POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Reinforcement Learning for Variational Quantum Circuits design: a PPO-based approach for solving QUBO problems

Tesi di Laurea Magistrale in
High Performance Computing Engineering

Simone Foderà, 224338

Abstract: Variational Quantum Algorithms (VQAs) [1] have emerged as promising tools for solving complex problems on quantum computers. They iteratively optimize quantum circuit parameters to lower the expectation value of the quantum system energy. One of the challenges of these algorithms is understanding how to identify a suitable circuit for solving a specific problem. In this thesis, we design and implement a Reinforcement Learning agent capable of autonomously generating variational quantum circuits. The Reinforcement Learning framework allows more adaptability, potentially discovering new, unconventional circuit structures that may outperform existing designs. The agent is trained on different instances, learning to build circuits that solve the Maximum Cut, the Maximum Clique and the Minimum Vertex Cover problems on different graph topologies with 8 and 14 vertices. We analyze the circuit gate usage, depth and performance. In particular, by examining the common structure of the circuits identified by the agent trained on the Maximum Cut problem, we were able to infer a novel family of ansatzes that achieve a high approximation ratio. We denote this family as the R_{yz}-connected circuits. We select a specific R_{yz}-connected element, denoted as Linear Circuit, and test its performance on problem instances with graphs with up to 16 vertices. The results are compared with state-of-the-art quantum algorithms such as QAOA and its variants and classical algorithms. We show that the Linear Circuit consistently achieves a lower energy expected value than the other algorithms. The use of entanglement and the effectiveness of the R_{yz}-connected ansatzes can be explained by considering the parametric expression of the expectation value of the Hamiltonian and how the solution space is explored. Furthermore, we show the straightforward hardware implementability of the R_{yz}-connected ansatzes. Overall, we show that Reinforcement Learning is a promising avenue for building ansatzes aimed at addressing a broad set of problems that are hard to be solved with classical computing.

Advisor:
Prof. Dacrema Ferrari Maurizio

Co-advisors:
Gloria Turati

Academic year:
2023-2024

Key-words: Variational Quantum Algorithms, Quantum Computing, Reinforcement Learning, Optimization

1. Introduction

Quantum computing has gained attention for its potential for achieving solutions to problems that are challenging to be solved with classical computing.
Variational Quantum Algorithms (VQAs) [1], which leverage a parametric quantum circuit and a classical optimizer to tune the circuit parameters, are a promising paradigm in the Noisy Intermediate-Scale Quantum (NISQ) [2] era. However, one of the challenges of these algorithms is understanding how to identify a suitable ansatz for solving the specific problem being addressed. Among the methods adopted for generating ansatzes, some exploit the proprieties of the problems [3], others consider the specifics of the hardware [4], and others use an adaptive approach [5].
In this thesis, we explore the implementation of a Reinforcement Learning agent trained for building parametric quantum circuits to solve specific mathematical problems that cannot be solved efficiently on classical computers. We first delve into the mathematical and physical aspects of quantum computing and quantum devices. Then, we explain the combinatorial optimization problems and present a range of classical and quantum algorithms. Subsequently, we illustrate the basic aspects of Reinforcement Learning and describe our implementation of an agent trained to solve specific QUBO problems built from different graph topologies. We analyze the circuits built by the agent, and, by examining the common structure of the circuits built for solving the Maximum Cut problem, we infer a family of ansatzes, denoted as R_{yz}-connected family. In particular, we consider a specific R_{yz}-connected ansatz, the Linear Circuit, and test its performance on different graph topologies and different problem typologies. Finally, we investigate the implementability of the R_{yz}-connected ansatzes on real quantum devices.

2. Background

In this section, we cover the background knowledge needed to understand our thesis. We explain quantum computing and quantum devices, followed by combinatorial optimization problems and some classical algorithms used to solve these problems.

2.1. Quantum computing

In this subsection, we cover the basis of quantum computing, starting from the mathematical postulates up to the formulation of quantum circuits and quantum gates.

2.1.1. Mathematical postulates

Quantum mechanics can be approached within a mathematical framework based on four postulates. These postulates construct a bridge from the physical world to the mathematical formalism.

1. Each quantum system can be associated with an element of a finite-dimensional Hilbert space $\mathcal{H}$. Such a space is equivalent to a vector space over the complex numbers with a scalar product $\langle \psi, \phi \rangle$, which is complete with respect to the norm. Therefore, we can represent our quantum state with a vector called statevector. Using the bra-ket notation, the ket $|\psi\rangle \in \mathcal{H}$ denotes a column vector. The bra $\langle \phi |$ is a linear form $\phi : \mathcal{H} \to \mathbb{C}$, such that $\langle \phi | \psi \rangle = \langle \phi, \psi \rangle$.
2. The evolution of a quantum system can be described by a Unitary operator. An operator $U : \mathcal{H} \to \mathcal{H}$ is unitary if $UU^\dagger = I$. This implies that the operator is invertible (the inverse is equal to the adjoint operator) and that evolutions in quantum mechanics must be reversible.
3. Quantum measurements are associated with observables. An operator is an observable if it is Hermitian. An operator is Hermitian if $U = U^\dagger$. From the spectral theorem, we know that Hermitian matrices have real eigenvalues and orthonormal eigenvectors. During the measurement, the quantum state collapses in one of the eigenvectors and the result is the eigenvalue of such eigenvector. Given the state $|\psi\rangle$, the probability of collapsing in state $|\psi_i\rangle$ is $|\langle \psi | \psi_i \rangle|^2$.
4. The Hilbert space $\mathcal{H}$ that describes a composite system is the tensor product of the Hilbert spaces $\mathcal{H}_i$ that describe the single components.

$$\mathcal{H} = \bigotimes_i \mathcal{H}_i$$

2.1.2. Qubit and its representation

Unlike classical computers, which use bits that can assume values 0 or 1, quantum computers use quantum bits or qubits. A single qubit can be in a superposition of the states $|0\rangle$ and $|1\rangle$. Therefore, it can be associated with an element of a 2-dimensional Hilbert space. Using as basis states $|0\rangle$ and $|1\rangle$, we can represent a qubit in a superposition state as a linear combination of the basis states

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

being α and β complex numbers (see postulate 1) called amplitudes that satisfy

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

Qubits are thus unit vectors of a Hilbert space. Moreover, it is physically impossible to distinguish $|\psi\rangle$ from $e^{i\theta}|\psi\rangle$. Consequently, we consider two systems that only differ from a phase as mathematically equivalent. The phase θ is called the global phase factor.
In quantum computing, quantum states are measured using $|0\rangle$ and $|1\rangle$ as the measurement basis. The measurement causes the state to assume either state $|\psi\rangle = |0\rangle$ or $|\psi\rangle = |1\rangle$. This phenomenon is called collapse onto one of the basis states. From postulate 3, we get that the probability of measuring $|0\rangle$ or $|1\rangle$ is given by

$$P(|0\rangle) = |\langle\psi|0\rangle|^2 = |\alpha|^2$$
$$P(|1\rangle) = |\langle\psi|1\rangle|^2 = |\beta|^2$$

Therefore, α and β are related to the probability of measuring the basis states.
Therefore, a computer with N bits can represent 2^N different values. On the other hand, a quantum computer with N qubits can be measured in 2^N states, but before the measurement, the qubits exist in multiple states simultaneously.
A common and helpful visualization of the qubit is the Bloch Sphere. We can easily prove that any qubit can be mapped on the surface of a sphere. To do so, we decompose α and β in their radial coordinates

$$|\psi\rangle = |\alpha|e^{i\phi_\alpha}|0\rangle + |\beta|e^{i\phi_\beta}|1\rangle$$

Given the global phase invariance of a single qubit, we can multiply our state for a global phase $e^{-\phi_\alpha}$

$$|\psi\rangle = |\alpha||0\rangle + |\beta|e^{i(\phi_\beta - \phi_\alpha)}|1\rangle$$

Knowing (2), we can rewrite $|\alpha|$ and $|\beta|$ as cosine and sine of an arbitrary angle,

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\phi}|1\rangle \tag{3}$$

With this notation, we can interpret ϕ and θ as the azimuthal and the elevation angles of a point on a sphere of radius 1.
The Bloch Sphere is fundamental because each evolution is unitary, and all the unitary operators can be described as rotations on the Bloch Sphere. Representing a quantum state as a point on a sphere makes the visualization and analysis of their evolutions clearer.
Geometrically, on the Bloch Sphere, the opposite of a state in the form of Equation (3) is given by

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

If we compute the scalar vector of an element with its opposite, we obtain

$$\begin{aligned}\langle\psi|\psi'\rangle &= \cos\left(\frac{\theta}{2}\right)\cdot\cos\left(\frac{\pi}{2}-\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\cdot\sin\left(\frac{\pi}{2}-\frac{\theta}{2}\right)e^{-i\phi}e^{i(\phi+\pi)} = \\ &= \cos\left(\frac{\theta}{2}\right)\cdot\sin\left(\frac{\theta}{2}\right) - \cos\left(\frac{\theta}{2}\right)\cdot\sin\left(\frac{\theta}{2}\right) = 0\end{aligned}$$

Therefore, two vectors are orthogonal when they are opposite.
The three most important pairs of orthogonal states are

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

Which can be seen in the Bloch Sphere in Figure 1[1]

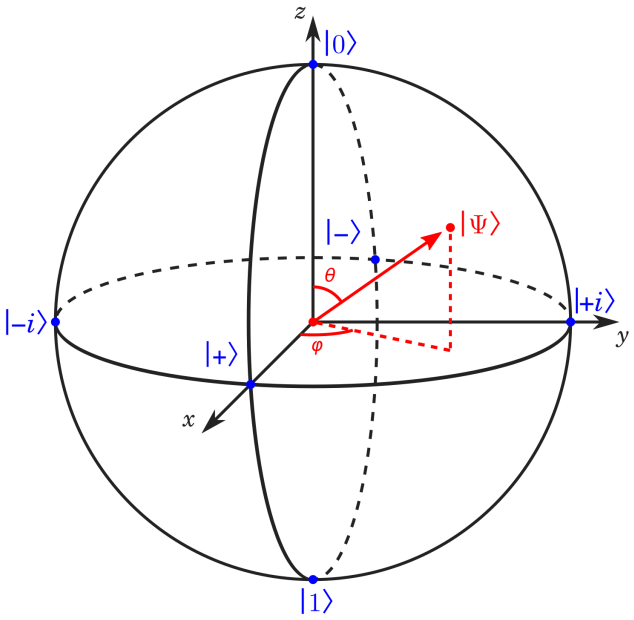


Figure 1: Visualization of the Bloch Sphere. $|\psi\rangle$ can be completely described by θ and ϕ.

As stated in postulate 1, we can represent those ket as complex-valued vectors. If we use as basis states $|0\rangle$ and $|1\rangle$, then we obtain

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \qquad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$
$$|+\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$$
$$|+i\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\i\end{bmatrix} \qquad |-i\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-i\end{bmatrix}$$

2.1.3. Multi-qubit states

Given two independent systems $|\psi_1\rangle \in \mathcal{H}_1$ of dimension n and $|\psi_2\rangle \in \mathcal{H}_2$ of dimension m, the system composed by these two qubits can be represented by an element of the Hilbert space $\mathcal{H}$, tensor product of the two subspaces

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$$

of dimension $n \cdot m$.
The element $|\psi\rangle \in \mathcal{H}$ can be expressed as

$$|\psi\rangle = |\psi_1\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \tag{4}$$

And its amplitudes are relative to the basis of $\mathcal{H}$. The basis can be obtained by the tensor product of the basis vectors of $\mathcal{H}_1$ and $\mathcal{H}_2$.
Let $\{|\phi_{1i}\rangle\}_{i=1,\dots,n}$ and $\{|\phi_{2i}\rangle\}_{i=1,\dots,m}$ be the basis states of the two subsystems.
The basis set of the composite system is defined as

$$\{|\phi_{1i}\rangle \otimes |\phi_{2j}\rangle\}_{i=1,\dots,n,\, j=1,\dots m}$$

[1]Figure at `https://prefetch.eu/know/concept/bloch-sphere/`

2.1.4. Entanglement

If an element of $\mathcal{H}$ cannot be factorised in the tensor product of two elements of the subsystems as in Equation (4), it is said to be in an entangled state.
Entanglement is a core quantum mechanical property: it allows qubits to be correlated so that one qubit state directly influences another state.
The most recurrent entangled states are the Bell states. They are four two-qubit entangled states.

$$\begin{aligned} |\phi^+\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ |\phi^-\rangle &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \end{aligned} \tag{5}$$

$$\begin{aligned} |\psi^+\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\ |\psi^-\rangle &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \end{aligned} \tag{6}$$

If we measure only one qubit, we also know the result of the measurement of the other qubit. In the states described by Equations 5, the two entangled qubits are said to be positively correlated since they give the same outcome when measured. On the contrary, the states in Equations 6 are said to be negatively correlated since the measure of the two qubits gives opposite outcomes when measured.
A classical computer can efficiently simulate any quantum algorithm which does not employ entanglement [6]. On the other hand, simulating the behaviour of entangled quantum systems becomes exponentially complex as the number of entangled qubits increases. The employment of entanglement during quantum computations is crucial in order to fully exploit the quantum properties and achieve an advantage with respect to classical computers.

2.1.5. Quantum Circuits

To graphically represent the evolution of a quantum state, we apply quantum operators through quantum gates in quantum circuits. This circuital approach is analogous to the classical logic circuit, in which gates operate on bits. In quantum circuits, gates act on quantum states, enabling the creation of complex quantum algorithms. The gates can be represented as matrices with complex values.

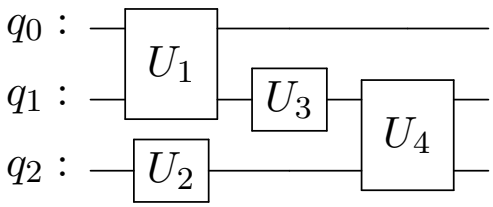


Figure 2: Graphical representation of a quantum circuit. U_1, U_2, U_3, U_4 are unitary operators which describe the evolution of the qubits q_0, q_1, q_2.

Figure 2 represents a circuit of three qubits to which four unitary gates are applied. A layer of a quantum circuit is a set of gates which can be applied in parallel during the computation. In this example, U_1 and U_2 are in the first layer, U_3 in the second and U_4 in the third. The depth of a circuit is equal to the number of layers. The generic final state of an n qubits circuit is a superposition of 2^n basis states. Those basis states are expressed as $|b_{n-1} \dots b_0\rangle$, with b_i the binary value of the i^{th} qubit.
The most commonly used gates are the Hadamard gate, the Pauli gates and the Control Not gate.

2.1.6. Single-qubit gates

The Hadamard gate is defined as

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

and its primary purpose is to introduce the superposition of states, since

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

For this reason, a quantum circuit first layer is often composed of Hadamard gates, initializing the system to the equally weighted superposition of all states.

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle$$

The Pauli gates are defined as

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$
$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

To understand their behaviour, we can study their eigenvectors.

$$\sigma_x|+\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = |+\rangle$$
$$\sigma_x|-\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -|-\rangle$$

$$\sigma_y|+i\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ i \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ i \end{bmatrix} = |+i\rangle$$
$$\sigma_y|-i\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -i \end{bmatrix} = -\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -i \end{bmatrix} = -|-i\rangle$$

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

All the eigenvalues are ± 1. A 2 dimensional operator with a $+1$ and a -1 as eigenvalues is effectively a reflection. However, a reflection on the 2 dimensional plane can be mapped to a rotation in the Bloch Sphere. Furthermore, it can be shown that

$$\sigma_a \sigma_a = I, \ a \in \{x, y, z\}$$

Therefore, each Pauli gate represents the rotation of π around its respective axis.
Single-qubit rotations of angle θ around the x, y, and z axis are expressed by

$$R_a(\theta) = e^{-i\frac{\theta}{2}\sigma_a}, \quad a \in \{x, y, z\} \tag{7}$$

We use as notation σ_x^j to indicate "σ_x applied to qubit j",with $j = 0, .., n-1$, which can be mathematically expressed as

$$\sigma_x^j = I^{\otimes j} \otimes \sigma_x \otimes I^{\otimes (n-j-1)} \tag{8}$$

2.1.7. Double-qubit gates

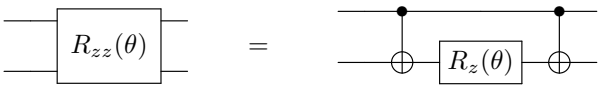
The $CNOT$ is defined as the operator that when is applied to the state $|b_0 b_1\rangle$ uses σ_x on the qubit $|b_1\rangle$ only if $b_0 = 1$. It maps the state $|10\rangle$ into $|11\rangle$ and vice versa while keeping the states $|00\rangle$ and $|01\rangle$ untouched. Its matrix representation is given by

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

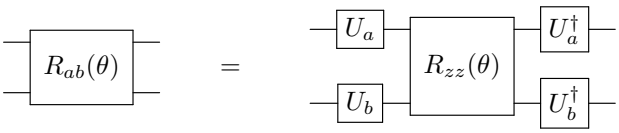
The double rotations are defined as

$$R_{ab}(\theta) = e^{-i\frac{\theta}{2}\sigma_a \otimes \sigma_b}, a, b \in \{x, y, z\}$$

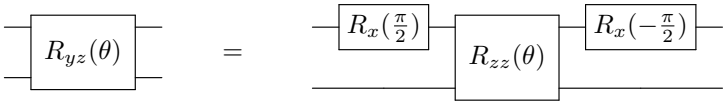
The equivalent circuit for the double rotation $R_{zz}(\theta)$ is



And the $R_{ab}(\theta)$ rotations are obtained from $R_{zz}(\theta)$ in the following way



Where U_a and U_b are gates which map the a and b axis into the z axis, i.e. $U_x = H$ and $U_y = R_x(\frac{\pi}{2})$ and $U_z = I$. Those double rotations are used to generate entanglement between qubits.
In particular, in Section 7 we will use



(9)

2.2. Quantum devices

In this subsection, we explain how common quantum devices are implemented to see how the quantum gates are physically realized.

