Numerical Analysis of QAOA for Financial Optimization

Nazim Turan

August 05, 2024

1 Introduction

Although practical quantum computing is still relatively new, it has been heavily explored for solving problems that are considered intractable or requiring an immense amount of time to be solved. Quantum computing can be applied in finance, either by improving current methods or providing a different perspective of solutions to existing problems. In contrast to classical computing, quantum computing holds the promise of highly efficient algorithms, providing exponential speedups for some technologically important problems. Typical problems that are currently being solved in finance include stock market prediction, portfolio optimization and fraud detection. The idea of applying quantum mechanics to finance is not a new one: some well-known financial problems can be directly expressed in a quantum-mechanical form. As an example, the Black Scholes Merton formula can be mapped to the Schrodinger equation, modeling the arbitrage relationships that led to its formulation. Even the entire financial market can be modeled as a quantum process, where quantities that are important to finance, such as the covariance matrix, emerge naturally [5]. Many problems in finance can be expressed as optimization problems. These tasks which are particularly hard for classical computers, but find a natural formulation using quantum optimization methods [17]. In recent years, this field has known a tremendous growth, partly due to the commercial availability of quantum annealers [5]. Another way to to approach financial problems is to search for patterns in past data. This is a natural way to consider economic forecasting problems, an area where machine learning methods have proved to be extremely successful. The computational cost of these approaches, however, is often prohibitive. In recent years, there has been an impressive effort to develop quantum machine learning algorithms [14], which many hope will provide the tools to satisfy our growing data requirements. Moreover, the behavior of some financial systems can be predicted by applying Monte Carlo methods. The act of sampling a distribution function can limit the speed, and hence the applicability, of the algorithm. In a series of recent papers, it was suggested this task could be done efficiently by sampling a quantum system [10, 16, 23]. Optimization problems are at the core of many financial problems, with the most obvious example being portfolio optimization. Portfolio optimization is an NP-Hard problem, which means that it is extremely difficult, if not impossible, for classical computers to efficiently determine the best choice of portfolio. This is where quantum computing can make a difference and the way to do that is by implementing quantum optimization algorithms on a quantum computer. The most common techniques for quantum optimization is Quantum Annealing (QA) and the Quantum Approximate Optimization Algorithm

(QAOA) [5]. Quantum annealing is the physical process of implementing an adiabatic quantum computation. This process is similar to classical or simulated annealing, where thermal fluctuations allow the system to jump between different local minima in the energy landscape. As the temperature is lowered, the probability of moving to a worse solution tends to zero. In quantum annealing, these jumps are driven by quantum tunneling events. This process explores the landscape of local minima more efficiently than thermal noise, especially when the energy barriers are tall and narrow [6]. In practice, it is difficult to fulfill the conditions required for adiabatic quantum computing in a quantum annealing process. It can be difficult, for instance, to guarantee that the system's evolution is fully adiabatic, or that the system is initiated in the true ground state of the initial Hamiltonian. Quantum annealing is therefore an approximate realization of adiabatic computing. It was shown by Zagoskin that approximate adiabatic computing can find a solution which is close to optimal - a problem which is also known to be NP-hard - in polynomial time [5, 24]. Both Quantum Annealing and the Quantum Approximate Optimization Algorithm search for a combination of two Hamiltonians (problem and mixer Hamiltonian) which prepares the ground state of one of the Hamiltonians as quickly and accurately as possible. QA smoothly interpolates between the two Hamiltonians, whereas QAOA applies one or the other in an alternating sequence a certain number of times p. QAOA is considered a heuristic of QA that assumes a certain precision factor. When the precision is set to be infinite, then QAOA is equivalent to QA. The adiabatic theorem guarantees that the system, if initially in the ground state of the mixer Hamiltonian and deformed sufficiently slowly, will remain in the ground state throughout. Optimization with quantum annealing can potentially give polynomial speedups over both classical algorithms and unoptimized quantum schedules. Optimization with QAOA applies the mixer and problem Hamiltonians alternately, using the timings of these pulses as variational parameters to be optimized over. There is evidence that restricted forms of QAOA are more powerful than adiabatie protocols with the same restrictions (although both are known to be quantum universal), but this does not address whether a non-adiabatic annealing procedure can outperform QAOA [9]. In this report, we are going to investigate problems in finance like securitization that can be expressed in the Quadratic Unconstraint Binary Optimization (QUBO) formulation [19, 12] and solve them with the QAOA algorithm. As benchmarks, we start by investigating the well studied problem of Max Cut for 3-regular graphs and euclidean distance based graphs with QAOA, to verify the correctness of our implementation [24]. In the following section. we provide an brief analysis of the QAOA algorithm and the Max Cut problem. Then, we continue with presentation of the mumerical simulations. Finally, we provide insight for the lessons that we learned based on this work and outlook for the future.

2 Quantum Approximate Optimization Algorithm

QAOA is believed to be a promising algorithm for proving quantum advantage during the NISQ era, which is why it has been heavily investigated in the last few years. Although QAOA was introduced in 2014 by Farhi [4], various additional proofs, modifications, bounds and graph analysis have been provided in the following years. Proofs concerning the performance of QAOA [4, 30, 3, 22, 26, 8, 13] and detailed analysis of the unique relevant subgraphs that contribute to the expectation value 20 have been provided. Modifications have been proposed such as the Quantum Alternating Operator Ansatz [29], which extends the original QAOA framework to include a larger family of unitaries.

Such an ansatz can support the representation of a larger and potentially more useful set of states than the original formulation, by mixing only within a desired subspace. Lower bounds for the Max-Cut problem for 3-regular graphs have been provided by Farhi [4] for $p = 1$ and more recently bounds for $p = 2$ and 3 have been provided in [18]. Graph analysis concerning the necessary lowest value of p in random graphs has been recently carried out in order to understand the locality requirements of QAOA in order to maximize its performance [27, 25].

One popular aspect of QAOA as an algorithm is the potential resiliency against noise that all parametrized quantum circuits exhibit. For example, parametrized quantum circuits are employed in Quantum Neural Networks and Variational Quantum Algorithms, which are typical researched algorithms that may allow for near-term quantum advantage. QAOA is a hybrid quantum-classical algorithm, in which a classical computer optimizes 2p angles parameterizing an ansatz wavefunction by querying a quantum device. This wavefunction encodes an approximate solution to some combinatorial optimization problem. For $p \to \infty$, it is known that the ansatz wavefunction encodes the exact solution, which follows from the adiabatic theorem. QAOA was introduced to solve combinatorial optimization problems that can be defined on N-bit binary strings $z = z_1, \cdots, z_N$, where the goal is to determine a string that maximizes a given classical objective function
$C(z) : \{+1, -1\}^N \to \mathbb{R}_{\geq 0}$. QAOA aims to find a string z that achieves a desired approximation ratio [2]

$$\frac{C(z)}{C_{max}} \geq r^* \tag{1}$$

where $C_{max} = \max_z C(z)$.

The problem is initially converted from a classical objective function to a quantum problem Hamiltonian by turning each binary variable z_i to a quantum spin σ_i^z:

$$H_C = C(\sigma_1^z, \sigma_2^z, \cdots, \sigma_N^z) \tag{2}$$

For p-level QAOA, we initialize the quantum system in the $|+\rangle^{\otimes N}$ state and then apply the problem Hamiltonian H_C and a mixing Hamiltonian $H_B = \sum_{j=1}^{N} \sigma_j^x$ alternately to generate a variational wavefunction

$$\left|\Psi_P(\vec{\gamma}, \vec{\beta})\right\rangle = e^{-i\beta_P H_B} e^{-i\gamma_P H_C} \cdots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |+\rangle^{\otimes N} \tag{3}$$

which is parametrized by $2p$ variational parameters γ_i and β_i $(i = 1, 2, \cdots, p)$ [2]. We then determine the expectation value H_C in this variational state

$$F_P(\vec{\gamma}, \vec{\beta}) = \left\langle \Psi_P(\vec{\gamma}, \vec{\beta}) \right| H_C \left| \Psi_P(\vec{\gamma}, \vec{\beta}) \right\rangle \tag{4}$$

which is done by repeated measurements of the quantum system in the computational basis. A classical computer is used to search for the optimal parameters $(\vec{\gamma}^*, \vec{\beta}^*)$ so as to maximize the averaged measurement output $F_P(\vec{\gamma}, \vec{\beta})$,

$$(\vec{\gamma}^*, \vec{\beta}^*) = arg \max_{\vec{\gamma},\vec{\beta}} F_P(\vec{\gamma}, \vec{\beta}) \tag{5}$$

This is typically done by starting with some initial guess of the parameters and performing simplex or gradient-based optimization. A figure of merit for benchmarking the performance of QAOA is the approximation ratio [2]

$$r = \frac{F_P(\vec{\gamma}^*, \vec{\beta}^*)}{C_{\max}} \tag{6}$$

or

$$r = \frac{F_P(\vec{\gamma}^*, \vec{\beta}^*) - C_{\max}}{C_{\max} - C_{\min}} \tag{7}$$

which is better metric since it avoids miscalculations when the $C_{\min}$ is a positive and the $C_{\max}$ is a negative number.

