Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk

Kwantowe wyszukiwanie architektury dla wariacyjnych algorytmów kwantowych wspomagane uczeniem ze wzmocnieniem

Rozprawa doktorska

mgr Akash Kundu
Promotor:
dr hab. Jarosław Miszczak

Gliwice, 14.02.2024

Institute of Theoretical and Applied Informatics, Polish Academy of Sciences

Reinforcement learning-assisted quantum architecture search for variational quantum algorithms

Doctoral dissertation

mgr Akash Kundu
Supervisor:
dr hab. Jarosław Miszczak

Gliwice, February 14, 2024

Contents

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my supervisor *Jarosław Adam Miszczak* (*Jarek*). During the transition from my master's to a doctoral degree, I was very unsure that I would even get a PhD position, having completed my master's thesis in cosmology. But during our first interview, his calm and friendly behaviour inspired me, and I guess later, he saw something in me and decided to give me the position of a contractor in his group. In the initial few months, I explored topics from quantum information to quantum computing to find something I could decide to work on throughout my thesis. Then, on a summer afternoon, I came across one of his papers on the certification of quantum devices, which encouraged me to implement the algorithm and delve into the world of variational quantum algorithms.

Throughout my doctoral studies, whenever I had an idea or a glimpse of an idea, I unhesitantly could discuss it with him and return with valuable insights. Besides the technical guidance, over the past four years, *Jarek* provided the necessary fuel to develop my soft skills and be a better researcher and human being. Thank you *Jarek*, I could never forget all those remarkable days when you took us on a cycling tour around the city and all those weekly lunch days to try various regional cuisines of the Silesian region in Poland.

I want to convey my gratitude to *Ludmila Botelho*, *Adam Glos*, *Özlem Salehi* and *Krzysztof Domino*. Me and *Ludmila* got the contractor position in *Jarek*'s group. When I arrived in Poland, we were more like best buddies than colleagues. We not only used to brainstorm about various cool ideas and concepts related to quantum physics, but we would take these discussions to restaurants and cafes and play many co-op video games together. During the first few months, it was *Adam*, who was then a final year PhD student in *Jarek*'s group, who came out with an exciting method of improving variational quantum algorithms based on mid-circuit measurement. Because of him, I got my very first project to work in variational quantum algorithms and learned an unprecedented amount of new stuff and programming tricks from him. Thank you *Zoltán Zimborás* for helping me clarify many concepts during the project. Soon, *Özlem* joined our group as a postdoc, whose supervision helped me to delve into the world of quantum annealing.

Along with *Adam* and *Krzysztof*, we solved many interesting real-world problems related to test vehicle production and railway scheduling problems with quantum annealing.

A very special thanks to *Mateusz Ostaszewski*, whose work on applying reinforcement learning for optimizing variational quantum circuit architectures initially inspired me to investigate quantum chemistry problems using machine learning. Afterwards, I had many technical and nontechnical discussions with him about various approaches to enhance the search for optimal quantum circuit architectures. This eventually led me to complete my reinforcement learning-assisted quantum architecture search thesis. We met only twice physically, but our several hours of long virtual meetings and writing enormous lines of code were equally entertaining. During my project with *Mateusz*, I met *Yash Patel*, *Onur Danaci*, and *Vedran Dunjko* from the University of Leiden and I learned a lot throughout our collaboration. I sincerely thank my doctoral committee for reading this thesis and giving me valuable comments to improve it.

During the past few years, I have shared memories with many offline and online *Mert Nakip, Godlove Kuaban, Aleksandra Krawiec, Ryszard Kukulski, Paulina Lewandowska, Konrad Jałowiecki, Mohammed Nasereddin, Nur Kelesoglu, Zakaria Mzaouali, Marek Gluza*. Thank you for those fantastic memories. Thanks to *Łukasz Zimny* for successfully resolving most technical issues related to my office desktop.

I am grateful to *Tamal Acharya*, who helped me meet *Aritra Sarkar*, a postdoc at TUDelft. In the past couple of years, we collaborated on various interesting projects and shared remarkable memories. It is *Aritra* who introduced me to one of the lectures by *Swami Sarvapriyananda*, which finally made me fascinated by Advaita philosophy and follow it—this period of my life enabled me to dive into philosophy and explore the spiritual aspects. Personally, it is impossible to express in words the amount of amazing time I have shared with *Aritra* online and offline, so I will pass it in silence.

One of the best decisions I made during my PhD was to join QIndia. It has emerged as a globally connected network driven by a shared passion to promote quantum research in India. Our very organic online meeting makes it hard to believe that I have never physically met many of the community members yet. Thank you *Rishi Sreedhar, Rajiv Krishnakumar, Karthigeyan Shankar, Jyoti Faujdar, Devika Sharma, Ameya Nambisan*, and all the other core team and participating members.

Thank you *Chandika* for inspiring me to pursue a master's degree in physics and for encouraging me to pursue a PhD. I express my gratitude to my friend *Kuba Górnicz* for always being there to support me whenever I needed him the most during my stay in Poland. My warmest gratitude is for one and only *Sara*, who emotionally supported me in the past few months. She definitely made me a stronger person. Her influence helped me finalise the thesis.

Last but not least, I want to thank my *Maa*, *Sujata Kundu* whose incredible hard work and dedication to raise me and my brother *Surja Kundu* up in a perfect environment even though the outside situation was far from it. You are my main inspiration to carry myself this far. Thank you for sacrificing so much and educating me. There are many instances I felt demotivated during my doctoral studies; I don't know what I would do if I did not have you and *Surja* to share my emotions and feelings and get advice from you.

List of publications

Publications relevant to this dissertation are highlighted using **bold font**.

1. **Akash Kundu, Przemysław Bedełek, Mateusz Ostaszewski, Onur Danaci, Yash J. Patel, Vedran Dunjko, Jarosław A. Miszczak; *Enhancing variational quantum state diagonalization using reinforcement learning techniques*, New Journal of Physics, Vol. 26, pp. 013034 (2024)**, arXiv:2306.11086, DOI:10.1088/1367-2630/ad1b7f
   Code: `https://github.com/iitis/RL_for_VQSD_ansatz_optimization`

2. **Yash J. Patel, Akash Kundu, Mateusz Ostaszewski, Xavier Bonet-Monroig, Vedran Dunjko, Onur Danaci; *Curriculum reinforcement learning for quantum architecture search under hardware errors*; Accepted to The Twelfth International Conference on Learning Representations, 2024,** `https://openreview.net/forum?id=rINBD8jPoP`**.** arXiv:2402.03500
   Code: `https://anonymous.4open.science/r/CRLQAS`

3. **Akash Kundu, Jarosław A. Miszczak, *Variational certification of quantum devices*; Quantum Science and Technology, Vol. 7, No. 4, pp 045017 (2022)**, DOI:10.1088/2058-9565/ac8572, arXiv:2011.01879.
   Code: `https://github.com/iitis/variational_channel_fidelity`

4. Ludmila Botelho, Adam Glos, Akash Kundu, Jarosław Adam Miszczak, Özlem Salehi, Zoltán Zimborás, *Error mitigation for variational quantum algorithms through mid-circuit measurements*; Physical Review A, Vol. 105, No. 2, pp. 022441 (2022), arXiv:2108.10927 DOI:10.1103/PhysRevA.105.022441.
   Code: `https://github.com/iitis/method-of-continuation-qaoa`

5. Krzysztof Domino, Akash Kundu, Özlem Salehi, Krzysztof Krawiec, *Quadratic and higher-order unconstrained binary optimization of railway rescheduling for quantum computing*; Quantum Information Processing, vol. 21, No. 9, pp. 337 (2022), DOI:10.1007/s11128-022-03670-y.
   Code: `https://github.com/iitis/railways_HOBO`

6. Adam Glos, Akash Kundu, Özlem Salehi, *Optimizing the Production of Test Vehicles Using Hybrid Constrained Quantum Annealing*, SN Computer Science,

Vol. 4, 609 (2023), DOI:10.1007/s42979-023-02071-x.
Code: `https://github.com/iitis/bmw-qchallenge`

7. Akash Kundu, Jarosław A. Miszczak, *Transparency and Enhancement in Fast and Slow Light in q-Deformed Optomechanical System*, Annalen der Physik, Vol. 534, No. 8, pp. 2200026 (2022), DOI:10.1002/andp.202200026, arXiv:2205.15800.

