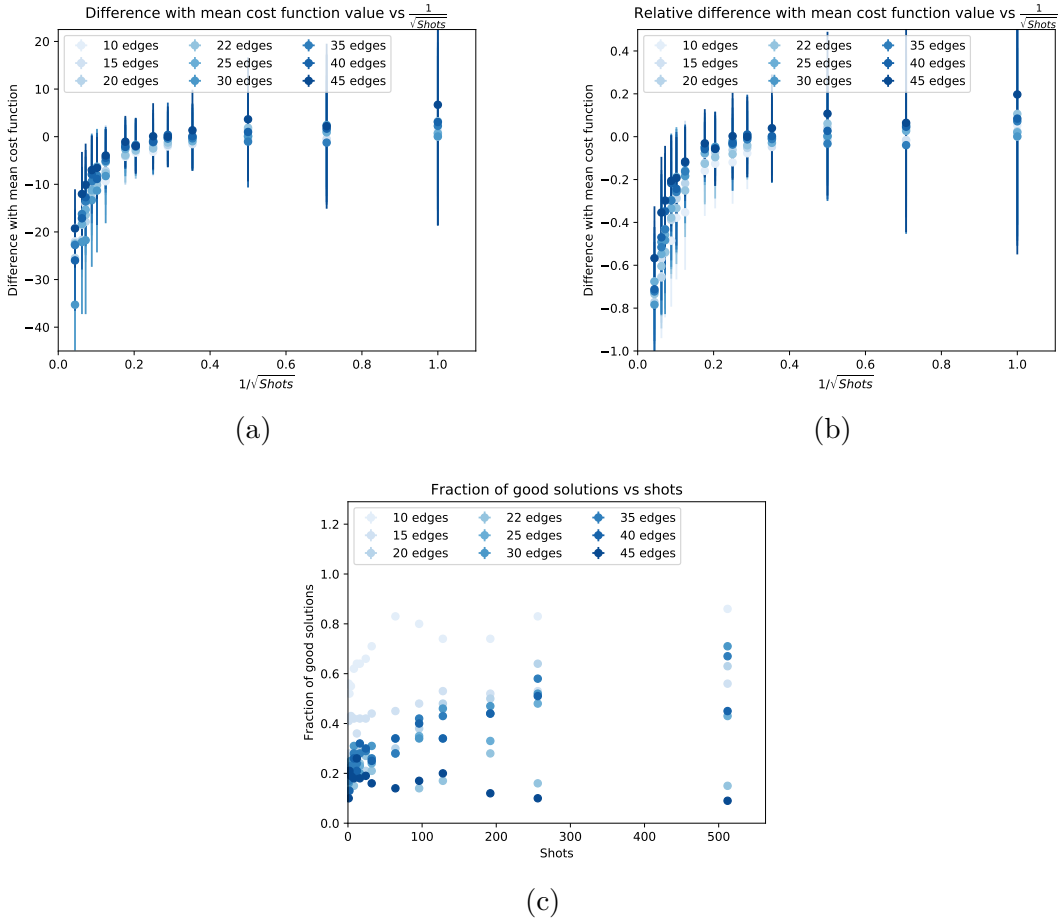
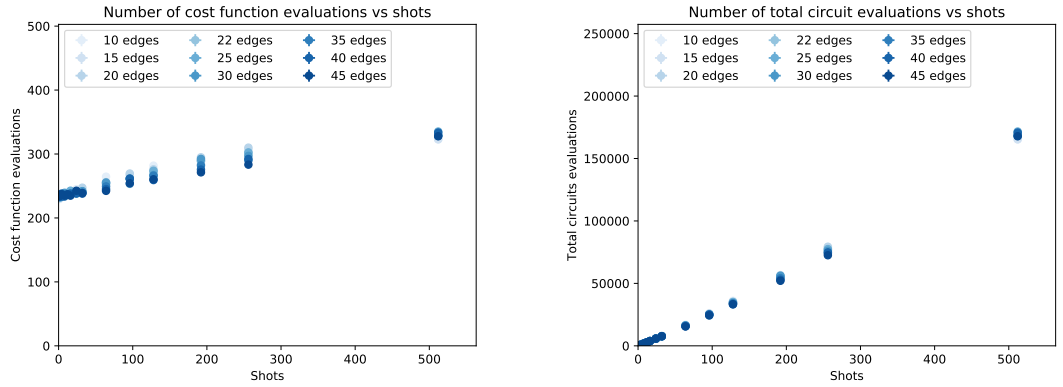


