An Analysis of Quantum Approximate Optimization Algorithm Performance Across Various Graph Types

Wouter Pennings[*]
Fontys University of Applied Sciences
465288@student.fontys.nl

Nico Kuijpers[†]
Fontys University of Applied Sciences
nico.kuijpers@fontys.nl

Qin Zhao[‡]
Fontys University of Applied Sciences
q.zhao@fontys.nl

This Version: June 2025

Abstract Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et.

Keywords: Quantum computing, QAOA, Quantum noise, Graphs, Maxcut, Optimization

[*]Student
[†]Lecturer
[‡]Lecturer

1. Introduction

In recent years, quantum computing has emerged as a promising field with the potential to revolutionize the way complex computational problems in fields such as medicine, material science and machine learning are approached [1], [2]. Various algorithms have gotton attention for their theoratical ability to solve problems more efficiently than their classical peers; Quantum Approximate Optimization Algorithm is one of them. To understand this paper, it is important to grasp two key quantum phenomena: superposition and entanglement. Superposition allows a quantum bit (qubit) to exist in a both the 0 and 1 state simultaneously until it is measured, at which point it collapses into a single definite state [3], [4], this mechanic is represented by a Hadamard (H) gate. Entanglement, on the other hand, describes a peculiar connection between two or more qubits where their states are intertwined, meaning the state of one qubit influences the state of another [3], [4], this mechanic is represented by a controlled-not (CNOT) gate.

Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum algorithm designed to accurately approximate optimization problems. A hybrid quantum algorithm, means it combines both classical and quantum computational resources, leveraging the strengths of each to solve complex problems [5]. This exploration uses QAOA to address the max-cut problem. Max-cut is a graph problem where the goal is to divide the nodes into two groups so that the number of cut edges connecting the different groups is as large as possible. In this paper uses the varient where all edges have the same weight. With QAOA, each node in the graph is represented by a qubit, which is put into a superposition state using a Hadamard (H) gate. The edges connecting these nodes are represented by entangling the corresponding qubits, using either a ZZ-gate or a sequence of CNOT-RZ-CNOT gates.

QAOA has three key parameters: β, γ, and p. The parameter p sets the depth of the circuit, essentially dictating how many layers of operations are applied. The parameters β and γ are the angles that are classically optimized to find the best possible solution to our problem. The β angles are used in the mixing layer, also known as the mixing Hamiltonian, where they control the mixing of quantum states by rotating qubits around the X-axis. This process helps explore different configurations of the solution space. On the other hand, the γ angles are applied in the cost layer, or the cost Hamiltonian, where they adjust the rotations, guiding the

solution towards an optimal outcome. The implementation used in this paper draws on the approach detailed in Farhi et al. [6], ensuring that the study builds on established methods while exploring new insights.

The objective of this study is to explore the behavior of QAOA across a range of graph types. The generation of the graphs is achieved by altering two parameters: the number of nodes and connectivity. A quantum computer simulator is used to run the experiments, this is done for two main reasons. Firstly, this is much more approachable than using an actual quantum computer. Secondly, simulation allows for functionality that actual quantum computers do not have, mainly being able to determine the amount of noise. The majority of experiments are conducted in a noise-less environment; however, several experiments are also conducted with noise to study its impact. There is a lack of comprehensive studies that explore the practical application of QAOA, and this paper aims to address this gap. In quantum computing research, algorithms are often proposed without sufficient experimentation to understand how well they work in practice, this study seeks to provide valuable insights in this regard.

2. Related work

3. Methodology

The objective of this study is to understand how the Quantum Approximate Optimization Algorithm (QAOA) behaves when solving the max-cut problem across various graphs in both ideal and noisy environments.

Two types of experiments are conducted. The first type, noisy experiments, aims to understand how noise impacts QAOA's effectiveness. This is achieved by applying increasing levels of noise to a single graph until a realistic noise level is reached. The second type, noiseless or ideal experiments, investigates QAOA performance across different graphs, each with increasing numbers of nodes or edges.

For the ideal experiments, graphs consist of 4, 5, 6, 7, 8, 9, 10, or 15 nodes with randomly generated edges. Each size category has two variants: low connectivity and high connectivity. Low connectivity implies a 40% chance of connection between nodes, while high connectivity implies an 80% chance. This results in a total of sixteen distinct graphs, all generated using NetworkX.

The noisy experiments utilize a single graph, referred to as the "house with a X" (Figure 1). These experiments explore how increasing noise levels degrade QAOA's effectiveness. The noise levels, implemented as `noise_factor`, considered are 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, and 1. Baseline experiments include randomly selected max-cut instances and an ideal circuit simulation. The noise models are based on Di Bartolomeo et al.'s implementation [7], with parameters used from Qiskit Kyiv quantum computer. These parameters represent the single-qubit depolarizing error probability (`p`), amplitude damping time of a qubit in ns (`T1`) and the dephasing time ofa qubit in ns (`T2`). `P` is multiplied by `noise_factor` while `T1` and `T2` are divided by `noise_factor` as they are inversy proportional to `noise_factor`.