8. Hao-Jie Cheng, Shu-Jie Zhou, Jia-Xin Peng, Akash Kundu, Hong-Xue Li, Lei Jin, Xun-Li Feng, *Tripartite entanglement in a Laguerre–Gaussian rotational-cavity system with an yttrium iron garnet sphere*, Journal of the Optical Society of America B, Vol. 38, No. 2, pp. 285–293 (2021), DOI:10.1364/JOSAB.405097.

9. Akash Kundu, Chao Jin, Jia-Xin Peng, *Study of the optical response and coherence of a quadratically coupled optomechanical system*, Physica Scripta, Vol. 96, No. 6, pp 065102 (2021), DOI:10.1088/1402-4896/abee4f.

10. Akash Kundu, Chao Jin, Jia-Xin Peng, *Optical response of a dual membrane active–passive optomechanical cavity*; Annals of Physics, vol. 249, pp. 168465 (2021),
DOI:10.1016/j.aop.2021.168465.

11. Akash Kundu, SD Pathak, VK Ojha; *Interacting tachyonic scalar field*, Communications in Theoretical Physics, Vol. 38, No. 2, pp 285–293 (2021), DOI:10.1088/1572-9494/abcfb1.

12. Turbasu Chatterjee, Shah Ishmam Mohtashim, Akash Kundu; *On the variational perspectives to the graph isomorphism problem*, arXiv:2111.09821 2021.

13. Akash Kundu, Tamal Acharya, Aritra Sarkar; *A scalable quantum gate-based implementation for causal hypothesis testing*, 2023, DOI:10.48550/arXiv.2209.02016.
Code: `https://github.com/Advanced-Research-Centre/QaCHT`

Abstract in Polish

Istotną przeszkodą w erze zaszumionych komputerów kwantowych średniej skali (ang. NISQ – Noisy Intermediate-Scale Quantum) jest konstrukcja obwodów kwantowych, które pozwolą na wykonanie użytecznych algorytmów kwantowych i są zgodne z ograniczeniami narzuconymi przez obecne ograniczenia sprzętu kwantowego. Aby sprostać tym wyzwaniom w obecnie dostępnych urządzeniach kwantowych, opracowano wariacyjne algorytmy kwantowe (ang. VQA – Variational Quantum Algorithms), które stanowią klasę hybrydowych algorytmów kwantowo-klasycznej dla problemów optymalizacji. Jednakże ogólna wydajność wariacyjnych algorytmów kwantowe zależy od (1) strategii inicjalizacji obwodu wariacyjnego, (2) struktury obwodu (znanej również jako ansatz) oraz (3) konfiguracji funkcji kosztu. Koncentrując się na (2), w tej pracy zaproponowane są metody poprawy wydajność wariacyjnych algorytmów kwantowych poprzez automatyzację wyszukiwania optymalnej struktury obwodów wariacyjnych za pomocą uczenia się ze wzmocnieniem (ang. RL – Reinforcement Learning). W ramach pracy skupiamy się na określeniu optymalności obwodu poprzez ocenę jego głębokości, całkowitą liczbę bramek i parametrów oraz dokładności w rozwiązaniu zadanego problemu. Zadanie automatyzacji wyszukiwania optymalnych obwodów kwantowych znane jest jako wyszukiwanie architektury kwantowej (ang. QAS – Quantum Architecture Search). Większość badań w zakresie wyszukiwania architektury kwantowej koncentruje się na scenariuszu bezszumowym. W związku z tym wpływ szumu na proces wyszukiwania architektury pozostaje niewystarczająco zbadany. W tej pracy zajmujemy się tym problemem poprzez wprowadzenie techniki łączącej kodowanie obwodów kwantowych opartego na tensorach, ograniczenie dynamiki środowiska w celu efektywnego badania przestrzeni poszukiwań możliwych obwodów, schemat zatrzymywania epizodów w celu nakierowania agenta na znajdź krótsze obwody, oraz poprzez wykorzystanie podwójnie głęboką sieć Q (DDQN) z polityką ϵ dla lepszej stabilności. Eksperymenty numeryczne na bezszumowym i zaszumionym sprzęcie kwantowym pokazują, że w radzeniu sobie z wybranymi algorytmami wariacyjnymi, zaproponowana metoda wyszukiwania architektury przewyższa istniejący metody. Dodatkowo metody, które proponujemy w pracy, można zostać łatwo zaadoptowane do szerokiego zakresu innych VQA.

Abstract in English

A significant hurdle in the noisy intermediate-scale quantum (NISQ) era is identifying functional quantum circuits. These circuits must also adhere to the constraints imposed by current quantum hardware limitations. Variational quantum algorithms (VQAs), a class of quantum-classical optimization algorithms, were developed to address these challenges in the currently available quantum devices. However, the overall performance of VQAs depends on the initialization strategy of the variational circuit, the structure of the circuit (also known as ansatz) and the configuration of the cost function. Focusing on the structure of the circuit, in this thesis, we improve the performance of VQAs by automating the search for an optimal structure for the variational circuits using reinforcement learning (RL). Within the thesis, the optimality of a circuit is determined by evaluating its depth, the overall count of gates and parameters, and its accuracy in solving the given problem. The task of automating the search for optimal quantum circuits is known as quantum architecture search (QAS). The majority of research in QAS is primarily focused on a noiseless scenario. Yet, the impact of noise on the QAS remains inadequately explored. In this thesis, we tackle the issue by introducing a tensor-based quantum circuit encoding, restrictions on environment dynamics to explore the search space of possible circuits efficiently, an episode halting scheme to steer the agent to find shorter circuits, a double deep Q-network (DDQN) with an ϵ-greedy policy for better stability. The numerical experiments on noiseless and noisy quantum hardware show that in dealing with various VQAs, our RL-based QAS outperforms existing QAS. Meanwhile, the methods we propose in the thesis can be readily adapted to address a wide range of other VQAs.

Chapter 1

Introduction

Quantum computing leverages the principles of quantum mechanics to gain a distinct advantage in information processing. Ongoing worldwide endeavours are actively striving to materialize a sufficiently large, controllable, and programmable quantum computer. Corporations such as Google [28], IBM [67], Rigetti [127], and Intel [69] are utilizing superconducting qubits where the quantum processing unit utilizes a superconducting architecture. Whereas, Honeywell [68] and IonQ [70] utilize ion traps as quantum processors where charged atoms, i.e. ions, are used as qubits due to the fact that ions can be trapped in one precise location with the help of electric fields. Meanwhile, D-Wave [46] quantum computers are based on quantum annealing [41]. The qubits are made from tiny superconducting loops.