The framework of QAOA can be applied to general combinatorial optimization problems. Here, we focus on its application the Max. Cut problem. which is a combinatorial problem whose approximate optimization beyond a minimum ratio r^* is NP-hard. The Max Cut problem is defined for any input graph $G = (V, E)$ where, $V = \{1, 2, \cdots, N\}$ denotes the set of vertices and $V = \{(\langle i, j\rangle), W_{i,j}\}$ is the set of edges, where $W_{i,j} \in \mathbb{R}_{\geq 0}$ is the weight of the edge $\langle i, j\rangle$ connecting vertices i and j. The goal of Max Cut is to maximize the following objective function

$$H_C = \sum_{\langle i,j\rangle} \frac{W_{i,j}}{2}(\mathbb{1} - \sigma_i^z \sigma_j^z) \tag{8}$$

where an edge $\langle i, j\rangle$ contributes with weight $W_{i,j}$ if and only if spins σ_i^z and σ_j^z are anti-aligned.

3 Codebase

The codebase of the QAOA algorithm alongside with the various functionality and connectivity with cloud services has been as a whole designed during this project. The current state of the codebase includes (but is not limited to) the following:

- Transformation of the initial problem into QUBO formulation
- Selection of the system size (number of qubits) and the alternating levels of problem and mixing Hamiltonians (p).
- Selection of platform that the QAOA is going to run, among simulators and Quantum Processing Units (QPUs) provided by different vendors (e.g. AWS)
- Selection of the classical optimization algorithm that will be used to find the optimal parameters of the QAOA algorithm.
- Selection of the initial angles strategy concerning the QAOA parameters, that will assist in the algorithm converging faster and with higher accuracy.

- Selection of the objective function that provides the energy landscape that we are investigating to find the global extremum solution.
- Selection of the circuit strategy that is going to be used by QAOA.
- Ability to consider shot noise and to simulate readout noise in the QAOA results, in order to have a more realistic environment to draw comparisons with the OPU noise.

  In each bullet point introduced here, there is a variety of options to choose from. The library has been designed in such a way that even more functionality can be easilv added and the effects to OAOA's performance be easily deduced and compared with other configurations.In each bullet point introduced here, there is a variety of options to choose from. The library has been designed in such a way that even more functionality can be easilv added and the effects to OAOA's performance be easily deduced and compared with other configurations.

4 Numerical simulation results

In this section, we present the mumerical results obtained from simulating the Max Cut problem with the Quantum Approximate Optimization Algorithm (QAOA). The simulations were carried out in a simulator provided by Rigetti Computing [1] through the cloud platform Forest, known as the Wavefunction Simulator.

QAOA requires the input problem to be provided as a graph, therefore we investigated two types of input graphs. These were 3-regular graphs and graphs that are designed as clusters of nodes (based on their euclidean distance) with a given distance between the clusters and a given connectivity inside the cluster as well as between cluster nodes.

This report contains a detailed explanation of how the performance of QAOA is affected by increasing the size of the quantum system N and the times that the problem Hamiltonian and mixing Hamiltonian are provided in the circuit described with the integer p, which defines the quality of the approximation. Furthermore, insights about the initialization of the 2p parameters is provided, as well as insights in how the optimizer should operate.

4.1 Approximation ratio results for a 3-regular graph (seed=0)

In the followins tables we provide the approximation ratio achieved after ontimization for various svstem sizes and values p for a single 3-regular graph with seed=0. We denote with **N** the number of qubits used in the graph, with **p** the number of alternating cost Hamiltonians and mixing Hamiltonians, with **Opt energy** the optimal energy for the specific graph (based on N and p) with **Energy** the energy achieved after the optimization based on the best set of angles that were found by QAOA and with **Ratio** the approximation ratio as calculated in equation 7.

N	p	Opt energy	Energy	Ratio
6	1	-2.5	-1.43922	0.84846
6	2	-2.5	-2.10295	0.94328
6	3	-2.5	-2.37802	0.98257
6	4	-2.5	-2.47253	0.99608
6	5	-2.5	-2.49772	0.99967
6	6	-2.5	-2.49996	0.99999

Table 1: Results for approximation ratio for the 3-regular graph (seed=0) for N=6.

N	p	Opt energy	Energy	Ratio
8	1	-4.0	-2.00692	0.80069
8	2	-4.0	-2.78424	0.87842
8	3	-4.0	-3.36098	0.93610
8	4	-4.0	-3.62749	0.96275
8	5	-4.0	-3.80144	0.98014
8	6	-4.0	-3.92370	0.99237
8	7	-4.0	-3.96720	0.99672
8	8	-4.0	-3.98578	0.99858
8	9	-4.0	-3.99483	0.99948
8	10	-4.0	-3.99794	0.99979

Table 2: Results for approximation ratio for the 3-regular graph (seed=0) for N=8.

N	p	Opt energy	Energy	Ratio
10	1	-5.5	-2.44178	0.76475
10	2	-5.5	-3.36847	0.83604
10	3	-5.5	-3.99394	0.88415
10	4	-5.5	-4.41197	0.91631
10	5	-5.5	-4.77194	0.94400
10	6	-5.5	-5.04800	0.96523
10	7	-5.5	-5.28828	0.98371
10	8	-5.5	-5.40603	0.99277
10	9	-5.5	-5.46272	0.99713
10	10	-5.5	-5.48536	0.99887
10	11	-5.5	-5.49418	0.99955

Table 3: Results for approximation ratio for the 3-regular graph (seed=0) for N=10.

N	p	Opt energy	Energy	Ratio
12	1	-7.0	-3.30272	0.76892
12	2	-7.0	-4.53196	0.84575
12	3	-7.0	-5.49924	0.90620
12	4	-7.0	-6.04603	0.94038
12	5	-7.0	-6.38125	0.96133
12	6	-7.0	-6.57759	0.97360
12	7	-7.0	-6.68604	0.98038
12	8	-7.0	-6.76754	0.98547
12	9	-7.0	-6.85116	0.99070
12	10	-7.0	-6.91733	0.99483
12	11	-7.0	-6.95430	0.99714

Table 4: Results for approximation ratio for the 3-regular graph (seed=0) for N=12.

N	p	Opt energy	Energy	Ratio
14	1	-7.5	-2.90904	0.74495
14	2	-7.5	-4.37977	0.82665
14	3	-7.5	-4.89842	0.85547
14	4	-7.5	-5.40160	0.88342
14	5	-7.5	-5.69723	0.89985

Table 5: Results for approximation ratio for the 3-regular graph (seed=0) for N=14.

N	p	Opt energy	Energy	Ratio
16	1	-9.0	-3.57517	0.74168
16	2	-9.0	-5.18964	0.81855
16	3	-9.0	-5.92861	0.85374
16	4	-9.0	-6.42334	0.87730
16	5	-9.0	-6.82132	0.89625

Table 6: Results for approximation ratio for the 3-regular graph (seed=0) for N=16.

N	p	Opt energy	Energy	Ratio
18	1	-10.5	-3.62400	0.71350
18	2	-10.5	-5.54042	0.79335
18	3	-10.5	-6.52271	0.83428
18	4	-10.5	-7.10220	0.85843

Table 7: Results for approximation ratio for the 3-regular graph (seed=0) for N=18.

N	p	Opt energy	Energy	Ratio
20	1	-11.0	-3.37599	0.70677
20	2	-11.0	-5.78943	0.79959
20	3	-11.0	-6.85212	0.84047
20	4	-11.0	-7.67772	0.87222

Table 8: Results for approximation ratio for the 3-regular graph (seed=0) for N=20.

N	p	Opt energy	Energy	Ratio
22	1	-12.5	-3.37799	0.68543
22	2	-12.5	-6.46523	0.79191

Table 9: Results for approximation ratio for the 3-regular graph (seed=0) for N=22.

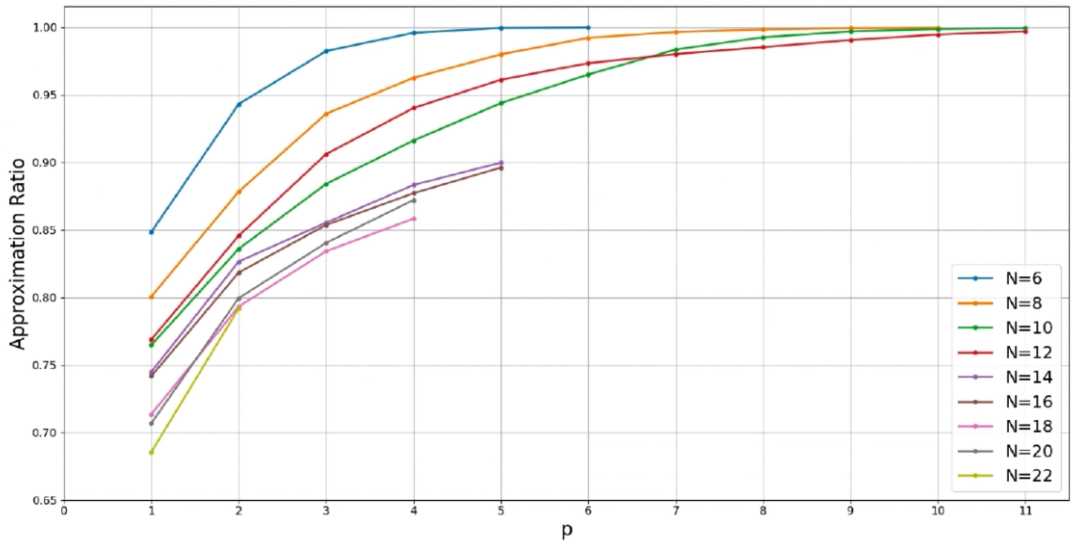

Figure 1: Approximation ratio for a single graph (seed=0) for the 3-regular Max Cut problem for various values of N and p.

