QAOA for Low Autocorrelation Binary Sequences

Keshav Deoskar, Angela Wu, Kailash Ranganathan, Evan Wang

Background

Motivation

- The accuracy of QAOA is sensitive to the Ansatz one begins with, and the standard ansatz of $|+\rangle^{\otimes N}$ is not always optimal. However, Zhu et. al. (2022) introduced problem-specific “ADAPT QAOA” with efficient ansatz.

- Shaydulin et. al. (2024) show evidence of QAOA possessing scaling advantage over classical algorithms for the Low Autocorrelation Binary Sequence.

- We aimed to display Quantum Supremacy, while gaining experience with this foundational algorithm and its many variants.

Challenges

Pivot

Initially implemented ADAPTIVE QAOA with custom ansatz and a family of mixer gates, drawing inspiration from work by Zhu et. al (2022), however:

- N > 19 intractable
- Incorrect minimum energies, local minima
- Variation in minimum energy over trials

Likely sources of error/inefficiency were

- Presence of Vastly more mixer gates due to quartic nature of Hamiltonian (as compared to quadratic Ising Hamiltonian in the case of, say, MaxCut)
- Lack of symmetry-based space-cutting in our implementation at that time

Pushed us to Merit-Figure based implementation!

Strategy

Minimizing the “Sidelobe” = Maximizing Merit

- Implemented QAOA as presented in Shaydulin et. al. (2024)
- For a given set of spins $s_i \in \{-1, 1\}$, minimizing the “Sidelobe” Energy $\boldsymbol{\varepsilon}$

$$\mathcal{E}_{\text{sidelobe}}(\boldsymbol{s}) = \sum_{k=1}^{N-1} \mathcal{A}_k^2(\boldsymbol{s}), \qquad \mathcal{A}_k(\boldsymbol{s}) = \sum_{i=1}^{N-k} s_i s_{i+k}$$

is equivalent to maximizing the ‘Merit Figure’

$$\mathcal{F}(\boldsymbol{s}) = \frac{N^2}{2\mathcal{E}_{\text{sidelobe}}(\boldsymbol{s})}$$

Strategy

- Use QAOA to obtain k-many low energy bitstrings to use as see for MTS algorithm.

- States produced by QAOA become “better” with increase in number of layers (p) as output approaches true ground state with p□∞.

Optimizations

QAOA by itself ends up being quite computationally expensive as the population-proposing subroutine for MTS. A few custom optimizations we did both on the cuda-q and vanilla python side:

1. Utilized the two symmetries of LABS to reduce the state-space
    a. Fixed first qubit to $|0\rangle$ in the QAOA kernel - this breaks the bit-flip symmetry and halves the search space from 2^N to 2^{N-1}
2. Optimized classical MTS algorithm
    a. In Tabu-search energy calculation, cache computed energy of sequences and dynamically adjust energy upon bitflips instead of naive recalculation — algorithmically, this gives a O(N) vs O(Nˆ2) routine
    b. We vectorized/batched as much of our code with numpy as possible, which empirically also improved our performance

Verification

1. Symmetry

LABS sequences exhibit symmetries under transformations which don't change their energy

1. Negation (1111 -> 0000)
2. Reflection (1000 -> 0001)

By asserting that the energies of these two sequences are equal in our tests, we can ensure our AI code is correct

Symmetric operators reduce our search space - greater efficiency

2. Physical Accuracy

The laws of physics prohibit negative energy

Lower bound: 0 Upper bound: sequence of all 1s or all 0s

Our unit tests ensure that these bounds are never violated – our code never outputs physically impossible data

Quantum circuit ⇔ Hamiltonian classical correspondence

- Unit tests ensure that <E> of a given LABS sequence computed classically and through the quantum circuit are equivalent

3. Checking against known optimal solutions

We verified that our answers were never below the pre-existing optimal Energy values for any given N we tested, by referring to Mertens' Exhaustive Table (1996).

Table 1 Ground states of the Bernasconi-model for $3 \leq N \leq 48$. Sequences are written in run-length notation: Each figure indicates the number of consecutive equal signed elements.