In the early and late 90s, pure quantum algorithms were introduced, such as Shor's [138], which is used to find the prime factor of integers, and Grover's [57] algorithm, which is used to search a unique input from an unstructured dataset. To reveal the true potential of these algorithms and achieve quantum advantage, we require quantum hardware with thousands to millions of qubits. Unfortunately, the current quantum devices are of small scale with 5-200 qubits, noise prone, and have constrained connectivity between qubits. These devices are called Noisy Intermediate Scale Quantum (NISQ) computers [125]. At the time of this thesis (2021-2023), no quantum devices exist that can execute quantum algorithms demonstrating a provable quantum advantage for real-world use cases.

To deal with these limitations and exploit the NISQ devices, Variational Quantum Algorithms (VQAs) [101] was introduced, where the task of solving a quantum problem is distributed into quantum and classical computers. The VQAs are fundamentally comprised of three essential components: a Parametrized Quantum Circuit (PQC) or ansatz, a cost function that encodes the problem, and a classical optimization procedure responsible for adjusting the PQC's parameters in order to minimize the cost function. Ongoing research efforts are dedicated to exploring and comprehending the potential of each of these building blocks within the realm

of VQAs [31].

The conventional approach for creating an ansatz involves predefining its structure before getting underway with the algorithm. Based on the user's ambition, the structure of the ansatz can be driven by physical considerations [120] or hardware constraints [76].

Nonetheless, fixing the ansatz's structure imposes a significant restriction on exploring the cost function landscape and can prevent us from reaching the true solution. Sophisticated methods [10, 21, 53, 56, 83] have been introduced to enhance the performance of VQAs. Meanwhile, to avoid these limitations, recent attention has been directed towards automating the construction of ansatz [6,56,148,148,169], known as Quantum Architecture Search (QAS). The QAS eliminates the need for domain-specific expertise, and it has the ability to produce an ansatz tailored for specific VQAs. Given a finite set of quantum gates, the objective of QAS is to discover the optimal arrangement of quantum operators in the form of an ansatz that minimizes the cost function.

A solution to address the challenges in QAS involves the application of Reinforcement Learning (RL) techniques, as proposed in [50, 86, 116, 166]. In the RL-based QAS methods, the cost function is optimized independently using a classical optimizer, providing an intermittent signal contributing to the cumulative reward function. This reward function, in turn, updates a policy that aims at maximizing expected returns. Based on the return, the RL-agent selects an optimal action for subsequent steps.

Meanwhile, to minimize the adverse effects of gate errors, constrained connectivity, and decoherence, it is crucial to design an ansatz that uses as few quantum gates and computational steps as possible and is as shallow as possible in terms of their depth. Gate errors, restricted connectivity, and decoherence are common challenges in NISQ devices that cause inaccuracies and lead to the loss of quantum information. By keeping the circuits gate-frugal (using fewer gates) and shallow (reducing the depth), quantum computations can be more resilient and less susceptible to the negative impacts of these quantum computing issues. This approach enhances the stability and reliability of quantum algorithms and makes them more suitable for practical applications.

In this thesis, we propose and analyze an RL-based QAS method whose agent operates on a double deep-Q network (DDQN) and an ϵ-greedy policy. The proposed QAS method improves the performance of the RL-agent to not only propose a very compact ansatz with fewer gates, depth, and number of parameters but also return a very low error while solving the problem. To achieve these results, we utilize:

1. **A tensor-based encoding scheme** to encode the ansatz as an observable for the RL-agent.

2. **A one-hot encoding scheme** to construct the action space. Each action is

represented by a parameterized rotation in either X, Y, or Z direction or a controlled-NOT (`CX`) gate.

3. **A dense and a sparse reward function** to encode the cost function.

4. **An illegal action scheme** to impose restrictions on the environment dynamics to explore the search space of possible ansatz efficiently.

5. **A random halting scheme** is an episode halting scheme to encourage the agent to find a shorter gate and depth ansatz.

These components are briefly discussed in upcoming sections. Using these, we automate the search for an optimal ansatz that finds the ground state of molecules using a curriculum reinforcement learning-based Variational Quantum Eigensolver (VQE). We also utilize a reinforcement learning enhanced Variational Quantum State Diagonalization (VQSD) algorithm that finds the optimal structure of an ansatz that diagonalizes an arbitrary quantum state.

Our results correspond to achieving an error 10^3 times lower than that of the previously proposed ansatz using less than half of the gates, indicating that the RL-agent enhances the exploration of the optimization of the cost function landscape leading to the design of smaller ansatz. These developments are summarized in the following hypothesis:

Hypothesis 1: *Utilizing reinforcement learning techniques enhances the exploration of the optimization landscape and leads to ansatz designs for Variational Quantum Algorithms (VQAs) with minimal gate count, depth, and parameters while consistently achieving a low error level in cost function evaluation.*

The **Hypothesis 1** deals with the performance of our RL-based algorithm under an idealistic scenario in the absence of any kind of device noise. Meanwhile, most algorithms for QAS have been formulated under the assumption of a noiseless scenario, free from physical noise and under consideration of all-to-all qubit connectivity, and there has been very little progress in automating the QAS problem [44]. However, in order to make the algorithm physically realizable, it is important to show that the QAS algorithm can solve the problem and provide us with a compact ansatz in the presence of constraints imposed by current NISQ devices, characterized by limited qubit connectivity and susceptibility to noise.

Noise imposes severe effects on the cost function landscape, and noise can cause the optimization process to get stuck in regions of the landscape where the gradients are vanishing to guide further progress, thereby hindering the overall optimization effort [161]. The phenomena of having a flat or extremely shallow cost function landscape are known as *barren plateau* [100]. Meanwhile, non-unitary noisy channels

such as amplitude damping, decoherence[1], and thermal relaxation transform the cost function landscape by exponentially converting the global minima into local optima. This increases the complexity of the optimization process [49]. Hence, performing QAS in the presence of noise is a critical step toward understanding and ultimately overcoming the challenges posed by real-world quantum hardware and advancing the field of quantum architecture search on NISQ devices.

In this thesis, we utilize a curriculum-based RL method for QAS and show that our algorithm can efficiently solve quantum chemistry problems by finding the ground state of molecules in a realistic noisy scenario with very few gates and lower energy error in estimating the ground state.

These results can be summarized through the following hypothesis:

Hypothesis 2: *Combining binary encoding for a quantum circuit, action space pruning, a variable episode length, and curriculum-based reinforcement learning methods can enhance the agent's learning, facilitating the quantum chemistry problem solution irrespective of quantum hardware noise and connectivity constraints.*

The majority of research in QAS focuses on a noiseless scenario. Yet, the impact of noise on the QAS remains inadequately explored. Through **Hypothesis 2**, we stress filling this gap by solving the ground state finding problem of difference molecules in realistic noisy noise scenarios. These realistic scenarios are imported from IBM quantum devices such as `ibmq_mumbai` and `ibmq_ourense`.