Figure 4.3: VQE algorithm performance results for the graph-complexity study. On the left, difference between the mean cost function (see Table 4.1) and the solution cost function as a function of $\frac{1}{\sqrt{\text{shots}}}$. On the right, the same quantity is re-scaled so that y-axis origin is the mean cost function and y-axis = -1 corresponds to the optimal solution (equivalent to the minimum cost function). In the bottom plot, the fraction of solutions including the optimal solution as a function of the number of shots.



(a) (b)

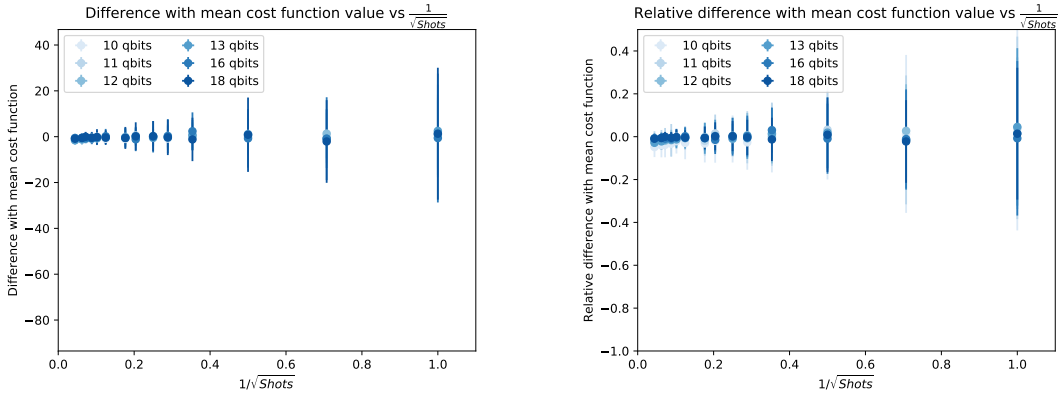
Figure 4.4: VQE time-of-convergence results for the graph-complexity study. On the left, number of iterations with the classical optimizer as a function of the number shots. On the right, the same quantity multiplied by the number of shots, corresponding to the total number of circuit evaluations.

4.3 VQE threshold effect interpretation

As observed in Figures 4.1 and 4.3, the VQE algorithm needs a minimum amount of shots to even start converging towards the optimal solution. In both Figures, this threshold is located at about $1/\sqrt{\text{shots}} = 0.2$, or shots $= 25$. To understand the reason for this behavior, we recalled that the COBYLA optimizer does not know a priori the cost-function gradient direction, and has to build it by sampling the cost function, starting from a set of input parameters. Our guess is that when the number of shots is below the threshold, the cost function value obtained is not reliable and leads to a bad gradient reconstruction, that eventually drives the algorithm towards a poor or random solution. To test our hypothesis, we proceeded as follows:

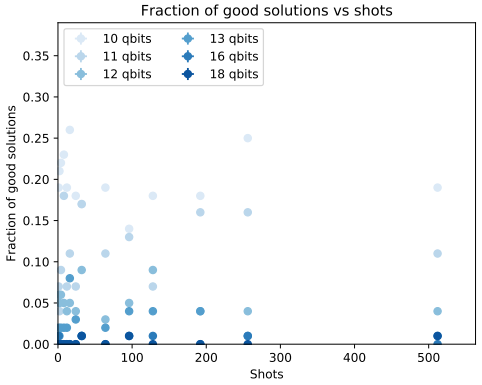
- create a random max-cut problem and a VQE circuit to solve it;
- choose a set of parameters for the VQE circuit;
- measure the circuit a large amount of times (i.e. 10000). We considered the resulting eigenstates distribution as the real eigenstates composition of the circuit. The cost function obtained was considered as the true cost function;
- measure the same circuit a smaller number of times (i.e. 1, 2, 4, 8, 12, 16, 24, 32, 64, 128, 256). Repeat each measurement 1000 times, considering the average cost function and comparing it and the eigenstates composition with the real ones.