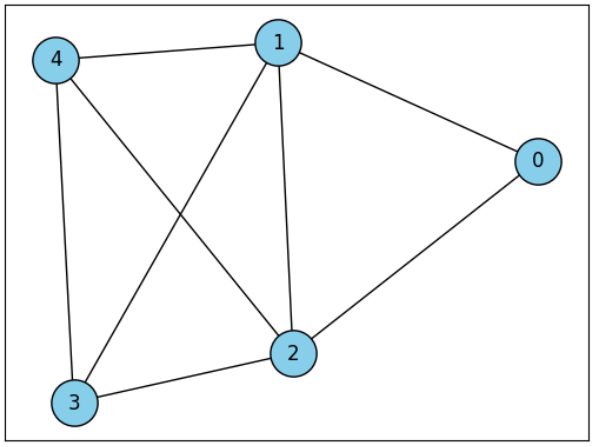

Figure 1: Visualization of "house with a X"

Both ideal and noisy experiments employ the Simultaneous Perturbation Stochastic Approximation (SPSA) to optimize γ and β parameters, parameter `P` is fixed at five across all experiments. SPSA's parameters are set to default values except for iteration (set to 10), learning rate (0.2), learning rate decay (0.602), and perturbation magnitude (0.2). The initial guess used in this sudy of γ and β for all layers is 0.5. There are many optimizers available, such as COBYLA, however SPSA seems to perform well in context such as these [8].

To evaluate and compare the performance of the experiments, several evaluation metrics are used. The metrics are the same for both type of experiments. The metrics are:

- **mean_random**: The average cut size based on probabilities, assuming P is 0.
- **mean**: The average cut size based on the statevector's probabilities.
- **mean_top_10**: The average cut size of the top ten cuts with the highest probability.
- **mean_top_5**: The average cut size of the top five cuts with the highest probability.
- **mean_top_3**: The average cut size of the top three cuts with the highest probability.

- **maxcut_qaoa**: The cut with the highest probability from QAOA.
- **maxcut_bruteforce**: The actual max-cut of the graph determined through classical brute-force methods.

Average cut size is a weighted mean where the weights are based on the probabilities.
In the case of noisy experiments, the statevector probabilities are also shown, which indicate the deviation in results due to noise. These metrics help determine QAOA's performance effectiveness in solving the max-cut problem.

3.1. QAOA implementation & measuring

The implementation used for the experiments in this paper and described below is based on Fahri et al [6]. The experiments are not run on actual hardware, but simulated using QuantumSim [9], an educational quantum computer simulator developed at Fontys University of Applied Sciences. This paper uses a computationally improved version which allows for faster simulation and for more qubits to be simulated.
QAOA's three main steps of this paper's implementation (Listing 1) are described below. Figure 2 provides a visualization of the circuit.

- **Step 1**: Creating quantum circuit: Initializing a quantum circuit, each qubit represents a node of the graph.
- **Step 2**: Circuit into superposition: Putting all qubits into superposition using a Hadamard (`H`) gate.
- **Step 3**: Construction layers of cost and mixer Hamiltonian: Applying a `ZZ` gate between two qubits, which entangles them, this is the cost Hamiltonian. Using CNOT-RZ-CNOT gates to simulate a `ZZ` gate . Applying a `RX` gate to each qubit, this is the mixer Hamiltonian, the β value of the `RX` gate is multiplied by two.

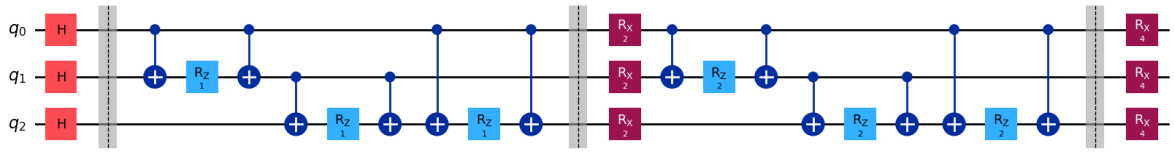

Figure 2: The QAOA circuit was generated by Qiskit. Barriers are added before and after the cost layer to prevent Qiskit from rearranging them. The circuit has two layers, and both β and γ are `[1, 2]`. The graph is a basic triangle, it has three nodes, 1 is connected to 2 and 3, and 3 is also connected to 2.

After these steps a QAOA circuit is created and can be measured to find the maxcut of a graph. We will be directly looking at the probabilities of the simulator instead of measuring the statevector thousands of times. Looking directly at the probabilities of the statevector is only possible in simulators. It does not change the results of the quantum circuit as with enough measurements the results would reflect the probabilities of the statevector anyway. It does make the simulation process faster as measuring a circuit 100.000 times does take a significant amount of time. In the case of a noisy circuit, the circuit is executed a hundred times, and a mean of the probabilities is taken. The probabilities of the noisy circuit are indeterministic, therefore, the probabilities can be quite different between executes; requiring several executes.

