Graph State Challenge

Problem Statement

Creating larger graph states with better fidelity Graph states entangle all involved qubits, and these entangled quantum states could be important for various applications, particularly those related to error correction, in the near future. The goal here is to create the largest graph states using the same benchmarking and error mitigation techniques that are also used to improve the individual quantum gates, ultimately looking for the best fidelity graph state with at least a 50% reduction in the infidelity as estimated by stabilizer measurement.

Our Solution

We focus on improving gate fidelity by manipulating the graph state function to minimize decoherence. We modify the way the circuits run to optimize and reduce noise error. We calibrate our measurements for the CTMP method using states with two-qubit excitations with custom parameters.

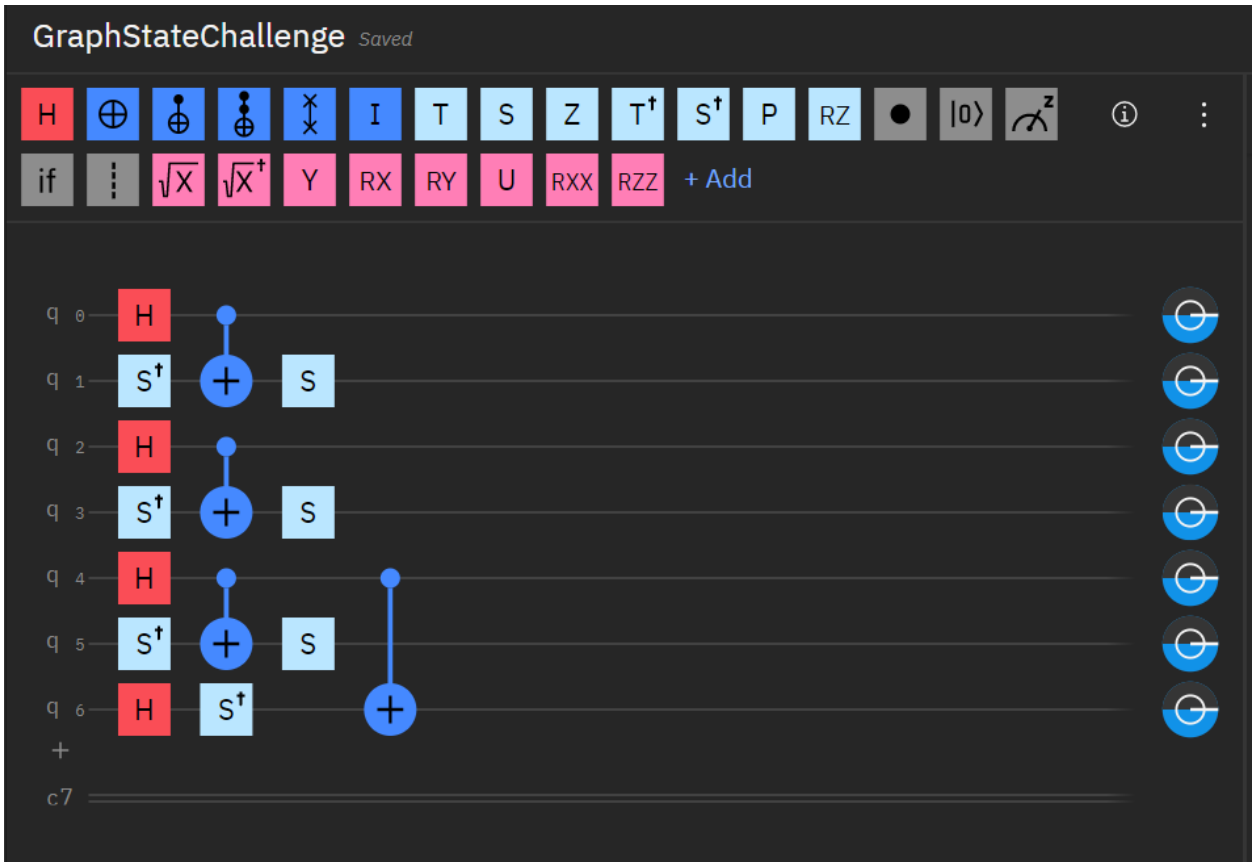
We have modified the Graph State by changing the gates in the raph_state_circuit to increase the fidelity from 0.782 ± 0.018 to 0.929 ± 0.003.

