IonQ 2025

Quantum Novices

The MAXCUT Ansatz

- Add Pauli-Z gates to expand search space

```
# Visualization will be performed in the cells below;

def build_ansatz(graph: nx.Graph) -> QuantumCircuit:

    ansatz = QuantumCircuit(graph.number_of_nodes())
    ansatz.h(range(graph.number_of_nodes()))

    theta = ParameterVector(r"$\theta$", graph.number_of_edges())
    for t, (u, v) in zip(theta, graph.edges):
        ansatz.cx(u, v)
        ansatz.ry(t, v)
        ansatz.rz(v, u)
        ansatz.cx(u, v)

    return ansatz
```

Challenge 2 Hamiltonian

$$
\begin{aligned}
M(x) = &\sum_{(v,w)\in E}(x_v + x_w - 2x_v x_w) - (\sum_{v\in V}(x_v) - \frac{n}{2})^2 \\
&\Rightarrow \frac{|E|}{2} - \frac{1}{2}\sum_{(v,w)\in E}(Z_v Z_w) - (\sum_{v\in V}(x_v) - \frac{n}{2})^2 \\
&\Rightarrow \frac{|E|}{2} - \frac{1}{2}\sum_{(v,w)\in E}(Z_v Z_w) - (\sum_{v\in V}(\frac{1}{2}(I - Z_v)) - \frac{nI}{2})^2 \\
&\Rightarrow \frac{|E|}{2} - \frac{1}{2}\sum_{(v,w)\in E}(Z_v Z_w) - (\frac{nI}{2} - \frac{1}{2}\sum_{v\in V}(Z_v) - \frac{nI}{2})^2 \\
&\Rightarrow \frac{|E|}{2} - \frac{1}{2}\sum_{(v,w)\in E}(Z_v Z_w) - (-\frac{1}{2}\sum_{v\in V}(Z_v))^2 \\
&\Rightarrow \frac{|E|}{2} - \frac{1}{2}\sum_{(v,w)\in E}(Z_v Z_w) + \frac{1}{4}(\sum_{v\in V}(Z_v))^2 \\
&\Rightarrow -\frac{1}{2}\sum_{(v,w)\in E}(Z_v Z_w) + \frac{1}{4}(\sum_{v\in V}(Z_v))^2
\end{aligned} \tag{1}
$$

Challenge 2 Hamiltonian

```python
def build_maxcut_hamiltonian(graph: nx.Graph) -> SparsePauliOp:
    """
    Build the MaxCut Hamiltonian for the given graph H = (|E|/2)*I - (1/2)*Σ_{(i,j)∈E}(Z_i Z_j)
    """
    num_qubits = len(graph.nodes)
    edges = list(graph.edges())
    num_edges = len(edges)

    pauli_terms = ["I"*num_qubits] # start with identity
    coeffs = [-num_edges / 2]

    for (u, v) in edges: # for each edge, add -(1/2)*Z_i Z_j
        z_term = ["I"] * num_qubits
        z_term[u] = "Z"
        z_term[v] = "Z"
        pauli_terms.append("".join(z_term))
        coeffs.append(0.5)

    for i in range(num_qubits):
        for j in range(num_qubits):
            z_term = ["I"] * num_qubits
            z_term[i] = "Z"
            z_term[j] = "Z"
            pauli_terms.append("".join(z_term))
            coeffs.append(0.25)

    return SparsePauliOp.from_list(list(zip(pauli_terms, coeffs)))
```

Results

1. Base score: 0.75519, Balanced score: 0.75519, Connected score: 0.0
2. Base score: 0.0001, Balanced score: 0.0001, Connected score: 0.02186
3. Base score: 0.00817, Balanced score: 0.00817, Connected score: 0.08848
4. Base score: 0.78876, Balanced score: 0.78606, Connected score: 0.00398
5. Pure max-cut: 4 out of 100000, Balanced max-cut: 4 out of 100000, Connected max-cut: 562 out of 100000
6. Base score: 0.00026, Balanced score: 0.70771, Connected score: 0.00018