4. Results

4.1. Noisy experiments

4.2. Noiseless experiments

Table 1 and Figure 3 show all metric tracked for the low connectivity graphs. Table 2 and Figure 4 show all metric tracked for the high connectivity graphs. `Nodes` describes how many nodes there are in the graph used in the experiment. `Mean random`, `Mean`, `Mean Top 10`, `Mean Top 5` and `Mean Top 3` all are average cut sizes, in the case of `Mean Top N` it only calculates the average cut size of the outcomes with the top N highest probabilities. `Maxcut QAOA` and `Maxcut BruteForce` show how many edges were cut. `Maxcut QAOA` is the cut with the highest probability of all possible soltions. `Figure` is a reference to the graph which are in the Section 6

4.2.1. Low connectivity

Nodes	Mean Random	Mean All	Mean Top 10	Mean Top 5	Mean Top 3	MaxCut QAOA	MaxCut Brute Force	Figure
4	1.50	2.65	2.66	2.89	2.96	3	3	Figure 5
5	2.50	3.17	3.61	4.00	4.00	4	4	Figure 7
6	3.00	4.48	5.00	5.00	5.00	5	5	Figure 9
7	3.00	4.71	5.48	5.69	5.87	6	6	Figure 11
8	6.00	6.65	7.58	7.56	7.30	7	10	Figure 13
9	8.00	8.83	10.14	10.37	10.56	11	13	Figure 15
10	8.00	9.02	12.86	13.00	13.00	13	13	Figure 17
15	24.50	27.54	32.95	32.56	32.30	32	35	Figure 19

Table 1: This table shows all the metrics tracked for the noiseless experiments with low connectivity

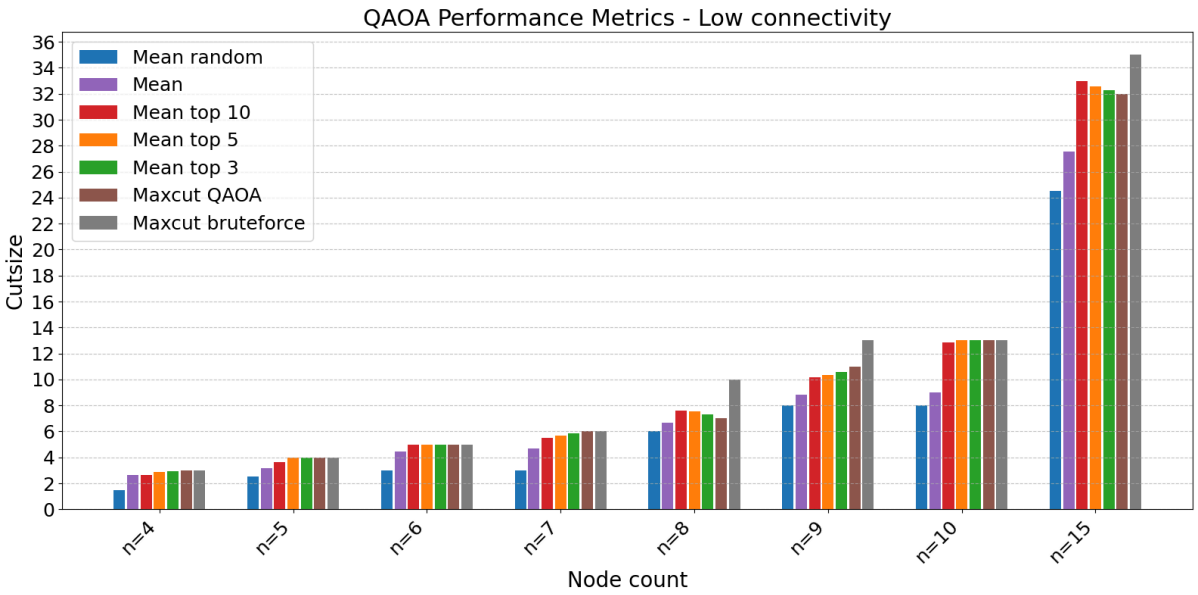

Figure 3: This graph show how well QAOA performance a maxcut on a variety of graphs will different sizes with low connectivity by comparing multiple metrics to `Maxcut bruteforce`. Same data as Table 1

4.2.2. High connectivity

Nodes	Mean Random	Mean All	Mean Top 10	Mean Top 5	Mean Top 3	MaxCut QAOA	MaxCut Brute Force	Figure
4	2.50	3.65	3.78	3.87	3.95	4	4	Figure 6
5	3.50	4.37	4.85	5.00	5.00	5	5	Figure 8
6	7.00	7.98	8.85	9.00	9.00	9	9	Figure 10
7	9.00	9.64	11.63	12.00	12.00	12	12	Figure 12
8	11.00	11.63	12.40	13.33	13.83	15	15	Figure 14
9	16.00	16.69	17.60	17.43	17.69	18	20	Figure 16
10	21.50	21.99	20.78	20.00	20.00	20	25	Figure 18
15	41.00	42.16	48.38	48.80	48.24	48	53	Figure 20

Table 2: This table shows all the metrics tracked for the noiseless experiments with high connectivity