2.2.1. Qubit

A qubit is a mathematical representation of an object with quantum properties with only two discrete values that can be assumed when measured. For example, the most straightforward physical example of the qubit is the spin. Indeed, spin has quantum properties, and when measured, it can only assume two values: spin up or spin down. Another example of the qubit is the anharmonic oscillator. In this case, the quantum information is the system energy. We can create a two-valued system with this, even if its energy can be measured in infinite discrete states. This is done by initializing the system in the ground state and constraining the system evolution only towards the ground state and the first excited state. Regardless of the physical interpretation, a qubit can always be described by Equation (1) and, therefore, also by Equation (3).

2.2.2. Evolution of the system

We have seen in Subsection 2.1.1 that a unitary operator can represent the system evolution. In particular, the system evolution is regulated by the time-dependent Schrödinger equation

$$\hat{H}|\psi\rangle = i\hbar\frac{\partial|\psi\rangle}{\partial t}$$

Solving this equation, we obtain that the evolution of the system is given by

$$|\psi(t)\rangle = e^{-i\frac{\hat{H}}{\hbar}t}|\psi(0)\rangle$$

Therefore the unitary that represents the evolution of the system is

$$U = e^{-i\frac{\hat{H}}{\hbar}t} \tag{10}$$

2.2.3. Rotations implementation

In the superconductive qubit, spin qubit and trapped ion qubit, the Hamiltonian is composed of two elements. The first represents a rotation around the Z axis, and the second is around a specific axis in the XY plane.

$$\hat{H} = \frac{\hbar}{2}\omega_{01}\sigma_z + \frac{\hbar}{2}\Omega(t)\sigma_x(\cos(\omega_D t - \phi))$$

We can note that a Hamiltonian in the form $\hat{H} = \frac{\hbar}{2}\omega\sigma_a$, substituted in Equation 10 leads to $U = e^{-i\frac{\omega t}{2}\sigma_a}$, which is exactly $R_a(\omega t)$ as defined in Equation (7). Therefore, ω_{01} and $\Omega(t)$ represent the frequencies of the respective rotations. If we set $\Omega(t) = 0$, the evolution of a system subject to this Hamiltonian only rotates around the Z axis at frequency ω_{01}, which is the frequency difference between the two energy states defined as

$$\omega_{01} = \frac{E_{|1\rangle} - E_{|0\rangle}}{\hbar}$$

where $E_{|1\rangle}$ is the energy of the state $|1\rangle$ and $E_{|0\rangle}$ is the energy of the state $|0\rangle$, assuming $E_{|1\rangle} > E_{|0\rangle}$
If we activate the drive field (i.e. $\Omega(t) \neq 0$) at a frequency $\omega_D = \omega_{01}$ one can mathematically prove that the evolution of the state is the sum of a rotation at frequency ω_{01} around the Z axis and at frequency $\Omega(t)$ around the axis on the XY plane obtained by rotating X by ϕ counterclockwise. If $\phi = 0$, we perform a rotation around the X axis, if $\phi = \frac{\pi}{2}$, we perform a rotation around the Y axis.
To perform rotations around Z efficiently, we exploit the form of $\hat{H}$ and the fact that in quantum physics, global phases do not influence the quantum states. Instead of performing a physical rotation around the Z axis of θ followed by one around the axis defined by ϕ, one can perform a single rotation around the axis obtained rotating the X axis by $\phi + \theta$ counterclockwise. Therefore, a $R_z(\theta)$ rotation can be implemented with negligible error and time [7] since the phase ϕ is often stored in a classical register and adding θ to it is trivial.
On the contrary, implementing a parametric $R_x(\lambda)$ rotation requires more discretion: the rotation must be applied for a specific time. The time depends on $\Omega(t)$ and λ. Modern technologies are usually calibrated to perform only $R_x(\frac{\pi}{2})$ rotations. This is because any arbitrary rotation can be performed by combining R_z and $R_x(\frac{\pi}{2})$ rotations.

2.2.4. Hardware connectivity topology

We can analyze one of IBM most advanced publicly available quantum processors with 127 qubits. The single-qubit gates it can use are only R_z, $\sqrt{\sigma_x}$ (i.e. $R_x(\frac{\pi}{2})$), and σ_x (i.e. $R_x(\pi)$).

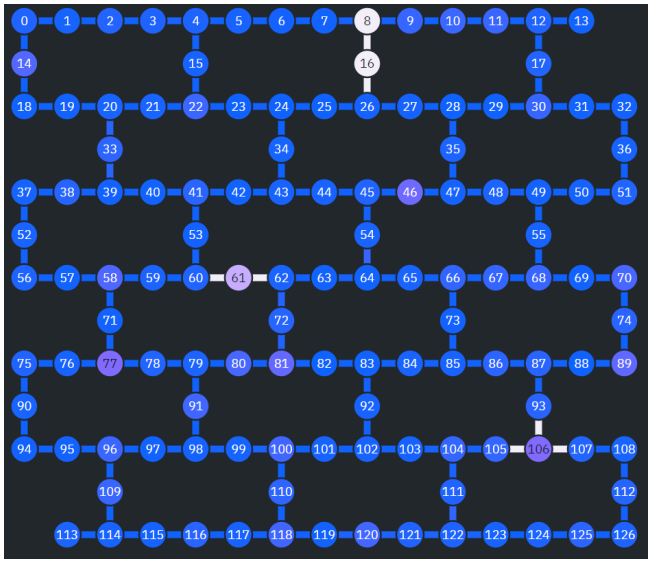


Figure 3: Graph of the IBM Osaka Quantum Processor qubit connectivity.

In Figure 3, we can observe the graph representing the physical qubit connections of the quantum processor. This is an example of a quantum computer based on superconductive qubits. In these devices, the qubits cannot be physically moved around freely, and to implement a double-qubit gate, the two qubits need to be physically close. For this reason, the graph describing the connection of the qubits is sparse. The sparseness is one of the problems of quantum computing. Performing a logical two-qubit gate on two distant qubits is challenging. You

need to swap the qubits two by two until the two qubits of interest are close. This method leads to an increase in the depth of the circuit and an increase in the errors due to noise and decoherence of the qubits.

2.3. Combinatorial optimization problems

In this subsection, we explain the combinatorial optimization problems. These problems are often used to benchmark the performance of algorithms, since they are challenging to be solved exactly because of their inherent complexity and the combinatorial nature of their solution spaces.
These problems fall into the category of NP-hard problems, meaning that no known algorithm can solve them in polynomial time in the input size. The difficulty arises from the exponential growth of the number of possible solutions as the size of the input increases. For example, in the Maximum Cut problem, the number of possible cuts grows exponentially with the number of vertices, making an exhaustive search impractical for large graphs. The lack of efficient algorithms for solving these problems leads to the development of approximation algorithms and heuristics. These methods sacrifice the guarantee of finding the optimal solution in favour of obtaining reasonably good solutions within a reasonable amount of time.
Furthermore, the potential advantage of quantum computers in handling certain types of computations exponentially faster than classical computers leads to the development of quantum algorithms for tackling combinatorial optimization problems. Quantum algorithms leverage principles of quantum mechanics to explore multiple solutions simultaneously, providing a potential speedup for solving problems with large solution spaces.
We will now have a look at some combinatorial optimization problems and a possible way to obtain their quantum formulation.

2.3.1. Maximum Cut problem

Given an undirected graph

$$G = (V, E)$$

where V is the set of n vertices and E is the set of edges, the Maximum Cut problem aims to find a partition $(S, \bar{S})$ (where $S \cup \bar{S} = V$ and $S \cap \bar{S} = \emptyset$) of the graph vertices into two sets such that the number of edges crossing the partition is maximized.
Formally, we can use n binary variables where $x_i = 0$ if $x_i \in S$ and $x_i = 1$ if $x_i \in \bar{S}$ for all the $i = 1, ..., n$.
Given an edge (i, j), we can represent the information that the edge is in the cut by the formula $x_i + x_j - 2x_i x_j$. Indeed, when $x_i + x_j - 2x_i x_j = 1$, then only one of x_i or x_j is equal to 1, which implies that the edge (i, j) is in the cut. When $x_i + x_j - 2x_i x_j = 0$, it means that x_i and x_j are both equal to either 0 or 1, implying that they are within the same set.
Therefore, the Maximum Cut problem can be modeled as

$$\max \left\{ \sum_{(i,j)\in E} (x_i + x_j - 2x_i x_j) \right\}$$

2.3.2. Maximum Clique problem

Given an undirected graph G, the Maximum Clique problem aims to find the biggest clique within the given graph. A clique is a subset of fully connected vertices. More precisely, a subset $S \subseteq V$ is considered a clique when the subgraph G' of G obtained using only the vertices in S, and the edges $E' = \{(i, j) \in E |\ i \in S,\ j \in S\}$ is a fully connected graph. Using n binary variables, we can model the problem by assigning $x_i = 1$ if $x_i \in S$, and $x_i = 0$ if $x_i \notin S$. Then the Maximum Clique problem can be easily formulated as a constrained problem. A solution that respects the constraints is denoted as a feasible solution. Otherwise, it is denoted as an unfeasible solution.
The mathematical formulation is

$$\max \left\{ \sum_{i=1}^{n} x_i \right\}$$

subject to

$$x_i + x_j < 1 \ \forall (i, j) \notin E$$

The constraint $x_i + x_j < 1$ excludes the case that these two nodes are both equal to 1 (i.e. cannot be both in the chosen subgraph) if they are not connected. This constraint ensures that only fully connected graphs are considered feasible solutions.

2.3.3. Minimum Vertex Cover problem

Given an undirected graph G, the Minimum Vertex Cover problem aims to find the smallest subset $S \subseteq V$ such that all the edges have one of the two ends within the subset. Using the same variable notation of the Maximum Clique, an edge (i, j) is said to be covered if either $x_i = 1$ or $x_j = 1$. Thus, we can model the problem as

$$\begin{aligned} &\min \left\{ \sum_{i=1}^{n} x_i \right\} \\ &\text{subject to} \\ &x_i + x_j \geq 1 \ \forall (i,j) \in E \end{aligned} \tag{11}$$

2.3.4. QUBO formulation

To make the problem suitable for being solved by a quantum computer, we need to formulate the problem as a Quadratic Unconstrained Binary Optimization (QUBO) problem, in particular, a minimization problem [8].
To convert a maximization problem into a minimization problem, we invert the sign of the objective function. Therefore, the Maximum Cut problem becomes

$$\min \left\{ \sum_{(i,j)\in E} (-x_i - x_j + 2x_i x_j) \right\}$$

To convert a constrained problem into an unconstrained one, we can employ penalties on the objective function to penalize unfeasible solutions. Using this method, we formulate the problem using weak constraints. A weak constraint does not ensure that we will obtain a feasible solution, however we take the constraints into account by adding a penalty when a constraint is violated. We set the penalties to be big enough that any unfeasible solution has a cost higher than any feasible one. Setting the penalties in this way makes all the unfeasable solutions worse than the feasible one, making them less likely to be the output of any algorithm. For both the Maximum Clique and Minimum Vertex Cover, a penalty $p = n + 1$ works for this purpose. We can model the Maximum Clique problem in QUBO formulation as

$$\min \left\{ \sum_{i=1}^{n} -x_i + \sum_{(i,j)\notin E} p x_i x_j \right\}$$

In the Maximum Clique, it is immediate to see that the worst feasible solution is the one with all the $x_i = 0$, which has a cost of 0. Since the minimum value that the cost function can theoretically achieve is $-n$, adding a penalty p cost whenever the constraint is violated ensures that the costs of unfeasible solutions are higher than the ones of the feasible solutions.
The penalty in the Minimum Vertex Cover is more complex, and the objective function becomes

$$\min \left\{ \sum_{i=1}^{n} x_i + \sum_{(i,j)\in E} p(1 - x_i - x_j + x_i x_j) \right\}$$

We can observe that $(1 - x_i - x_j + x_i x_j)$ equals 1 and adds the penalty to the objective function only when neither x_i nor x_j equals 1. This reflects exactly the constraint stated in Equation (11).

2.3.5. Ising Hamiltonian

We now explain how to convert problems in QUBO formulation to a Hamiltonian, which can be used to solve the mathematical problem.
We can see that all the QUBO formulations can be written in the following form

$$\min \left\{ x^T Q x + c^T x \right\} \tag{12}$$

where Q is a symmetric square matrix with zeros on the diagonal representing the coefficients of the quadratic terms of the cost function, and c is a vector representing the coefficients of the linear terms. We can ignore all the constant terms because they are irrelevant to the optimization process.
This is useful because we can now define a Hamiltonian $\hat{H}$ based on the above QUBO formulation that satisfies

$$\text{argmin}\left\{x^T Q x + c^T x\right\} = \text{argmin}\left\{\langle x|\hat{H}|x\rangle\right\} \tag{13}$$

often $\langle x|\hat{H}|x\rangle$ is just referred as $\langle\hat{H}\rangle$
In quantum physics, the energies that can be measured in a system are the eigenvalue of the Hamiltonian of that system. Thus, if we define a Hamiltonian which respects

$$\hat{H}|x\rangle = (x^T Q x + c^T x)|x\rangle \tag{14}$$

we can see that Equation (13) holds. Being able to calculate the ket $|x\rangle$ for which $\langle\hat{H}\rangle$ is minimized is equivalent to find the binary number x which minimizes Equation (12).
To create a Hamiltonian which satisfies Equation (14), we need to substitute each x_i with $\frac{I-\sigma_z^i}{2}$. To understand that the substitution is adequate, we consider a trivial case where $f(x) = x$. In this case, our Hamiltonian is $\hat{H} = \frac{I-\sigma_z}{2}$

$$\begin{aligned}\frac{I-\sigma_z}{2}|0\rangle &= \frac{I}{2}|0\rangle - \frac{\sigma_z}{2}|0\rangle \\ &= \frac{1}{2}|0\rangle - \frac{1}{2}|0\rangle \\ &= 0|0\rangle\end{aligned}$$

and

$$\begin{aligned}\frac{I-\sigma_z}{2}|1\rangle &= \frac{I}{2}|1\rangle - \frac{\sigma_z}{2}|1\rangle \\ &= \frac{1}{2}|1\rangle + \frac{1}{2}|1\rangle \\ &= 1|1\rangle\end{aligned}$$

which is exactly what we wanted to accomplish.
It can be proved that the substitution is correct also in the general case.
With these considerations and using σ_z as in Equation 8, we can now express our QUBO problems as Hamiltonians.
The Maximum Cut Ising Hamiltonian is

$$\hat{H} = \sum_{(i,j)\in E} \frac{1}{2}(\sigma_z^i \sigma_z^j - I)$$

The Maximum Clique Ising Hamiltonian is

$$\hat{H} = \sum_{i=1}^{n} \frac{1}{2}(\sigma_z^i - I) + \sum_{(i,j)\notin E} \frac{p}{4}(I - \sigma_z^i - \sigma_z^j + \sigma_z^i \sigma_z^j)$$

The Minimum Vertex Cover Ising Hamiltonian is

$$\hat{H} = \sum_{i=1}^{n} \frac{1}{2}(I - \sigma_z^i) + \sum_{(i,j)\in E} \frac{p}{4}(I + \sigma_z^i + \sigma_z^j + \sigma_z^i \sigma_z^j)$$

2.4. Classical Algorithms

In this subsection, we explain some classical algorithms that we will use to compare the performance of quantum algorithms with classical ones.
Each of these algorithms uses the notion of neighbourhood. Given a distance function $d(x, y)$, the ε-neighbourhood of a solution x is the set of solutions with distance at most ε from x.

$$N_\varepsilon(x) = \{y|\ d(x, y) < \varepsilon\}$$

The neighbourhood is used to have a representation of closeness and allows the algorithms to explore and potentially move to nearby solutions in the search space. The neighbourhood size and effectiveness depend from implementation to implementation.

2.4.1. Steepest Descent

Steepest Descent is the discrete analogue of gradient descent. Here, instead of calculating the gradient of the objective function and moving towards the direction of the gradient, we move discretely within the neighbourhood of the current solution.
At each step, each element of the neighbourhood of the current solution is computed, and the best neighbour is chosen for the next step. If none of the neighbours is better than the current one, the algorithm ends, and the current solution is returned.
Steepest Descent is fast, however it can get easily stuck in local minima.

