Quantum Resource Estimation of the Quantum Approximate Optimization Algorithm (QAOA)

QRISE2024

Special thanks to our sponsor Microsoft

Presentation by the Qu-Cats

Meet the Team

Nikhil Londhe

M.Sc. Physics

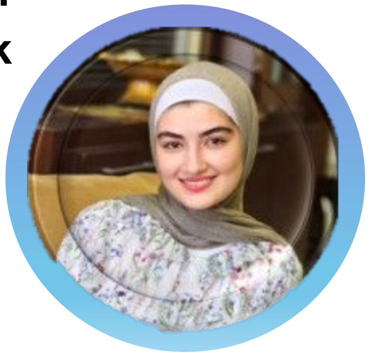
My research interests center on Quantum Learning Theory and its applications in Quantum Machine Learning, especially interested in the role of quantum feature maps in developing efficient quantum kernel algorithms.

Katie Harrison

B.S. Physics, Quantum Program Admin

Interested in quantum algorithms, specifically in graph theory and combinatorial optimization. Currently works at the Chicago Quantum Exchange, pursuing personal research interests alongside professional development.

Sarah Dweik



Vestibulum congue tempus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.

Muhammad Waqar Amin

B.S. Physics, Senior Researcher

Interested in developing novel quantum hardware for quantum information processing.

Presentation Agenda

Our goal is to provide reasonable motivation and background on what QAOA is, how it works, and provide based on Microsoft’s resource estimator quantum parameter values for the best results

- Project Selection
- Background
- Q# implementation of QAOA
- Resource Estimation of QAOA
- Conclusion and recommended next steps

Project Selection

Project selection took into account topic novelty, applicability, and solvability

1. Microsoft's Resource Estimator (RE) has yet to be utilized to predict optimal resource allocation for QAOA implementation on quantum computers
   - Our project provides resource benchmark recommendations based on Microsoft's RE
2. QAOA shows promise for improving the runtime of QUBO problems that are NP
   - We look at how the Resource Estimations scale with increasing input size
3. QAOA, Q#, and the Resource Estimator were all relatively new to many of us
   - Given the time constraint and amount we needed to learn, it was important that our algorithm have a plethora of documentation

Quantum Approximate Optimization Algorithm (QAOA)

Background and Overview

QAOA is a Quantum Algorithm used to solve combinatorial optimization problems

- Draws inspiration from the *Adiabatic Theorem*
- Encodes the cost function of an optimization problem as $\mathbf{H}_c$ and alternates it with a mixer $\mathbf{H}_m$
- Finds Approximate Solutions by evolving $\mathbf{H}_c$, $\mathbf{H}_m$ slowly enough so that we can treat $\mathbf{H}_c$ and $\mathbf{H}_m$ as hermitian

High-level Overview

- Prepare a quantum state that encodes all potential solutions
- Evolve the state using ***p*** layers
- Tune the adjustable parameters ($\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$) using an optimizer to find the optimal solution

Quantum Approximate Optimization Algorithm (QAOA)

Implementation Walkthrough

1. **Formulate the optimization problem as a QUBO**
2. **Express the QUBO cost function as a Hamiltonian H_c and add a mixer Hamiltonian H_m**
3. **Run the QAOA circuit which consists of p layers of the unitary operations U_c , U_m alternating**
4. **Use the results of the QAOA circuit to optimize the parameters using a classical optimizer (COBYLA)**