4.2 Optimal angles for a single 3-regular graph (seed=0)

In the following tables, we provide the optimal angles found after optimization for various values of N and p for the single 3-regular graph (seed=0). Each beta parameter is denoted with B and each gamma parameter is denoted with G.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5
6	1	0.6964	3.7119										
6	2	0.9569	3.6439	0.5719	0.9168								
6	3	1.0134	3.5501	0.6859	0.8576	2.8032	4.1865						
6	4	1.0666	3.5203	0.7326	0.7705	2.7416	4.1703	2.9534	1.2682				
6	5	1.0755	3.4200	0.8314	0.6607	2.5864	3.9566	2.8602	1.1272	2.9617	1.2832		
6	6	1.0255	3.3577	0.7118	0.4720	2.5052	3.7627	2.7231	0.8180	2.9092	1.1231	0.1562	4.3277

Table 10: Optimal angles for the 3-regular graph (seed=0) for N=6.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5	B6	G6	B7	G7	B8	G8	B9	G9
8	1	0.7174	3.7231																		
8	2	0.9859	3.6501	0.5513	0.9279																
8	3	1.0513	3.5833	0.8337	0.8725	2.6381	4.1332														
8	4	1.1546	3.5023	0.8983	0.7570	2.3910	4.0469	2.7409	0.9657												
8	5	1.1582	3.4919	0.9701	0.6922	2.3748	3.9548	2.5648	0.9418	2.8534	1.0800										
8	6	1.1565	3.4597	0.9462	0.6613	2.2718	3.9178	2.4665	0.8249	2.7295	1.0835	0.2444	4.3957								
8	7	1.2118	3.4108	0.9728	0.6014	2.2858	3.8612	2.3233	0.7904	2.5738	0.8765	0.3416	4.2825	2.9171	4.3286						
8	8	1.2202	3.3815	0.9948	0.5440	2.2740	3.8150	2.3394	0.7383	2.4300	0.8149	0.4722	4.0570	2.8360	4.3040	2.9553	1.1783				
8	9	1.2201	3.3585	0.9848	0.4951	2.2809	3.7536	2.3421	0.6905	2.4456	0.7392	0.5997	3.9347	2.7455	4.0564	2.8720	1.1291	2.9721	1.1358		
8	10	1.2003	3.3271	0.9572	0.4310	2.3046	3.6864	2.3735	0.6218	2.4424	0.6677	0.6128	3.8579	2.6257	3.9196	2.7953	0.9235	2.8980	1.1190	3.0016	1.0930

Table 11: Optimal angles for the 3-regular graph (seed=0) for N=8.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5	B6	G6	B7	G7	B8	G8	B9	G9	B10	G10
10	1	0.7046	3.7164																				
10	2	0 9941	3.6257	0.5523	0.9069																		
10	3	1.1301	3.5801	0.8500	0.8381	2.6815	4.1374																
10	4	1.1350	3.5362	0.9444	0.7647	2.3953	4.0370	2.7273	0.9925														
10	5	1.2336	3.5121	1.0679	0.7205	2.2163	3.9718	2.3217	0.8969	2.6828	0.9574												
10	6	1.2209	3.4891	1.0807	0.6741	2.19301	3.9101	2.2777	0.8436	2.4093	0.9094	0.3556	4.1288										
10	7	1.1936	3.4830	1.0517	0.6807	2.1643	3.8850	2.2103	0.7988	2.3733	0.8635	0.5641	4.1515	2.8327	4.2618								
10	8	1.2246	3.4417	1.0448	0.6434	2.1750	3.8766	2.2044	0.7658	2.2587	0.7950	0.6518	4.0355	2.6570	4.2129	2.8728	1.0825						
10	9	1.2068	3.4057	1.0024	0.5674	2.2212	3.8132	2.2358	0.7262	2.2809	0.7399	0.7545	3.9026	2.5928	4.0414	2.7023	1.0518	2.9081	1.0577				
10	10	1.2262	3.3666	1.0122	0.5123	2.2377	3.7644	2.2718	0.6858	2.2915	0.7067	0.7911	3.8618	2.4815	3.9132	2.6544	0.9202	2.7530	1.0427	2.9490	1.0423		
11	11	1.2303	3.3387	1.0051	0.4566	2.2426	3.7127	2.2910	0.6450	2.3178	0.6751	0.8011	3.8309	2.4217	3.8647	2.5578	0.7946	2.7065	0.9288	2.8048	1.0286	2.9884	1.0436

Table 12: Optimal angles for the 3-regular graph (seed=0) for N=10.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5	B6	G6	B7	G7	B8	G8	B9	G9	B10	G10
12	1	0.7618	3.7457																				
12	2	1.0956	3.6247	0.5985	0.8834																		
12	3	1.1991	3.5738	0.9877	0.7723	2.5723	4.0420																
12	4	1.2402	3.5210	1.0539	0.7221	2.2573	3.9680	2.7032	0.9312														
12	5	1.2288	3.5000	1.0550	0.6961	2.2713	3.9501	2.5185	0.9379	2.8472	1.0885												
12	6	1.2233	3.4493	1.0495	0.6267	2.2126	3.8647	2.4308	0.8143	2.6399	0.9854	0.2489	4.2476										
12	7	1.9545	3.4220	1.0696	0.5930	2.1789	3.8982	2.2934	0.7422	2.5218	0.8617	0.4153	4.1973	2.9264	1.3057								
12	8	1.2706	3.4158	1.1024	0.5697	2.1393	3.8051	2.2320	0.7137	2.3228	0.7797	0.5937	4.0110	2.7552	4.2131	2.9152	1.1893						
12	9	1.2573	3.4192	1.1118	0.5687	2.1062	3.7753	2.1841	0.6807	2.2554	0.7456	0.8139	3.9158	2.5333	3.9990	2.7310	1.0434	2.9135	1.1423				
12	10	1.2544	3.4019	1.0957	0.5477	2.1091	3.7645	2.1722	0.6561	2.2376	0.7168	0.8508	3.8873	2.3645	3.9103	2.5828	0.8713	2.7635	1.0611	2.9332	1.1391		
12	11	1.2493	3.3783	1.0658	0.5106	2.1463	3.7292	2.1834	0.6268	2.2333	0.6744	0.8477	3.8445	2.3423	3.8678	2.4455	0.7555	2.6507	0.8721	2.8034	1.0552	2.9527	1.1102

Table 13: Optimal angles for the 3-regular graph (seed=0) for N=12.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4
14	1	2.2145	0.4334								
14	2	1.2123	0.5232	1.1089	6.0626						
14	3	1.8674	0.4939	0.5485	3.5567	2.4284	5.7730				
14	4	2.1241	0.4240	0.4924	4.0892	2.8668	4.4906	3.0612	0.9657		
14	5	1.7569	0.3194	1.1112	3.8238	2.2991	4.0301	2.8480	1.2450	0.2526	4.7512

Table 14: Optimal angles for the 3-regular graph (seed=0) for N=14.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4
16	1	0.7744	3.5827								
16	2	1.5622	0.4760	0.8015	6.0594						
16	3	1.9091	0.4714	0.6345	3.5419	2.4394	5.8233				
16	4	2.0806	0.4262	0.5014	4.0521	2.8889	4.5614	3.0342	0.9620		
16	5	1.7983	0.2829	1.0538	3.7434	2.0504	3.8838	2.9623	1.0582	0.4884	5.0378

Table 15: Optimal angles for the 3-regular graph (seed=0) for N=16.

N	p	B0	G0	B1	G1	B2	G2	B3	G3
18	1	0.8915	3.5462						
18	2	1.7932	0.4595	0.6202	6.0517				
18	3	1.0805	0.4482	0.7093	3.5521	2.4767	5.8639		
18	4	2.0592	0.4201	0.5049	4.0509	2.8919	4.5958	3.0325	0.9638

Table 16: Optimal angles for the 3-regular graph (seed=0) for N=18.

N	p	B0	G0	B1	G1	B2	G2	B3	G3
20	1	0.9763	3.4917						
20	2	1.9745	0.4500	0.4837	6.0409				
20	3	2.0744	0.4009	0.7749	3.5685	2.5351	5.9161		
20	4	2.0900	0.4020	0.5328	4.0194	2.8881	4.4914	3.0211	1.1297

Table 17: Optimal angles for the 3-regular graph (seed=0) for N=20.

N	p	B0	G0	B1	G1
22	1	1.0395	3.4632		
22	2	2.1018	0.4586	0.3897	6.0217

Table 18: Optimal angles for the 3-regular graph (seed=0) for N=22.

4.3 Approximation ratio results for a euclidean distance created graph

In the following tables, we provide the approximation ratio achieved after optimization for various system sizes and values p for a single euclidean distance created graph with seed=0 and distance between the two clusters equal to 3. We investigated various values of distance between the two clusters (1,2,3,4,5) for N=6 and we noticed that the highest approximation ratio was clearly achieved with distance=3. We denote with **N** the number of qubits used in the graph, with **p** the number of alternating cost Hamiltonians and mixing Hamiltonians, with **Opt energy** the optimal energy for the specific graph (based on N and p), with **Energy** the energy achieved after the optimization based on the best set of angles that were found by QAOA and with **Ratio** the approximation ratio as calculated in equation 7.

