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

Wouter Pennings
Fontys University of Applied Sciences
465288@student.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

[*]This paper is a work in progress. Please do not cite without permission.

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. Various algorithms have garnered attention for their ability to solve problems more efficiently than their classical peers; Quantum Approximate Optimization Algorithm is one of them.

Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum algorithm designed to accurately approximate optimization problems . 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 is as large as possible. 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 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., ensuring that the study builds on established methods while exploring new insights.

The objective of this paper 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. 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 literature. 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. 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 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, with parameters derived from Qiskit. An additional parameter, noise_factor, adjusts the noise model's T1, T2 (inversely proportional to noise_factor), and P (proportional to noise_factor) to assess its impact on QAOA output quality.

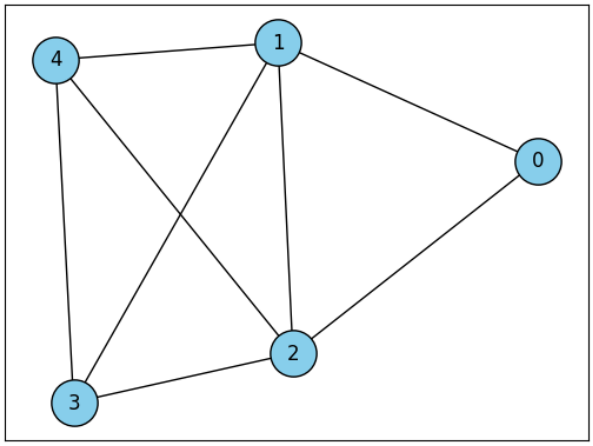

Figure 1: Visualization of "house with a X"

Both ideal and noisy experiments employ the Simultaneous Perturbation Stochastic Approximation (SPSA) to optimize gamma and beta parameters. The parameter P is fixed at five across all experiments, with SPSA parameters adjusted to default values except for iteration (set to 10), learning rate (0.2), learning rate decay (0.602), and perturbation magnitude (0.2).

Evaluation metrics for both experiment types include:

- **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.

In the case of noisy experiments, the state vector 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.

2.1. QAOA implementation & measuring

The implementation used for the experiments in this paper and described below is based on Fahri et al. The experiments are not run on actual hardware, but simulated using QuantumSim, 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 four main steps of this paper's implementation (Figure 2) are described below:

- **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 beta value of the RX gate is multiplied by two.

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.

```python
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}")

    # Step 1: Initializing quantum circuit with N qubits, n = number of nodes
    n = len(nodes)
    circuit = Circuit(n)

    # Step 2: Bringing the whole circuit into superposition
    # Applying Hadamard gate (H) to all qubits
    for q in range(n):
        circuit.hadamard(q)

    # Step 3: Construct P layers, by alternating cost and mixer hamiltonian
    for i in range(p):

        # Step 3.1: Construct cost layer with parameter gamma[i]
        for edge in edges:
            circuit.cnot(edge[0], edge[1])
            circuit.rotate_z(gamma[i], edge[1])
            circuit.cnot(edge[0], edge[1])

        # Step 3.2: Construct mixer layer with parameter beta[i]
        for q in range(n):
            circuit.rotate_x(2 * beta[i], q)

    return circuit
```

Figure 2: Example implemenation of QAOA, implemented in QuantumSim

3. Results

3.1. Noisy experiments

3.2. Noiseless experiments

3.2.1. 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 3
5	2.50	3.17	3.61	4.00	4.00	4	4	Figure 5
6	3.00	4.48	5.00	5.00	5.00	5	5	Figure 7
7	3.00	4.71	5.48	5.69	5.87	6	6	Figure 9
8	6.00	6.65	7.58	7.56	7.30	7	10	Figure 11
9	8.00	8.83	10.14	10.37	10.56	11	13	Figure 13
10	8.00	9.02	12.86	13.00	13.00	13	13	Figure 15
15	24.50	27.54	32.95	32.56	32.30	32	35	Figure 17

3.2.2. 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 4
5	3.50	4.37	4.85	5.00	5.00	5	5	Figure 6
6	7.00	7.98	8.85	9.00	9.00	9	9	Figure 8
7	9.00	9.64	11.63	12.00	12.00	12	12	Figure 10
8	11.00	11.63	12.40	13.33	13.83	15	15	Figure 12
9	16.00	16.69	17.60	17.43	17.69	18	20	Figure 14
10	21.50	21.99	20.78	20.00	20.00	20	25	Figure 16
15	41.00	42.16	48.38	48.80	48.24	48	53	Figure 18

