Variational Quantum Eigensolver
– SQUANDER –
- Final Report -

Balázs Menkó
Supervisor: Péter Rakyta

Eötvös Loránd University

May 6, 2025

Contents

1 Introduction

1.1 Theoretical background

In quantum computing, calculations are performed using quantum bits, or qubits for short. Qubits leverage the principles of quantum mechanics, such as superposition and entanglement. These properties enable certain algorithms to achieve quadratic or even exponential speedups compared to classical methods.

1.2 Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) is a hybrid algorithm that combine quantum and classical computation [1] to estimate the ground-state energy of quantum systems [2]. It works by employing a parameterized quantum circuit to create trial wavefunctions, while a classical optimization routine iteratively reduces the expected value of the Hamiltonian.

1.2.1 SQUANDER package

The SQUANDER package[1] is an optimization-based quantum compiler [3]. It is built in C/C++ and also provides Python interfaces. Using this tool, quantum circuits can be decomposed to simpler quantum gates.

1.3 Hamiltonian generation process

To run simulations, we need the Hamiltonian. It can be given exactly, but it is more accurate to generate the matrix explicitly. Initially, we need to create a topology, which can be represented as a random regular graph. The `networkx` package provides a suitable function suitable for this purpose[2]. This function generates a random d-regular graph on n nodes, where n corresponds to the number of qubits in our case. By providing a seed value, we can ensure the generation is reproducible. Finally, the random graph is modified to implement the XXX Heisenberg model [3].

1.4 Barren plateau problem

The barren plateau (BP) problem [4] in VQE refers to the phenomenon where the gradient of the cost function vanishes exponentially with the size of the quantum system, making optimization extremely difficult. As a result, the parameter landscape becomes flat, and classical optimizers struggle to find a direction for improvement. This issue is particularly problematic in deep or unstructured quantum circuits, as it hinders the convergence of the algorithm and limits the scalability of the algorithm. Understanding and mitigating barren

[1] See on GitHub: https://github.com/rakytap/sequential-quantum-gate-decomposer
[2] See: networkx.org/random_regular_graph

plateaus is crucial for the practical implementation of VQE on larger quantum systems.

2 Optimization methods

The optimization methods used in the VQE algorithm (see Sec. 1.2) include one the following:

- **Gradient Descent:** This method iteratively updates the model parameters by moving in the direction of the negative gradient of the loss function, gradually approaching a local minimum.
- **Parameter Shift Rule** (also called Quantum Gradient Descent) [5]: This technique estimates gradients by evaluating quantum circuits at shifted parameter values, enabling gradient-based optimization in quantum machine learning.
- **ADAM** (Adaptive Moment Estimation) [6]: ADAM combines momentum and adaptive learning rates by keeping track of both first and second moments of the gradients, leading to fast and stable convergence.
- **BFGS** (Broyden–Fletcher–Goldfarb–Shanno) [7]: BFGS is a quasi-Newton method that builds an approximation to the inverse Hessian matrix to perform more informed and efficient parameter updates.
- **Powell's method** [8]: Powell's method performs optimization by iteratively minimizing the function along a set of conjugate directions, without requiring gradient information.
- **COBYLA** (Constrained Optimization BY Linear Approximation) [9]: COBYLA solves constrained optimization problems by constructing successive linear approximations of the objective and constraint functions.
- **Nelder–Mead** [10]: The Nelder–Mead algorithm minimizes functions using a simplex of points and transformations such as reflection, expansion, and contraction, making it useful for derivative-free optimization.
- **Batched Line Search Strategy** [3] (also called Cosine Strategy): This approach avoids the need for learning rate tuning by dynamically adapting the step size using cosine similarity, helping to prevent vanishing gradients and enabling traversal through narrow valleys.

3 Results

This project is based on the Squander's *examples* folder was very helpful. My project is based on the `VQE/Heisenberg_VQE.py` file. In this source file, I made the following modifications:

- Added argument parsing to retrieve arguments from bash command lines.
- Modified the data-saving methods.
- Added three new optimizers, which are *gradient-free* methods.