The rest of the thesis is organized as follows. Chapter 2 initiates by giving a brief introduction to variational quantum algorithms, including their components (such as ansatz and cost function) and the basics of reinforcement learning with suitable examples. While introducing RL, we primarily focus on model-free learning and algorithms. At the end of this Chapter, we discuss various approaches to QAS and provide an overview of our RL-based QAS approach. In Chapter 3, we discuss an RL-assisted variational quantum state diagonalization, namely the RL-VQSD algorithm. The main task of RL-VQSD is to find a compact structure of an ansatz that diagonalizes an arbitrary quantum state with very low error in eigenvalue and eigenvector estimation. The Chapter starts by benchmarking the existing quantum state diagonalization algorithm. Afterwards, we show that the RL-VQSD proposed a more compact ansatz containing a smaller gate and lower depth compared to state-of-the-art diagonalization algorithms in a noiseless scenario. The Chapter ends by providing a pointwise summary of the algorithm and results. In Chapter 4, we utilize a Curriculum-based Reinforcement Learning (CRL) approach to find the ground state of molecules under realistic noisy scenarios. We show that the ansatz proposed by the CRL-agent is more compact in terms of the number of gates and

[1]A coherence is a phase damping channel that belongs to a class of doubly stochastic channels [64].

depth as well as provides better accuracy compared to existing variational quantum eigensolver. This is followed by a thorough discussion of how different novel schemes, RL-state encoding, and the choice of reward function impact the performance of the CRL-based VQE algorithm in various realistic noisy scenarios. The Chapter ends by giving a pointwise summary of the algorithm and the results. In the following Chapter 5, we discuss a quantum state diagonalization-based quantum channel certification method. The method primarily depends on the Choi-Jamiołkowski isomorphism and variational quantum fidelity estimation [31] algorithm. As the variational quantum state diagonalization is an indispensable part of our quantum channel certification process, we discuss the possible application of RL-VQSD. This includes how it can improve the channel certification method and make it currently available quantum device friendly by deciding the number of gates and depth of diagonalizing quantum circuits. In Chapter 6, we summarise the thesis. Finally, in Chapter 7, we evaluate the merits and drawbacks of our methodology and conclude the results in the thesis. Furthermore, we delve into potential avenues for future exploration and development.

Chapter 2

Preliminaries

This chapter introduces the basic concepts and notation used in the thesis. We start the chapter by introducing variational quantum algorithms (VQAs). This introduction includes a detailed discussion of various components of VQAs relevant to understanding the later concepts of the thesis. For example, we briefly define the construction of the *cost function* and its importance in finding the optimal solution for a problem. This discussion is followed by elaborating on various *ansatz* (also known as a variational quantum circuit) constructions. Through this discussion, we will see that the number of two-qubit gates and depth for the problem-inspired ansatz increases exponentially with the number of qubits. Meanwhile, for the problem agnostic ansatz, the constrained connectivity and the elevation in the number of parameters with increasing problem size cause trainability issues.

In the following section, we provide a detailed discussion of the basics of reinforcement learning with proper examples. For the sake of the thesis, we primarily focus on model-free learning methods. The model-free learning discussion includes Q-learning, the deep and double-deep Q-network with proper code blocks for implementation purposes. In the same section, we stress the discussion of the exploration and exploitation scenario.

We conclude the chapter by introducing the basic concepts and infrastructure of our RL-based quantum architecture search algorithm.

We encourage the reader to consult Appendix A for more details on quantum states, gates, and channels.

2.1 Variational quantum algorithms

Over the past few decades, multiple scientific fields have collaborated in the exploration and advancement of quantum algorithms and their experimental implementation. However, while numerous quantum algorithms were initially proposed,

the majority require millions of physical qubits to operate on quantum hardware. Unfortunately, contemporary quantum hardware is only capable of accommodating a few hundred physical qubits, which are classified as Noisy Intermediate-Scale Quantum (NISQ) [16,125] devices. In light of this, Variational Quantum Algorithms (VQAs) [29,101] represent a specific class of NISQ algorithms that can run on these devices, as they were specifically designed with such limitations in mind. VQAs utilize quantum hardware to run a parametric quantum circuit (PQC), and then the parameters of the PQC are optimized using a classical optimization method. If not optimized up to an expected threshold, the parameters are fed back to the PQC. This quantum-classical hybrid process helps us keep the depth of the quantum circuit low and mitigate the noise. The two basic components of a VQA are (1) PQC, which is famously known as an ansatz, and (2) the cost or loss or objective function that encodes the problem.

In the remaining subsection, we briefly describe the two basic aspects of VQA.

In the subsection 2.1.1, we briefly introduce the cost function by mentioning the main criterion one should consider while choosing a cost function. Then, in section 2.1.2, we define the problem-inspired and agnostic structure of ansatzes and their advantages and disadvantages.

2.1.1 Cost function

In classical machine learning (ML) methods, a cost function is introduced to evaluate the model's performance. The main objective of a model, then, is to determine the optimal set of model parameters, which indicates a global minimum of the cost function. Hence, the cost function is sometimes called an objective function. For the optimization procedure, either gradient-based or gradient-free optimization algorithms are frequently used.

VQAs serve as quantum analogues of machine learning techniques. A crucial operational aspect of these algorithms depends on the ability to encode problems effectively into cost functions. From a geometric perspective, a cost function is defined by a hyper-surface over the parameters—a cost landscape—whereas the optimizer's role involves traversing this landscape – as depicted in Fig. (2.1) – to locate the global minima. The Fig. (2.1) illustrates the optimization process within VQAs, highlighting the quest to navigate this parameterized space to achieve optimal solutions efficiently.

One can express the cost function in the form

$$C(\vec{\theta}) = \sum_j c_j \mathrm{Tr}\left(O_j \rho'_j(\vec{\theta})\right), \tag{2.1}$$

where f_j is a set of functions. $\rho'_j(\vec{\theta}) = U(\vec{\theta})\rho_j U^\dagger(\vec{\theta})$, $U(\vec{\theta})$ is the PQC with discrete

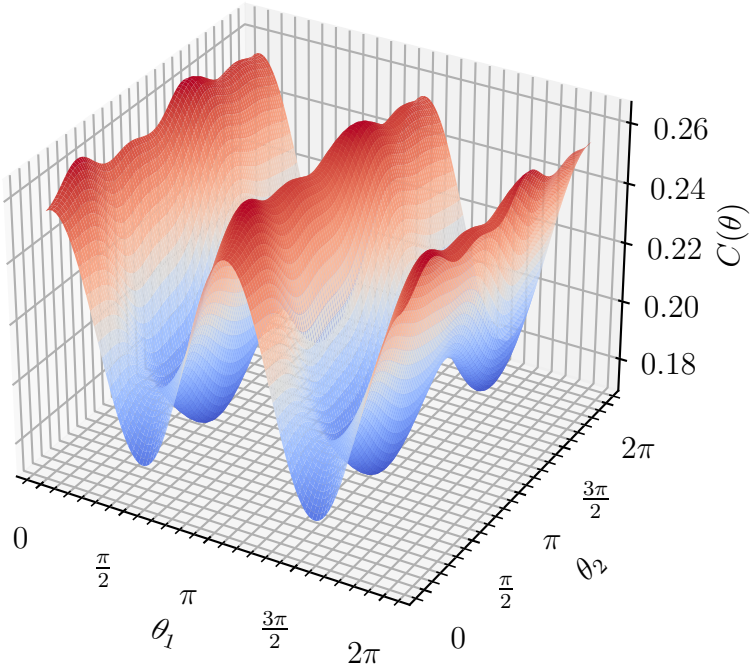


Figure 2.1: An illustration of the cost function landscape for the variational quantum state diagonalization algorithm [87] with a two-qubit mixed quantum state. As an ansatz, we choose a layer of rotation `RY` and `RZ` on both the qubits, followed by a `CX` with control on the first qubit.

or continuous vector of parameters $\vec{\theta}$. $\{\rho_j\}$ are the input state from a training set. The set $\{O_j\}$ is observable, which can be defined by a Hamiltonian in the case of the VQE problem. The problem under consideration completely determines the choice of f_k. An ideal cost function should follow a list of criteria discussed below.

Faithfulness If we have a problem under consideration, we need to formulate a cost function to solve it using a variational method. The *faithfulness* of the cost function implies that its minimum must correspond to the solution of the problem. If we consider a problem p and we define the cost function corresponds to the problem $C(\vec{\theta})$ then the solution to the problem (p^*) is given by

$$p^* = \min_{\theta_j} C(\theta_j), \tag{2.2}$$

if $C(\vec{\theta})$ is faithful cost function.