4. Discussion

5. Figures

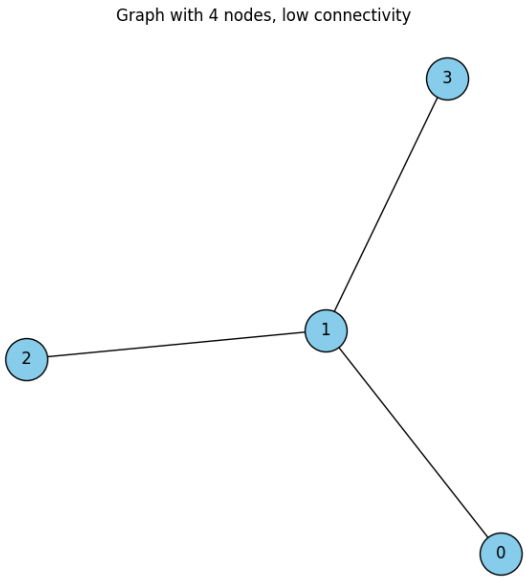

Figure 3: Graph with 5 nodes and low connectivity

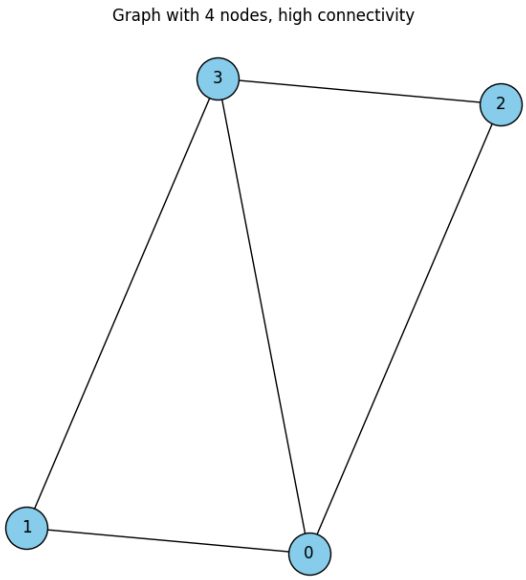

Figure 4: Graph with 5 nodes and high connectivity

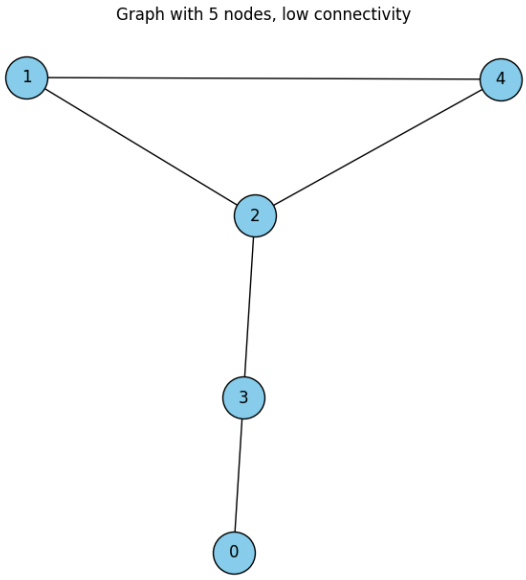

Figure 5: Graph with 5 nodes and low connectivity

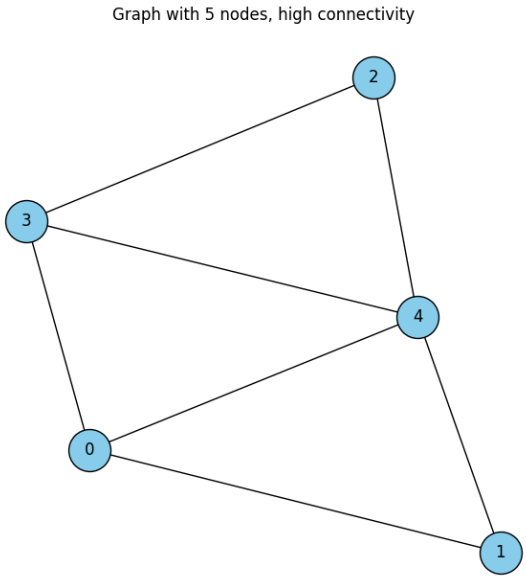

Figure 6: Graph with 5 nodes and high connectivity

