POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

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

Laurea Magistrale in High Performance Computing Engineering

**Author:** Simone Foderà

**Advisor:** Prof. Maurizio Ferrari Dacrema

**Co-advisor:** Gloria Turati

**Academic year:** 2023-2024

## 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.

In this work, we implement a Reinforcement Learning agent that is able to build variational quantum circuits to solve QUBO problems by minimizing the expected value of the associated Ising Hamiltonian. We train the agent on Maximum Cut, Maximum Clique, and Minimum Vertex Cover problem instances and analyze the circuits obtained. During the training for solving the Maximum Cut problem, the agent was able to build circuits with a high approximation ratio and a shared common structure. Therefore, we define and analyze a new family of ansatzes with these circuits, denoted as $R_{yz}$-connected family. In particular, we test a specific $R_{yz}$-connected circuit, denoted as Linear Circuit, characterized by each qubit being connected to the following one, and compare its performance with multiple quantum and classical algorithms. The experiments are conducted on Maximum Cut, Maximum Clique and Minimum Vertex Cover problem instances with different underlying graph topologies. Finally, we show the straightforward hardware implementability of the $R_{yz}$-connected ansatzes.

## 2. Design of the agent

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.

### 2.1. 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 when measuring the final state of the circuit. The initial environment of each episode is a circuit with a single layer of Hadamard gates.

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

where $R_a$ is a rotation around the $a$ axis and $R_a^i$ denotes the $R_a$ gate applied to the qubit $i$.
The action set of the agent is $A = S \cup D$.

### 2.3. Reward

Since our goal is to minimize the expectation value of the Hamiltonian, we define the reward of the agent after each step as $-\langle \hat{H} \rangle$. We subtract to the reward a penalty $\beta \cdot depth$, with $\beta = 0.015$, with the goal of favouring shallower circuits over deeper ones. Shallower circuits are preferable because are more resilient to the decoherence of the quantum state.

### 2.4. Step Function

The step function takes the action as input, modifies the environment and returns the observation and the reward. In our specific context, a step consists of applying the action gate to the circuit and optimizing the circuit parameters using the COBYLA [3] classical optimizer. The optimizer modifies the parameters to minimize the estimate of the expectation of the problem Hamiltonian on the final quantum state of the circuit. All the hyperparameters of the classical optimization are the default ones of the scipy.optimize.minimize function.

### 2.5. Actor and Critic

The Reinforcement Learning algorithm we use is PPO [4] with actor and critic. All the hyperparameters are set using the default values of the OpenAI implementation. The actor and the critic 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. The input and output layers of the network depend on the size of the observation vector and the action sets. The input layer has $2^n$ neurons, matching the observation vector size. The output layer has size $|A|$. Both the actor and critic networks sample only a single action as output.

### 2.6. Training

Our agent is trained on three different problems: Maximum Cut, Maximum Clique and Minimum Vertex Cover. In particular, for each problem, we choose three different graph topologies: 3-regular, 2D-grid, and star. We train the agent on graphs with 8 vertices and 14 vertices.
The agent is trained for 64 epochs of at most 64 steps. An episode is long a maximum of $2 \cdot n$ steps but can end sooner using the parameter *patience*. The *patience* parameter helps to prevent the agent from spending excessive time in unproductive or challenging situations and, at the same time, allows not to truncate the episode immediately if such a scenario is reached, giving the agent a longer-term view.
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 carried out on the master thread. In our experiments, we use six threads.
At the end of the training, we analyze the circuit with the highest reward considering its approximation ratio and the gate usage.

## 3. $R_{yz}$-connected ansatzes

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 achieving a high approximation ratio, which we denote as $R_{yz}$-connected.
This family of ansatzes is composed of circuits with an initial layer of Hadamard gates, followed by $n-1$ $R_{yz}$ rotations. In particular, these rotations are applied such 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 $C_j$ associated to each $R_{yz}$-connected ansatz are connected graphs for all $j = 1, \ldots, n-1$.

### 3.1. Linear Circuit

We test a specific $R_{yz}$-connected circuit, denoted as Linear Circuit (see Figure 1), characterized by each qubit being connected to the following one, on problem instances with graphs with up to 16 vertices.

The Linear Circuit is compared with ansatzes of state-of-the-art quantum algorithms, namely QAOA [5] with depth $p = 1$, QAOA with depth $p = 2$, QAOA+, and ma-QAOA, as well as with 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 default settings.

