ECE 520 Final Project

Hanlin Wang

December 11, 2025

1 Introduction

The Variational Quantum Eigensolver is a hybrid quantum-classical algorithm designed to find the eigenvalues of a Hamiltonian, typically the ground state energy of a molecular or condensed matter system [1]. VQE offloads the optimization task to a classical computer while using the quantum circuit solely for state preparation and energy estimation. However, the efficiency of VQE is often limited by the hardware constraints, specifically the qubit connectivity map [2].

In superconducting quantum processors, qubits are typically coupled in fixed topologies such as linear chains. If the ansatz requires entangling two qubits that are not physically connected, the compiler must route the quantum states using SWAP gates. Since a single SWAP gate is typically decomposed into three CNOT gates, and CNOTs are among the noisiest operations on current hardware, this routing overhead can introduce substantial errors [2]. Thus, choosing a suitable hardware geometry is critical in accurately simulating the problem hamiltonian with VQE.

Therefore, this project studies the impact of the computer hardware problem geometries on the ground state estimation of 1D and 2D Heisenberg spin models. We define four different trials: 1D and 2D problem goemetries against 1D and 2D hardware emulations. We further analyze the performance of the COBYLA and SPSA optimizers to determine which is more robust to the noise induced by topological mismatch.

2 Background

2.1 Background of VQE

VQE is founded on the Variational Principle, which states that for a given Hamiltonian H with ground state energy E_0, the expectation value of H with respect to any normalized

trial wavefunction $|\psi\rangle$ is bounded below by E_0:

$$E_0 \leq \frac{\langle\psi(\theta)|H|\psi(\theta)\rangle}{\langle\psi(\theta)\psi(\theta)\rangle} \tag{1}$$

Thus, the objective of VQE is to minimize the expectation value of the hamiltonian, with the objective function of:

$$E_{\text{VQE}} = \min_{\theta} \sum_{a=1}^{\mathcal{P}} w_a \, \langle 0|U^\dagger(\theta)\, P_a\, U(\theta)|0\rangle \tag{2}$$

The algorithm proceeds in a hybrid loop by preparing the state using a quantum circuit, the ansatz, and measuring the expectation of each Pauli string in the Hamiltonian. Next, a classical gradient-free optimizer processes the measured energy and updates the parameters to minimize this energy. Finally, by iteratively minimizing $\langle H\rangle$, the algorithm converges towards an approximation of the ground state wavefunction and its energy.

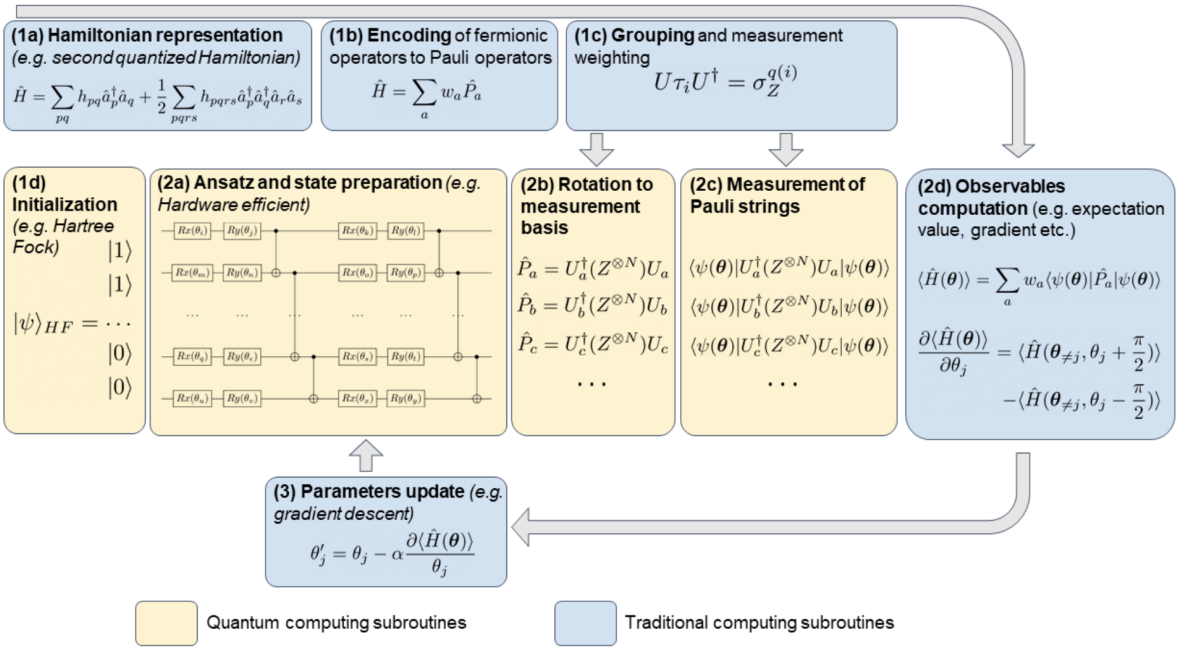

Figure 1: VQE Pipeline [1]