The results of the study are shown in Figure 4.10 and, particularly for a very low number of shots, confirm our supposition. The cost function evaluations, for number of shots between 1 and 16 at least (first two rows), are so dispersed that the classical optimizer cannot reconstruct the gradient at all. Increasing the number of measurements, the measured cost function starts to have reasonable compatibility



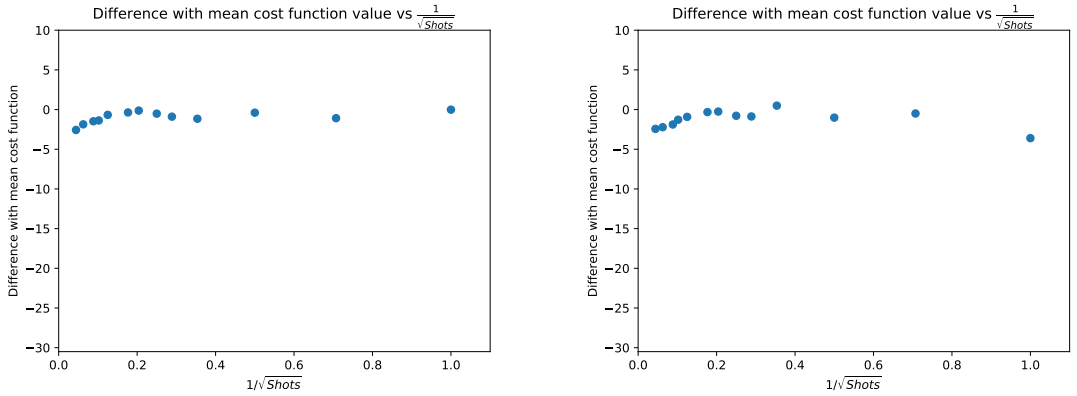
(a)

(b)



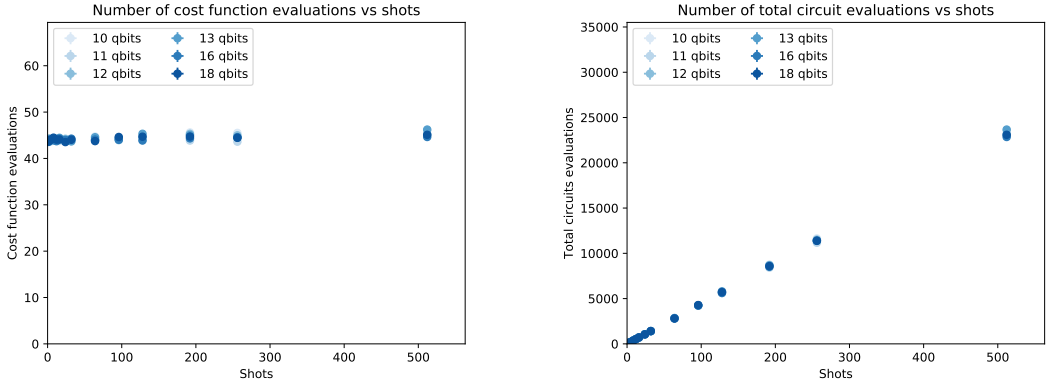
(c)

Figure 4.5: QAOA algorithm performance results for the problem-size study. On the left, difference between the mean cost function (see Table 4.1) and the solution cost function as a function of $\frac{1}{\sqrt{\text{shots}}}$. On the right, the same quantity is re-scaled so that y-axis origin is the mean cost function and y-axis = -1 corresponds to the optimal solution (equivalent to the minimum cost function). In the bottom plot, the fraction of solutions including the optimal solution as a function of the number of shots.