Number Partition Cost function

$$Q = \left(B - 2\sum_{i=1}^{n} e_i(1 - x_i)\right)^2$$

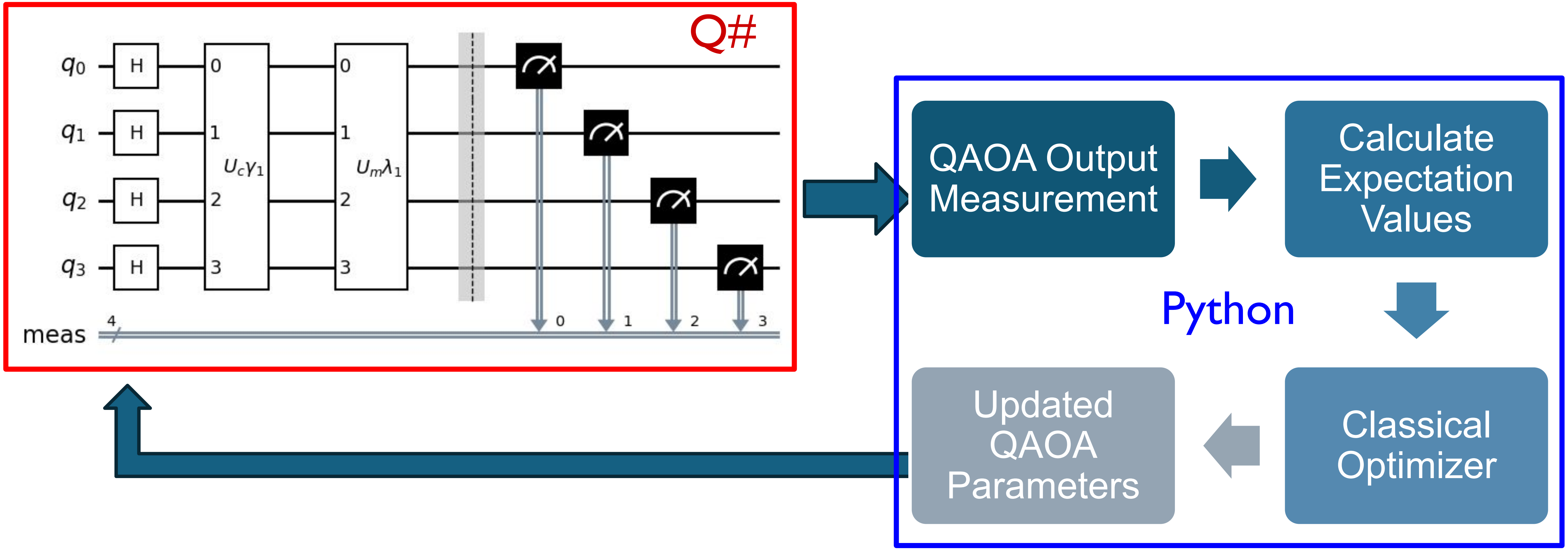
QUBO Cost function

$$C(x) = \sum_{i,j=1}^{n} x_i Q_{ij} x_j + \sum_{i=1}^{n} c_i x_i = x^T Q x + c^T x$$

Cost & Mixer Hamiltonian's

$$H_C = \sum_{i,j}^{n} \frac{1}{4} Q_{ij} Z_i Z_j - \sum_{j}^{n} \frac{1}{2}\left(c_i + \sum_{i}^{n} Q_{ij}\right) Z_i \quad , \quad H_M = \sum_{i}^{n} X_i$$