N	$E_{\min}$	sequence
3	1	21
4	2	211
5	2	311
6	7	1113
7	3	1123
8	8	12113
9	12	42111
10	13	22114
11	5	112133
12	10	1221114
13	6	5221111
14	19	2221115
15	15	52221111
16	24	225111121
17	32	252211121
18	25	441112221
19	29	4111142212
20	26	5113112321
21	26	27221111121
22	39	51221111233
23	47	212121111632

Results

Scaling of Various Algorithms

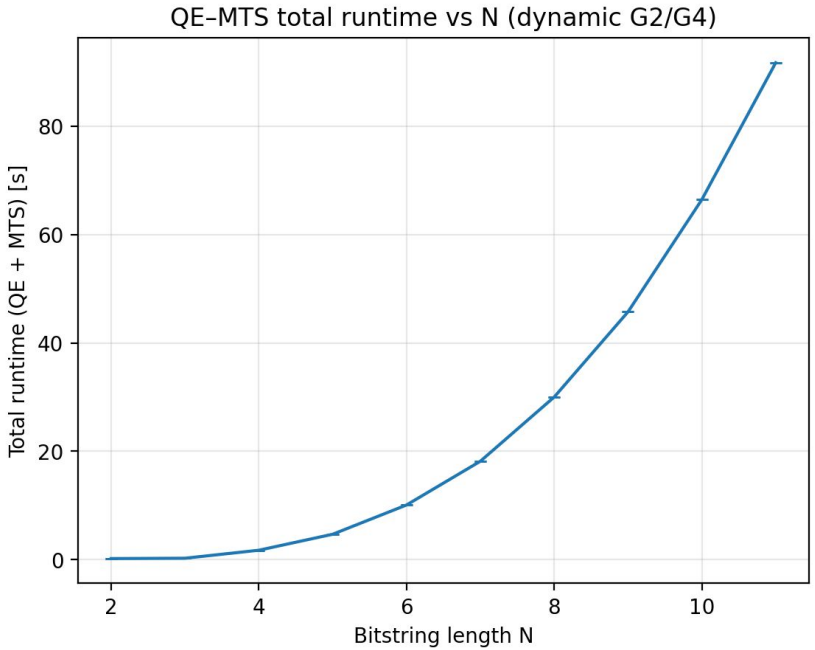
Because the GPU is only a classically parallelized acceleration of quantum simulation, it still exhibits the exponential runtime scaling as we increase bitstring length (aka # of qubits in our Hamiltonian).

Moreover, due to VRAM limitations (1xA100 = 80 GB), all our experiments are capped at N = 32–33 qubits.

Cuda-Q CPU runtime vs N

When we set the target backend to qpp-cpu (CPU-based statevector simulator), even small values of N exhibit long runtimes) →

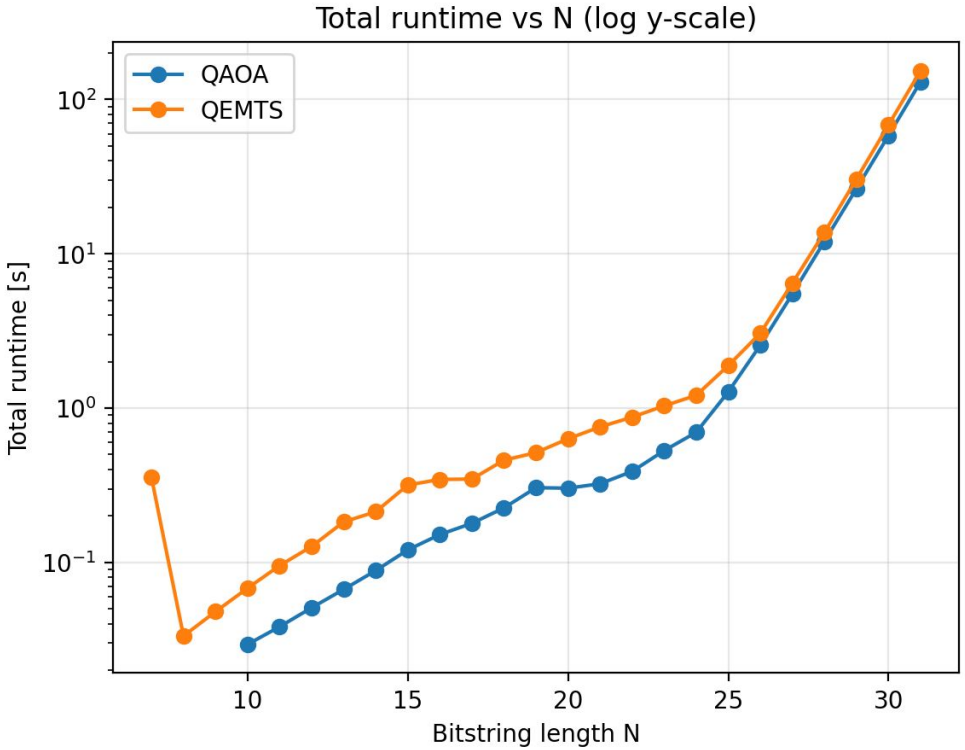
This is to be expected – we can get significant gains from pushing parallelizable matmul operations in quantum circuit simulation to GPU