N	p	Opt energy	Energy	Ratio
6	1	-9.83728	-3.37546	0.72685
6	2	-9.83728	-6.73818	0.86900
6	3	-9.83728	-8.77447	0.95507
6	4	-9.83728	-9.64414	0.99184
6	5	-9.83728	-9.76589	0.99698
6	6	-9.83728	-9.80732	0.99873
6	7	-9.83728	-9.82135	0.99933
6	8	-9.83728	-9.82319	0.99940
6	9	-9.83728	-9.82513	0.99949
6	10	-9.83728	-9.82646	0.99954
6	11	-9.83728	-9.83153	0.99976

Table 19: Results for approximation ratio for the euclidean distance created graph (seed=0) for N=6.

N	p	Opt energy	Energy	Ratio
8	1	-17.77849	-5.02576	0.72583
8	2	-17.77849	-10.47441	0.84297
8	3	-17.77849	-14.09097	0.92072
8	4	-17.77849	-16.18225	0.96568
8	5	-17.77849	-17.25728	0.98880
8	6	-17.77849	-17.53122	0.99468
8	7	-17.77849	-17.67189	0.99771
8	8	-17.77849	-17.67867	0.99785
8	9	-17.77849	-17.69620	0.99823
8	10	-17.77849	-17.71313	0.99860
8	11	-17.77849	-17.71329	0.99608

Table 20: Results for approximation ratio for the euclidean distance created graph (seed=0) for N=8.

N	p	Opt energy	Energy	Ratio
10	1	-31.70878	-7.56247	0.71332
10	2	-31.70878	-17.33204	0.82931
10	3	-31.70878	-24.09536	0.90961
10	4	-31.70878	-28.89760	0.96662
10	5	-31.70878	-30.74372	0.98854
10	6	-31.70878	-31.19272	0.99387
10	7	-31.70878	-31.45601	0.99700
10	8	-31.70878	-31.60408	0.99876
10	9	-31.70878	-31.65797	0.99940
10	10	-31.70878	-31.67367	0.99958
10	1 1	-31.70878	-31.68413	0.99971

Table 21: Results for approximation ratio for the euclidean distance created graph (seed=0) for N=10.

N	p	Opt energy	Energy	Ratio
12	1	-45.93418	-9.67121	0.70032
12	2	-45.93418	-24.13374	0.81984
12	3	-45.93418	-33.62432	0.89827
12	4	-45.93418	-40.90837	0.95847
12	5	-45.93418	-44.11694	0.98498
12	6	-45.93418	-44.87184	0.99122
12	7	-45.93418	-45.29129	0.99469
12	8	-45.93418	-45.45997	0.99608
12	9	-45.93418	-45.55154	0.99684
12	10	-45.93418	-45.61508	0.99736
12	11	-45.93418	-45.63918	0.99756

Table 22: Results for approximation ratio for the euclidean distance created graph (seed=0) for N=12.

N	p	Opt energy	Energy	Ratio
14	1	-66.96014	-6.80840	0.65523
14	2	-66.96014	-27.23838	0.77233
14	3	-66.96014	-42.41057	0.85929
14	4	-66.96014	-54.23392	0.92706
14	5	-66.96014	-61.20554	0.96702
14	6	-66.96014	-62.25307	0.97302
14	7	-66.96014	-64.25735	0.98451
14	8	-66.96014	-64.48870	0.98581
14	9	-66.96014	-65.07487	0.98919

Table 23: Results for approximation ratio for the euclidean distance created graph (seed=0) for N=14.

N	p	Opt energy	Energy	Ratio
16	1	-80.90278	-6.94345	0.65632
16	2	-80.90278	-33.08181	0.77778
16	3	-80.90278	-47.50294	0.84480
16	4	-80.90278	-61.63884	0.91048
16	5	-80.90278	-70.52911	0.95180
16	6	-80.90278	-72.49501	0.96093
16	7	-80.90278	-76.05297	0.97746
16	8	-80.90278	-78.40711	0.98840

Table 24: Results for approximation ratio for the euclidean distance created graph (seed=0) for N=16.

N	p	Opt energy	Energy	Ratio
18	1	-95.57189	-7.28723	0.66184
18	2	-95.57189	-33.69307	0.76298
18	3	-95.57189	-52.31827	0.83432
18	4	-95.57189	-67.61457	0.89291
18	5	-95.57189	-79.85089	0.93978
18	6	-95.57189	-83.25423	0.95282

Table 25: Results for approximation ratio for the euclidean distance created graph (seed=0) for N=18.

N	p	Opt energy	Energy	Ratio
20	1	-116.58154	-7.83192	0.66010
20	2	-116.58154	-44.20470	0.77378
20	3	-116.58154	-61.01297	0.82632
20	4	-116.58154	-80.31623	0.88665
20	5	-116.58154	-95.51660	0.93416

Table 26: Results for approximation ratio for the euclidean distance created graph (seed=0) for N=20.

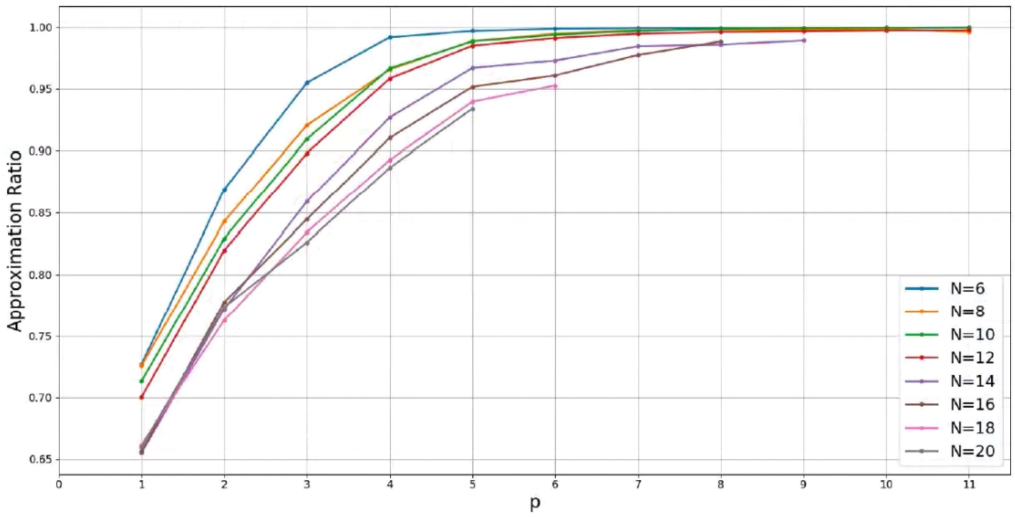

Figure 2: Approximation ratio for a single graph (seed=0) for the euclidean distance created graph for the Max Cut problem for various values of N and p.

4.4 Optimal angles for a single euclidean distance created graph (seed=0)

In the following tables, we provide the optimal angles found after optimization for various values of N and p for the single euclidean distance created graph (seed=0). Each beta parameter is denoted with B and each gamma parameter is denoted with G.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5	B6	G6	B7	G7	B8	G8	B9	G9	B10	G10
6	1	2.5073	0.2007																				
6	2	2.0664	0.2067	2.3250	0.3082																		
6	3	2.0519	0.1917	1.9889	0.2962	2.5070	0.3440																
6	4	2.2453	0.1242	2.2581	0.2412	2.2368	0.2388	2.6259	0.3344														
6	5	2.1831	0.0963	2.4398	0.1975	2.3725	0.1940	2.4445	0.2281	2.7580	0.3127												
6	6	2.1982	0.0871	2.4278	0.1745	2.4232	0.1885	2.4806	0.1954	2.5887	0.2514	2.9222	0.3283										
6	7	2.1526	0.0819	2.4382	0.1664	2.4412	0.1783	2.4834	0.1889	2.5740	0.2204	2.7372	0.2756	3.0008	0.3764								
6	8	2.0249	0.0678	2.3341	0.1508	2.4535	0.1711	2.4889	0.1834	2.5051	0.2010	2.6298	0.2371	2.7967	0.2875	3.0239	0.3833						
6	9	2.0256	0.0678	2.3347	0.1508	2.4524	0.1711	2.4875	0.1834	2.5041	0.2010	2.6276	0.2371	2.7927	0.2861	3.0224	0.3851	3.1324	4.0744				
6	10	2.0377	0.0675	2.3395	0.1508	2.4498	0.1716	2.4903	0.1816	2.5064	0.2001	2.6231	0.2369	2.7915	0.2828	3.0210	0.3741	3.1284	4.0235	0.0106	4.9761		
6	11	2.0491	0.0675	2.3445	0.1509	2.4479	0.1719	2.4923	0.1801	2.5071	0.1996	2.6196	0.2370	2.7902	0.2814	3.0174	0.3758	3.1232	4.0297	0.0100	4.8879	0.0122	2.8076