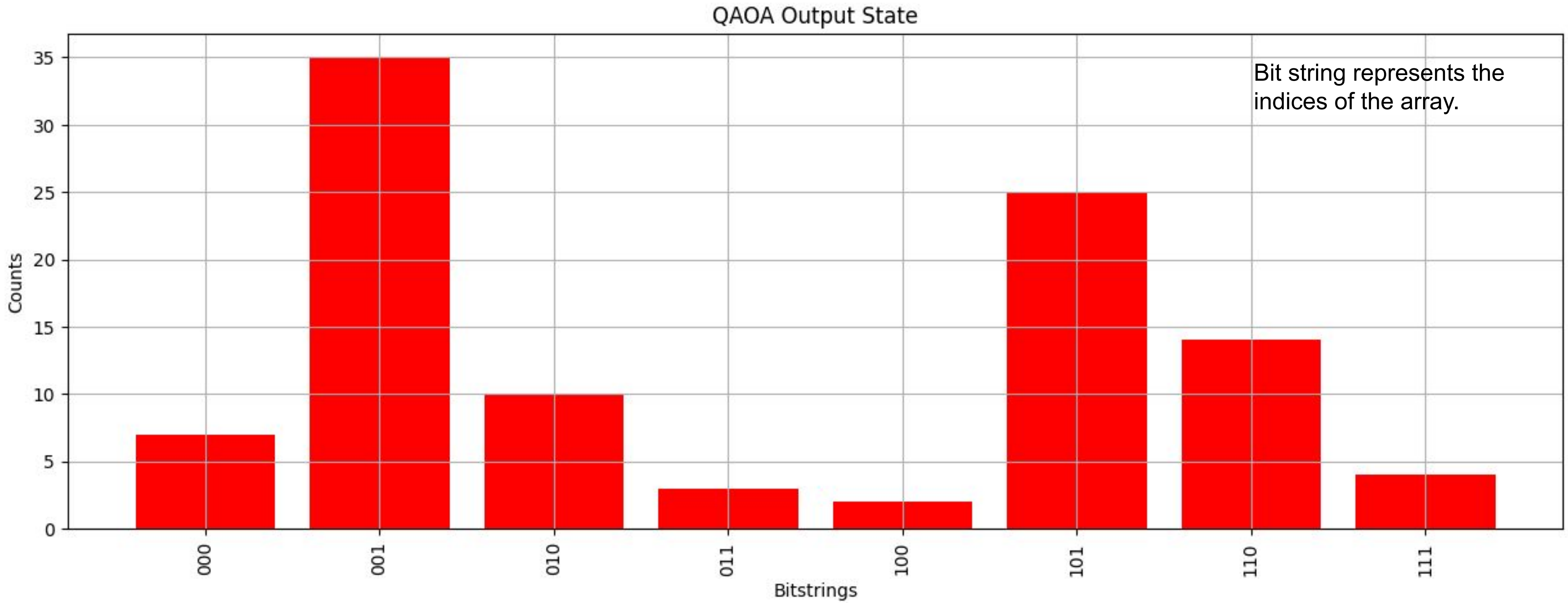
QAOA Quantum Circuit and Classical Optimizer



QAOA Test Example Results

Our sample problem is partitioning a list ([1,5,6]) into two lists such that the difference between the sums of the two partitions is minimum.