Scaling of QE-MTS vs QAOA

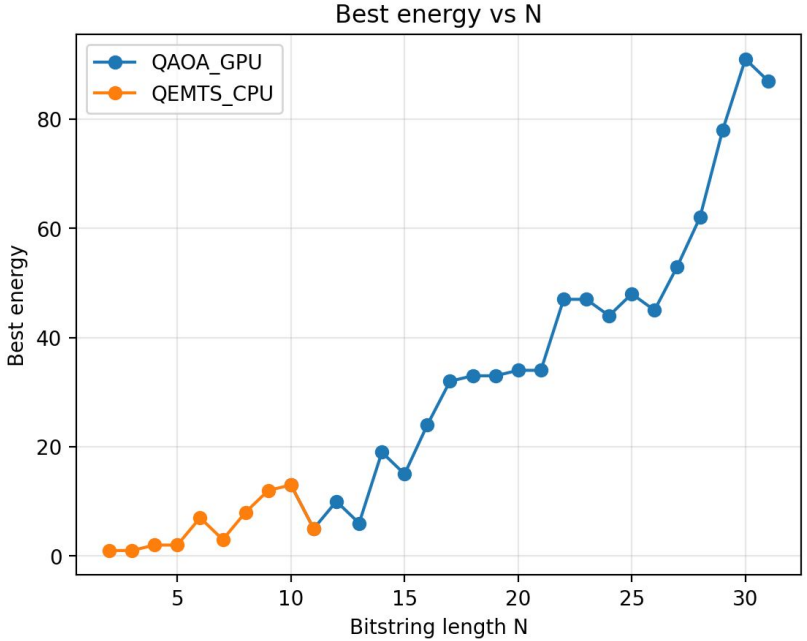
When we move to GPU, our runtime improves significantly (100x), but still exhibits similar exponential scaling laws.

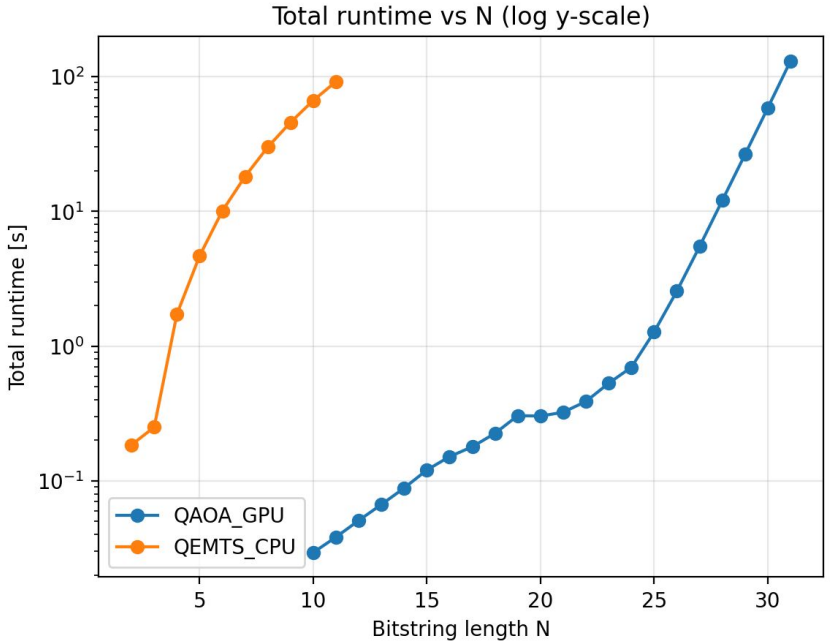
Our implementation of QAOA is marginally faster than the default QE-MTS proposed in [4], shown →