2.4.2. Simulated Annealing

Simulated Annealing is a probabilistic optimization algorithm inspired by the annealing process in metallurgy. It is used to find approximate solutions to optimization problems. The algorithm mimics the annealing process where a material is heated and then gradually cooled to achieve a low-energy state.
In Simulated Annealing, the objective is to find the solution with the minimum energy within a set of solutions. The algorithm explores the solution space by iteratively moving from one solution to another, choosing the next solution within the neighbourhood of the current one. The algorithm accepts solutions that lead to a lower energy and occasionally accepts solutions that lead to a higher energy. This randomness allows the algorithm to escape from local minima and explore a broader solution space.
The probability of accepting solutions with higher energy is controlled by the temperature parameter. Initially, the temperature is set high, allowing the algorithm to explore a wider space. Then, it gradually decreases over time, making it less likely to accept a solution with higher energy. This cooling schedule is crucial for balancing exploration and exploitation.
The probability of accepting a new solutions is given by

$$P = \begin{cases} 1, & \text{if } E_n < E_c \\ e^{-\frac{E_n - E_c}{k \cdot T}}, & \text{otherwise} \end{cases}$$

with E_c being the energy of the current solution, E_n being the energy of the new solution, k being the Boltzmann constant, and T is the variable representing the temperature. T decreases at each iteration, and its cooling schedule is a hyperparameter of the algorithm.

2.4.3. Tabu Search

Tabu Search represents an evolution of Steepest Descent. The drawback of the latter method is that the search stops if there are no better solutions than the current one in its neighbourhood. However, the optimal solution identified by this method is only a local minimum, which is often far, especially in terms of quality, from the global optimal solution.
Differently from Steepest Descent, Tabu Search algorithm can continue the search beyond local minima. First, to escape the trap of local minima, Tabu Search can choose worse solutions and, to avoid falling back into the local minimum soon after, is able to make the recently executed moves in the search path forbidden. Therefore, a characteristic of Tabu Search is the extensive use of memory: to enhance the effectiveness of the search process, not only local information (such as the current value of the objective function) is tracked, but also some information about the path taken. This information is used as a guide when choosing the next solution within the neighbourhood.

3. Variational Quantum Algorithms

Variational Quantum Algorithms (VQAs) [1] have emerged in the NISQ [2] era as a promising class of quantum algorithms. In the NISQ era, quantum computers have a limited number of qubits and high error rates. VQAs provide a pragmatic approach to leverage quantum capabilities to address these challenges. VQAs are part of the hybrid quantum-classical paradigm, where an algorithm uses a quantum processor and a classical routine. This hybrid approach capitalizes on the quantum computer strengths and the classical computer ability to handle more optimization tasks. One of the main features of VQAs is the use of shallow quantum circuits. Instead of employing deep circuits, susceptible to errors in NISQ devices, VQAs use circuits with a small depth, which mitigate the effect of noise and qubit decoherence of the currently available devices.

Barren Plateaus Investigating the training of variational circuits, one can identify the presence of barren plateaus in the optimization landscape [9]. Similar to the concept in classical Machine Learning, a barren plateau in quantum computing refers to regions of the parameter space where the cost function gradient approaches zero exponentially as the size of the system increases. This is a problem because the classical optimizer may stop prematurely, not reaching a minimum, due to entering a barren plateau. There are some methods to mitigate the negative effects of the plateaus. The most effective one is employing shallower circuits [10]. Another one is adjusting the parameter initialization scheme. This method mitigates the risk of encountering barren plateaus during quantum circuit optimization. When a parametric gate is added to the circuit, one technique for preventing the occurrence of barren plateaus consists of setting its initial parameters so that the gate is "evaluated to the identity", i.e. the newly added gate acts as an identity operator, so that it does not modify the statevector. This precaution is taken to prevent the occurrence of barren plateaus, as adding a gate that modifies the system can lead to non-smooth transitions of the statevector. By ensuring that the gate operates as an identity, the quantum optimization process becomes more robust against the barren plateaus [11].

Here we list the most commonly used VQAs. Given their known effectiveness and ease of implementation, we introduce the VQE and then focus on the QAOA and its variants.

3.1. VQE

The most critical and field-defining hybrid algorithm is the Variational Quantum Eigensolver (VQE) [12]. It has various applications in quantum chemistry in the simulation of molecules. Its objective is to find the energy ground state of a given Hamiltonian. The first step of the algorithm is to define a parametric quantum circuit, which is called ansatz. This circuit should be designed to have the energy ground state within the states that can be obtained at the end of the circuit. The algorithm classically optimizes the parameters of the circuits using the expected value of the energy of the quantum system as the objective function. If the parameters are optimized successfully, we obtain the eigenvector and eigenvalue of the ground state of the Hamiltonian. This algorithm is usually used to calculate the ground states of a physical quantum system. However, one can map many mathematical problems to physical Hamiltonians and solve abstract problems using VQE. To do this, the primary approach is converting a mathematical problem into a QUBO problem (see in Subsection 2.3.4). Then the QUBO problem can be mapped into a physical system Hamiltonian called Ising Hamiltonian (see in Subsection 2.3.5). The performance of the VQE highly depends on the ansatz; the development of new ansatzes is a highly investigated field in this sector.

3.2. QAOA

One of the most used variational algorithms is the QAOA [3]. It is characterised by a specific ansatz structure. The first layer is a layer of Hadamard gates. Then the circuit is constructed by alternating gates implementing two parametric operators, one dependent on the problem, called the Cost Operator, which encodes the Hamiltonian into the circuit, and one called the Mixer Operator, which aims to explore the solution space. The Cost Operator is defined as

$$U_C(\gamma) = e^{-i\gamma C} \tag{15}$$

where C is the Hamiltonian Operator of which we want to calculate the ground state, and the Mixer Operator is defined as

$$U_B(\beta) = e^{-i\beta B}$$

where B is defined as

$$B = \sum_{i=1}^{n} \sigma_x^i \tag{16}$$

The QAOA circuit is defined by the depth p, representing the number of times the two alternating operators are added to the circuit.
QAOA can be seen as a discretization of a continuous-time quantum evolution. In other words, it discretizes the quantum evolution to make it compatible with the gate-based model of quantum computation.
If the initial state of the quantum system is

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0\ldots0}^{1\ldots1} |k\rangle = H^{\otimes n}|0\rangle^{\otimes n}$$

Then, the alternating application of the Cost and Mixer operator will lead the system state to the state associated with the solution to the problem.
If $p = 1$, then the circuit is

$$e^{-i\beta B}e^{-i\gamma C}H^{\otimes n}$$

If $p > 1$ we have

$$\prod_{k=0}^{p-1}\left(e^{-i\beta_{p-k}B}e^{-i\gamma_{p-k}C}\right)H^{\otimes n}, \boldsymbol{\gamma} = (\gamma_1, ..., \gamma_p) \text{ and } \boldsymbol{\beta} = (\beta_1, ..., \beta_p)$$

and can be represented by the circuit in Figure 4.

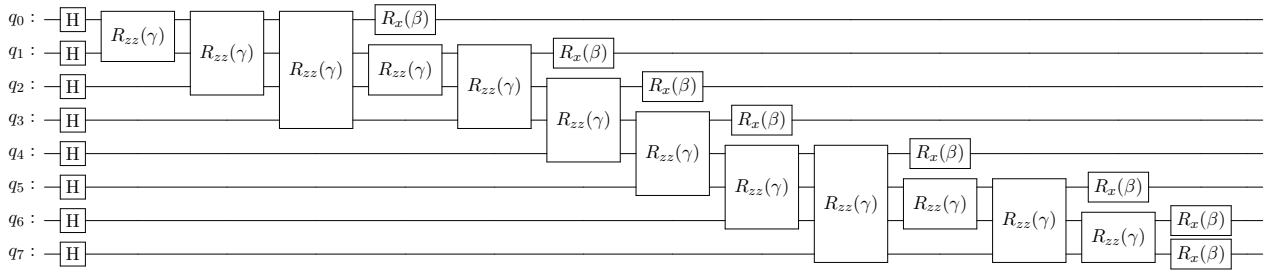


Figure 4: Circuit constructed using QAOA on Maximum Cut on a 3-regular graph with $p = 1$.

As the number of steps in the QAOA circuit increases, the algorithm has the potential to provide better approximations to the optimal solution since the expressivity of the algorithm increases. However, increasing the steps increases the number of parameters which may be a problem for the classical optimization if the number is too big. More importantly, it increases the circuit's depth, leading to higher noise and decoherence. Indeed, the algorithm efficiency depends on finding the right balance between the number of steps and the quality of the approximate solution.
The QAOA has been shown to be effective on several problems. However, there is room for improvement. Some variants of the QAOA have been formulated, trying to address some of its limitations and improve its performance.

3.3. ma-QAOA

One variant of the QAOA is the multiangle QAOA (ma-QAOA) [13], which uses the same operators C and B as the QAOA, but each parametric gate has a different parameter.
The difference stands in the definition of the Cost and Mixer operators: whereas in standard QAOA we have Equation (15), in the ma-QAOA, we have

$$U_C(\boldsymbol{\gamma}) = e^{-i\sum\gamma_k C_k}$$

where C_k is a collection of Hamiltonians such that $\sum C_k = C$, with C being the Hamiltonian derived by the problem and $\boldsymbol{\gamma}$ being the vector of γ_k. Similarly, in this case, the Mixer operator is

$$U_B(\boldsymbol{\beta}) = e^{-i\sum_k \beta_k B_k}$$

where $B_k = \sigma_x^i$, so that we obtain $\sum B_k = B$, with B being the same as Equation (16) and $\boldsymbol{\beta}$ the vector of β_k.
The strength of the ma-QAOA is that keeping the same circuit structure, depth, and number of CNOT, we can improve the approximation ratio of the solution thanks to the multiple parameters. However, we heavily increase the classical load that the computer must face. Therefore, we need to find a trade-off between the expressivity of the quantum circuit and the effectiveness of the classical optimizer.

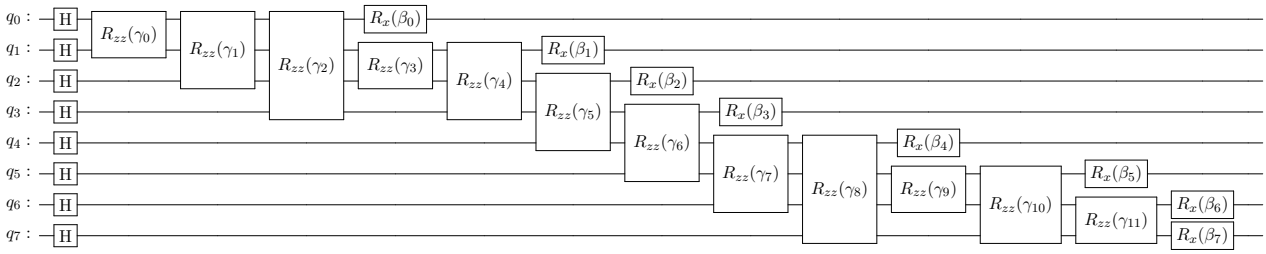


Figure 5: Circuit constructed using ma-QAOA on Maximum Cut on a 3-regular graph with $p = 1$.

3.4. QAOA+

Another example of a different ansatz with the goal of enhancing the performance of QAOA is the so-called QAOA+ [14], where the ansatz is the same as QAOA with $p = 1$, but with an added problem-independent layer. This layer is built with gates representing the operator

$$\left(\sum_{i=0}^{n-1} R_x^i(\beta_i)\right)\left(\prod_{i=2}^{n} R_{zz}^{n-i,n-i+1}(\gamma_{n-i})\right)$$

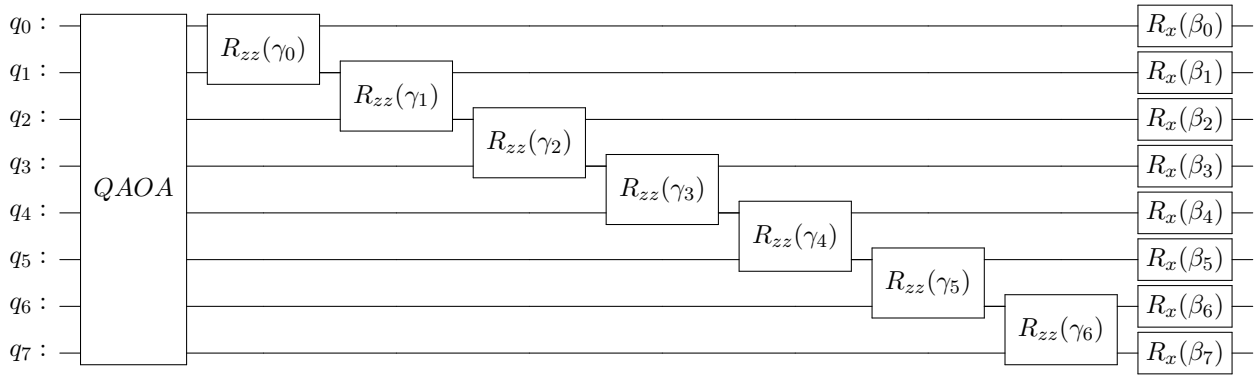


Figure 6: QAOA+ circuit.

Here, we obtain a deeper circuit, with $2n - 1$ more parameters than QAOA, which has a higher classical load but has better performance. Therefore, we need to find a trade-off similar to the one of the ma-QAOA

4. Reinforcement Learning

Reinforcement Learning (RL) is a paradigm in Machine Learning where an agent interacts with an environment, aiming to learn how to behave in the environment to maximize the cumulative reward it receives by performing actions. In this section, we explain the functionality of Reinforcement Learning and different methodologies which can be employed.

4.1. Key components

The agent is an entity that can interact with an environment. The information that the agent can obtain from the environment is called observation. When the agent obtains an observation, it chooses an action to perform according to its policy, a function which maps observations to actions. Performing the action, the agent interacts with the environment, modifying it. The modified environment will lead to a new observation, and the agent will perform a new action cyclically. Observing the environment and performing an action is called a step. Given the observation of the state (or just "state") S_t at step t, the agent chooses the action A_t and receives the reward R_t. An episode is a sequence of steps starting from an initial state of the environment and terminating when specific stopping conditions are met. Examples of these conditions are reaching a predefined maximum number of steps, encountering a terminal state (where the agent can no longer perform actions), or failing to

achieve a particular goal. An epoch is composed of multiple episodes. The agent aims to learn a policy which maximises the cumulative reward.

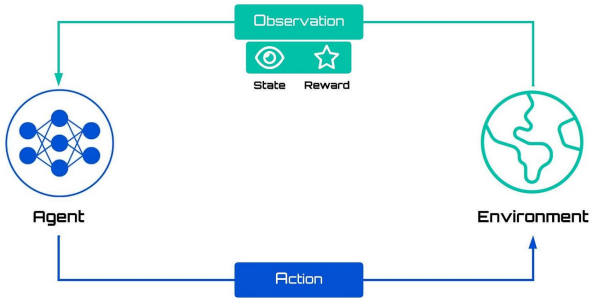


Figure 7: Schematic loop of Reinforcement Learning. The agent performs an action that interacts with the environment, leading to a new observation of the state and a new reward.

4.2. Policy and value function

The policy can be either deterministic or stochastic. A deterministic policy is in the form

$$\pi(s) = a$$

where s is a state and a is an action. It maps each state to a single action: each time the agent observes the same state, it performs the same action. A stochastic policy is in the form

$$\pi(a|s) = P(A_t = a|S_t = s)$$

where A_t and S_t are the action performed and the state observed at step t as described in Figure 7 [2]. It maps each state to a probability distribution of actions: when the agent observes twice the same state, it may perform two different actions.
At each step, the agent gains a reward. To let the agent have a long-term view of the rewards, the agent does not aim to maximize the reward achieved at each step but rather maximize the cumulative reward. The cumulative reward is given by

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$$

where T is the finite or infinite temporal horizon. Here, the parameter γ is an important Reinforcement Learning hyperparameter. It defines the importance of rewards achieved in the distant future. $\gamma = 1$ means that the reward of each step is equally important. γ close to 1 means that the agent also considers future rewards but that imminent ones are more significant. γ small means that the agent is short-sighted.
Two commonly used functions exist to evaluate the performance of a policy.
The first one is the value function. The value function is defined as the expected future reward the agent will receive using policy π given the state s:

$$V_\pi(s) = E_\pi[G_t|S_t = s]$$

The other one is the action-value function. The action-value function is defined as the expected reward for taking a particular action a in a given state s:

$$Q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a]$$

These two functions provide different perspectives on the learning process. Using the value function, the agent does not evaluate the actions but focuses only on the desirability of states. On the contrary, using the action-value function, the agent focuses on states desirability and potential actions.

[2] `https://medium.com/@achronus/learning-by-doing-a-detailed-overview-of-the-reinforcement-learning-process-05011883ad9c`

4.3. Model-based and model-free methods

A Reinforcement Learning algorithm can be either model-based or model-free. In model-based RL, the agent builds an internal model of the environment it interacts with. This model captures the dynamics of the environment, including the transitions between states and the associated rewards. The agent then uses this model to simulate possible future scenarios and make decisions based on the predictions of the model. In model-free RL, the agent directly learns the policy. Instead of building an explicit model of the environment, the agent learns from interactions with the environment by trial and error. In scenarios with a large state or action space, opting for a model-based approach poses challenges such as computational complexity and impractical memory requirements. Model-free approaches are favoured due to their scalability and adaptability in handling large environments.