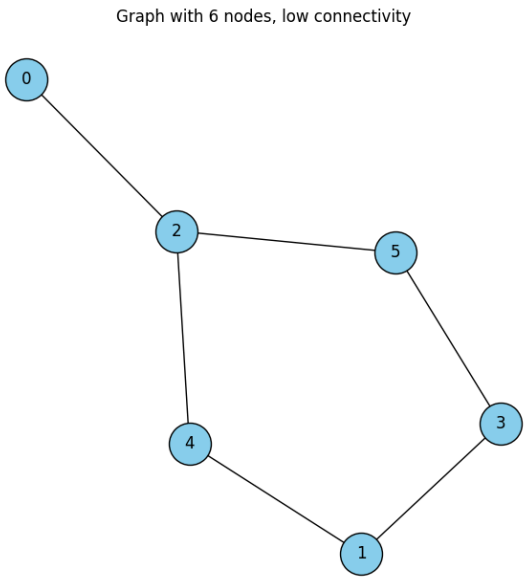

Figure 7: Graph with 6 nodes and low connectivity

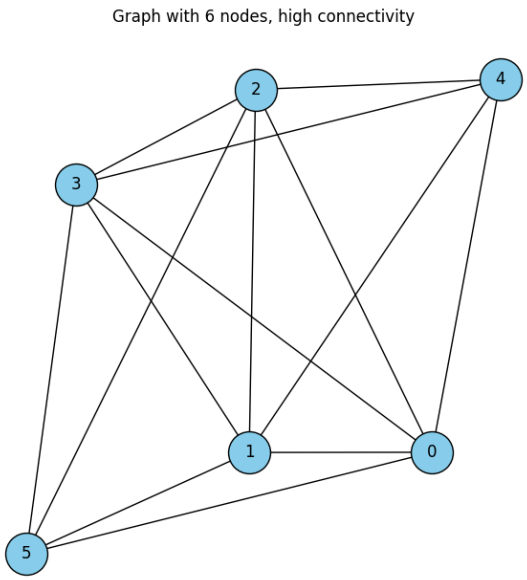

Figure 8: Graph with 6 nodes and high connectivity

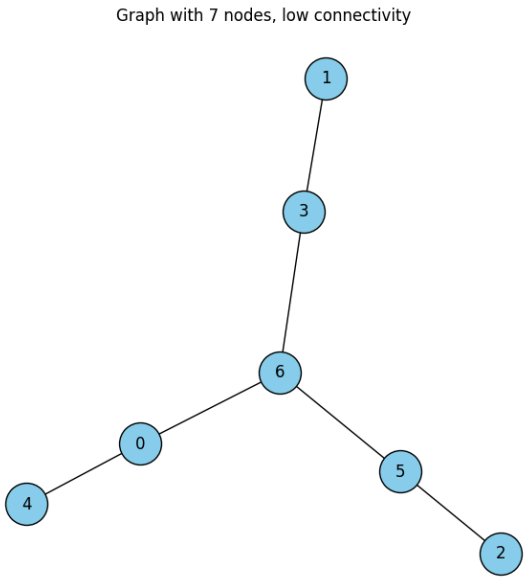

Figure 9: Graph with 7 nodes and low connectivity

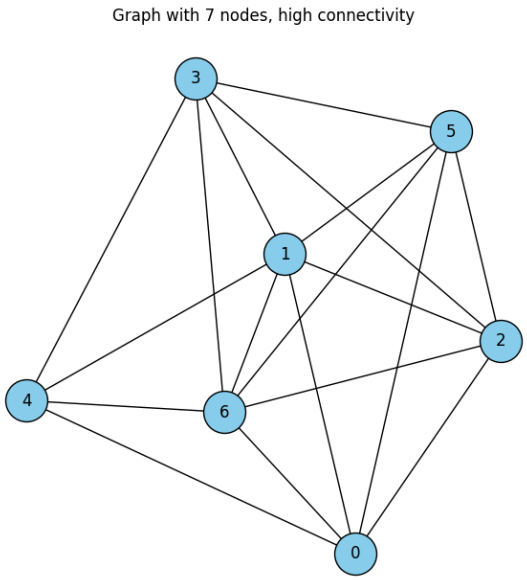

Figure 10: Graph with 7 nodes and high connectivity