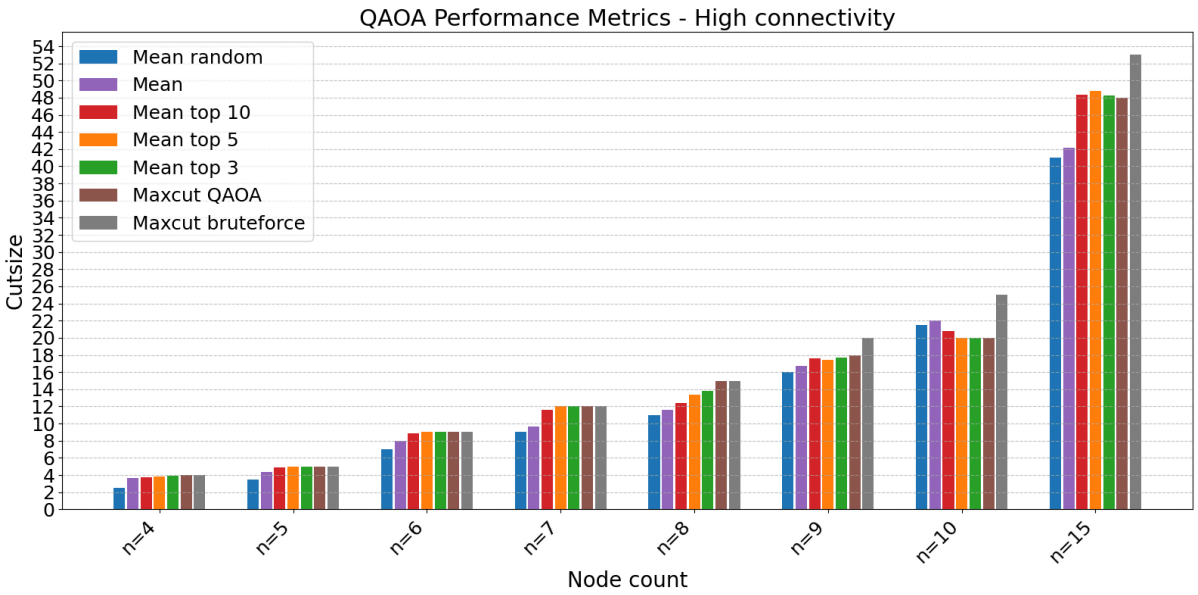

Figure 4: This graph show how well QAOA performance a maxcut on a variety of graphs will different sizes with high connectivity by comparing multiple metrics to `Maxcut bruteforce`. Same data as Table 2

4.2.3. Conclusion

5. Discussion

5.1. Noiseless experiments

The graphs and table (Figure 3, Table 1, Figure 4, Table 2) provide insights into the performance and behaviour of noiseless QAOA. $N = 10$ from high connectivity, will be ignored for the

following general trends and insights seen in the data, as it is an outlier. It is the only graph in with `Mean random` outperforms all the `Mean Top N` metrics.

- QAOA is better than picking a random maxcut. Even when QAOA performs poorly it still outperformance a random maxcut.
- For both low and high connectivity experiments QAOA find the most optimal maxcut up to $N = 7$. However from $N = 8$ on there is a clear gap between `Maxcut QAOA` and `Maxcut bruteforce`, except for the low connectivity graph $N = 10$, where it was again quite effective for this one instance.
- Taking a mean of all probabilities does not represent the effectiveness of QAOA well. In was always better to take the average cutsize of at least the top 10 cuts with the highest probabilities.
- The larger the graphs get, for both high and low connectivity, there is a general trend that the smaller the `N` in `Mean Top N` does not nescecarily mean I higher average cut. This indicates that the cuts with the greatest amount of cut edges are not the ones with the highest probability.

Overall, these experiments show that QAOA, at small graphs sizes, regardless of connectivity level, approximates the optimal Maxcut solution well. The effectiveness of QAOA does degrade the larger the graphs get, Connectivity does not have a large impact.

5.2. Noiseless QAOA P Count

For the experiments in the study a constant P value of five was picked; this choise was arbitrary. QAOA performed the worst with the largest graphs ($n = 15$) tested in this study, but this is also realy kind of visable with $N = 9$ and $N = 10$. Increasing `P` could give QAOA more iteration to find a better maxcut.

5.3. Limitations

- Generate more than one type of graph for each experiment. QAOA sometimes behaves strangely with certain graphs. Averaging the outcomes of each experiment might have reduced that. The outliers, $N = 8$ for low connectivity and $N = 10$ for high connectivity, are most likely the result.
- TODO

5.4. Conceptual problem: Parameter selection

The strategy of optimizing QAOA parameters individually for each graph raises a conceptual concern: using a classical optimizer, such as COBYLA, to determine parameters that yield a good MaxCut solution effectively involves solving the problem in advance. This introduces an apparent circularity to the approach. The process involves sampling a large set of parameter combinations and selecting those that yield favorable results. However, this raises a concern: if the goal is simply to identify high-quality solutions through classical optimization, would it not be more efficient to use a classical heuristic, such as simulated annealing, to solve MaxCut directly? A common response to this concern might be that exploring a wide range of QAOA parameters and selecting the best-performing configurations is ultimately more effective or computationally efficient than relying on purely classical heuristics; particularly as quantum hardware improves. Nonetheless, this highlights an area where further research is needed. QAOA, and quantum computing more broadly, remains largely theoretical and is still far from practical deployment for real-world problems. It would be especially interesting to investigate whether a set of "universal parameters" could be found that perform reasonably well across many graph instances, even if not optimally for each one. Alternatively, future work could explore whether good parameter choices can be heuristically "guessed" based on structural properties of the graph, such as its size, degree distribution, or symmetry. Such approaches could help mitigate the inefficiencies and circularity of instance-specific optimization, potentially making QAOA a more scalable and practical algorithm.