4.4. Exploration vs Exploitation

Exploration vs exploitation is one of the most critical trade-offs in Reinforcement Learning, where an agent must decide between exploring new possibilities or exploiting known information to maximize cumulative rewards. Exploration involves trying new actions that lead to unexplored environments to discover potential high-reward outcomes. On the other hand, exploitation makes decisions based on the current knowledge to maximize the gains by choosing actions with known positive outcomes. Having the right balance between exploration and exploitation is crucial for effective learning, as an unbalanced choice of one over the other may lead to either suboptimal policies due to insufficient exploration or missed opportunities for immediate reward due to excessive exploration.

4.5. Value-based methods

Value-based methods in Reinforcement Learning focus on estimating the value of states or state-action pairs to learn the optimal policy. These methods are particularly relevant in scenarios where the agent aims to learn an optimal policy indirectly.
The learning process involves updating these value estimates based on the observed rewards and transitions during the agent's interaction with the environment. Over time, the value function converges towards more accurate values, guiding the agent to make better decisions.
Value-based methods are known for their simplicity and effectiveness. Being model-free, these methods do not explicitly model the environment dynamics but instead focus on learning from experiences to approximate the optimal value function.

4.6. Policy-based methods

Policy-based Reinforcement Learning methods centre around the policy parameterisation and optimisation. Unlike value-based methods that estimate the value of states or state-action pairs, policy-based methods aim to find the optimal policy itself.
These approaches define a parameterised policy, which maps observations to probability distributions of actions. The parameters of this policy are iteratively adjusted to maximize the expected cumulative reward.
Policy gradients represent the direction in the parameter space that increases the probability of actions leading to higher rewards. By updating the policy parameters along these gradients, the agent improves its expected cumulative reward.
Policy-based methods have several advantages, including their suitability for dealing with high-dimensional and continuous action spaces and their ability to handle stochastic policies.
Furthermore, policy-based methods are more robust in complex and noisy environments. Directly optimising policies allows for greater flexibility and adaptability.

4.7. Actor-Critic

The actor-critic method uses both value-based and policy-based approaches, introducing a dual-network architecture. The agent consists of two components: an actor and a critic.
The actor is responsible for selecting actions based on the current policy, which is a parameterized function mapping observations to actions. It represents the policy of the agent.
On the other hand, the critic evaluates the chosen actions by estimating the value of states and state-action pairs. This estimation provides feedback to the actor. The critic's role is to be the value function in traditional

value-based methods, assessing the desirability of states and actions.
The actor and critic are often neural networks with shared or separate parameters. During the learning process, the critic's estimates are used to compute advantages, defined as

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s) \tag{17}$$

which are then used to update both the actor and critic parameters. This integration of value estimation (critic) and policy improvement (actor) allows for more stable and efficient learning.
Actor-Critic methods have the strengths of value-based and policy-based approaches, and incorporating the critic helps mitigate the high variance associated with pure policy gradients.

4.8. Proximal Policy Optimization

Proximal Policy Optimization (PPO) [15] is a state-of-the-art Reinforcement Learning policy optimization method. PPO addresses some of the challenges associated with traditional policy gradient, ensuring more stable and efficient learning.
The distinctive feature of PPO is the use of a clipping function in the objective function. This function limits the policy update to ensure that the new policy does not deviate too far from the old policy. The motivation behind this clipping is to prevent large policy updates that could lead to unstable learning.
PPO uses an objective function, quantifying the difference between the new and old policies regarding expected rewards. By optimizing this objective while constraining the policy update, PPO balances exploring new policies and exploiting the current best-known policy. The objective function of the PPO is

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\cdot\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\cdot\hat{A}_t)]$$

where $\hat{A}_t$ is the estimate of 17 and $r_t(\theta)$ is the probability ratio defined as

$$r_t(\theta) = \frac{\pi(\theta)(a_t, s_t)}{\pi(\theta_{old})(a_t, s_t)}$$

Additionally, PPO often employs an actor-critic architecture, where a value function (critic) estimates the advantage of taking certain actions in specific states.
Its robustness and ability to handle diverse environments make PPO a popular choice for researchers and practitioners.

5. Design of the agent

In this section, we explain our implementation of the Reinforcement Learning agent, its training procedure and how we analyze the results obtained.
Our Reinforcement Learning agent is designed to acquire the knowledge of building quantum circuits to minimize the expectation of the Hamiltonian associated with a specific problem.
The algorithm we use is PPO, and all the hyperparameters are set using the default values of the OpenAI implementation: gamma = 0.99, clip_ratio = 0.2, pi lr = 3e-4, vf_lr=1e-3, lam=0.97, target kl=0.01.[3]
The fundamental elements of the Reinforcement Learning algorithms are the environment and its observation, the actions, the step function and the reward function (see Section 4). In this section, we describe the specific choices made for our algorithm.

5.1. Observations

The environment is a parametric circuit with n qubits, and its parameters. An element of the observation space is a vector of 2^n real numbers representing the estimated probability of obtaining each basis state after simulating the circuit and performing a measurement. In each simulation the circuit is run 1000 times, and the final state is measured each time. The estimated probabilities are given by assigning to each basis state a value equal to the number of times the state has been measured over the number of total measurements. The initial environment of each episode is a circuit with a single layer of Hadamard gates.

[3] `https://github.com/openai/spinningup/blob/master/spinup/algos/pytorch/ppo/ppo.py`

5.2. Actions

An agent's action is an element of the action set. Each action is the application of a quantum gate to one or two qubits of the environmental circuit.
We define the following sets:

- S = $\{R_a^i(\theta) \mid a \in \{x, y, z\},\ i = 0, ..., n-1\}$
- D = $\{R_{ab}^{ij}(\theta) \mid a, b \in \{x, y, z\},\ i, j = 0, ..., n-1,\ i \neq j\}$

The set S is composed by single R_x, R_y, R_z rotations applied to qubits from 0 to $n-1$, and the set D is composed by double R_{xx}, R_{xy}, R_{xz}, R_{yy}, R_{yz}, R_{zz} rotations applied to all the pairs of different qubits. Every single rotation appears n times in S, once per each qubit it can be applied to. The xx, yy, and zz rotations appear $\frac{n \cdot (n-1)}{2}$ times in D. The xy, xz, and yz rotations appear $n \cdot (n-1)$ times in D, since R_{xy}^{12} is different from R_{xy}^{21}. In our experiments, we train an agent using $A = S \cup D$ as action set.
A valuable characteristic of the aforementioned gates is that they can all be "evaluated to the identity" (see Section 3).

5.3. Reward

In the context of an agent trained to build variational circuits, the reward function can be any strictly monotonic function that decreases when the expectation value $\langle \hat{H} \rangle$ increases. We use $-\langle \hat{H} \rangle$ [16] to which we subtract a penalty proportional to the depth of the circuit, equal to $0.015 \cdot depth$. This penalizes deeper circuits over shallower circuits in circuits having a similar expectation of the Hamiltonian. A shallower circuit is more desirable since a deeper circuit is more sensitive to the decoherence of the quantum state and the Barren Plateau phenomenon.

5.4. Step function

The step function takes the action as input, modifies the environment and returns the observation and the reward. A step consists of applying the action gate to the circuit and optimizing the circuit parameters using the COBYLA [17] classical optimizer, minimizing the estimate of the expectation of the problem Hamiltonian on the final quantum state of the circuit. All the hyperparameters of the classical optimization (i.e. maximum steps, stopping criterion, tolerance...) are the default ones of the scipy.optimize.minimize function.
In the end, the step function calculates the reward and the estimated probability distribution associated with the final state of the circuit with optimized parameters.

5.5. Actor and Critic

The agent is composed of an actor and a critic. They are modelled using a neural network. They share the same network architecture but are two separate instances, and their parameters are optimized separately.
The chosen neural network structure is the default one in the OpenAI implementation, which has two hidden layers, each with 64 neurons, with ReLU as the activation function between them. The actor and critic networks can be interpreted as a classification network, followed by a sampling layer, which samples the action to choose. The input and output layers of the classification network depend on the size of the observation vector and the action sets. The input layer has 2^n neurons. The output layer has size $|A|$. Afterwards, the actor and critic networks return only a single action as output.

5.6. Training

In this subsection, we describe the problem instances on which the agent is trained and explain the details of the training procedure.

5.6.1. Problem instances

Our agent is trained on three different problems: Maximum Cut, Maximum Clique and Minimum Vertex Cover problem. The specific problem instances are built by specifying the underlying graph, with its number of vertices and its topology. In particular, for each problem, we train the agent on three different graph topologies: graphs of regularity three, graphs with a 2D-grid topology, and graphs with a star topology. We train the agent for each problem typology and graph topology on graphs with 8 vertices and 14 vertices.

5.6.2. Training details

The agent is trained for 64 epochs of at most 64 steps. We use a message passing model in which the multiple threads communicate at the end of each epoch. The threads are independent while performing the steps of each episode. When an epoch ends, they share the information accumulated and the optimization of the actor and critic networks is done on the master thread. In our experiment, we use six threads. An episode is long a maximum of $2 \cdot n$ steps but can end sooner using the parameter *patience*. *patience* is initially set to 3, and if an action does not achieve a better reward, then *patience* decreases by one; else, *patience* increases by one but only if it does not exceed the initial value. When *patience* reaches 0, the episode ends. The *patience* parameter helps to prevent the agent from spending excessive time in unproductive or challenging situations. However, *patience* will not truncate the episode immediately if such a scenario is reached, giving the agent a longer-term view.
At the end of the training, we analyze the circuit with the highest reward by considering its approximation ratio and the gate usage.

6. Linear Circuit analysis methodology

During the training on the Maximum Cut problem, the agent found a family of circuits that with a specific structure, able to achieve an approximation ratio close to 1. Therefore, following the pattern the agent discovered we were able to build a family of new ansatzes, which we denote as R_{yz}-connected family, and decided to analyze its performance and suitability when the problem instance is changed.
This section describes how we conduct the analysis of a specific R_{yz}-connected ansatz, that we denote as Linear Circuit.

6.1. Baselines

The Linear Circuit is analyzed by comparing it with the ansatzes of some state-of-the-art algorithms. We select QAOA with depth $p = 1$, QAOA with depth $p = 2$, QAOA+ and ma-QAOA and the classical algorithms Steepest Descent (SD), Tabu Search (TS), and Simulated Annealing (SA). The classical algorithms have been implemented using the D-Wave library with all the default settings.
Specifically, we analyze its performance in solving the Maximum Cut problem with different underlying graph topologies and compare it to the performance of the other algorithms: we use a 3-regular graph, a 2D-grid graph, a cycle graph and three Erdős–Rényi graphs with edge probabilities $0.2, 0.5$ and 0.8.
Then, to test the suitability of the Linear Circuit with different problem typologies, we compare the Linear Circuit with the other algorithms on the Maximum Clique and the Minimum Vertex Cover problem. The algorithms are tested on instances having 8, 14, and 16 vertices.

6.2. Experiments

We use the Qiskit QASM simulator with 1000 shots for all the quantum algorithms. Each time, we estimate the probability distribution associated with the final state of the circuit. We use COBYLA as a classical optimizer and use random initial parameters for each experiment. The experiments of all the algorithms and the Linear Circuit are run ten times.
We consider the solution of an experiment as the solution with the best approximation ratio found during the algorithm execution. It is worth noting that in the context of variational algorithms, the approximation ratio is defined for the final state of a quantum circuit. However, we introduce the notion of approximation ratio of a binary solution, defined as the approximation ratio of the basis state associated with that solution.

6.3. Metrics

The metrics used for the evaluation of the algorithms are the number of steps, the solution ratio (i.e. the ratio between the cost of the obtained solution and the cost of the exact solution) and the hamming distance of the solution from the closest exact solution.
Since the quantum algorithms do not directly give us a binary solution, but rather probabilities associated with the basis states, we consider additional metrics: the approximation ratio and the weighted Hamming distance using such probabilities as weights.
Given the small dimension of the problem instances, we obtain the actual minimum value of the cost function and the exact solutions necessary for computing the approximation ratio and the Hamming distance, by using

an exact solver. If the problem has a negative minimum value and a positive maximum value, the approximation ratio might assume a negative value in the worst cases. To address this issue, when we calculate the approximation ratio, we subtract an offset to both terms of the ratio to obtain a range of values between 0 and 1.
We evaluate the algorithms and ansatz performance by analyzing the values of the mean and standard deviation of the aforementioned metrics throughout the ten independent experiments.

7. Results

In this section, we present the results obtained by the Reinforcement Learning agent. Then, we describe the R_{yz}-connected ansatzes and analyze the Linear Circuit.

7.1. General circuits analysis

In this subsection, we analyze the circuits obtained with the Reinforcement Learning algorithm as explained in Subsection 5.6.2. For each problem instance, we consider the circuit having the highest reward found during the training. Moreover, we apply some simplifications to these circuits. Specifically, we eliminate rotation gates with angles distant from 0 or 2π less than 0.1, as these gates do not contribute significantly to modifying the final quantum state. If there are two consecutive equal rotations, we merge them by summing their rotation angles. Furthermore, we apply heuristic substitutions of pairs of gates if we find one gate that can replace both without changing the statevector amplitudes. This is done to reduce the circuit depth and the number of parameters without preventing the circuit from achieving the same expectation value.

Size	Problem	Graph	$\#R_x$	$\#R_y$	$\#R_z$	$\#R_{xx}$	$\#R_{xy}$	$\#R_{xz}$	$\#R_{yy}$	$\#R_{yz}$	$\#R_{zz}$	Depth	A.R.	A.R. Threshold
8	Maximum Cut	Star	0	0	0	0	0	0	0	7	0	28	0.99	0.00
		Grid	0	0	0	0	0	0	0	7	0	28	0.99	0.00
		3-Reg	0	0	0	0	0	0	0	7	0	28	0.99	0.00
	Maximum Clique	Star	0	3	0	0	4	0	0	5	0	34	0.70	0.95
		Grid	1	2	0	0	4	0	0	6	0	39	0.83	0.94
		3-Reg	0	3	0	0	5	0	0	6	0	36	0.75	0.91
	Minimum Vertex Cover	Star	0	3	0	0	2	0	0	0	2	16	0.93	0.69
		Grid	0	5	0	0	1	0	0	0	2	13	0.96	0.73
		3-Reg	0	4	0	0	8	0	0	1	0	39	0.99	0.77
14	Maximum Cut	Star	0	0	0	0	0	0	0	7	0	28	0.5	0.00
		Grid	0	0	0	0	0	0	0	12	0	48	0.76	0.00
		3-Reg	0	0	0	0	0	0	0	13	0	52	0.92	0.00
	Maximum Clique	Star	0	0	0	1	7	1	1	7	0	51	0.63	0.99
		Grid	0	0	0	1	6	1	0	4	0	31	0.72	0.99
		3-Reg	0	0	1	0	6	1	0	6	0	44	0.64	0.98
	Minimum Vertex Cover	Star	0	0	0	0	3	0	0	4	0	28	0.65	0.71
		Grid	0	1	0	0	8	1	0	0	0	29	0.87	0.74
		3-Reg	0	0	1	0	9	0	0	4	1	50	0.85	0.74

Table 1: Reinforcement Learning results. The first nine columns represent the number of occurrences of each gate instance. The table also includes information on the depth of the circuit and the approximation ratio achieved, followed by the approximation ratio that a solution must surpass to be feasible.

Table 1 summarizes the gate usage and performance of the circuits with the highest reward. The first nine columns represent the number of occurrences of each specific gate in the circuit after the simplifications. Then, the table shows the circuit depth, followed by the approximation ratio and the approximation ratio threshold that a solution must surpass to be feasible.

Gate usage We can note that the agent trained on instances of the Maximum Cut problem creates circuits that use only R_{yz} rotations. In particular, these circuits follow a specific pattern which is described in Subsection 7.2. We observe that in the Maximum Clique and Minimum Vertex Cover 8 vertices instances, both one-qubit and two-qubit gates are used, in particular, there is a preference for using R_y gates. This is an indication that

the agent is trained correctly and builds circuits that try to achieve a final state with high probabilities for the basis states corresponding to optimal solutions. Indeed, if a circuit of n qubits is composed of a layer of n Hadamard gates and a layer of n parametric $R_y(\theta)$ rotations, the final state of the circuit can assume the value of all the basis states, in particular the ones of the exact solutions. To see this, we can notice the absence of entanglement in the circuit, therefore we analyze each qubit independently. In particular, it is immediate to prove that $R_y(\frac{\pi}{2})H|0\rangle = |0\rangle$ and that $R_y(-\frac{\pi}{2})H|0\rangle = |1\rangle$, except for a global phase. Therefore, using all the possible combinations of $\frac{\pi}{2}$ and $-\frac{\pi}{2}$ values for the rotation parameters, we are able to obtain all the basis states. However, the optimization of these parameters takes exponential time [18] since this circuit does not exploit entanglement and can be efficiently classically simulated (see Subsection 2.1.4). Therefore, alternative circuits able to leverage quantum properties might be preferable for solving the same task.
By the analysis of Table 1, for the Maximum Clique and Minimum Vertex Cover 14 vertices instances, the agent needs to consider gates different from the R_y rotation to find good solutions. The gates most used by the agent are the R_{yz} and the R_{xy}. The agent's choice of using almost only two-qubit gates could be related to the fact that a circuit with high entanglement might lead to faster convergence to a state with a higher approximation ratio [19].

