QOSF Mentorship Program - Spring 2023
Error Correction - Perceval

Mentor - Victor Onofre
Mentee - Ramachandran SS

Overview

The project focuses on performing error correction to Discrete Variable Photonic Quantum Computing - Perceval Framework.

The second part of the project is to develop a photonic version of the QAOA algorithm in the Perceval Framework.

Code availability

The entire code is made available open sourced in GitHub.

Part 1 - Error Mitigation

Objective 1

- Performed Zero Noise Extrapolation (ZNE) error correction technique in Perceval.
- The ZNE technique was tested using a simulator (Qbraid) and a real device from Perceval.

Objective 2

- Proposed a ported version which can be used along with Mitiq Error correction framework.
- The ported version is tested using simulator (Qbraid).

Part 2 - QAOA algorithm

Objective 1

- Developed the gate based version of QAOA algorithm using Perceval.

Objective 2

- The project was tested in high performance simulator provided by Qbraid and on real device.

Section 1 - Error mitigation

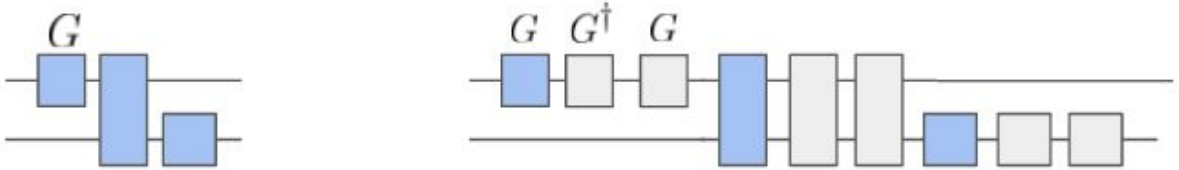
Part 1 - Error mitigation

Zero Noise extrapolation - It is an error correction technique where the expectation value of the circuit is computed with various noise levels and the value without the noise is extrapolated using these values.

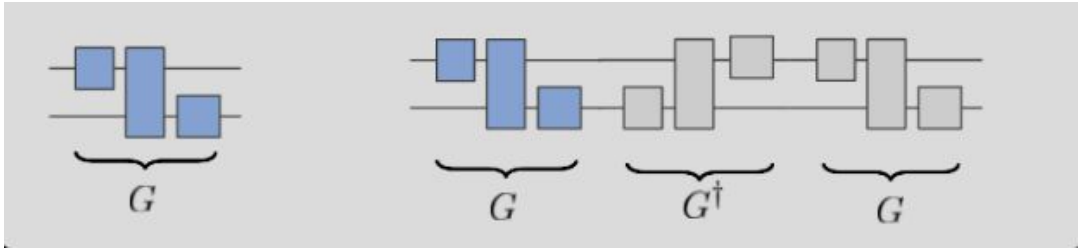
Unitary folding - We fold the circuit many times without changing the result by applying the mapping G -> GG†G. This can be applied locally or globally.

Unitary folding - types

Local folding



Global folding



Underlying concept behind ZNE

Assuming the circuit has a depth d and has a error value ε, on repeating the circuit n times with the depth becoming n*d, the newer error value will be x * ε, where x can be a float value.

We keep increasing the circuit repetition (n) with various values and save the expected measurement results for each experiment.

Finally, when the extrapolation is being done, the value will be obtained for noiseless version.

Implementation 1

We manually perform the extrapolation using polynomial functions. The ZNE is tested with a circuit having a single X gate with depth 1.

The photonic QC contains a single beam splitter with a reflectivity of 0. The photons are passed by the source and being received by a photo detector.

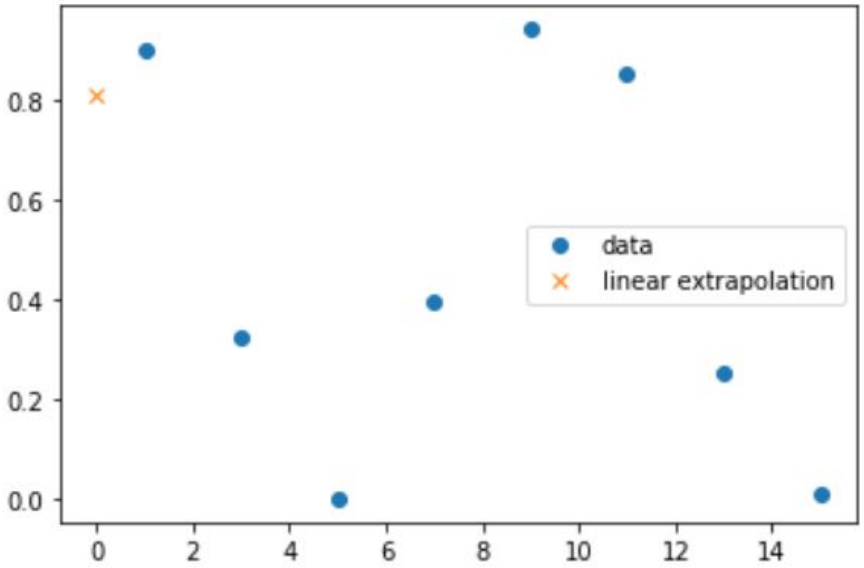
Noise model

The noise model for Perceval DV Photonic QC is being created manually as below.