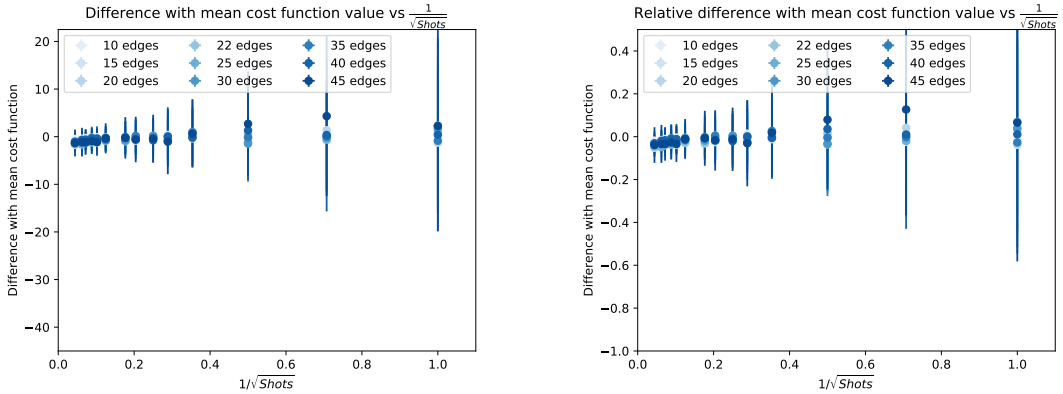
(a) (b)

Figure 4.6: Difference between mean cost function and the solution cost function, as a function of $\frac{1}{\sqrt{\text{shots}}}$. For the left plot, the optimization used a circuit with 6 trotterization layers, for the right one, the optimization used a circuit with 10 trotterization layers.



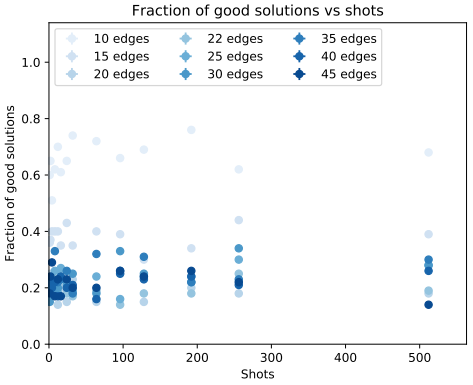
(a) (b)

Figure 4.7: QAOA time-of-convergence results for the problem-size study. On the left, number of iterations with the classical optimizer as a function of the number shots. On the right, the same quantity multiplied by the number of shots, corresponding to the total number of circuit evaluations.



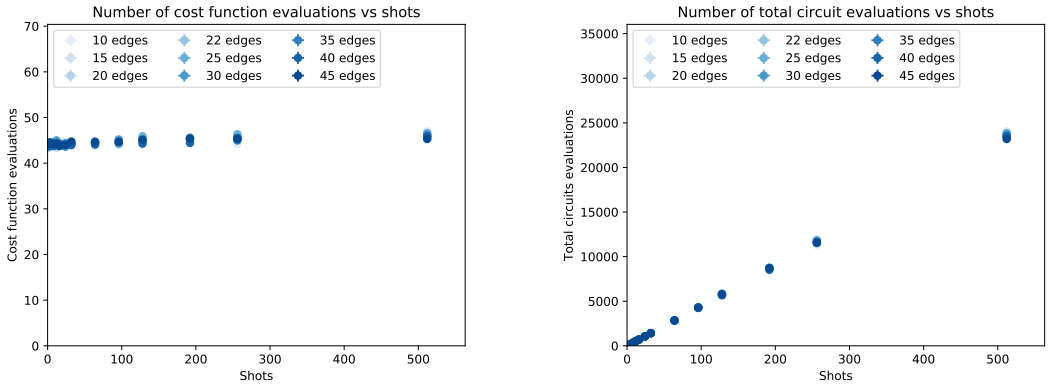
(a)

(b)



(c)

Figure 4.8: QAOA algorithm performance results for the graph-complexity study. On the left, difference between the mean cost function (see Table 4.1) and the solution cost function as a function of $\frac{1}{\sqrt{\text{shots}}}$. On the right, the same quantity is re-scaled so that y-axis origin is the mean cost function and y-axis = -1 corresponds to the optimal solution (equivalent to the minimum cost function). In the bottom plot, the fraction of solutions including the optimal solution as a function of the number of shots.