6. Figures

```
def qaoa_circuit(gamma:list[float], beta:list[float], nodes:list, edges:list,
p:int) -> Circuit:
    # Consistency check
    if len(gamma) != p or len(beta) != p:
        raise ValueError(f"Lists gamma and beta should be of length p = {p}")

    # Create circuit witn n qubits, where n is the number of nodes
    n = len(nodes)
    circuit = Circuit(n)

    # Initialize circuit by applying the Hadamard gate to all qubits
    for q in range(n):
        circuit.hadamard(q)

    # Construct p alternating cost and mixer layers
    for i in range(p):

        # Construct cost layer with parameter gamma[i]
        for edge in edges:
            circuit.cnot(edge[0], edge[1])
            circuit.rotate_z(2 * gamma[i], edge[1])
            circuit.cnot(edge[0], edge[1])

        # Construct mixer layer with parameter beta[i]
        for q in range(n):
            circuit.rotate_x(2 * beta[i], q)

    return circuit
```

Listing 1: Example implemenation of QAOA, implemented in QuantumSim

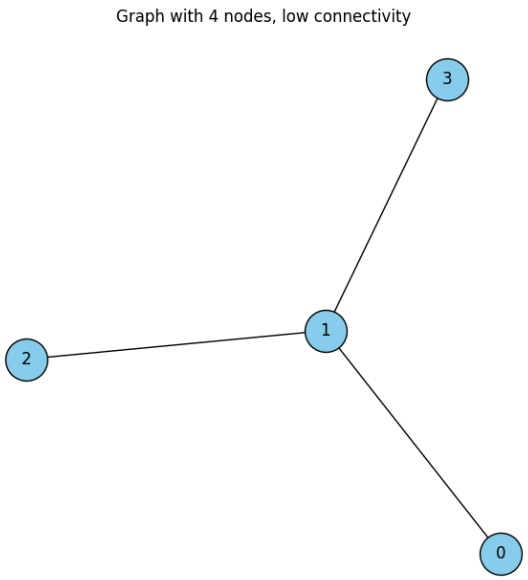

Figure 5: Graph with 5 nodes and low connectivity

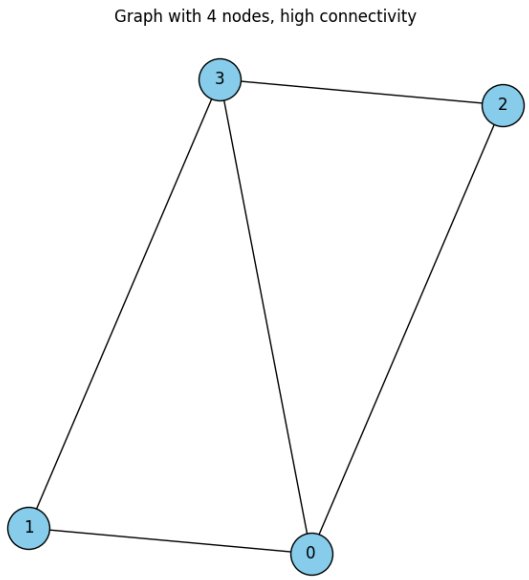

Figure 6: Graph with 5 nodes and high connectivity

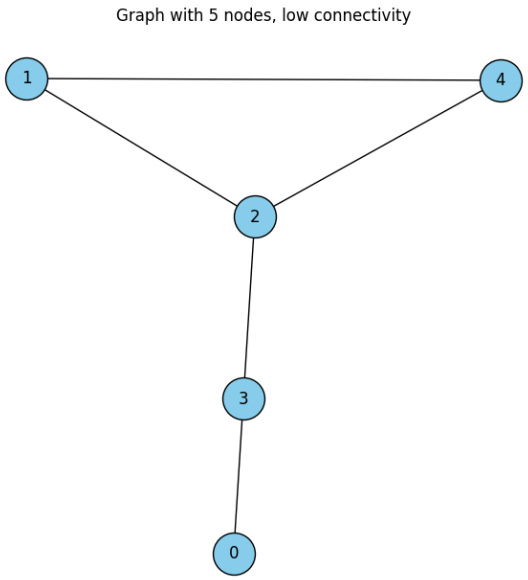

Figure 7: Graph with 5 nodes and low connectivity

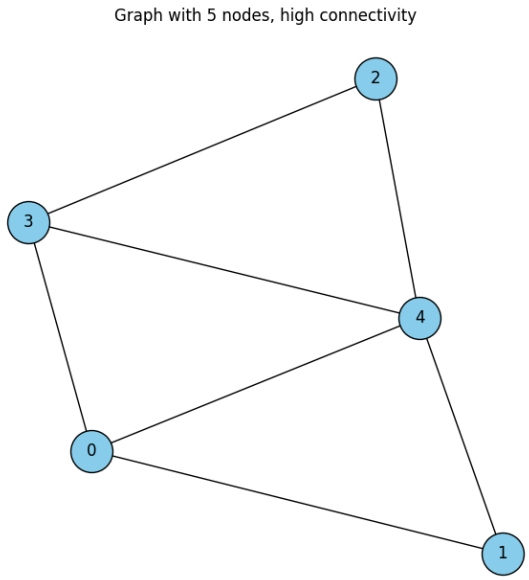

Figure 8: Graph with 5 nodes and high connectivity

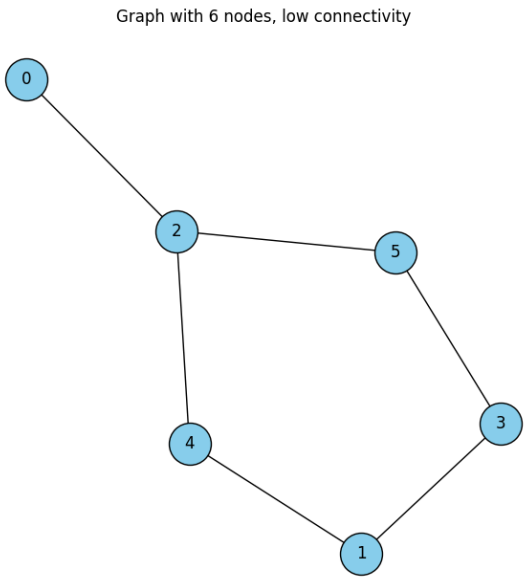

Figure 9: Graph with 6 nodes and low connectivity

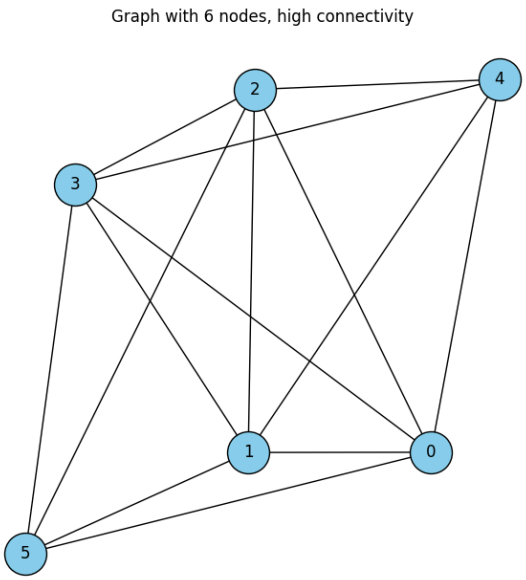

Figure 10: Graph with 6 nodes and high connectivity

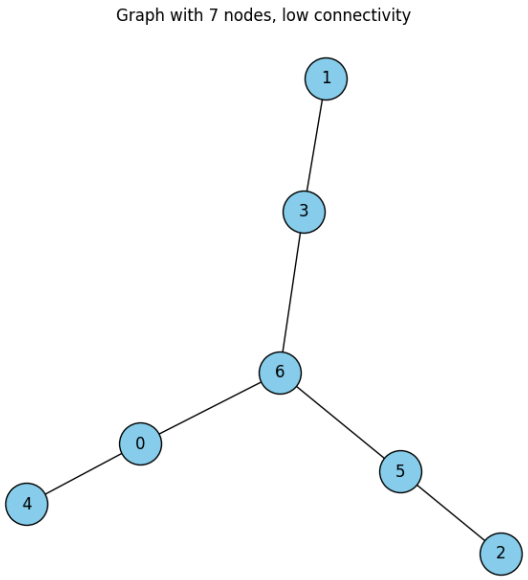

Figure 11: Graph with 7 nodes and low connectivity

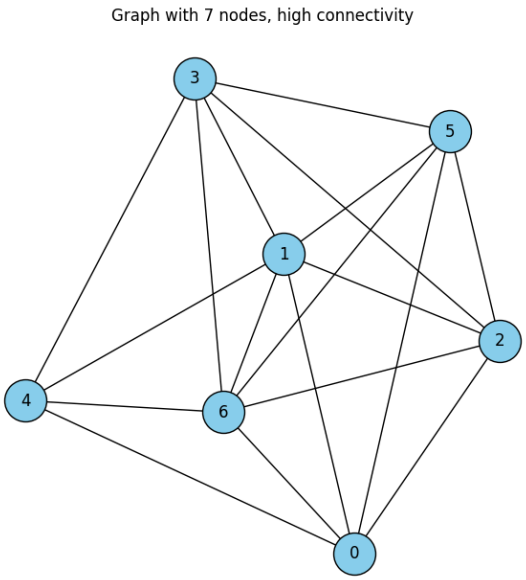

Figure 12: Graph with 7 nodes and high connectivity

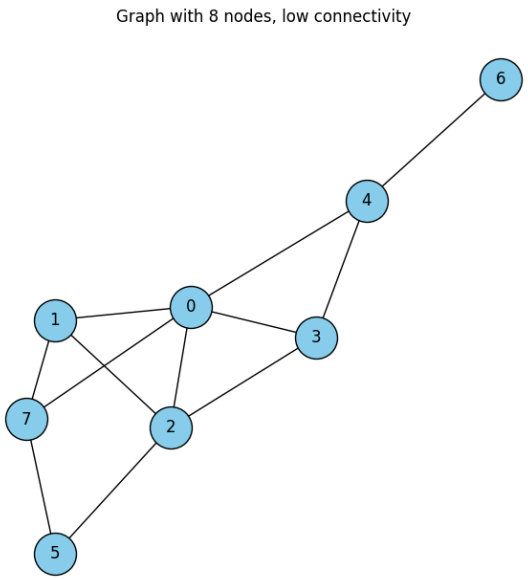

Figure 13: Graph with 8 nodes and low connectivity

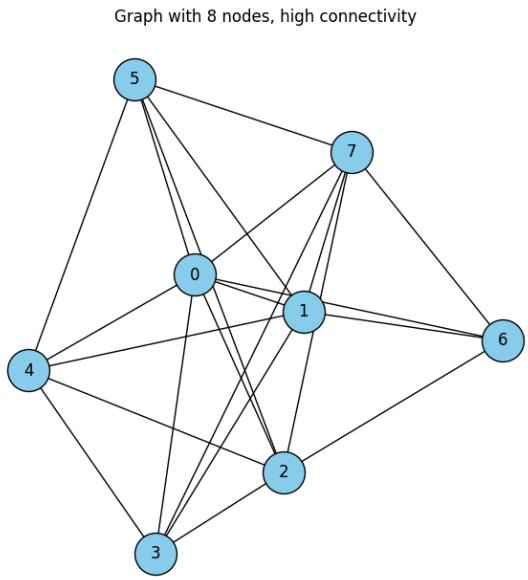

Figure 14: Graph with 8 nodes and high connectivity