Furthermore, I created Python scripts to analyze the results of the simulations. In the next subsections, the results are presented.

3.1 Zero initial parameter vector

The Python script initially used a parameter vector containing only zeros. On the Fig. 1 and Fig. 2 the minimization process over the number of cost function evaluation can be seen with different gradient-based and gradient-free methods. Due to the zero initialization, gradient-based algorithms failed to converge, they were unable to locate the minima. The situation is the same for the BFGS method. In this case, the Cosine method was able to find the minimal energy level then the ADAM method.

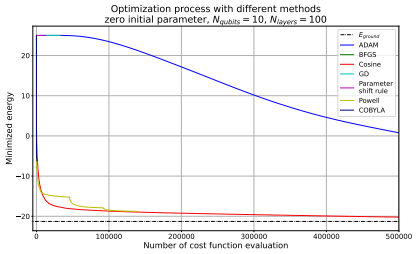

Figure 1: $N_{qb} = 10$

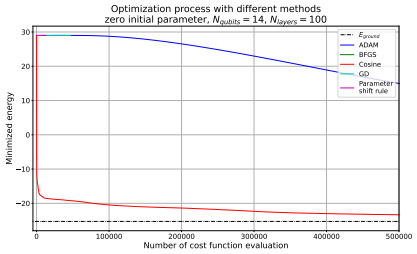

Figure 2: $N_{qb} = 14$

3.2 Random initial parameter vector

To address this issue, the parameter vector was initialized with random values. In the simulations I used a uniform distribution between $[-0.01, 0.01]$. This change resulted in the plots shown in Fig. 3 and Fig. 4. In this case the Cosine method performed as well as before.

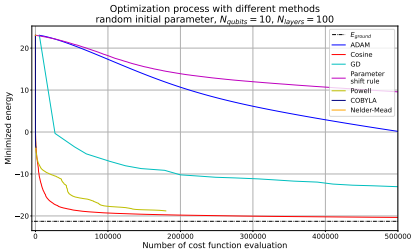

Figure 3: $N_{qb} = 10$

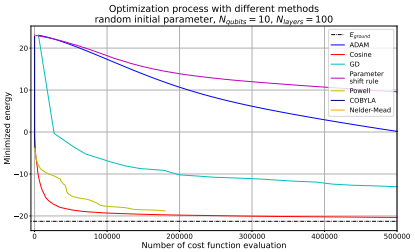

Figure 4: $N_{qb} = 14$

3.3 Hamiltonian Generation

In the second part of the project I focused on the Hamiltonian $\hat{H}$ generation process. Initially, in can be given in an exact form, but it can also be generated based on random regular graphs. First, I studied how changing the degree value affected performance (see in Sec. 3.3.1), then I investigated the performance of models generated with different random seed values (see in Sec. 3.3.2)

3.3.1 Changing the Degree Value

I generated datasets for degree values $d \in 2, 3, 4$ using the *ADAM*, *Powell*, and *Batched Line Search* optimization methods. Figures 5, 7, and 9 show the corresponding graphs, with nodes representing qubits and edges indicating interactions. Figures 6, 8, and10 present the performance of the optimization methods. Across all cases, *Batched Line Search* consistently converges faster and reaches lower energy values, followed by *Powell*, while *ADAM* converges slowest, particularly for higher d.

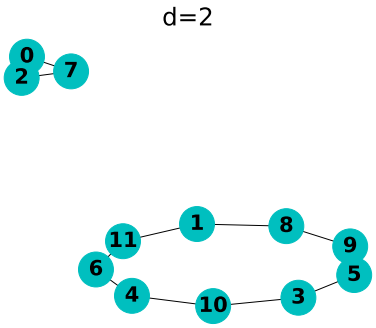

Figure 5: The random graph of the simulation.