(a) (b)

Figure 4.9: QAOA time-of-convergence results for the graph-complexity study. On the left, number of iterations with the classical optimizer as a function of the number shots. On the right, the same quantity multiplied by the number of shots, corresponding to the total number of circuit evaluations.

with the real cost function, and the algorithm can start converging. At this point, adding measurements squeezes the distribution, allowing a better convergence, the improves as $1/\sqrt{\text{shots}}$.

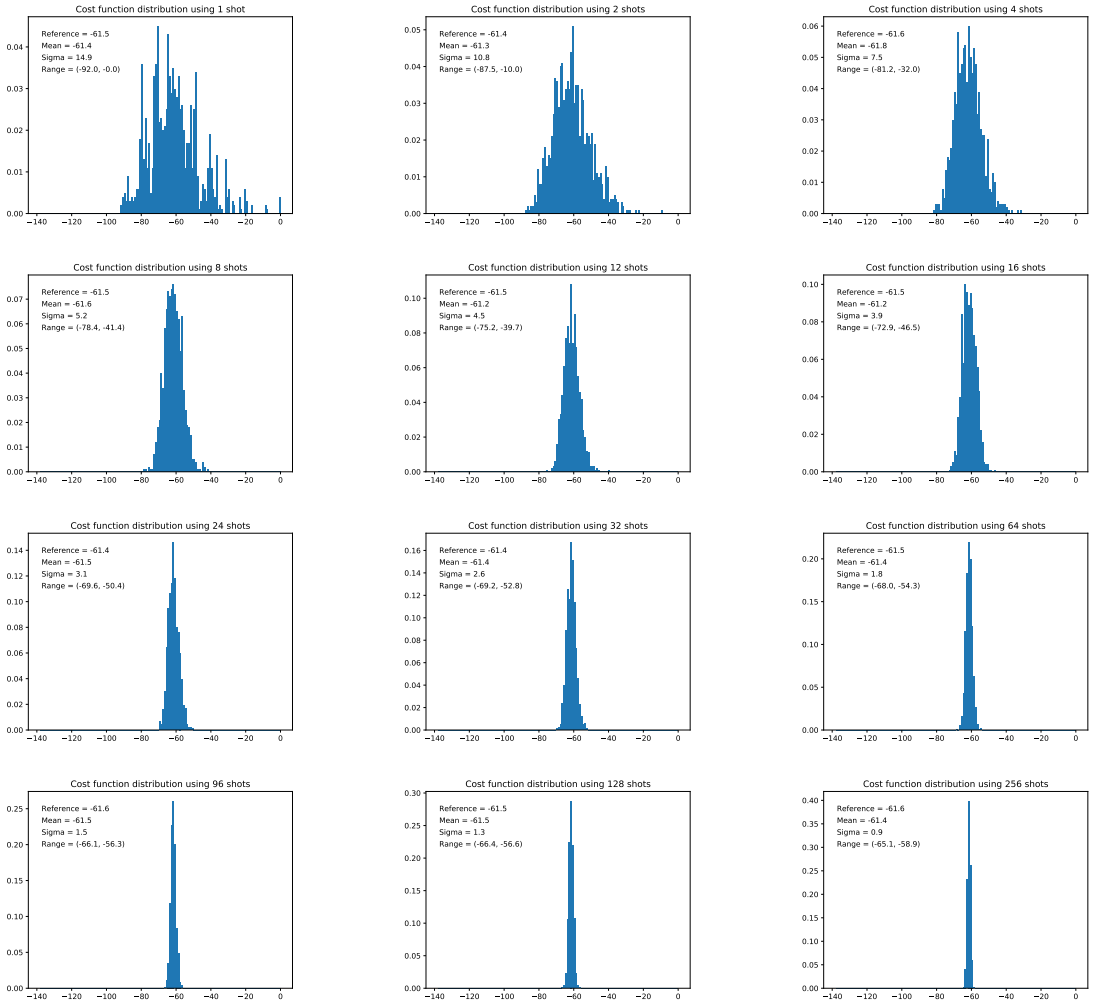


Figure 4.10: Results of measuring a VQE circuit using a given number of shots. Each plot shows the results distribution of 1000 repeated measurements. The number of shots of each measurements increases from top-left to bottom-right. The results displayed used the following number of shots: {1, 2, 4, 8, 12, 16, 24, 32, 64, 96, 128, 256}.

Chapter 5

Conclusions

We presented a study on the convergence of hybrid quantum-classical algorithms applied to the resolution of combinatorial optimization problems. We studied how the variational quantum eigensolver (VQE) and the quantum approximate optimization algorithm (QAOA) converge to the solution of a set of max-cut problems. We selected several max-cut problems associated with graphs of different size and complexity, and use low-depth circuits for their resolution, to cope with the non-negligible error probability of the currently-available qubits. For the cost function evaluation, we inspected a large set of shots values, observing how the choice affected the algorithm convergence. For very low number of shots (up to 24), the VQE algorithm returns random results, but once this threshold is overcome, the solution mean-cost-function gets closer to the optimal solution cost function as $1/\sqrt{\text{shots}}$. Despite the promising result, the fraction of solutions containing the eigenstate associated with the optimal solution does not always increase with the number of shots. We explain this observation by suggesting that the mean cost function is not the most appropriate quantity to minimize if the target is to get a larger fraction of *good* solutions. On the other hand, the QAOA algorithm does not show any sign of convergence, neither for low-depth circuits, nor deeper circuits.
Possible developments of this work include the resolution of additional optimization problems, the inspection of different variables, other than the mean cost function, for the optimizations process, and use real quantum computers, since currently, all the results presented were obtained from simulations.

Appendix A

Summary