To gain higher fidelity states we modified the existing graph states circuit as the fidelity of the whole circuit depends upon the interaction of the qubits and the purity of the gates.

CZ gates are normally used for the graph states circuit as it creates a strong entanglement with a purity of O.5. We replaced the gates in the circuit with a combination of sdg, cnot and s gates on the two entangled qubits to create this circuit.

Given below are the circuits, stabilizer modifications, final results:

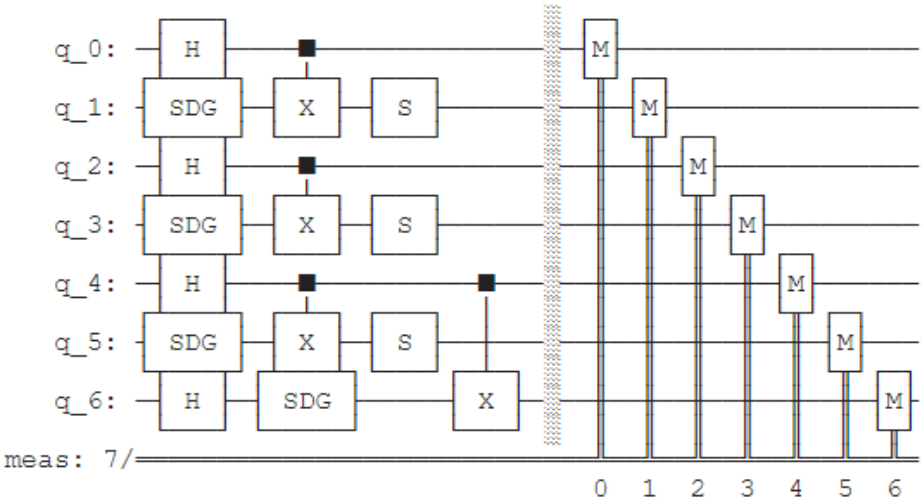
Circuit:



Stabilizer circuits:

```
[126] ## Append the stabilizer measurements to the graph state circuit
      stabilizer_circuits = [stabilizer_measure_circuit(stab, state_circuit)
                             for stab in stabilizers]

      stabilizer_circuits[0].draw()
```



Result: Higher gate fidelities in both error mitigation and CTMP error mitigation

```
[116] ## The final results

      print('Graph-state fidelity estimates')
      print('\nNo mitigation')
      print('F({}) = {:.3f} \u00B1 {:.3f}'.format(
          properties.backend_name, np.mean(F_nomit_backend), np.std(F_nomit_backend)))

      print('\nCTMP error mitigation')
      print('F({}) = {:.3f} \u00B1 {:.3f}'.format(
          properties.backend_name, np.mean(F_mit_backend), np.std(F_mit_backend)))
```

```
Graph-state fidelity estimates

No mitigation
F(ibmq_casablanca) = 0.797 ± 0.002

CTMP error mitigation
F(ibmq_casablanca) = 0.929 ± 0.003
```

```
[117] import qiskit.tools.jupyter
      %qiskit_version_table
```

Version Information

Qiskit Software	Version
Qiskit	0.25.1
Terra	0.17.1
Aer	0.8.1
Ignis	0.6.0
Aqua	0.9.1
IBM Q Provider	0.12.2
System information	
Python	3.7.10 (default, Feb 20 2021, 21:17:23) [GCC 7.5.0]
OS	Linux
CPUs	1
Memory (Gb)	12.71588134765625