Table 27: Optimal angles for the euclidean distance created graph (seed=0) for N=6.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5	B6	G6	B7	G7	B8	G8	B9	G9	B10	G10
10	1	2.5867	0.1086																				
10	2	2.2344	0.1060	2.3337	0.1607																		
10	3	2.0567	0.0970	2.0553	0.1311	2.1443	0.1797																
10	4	2.1083	0.0930	1.9319	0.12707	2.2456	0.1463	95518	0.1700														
10	5	2.2715	0.0518	2.1426	0.1281	2.1017	0.1066	2.3531	0.1520	2.6572	0.1550												
10	6	2.2715	0.0562	2.3674	0.1014	2.2882	0.0845	2.4202	0.1050	2.4618	0.1277	2.7935	0.1543										
10	7	2.2456	0.0530	2.3785	0.0979	2.3188	0.0828	2.4090	0.0976	2.4210	0.1218	2.6037	0.1409	2.9535	0.1879								
10	8	2.1842	0.0463	2.4002	0.0888	2.3397	0.0841	2.3891	0.0941	2.4440	0.1073	2.5277	0.1233	2.7262	0.1492	2.9829	0.1981						
10	9	2.1113	0.0398	2.3644	0.0818	2.3868	0.0831	2.3774	0.0911	2.4446	0.0983	2.5137	0.1086	2.6190	0.1236	2.7973	0.1494	2.9964	0.1900				
10	10	2.0823	0.0356	2.3339	0.0752	2.3821	0.0819	2.3843	0.0904	2.4260	0.0950	2.5058	0.1014	2.5768	0.1111	2.7078	0.1273	2.8645	0.1496	3.0275	0.1790		
10	11	2.0931	0.0329	2.3459	0.0693	2.3705	0.0812	2.4009	0.0899	2.4331	0.0901	2.4875	0.0989	2.5518	0.1079	2.6549	0.1201	2.7877	0.1415	2.9273	0.1690	3.0829	0.1987

Table 29: Optimal angles for the euclidean distance created graph (seed=0) for N=10.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5	B6	G6	B7	G7	B8	G8	B9	G9	B10	G10
12	1	2.5947	0.0993																				
12	2	2.2648	0.0911	2.3133	0.1354																		
12	3	2.0689	0.0803	2.0581	0.1087	2.4501	0.1463																
12	4	2.1205	0.0766	1.8888	0.1022	2.2548	0.1184	2.5057	0.1407														
12	5	2.1142	0.0637	1.9594	0.1057	2.1422	0.1024	2.3254	0.1205	2.6759	0.1314												
12	6	2.0529	0.0590	1.9892	0.1004	2.1069	0.1038	2.2664	0.1120	2.4633	0.1213	2.8660	0.1447										
12	7	2.0337	0.0555	2.0059	0.0988	2.0964	0.1034	2.2447	0.1087	2.3986	0.1179	2.6580	0.1310	2.9494	0.1654								
12	8	2.0029	0.0521	2.0094	0.0927	2.1163	0.1032	2.2264	0.1020	2.3884	0.1076	2.5587	0.1111	2.7472	0.1267	2.9678	0.1582						
12	9	1.9933	0.0517	2.0062	0.0916	2.1148	0.1026	2.2257	0.1005	2.3814	0.1031	2.5162	0.1036	2.6532	0.1180	2.8156	0.1421	3.0238	0.1655				
12	10	1.9789	0.0504	1.9844	0.0892	2.1262	0.1011	2.2083	0.1006	2.3578	0.0996	2.4683	0.1022	2.5649	0.1127	2.7162	0.1326	2.8709	0.1548	3.0501	0.1805		
12	11	1.9697	0.0489	1.9732	0.0871	2.1166	0.0989	2.1994	0.1003	2.3330	0.0978	2.4423	0.0973	2.5073	0.1061	2.6412	0.1186	2.7993	0.1283	2.9112	0.1379	3.0506	0.1698

Table 30: Optimal angles for the euclidean distance created graph (seed=0) for N=12.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5	B6	G6	B7	G7	B8	G8
14	1	2.7476	0.0383																
14	2	2.2876	0.0849	2.3020	0.0757														
14	3	2.0985	0.0791	1.9875	0.0833	2.4564	0.1069												
14	4	1.9910	0.0730	1.8480	0.0810	2.0940	0.1043	2.4990	0.1114										
14	5	2.1084	0.0518	1.9121	0.0843	2.0414	0.0843	2.2037	0.0986	2.5835	0.1089								
14	6	1.9590	0.0402	2.0346	0.0754	2.0777	0.0825	2.1849	0.0786	2.3105	0.0918	2.6631	0.1026						
14	7	2.0775	0.0514	1.9497	0.0848	2.0767	0.0804	2.2272	0.0931	2.2627	0.0966	2.5210	0.1104	2.7756	0.1273				
14	8	2.1200	0.0505	1.9741	0.0841	2.1493	0.0749	2.3173	0.0844	2.3742	0.0849	2.6410	0.0789	2.7455	0.0803	2.8210	0.1020		
14	9	2.0158	0.0445	2.0474	0.0736	2.2049	0.0758	2.2937	0.0698	2.4135	0.0790	2.5210	0.0780	2.6035	0.0800	2.7054	0.0980	2.8777	0.1000

Table 31: Optimal angles for the euclidean distance created graph (seed=0) for N=14.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5	B6	G6	B7	G7
16	1	2.8017	0.0336														
16	2	2.1634	0.0749	2.1814	0.0799												
16	3	2.0792	0.0724	1.9623	0.0760	2.4286	0.0957										
16	4	2.0609	0.0651	1.8506	0.0733	2.1391	0.0957	2.5225	0.0938								
16	5	2.0496	0.0524	1.8823	0.0777	1.9740	0.0831	2.1862	0.0925	2.5523	0.0998						
16	6	1.8816	0.0424	1.9849	0.0689	1.9577	0.0846	2.0821	0.0823	2.2213	0.0902	2.6149	0.1010				
16	7	1.9980	0.0523	1.9271	0.0769	1.9889	0.0808	2.1837	0.0919	2.2509	0.0924	2.4781	0.1044	2.7592	0.1214		
16	8	2.0424	0.0486	1.9280	0.0764	2.0481	0.0768	2.2271	0.0852	2.3179	0.0879	2.5677	0.1007	2.6961	0.1247	2.9667	0.1365

Table 32: Optimal angles for the euclidean distance created graph (seed=0) for N=16.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4	B5	G5
18	1	2.8417	0.0300										
18	2	2.3480	0.0799	2.2783	0.0635								
18	3	2.2288	0.0560	2.0012	0.0760	2.5593	0.0647						
18	4	1.9341	0.0697	1.8679	0.0667	2.0435	0.0866	2.4169	0.0998				
18	5	2.2366	0.0568	1.9744	0.0631	2.2439	0.0585	2.2341	0.0779	2.5355	0.0863		
18	6	2.1925	0.0428	2.0128	0.0692	2.1654	0.0599	2.3453	0.0606	2.4100	0.0673	2.6230	0.0750

Table 33: Optimal angles for the euclidean distance created graph (seed=0) for N=18.

N	p	B0	G0	B1	G1	B2	G2	B3	G3	B4	G4
20	1	2.8739	0.0262								
20	2	2.2894	0.0694	2.1433	0.0694						
20	3	2.1292	0.0720	1.9437	0.0638	2.4638	0.0888				
20	4	1.9891	0.0565	1.9073	0.0625	2.0668	0.0777	2.4922	0.0859		
20	5	2516	0.0509	1.8926	0.0577	2.1914	0.0579	2.1927	0.0726	2.5295	0.0789

Table 34: Optimal angles for the euclidean distance created graph (seed=0) for N=20.

5 Going from a single graph to many

An important aspect that needs to be evaluated is whether we can get generalized results for a single graph. More accurately, whether the optimal angles that were found for a single graph can be used as the optimal angles of a different graph of the same type, same size and same level p. It has been shown that for graphs that have some underlying pattern, this type of generalization can be achieved.

For example, it has been proven for 3-regular graphs that generalization can occur based on the optimal parameters found from a single graph, which is also indicated in our simulations (see the following two figures). It was observed that for any p, if we fix parameters such that the objective function has a high value at some small number of qubits. then those same parameters will produce a high value at a larger number of qubits (N) [28]

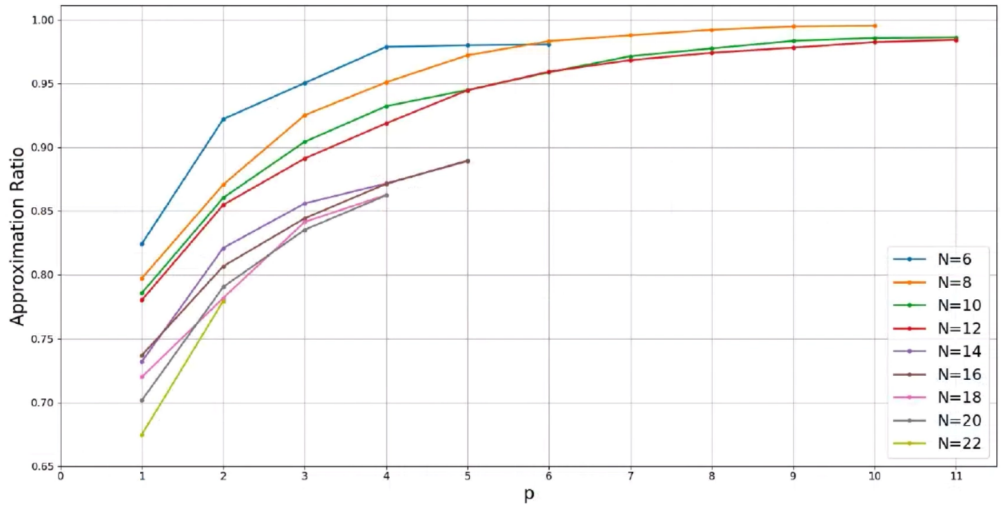

Figure 3: Approximation ratio for a 100 graphs with different seeds for the 3-regular graphs for the Max Cut problem for various values of I and p.