Even if the quantum computing popularity has recently grown, the currently-available technology is not ready for a fundamental revolution of informatics. In this transition phase, hybrid quantum-classical algorithms can help in understanding the real power that this novel paradigm can reach in the future. In this work, we tested the capability of two such algorithms to solve combinatorial optimization problems: the variational quantum eigensolver (VQE) and the quantum approximate optimization algorithm (QAOA). We used them to solve max-cut problems, consisting of separating the vertices of an undirected graph in two sub-groups so that the number of edges connecting vertices of different sub-groups is maximal. To do that, we mapped the problem into a quantum system, using the Ising formalism. In particular, we investigated the effect of changing the number of measurements of the quantum circuit associated with the system on the algorithms convergence. A large number of measurements (or shots) allows a better knowledge of the quantum state, but on the other hand, increases the convergence time, making the algorithms less competitive compared to classical techniques. The results showed that VQE needs a minimum number of measurements to start converging towards the optimal problem solution. Once this threshold is overcome, the solution improves as $1/\sqrt{\text{shots}}$. QAOA, on the other hand, does not seem to converge, independently of the number of shots.

Appendix B

Resumen

Aunque la popularidad de la computación cuántica haya crecido recientemente, la tecnología actualmente disponible no es todavía suficiente para revolucionar profundamente la informática. En esta fase de transición, los algoritmos híbridos cuánticos-clásicos pueden ayudar a comprender el real poder que esta nueva técnica puede alcanzar en el futuro. En este trabajo, estudiamos la capacidad de dos de estos algoritmos para resolver problemas de optimización combinatoria: el *variational quantum eigensolver* (VQE) y el *quantum approximate optimization algorithm* (QAOA). Los usamos para resolver problemas de tipo *max-cut*, que consisten en separar los vértices de un grafo no dirigido en dos subgrupos de modo que el número de conexiones entre vértices de diferentes subgrupos sea máximo. Para hacer eso, mapeamos el problema a un sistema cuántico, utilizando el formalismo de Ising. En particular, investigamos como cambiar el número de medidas del circuito cuántico asociado al sistema afecta la convergencia de los algoritmos. Un gran número de medidas (o *shots*) permite un mejor conocimiento del estado cuántico, pero por otro lado aumenta el tiempo necesario para converger, haciendo que los algoritmos sean menos competitivos en comparación con las técnicas clásicas. Los resultados mostraron que VQE necesita un número mínimo de medidas para comenzar a converger hacia la solución óptima del problema. Una vez que se supere este umbral, la solución mejora como $1/\sqrt{\text{shots}}$. QAOA, por otro lado, no parece converger, independientemente del número de medidas.