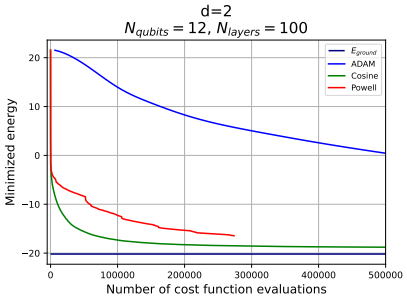

Figure 6: Simulations with $d = 2$.

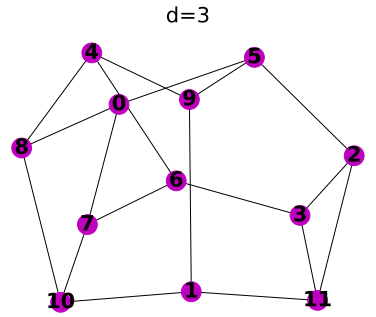

Figure 7: The random graph of the simulation.

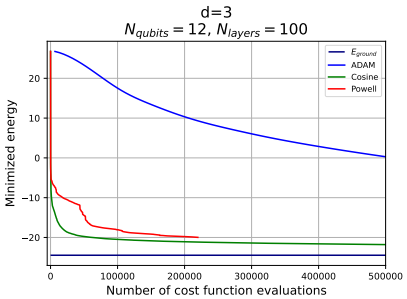

Figure 8: Simulations with $d = 3$.

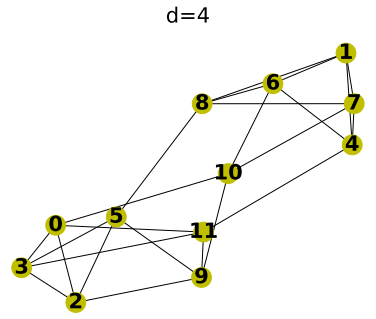

Figure 9: The random graph of the simulation.

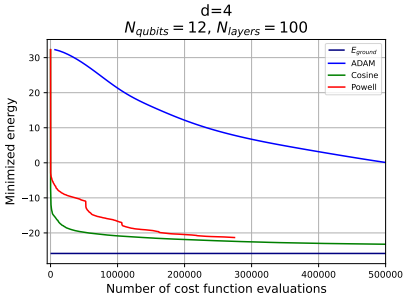

Figure 10: Simulations with $d = 4$.

3.3.2 Changing the Seed Value

Finally, I analyzed the effect of varying the random seed during Hamiltonian generation with a fixed degree $d = 3$, while varying the random seed. This resulted in different connection patterns between qubits, as shown in Figures 11, 13, and 15. However, based on the simulation results in Figures 12, 14, and 16, the performance appears to be largely independent of the specific random seed used.

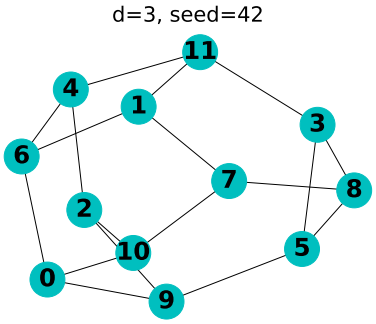

Figure 11: The random graph of the simulation.

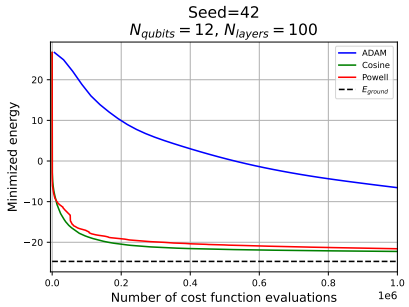

Figure 12: Simulations with $seed = 42$.