The graph shows the frequency output of our QAOA circuit

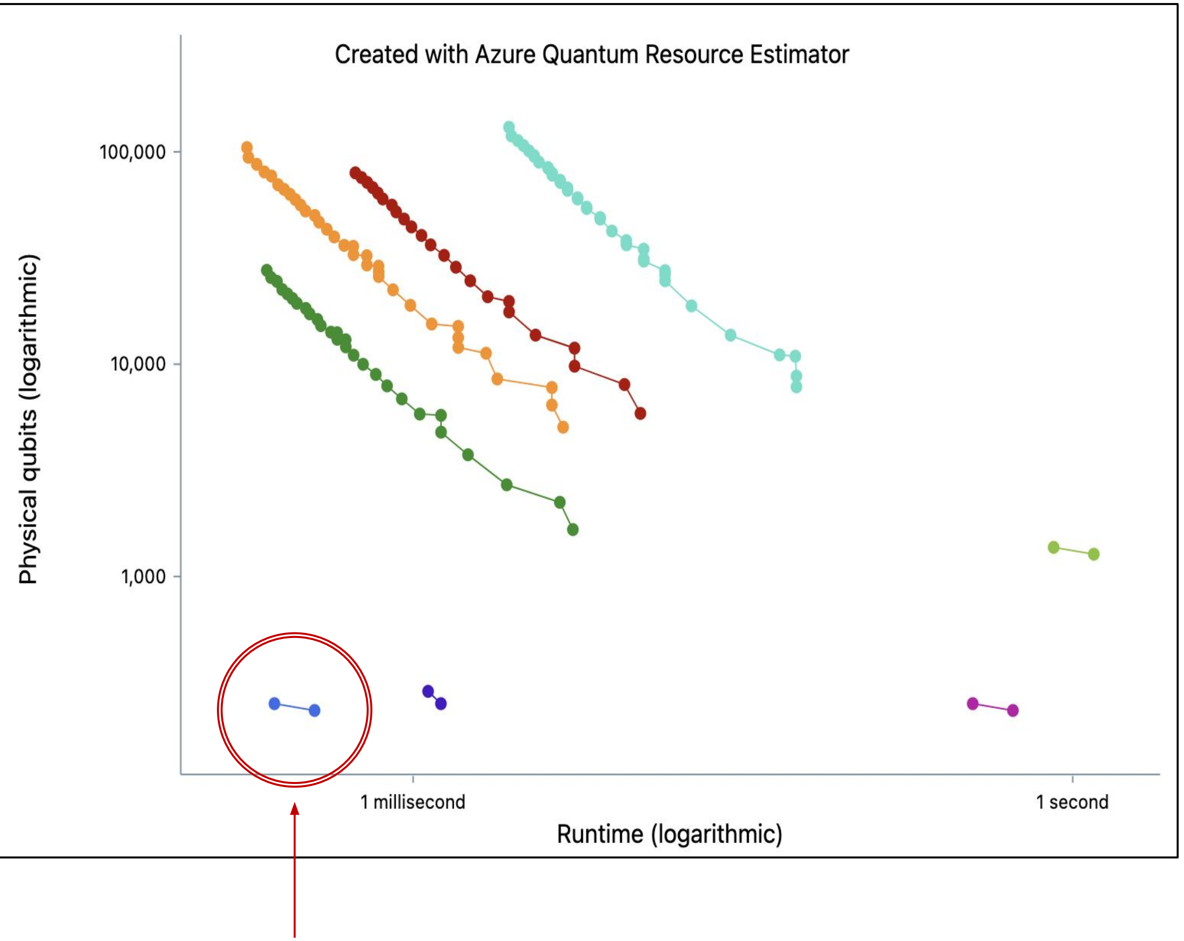


A = [1,5,6]
S = ‘001’ → [6] → Sum(S) = 6
S/A = [1,5] → Sum(S/A) = 6

The difference in sums is, abs[Sum(S) - Sum(S/A)] = 0

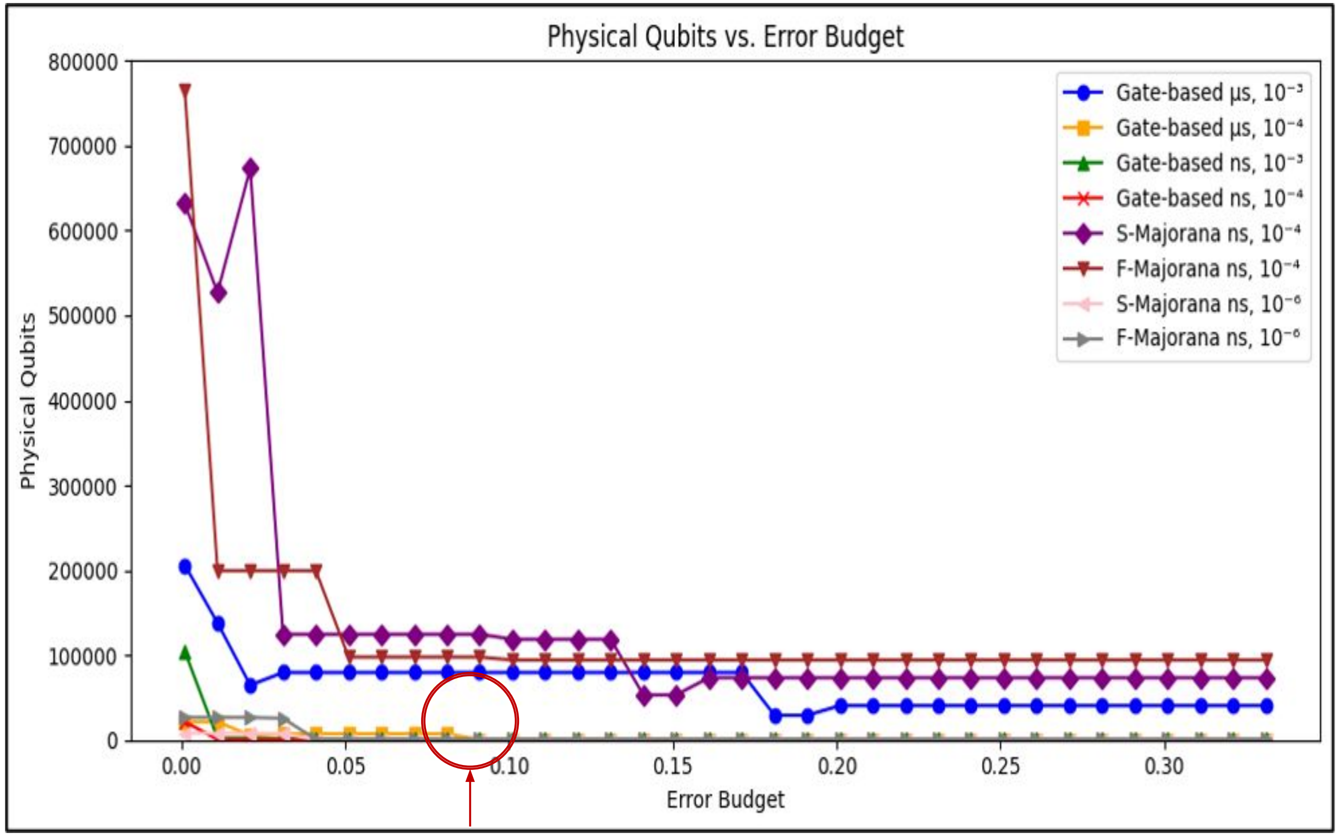