For X gates, the reflectivity for the beam splitter is 0. So on varying this reflectivity of the beam splitter slightly by a random value, we can add noise to the circuit.

With a reflectivity value of 0.1, we have a X beam splitter with 99% accuracy.

Results - Implementation 1



The reflectivity is been set to 0.1 for X gate and the ZNE is performed.
The x axis denoted the repetition depth and y axis denoted the expected measurement value.

Implementation 2

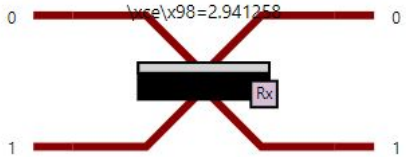
In this implementation, we perform the ZNE using the features available in the Mitiq package. Currently, Mitiq does not support error correction in photonic devices because of the operation principles compared to superconducting devices.

Implementation steps

- The key idea here is that, the photonic circuits developed in Perceval with the created noise model are converted back into the Qiskit circuit object.
- The photonic circuit with our noise model is converted to unitary matrix and passed on the qiskit library (since there is no direct conversion available yet).
- This new Qiskit circuit object is transpiled into a set of basic gates including Identity, RX, RY and CX (since mitiq does not support error correction for general unitary).
- This circuit object is passed to mitiq library to perform ZNE.

Implementation 2 - Results

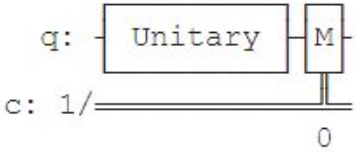
X gate wit error if 0.01%



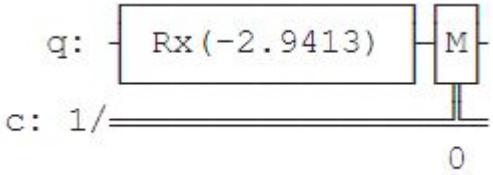
Unitary matrix

$$\begin{bmatrix} \frac{1}{10} & 0.994987i \\ 0.994987i & \frac{1}{10} \end{bmatrix}$$

Unitary circuit



Transpiled circuit



Mitigated result

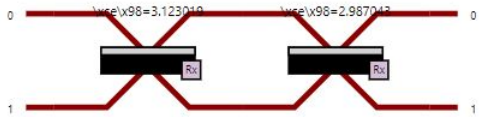
```
Mitigated result 0.989
```

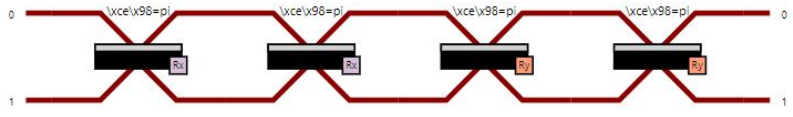
Testing

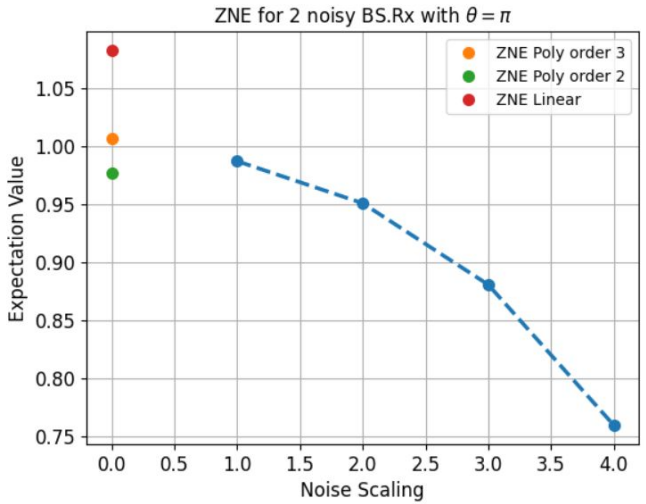
We test our implementation 1 using various random circuit configurations and formulated the results for those. The testing in done below