2.2 Background of Heisenberg Model

The quantum Heisenberg model is a statistical mechanical model used to study the phase transitions of magnetic systems. It describes the interaction between spins on a lattice

and serves as a standard benchmark for quantum simulation because it exhibits non-trivial quantum entanglement and is computationally expensive to solve classically. The general Hamiltonian is:

$$H = \sum_{\langle i,j \rangle} (J_x X_i X_j + J_y Y_i Y_j + J_z Z_i Z_j) + \sum_i h_i Z_i \tag{3}$$

In this project, we utilize the XYZ model, which is an anisotropic Heisenberg model where $J_x \neq J_y \neq J_z$, which creates a complex energy problem that serves as a robust test for our VQE.

3 Approach and Experiment

3.1 Problem and Hardware Geometries

The limited qubit connectivity of superconducting quantum computers restrict the qubits that can directly interact via two qubit gates. If the problem Hamiltonian requires entanglement between qubits that are not physically connected, the compiler must insert SWAP gates. SWAP gates are gates in quantum computers that swap the states of two qubits, and are typically made out of 3 CNOT gates, as shown below. Since CNOT gates are noisy operations in quantum computers, choosing a suitable hardware geometry is critical in accurately simulating the problem hamiltonian with VQE.

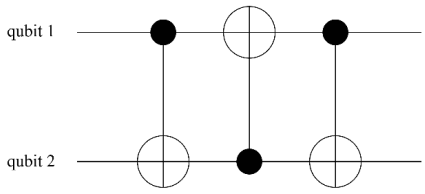

Figure 2: SWAP Gate

Furthermore, the Heisenberg model is an ideal model for quantum simulation since while solvable for small N, its dynamics become computationally complex for classical methods as the system size grows, making it an ideal problem for our experiment.

Therefore, to test this, we constructed four distinct experimental trials using a 4-qubit system:

- 1D Problem on 1D Hardware (linear chain)

- 2D Problem on 2D Hardware (square grid)
- 2D Problem on 1D Hardware (linear chain)
- 1D Problem on 2D Hardware (square grid)

3.2 Choosing The Ansatz

The choice of the ansatz is a central part of the VQE pipeline [1]. Thus, to minimize the circuit depth, I chose EfficientSU2, which is a hardware-efficient circuit structure. Our circuit consists of a layer of Ry gates followed by entangling CNOT gates repeated twice. Crucially, we adapted the entanglement structure of the ansatz to match the problem geometry rather than the hardware. For 1D problems, we utilized a linear entanglement map, while for 2D problems, we used a cyclic/square map. This choice ensures that the compiler is forced to insert SWAP gates to realize the non-local connectivity, allowing us to analyze the impact of geometry mismatches.