Results

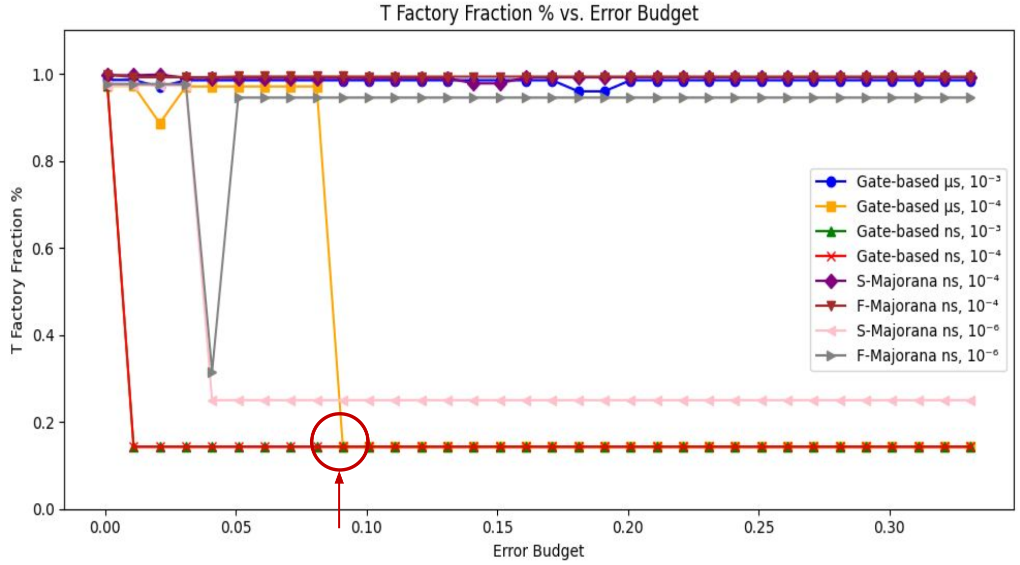
The Trapped-Ion based quantum computer stands out as the top performer, achieving optimal results with minimal physical qubits, significantly reduced processing time, and a lower T-factory ratio