Depth Table 1 also shows the depth of the circuits. We can note that the depths of the circuits built to solve the Maximum Cut problem are four times the number of R_{yz}. For the other problem types, the depths are more irregular. In particular, while solving the Minimum Vertex Cover, the agent employs deeper circuits on 3-regular graph than on other topologies, achieving higher approximation ratios. On the other hand, when solving the Maximum Clique problem, deeper circuits do not lead to a better approximation ratio.

Approximation ratio The approximation ratio obtained by the final state of the circuit with optimal parameters is shown in the third to last column of Table 1. In the Maximum Cut instances, the approximation ratio in the 8 vertices instances is very high and the solutions are measured with probability almost 1. In the 14 vertices instances, the approximation ratios are lower and seem to be related to the number of R_{yz} employed in the circuit. The more R_{yz} are used, the higher the approximation ratio is.
For the Maximum Clique problem, the approximation ratio is lower than in the other two 8 vertices problem instances, and it is comparable with the approximation ratio obtained solving the other two problems on 14 vertices instances.
However, the probability of measuring an unfeasible solution in the Maximum Clique problem is very high, since the achieved approximation ratio is lower than the threshold shown in the last column of Table 1. Indeed, Maximum Clique and Minimum Vertex Cover are constrained problems and, in their QUBO formulation, their constraints are modelled as penalties. These penalties are valued so that all the unfeasible solutions are associated with higher energies than all the energies related to feasible solutions. This means either that we measure only unfeasible solutions or that the probability of measuring a feasible solution is low.
The approximation ratios obtained by the agent trained on solving the Minimum Vertex Cover are close to 1 for the 8 vertices instances and below 0.9 for the 14 vertices instances. In particular, in the 14 vertices instances, the approximation ratio is under the feasibility threshold only with the star topology graph. This might be due to the peculiar solution of the Minimum Vertex Cover on a star graph, in which only one vertex is needed to cover all the edges.
In the next subsection, we will focus on the circuits built by the agent trained on the Maximum Cut problem and infer a generalised structure.
However, it is worth noting that also the results of the circuit built by the agent solving the Minimum Vertex Cover seem promising. Therefore, a deeper analysis might be a good expansion of this work.

7.2. R_{yz}-connected ansatzes

In this subsection, we present the R_{yz}-connected family, discovered by the Reinforcement Learning agent trained on the Maximum Cut problem. The R_{yz}-connected ansatzes are characterized by an initial layer of Hadamard gates, followed by $n-1$ R_{yz} rotations. In particular, these rotations are applied so that each new R_{yz} adds only one qubit to the "chain" of entangled qubits. More formally, calling C_j the graph representing the connectivity of the circuit after applying the first j R_{yz} rotations, the C_j of the ansatzes of the R_{yz}-connected family are connected graphs for all $j = 1, \dots, n-1$. We denote the family of ansatzes discovered while training on the Maximum Cut as the R_{yz}-connected family.
We now analyze the mathematical properties of R_{yz}-connected ansatzes and then discuss the experimental results of the Linear Circuit.

7.2.1. R_{yz}-connected proprierties

As shown in Appendix A, it is possible to show that the parametric statevector of the final state of the generic member of the family is given by

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\frac{1+i}{\sqrt{2}}\right)^{n-1} \sum_{b\in\{0,1\}^n} \prod_{i,j\in C} \cos\left(\frac{\theta_{ij}}{2} + (-1)^{b_i \neq b_j}\frac{\pi}{4}\right)|b\rangle$$

where $b = b_{n-1}\dots b_0$ is a binary number, C is the graph representing the circuit connectivity and

$$(-1)^{x=y} = \begin{cases} -1, & \text{if } x = y \\ 1, & \text{otherwise} \end{cases}$$

Defining c_b as the eigenvalue associated to the $|b\rangle$ eigenvector of the Hamiltonian $\hat{H}$

$$\hat{H}|b\rangle = c_b|b\rangle$$

the parametric expression of the expected value of the Hamiltonian is calculated in Appendix B and is

$$\langle\hat{H}\rangle = \frac{1}{2}\sum_{b\in\{0,1\}^n} c_b \prod_{i,j\in C}\left(\cos\left(\frac{\theta_{ij}}{2} + (-1)^{b_i\neq b_j}\frac{\pi}{4}\right)\right)^2 \tag{18}$$

The Hamiltonian and its eigenvalues depend on the specific QUBO problem we are tackling. Optimizing Equation (18) can be seen as the continuous relaxation of the original binary QUBO problem where, instead of optimizing the discrete variables $\{b_0,\dots,b_{n-1}\}$, one optimizes the continuous variables $\{\theta_{ij}\}$. The effectiveness of this relaxation depends on the problem and the graph connectivity.
It is important to note that the expectation value formula in Equation (18) has a summation with an exponential number of terms in the size of the problem and cannot be efficiently classically simulated.

7.2.2. Linear Circuit

The Linear Circuit with n qubits uses $n-1$ R_{yz} rotations applied to all the pairs of consecutive qubits (see Figure 8).

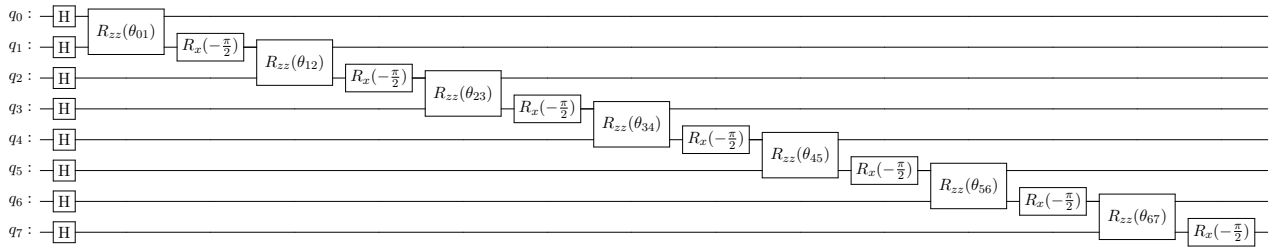


Figure 8: Linear Circuit, a specific R_{yz}-ansatz characterized by each qubit being connected to the following one.

As seen in Equation (9), a R_{yz} rotation can be decomposed in two R_x rotations and a R_{zz} rotation, but in the circuits the $R_x(\frac{\pi}{2})$ that should precede each R_{zz} disappears because $R_x(\frac{\pi}{2})H|0\rangle = H|0\rangle$ except for a global phase.

7.2.3. Use of entanglement

In this subsection we describe an ansatz belonging to the R_{yz}-connected family and show that it is equivalent to a circuit without entanglement. Our goal is to highlight the importance of verifying that the circuit effectively uses entanglement, in order to fully leverage its quantum properties. The circuit we analyze is characterized by the i^{th} rotation being applied to the qubit 0 and the qubit i, for $i = 1,\dots,n-1$ (see Figure 9).
This circuit is similar to the circuit built by applying QAOA with depth $p = 1$ to star topology graph problem instances (see Figure 10).

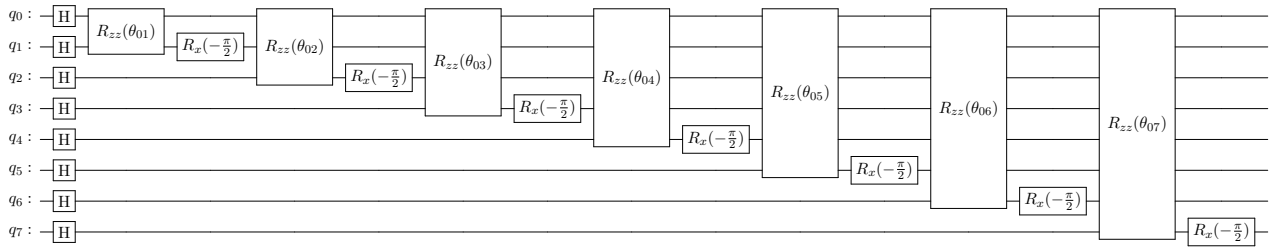


Figure 9: R_{yz}-connected circuit with all qubits entangled to a single one.

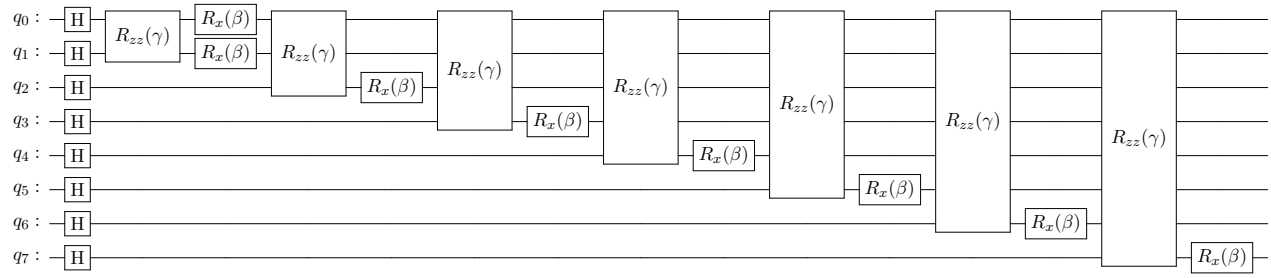


Figure 10: QAOA circuit for solving the Maximum Cut on a star topology graph.

We can prove that this specific family member structure, which entangles only one qubit to all the others, has a parametric expectation value of the Hamiltonian, when solving the Maximum Cut problem, equivalent to the one of a circuit without entanglement.
As calculated in Appendix B, the expected value in the R_{yz}-connected circuit in Figure 9 is

$$\langle \hat{H} \rangle = \frac{1}{2} \sum_{b \in \{0,1\}^n} c_b \prod_{j=1}^{n-1} \left(\cos\left(\frac{\theta_{0j}}{2} + (-1)^{b_0 \neq b_j} \frac{\pi}{4} \right) \right)^2 \tag{19}$$

where b is a binary number $b_{n-1} \dots b_0$.
We can rewrite the expectation value related to the circuit in Figure 11 calculated in Appendix B as

$$\langle \hat{H} \rangle = \sum_{b \in \{0,1\}^{n-1}} c_{2b} \prod_{j=1}^{n-1} \left(\cos\left(\frac{\theta_j}{2} + (-1)^{b_0 \neq b_j} \frac{\pi}{4} \right) \right)^2 \tag{20}$$

where $2b$ is an even number in binary representation and b_j is the j^{th} least significant bit of $2b$.
We know that in the Maximum Cut problem, the c_k are symmetrical: if $\bar{l}$ is the bitflip of l, then $\hat{H}(|l\rangle \otimes |0\rangle) = \hat{H}(|\bar{l}\rangle \otimes |1\rangle)$ for any l. Thus $|l\rangle \otimes |0\rangle$ and $|\bar{l}\rangle \otimes |1\rangle$ have the same eigenvalue. Therefore, Equations (19) and (20) are equivalent because in the first equation, b_0 can be either 0 or 1, and in the second equation, b_0 is 0. This is because we are considering $2b$, which is an even number and always has $b_0 = 0$.
Therefore, Equation (19) considers all 2^n c_b and Equation (20) considers half of the c_b, but with double the magnitude (since Equation 19 has a $\frac{1}{2}$ before the summation). Since Maximum Cut has symmetric solutions, each element of the summation in Equation (20) is equivalent to two elements of the summation in Equation (19), making the two equations equivalent.
However, only the circuits where all the $n-1$ R_{yz} have a qubit in common can be substituted by one without entanglement. This is because it entangles pairs of qubits, one pair independently from the other. In other R_{yz}-connected circuits, such as the Linear Circuit, the entanglement structure created is more complex, and they generate parametric statevectors that are not replicable by circuits without entanglement. The formal proof is presented in Appendix C.
The same reasoning can be applied to QAOA too. When QAOA is applied to solve the Maximum Cut on a star topology graph (see Figure 10). The parametric expectation value is the same as the one of the circuit in Figure 12.

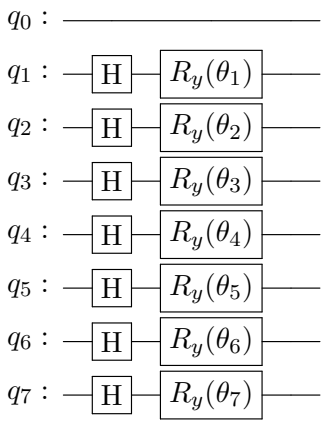


Figure 11: Circuit with no entanglement which has the same expectation value of the Hamiltonian associated to the circuit in Figure 9 when solving the Maximum Cut problem.

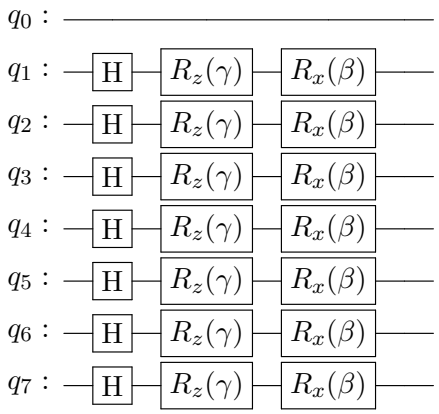


Figure 12: Circuit with no entanglement. When this circuit is used to solve the Maximum Cut problem, the expectation value of the Hamiltonian is the same as the one achieved by the circuit in Figure 10.

This is true because each R_{zz} rotation adds a relative positive phase to the states $|00\rangle$ and $|11\rangle$ and a relative negative phase to the states $|01\rangle$ and $|10\rangle$, which is equal (specifically to solve the Maximum Cut problem) to set one qubit to $|0\rangle$ and apply to the other qubits a relative positive phase if equal to $|0\rangle$ and a relative negative phase if equal to $|1\rangle$.
This R_{yz}-connected ansatz, when only one qubit is connected to all the others and the QAOA circuit is applied to a star graph, can be substituted by one without entanglement. This fact might lead to the conclusion that the generic QAOA circuit and the R_{yz}-connected family might not be worth using on the Maximum Cut problem. However, this is not true: the power of QAOA and the R_{yz}-connected family arises from their flexibility. The QAOA applied to a generic graph and the other family members are not equivalent to a circuit without entanglement. Therefore, they are exploiting the entanglement correctly. QAOA is flexible and can be adapted to the topology of the problem instance. Our R_{yz}-connected ansatzes can be instead adapted to any physical topology (see Subsection 7.3.5). Experimental results lead us to the conclusion that R_{yz}-connected ansatzes are effective even in scenarios where the circuit connectivity does not match the problem instance graph (see Subsection 7.3.1).

7.3. Linear Circuit analysis

In this subsection, we analyze the performance of the Linear Circuit, depicted in Figure 8, comparing it with the state-of-the-art quantum algorithms described in Subsection 3 and the classical algorithms described in Subsection 2.4.

7.3.1. Performance on Maximum Cut problem

All the algorithms are initially applied to Maximum Cut instances built from graphs with 8, 14 and 16 vertices having different graph topologies. Table 2 shows the average results obtained with ten executions of each algorithm on each graph topology of 16 vertices, along with their standard deviations. The results of the experiments with smaller problem instances are are not reported since they exhibit a similar pattern.

Steps The column steps of Table 2 represents the number of iterations the classical optimizer performs before convergence. Given the distinct nature of the classical algorithms compared to the quantum ones, establishing a comparable definition of steps is not possible. Therefore, the values of steps for classical algorithms are not included in the table. The Linear Circuit requires a number of steps superior to those required by QAOA with depth $p = 1$ and $p = 2$ but inferior to those needed by QAOA+ and ma-QAOA. This can be explained by the fact that the number of parameters, which is higher in QAOA+ and ma-QAOA, heavily influences the number of steps: as the number of parameters increases, the number of iterations required for the classical optimization increases.

Solution ratio and solution Hamming distance Table 2 also shows the solution ratio and the Hamming distance relative to the solution (see Subsection 6.2) of the experiment. The solution ratio is defined as the solution cost over the exact solution cost. The solutions ratios are very high. Tabu Search, QAOA+ and ma-QAOA always find the exact solutions, and all the quantum algorithms achieve a solution ratio above 0.95. The only case were the solution ratios are not high is with classical algorithms applied to the cycle topology. In this case all the quantum algorithms achieve a solution ratio equal to 1, whereas Simulated Annealing and Steepest Descent obtain an average of 0.52 and 0.78. When the solution ratio has a high mean, it has a low standard deviation, which means that good results are achieved consistently. The behaviour of the Hamming distances reflect the one of the Solution ratios. Where the solution ratios are high, the Hamming distances are low and vice versa. In particular, when the mean Solution ratio is 1, the mean Hamming distance is 0. The two highest value of the Hamming distance are obtained by Simulated Annealing and Steepest Descent applied on the cycle topology.
Finally, Table 2 shows that, while the Linear Circuit provides high-quality solutions, they are less reliable in finding optimal solutions than the other tested algorithms. This is due to how the solution space is explored (see Subsection 7.3.3).