Efficiency From the term efficiency of a cost function, we indicate that one must be able to estimate it by performing measurements on a quantum device.

To maintain the validity of the quantum advantage, it is important to devise a cost function that proves computationally challenging for classical computers to compute.

For an example, for a quantum state, ρ, $\mathrm{Tr}(\rho^2)$ defines the *purity* of the state. Minimization of purity is a very useful primitive method to solve a range of problems that are relevant to quantum physics problems such as quantum state rank estimation [115], quantum state learning [34, 89], quantum device certification [85] and many more. It is also well-known that a quantum computer can find the purity of an n-qubit state with complexity scaling linearly in n, which gives exponential speed-up over classical computers [25, 54]. So, a cost function of the form

$$C = \sum_j f_j(\mathrm{Tr}(\rho_j^2)), \tag{2.3}$$

is efficiently computable and provides a quantum advantage.

Trainability Through trainability, we emphasize the fact that the cost function must be trainable, i.e. it should be possible to efficiently optimize the parameters of the cost function $\vec{\theta}$. One of the main hindrances of trainability is the occurrence of barren plateaus with increasing depth and number of qubits. Technically, a barren plateau is defined by the exponentially vanishing average partial derivatives of the cost function with the size of a quantum system [100]. This makes the landscape of

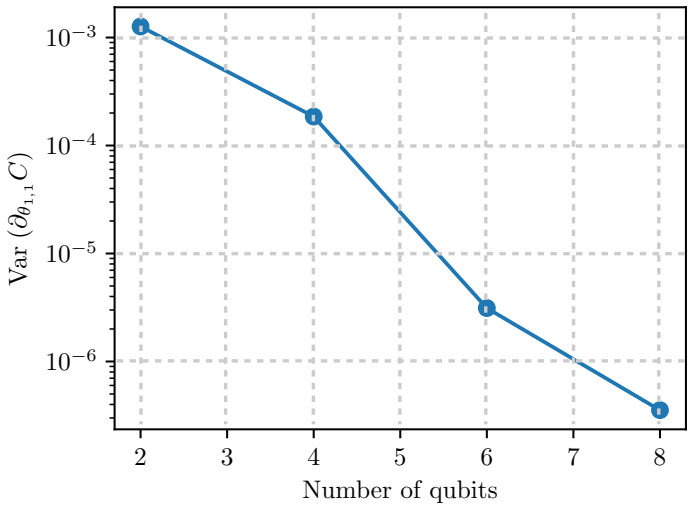


Figure 2.2: Exponential decay of the variance of the cost function gradient for quantum state diagonalization problem for three-qubit of depth 2 ansatz.

the cost function essentially flat, so it requires exponentially enhanced precision to tackle the finite sampling noise and to determine the direction of the global minima. This is irrespective of the fact that one uses a gradient-based [30] or gradient-free optimization method [8]. In a recent work [33], the authors claim that the issue of barren plateau could be tackled by carefully reconstructing the cost function and by making it local. They propose two variants of cost functions in Eq. (2.1) as

$$C^{\text{global}}(\vec{\theta}) = \sum_j c_j \text{Tr}\left(O_j^{\text{global}} \rho_j'(\vec{\theta})\right), \text{ where } O_j^{\text{global}} = \mathbb{1}_j - |0\rangle\langle 0|_j, \tag{2.4}$$

or as

$$C^{\text{local}}(\vec{\theta}) = \sum_j c_j \text{Tr}\left(O_j^{\text{local}} \rho_j'(\vec{\theta})\right), \text{ where } O_j^{\text{local}} = \mathbb{1} - \frac{1}{n}\sum_{j'=1}^{n}\left(|0\rangle\langle 0|_j \otimes \mathbb{1}_{\bar{j'}}\right). \tag{2.5}$$

Here, $\mathbb{1}$ is the identity operation and $\mathbb{1}_{\bar{j'}}$ defines $\mathbb{1}$ over all qubits except j'. Irrespective of the PQC, the variance of the gradient for the local cost function in Eq. (2.4) at worst vanishes polynomially, which makes it trainable up to a depth of order $\mathcal{O}(\log n)$, which is not the case for the global cost function Eq. (2.5), whose variance of gradient decays exponentially.

2.1.2 Ansätze

In the previous section, we stress the point that VQAs are analogous to classical ML methods. One of the fundamental aspects of the ML model is the Neural Network (NN). The quantum version of NN is a PQC that is famously known as an ansatz. Inspired by the success of classical NNs, some ansatz architectures such as quantum convolutional NN [39], recurrent quantum NN [12], quantum long short-term memory [35], and quantum graph NN [158] has been introduced.

As shown in Eq. (2.1), the ansatz $U(\vec{\theta})$ contains trainable parameters $\vec{\theta}$, these can be trained to minimize the cost function. Now, the $U(\vec{\theta})$ does not have a specific structure, but it depends on the problem under consideration, for example, UCC, quantum alternating operator, variational Hamiltonian ansatz. These types of ansatz fall under the group *problem inspired ansatz*. One can generically express $U(\vec{\theta})$ as a product of L succeeding unitaries where one part of the unitary is parametrized by parameters $\vec{\theta}_j$ and another part is non-parametrized i.e.

$$U(\vec{\theta}) = \prod_{l=1}^{L} V(\vec{\theta}_l) W_l, \tag{2.6}$$

where $V(\vec{\theta}_l)$ is the parametrized part of the ansatz and is of the form $\exp(-i\vec{\theta}_l H_l)$

and H_l is a Hermitian operator and W_l is the non-parametrized unitary. Each layer l contains a vector of parameters $\vec{\theta}$. Depending on the problem under consideration, the parametrized and the non-parametrized part takes different forms. This leads to a class of sophisticated structures of ansatz that we briefly discuss in the following.

UCC The Unitary coupled cluster is a *problem-inspired ansatz* that is widely utilized in quantum chemistry problems. In this case, the problem statement is to find the ground state energy of a molecule represented through a fermionic Hamiltonian H.

According to the Born-Oppenheimer approximation [20], one can describe the interaction of a system of electrons with its nucleus in a second quantized form where single-particle orbitals can either be filled or empty. And any interaction between electrons can be represented using annihilation ($\hat{a}$) and creation ($\hat{a}^\dagger$) following an anti-commutation relationship. Hence, a non-relativistic molecule Hamiltonian can be written in the form

$$H_{\text{mol}} = H_{\text{nuc}} + \underbrace{\sum_{pq} h_{pq}\hat{a}_p^\dagger \hat{a}_q}_{\text{single excitations}} + \underbrace{\frac{1}{2}\sum_{pqrs} h_{pqrs}\hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_r \hat{a}_s}_{\text{double excitations}} + \dots \tag{2.7}$$

To obtain UCC ansatz, we replace the traditional Hamiltonian cluster operator in Eq. (2.7) in terms of coupled cluster theory with its anti-Hermitian as follows [120, 149]

$$\begin{aligned} |\psi\rangle_{\text{UCC}} &= U_{\text{cc}}|\psi\rangle = e^{T_1+T_2+\cdots}|\psi\rangle, \\ T_1 &= \sum_{\substack{v\in\text{vacant},\\ o\in\text{occupied}}} \theta_{vo}\left(\hat{a}_v^\dagger \hat{a}_o - \hat{a}_o^\dagger \hat{a}_v\right) \\ T_2 &= \sum_{\substack{v,v'\in\text{vacant},\\ o,o'\in\text{occupied}}} \theta_{vv'oo'}\left(\hat{a}_v^\dagger \hat{a}_{v'}^\dagger \hat{a}_o \hat{a}_{o'} - \hat{a}_o^\dagger \hat{a}_{o'}^\dagger \hat{a}_v \hat{a}_{v'}\right), \\ &\vdots \end{aligned} \tag{2.8}$$

where $|\psi\rangle$ is an uncorrelated reference state, usually a Hartree-Fock state. The ansatz shown in Eq. (2.8) is the UCC ansatz. One can obtain a popular variant of UCC – which is UCCSD, where SD stands for single and double – just by considering T_1 and T_2 in the coupled cluster representation. To implement the ansatz in a quantum computer, one can use the fermion to spin mappings such as Jordan-Wigner, Parity, and Bravyi-Kiteav transformations [143, 155].