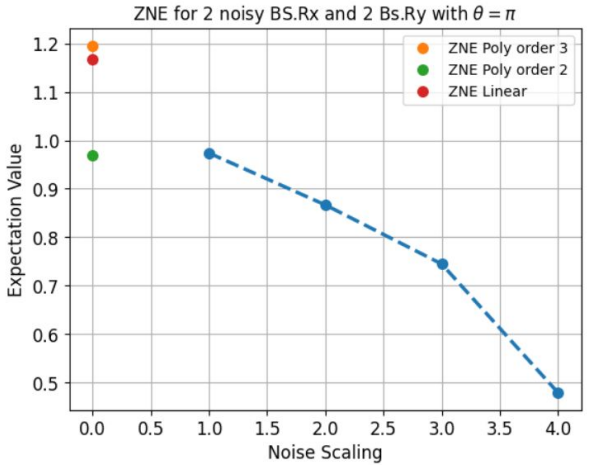
- Creating a beam splitter configuration which produces identity.
- Use the noise model which we created before. A small random angle is added to the gate angles to add noise.
- Perform ZNE using polyfit functions.
- It is also tested in QPU.

Results - testing

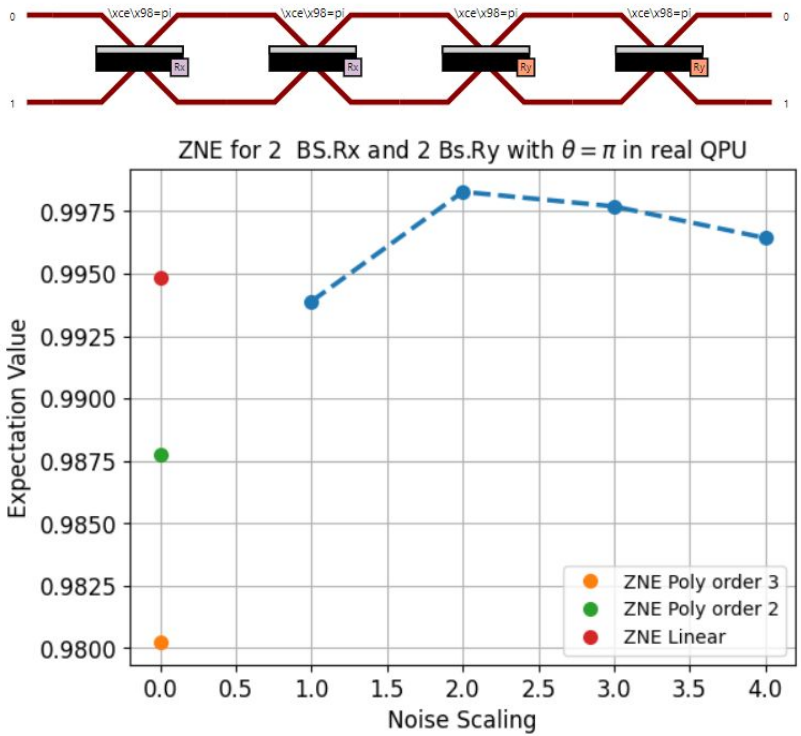








Results -QPU testing



The QPU's available in Perceval cloud have been used to test the technique. However, the simulation ran into some issues.

When we read the results, we get more counts than the number of samples that have been assigned. An issue has been raised in the Perceval forum and it seems that the issue is with the QPU control.
[Error in Sampler counts - Perceval Discourse (quandela.net)]

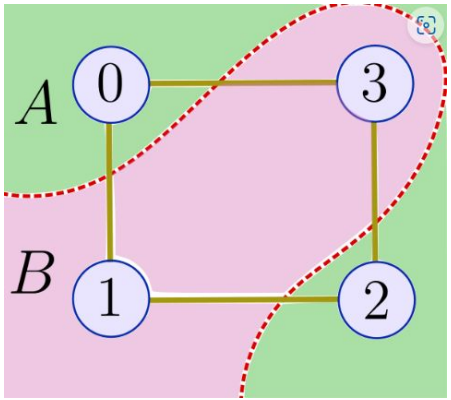
Section 2 - QAOA algorithm

Part 2 - QAOA algorithm

The Quantum Approximate optimization algorithm is widely used to solve combinatorial optimization problems on NISQ devices. Here, we will be mainly be using it to solve MaxCut problem.

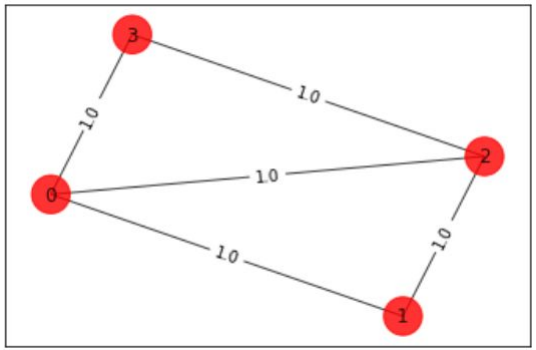
Maxcut problem

The aim of the maxcut problem is to maximize the number of edges in a graph that are cut by a given partition of the vertices into 2 sets.