Approximation ratio and mean Hamming distance In Table 2, the approximation ratio and the mean Hamming distance are obtained by averaging the costs of the basis states weighted with the estimated probabilities of the final quantum state of the circuit with optimal parameters. The comparison of approximation ratios is also illustrated in Figure 13, providing a visual comparison between the different algorithms for each graph topology.

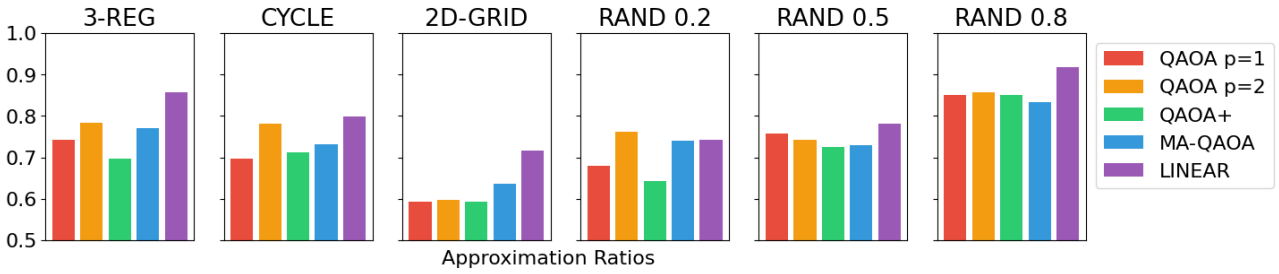


Figure 13: Approximation Ratios on Maximum Cut of different algorithms, across diverse graph topologies.

Table 2 and the histograms in Figure 13 show that the approximation ratios of the Linear Circuit are almost always the highest, except for the random topology with edge probability 0.2, where the approximation ratio is similar to the other.

Topology	Algorithm	Steps	Solution ratio	Solution Hamming distance	Approx. ratio	Mean Hamming distance
3 reg	SA	*N.A.*	0.97 ± 0.03	2.2 ± 3.00	*N.A.*	*N.A.*
	SD	*N.A.*	0.95 ± 0.03	3.2 ± 2.22	*N.A.*	*N.A.*
	TS	*N.A.*	1.00 ± 0.00	0.0 ± 0.00	*N.A.*	*N.A.*
	QAOA1	30.2 ± 2.31	0.99 ± 0.01	0.1 ± 0.30	0.74 ± 0.07	6.00 ± 0.24
	QAOA2	48.3 ± 3.37	1.00 ± 0.00	0.0 ± 0.00	0.78 ± 0.06	5.69 ± 0.28
	QAOA+	319.2 ± 28.00	1.00 ± 0.00	0.0 ± 0.00	0.69 ± 0.03	6.19 ± 0.19
	maQAOA	429.8 ± 35.02	1.00 ± 0.00	0.0 ± 0.00	0.77 ± 0.02	5.78 ± 0.15
	Linear	182.5 ± 11.06	0.97 ± 0.02	1.3 ± 1.41	0.85 ± 0.03	4.50 ± 1.18
2d grid	SA	*N.A.*	1.00 ± 0.00	0.0 ± 0.00	*N.A.*	*N.A.*
	SD	*N.A.*	0.89 ± 0.08	2.6 ± 2.20	*N.A.*	*N.A.*
	TS	*N.A.*	1.00 ± 0.00	0.0 ± 0.00	*N.A.*	*N.A.*
	QAOA1	29.1 ± 2.21	0.98 ± 0.03	0.2 ± 0.40	0.59 ± 0.09	5.96 ± 0.51
	QAOA2	48.6 ± 2.93	1.00 ± 0.00	0.0 ± 0.00	0.59 ± 0.08	5.87 ± 0.52
	QAOA+	322.3 ± 37.44	1.00 ± 0.00	0.0 ± 0.00	0.59 ± 0.03	5.96 ± 0.19
	maQAOA	391.2 ± 34.23	1.00 ± 0.00	0.0 ± 0.00	0.63 ± 0.01	5.75 ± 0.13
	Linear	183.8 ± 24.90	0.97 ± 0.04	0.2 ± 0.40	0.71 ± 0.06	4.29 ± 0.82
cycle	SA	*N.A.*	0.52 ± 0.10	5.4 ± 1.50	*N.A.*	*N.A.*
	SD	*N.A.*	0.78 ± 0.08	4.8 ± 1.30	*N.A.*	*N.A.*
	TS	*N.A.*	1.00 ± 0.00	0.0 ± 0.00	*N.A.*	*N.A.*
	QAOA1	31.8 ± 2.56	1.00 ± 0.00	0.0 ± 0.00	0.69 ± 0.09	5.76 ± 0.33
	QAOA2	53.2 ± 4.30	1.00 ± 0.00	0.0 ± 0.00	0.78 ± 0.02	5.41 ± 0.20
	QAOA+	354.0 ± 35.55	1.00 ± 0.00	0.0 ± 0.00	0.71 ± 0.03	5.71 ± 0.12
	maQAOA	335.1 ± 18.10	1.00 ± 0.00	0.0 ± 0.00	0.73 ± 0.01	5.68 ± 0.07
	Linear	182.8 ± 14.71	1.00 ± 0.00	0.0 ± 0.00	0.79 ± 0.05	4.83 ± 0.53
random 0.2	SA	*N.A.*	0.98 ± 0.02	1.2 ± 1.8	*N.A.*	*N.A.*
	SD	*N.A.*	0.94 ± 0.05	3.0 ± 3.00	*N.A.*	*N.A.*
	TS	*N.A.*	1.00 ± 0.00	0.0 ± 0.00	*N.A.*	*N.A.*
	QAOA1	33.1 ± 3.80	1.00 ± 0.00	0.0 ± 0.00	0.68 ± 0.05	4.87 ± 0.23
	QAOA2	54.5 ± 5.53	1.00 ± 0.00	0.0 ± 0.00	0.76 ± 0.03	4.48 ± 0.28
	QAOA+	307.9 ± 24.01	1.00 ± 0.00	0.0 ± 0.00	0.64 ± 0.02	5.08 ± 0.11
	maQAOA	392.0 ± 50.19	1.00 ± 0.00	0.0 ± 0.00	0.74 ± 0.02	4.62 ± 0.12
	Linear	184.9 ± 30.64	0.98 ± 0.02	0.7 ± 1.18	0.74 ± 0.06	4.76 ± 0.55
random 0.5	SA	*N.A.*	0.99 ± 0.01	0.6 ± 1.80	*N.A.*	*N.A.*
	SD	*N.A.*	0.95 ± 0.04	3.0 ± 2.56	*N.A.*	*N.A.*
	TS	*N.A.*	1.00 ± 0.00	0.1 ± 0.30	*N.A.*	*N.A.*
	QAOA1	32.2 ± 5.01	0.99 ± 0.00	0.4 ± 1.20	0.75 ± 0.05	5.42 ± 0.27
	QAOA2	49.8 ± 7.74	0.99 ± 0.01	0.8 ± 1.60	0.74 ± 0.05	5.48 ± 0.46
	QAOA+	313.1 ± 38.66	1.00 ± 0.00	0.0 ± 0.00	0.72 ± 0.02	5.54 ± 0.20
	maQAOA	631.1 ± 73.89	1.00 ± 0.00	0.0 ± 0.00	0.72 ± 0.02	5.57 ± 0.14
	Linear	178.3 ± 42.44	0.96 ± 0.02	2.1 ± 1.51	0.78 ± 0.03	5.60 ± 0.59
random 0.8	SA	*N.A.*	1.00 ± 0.00	0.0 ± 0.00	*N.A.*	*N.A.*
	SD	*N.A.*	0.96 ± 0.01	4.8 ± 0.98	*N.A.*	*N.A.*
	TS	*N.A.*	1.00 ± 0.00	0.0 ± 0.00	*N.A.*	*N.A.*
	QAOA1	32.8 ± 3.40	1.00 ± 0.00	0.0 ± 0.00	0.85 ± 0.02	5.14 ± 0.17
	QAOA2	51.0 ± 6.00	1.00 ± 0.00	0.0 ± 0.00	0.85 ± 0.01	5.18 ± 0.18
	QAOA+	297.4 ± 32.91	1.00 ± 0.00	0.0 ± 0.00	0.85 ± 0.02	5.08 ± 0.15
	maQAOA	935.6 ± 63.00	1.00 ± 0.00	0.0 ± 0.00	0.83 ± 0.01	5.13 ± 0.03
	Linear	181.5 ± 13.91	0.99 ± 0.01	0.6 ± 1.28	0.91 ± 0.03	4.05 ± 0.80

Table 2: Algorithms performance on Maximum Cut. Comparison of different algorithms on various graph topologies. The Solution ratio and the Solution Hamming distance are referred to the solution, whereas the approximation ratio and the mean Hamming distance are related to the estimated probability given by the statevector at the end of the circuit.

Furthermore, the approximation ratios of the different algorithms are similar when applied to problems based on 3-regular and cycle graphs. This might be because they are both regular graphs. Indeed, a cycle graph is a 2-regular graph. Among the algorithms we analyze, the Linear Circuit has the highest approximation ratio on these two graphs, followed by QAOA with depth $p = 2$. The Linear Circuit also has a higher approximation ratio than the other algorithms on the 2D grid.
It is interesting to see the behaviour of the various algorithms with random topologies. QAOA builds a circuit with many gates replicating the graph connectivity, while our circuit has sparse connections. Since the QAOA circuit has a connectivity that replicates the connectivity of the graph associated with the problem being solved, one might assume that QAOA would outperform the Linear Circuit on a densely connected graph since QAOA employs many more gates. However, our findings reveal the contrary: Linear Circuit has comparable approximation ratios with QAOA and the other algorithms when the problem instances are random graphs with edge probability 0.2 and 0.5. When the probability of having an edge is 0.8, Linear Circuit has a higher approximation ratio than all the other algorithms. This emphasizes that an ansatz has the potential to outperform other approaches even if its qubit connectivity does not reflect the graph connectivity.

7.3.2. Circuit resources

We now analyze scaling of the depth of the circuits, the number of single-qubit gates, $CNOT$ and parameters for the different quantum algorithms.

Algorithm	#Single gates	#CNOT	Max depth	#Parameters
Linear	$2n-2$	$2n-2$	$4n-4$	$n-1$
QAOA,p=1	$E+n$	$2E$	$3n+1$	2
QAOA,p=2	$2E+2n$	$4E$	$6n+2$	4
QAOA+	$E+3n-1$	$2E+2n-2$	$6n+2$	$3+2n$
ma-QAOA	$E+n$	$2E$	$3n+1$	$E+n$

Table 3: Analysis of circuit resources. The table shows the gate usage, the maximum depth of the circuit and the number of parameters.

The counts in Table 3 are obtained after decomposing a R_{zz} in two $CNOT$ and one R_z. Moreover, it is worth observing that, since the ansatzes are topology independent, the number of gates, parameters and the depth of the Linear Circuit are valid for all the R_{yz}-connected ansatzes.
We can note that, for the Linear Circuit and the other R_{yz}-connected ansatzes, there is no dependence on the number of edges E but only on the number of vertices n. On the contrary, the other algorithms build a circuit with a number of gates which depends on the number of edges E and vertices n in the graph. Typically, the number of edges of a graph is superior to the number of vertices. Therefore, the number of gates, especially $CNOT$, in the R_{yz}-connected ansatzes is usually lower than in the other circuits. Limiting the use of $CNOT$ gates is beneficial as they are a common source of errors and susceptibility to decoherence in quantum circuits.
Observing Table 3 we note that QAOA with $p = 1$ and $p = 2$ have few parameters, whereas ma-QAOA has the largest number. The few degrees of freedom in the QAOA might not be enough during the classical optimization of the parameters. On the other hand, optimizing a large number of parameters is a challenging task for any optimizer, making ma-QAOA less effective. Moreover, it has been proved that ma-QAOA does not always need all its angles [20]. Determining the optimal number of parameters for a circuit, ensuring that the ansatz is able to produce the exact solution of the problem while minimizing the count of parameters, is still an open research question.

7.3.3. Suitability to other problems

In this section, we analyze the suitability of the R_{yz}-connected ansatzes to the change of problem.
Histogram 14 and 15 show that the approximation ratios achieved by the Linear Circuit are not higher than the ones achieved by the other algorithms, differently from what was observed with the Maximum Cut problem. The approximation ratios of the Linear Circuit on both problems are lower than those achieved by the other algorithms, except for some cases where they are similar to those achieved by the ma-QAOA. These results lead us to the conclusion that the R_{yz}-connected ansatzes are not suitable for other problems and that for solving such problems the QAOA with $p = 2$ seems to be the most well-performing method.
The inability of the Linear Circuit to achieve good solutions for Maximum Clique and Minimum Vertex Cover can be understood by analyzing how the Linear Circuit explores the solution space. Indeed, the statevector of the final state of the R_{yz}-connected ansatzes is such that the basis states $|k\rangle$ and $|\bar{k}\rangle$ (with $\bar{k}$ being the

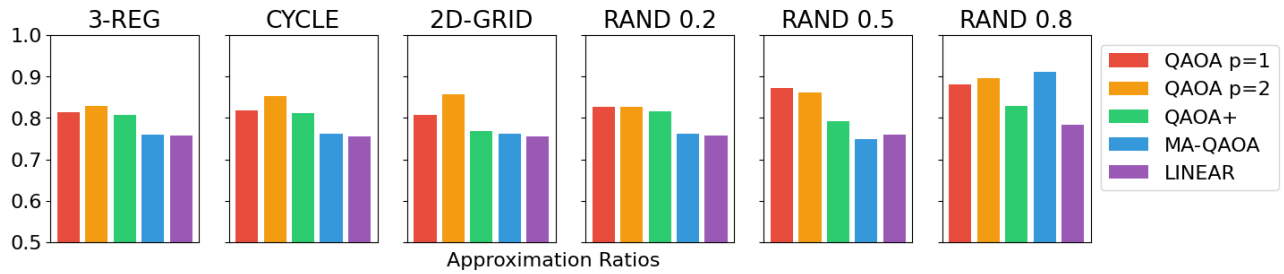


Figure 14: Approximation Ratios on Maximum Clique of different algorithms, across diverse graph topologies.

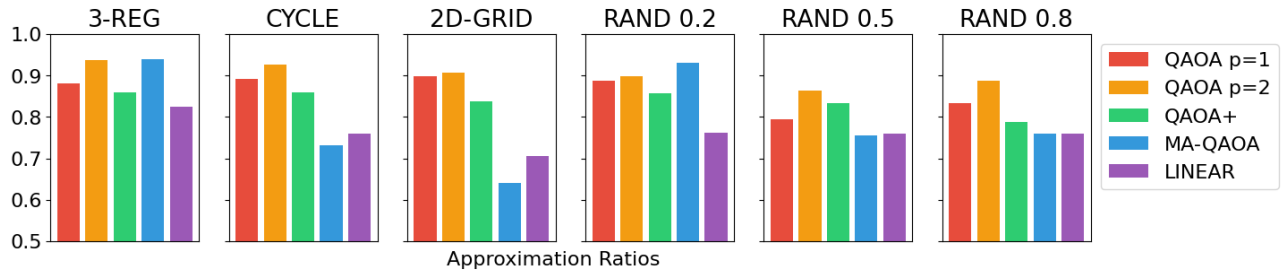
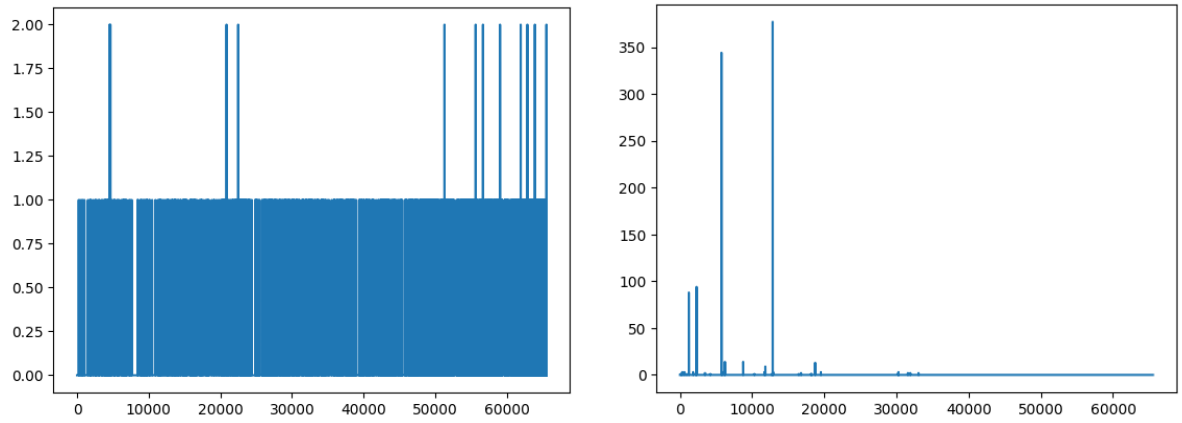


Figure 15: Approximation Ratios on Minimum Vertex Cover of different algorithms, across diverse graph topologies.

bitflip of k) always have the same amplitude (see Appendix A). However, problems like Maximum Clique and Minimum Vertex Cover do not have symmetric solutions like the Maximum Cut. Therefore, in this setting, it is impossible to obtain a statevector where the amplitude of the basis state corresponding to the exact solution has the highest amplitude unless its bitflip, potentially a suboptimal solution, also has the highest amplitude.