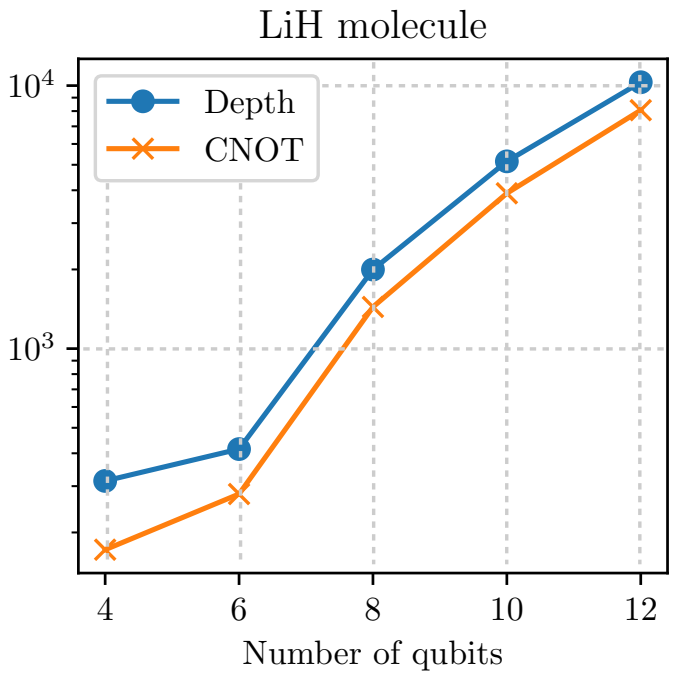


Figure 2.3: Illustration of the variation of the number of `CX` gates and *depth of the circuit* with increasing spin orbitals in LiH molecule with UCCSD ansatz.

Even though the unitarity of UCCSD suggests ease of implementation on quantum hardware, the current gate-based quantum computing asks for a decomposition in terms of one and two-qubit gates. However, the number of one and two-qubit gates and the depth of the circuit proliferates with the number of qubits as shown in Fig. (2.3).

One can utilize the Suzuki-Trotter approximation of the T_i operators to deal with the issue, keeping in mind the correct operator ordering of trotterized UCC ansatz [55].

Hardware efficient As the name suggests, the Hardware Efficient Ansatz (HEA) aims to reduce the circuit depth and gate count in $U(\vec{\theta})$ and is efficiently implementable in currently available quantum hardware. The generic form of HEA follows the form

$$U_{\text{HEA}} = \prod_{q=1}^{N} \prod_{d=D}^{1} \left(G^{q,d}(\vec{\theta}) \times U_{\text{Ent}} \right). \tag{2.9}$$

Here q is the number of qubits up to N and d is the depth of the ansatz up to D. The product on the number of qubits is over parameterized gates $G(\vec{\theta})$. U_{Ent}

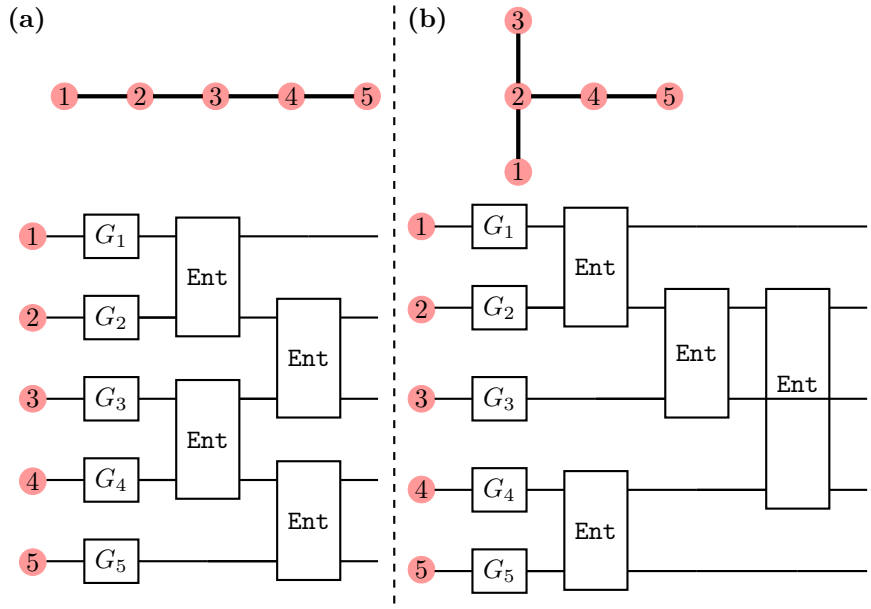


Figure 2.4: An example of HEA depending on the topology of `IBM` quantum devices. In (a), we present HEA that follows the topology of `ibmq_manila` whereas in (b) the HEA follows the topology of `ibm_quito`, `ibm_belem` and `ibm_lima`. It should be noted that the G_i are parametrized, and `Ent` is the entangling unitary. The quantum wire through the last `Ent` gate means the gate does not apply on that qubit.

defines entangled gates.

One determines the U_{Ent} and $G(\vec{\theta})$ from a predefined quantum gate set, and the placement of the gates is determined by the topology of real quantum hardware. This, in turn, helps avoid circuit depth overhead while translating an arbitrary ansatz into a sequence of gates, as shown in Fig. (2.4). This makes HEA very applicable to Hamiltonian, which has interactions similar to the quantum hardware [80]. One of the primary advantages of HEA is that while implemented, it can incorporate encoding symmetries [51, 117] and depth reduction [152]. A well-known variant of HEA is layered HEA (LHEA), where gates act on alternating pairs of qubits in a brick-like structure as shown in Fig. (2.5). But this has trainability issues. In [90], it has been shown that the trainability of HEA depends on the entanglement in input states, i.e. HEAs are trainable if input states follow the area law of entanglement [47] whereas it becomes untrainable if the input follows volume law of entanglement [18].

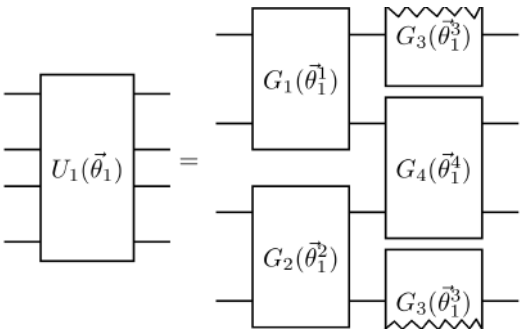


Figure 2.5: Structure of a layered ansatz, where the ansatz $U_l(\vec{\theta})$ is decomposed into layer-wise unitaries $U_l(\vec{\theta_l})$ for $l = 1, 2, \ldots, l$. Each $U_l(\vec{\theta_l})$ is further decomposed into two-qubit rotations for $\vec{\theta_i^j}$ the i denotes the layer and j is the parameter count.