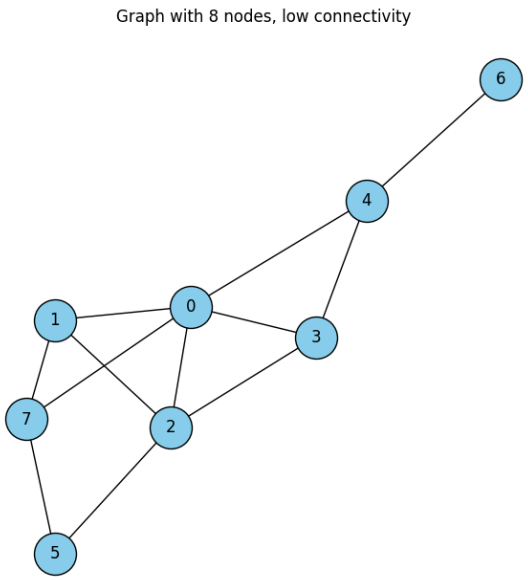

Figure 11: Graph with 8 nodes and low connectivity

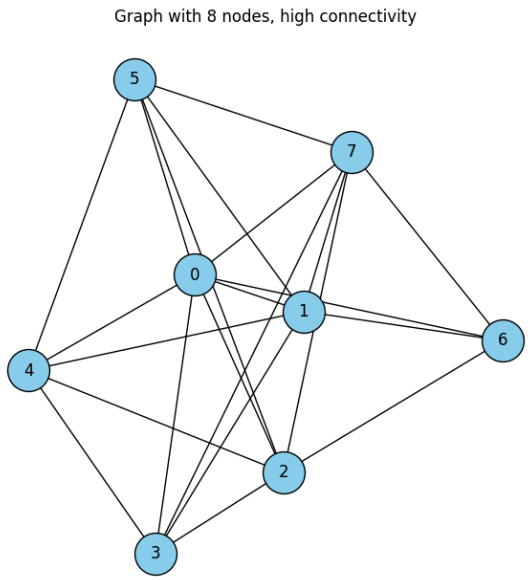

Figure 12: Graph with 8 nodes and high connectivity

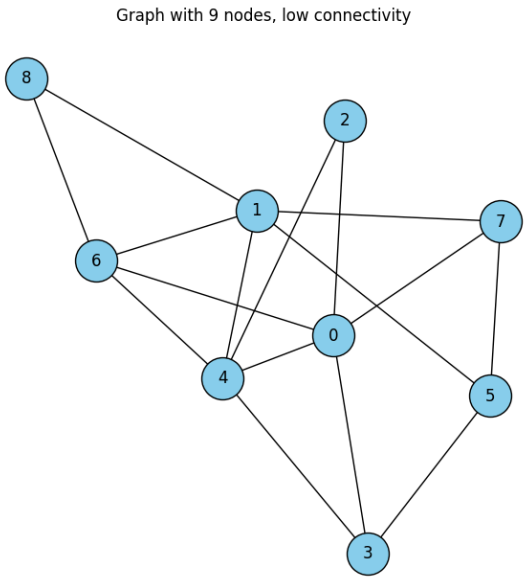

Figure 13: Graph with 9 nodes and low connectivity

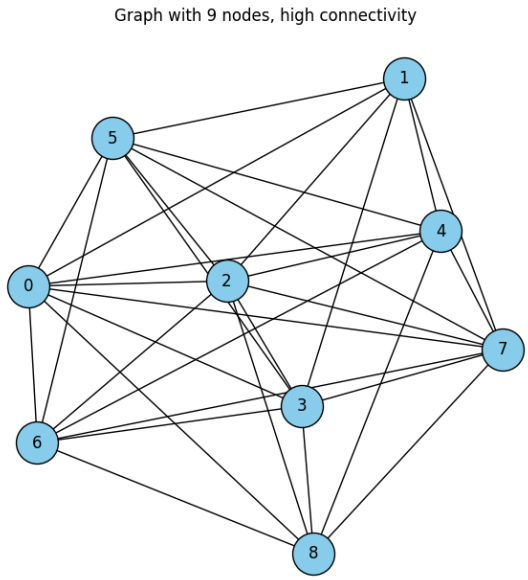

Figure 14: Graph with 9 nodes and high connectivity