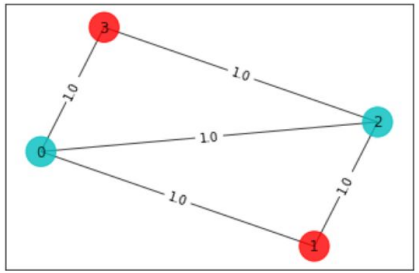


Brute force solution

For the below graph with 4 nodes, through brute force solution we can try all possible combinations of 2^4 and get the cost associated with them. The max value is the answer. This becomes computationally expensive for large graphs.



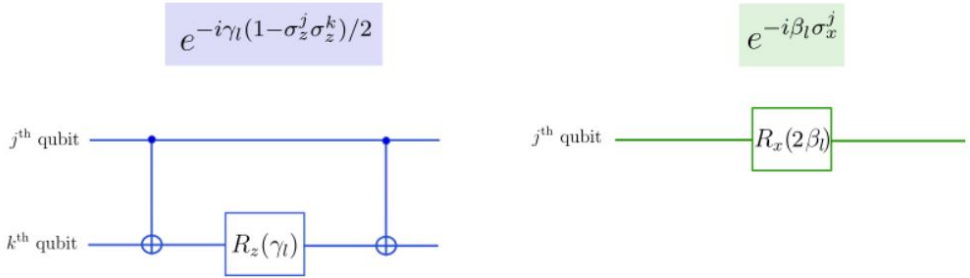

```
case = [0, 0, 0, 0] cost = 0.0
case = [1, 0, 0, 0] cost = 3.0
case = [0, 1, 0, 0] cost = 2.0
case = [1, 1, 0, 0] cost = 3.0
case = [0, 0, 1, 0] cost = 3.0
case = [1, 0, 1, 0] cost = 4.0
case = [0, 1, 1, 0] cost = 3.0
case = [1, 1, 1, 0] cost = 2.0
case = [0, 0, 0, 1] cost = 2.0
case = [1, 0, 0, 1] cost = 3.0
case = [0, 1, 0, 1] cost = 4.0
case = [1, 1, 0, 1] cost = 3.0
case = [0, 0, 1, 1] cost = 3.0
case = [1, 0, 1, 1] cost = 2.0
case = [0, 1, 1, 1] cost = 3.0
case = [1, 1, 1, 1] cost = 0.0

Best solution = [1, 0, 1, 0] cost = 4.0
```



Quantum Algorithm

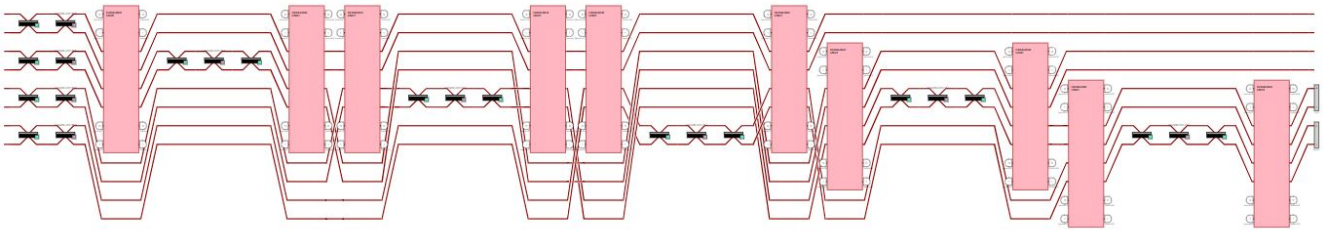
The cost function for quantum version is as below.

$C_\alpha = \frac{1}{2}\left(1 - \sigma_z^j \sigma_z^k\right),$ where alpha edge (j,k). The cost becomes 1 if j and k qubits have different z measurement values, representing separate partitions. The cost function is implemented using the below 2 circuits.



Perceval Implementation

We will be using the beam splitters with tunable angles (based on tunable reflectivities) and heralded CNOT gates for construction this circuit. The final circuit is shown below.



Implementation challenges

The main problem with the previous circuit is the computational cost associated with it. We tried with various instances from QBraid having 2vCPU_4GB mem, 4vCPU_8GB and 10vCPU_59GB. The simulation even on larger instances takes a very long time and times out.

References

References

Welcome to the perceval documentation!. Welcome to the Perceval documentation! - perceval 0.9.0 documentation. (n.d.). https://perceval.quandela.net/docs/

Tudor Giurgica-Tiron, Yousef Hindy, Ryan LaRose, Andrea Mari, and William J. Zeng. Digital zero noise extrapolation for quantum error mitigation. (2020). arXiv:2005.10921.

QAOA for MaxCut | PennyLane Demos

Thank you!