Quantum approximate operator The Quantum Approximate Optimization Ansatz (QAOA) was first introduced in [48] to obtain solutions for combinatorial optimization problems. The ansatz provides an approximation to a Hamiltonian H by constructing a specific variational ansatz through first-order Suzuki-Trotter decomposition approximating the adiabatic evolution. The operators $\exp(\mathrm{i}\beta_j H_{\mathrm{mix}})$ and $\exp(\mathrm{i}\gamma_j H_{\mathrm{obj}})$ are applied in alternating manner, resulting in the ansatz

$$U_{\mathrm{QAOA}} = \prod_{j=1}^{l} \exp(\mathrm{i}\beta_j H_{\mathrm{mix}}) \exp(\mathrm{i}\gamma_j H_{\mathrm{obj}}), \tag{2.10}$$

where $H_{\mathrm{mix}} = -\sum_i \sigma_x^i$, σ_x is the Pauli X operator. The computational power and reachability of the QAOA ansatz are rigorously discussed in [62, 96, 111].

While it's true that breaking down Eq. (2.10) into native gates may result in a long circuit, this is often due to the presence of many-body terms in H and limited device connectivity. However, one notable advantage of this ansatz is that for certain problems, the feasible subspace is smaller than the full Hilbert space. This restriction can possibly lead to better algorithmic performance, making this approach highly effective in certain scenarios.

Variable structure Although the constancy in the structure of the ansatz while solving a problem enables control over the overall ansatz complexity, it fails to harness the refinement that is attained by optimizing the circuit. The refinement can be realized through the addition or removal of unnecessary quantum operators. This novel approach was first introduced in ADAPT-VQE [56], which adaptively inserts a quantum fermionic operator to provide a desired level of accuracy while maintaining a minimal number of operators. The operator is chosen in such a

way that it affects the minimization of energy the most. The operators are chosen from a pool of fermionic operators, and it has been shown that the ADAPT-VQE substantially outperforms UCCSD in terms of both the number of variational parameters and accuracy. Following this trail in [148], the authors introduce a hardware-efficient variant of ADAPT-VQE, i.e. qubit ADAPT-VQE, which drastically reduces the operator pool. Other more efficient variants of ADAPT-VQE are introduced in [167].

Another way of variable ansatz construction is introduced in [87], where the ansatz is allowed to grow. And if the algorithm cannot minimize the cost function for a specified number of iterations, then one adds an identity gate spanned by new variational parameters that are randomly added to the ansatz.

2.2 Basics of reinforcement learning

One of the most prominent sub-fields of machine learning is Reinforcement Learning (RL), which is concerned with how an *agent* can learn to make decisions in an *environment* to maximize a cumulative *reward function*. The primary goal of the *agent* is to learn a *policy*, which represents a mapping from the states of the environment to a decision (i.e. an *action*), that in turn maximizes cumulative reward.

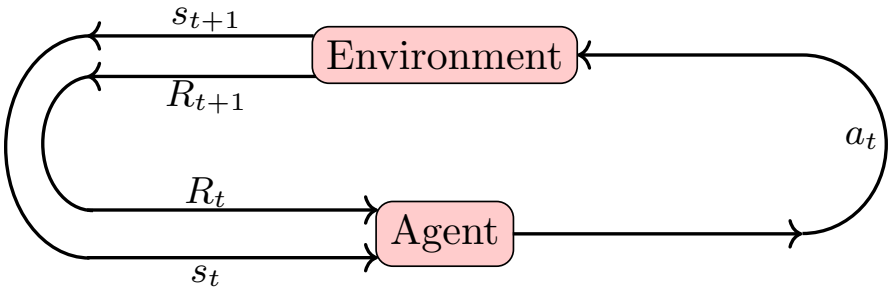


Figure 2.6: Illustration of the agent and the environment interaction.

RL is intrinsically distinct from *supervised learning* [40, 61, 88] (SL) because, in supervised learning, the learning process is conducted from a training set of labeled exemplary data, which is in turn provided by a knowledgeable external supervisor. Each exemplary data corresponds to a situation, and the label denotes the action the system should take in that situation. In RL, the agent's learning process is mediated through interaction with the environment.

Meanwhile, RL differs from unsupervised learning [11, 42, 52] (USL) in the sense that in USL, the main ambition is to find structure hidden in connections of unlabeled data. Whereas in RL all one tries to achieve is the maximization

Name	Notation	Definition
State	s	Refers to the current situation of an agent in an environment, which includes all the relevant information necessary to make decisions about what action to take next
Action	a	Refers to the choices an agent can make based on its current state to interact with the environment. The agent chooses the action by using a policy.
Policy	π	Refers to a mapping from perceived states of the environment to actions to be taken when in those states.
Reward	R	Refers to the goal in a problem. On each time step, the environment sends to the agent a single number, a reward. The agent's sole objective is to maximize the total reward it receives over the long run.
Value Function	v	It refers to the value of a state as the total amount of reward an agent can expect to accumulate over the future, starting from that state.
Transition probability	$p[s'\|s, a]$	It refers to the probability of transition to state s' if action a is taken on the state of the environment s.

Table 2.1: The necessary notations and their definitions utilized to provide an introduction to RL and in the upcoming sections.

of the reward function signal instead of trying to find a hidden structure. In the following, we will briefly specify RL problems in terms of optimal control of the Markov decision process (MDP) and the challenges in RL.

The summary of notations and definitions used in this thesis in the context of reinforcement learning is provided in Table 2.1.

2.2.1 Finite Markov decision process

In this section, we discuss the mathematical form of reinforcement learning problems, which includes key elements of the problems' structure, such as the value function and Bellman equation.

Environment-agent interaction In RL, the *agent*, which is the learner and decision-maker, interacts with everything outside it called the *environment*.

The whole RL process can be characterized by time steps $t = 0, 1, \ldots, T$ where at each time step, the agent receives the state of the environment $s_t \in \mathcal{S}$ where $\mathcal{S}$ is the set of all possible states. After interacting with the environment, the agent gives out an action $a_t \in \mathcal{A}(s_t)$, where $\mathcal{A}(s_t)$ defines the space of all actions available in the state s_t. In the next time step, the environment gives out (1) a new state: $s_t \to s_{t+1}$ and (2) a reward $R_{t+1} \in \mathcal{R} \in \mathbb{R}$, which is depicted in the Fig. (2.6).

At each time step t, the agent forms a mapping between the states to the probabilities of selecting each possible action. This mapping is termed as *Policy* (π_t), where $\pi_t(a|s)$ tells us the probability of selecting the action $a_t = a$ if the environment is at state $s_t = s$.

The above-mentioned framework can be used for many different problems. As a fundamental principle in the study of reinforcement learning, we adhere to the notion that an agent's environment comprises all elements that are beyond its arbitrary control, forming a context in which the agent operates. The boundary between the agent and the environment represents the limit of the agent's *absolute control*, not its knowledge. For example, the agent might know almost everything about the interacting environment, but still, it faces difficulty in solving the task just as we know how Rubik's cube works and still might not be able to solve it.

At its core, the RL framework represents a powerful abstraction of the challenge of learning to achieve goals through interaction with an environment. Any problem with learning goal-directed behaviour can be efficiently reduced to three sequences of processes and repeating back and forth. (1) the choice, that is made by the agent, which we call *action* (2) the basis upon which choices are determined i.e. the *states* and (3) the definition of the agent's goal in the form of *reward*. In the next, we briefly describe the *rewards* with a simple example.