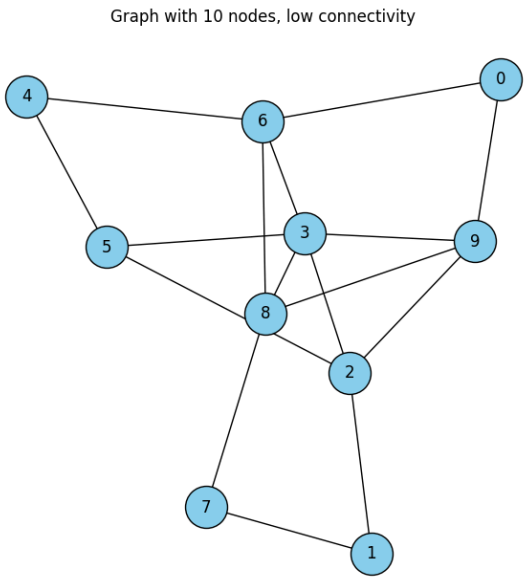

Figure 15: Graph with 10 nodes and low connectivity

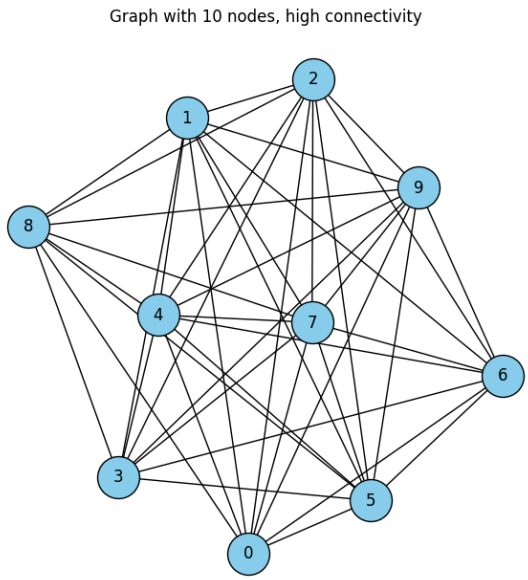

Figure 16: Graph with 10 nodes and high connectivity

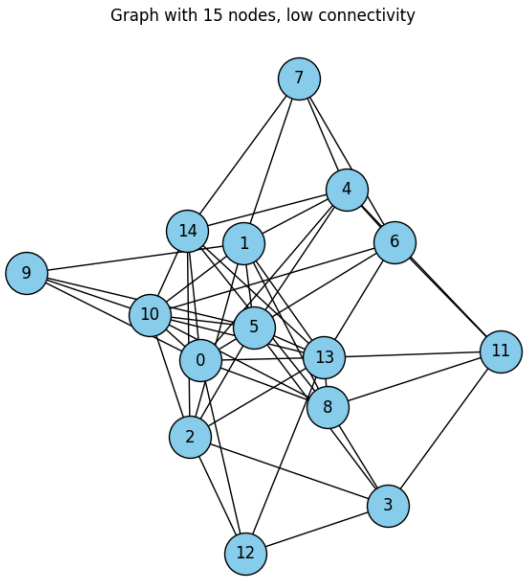

Figure 17: Graph with 15 nodes and low connectivity

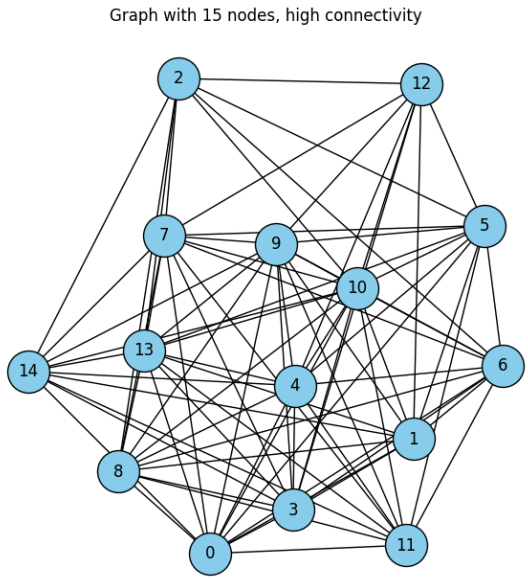

Figure 18: Graph with 15 nodes and high connectivity

References