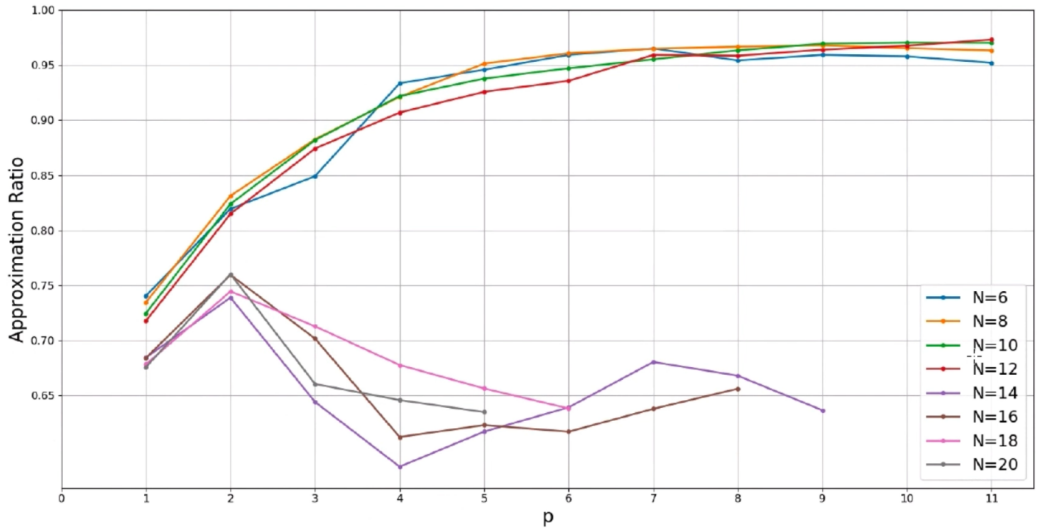

Figure 4: Approximation ratio for a 100 graphs with different seeds for the euclidean distance created graph for the Max Cut problem for various values of N and p.

On the contrary to the 3-regular graphs, which have a straight forward underlying construction pattern that helps generalization, the euclidean distance created graphs cannot generalize as well. The reasoning for that can be either that the true optimal parameters have not been found in our simulations when the optimization occurred for the single graph with seed=0 or that such generalization cannot easily be extrapolated from small system sizes. Based on the nature of the construction of these graphs, it can be assumed that the position of the separating plane between the two clusters plays an important role to generalization. This is even more evident as the system size increases, for example for $N \geq 14$ in Figure 4.

6 Initial angles strategy analysis

Providing QAOA with good initial angles before it starts the optimization process is an extremely meaningful step that can only increase the performance of QAOA. The initial angles are a set of angles that have been calculated in a classical way, in a pre-processing step and are then provided to QAOA as the starting point for the search of the optimal angles. In that way, QAOA spends time in the subspace where the optimal angles can be found, instead of randomly searching the whole Hilbert space querying the energy landscape with calls to the quantum computer, which is exponentially difficult as level p increases.

The first thing that should be considered before starting with any strategy for finding good initial angles is whether the problem that is being solved has an degeneracies because of symmetries in the parameter space.

Generally, QAOA has a time-reversal symmetry, $F_P(\vec{\gamma}, \vec{\beta}) = F_P(-\vec{\gamma}, -\vec{\beta})$ since both H_C and H_B

are realvalued. Furthermore, for problems like the Max Cut, there is an additional $\mathbb{Z}_2$ symmetry as $e^{-i(\pi/2)H_B} \equiv (\sigma^x)^{\otimes N}$ commutes through the circuit. Moreover, the structure of the Max Cut problem on unweighted d-regular graphs creates redundancy, since $e^{-i\pi H_C} = \mathbb{1}$ if d is even and $(\sigma^z)^{\otimes N}$ if d is odd. This allow us to restrict $\beta_i = [-\pi/4, \pi/4)$ in general and $\gamma_i = [-\pi/2, \pi/2)$ for unweighted d-regular graphs. [2]

Specifically for the d-regular graphs. it has been observed that there is concentration of the parameters, with the distribution of each parameter becoming narrower as level p is increasing. Such behaviour can be exploited by finding the average optimal parameters once and reuse them for all instances, thereby eliminating the computational cost per instance of finding the optimal parameters. In addition to that one can find the optimal parameters without any call to the QPU leveraging the experimental finding that the optimal parameters do not depend on the problem size [20]. According to this scenario, one can find the optimal parameters for a small instance problem, and then utilize these as initial parameters in a larger instance of the same problem [28].

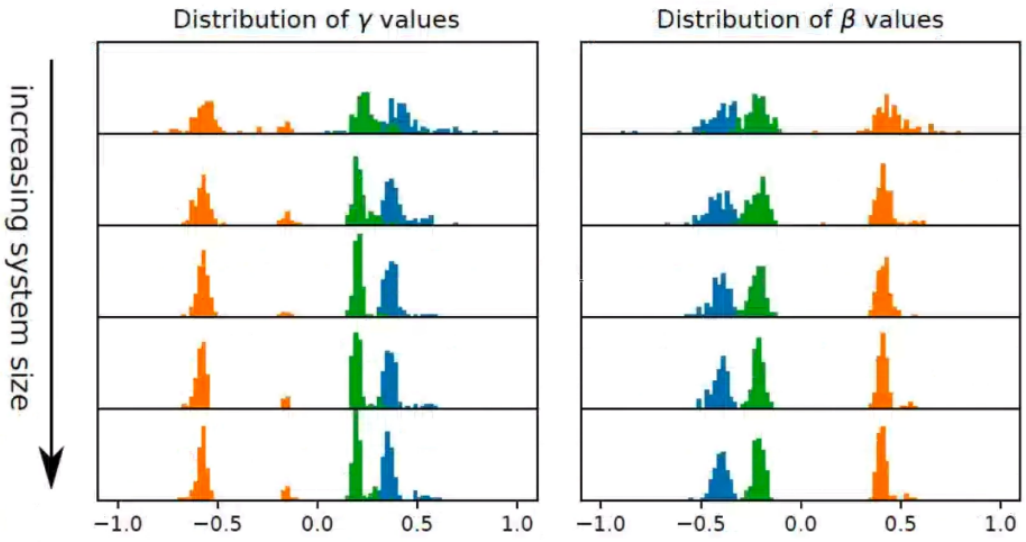

Figure 5: Evidence of concentration of the optimal parameter distribution for d-regular graphs [20]

Furthermore it has been shown that for d-regular graphs the optimal γ_i^* tends to increase smoothly with $i = 1, 2, \cdots, p$, while β_i^* tends to decrease smoothly [2].

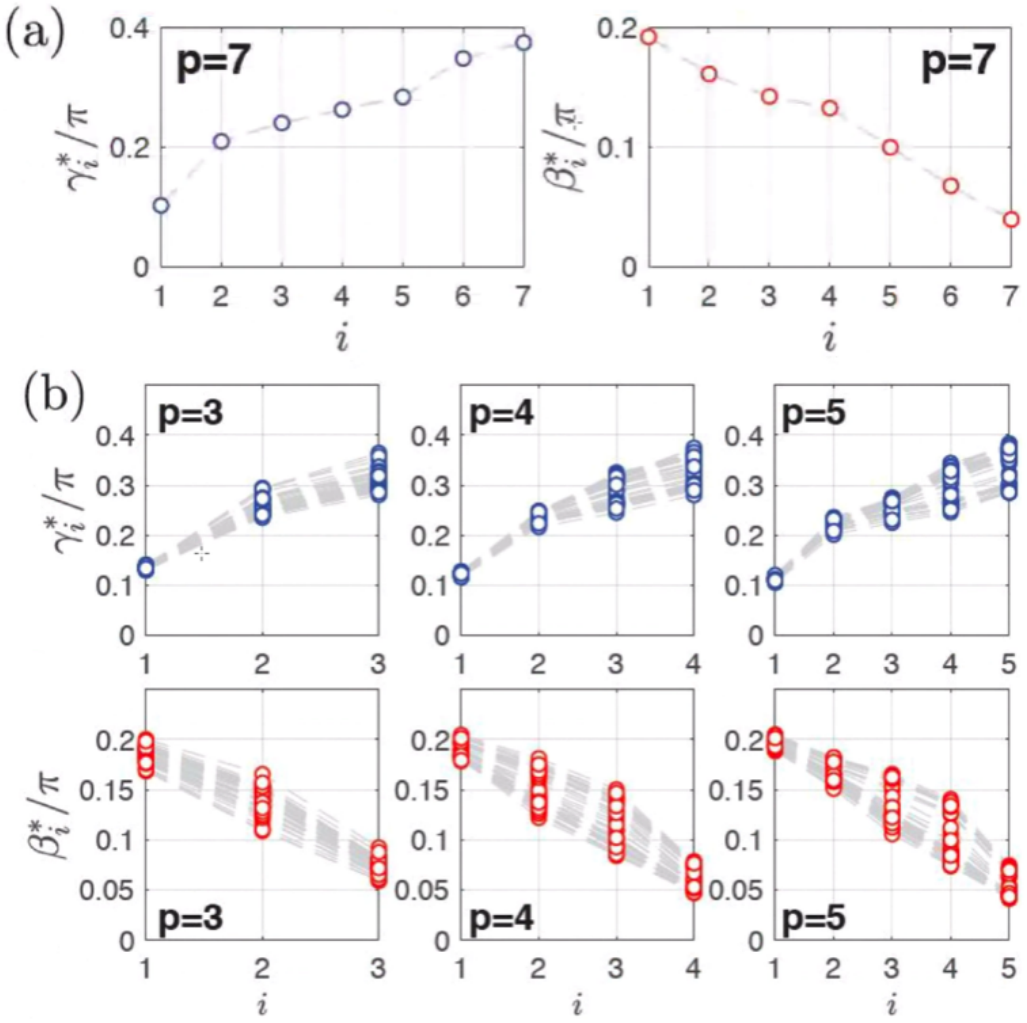

Figure 6: Evidence of trends for γ_i^* and β_i^* for d-regular graphs [2]

