Lab 4 QEC and QAOA

CS7400 Online
Georgia Institute of Technology
(Dated: Summer 2025)

For each question, you need to complete the associated `question*.py` file by implementing the required functions for that question. Submit `question1.py`, `question2.py`, and `question3.py` to Gradescope Lab-4.

I. BIT-FLIP QUANTUM ERROR CORRECTION

The goal of this question is to better understand quantum error correction. In 'question1.py', you need to complete the functions to implement bit-flip QEC algorithm. For a given 3-qubit input

$$|\psi\rangle = |q_2\rangle \otimes |q_1\rangle \otimes |q_0\rangle = |0\rangle \otimes |0\rangle \otimes (\alpha\,|0\rangle + \beta\,|1\rangle).$$

For simplicity we restrict α and β to be real numbers, and the qubits in $|\psi\rangle$ are already in qiskit default little-endian ordering. You need to create:

- The encoder circuit to encode the qubits.
- The decoder circuit to decode the circuit with error and recover q_0, q_1, and q_2 to their original states.

You must return `QuantumCircuit` objects for both encoder and decoder functions, and you are not allowed to measure any one of q_0, q_1, and q_2. Your implementation will be tested in the following way:

```
qc.initialize(psi)
qc = qc.compose(encoder)
inject_error(qc)
qc = qc.compose(decoder)
```

A. ONE Bit-Flip Error

To test your implementation, we will

1. Apply your encoder `encoder()` to a randomly generated $|\psi\rangle$ based on the equation above.
2. Inject one** bit-flip error: an X gate at random on q_0, q_1, and q_2.
3. Apply your decoder `decoder1()`.
4. Measure q_0, q_1, and q_2.

Your score is based on the number of tests you passed for 10 random initial states.

B. TWO Bit-Flip Errors

To test your implementation, we will

1. Apply your encoder `encoder()` to a randomly generated $|\psi\rangle$ based on the equation above.
2. Inject **two** bit-flip errors: two X gates at random on q_0, q_1, and q_2.
3. Apply your decoder `decoder2()`.
4. Measure q_0, q_1, and q_2.

Your score is based on the number of tests you passed for 10 random initial states.

II. PHASE-FLIP QUANTUM ERROR CORRECTION

The goal of this question is to better understand quantum error correction. In ‘question2.py‘, you need to complete the functions to implement phase-flip QEC algorithm. For a given 3-qubit input

$$|\psi\rangle = |q_2\rangle \otimes |q_1\rangle \otimes |q_0\rangle = |0\rangle \otimes |0\rangle \otimes (\alpha |0\rangle + \beta |1\rangle).$$

For simplicity we restrict α and β to be real numbers, and the qubits in $|\psi\rangle$ are already in qiskit default little-endian ordering. You need to create:

- The encoder circuit to encode the qubits.
- The decoder circuit to decode the circuit with error and recover q_0, q_1, and q_2 to their original states.

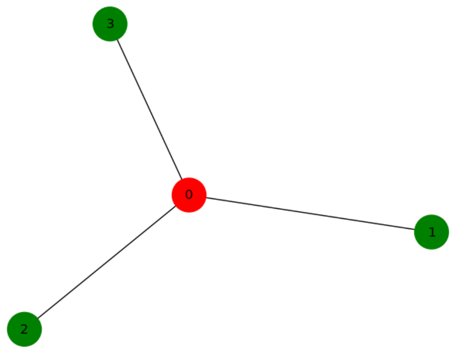
You must return `QuantumCircuit` objects for both encoder and decoder functions, and you are not allowed to measure any one of q_0, q_1, and q_2. Your implementation will be tested in the following way:

```
qc.initialize(psi)
qc = qc.compose(encoder)
inject_error(qc)
qc = qc.compose(decoder)
```

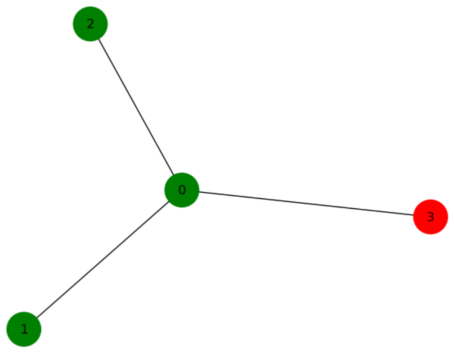
III. QAOA

The goal of this question is to better understand QAOA and see how it is used to solve the max cut problem. You need to complete the functions in the ‘QAOASolver‘ class in ‘question3.py‘.

- All input graphs are simple, undirected, and connected.
- There is no need to implement the `optimize_parameters` function, since we will just return the maximum cut value we find in `find_maxcut`, however you are encouraged to think how this can be done.
- The string of cut/partition of nodes in `self.qaoa_counts` should be treated as little-endian, so for a graph of edges `[(0,1),(0,2),(0,3)]`, the cut `0001` has value `3` while the cut `1000` has value `1`. See the example below.



(a) `0001`, Cut=3



(b) `1000`, Cut=1