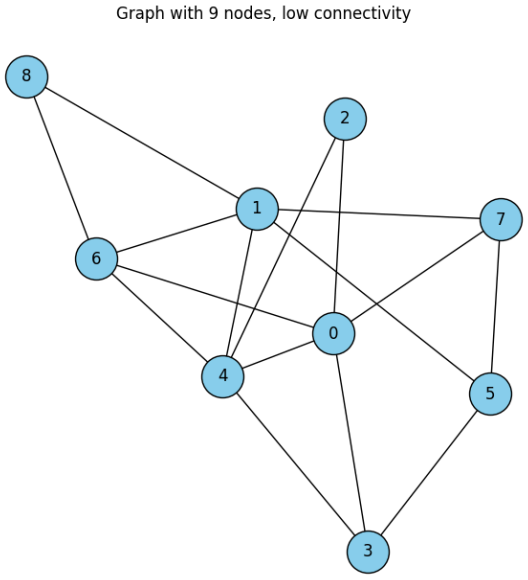

Figure 15: Graph with 9 nodes and low connectivity

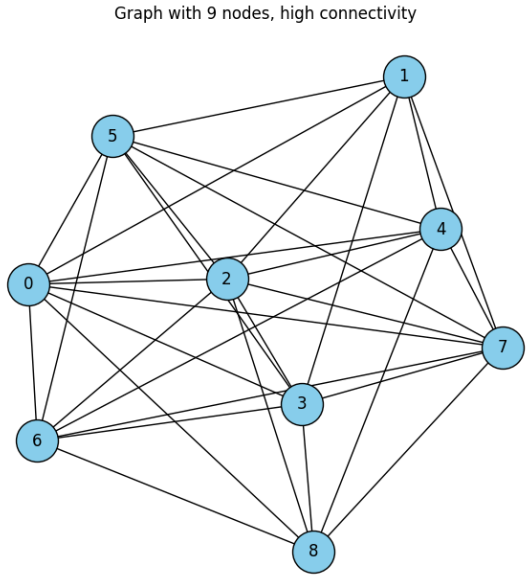

Figure 16: Graph with 9 nodes and high connectivity

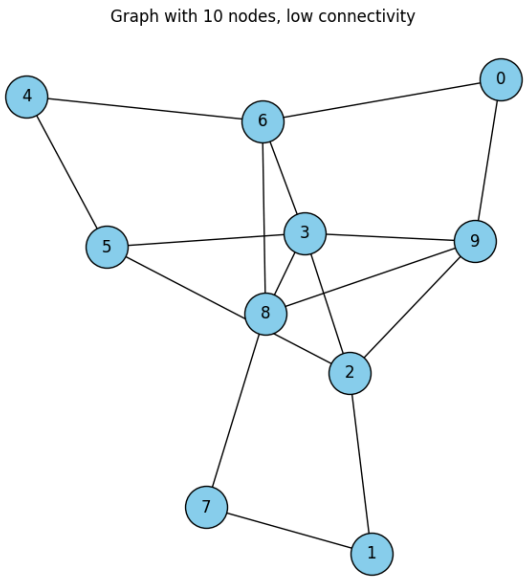

Figure 17: Graph with 10 nodes and low connectivity

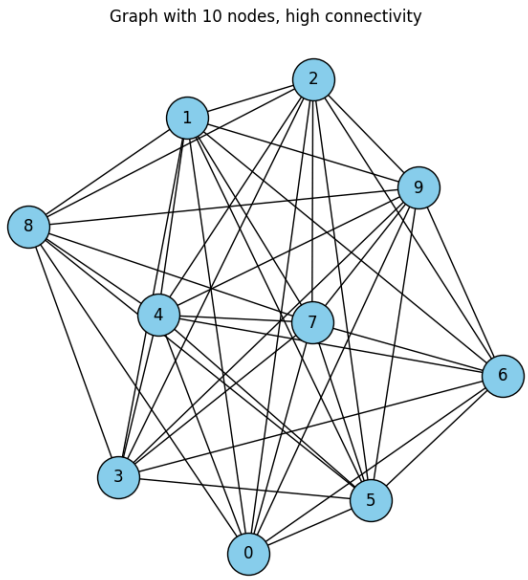

Figure 18: Graph with 10 nodes and high connectivity

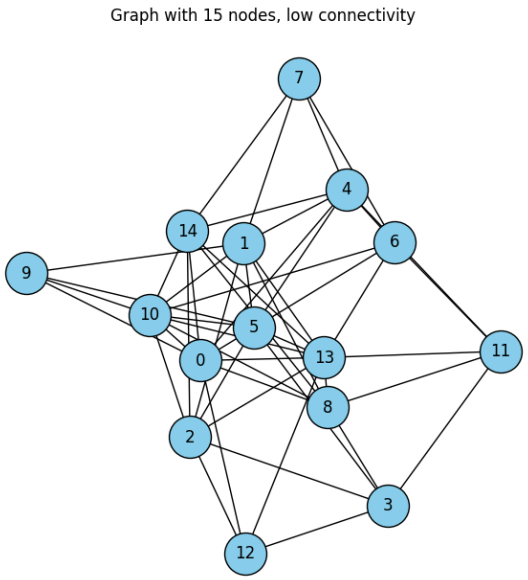

Figure 19: Graph with 15 nodes and low connectivity

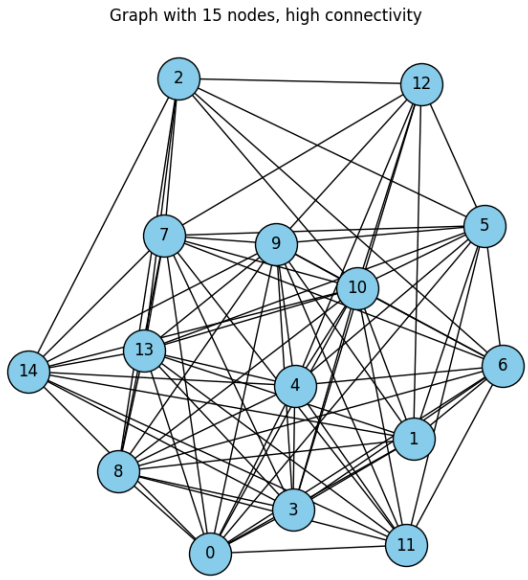

Figure 20: Graph with 15 nodes and high connectivity

References

[1] M. B. Hastings, D. Wecker, B. Bauer, and M. Troyer, "Improving Quantum Algorithms for Quantum Chemistry." Accessed: Jun. 19, 2025. [Online]. Available: http://arxiv.org/abs/1403.1539

[2] K. M. Svore and M. Troyer, "The Quantum Future of Computation," *Computer*, vol. 49, no. 9, pp. 21–30, Sep. 2016, doi: 10.1109/MC.2016.293.

[3] N. S. Yanofsky, "An Introduction to Quantum Computing." Accessed: Jun. 19, 2025. [Online]. Available: https://arxiv.org/abs/0708.0261v1

[4] Prashant, "A Study on the basics of Quantum Computing." Accessed: Jun. 19, 2025. [Online]. Available: http://arxiv.org/abs/quant-ph/0511061

[5] R. Campos, "Hybrid Quantum-Classical Algorithms." Accessed: Jun. 19, 2025. [Online]. Available: http://arxiv.org/abs/2406.12371

[6] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm." Accessed: May 09, 2025. [Online]. Available: http://arxiv.org/abs/1411.4028

[7] G. Di Bartolomeo *et al.*, "Noisy gates for simulating quantum computers," *Physical Review Research*, vol. 5, no. 4, p. 43210, Dec. 2023, doi: 10.1103/PhysRevResearch.5.043210.

[8] G. E. Crooks, "Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem." Accessed: May 13, 2025. [Online]. Available: http://arxiv.org/abs/1811.08419

[9] N. Kuijpers, "QuantumSim." [Online]. Available: https://github.com/nicokuijpers/QuantumSim