Bibliography

[1] Arute, F., Arya, K., Babbush, R. et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574 (2019), pp. 505–510. DOI: https://doi.org/10.1038/s41586-019-1666-5.

[2] Edwin Pednault, John Gunnels, Dmitri Maslov, and Jay Gambetta. *On 'Quantum supremacy'*. https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/. 2019.

[3] Fred Glover, Gary Kochenberger, and Yu Du. *A Tutorial on Formulating and Using QUBO Models*. 2018. arXiv: 1811.11538 [cs.DS].

[4] L. W. McKeehan. "A Contribution to the Theory of Ferromagnetism". In: *Phys. Rev.* 26 (2 Aug. 1925), pp. 274–279. DOI: 10.1103/PhysRev.26.274. URL: https://link.aps.org/doi/10.1103/PhysRev.26.274.

[5] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5.1 (June 2014). ISSN: 2041-1723. DOI: 10.1038/ncomms5213. URL: http://dx.doi.org/10.1038/ncomms5213.

[6] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv: 1411.4028 [quant-ph].

[7] Panagiotis Kl. Barkoutsos et al. "Improving Variational Quantum Optimization using CVaR". In: *Quantum* 4 (Apr. 2020), p. 256. ISSN: 2521-327X. DOI: 10.22331/q-2020-04-20-256. URL: https://doi.org/10.22331/q-2020-04-20-256.

[8] Héctor Abraham et al. *Qiskit: An open-source framework for quantum computing*. https://qiskit.org/. 2019.

[9] Feynman, R.P. "Simulating physics with computers". In: *Int J Theor Phys* 21 (1982), pp. 467–488. DOI: 10.1007/BF02650179. URL: https://doi.org/10.1007/BF02650179.

[10] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331/q-2018-08-06-79.

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

[12] C. J. Ballance et al. “High-Fidelity Quantum Logic Gates Using Trapped-Ion Hyperfine Qubits”. In: *Physical Review Letters* 117.6 (Aug. 2016). ISSN: 1079-7114. DOI: 10.1103/physrevlett.117.060504. URL: http://dx.doi.org/10.1103/PhysRevLett.117.060504.

[13] R. Barends et al. “Superconducting quantum circuits at the surface code threshold for fault tolerance”. In: *Nature* 508.7497 (Apr. 2014), pp. 500–503. ISSN: 1476-4687. DOI: 10.1038/nature13171. URL: http://dx.doi.org/10.1038/nature13171.

[14] Powell M.J.D. “A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation.” In: *Gomez S., Hennart JP. (eds) Advances in Optimization and Numerical Analysis. Mathematics and Its Applications* (1994). DOI: 10.1007/978-94-015-8330-5_4. URL: https://doi.org/10.1007/978-94-015-8330-5_4.

[15] Edward Farhi et al. *Quantum Computation by Adiabatic Evolution*. 2000. arXiv: quant-ph/0001106 [quant-ph].

[16] Ryan Hoque. https://ryanhoque.github.io/data/QAOA.pdf. May 2019.

[17] M. Born and V. Fock. “Beweis des Adiabatensatzes”. In: *Z. Physik* 51 (1928), pp. 165–180. DOI: 10.1007/BF01343193. URL: https://doi.org/10.1007/BF01343193.

[18] Yin Sun et al. *Adiabatic Quantum Simulation Using Trotterization*. 2018. arXiv: 1805.11568 [quant-ph].