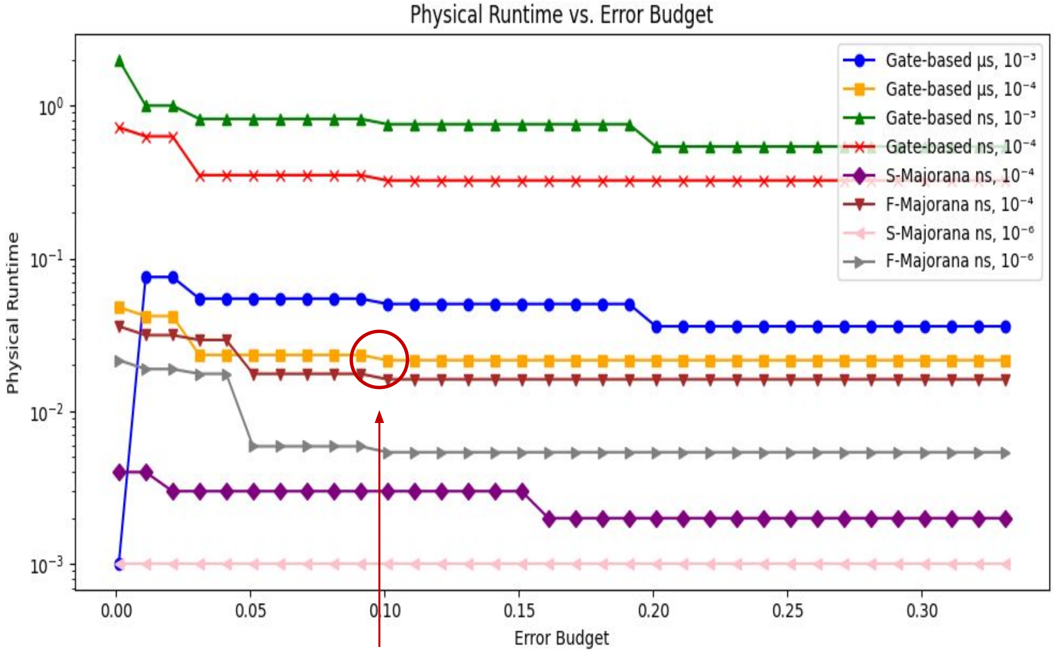


Run name	T factory fraction	Physical qubits	Runtime	rQOPS
Gate-based ns, 10^{-3}	98.52 %	79,576	546 microsecs	4,285,715
Gate-based ns, 10^{-4}	14.29 %	252	234 microsecs	10,000,000
Gate-based us, 10^{-3}	14.29 %	1,372	819 millisecs	2,858
Gate-based us, 10^{-4}	14.29 %	252	351 millisecs	6,667
S-Majorana ns, 10^{-4}	99.10 %	130,536	3 millisecs	857,143
F-Majorana ns, 10^{-4}	99.40 %	104,664	176 microsecs	13,333,334
S-Majorana ns, 10^{-6}	25.00 %	288	1 millisecs	2,000,000
F-Majorana ns, 10^{-6}	97.74 %	27,664	216 microsecs	13,333,334

Determining the optimal trade-off between performance and error budget







The Graphs show a comparison of the Error Budget vs.

1. **The number of Physical Qubits required**
2. **The T Factory Fraction %**
3. **The Physical Run Time**