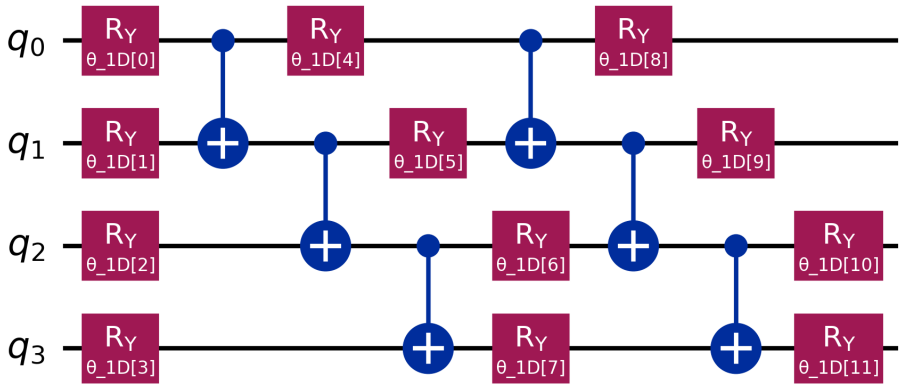

Figure 3: 1D Linear Ansantz

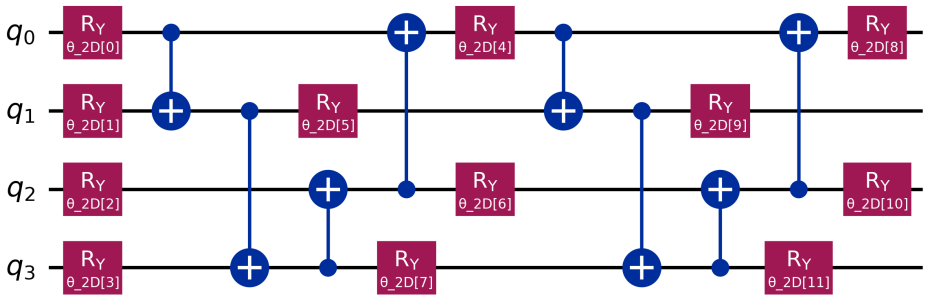

Figure 4: 2D Square Ansantz

3.3 Code and Noise Model

I conducted the experiments using the IBM Qiskit framework and the qiskit-aer simulator. To rigorously test the hypothesis. I implemented a noise model that focuses specifically on the error introduced by the SWAP gates. A standard depolarizing error channel was applied to the CNOT gates with a gate error rate of p=0.03. Since SWAP uses 3 CNOT gates, any topological mismatch will incur a 0.09 error rate per swap. Additionally, I chose to compare COBYLA and SPSA optimization schemes. The code can be found at: https://github.com/Hanlin2005/Analyzing-VQE-Geometry.

4 Results and Analysis

4.1 Impact of Geometry Mismatch

Table 1: VQE trials with various hardwares (H) and problems (P) with and without noise

Config	Opt	E_{exact}	E_{ideal}	E_{noisy}	ΔE_{ideal}	ΔE_{noisy}	depth(swaps)
P:1D, H:1D	COBYLA	-4.3723	-3.8782	-2.8215	0.4941	1.5507	8(0)
P:1D, H:1D	SPSA	-4.3723	-3.9961	-2.7329	0.3762	1.6394	8(0)
P:1D, H:2D	COBYLA	-4.3723	-4.0840	-2.8445	0.2883	1.5278	12(1)
P:1D, H:2D	SPSA	-4.3723	-3.5618	-1.9575	0.8105	2.4148	10(1)
P:2D, H:1D	COBYLA	-5.4641	-4.1099	-1.4780	1.3542	3.9861	25(6)
P:2D, H:1D	SPSA	-5.4641	-4.4653	-1.6416	0.9988	3.8225	25(6)
P:2D, H:2D	COBYLA	-5.4641	-4.2920	-3.4219	1.1721	2.0422	11(0)
P:2D, H:2D	SPSA	-5.4641	-4.5317	-3.4351	0.9324	2.0290	11(0)

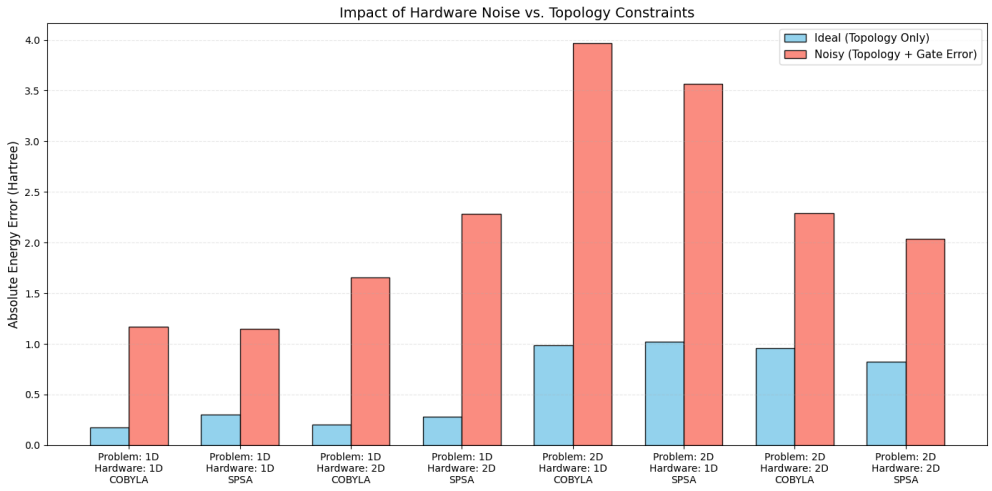

Figure 5: Results from VQE with various hardware and problem geometry with and without noise using COBYLA and SPSA ooptimizers

First, we confirm that in the ideal case, the difference hardware and problem geometries is not significant. That is, given the same optimizer, (P:2D, H:1D) vs. (P:2D, H:2D) do not have significant differences, with the only differences being due to probabilistic differences during optimization.