Our analysis focuses on assessing the suitability of the Linear Circuit to different underlying graph topologies by comparing its performance with the other algorithms on the Maximum Cut problem across different graph topologies. We consider 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, we compare the Linear Circuit with the other algorithms on the Maximum Clique and Minimum Vertex Cover problems built on graphs with the same topologies to test its suitability to different problem typologies. The algorithms are tested on instances having 8, 14, and 16 vertices.

We employ the Qiskit QASM simulator with 1000 shots for all the quantum algorithms. We adopt COBYLA as classical optimizer and choose random initial parameters for initializing each experiment. The experiments for all the algorithms, including the Linear Circuit, are run ten times. We consider the solution of an experiment as the solution with the best approximation ratio found.

The metrics used to evaluate 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.

We also use the approximation ratio and the mean Hamming distance to compare the performance of the quantum algorithms.

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.

## 4. General circuits analysis

| Size | Problem | Graph | Single Gates | Double Gates | A.R. | A.R. Thresh. |
|---|---|---|---|---|---|---|
| 8 | Maximum Cut | Star | 0 | 7 | 0.99 | 0.00 |
| | | Grid | 0 | 7 | 0.99 | 0.00 |
| | | 3-Reg | 0 | 7 | 0.99 | 0.00 |
| | Maximum Clique | Star | 3 | 9 | 0.70 | 0.95 |
| | | Grid | 3 | 10 | 0.83 | 0.94 |
| | | 3-Reg | 3 | 11 | 0.75 | 0.91 |
| | Minimum Vertex Cover | Star | 3 | 4 | 0.93 | 0.69 |
| | | Grid | 5 | 3 | 0.96 | 0.73 |
| | | 3-Reg | 4 | 9 | 0.99 | 0.77 |
| 14 | Maximum Cut | Star | 0 | 7 | 0.5 | 0.00 |
| | | Grid | 0 | 12 | 0.76 | 0.00 |
| | | 3-Reg | 0 | 13 | 0.92 | 0.00 |
| | Maximum Clique | Star | 0 | 17 | 0.63 | 0.99 |
| | | Grid | 0 | 12 | 0.72 | 0.99 |
| | | 3-Reg | 1 | 13 | 0.64 | 0.98 |
| | Minimum Vertex Cover | Star | 0 | 7 | 0.65 | 0.71 |
| | | Grid | 1 | 9 | 0.87 | 0.74 |
| | | 3-Reg | 1 | 14 | 0.85 | 0.74 |

Table 1: Analysis of the general circuits found by the Reinforcement Learning agent. The first two columns represent the occurrences of single and two-qubit gates in the circuits with the highest rewards. The table also shows the approximation ratio achieved, followed by the threshold of approximation ratio that a solution must exceed to be considered feasible.

In Table 1, we summarize the gate usage and performance of the circuits found by the agent with the highest reward.

### 4.1. Maximum Cut problem

The Reinforcement Learning agent successfully generated efficient circuits when trained on Maximum Cut instances. On the 8 vertices instances, the agent built circuits that use 7 $R_{yz}$ rotations and achieve an approximation ratio close to one. On the 14 vertices instances, the agent built circuits with 13 or less $R_{yz}$ rotations, achieving a higher approximation ratio when more rotations are used. In particular, when the agent uses 13 $R_{yz}$ rotations, the circuit reaches an approximation ratio above 0.9.

### 4.2. Maximum Clique problem

For the Maximum Clique problem, the approximation ratios are not high, indeed, they range

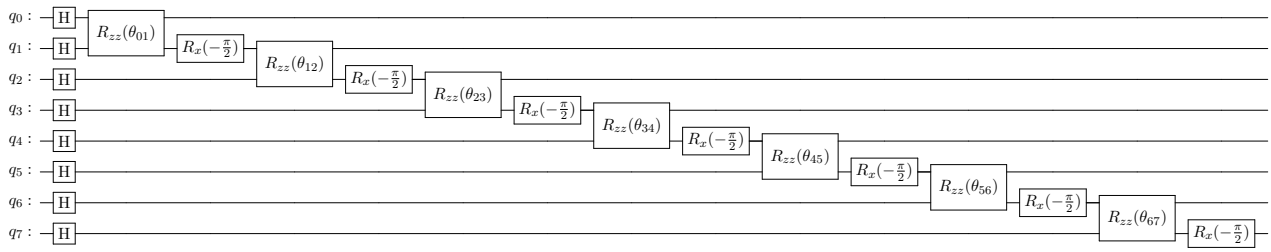



Figure 1: Linear Circuit, a specific element of the family of ansatzes found during the training on the Maximum Cut.

between 0.63 and 0.83. Moreover, in both the 8 and 14 vertices instances, the approximation ratios are lower than the thresholds. 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 ones. The high difference between the approximation ratios and the threshold means that the probability of measuring feasible solutions is low.