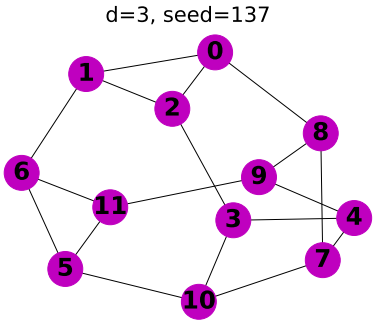

Figure 13: The random graph of the simulation.

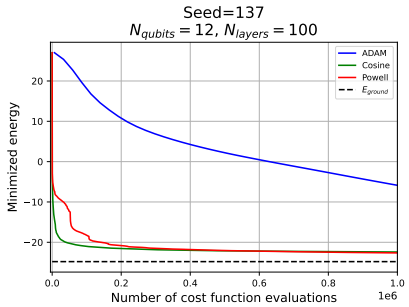

Figure 14: Simulations with $seed = 137$.

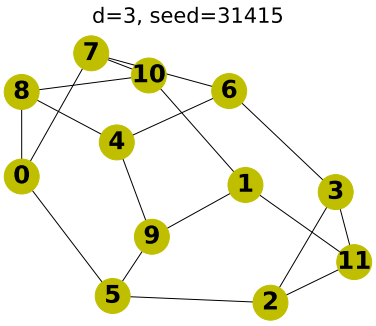

Figure 15: The random graph of the simulation.

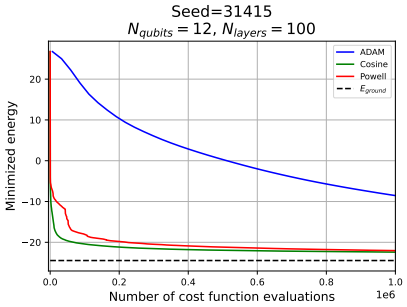

Figure 16: Simulations with $seed = 31415$.

4 Conclusion

In this project, I explored the Variational Quantum Eigensolver algorithm and its implementation using the SQUANDER package. I examined the effects of different optimization methods, initialization strategies, and Hamiltonian generation processes on the convergence behavior. My results highlight the effectiveness of gradient-free methods like Batched Line Search in overcoming issues such as barren plateaus. Furthermore, the structure of the underlying Hamiltonian, influenced by graph degree and randomness, can significantly affect optimization performance. These insights contribute to a deeper understanding of how to enhance VQE scalability and efficiency for quantum simulations.

References

[1] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. *The theory of variational hybrid quantum-classical algorithms*. New Journal of Physics, 18(2):023023, 2016

[2] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. *A variational eigenvalue solver on a photonic quantum processor*. Nature communications, 5(1):4213, 2014

[3] Jakab Nádori, Gregory Morse, Zita Majnay-Takács, Zoltán Zimborás, and Péter Rakyta. *Line search strategy for navigating through barren plateaus in quantum circuit training*, 2025. 2402.05227

[4] Jakab Nádori, Gregory Morse, Zita Majnay-Takács, Zoltán Zimborás, and Péter Rakyta. *Line Search Strategy for Navigating through Barren Plateaus in Quantum Circuit Training*. arXiv preprint arXiv:2402.05227, 2024

[5] David Wierichs, Josh Izaac, Cody Wang, and Cedric Yen-Yu Lin. *General parameter-shift rules for quantum gradients*. Quantum, 6:677, 2022

[6] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*, 2017. 1412.6980

[7] José Luis Morales. *A numerical study of limited memory BFGS methods*. Applied Mathematics Letters, 15(4):481–487, 2002

[8] Oliver Kramer. *Iterated local search with Powell's method: a memetic algorithm for continuous global optimization*. Memetic Computing, 2:69–83, 2010

[9] Tom M Ragonneau. *Model-based derivative-free optimization methods and software*. arXiv preprint arXiv:2210.12018, pages 46–47, 2022

[10] Yoshihiko Ozaki, Masaki Yano, and Masaki Onishi. *Effective hyperparameter optimization using Nelder-Mead method in deep learning*. IPSJ Transactions on Computer Vision and Applications, 9:1–12, 2017