Recommendation

Trapped Ion

Quantum Computer machine type

Surface Code

Error Correction Scheme

9%

Total Error Budget allowed

Even Split

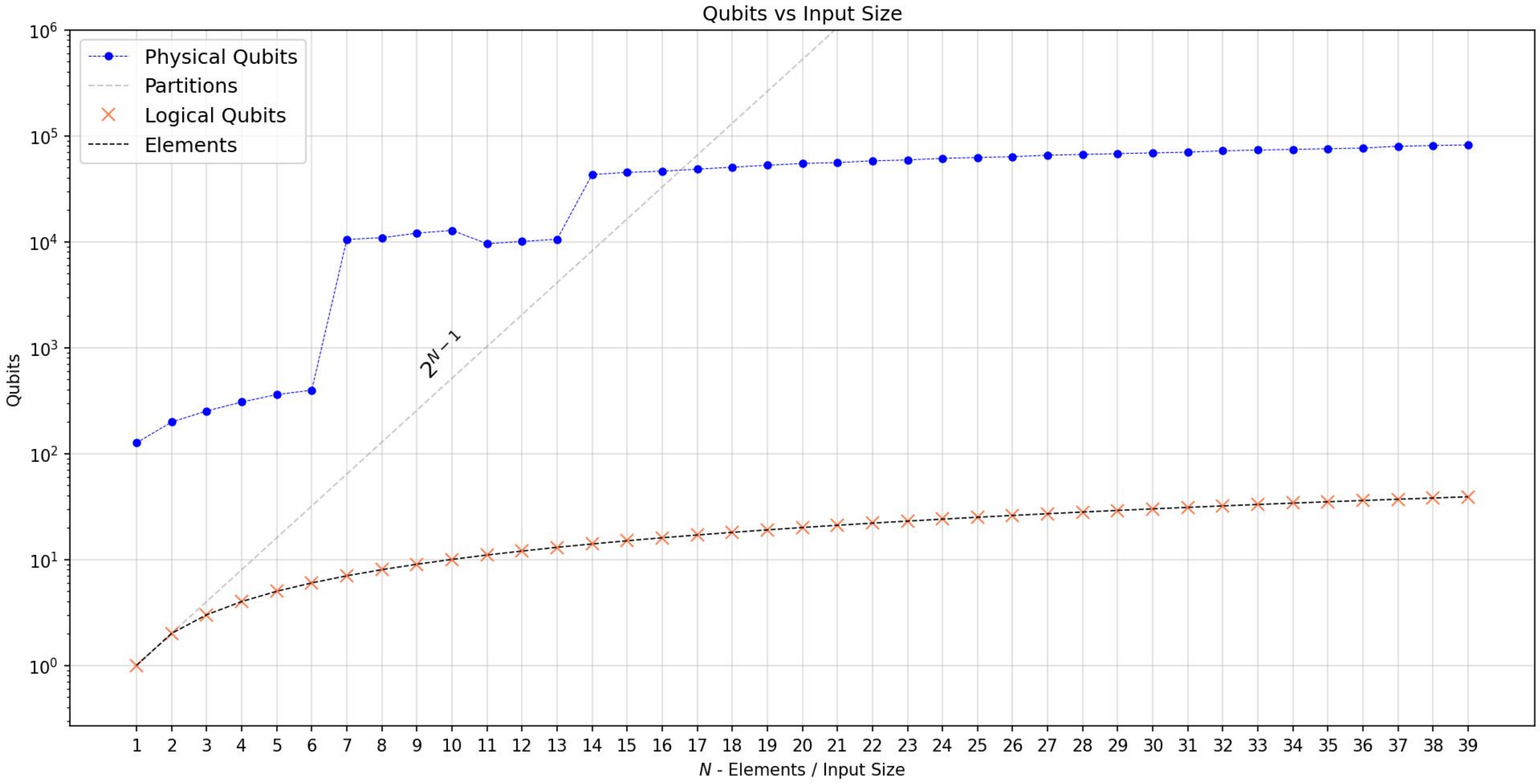
Error Budget split evenly 3 ways

- Implementation of logical qubits
- Production of T states
- Synthesis of rotation gates

If you configure your parameters as described and attempt the number partition problem using the QAOA method for a list of three elements, the resource estimations will indicate the necessity of 252 physical qubits and 234 microseconds of runtime. Moreover, approximately 14.29% of the T factories implemented will be fault-tolerant, and you can reliably compute 10 million quantum operations per second.