### 4.3. Minimum Vertex Cover problem

The approximation ratios obtained by the agent trained on solving the Minimum Vertex Cover are above 0.9 for the 8 vertices instances, but the agent created only circuits with primarily $R_y$ rotations. This type of rotation applied to different qubits allows to obtain circuits that have the exact solutions within the space of states generated by the circuit. However, optimizing their parameters takes exponential time. Therefore, it is not a scalable approach. In the 14 vertices instances, with the employment of multiple $R_{yz}$ and $R_{xy}$, the approximation ratios achieved are relatively good, which leads us to believe that using two-qubit gates is, in general, more effective than using single-qubit gates.

## 5. Linear Circuit results

In this section, we analyze the performance of the Linear Circuit (see Figure 1), as explained in Subsection 3 and show the ease of hardware implementation of the $R_{yz}$-connected ansatzes.

### 5.1. Results on Maximum Cut problem

The histograms in Figure 3 show that the Linear Circuit achieves almost always the highest approximation ratio among the various tested algorithms on the Maximum Cut problem. In particular, the approximation ratios of the different algorithms are similar when applied to problems based on 3-regular and cycle graphs. The Linear Circuit also has a higher approximation ratio than the other algorithms on the 2D-grid.
Moreover, since the QAOA circuit has a connectivity that replicates the connectivity of the graph associated to the problem being solved, one might assume that QAOA would outperform the Linear Circuit on a densely connected graph since QAOA employs a higher number of gates. However, our findings reveal the contrary: the 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, the 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.
However, the Linear Circuit is less reliable in finding a good solution for the Maximum Cut problem than the other tested algorithms. This is due to the way the solution space is explored (see Subsection 5.3).
Finally, we observe that the number of iterations the classical optimizer performs before convergence required by the Linear Circuit is superior to the one needed by QAOA with depth $p = 1$ and $p = 2$ but inferior to the one required by QAOA+ and ma-QAOA. This can be explained

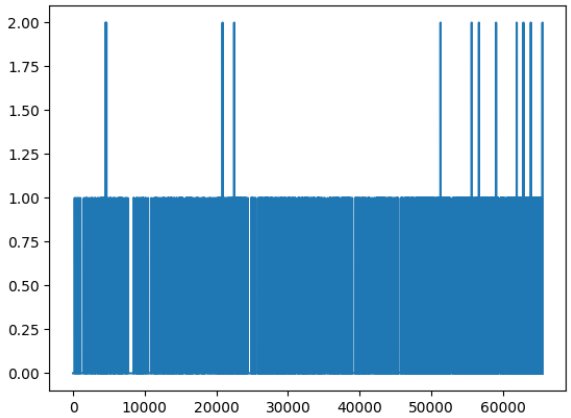
by the number of parameters used by the algorithms: as the number of parameters increases, the number of iterations required for classical optimization increases. Therefore, the Linear Circuit has less degrees of freedom than ma-QAOA and QAOA+, but converges faster.

## 5.2. Suitability to other problems

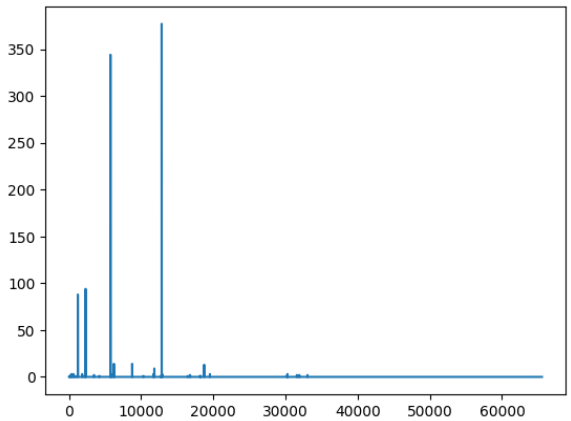
Differently from what was observed with the Maximum Cut problem, the approximation ratios achieved by the Linear Circuit are not higher than the ones achieved by the other algorithms, when tested on instances of the Maximum Clique and the Minimum Vertex Cover problems. The approximation ratios of the Linear Circuit on these problems are lower than those achieved by the other algorithms, except for a few 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 to other problems and that for solving such problems the QAOA with $p = 2$ seems to be the most well-performing method.