After the suitable subspace has been found, it can be explored for the best possible initial angles (parameters). By initializing parameters in the neighborhood of a local minimum of the cost landscape, one ensures more consistent local optimization convergence in a fewer number of iterations and a better overall answer with respect to the global landscape. Good initialization is thus crucial to promote the convergence of local optimizers to local extrema and to select reasonably good local minima [21]. This process occurs in the classical computer and many schemes have been proposed. The simplest strategy that can be employed is random initialization, where one picks a random set of parameters in the chosen subspace to begin optimization. However, in order to find the global optimum in a highly non-convex landscape, the number of random initialization runs needed generically scales exponentially with the number of parameters, which becomes intractable for large number of parameters [30]. Furthermore, parametrized quantum circuits initialized with random initial parameter values are characterized by barren plateaus where the gradient becomes exponentially small in

the number of qubits [8], therefore an exponentially large precision is needed to navigate through the landscape [13]. There have been proposed strategies [18, 29, 27, 25, 2, 1] to avoid the vanishing gradient problein (barren plateaus) by providing solutions to overcome these limitations, however it is not that straightforward how to solve them. Also, noise from experimental devices will definitely affect the cost landscape, therefore making it harder to find the optimal parameters under noisy conditions. Besides random initialization, there have been proposed various heuristic strategies to find the best initial parameters, such as the

- **The interpolation method**
  The interpolation method uses linear interpolation to choose initial parameters starting at level $p = 1$. After level $p = 1$ is optimized one can linearly interpolate the curve formed by optimized parameters at level p to extract a set of initial parameters for level $p + 1$ [30].

- **The Fourier method**
  The Fourier method uses a new parameterization of QAOA. Instead of using the 2p parameters $(\vec{\gamma}, \vec{\beta}) \in \mathbb{R}^{2p}$, we switch to 2q parameters $(\vec{u}, \vec{v}) \in \mathbb{R}^{2p}$, where the individual elements γ_i and β_i are written as functions of $(\vec{u}, \vec{v})$ through the following transformation:

$$\gamma_i = \sum_{k=1}^{q} u_k \sin[(k - 1/2)(i - 1/2)\pi/p] \tag{9}$$

$$\beta_i = \sum_{k=1}^{q} v_k \cos[(k - 1/2)(i - 1/2)\pi/p] \tag{10}$$

  These transformations are known as Discrete Sine/Cosine Transform, where u_k and v_k can be interpreted as the amplitude of k^{th} frequency component for $\vec{\gamma}$ and $\vec{\beta}$, respectively. In this strategy, when optimizing level $p+1$, the initial parameters are generated by simple re-using the optimized amplitudes $(\vec{u}^*, \vec{v}^*)$ from level p. Note that when $q \geq p$, the $(\vec{u}, \vec{v})$ parametrization is capable of describing all possible QAOA protocols at level p [30].

- **Classical neural networks, meta-learners**
  In [26], the authors train classical neural networks to assist in the quantum learning process, which is also know as meta-learning, to rapidly find approximate optima in the parameter landscape for several classes of quantum variational algorithms. Specifically, they train classical recurrent neural networks to find approximately optimal parameters within a small number of queries of the cost function. In [28], the authors similarly train classical neural networks to evaluate the optimization of quantum heuristics. They show that the meta-learner comes near to the global optima more frequently than all other optimizers tested in a noisy parameter setting environment. They also find that the meta-learner is generally more resistant to noise, for example seeing a smaller reduction in performance in Noisy and Sampling environments and performs better on average by a 'gain' metric than its closest comparable competitor L-BFGS-B.

- Elimination of the classical outer learning loop and temsor network techniques In [22], the authors propose a strategy that eliminates the classical outer learning loop of the QAOA, which speeds up the search for initial parameters significantly, since there are no calls to the quantum computer. They present a strategy to find good parameters for QAOA based on

topological arguments of the problem graph and tensor network techniques. Starting from the observation of the concentration of control parameters of QAOA, they find a way to classically infer parameters which scales polynomially in the number of qubits and exponentially with the depth of the circuit. Using this strategy, the quantum processing unit (QPU) is only needed to infer the final state of QAOA.

In our simulations, we used a grid search optimizer with a decreasing step ranging from 0.1 to 0.0001 to find the optimal parameters for $p = 1$ for all N explored in the two problems that were studied. For every following level p, we used a linear interpolation technique for the parameters up to the last pair and the same grid search approach for the last pair of parameters.

We have verified that there exists concentration in the parameter distribution for both type of problems that were tested and that the interpolation technique can be effortlessly utilized. To better showcase these findings, we present the following figures. [**?**] We start with the parameter (β, γ) analysis for N=12 and p=1 up to 4 and present the trend of the angles for the 3-regular graph with seed=0.

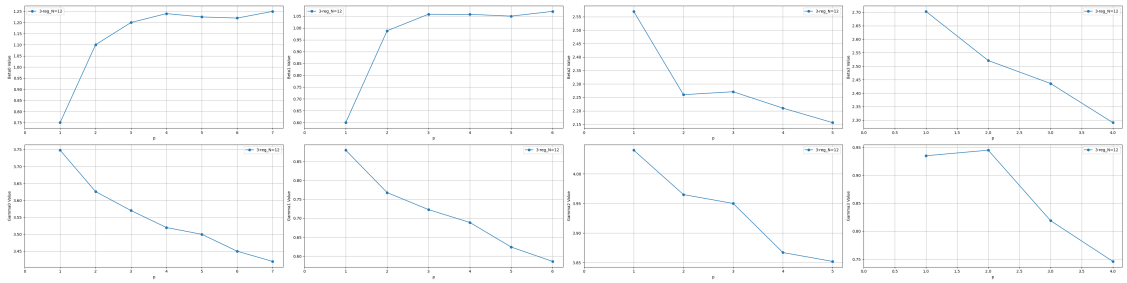

Figure 7: Presentation of the trends in the four paris of (β, γ) angles for the 3-regular graph with seed=0 with a quantum system of N=12 qubits.

We present the corresponding figure for the euclidean distance created graph with seed=0 (N=12, p=15).

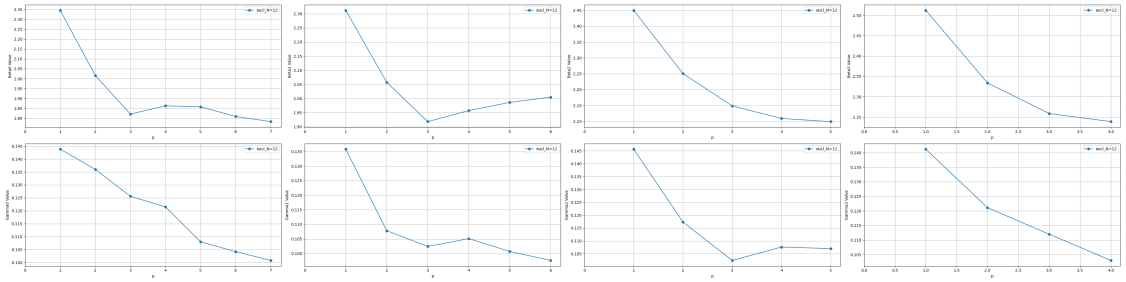

Figure 8: Presentation of the trends in the four paris of (β, γ) angles for the euclidean distance created graph with seed=0 with a quantum system of N=12 qubits.

Based on the figures 7 and 8, it is straightforward to see what the initial angles for each (β, γ) angle should be for the same set of angles for the following level p. Then the only unknown remains the last pair of the new level p, which can be found either by classical optimization (for example grid search) or by looking at the last pair of the same p of another N.
In order to compare with Figure 6, we present the following graphs indicating how the trend of the (β, γ) values progress for the same p (in our case we considered p=3) for the 3-regular and euclidean distance created graph with seed=0 for N=6 up to 20.

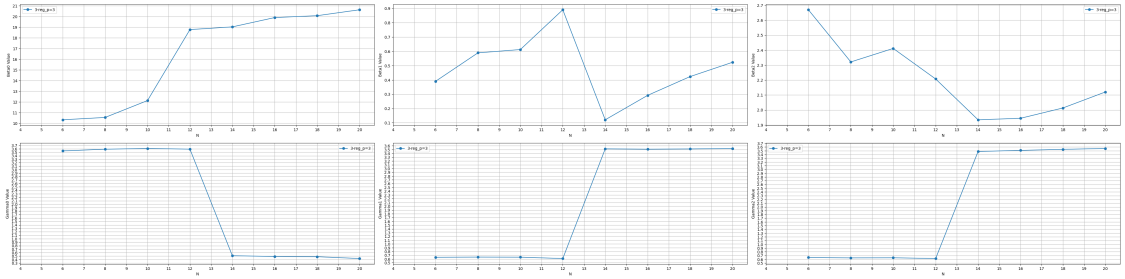

Figure 9: Presentation of the trends in the three pairs of (β, γ) angles for the 3-regular graph with seed=0 with a quantum system ranging from N=6 to 20 qubits.