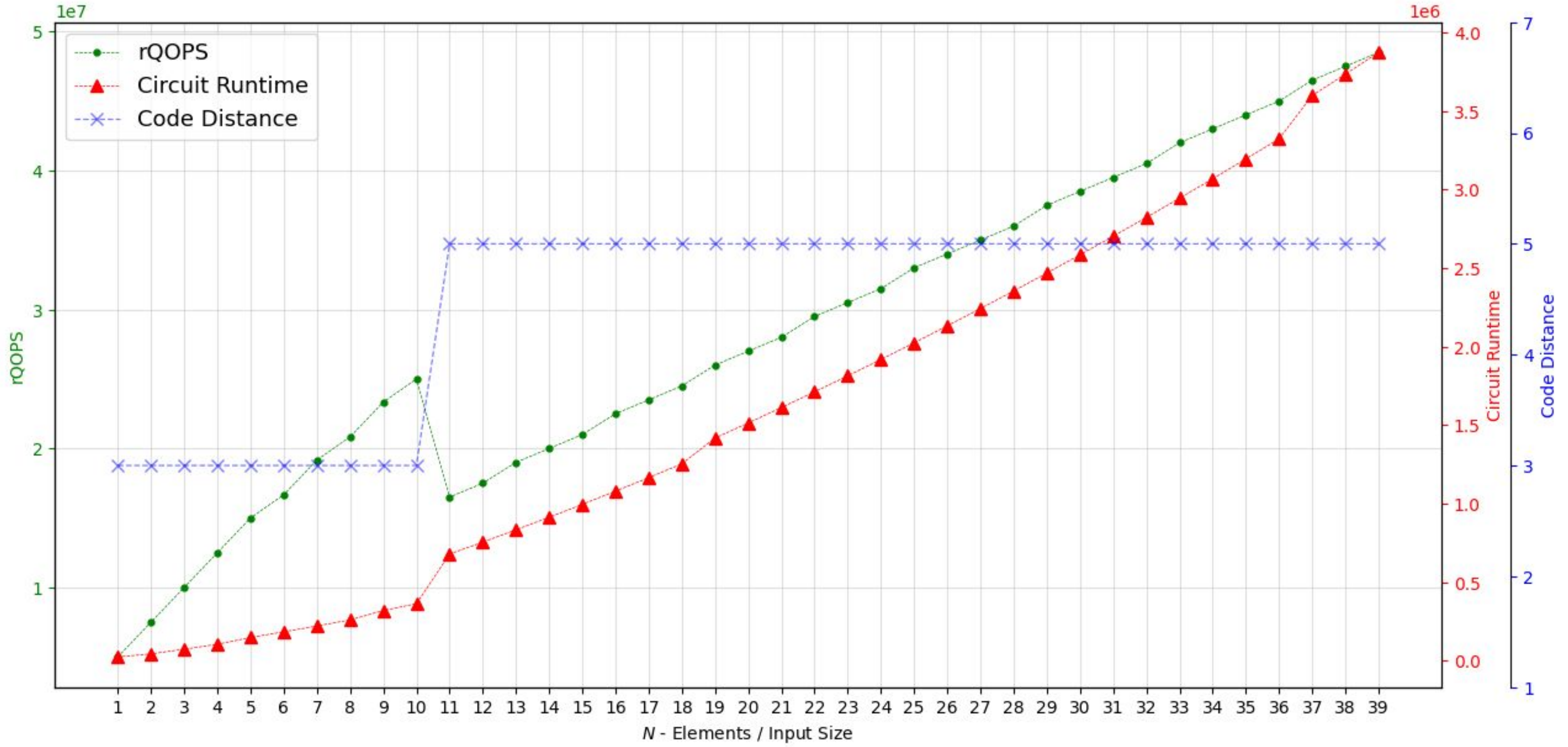
Resource Estimation for varying input sizes

- Gate_based ion quantum computer
- Error Budget: 0.09
- QECC: Surface Code
- Depth: 1



Resource Estimation for varying input sizes

- Gate_based ion quantum computer
- Error Budget: 0.09
- QECC: Surface Code
- Depth: 1



Conclusion

- Successfully created a working implementation of QAOA using Q# and Python
- Our project pinpoints specific QPU parameters that minimize resources for effective QAOA execution
- Created a project that can be used for benchmarking different optimization problems that use QAOA

Possible Next Steps

- Explore QAOA circuit resource estimates with varying layer counts
- Conduct further study of the Resource Estimator parameters
- Explore additional Optimizers both Classical and Quantum
- Survey other variants of QAOA

THANKS

Presented by the Qu-Cats

Special thanks to QRISE and Microsoft for supporting this project

References

[1] Lotshaw, P.C., Nguyen, T., Santana, A. *et al.* Scaling quantum approximate optimization on near-term hardware. *Sci Rep* **12**, 12388 (2022). https://doi.org/10.1038/s41598-022-14767-w.
[2] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph], (arXiv:1411.4028), November 2014.