More importantly, when noise is added, wee see that the (P:2D, H:1D) configurations yield the most significant errors, with $\Delta E_{noisy} \approx 3.9$ across both optimizers. This is double the error in the case with (P:1D, H:1D), confirming that the SWAP gate error for hardware mismatches is dominant for the linear geometry.

Conversely, the (P:1D, H:2D) mismatch performed comparably to the (P:1D, H:2D). Since the 2D hardware geometry is more complex and includes connections to the 1D geometry, this confirms our expectation that having excess connectivity does not penalize performance, provided the problem geometry is a subgraph of the hardware graph.

4.2 Impact of Optimization Algorithm

Finally, we compare the COBYLA and SPSA optimizers with SWAP noise. As shown below, SPSA consistently outperforms COBYLA in final accuracy, while COBYLA converges rapidly but often settles into local minima. This suggests that for topologically constrained problems where SWAP noise is prevalent, stochastic optimizers like SPSA are essential.

Additionally, we do not observe significant differences between the performances of varying hardware geometry. The performance of both geometries converge at around 100 iterations for SPSA and 20 for COBYLA.

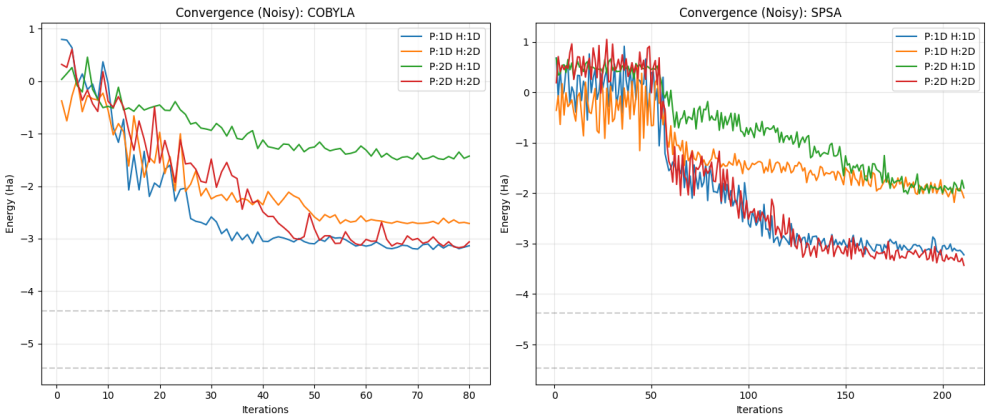

Figure 6: Comparison of COBYLA and SPSA Optimizers with Noise

5 Conclusion

In this project, we successfully implemented a VQE simulation for 1D and 2D Heisenberg models. Our data confirms that hardware topology is a critical factor in algorithm success, and that the impact is greater for 1D hardware mismatches, since sparsely connected hardware (1D) introduces significant SWAP-induced noise for complex problems. Furthermore, we verified that SPSA (stochastic) is generally a superior optimizer choice for these noisy, structural optimization landscapes compared to COBYLA (not stochastic). These findings underscore the importance of co-designing quantum algorithms and quantum hardware architectures.

References

[1] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, and J. Tennyson, "The Variational Quantum Eigensolver: a review of methods and best practices," *Physics Reports* **986**, 1–128 (2022), arXiv:2111.05176.

[2] A. Holmes, S. Johri, G. G. Guerreschi, J. S. Clarke, and A. Y. Matsuura, "Impact of qubit connectivity on quantum algorithm performance," *Quantum Science and Technology* **5**, 025009 (2020), arXiv:1811.02125.

[3] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a quantum processor," *Nature Communications* **5**, 4213 (2014), arXiv:1304.3061.

[4] "Quantum Heisenberg model," *Wikipedia, The Free Encyclopedia*, `https://en.wikipedia.org/wiki/Quantum_Heisenberg_model` (accessed 11 December 2025).

[5] PennyLane, "A brief overview of VQE," PennyLane Demos (2020), `https://pennylane.ai/qml/demos/tutorial_vqe`.

[6] PennyLane, "What is a SWAP gate?," PennyLane Glossary, `https://pennylane.ai/qml/glossary/what-is-a-swap-gate`.

[7] IBM Quantum, "TwoLocal," Qiskit Circuit Library Documentation, `https://quantum.cloud.ibm.com/docs/en/api/qiskit/qiskit.circuit.library.TwoLocal`.

[8] Qiskit Textbook, "Simulating Molecules using VQE," GitHub notebook `vqe-molecules.ipynb`, `https://github.com/Qiskit/textbook/blob/main/notebooks/ch-applications/vqe-molecules.ipynb`.