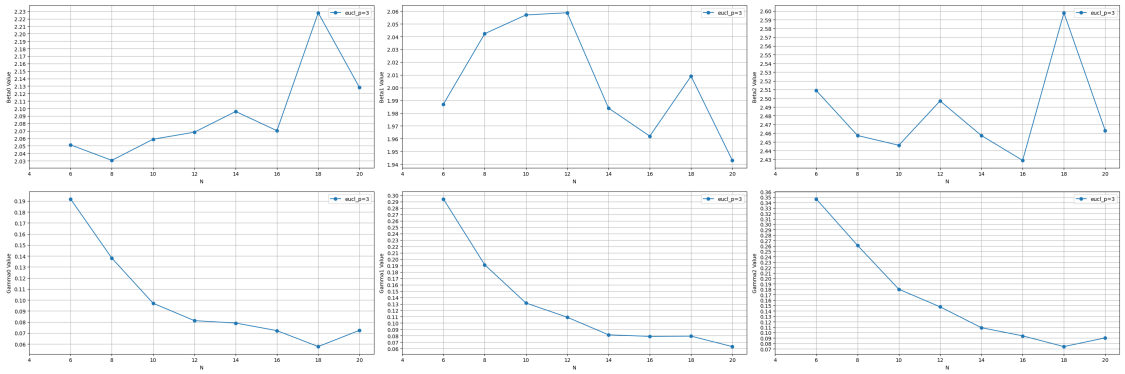

Figure 10: Presentation of the trends in the three pairs of (β, γ) angles for the euclidean distance created graph with seed=0 with a quantum system ranging from N=6 to 20 qubits.

Based on the figures 9 and 10, we draw the conclusion that for the 3-regular graphs there is a clear trend for each (β, γ) angle. However, the actual values are not entirely consistent in our simulations (there is a change in the values, although respecting the trend) that appears at N=14. We believe that this has occurred due to the degeneracies of the 3-regular graph, since the bounds that we assumed for both types of graphs that were considered were $\gamma = [0, 2\pi]$ and $\beta = [0, \pi]$ which allows many degeneracies in the case of 3-regular graphs. Thus, we believe that by taking the symmetrical values to the ones reported for $N \geq 14$, we could follow closely the values obtained for (β, γ) for $N \leq 12$. In the case of the euclidean distance created graph, things are even clearer. We observe that especially for the γ angles the trend is clear, but in general the changes in the actual values for both β and γ are very small and predictable. Therefore, we can get a good estimate for the initial (β, γ) angles for the same level p for a different N based on these values. Then, the optimizer should

be sufficiently close to the optimal values, which gives a higher probability of finding the optimal angles for the system size N that is being tested.

7 Access to QPU's

An important aspect of the designed QAOA library that has not been highlighted enough so far in this report is the ability to run this implementation of QAOA to various QPUs through cloud services. We have an operational connection to the different QPUs of Rigetti [1] and IonQ [11] through the Amazon Web Services (AWS) [7] cloud services. AWS is providing on-demand cloud computing platforms and APIs to individuals and companies, on a metered pay-as-you-go basis. These cloud computing web services provide a variety of basic abstract technical infrastructure and distributed computing building blocks and tools, through the Internet [15].

The ability to run algorithms in such devices is extremely important, because there are a lot of factors that are not taken into account when designing the quantum algorithm. Factors like the realistic levels of noise, routing of quantum states, extra resources required to correctly perform the algorithm and many more, end up giving much worse results compared to simulations of the quantum algorithm. Addressing such issues helps design more efficient quantum algorithms and increase their performance under a realistic scenario. Furthermore, quantum hardware is also constantly becoming more trustworthy (increased fidelities of quantum gates and operations), available to hold more qubits and able to run faster than before.

8 Conclusions

This project involved the implementation of the Quantum Approximate Optimization Algorithm (QAOA) in an in-house library and experimenting with it. The classification problem of Max Cut was selected as the problem to be analyzed and two types of graph problems were investigated: 3-regular graphs and Euclidean distance created graphs. The library that was created has the capabilities to run the QAOA algorithm in a variety of simulators and QPUs offered by various platforms such as Rigetti, IBM and AWS. At each platform there is the ability to either run in a simulator (WaveFunction or BitString) or a Quantum Processing Unit (QPU). Furthermore, we added a module that simulates readout (measurement) noise in the system, so that we have the ability to potentially test the performance against a noisy environment which makes the implementation more realistic compared to current hardware. Moreover, we implemented a custom made optimizer as well as added many available optimizers from other sources. Also, we implemented a strategy of finding good initial angles before the starting the optimization by designing a technique that incorporates proposals from publications as well as our own observations.
We have presented the results of the simulations that were run on the Wave Function simulator of the Rigetti platform without adding any readout noise. These simulations indicated that the implemented QAOA works well and is able to obtain similar results compared to other recent publications. Future work could include comparing different optimizers to each other and ranking them, experimenting with shot noise in the BitString simulator and readout noise in the results of QAOA. Furthermore, running QAOA to an available QPU would be extremely interesting since it will in-

dicate how much the performance of QAOA is affected because of the hardware noise. Comparing these results with no noise, shot noise and readout noise would be greatly beneficial. [**?**]

References

[1] Abhinav Anand, Matthias Degroote, and Alán Aspuru-Guzik. Natural evolutionary strategies for variational quantum computation. *Machine Learning: Science and Technology*, 2(4):045012, 2021.

[2] Andrew Arrasmith, Marco Cerezo, Piotr Czarnik, Lukasz Cincio, and Patrick J Coles. Effect of barren plateaus on gradient-free optimization. *Quantum*, 5:558, 2021.

[3] Fernando GSL Brandao, Michael Broughton, Edward Farhi, Sam Gutmann, and Hartmut Neven. For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances. *arXiv preprint arXiv:1812.04170*, 2018.

[4] Edward Farhi, David Gamarnik, and Sam Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: Worst case examples. *arXiv preprint arXiv:2005.08747*, 2020.

[5] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[6] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[7] Jose Garcia-Alonso, Javier Rojo, David Valencia, Enrique Moguel, Javier Berrocal, and Juan Manuel Murillo. Quantum software as a service through a quantum api gateway. *IEEE Internet Computing*, 26(1):34–41, 2021.

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

[9] Tad Hogg and Dmitriy Portnov. Quantum optimization. *Information Sciences*, 128(3-4):181–197, 2000.

[10] 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, 2017.

[11] David Kielpinski, Chris Monroe, and David J Wineland. Architecture for a large-scale ion-trap quantum computer. *Nature*, 417(6890):709–711, 2002.

[12] Andrew Lucas. Ising formulations of many np problems. *Frontiers in physics*, 2:5, 2014.

[13] 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):4812, 2018.

[14] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, et al. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, 2018.

[15] Hoa T Nguyen, Muhammad Usman, and Rajkumar Buyya. Qfaas: A serverless function-as-a-service framework for quantum computing. *Future Generation Computer Systems*, 154:281–300, 2024.

[16] Johannes S Otterbach, Riccardo Manenti, Nasser Alidoust, A Bestwick, M Block, B Bloom, S Caldwell, N Didier, E Schuyler Fried, S Hong, et al. Unsupervised machine learning on a hybrid quantum computer. *arXiv preprint arXiv:1712.05771*, 2017.

[17] 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):1–7, 2014.

[18] Arthur Pesah, Marco Cerezo, Samson Wang, Tyler Volkoff, Andrew T Sornborger, and Patrick J Coles. Absence of barren plateaus in quantum convolutional neural networks. *Physical Review X*, 11(4):041011, 2021.

[19] Ivo G Rosenberg. Reduction of bivalent maximization to the quadratic case. 1975.

[20] Andrea Skolik, Jarrod R McClean, Masoud Mohseni, Patrick Van Der Smagt, and Martin Leib. Layerwise learning for quantum neural networks. *Quantum Machine Intelligence*, 3:1–11, 2021.

[21] Robert Smith. Practical quantum computers: Challenges and opportunities. *Rigetti Computing Technical Report*, 2016.

[22] Michael Streif and Martin Leib. Training the quantum approximate optimization algorithm without access to a quantum processing unit. *Quantum Science and Technology*, 5(3):034008, 2020.

[23] Kristan Temme, Tobias J Osborne, Karl G Vollbrecht, David Poulin, and Frank Verstraete. Quantum metropolis sampling. *Nature*, 471(7336):87–90, 2011.

[24] Nazim Turan. *Quantum computing: eine Einführung ins Unbekannte*. Nazim Turan, 2022.

[25] AV Uvarov and Jacob D Biamonte. On barren plateaus and cost function locality in variational quantum algorithms. *Journal of Physics A: Mathematical and Theoretical*, 54(24):245301, 2021.

[26] Guillaume Verdon, Michael Broughton, Jarrod R McClean, Kevin J Sung, Ryan Babbush, Zhang Jiang, Hartmut Neven, and Masoud Mohseni. Learning to learn with quantum neural networks via classical neural networks. *arXiv preprint arXiv:1907.05415*, 2019.

[27] Samson Wang, Enrico Fontana, Marco Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature communications*, 12(1):6961, 2021.

[28] Max Wilson, Rachel Stromswold, Filip Wudarski, Stuart Hadfield, Norm M Tubman, and Eleanor G Rieffel. Optimizing quantum heuristics with meta-learning. *Quantum Machine Intelligence*, 3:1–14, 2021.

[29] Kaining Zhang, Min-Hsiu Hsieh, Liu Liu, and Dacheng Tao. Toward trainability of quantum neural networks. *arXiv preprint arXiv:2011.06258*, 2020.

[30] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067, 2020.