7.3.4. Solution space exploration

We can also analyze the estimate of the distribution probability of the basis states. Those are shown in Figure 16.



(a) Solution distribution of QAOA with depth $p = 1$.

(b) Solution distribution of the Linear Circuit.

Figure 16: Comparison of the solution distributions in circuits with optimal parameters. On the abscissa, the solutions are ordered from the best to the worst. The labels on the x-axis represent their index after the sorting. On the ordinate, the number of times that each solution is measured within 1000 shots is reported.

When optimising the circuit parameters in the QAOA with depth $p = 1$, we can heuristically see that after each optimization step, the 1000 measurements of the final state of the circuit lead to almost 1000 different states.

This is good for exploration purposes and is the reason why QAOA has a high probability of finding an optimal solution, at least in one of the measurements.
We can observe in Figure 16a that, even with optimal parameters, the measurement of the final state of the circuit gives numerous different solutions, whereas from Figure 16b, we note that the amplitude distribution in the Linear Circuit is more concentrated on good solutions and therefore we measure less unique solutions. Therefore, it is easier to find an optimal solution with QAOA.
Nevertheless, if the goal is to obtain a higher approximation ratio, the Linear Circuit is better, as shown in the histograms in Figure 13. This is true because in the statevector of the Linear ansatz, only few basis states that are associated with good solutions have a high amplitude. In contrast, in the QAOA, there are no basis states with a particularly high amplitude compared to the other basis states.
Furthermore, we can assert that the optimal solutions are within the state space of solutions which can be sampled from any R_{yz}-connected circuit for some parameters. Indeed, given the set

$$\Theta = \left\{\theta = (\theta_1, ..., \theta_{n-1}) |\ \theta_i \in \left\{-\frac{\pi}{2}, \frac{\pi}{2}\right\},\ i = 1, .., n-1\right\}$$

when the circuit parameter vector is an element of Θ, the statevector has amplitude 0 to all the basis states except for two of them. This pair of basis states is different for every $\theta \in \Theta$ (see Appendix A). Therefore, since $|\Theta| = 2^{n-1}$, all the pairs of solutions can be the output of our circuit.

7.3.5. Hardware implementation of the R_{yz}-connected ansatzes

Interestingly, R_{yz}-connected ansatzes are easy to implement on some real quantum devices.
In Subsection 2.2.3, we talked about the physical implementation of the gates. Decomposing the R_{yz} rotations, we can observe that the only single-qubit gates used by our family of ansatz are R_z and $R_x(-\frac{\pi}{2})$ rotations. In particular, the only parametric gates are the R_z rotations, which can be executed virtually without error.
To implement a $R_x(-\frac{\pi}{2})$ one could use three times $R_x(\frac{\pi}{2})$ (since $R_x(\frac{\pi}{2})^3 = R_x(-\frac{\pi}{2})$ except a global phase). However, a more convenient way is using one single $R_x(\frac{\pi}{2})$.
If we substitute the $R_x(-\frac{\pi}{2})$ with $R_x(\frac{\pi}{2})$, we obtain the circuit in Figure 17.

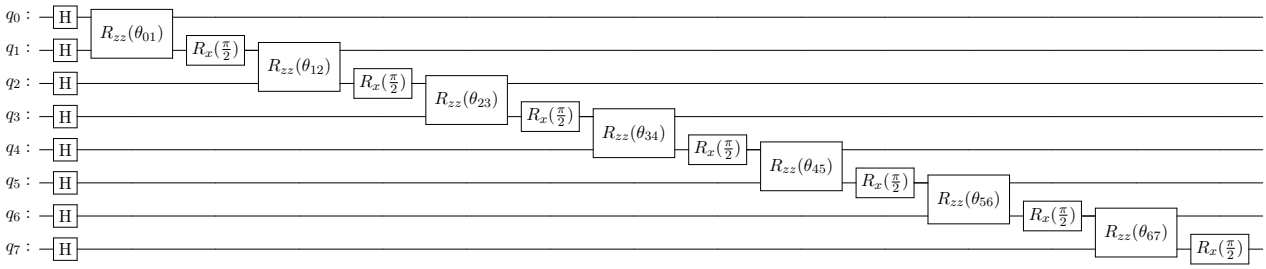


Figure 17: Linear Circuit with $R_x(\frac{\pi}{2})$.

The parametric expectation value of the circuit in Figure 17, is

$$\langle \hat{H} \rangle = \frac{1}{2^n} \sum_{b \in \{0,1\}^n} c_b \prod_{i,j \in C}^{n-1} \frac{1 + (-1)^{b_i = b_j} \sin(-\theta_{ij})}{2} \tag{21}$$

which is the same as Equation (18), just having $-\theta_{ij}$ instead of θ_{ij}. Any unconstrained classical optimizer has the same performance in optimizing $f(x)$ and $f(-x)$. Therefore, we can use $R_x(\frac{\pi}{2})$ rotations and expect the same performance as the one obtained experimentally in Subsection 7.3.1.
Since the topology of the circuit connectivity is not bound to the graph of the problem instance, like in the case of the QAOA, we can apply the two-qubit gates based on the hardware topology. In this way, we are able to find a R_{yz}-connected ansatz that can be used directly, without the need for swap gates or other gates to match the hardware topology. This is particularly useful when mapping ansatzes with dense connectivity. Indeed, as seen in Subsection 2.2, the hardware topology is usually a sparse graph, and mapping a circuit with dense connectivity to such physical topology returns a physical circuit much deeper than the logical one.

8. Conclusion

In conclusion, we implemented a PPO-based Reinforcement Learning agent that was able to build variational quantum circuits. The agent has shown the ability to learn and generate effective ansatzes. In particular,

we were able to derive the R_{yz}-connected family based on the circuits that the Reinforcement Learning agent created when solving the Maximum Cut problem. Regarding the R_{yz}-connected ansatzes, we have empirically proved their capability of obtaining a high approximation ratio when solving the Maximum Cut on different graph topologies. We have also seen their straightforward hardware implementability and capability to adapt to any physical qubit connectivity.
In future works, this Reinforcement Learning agent may be tested on various other problems and with alternative settings. Moreover, different action sets could be used during the training, with the goal of creating brand new circuits. Furthermore, according to the specific needs, hyperparameters like the depth penalty β, could be modified to obtain ansatzes with the desired properties.
Finally, we remark that there is a need for careful examination of the ansatzes that are built by the agent: a detailed analysis of the circuits should be conducted in order to verify the effective exploitation of the quantum mechanical properties, such as entanglement or superposition.
Overall, we have shown that Reinforcement Learning is a promising avenue for developing ansatzes to address challenging problems.

Acknowledgments

We acknowledge the financial support from ICSC - "National Research Centre in High Performance Computing, Big Data and Quantum Computing", funded by European Union – NextGenerationEU.

References

[1] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. Nature Reviews Physics, 3(9):625–644, August 2021.

[2] John Preskill. Quantum computing in the nisq era and beyond. Quantum, 2:79, August 2018.

[3] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[4] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature, 549(7671):242–246, September 2017.

[5] Gloria Turati, Maurizio Ferrari Dacrema, and Paolo Cremonesi. Benchmarking adaptative variational quantum algorithms on qubo instances, 2023.

[6] Mithuna Yoganathan and Chris Cade. The one clean qubit model without entanglement is classically simulable, 2019.

[7] David C. McKay, Christopher J. Wood, Sarah Sheldon, Jerry M. Chow, and Jay M. Gambetta. Efficient z-gates for quantum computing. Physical Review A, 96(2), August 2017.

[8] Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using qubo models, 2019.

[9] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. Nature Communications, 9(1), November 2018.

[10] Sergey Bravyi, David Gosset, and Robert König. Quantum advantage with shallow circuits. Science, 362(6412):308–311, October 2018.

[11] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. Quantum, 3:214, December 2019.

[12] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. Nature Communications, 5(1), July 2014.

[13] Rebekah Herrman, Phillip C. Lotshaw, James Ostrowski, Travis S. Humble, and George Siopsis. Multi-angle quantum approximate optimization algorithm, 2021.

[14] Michelle Chalupnik, Hans Melo, Yuri Alexeev, and Alexey Galda. Augmenting qaoa ansatz with multiparameter problem-independent layer, 2022.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[16] Matteo M. Wauters, Emanuele Panizon, Glen B. Mbeng, and Giuseppe E. Santoro. Reinforcement-learning-assisted quantum optimization. Physical Review Research, 2(3), September 2020.

[17] M. J. D. Powell. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation, pages 51–67. Springer Netherlands, Dordrecht, 1994.

[18] E. Farhi, J. Goldstone, S. Gutmann, and H. Neven. Quantum algorithms for fixed qubit architectures, 2017.

[19] Linghua Zhu, Ho Lun Tang, George S. Barron, F. A. Calderon-Vargas, Nicholas J. Mayhall, Edwin Barnes, and Sophia E. Economou. An adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer, 2022.

[20] Kaiyan Shi, Rebekah Herrman, Ruslan Shaydulin, Shouvanik Chakrabarti, Marco Pistoia, and Jeffrey Larson. Multiangle qaoa does not always need all its angles. In 2022 IEEE/ACM 7th Symposium on Edge Computing (SEC). IEEE, December 2022.

A. Parametric statevector

Here, we explicitly compute the expression of the parametric statevector at the end of a generic element of the R_{yz}-connected family and of the circuit without entanglement in Figure 11. We use the little-endian notation: the least significant qubit is at the top of the circuit.

Parametric statevector of R_{yz}-connected In order to compute the expression of the parametric statevector at the end of a generic element of the R_{yz}-connected family, firstly, we recall the ket-bra decomposition of $R_{zz}(\theta)$ and $R_x(-\frac{\pi}{2})$:

$$R_{zz}(\theta) = e^{-i\frac{\theta}{2}}|00\rangle\langle 00| + e^{i\frac{\theta}{2}}|01\rangle\langle 01| + e^{i\frac{\theta}{2}}|10\rangle\langle 10| + e^{-i\frac{\theta}{2}}|11\rangle\langle 11|$$

$$R_x\left(-\frac{\pi}{2}\right) = \frac{(|0\rangle + i|1\rangle)}{\sqrt{2}}\langle 0| + \frac{(i|0\rangle + |1\rangle)}{\sqrt{2}}\langle 1|$$

The initial state of the circuit is

$$|\psi_0\rangle = |0\rangle^{\otimes n}$$

By applying the layer of Hadamard gates, we obtain

$$|\psi_1\rangle = H^{\otimes n}|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{b\in\{0,1\}^n} |b\rangle$$

Then, the application of a R_{zz} gate to qubits j and k gives

$$|\psi_2\rangle = R_{zz}^{jk}(\theta_{jk})|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \left(\sum_{b\in\{\{0,1\}^n | b_j = b_k\}} e^{-i\frac{\theta_{jk}}{2}}|b\rangle + \sum_{b\in\{\{0,1\}^n | b_j \neq b_k\}} e^{i\frac{\theta_{jk}}{2}}|b\rangle \right)$$

By applying the $R_x^k(-\frac{\pi}{2})$ we obtain

$$\begin{aligned} |\psi_3\rangle &= R_x^k\left(-\frac{\pi}{2}\right)|\psi_2\rangle = \\ &= \frac{1}{\sqrt{2^n}}\frac{1}{\sqrt{2}} \left(\left(e^{-i\frac{\theta_{jk}}{2}} + ie^{i\frac{\theta_{jk}}{2}} \right) \sum_{b\in\{\{0,1\}^n | b_j = b_k\}} |b\rangle + \left(ie^{-i\frac{\theta_{jk}}{2}} + e^{i\frac{\theta_{jk}}{2}} \right) \sum_{b\in\{\{0,1\}^n | b_j \neq b_k\}} |b\rangle \right) \end{aligned}$$

Now, we can rewrite the sum of the complex exponentials in the above equation as follows:

$$
\begin{aligned}
e^{-i\frac{\theta}{2}} + ie^{i\frac{\theta}{2}} &= \frac{1}{\sqrt{2}}\left(\cos\left(\frac{\theta}{2}\right) - i\sin\left(\frac{\theta}{2}\right) + i\cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)\right) \\
&= \frac{1}{\sqrt{2}}\left(\left(\cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)\right) + i\left(\cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)\right)\right) \\
&= \frac{1}{\sqrt{2}}\left(\cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right)\right) + i\left(\cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right)\right) \\
&= \frac{1+i}{\sqrt{2}}\cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right)
\end{aligned}
$$

With an analogous computation, we can obtain also

$$
ie^{-i\frac{\theta}{2}} + e^{i\frac{\theta}{2}} = \frac{1+i}{\sqrt{2}}\cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right)
$$

Therefore, we get

$$
|\psi_3\rangle = \frac{1}{\sqrt{2^n}}\frac{1}{\sqrt{2}}\frac{1+i}{\sqrt{2}}\left(\cos\left(\frac{\theta_{jk}}{2} + \frac{\pi}{4}\right)\sum_{b\in\{\{0,1\}^n | b_j = b_k\}}|b\rangle + \cos\left(\frac{\theta_{jk}}{2} - \frac{\pi}{4}\right)\sum_{b\in\{\{0,1\}^n | b_j = b_k\}}|b\rangle\right)
$$

Furthermore, by applying another R_{zy} we obtain

$$
\begin{aligned}
|\psi_4\rangle &= R_{zy}^{kl}(\theta_{kl})|\psi_3\rangle \\
&= \frac{1}{\sqrt{2^n}}\frac{1}{\sqrt{2^2}}\left(\frac{1+i}{\sqrt{2}}\right)^2 \cdot \\
&\left(\cos\left(\frac{\theta_{jk}}{2} + \frac{\pi}{4}\right)\cos\left(\frac{\theta_{kl}}{2} + \frac{\pi}{4}\right)\sum_{b\in\{\{0,1\}^n | b_j = b_k \wedge b_k = b_l\}}|b\rangle + \right. \\
&\cos\left(\frac{\theta_{jk}}{2} + \frac{\pi}{4}\right)\cos\left(\frac{\theta_{kl}}{2} - \frac{\pi}{4}\right)\sum_{b\in\{\{0,1\}^n | b_j = b_k \wedge b_k \neq b_l\}}|b\rangle + \\
&\cos\left(\frac{\theta_{jk}}{2} - \frac{\pi}{4}\right)\cos\left(\frac{\theta_{kl}}{2} + \frac{\pi}{4}\right)\sum_{b\in\{\{0,1\}^n | b_j \neq b_k \wedge b_k = b_l\}}|b\rangle + \\
&\left.\cos\left(\frac{\theta_{jk}}{2} - \frac{\pi}{4}\right)\cos\left(\frac{\theta_{kl}}{2} - \frac{\pi}{4}\right)\sum_{b\in\{\{0,1\}^n | b_j \neq b_k \wedge b_k \neq b_l\}}|b\rangle\right)
\end{aligned}
$$

Finally, by applying all the other R_{yz} rotations used in the R_{yz}-connected, we obtain

$$
\begin{aligned}
|\psi\rangle &= \frac{1}{\sqrt{2^n}}\frac{1}{\sqrt{2^{n-1}}}\left(\frac{1+i}{\sqrt{2}}\right)^{n-1}\sum_{b\in\{0,1\}^n}\prod_{(j,k)\in C}\cos\left(\frac{\theta_{jk}}{2} + (-1)^{b_j\neq b_k}\frac{\pi}{4}\right)|b\rangle \\
&= \frac{1}{\sqrt{2}}\left(\frac{1+i}{\sqrt{2}}\right)^{n-1}\sum_{b\in\{0,1\}^n}\prod_{(j,k)\in C}\cos\left(\frac{\theta_{jk}}{2} + (-1)^{b_j\neq b_k}\frac{\pi}{4}\right)|b\rangle
\end{aligned}
\tag{22}
$$

with C being the set of the pairs of qubits to which a double rotation is applied.
We can see that the amplitudes of the statevector depend only on $(-1)^{b_j\neq b_k}$, therefore $|b\rangle$ and $|\bar{b}\rangle$ (with $\bar{b}$ being the bitflip of $\bar{b}$) have the same amplitude.