Comparison — CPU vs GPU Runtime + Energy

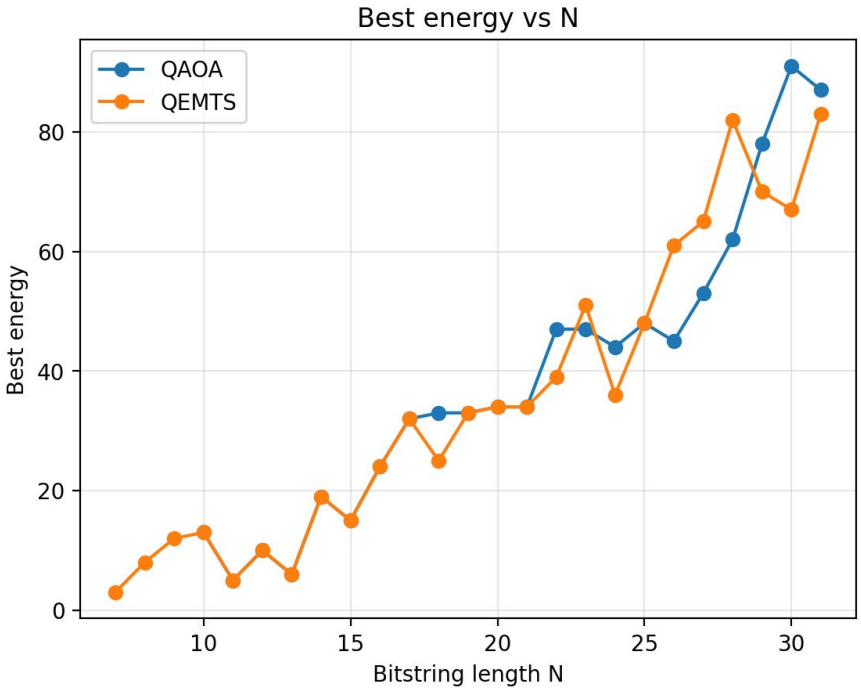
Note – even for small values of N, CPU simulation becomes very long, so given our time constraints we could only run for up to N = 11, but the significant CPU/GPU scaling difference is already clear.





Sanity Check — Ground State Energy Approximation

As a sanity check, both of our quantum algorithms should approximate similar ground state energies for the solution to LABS across values of n. We show this in the following plot →



In Retrospect...

Scaling Direction – Multi-GPU Cuda-Q

One additional scaling direction we explored is multi-gpu simulation on cuda-Q, with an 8xA100 node available to us. While this provides us 8x more VRAM (640 GB total), this only enables scaling to an extra 2-3 qubits of state given the exponential memory dependence.

Scaling Direction – Tensor Network MPS

Using tensor network matrix product states, we can factorize the total state vector into products of lower-rank matrices and thus reduce GPU VRAM usage at the cost of additional runtime.

In practice, we found tensornet-MPS reduces VRAM usage by about 60% for our tail end experiments (N = 30-31), but the increase in runtime prevented us from scaling up further due to time constraints.

Code Release

- All of our experiment + plotting code for QE-MTS and adapt-QAOA, as well as our PRD draft plan are all available publicly at https://github.com/Hoponga/2026-NVIDIA

Reflections

- Though GPU-accelerated quantum simulation offers significant speedups compared to CPU simulation, the classical nature of GPUs means the exponential memory requirement still becomes a bottleneck very quickly when scaling up experiments — ie. N = 30-32 qubits is fine, but immediately 35+ fails without large compute resources
    - Because of this, our initial scaling visions from our PRD proved infeasible
- (Related to above) — for scaling hybrid quantum-classical results, methods to compress/factorize the full statevector *efficiently* are especially important, which we briefly explored through MPS
- Even with GPU acceleration, it's important to similarly optimize the CPU scheduling/driving components of our program to fully capitalize on GPU parallelism (Amdahl's law) — we noticed this as a big improvement to our QAOA code
- The quantum algorithm and state space chosen has a very significant impact on runtime and implementation complexity — through exploiting the symmetry of the LABS problem and moving from Trotterized time-evolution circuits to QAOA, we were able to reduce total gates in our circuit by over 50%

Thank you to the lovely folks from NVIDIA!
We learnt a ton doing this project, and hopefully the iQuHack experience was just as rewarding for you all.

Evan Wang

Physics

Sophomore, UC Berkeley

Kailash Ranganathan

EECS

Senior, UC Berkeley

Angela Wu

Data Science

Junior, UC Berkeley

Keshav Deoskar

Engineering Physics

Senior, UC Berkeley

References

1. Zhu, L., Tang, H. L., Barron, G. S., Calderon-Vargas, F. A., Mayhall, N. J., Barnes, E., & Economou, S. E. (2022). Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer. Physical Review Research, 4(3), 033029.

2. Shaydulin, R., Li, C., Chakrabarti, S., DeCross, M., Herman, D., Kumar, N., ... & Pistoia, M. (2024). Evidence of scaling advantage for the quantum approximate optimization algorithm on a classically intractable problem. Science Advances, 10(22), eadm6761.

3. Mertens, S. (1996). Exhaustive search for low-autocorrelation binary sequences. Journal of Physics A: Mathematical and General, 29(18), L473.

4. Cadavid, Alejandro Gomez, et al. “Scaling Advantage with Quantum-Enhanced Memetic Tabu Search for LABS.” arXiv, 7 Nov. 2025, arXiv:2511.04553. DOI: 10.48550/arXiv.2511.04553.