## 5.3. Solution space exploration

We can also analyze an estimate of the distribution probability of the basis states at the end of the QAOA with $p = 1$ and the Linear Circuit, both with optimized parameters. These distributions are shown in Figure 2.
We can heuristically see that after each optimization step in the QAOA, the measurements of the final state of the circuit lead to almost all different states. This also happens with the optimized parameters, as we can see in Figure 2a. 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.
On the other hand, by observing the distribution of probabilities for the Linear Circuit in Figure 2b, we note that the amplitude distribution is more concentrated on the good solutions and therefore fewer unique solutions are measured. This implies a lower probability of sampling the optimal solution, but allows to achieve a higher approximation ratio consistently.
Therefore, if the goal is to obtain a higher approximation ratio, the Linear Circuit is better, as shown in the histograms in Figure 3.




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




(b) Solution distribution of the Linear Circuit.

Figure 2: 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.

## 5.4. Hardware implementation

Interestingly, the Linear Circuit, and more in general the $R_{yz}$-connected ansatzes, are easy to implement on specific real quantum devices. Modern technologies are often calibrated to perform only $R_z$ and $R_x(\frac{\pi}{2})$ rotations. This is sufficient, as any arbitrary rotation can be achieved by combining $R_z$ and $R_x(\frac{\pi}{2})$ rotations, making the full set of $R_x$ rotations unnecessary.
Decomposing the $R_{yz}$ rotations, we can observe that the only single-qubit gates used by $R_{yz}$-connected ansatzes 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. Additionally, to implement

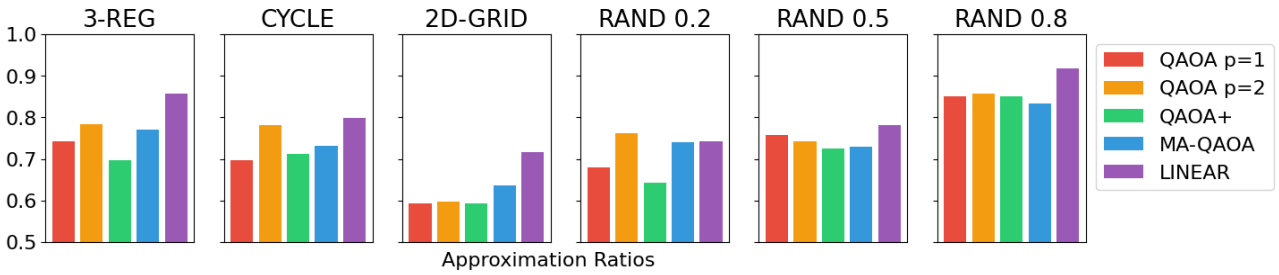



Figure 3: Approximation Ratio obtained by the tested quantum algorithms on Maximum Cut instances on graphs with 16 vertices.

the $R_x(-\frac{\pi}{2})$ rotations, we can substitute each of these gates with a $R_x(\frac{\pi}{2})$. $R_{yz}$-connected ansatzes maintain their performance after this substitution, allowing us to exploit the calibration of the hardware.
Furthermore, we know that mapping a circuit with dense connectivity to the physical topology returns a physical circuit much deeper than the logical one. Indeed, QAOA often needs multiple swap gates to be implemented on hardware with sparse connectivity. Since the topology of the connectivity of the $R_{yz}$-connected ansatzes 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 particular, we can always find a $R_{yz}$-connected ansatz that does not need swap gates and can be used directly.

## 6. Conclusion

In conclusion, we implemented a PPO-based Reinforcement Learning agent that was able to build variational quantum circuits. The agent showed the ability to learn and generate effective ansatzes. In particular, we were able to derive the family of $R_{yz}$-connected ansatzes, based on the circuits created by the Reinforcement Learning agent when solving the Maximum Cut problem. We analyzed a specific $R_{yz}$-connected ansatz, the Linear Circuit, and empirically proved its capability to obtain a high approximation ratio when solving the Maximum Cut on different graph topologies. Moreover, we showed that the Linear Circuit is not suitable for other problem typologies. Finally, we analyzed the straightforward hardware implementability of the $R_{yz}$-connected ansatzes and their capability to adapt to any physical qubit connectivity. Overall, we showed that Reinforcement Learning is a promising avenue for developing ansatzes to address challenging problems with quantum computing.

## 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] 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.

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

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