Parametric statevector of the circuit without entanglment To calculate the parametric statevector of the circuit without entanglement in Figure 11, we firstly need to apply a layer of $n-1$ Hadamard gates, as already done in the previous computation.
Then, recalling the ket-bra decomposition of $R_y(\theta)$

$$
R_y(\theta) = \left(\cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)|1\rangle\right)\langle 0| + \left(-\sin\left(\frac{\theta}{2}\right)|0\rangle + \cos\left(\frac{\theta}{2}\right)|1\rangle\right)\langle 1|
$$

we find that the application of $R_y^j(\theta)$ to the generic ket $|b\rangle$ acts in this way:

- if $|b\rangle$ has $b_j = 0$, it is mapped to
$$\cos\left(\frac{\theta_j}{2}\right)|b\rangle + \sin\left(\frac{\theta_j}{2}\right)|\tilde{b}^j\rangle$$
where $\tilde{b}^j$ is the same as b with the j^{th} bit flipped
- if $|b\rangle$ has $b_j = 1$, it is mapped to
$$\cos\left(\frac{\theta_j}{2}\right)|b\rangle - \sin\left(\frac{\theta_j}{2}\right)|\tilde{b}^j\rangle$$

Applying $R_y^j(\theta)$, we obtain the sum of all the aforementioned mapping. Calling α_b the amplitude of $|b\rangle$ before the application of $R_y^j(\theta)$, the new amplitudes of kets are:

- if $|b\rangle$ has $b_j = 0$, its amplitude is now $\alpha_b \cdot \left(\cos\left(\frac{\theta_j}{2}\right) - \sin\left(\frac{\theta_j}{2}\right)\right)$
- if $|b\rangle$ has $b_j = 1$, its amplitude is now $\alpha_b \cdot \left(\cos\left(\frac{\theta_j}{2}\right) + \sin\left(\frac{\theta_j}{2}\right)\right)$

If we apply all the $n-1$ R_y rotation, we obtain the following statevector:

$$|\psi\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{b\in\{0,1\}^{n-1}} \prod_{j=1}^{n-1} \left(\cos\left(\frac{\theta_j}{2}\right) + (-1)^{1-b_i}\sin\left(\frac{\theta_j}{2}\right)\right)(|b\rangle \otimes |0\rangle) \tag{23}$$

B. Hamiltonian expectation value

In this appendix, we calculate the parametric expression of the expectation of the Hamiltonian on the final state of a generic R_{yz}-connected ansatz (see the example in Figure 8) and of the circuit without entanglement (Figure 11).

Defining c_b as the eigenvalue associated to the $|b\rangle$ eigenvector of the Hamiltonian $\hat{H}$, and considering that the final state $|\psi\rangle$ of the generic R_{yz}-connected circuit is given by (22), the expectation value of the Hamiltonian on $|\psi\rangle$ is given by

$$\begin{aligned}
\langle\psi|H|\psi\rangle = &\left(\frac{1}{\sqrt{2}}\left(\frac{1-i}{\sqrt{2}}\right)^{n-1} \sum_{b\in\{0,1\}^n} \prod_{(i,j)\in C} \cos\left(\frac{\theta_{ij}}{2} + (-1)^{b_i\neq b_j}\frac{\pi}{4}\right)\right)\cdot \\
&\langle b|H\left(\frac{1}{\sqrt{2}}\left(\frac{1+i}{\sqrt{2}}\right)^{n-1} \sum_{b\in\{0,1\}^n} \prod_{(i,j)\in C} \cos\left(\frac{\theta_{ij}}{2} + (-1)^{b_i\neq b_j}\frac{\pi}{4}\right)\right)|b\rangle \\
= &\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\left(\frac{1-i}{\sqrt{2}}\right)^{n-1}\left(\frac{1+i}{\sqrt{2}}\right)^{n-1} \sum_{b\in\{0,1\}^n} c_b \prod_{(i,j)\in C}\left(\cos\left(\frac{\theta_{ij}}{2} + (-1)^{b_i\neq b_j}\frac{\pi}{4}\right)\right)^2
\end{aligned}$$

Thus,

$$\langle\psi|H|\psi\rangle = \frac{1}{2}\sum_{b\in\{0,1\}^n} c_b \prod_{(i,j)\in C}\left(\cos\left(\frac{\theta_{ij}}{2} + (-1)^{b_i\neq b_j}\frac{\pi}{4}\right)\right)^2 \tag{24}$$

On the other hand, the expected value of the Hamiltonian given by the circuit without entanglement in Figure 11 is

$$\begin{aligned}
\langle\psi|H|\psi\rangle = &\frac{1}{\sqrt{2^{n-1}}} \sum_{b\in\{0,1\}^{n-1}} \prod_{j=1}^{n-1}\left(\cos\left(\frac{\theta_j}{2}\right) + (-1)^{1-b_i}\sin\left(\frac{\theta_j}{2}\right)\right)\cdot \\
&\left(\langle b|\otimes\langle 0|\right) H \frac{1}{\sqrt{2^{n-1}}} \sum_{b\in\{0,1\}^{n-1}} \prod_{j=1}^{n-1}\left(\cos\left(\frac{\theta_j}{2}\right) + (-1)^{1-b_i}\sin\left(\frac{\theta_j}{2}\right)\right)(|b\rangle\otimes|0\rangle) \\
= &\frac{1}{2^{n-1}} \sum_{b\in\{0,1\}^{n-1}} c_{2b} \prod_{j=1}^{n-1}\left(\cos\left(\frac{\theta_j}{2}\right) + (-1)^{1-b_i}\sin\left(\frac{\theta_j}{2}\right)\right)^2
\end{aligned}$$

Given the trigonometric identities

$$\cos(x) \pm \sin(x) = \sqrt{2}\cos\left(x \mp \frac{\pi}{4}\right) \tag{25}$$

We can rewrite the last expected value as

$$\langle\psi|H|\psi\rangle = \sum_{b\in\{0,1\}^{n-1}} c_{2b} \prod_{j=1}^{n-1}\left(\cos\left(\frac{\theta_j}{2} + (-1)^{b_i}\frac{\pi}{4}\right)\right)^2 \tag{26}$$

C. Use of entanglement of R_{yz}-connected circuits

In Subsection 7.2.3 we have shown that the circuit in Figure 11 has the same parametric expectation value of the Hamiltonian as the circuit in Figure 9 and therefore the circuit is equivalent to a circuit without entanglement. This means that the circuit in Figure 9 does not use entanglement effectively. Here we show that every other circuit of the R_{yz}-connected family use entanglement effectively, as stated by the following theorem.

Theorem C.1. *All the R_{yz}-connected ansatzes use entanglement effectively when solving the Maximum Cut problem, except for the circuits with the same structure as the circuit in Figure 9.*

Proof. Assume a circuit without entanglement with a parametric Hamiltonian equal to the one of a R_{yz}-connected circuit exists.
The circuit without entanglement must have $n-1$ parameters. Therefore, there must be a qubit without a parametric gate.
Since the Maximum Cut has the propriety that the cost on each solution is equal to the cost on its bitflip, we can just set a qubit to 0. We leave the first qubit without gates, without loss of generality. This will lead to the conclusion that such a circuit can only substitute the R_{yz}-connected ansatz where the first qubit is connected to the others. The reasoning is analogous for the other cases.
Regarding the non-parametric gates, we know that when the parameters are all equal to 0, the circuit is equivalent to a layer of Hadamard gates, so all the non-parametric gates must be a layer of Hadamard gates (or an equivalent composition of gates). Moreover, there cannot be any non-parametric R_x or R_y before the Hadamard gates or any non-parametric R_y after the Hadamard gates because they would change the distribution of the states. However, there might be R_z anywhere or R_x after the Hadamard gates.
Regarding the parametric gates, applying a $R_z(\theta)$ cannot influence the outcome of the measurements, therefore it cannot be used as a parametric gate.
We know that for $\theta_{ij} = \frac{\pi}{2}$ and $\theta_{ij} = -\frac{\pi}{2}$ Equation (22) has qubits i and j in the form $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ or $\frac{|01\rangle+|10\rangle}{\sqrt{2}}$. To replicate this, in the non entangling circuit we must use rotation such that $R_a^j\left(\frac{\pi}{2}\right)$ and $R_a^j\left(-\frac{\pi}{2}\right)$ lead the qubit j to the states $|0\rangle$ and $|1\rangle$. If after the Hadamard gate a rotation $R_z(\phi)$ has been applied, the only rotation that we can use to achieve the states $|0\rangle$ and $|1\rangle$ is the rotation around the axis obtained by rotating X of $\phi + \frac{\pi}{2}$ radians around the XY plane. Without loss of generality, we assume there is no rotation around Z after the Hadamard gates and proceed using R_y rotations.
We call the parameters of the circuit without entanglement θ's.
To obtain the same expectation value, we need to find θ's so that the probability of measuring any state $|b\rangle \otimes |0\rangle$ is twice in the circuit without entanglement than in the entangling one. This is needed because Maximum Cut has symmetric solutions. In the non entangled circuit, since the first qubit is fixed to 0, we can measure half of the solutions. To make the expectation equal, each solution in the non entangled circuit needs to have double the probability.
Therefore, for the assumption at the start of the proof to be true, we need

$$P_{no\ ent}(|b\rangle \otimes |0\rangle) = P_{ent}(|b\rangle \otimes |0\rangle) + P_{ent}(|\bar{b}\rangle \otimes |1\rangle)$$

with $\bar{b}$ being the bitfilp of b, must be valid for all $b \in \{0,1\}^{n-1}$.
We know that in the circuit without entanglement $P_{no\ ent}(|0\rangle^{\otimes n})$ (i.e. the probability of measuring $|0\rangle^{\otimes n}$) is

$$|(\langle 0|)^{\otimes n}|\psi\rangle|^2 = \left(\frac{1}{\sqrt{2^{n-1}}}\prod_{j=1}^{n-1}\left(\cos\left(\frac{\theta'_j}{2}\right) - \sin\left(\frac{\theta'_j}{2}\right)\right)\right)^2$$

with $|\psi\rangle$ obtained in Equation (23).
In the R_{yz}-connected ansatzes, independently from the circuit connectivity, using $|\psi\rangle$ obtained in Equation (22),

$P_{ent}(|0\rangle^{\otimes n}) + P_{ent}(|1\rangle^{\otimes n})$ is

$$
\begin{aligned}
&|(\langle 0|)^{\otimes n}|\psi\rangle|^2 + |(\langle 1|)^{\otimes n}|\psi\rangle|^2 = \\
&= \left(\frac{1}{\sqrt{2}} \prod_{(j,k)\in C} \cos\left(\frac{\theta_{jk}}{2} + \frac{\pi}{4}\right)\right)^2 + \left(\frac{1}{\sqrt{2}} \prod_{(j,k)\in C} \cos\left(\frac{\theta_{jk}}{2} + \frac{\pi}{4}\right)\right)^2 \\
&= 2 \cdot \left(\frac{1}{\sqrt{2}} \prod_{(j,k)\in C} \cos\left(\frac{\theta_{jk}}{2} + \frac{\pi}{4}\right)\right)^2 \\
&= 2 \cdot \left(\frac{1}{\sqrt{2^n}} \prod_{(j,k)\in C} \left(\cos\left(\frac{\theta_{jk}}{2}\right) - \sin\left(\frac{\theta_{jk}}{2}\right)\right)\right)^2 \\
&= \left(\frac{\sqrt{2}}{\sqrt{2^n}} \prod_{(j,k)\in C} \left(\cos\left(\frac{\theta_{jk}}{2}\right) - \sin\left(\frac{\theta_{jk}}{2}\right)\right)\right)^2 \\
&= \left(\frac{1}{\sqrt{2^{n-1}}} \prod_{(j,k)\in C} \left(\cos\left(\frac{\theta_{jk}}{2}\right) - \sin\left(\frac{\theta_{jk}}{2}\right)\right)\right)^2
\end{aligned}
$$

Given the identical formulation of the two probabilities, it is immediate to see that

$$P_{no\ ent}(|0\rangle^{\otimes n}) = P_{ent}(|0\rangle^{\otimes n}) + P_{ent}(|1\rangle^{\otimes n}) \tag{27}$$

only when all θ's are equal to θs
Moreover, we note that in the circuit without entanglement $P_{no\ ent}(|1\rangle^{\otimes n-1} \otimes |0\rangle)$ is

$$|((\langle 0|)^{\otimes n} \otimes \langle 1|)|\psi\rangle|^2 = \left(\frac{1}{\sqrt{2^{n-1}}} \prod_{j=1}^{n-1} \left(\cos\left(\frac{\theta'_j}{2}\right) + \sin\left(\frac{\theta'_j}{2}\right)\right)\right)^2$$

whereas in the R_{yz}-connected ansatz, $P_{ent}(|1\rangle^{\otimes n-1} \otimes |0\rangle) + P_{ent}(|0\rangle^{\otimes n-1} \otimes |1\rangle)$ is equal to

$$
\begin{aligned}
&|((\langle 0|)^{\otimes n} \otimes \langle 1|)|\psi\rangle|^2 + |((\langle 1|)^{\otimes n} \otimes \langle 0|)|\psi\rangle|^2 \\
&= \left(\frac{1}{\sqrt{2}} \prod_{(j,k)\in C} \cos\left(\frac{\theta_{jk}}{2} + (-1)^{b_j \neq b_k} \frac{\pi}{4}\right)\right)^2 + \left(\frac{1}{\sqrt{2}} \prod_{(j,k)\in C} \cos\left(\frac{\theta_{jk}}{2} + (-1)^{b_j \neq b_k} \frac{\pi}{4}\right)\right)^2 \\
&= 2 \cdot \left(\frac{1}{\sqrt{2}} \prod_{(j,k)\in C} \cos\left(\frac{\theta_{jk}}{2} + (-1)^{b_j \neq b_k} \frac{\pi}{4}\right)\right)^2 \\
&= 2 \cdot \left(\frac{1}{\sqrt{2^n}} \prod_{(j,k)\in C} \cos\left(\frac{\theta_{jk}}{2}\right) + (-1)^{b_j = b_k} \sin\left(\frac{\theta_{jk}}{2}\right)\right)^2 \\
&= \left(\frac{\sqrt{2}}{\sqrt{2^n}} \prod_{(j,k)\in C} \cos\left(\frac{\theta_{jk}}{2}\right) + (-1)^{b_j = b_k} \sin\left(\frac{\theta_{jk}}{2}\right)\right)^2 \\
&= \left(\frac{1}{\sqrt{2^{n-1}}} \prod_{(j,k)\in C} \cos\left(\frac{\theta_{jk}}{2}\right) + (-1)^{b_j = b_k} \sin\left(\frac{\theta_{jk}}{2}\right)\right)^2
\end{aligned}
$$

where b_j is the j^{th} digit of $b = 1...10$ or $b = 0...01$.
For

$$P_{no\ ent}(|1\rangle^{\otimes n-1} \otimes |0\rangle) = P_{ent}(|1\rangle^{\otimes n-1} \otimes |0\rangle) + P_{ent}(|0\rangle^{\otimes n-1} \otimes |1\rangle) \tag{28}$$

to be true with the θ's equivalent to the θs (to ensure Equation (27)), the only case is when $(-1)^{b_j \neq b_k}$ is always -1, which corresponds the case when $j = 0$ and $k = 1, ..., n-1$. In all the other cases, this is not true.
If we try to modify the θ's or we try to add an R_x rotation after the R_y in order to make Equation (28) true, then Equation (27) will inevitably become false.
Therefore, since Equations (27) and (28) cannot be both true (except for the highlighted case), the thesis is proven by contradiction.

Abstract in lingua italiana

Gli algoritmi quantistici variazionali [1] si sono affermati come uno strumento promettente per la risoluzione di problemi complessi su computer quantistici. Questi algoritmi ottimizzano iterativamente i parametri dei circuiti quantistici in modo tale da diminuire il valore attesto dell'energia di un sistema quantistico. In questa tesi, progettiamo e implementiamo un agente di Apprendimento per Rinforzo capace di generare circuiti quantistici variazionali autonomamente. Una delle difficoltà di questi algoritmi è l'identificazione di un circuito adatto per la risoluzione di uno specifico problema. Il contesto di Apprendimento per Rinforzo permette una maggiore adattabilità, con la potenzialità di scoprire nuovi circuiti non convenzionali in grado di performare meglio dei circuiti esistenti. L'agente è allenato su diverse istanze, imparando a costruire circuiti per la risoluzione dei problemi del Taglio Massimo, della Cricca Massima e della Copertura Minima dei Vertici su diverse topologie di grafi con 8 e 14 vertici. I circuiti costruiti dall'agente vengono analizzati osservandone l'utilizzo dei gate, la profondità e le prestazioni. Inoltre, esaminando la struttura comune dei circuiti identificati dall'agente allenato sul problema del Taglio Massimo, siamo stati in grado di dedurre una nuova famiglia di ansatz che ottiene un alto rapporto di approssimazione, che denominiamo famiglia R_{yz}-connessa. In particolare, selezioniamo uno specifico circuito R_{yz}-connesso, denominato Circuito Lineare, e testiamo le sue prestazioni su istanze di problemi su grafi di dimensione fino a 16 vertici. I risultati sono confrontati con algoritmi quantistici dello stato dell'arte come il QAOA e le sue varianti e con algoritmi classici, mostrando che il Circuito Lineare ottiene consistentemente un valore atteso dell'energia più basso rispetto agli altri algoritmi. L'uso dell'entanglement e l'efficacia degli ansatz R_{yz}-connessi possono essere spiegati considerando l'espressione parametrica del valore attesto dell'Hamiltoniana e come lo spazio di soluzioni è esplorato. Inoltre, viene mostrata l'immediata implementabilità hardware degli ansatz R_{yz}-connessi. Nel complesso, mostriamo che il Apprendimento per Rinforzo è una direzione promettente per la costruzione di ansatz con l'obiettivo di affrontare una vasta quantità di problemi che sono difficili da risolvere con i computer classici.

Parole chiave: Algoritmi Quantistici Variazionali, Computer Quantistici, Apprendimento per Rinforzo, Ottimizzazione