Rewards and returns At each time step t, the reward is defined by a real number $R_t \in \mathbb{R}$ which the agent receives from the environment as a signal. The main purpose of the agent is to maximize the cumulative reward in the long run. Formulation of a goal corresponding to a problem in terms of reward is a hard problem. Based on a problem one can define either a (1) sparse or (2) dense reward, where the sparse reward is easier to formulate than the dense one.

For example in the case of a robot in a maze problem, the goal is to teach the robot to escape from the maze. If we formulate the goal in terms of a dense reward where for each time step t, for each movement of the robot in the maze, the agent receives a reward signal $R_t = -1$. This encourages the robot to find a solution to escape the maze quickly as each step of the robot is penalized (-1) and the robot will focus on minimizing the penalty.

However, the same problem can be formulated using a sparse reward where the agent receives a reward when it escapes the maze, but it slows down learning because the agent needs to take many actions before getting any reward. In problems like chess or backgammon, the only way to formulate the reward is by giving the player a big reward if wins. This is known as the *credit assignment problem*.

If we consider that after each time step the agent receives a reward $R_{t+1}, R_{t+2}, \ldots$ then we seek to maximize the expected return G_t. The G_t is defined by

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots + R_T, \tag{2.11}$$

where T is the final time step. Considering the notion of the final time step, the agent– environment interaction can be broken into subsequences, which we call *episodes*. An episode might be represented by a game of chess or a complete trip in the maze by a robot. Meanwhile, in many problems, it might not be possible to break the interaction between the agent and the environment, which can not be broken into identifiable episodes but goes on continually without limit. For these tasks, the definition of return in Eq. (2.11) is not valid, and it can be infinite.

An important additional concept to consider while formulating a reward function is called *discounting*. With this approach, the agent aims to choose actions that maximize the total of discounted rewards it will receive in the future as follows

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{j=0}^{\infty} \gamma^j R_{t+j+1}, \tag{2.12}$$

where γ is the discount rate parameter that ranges as $0 \leq \gamma \leq 1$. It determines the present value of the future rewards, i.e. a reward received at j-th time in the future is worth only γ^{j-1} times what it would be worth if it were received immediately.

If $\gamma < 1$, the infinite sum has a finite value as long as the reward sequence $\{R_j\}$ is bounded. If γ is close to 0, the agent is *myopic* i.e. the agent is concerned only with maximizing immediate rewards. If γ is close to 1, the agent is *far-sighted*. In the following, we provide an elaborate example of how a *myopic* and *far-sighted* agent impacts the cumulative return.

Toy Example Before we move on further, let us take a simple example and run through the process of formulating the G step by step. In Fig. (2.7), we give an illustration of a work-sleep schedule of a PhD student as an MDP. In the problem, we consider *sleep* as the terminal step. There are many possible ways to reach the terminal step, such as

- Awake → Work → Sleep

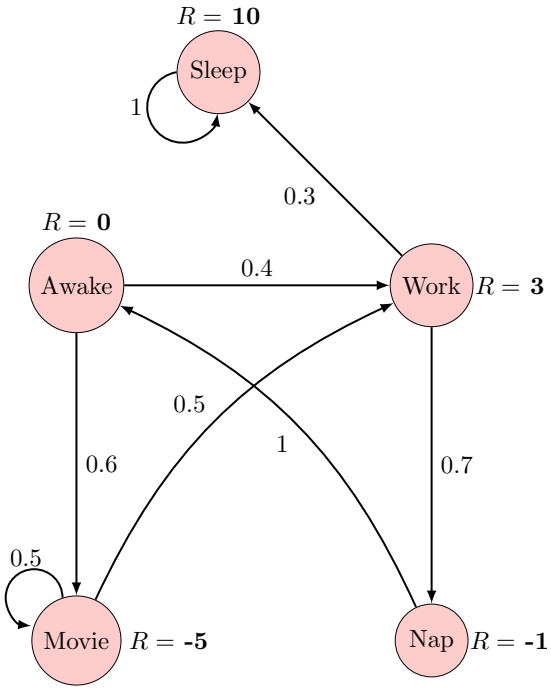


Figure 2.7: A toy MDP of the everyday schedule of a PhD student.

- Awake → Work → Nap → Awake → Movie → Work → Sleep etc.

The transition matrix corresponding to the MDP is defined by

$$p[s' = s_{t+1}|s_t] = \begin{array}{c|ccccc} & \text{Awake} & \text{Movie} & \text{Nap} & \text{Sleep} & \text{Work} \\ \hline \text{Awake} & & 0.6 & & & 0.4 \\ \text{Movie} & & 0.5 & & & 0.5 \\ \text{Nap} & 1 & & & & \\ \text{Sleep} & & & & 1 & \\ \text{Work} & & & 0.7 & 0.3 & \end{array} . \quad (2.13)$$

As all the states are associated with a particular reward, we can calculate the discounted return using Eq. (2.12). For an example for the path: Awake → Movie → Work → Sleep, if we are currently at the state “Awake" then the discounted reward

for the "Awake" state becomes

$$G_{\text{Awake}} = R_{\text{Movie}} + \gamma R_{\text{Work}} + \gamma^2 R_{\text{Sleep}} = -5 + 3\gamma + 10\gamma^2, \tag{2.14}$$

- **Myopic agent:** If we consider γ is close to 0 say $\gamma = 0.1$ then we get $G_{\text{Awake}} = -4.6$,
- **Far-sighted agent:** On the other hand, if we consider $\gamma = 0.9$, which is close to 1 then we get $G_{\text{Awake}} = +5.8$.

From this, we can say that for the *far-sighted case*, the agent might prefer to take the route Awake → Movie → Work → Sleep. But a *myopic agent* is more physical because animals have a preference for immediate reward. In the following part, we elaborate on Markov decision processes (MDPs) and the importance of value function in RL.

Markov decision process The Markov decision processes (MDPs) provide a modelling framework for sequential decision problems [94], as shown in the example of the previous section. MDPs have something called *Markov property*, which is defined by the dependency of the next state of the environment on the current state and the action. To express it more elaborately, we consider an environment that might respond at the time $t+1$ corresponding to the action taken at t. In the most general case, the nature at $t+1$ may depend on everything that happened in the past events from $t = 0$ to $t = t$ i.e. the probability of the state appearing on state $s' = s_{t+1}$ with reward $r = R_{t+1}$ can be written as

$$\mathrm{P}\left[R_{t+1} = r, s_{t+1} = s' | s_0, a_0, R_1, \ldots s_{t-1}, a_{t-1}, R_t, s_t, a_t\right]. \tag{2.15}$$

On the contrary, if the state has *Markov property* then the same probability can be rewritten as

$$p(r, s' | s_t, a_t) = \mathrm{P}\left[R_{t+1} = r, s_{t+1} = s' | s_t, a_t\right], \tag{2.16}$$

which says that *the future state of the system depends solely on the current state and the action taken, rather than on any prior history* [66]. This means instead of memorizing all the information about all the past events and defining the nature of the environment at $t+1$ the agent now can characterize $t+1$ by just inquiring about its previous event at time t.

One can define an MDP as a 5-tuple process ($\mathcal{S}$, $\mathcal{A}$, $p(s'|s_t, a_t)$, R, γ) depending on:

- $\mathcal{S}$, the finite set of all states of the environment;

- $\mathcal{A}$, the finite set of all legal actions that can be executed at state $s_t \in \mathcal{S}$;
- $p[s_{t+1} = s'|s_t, a_t]$, the probability of transition to state s' at the time $t+1$ if a_t action is taken on the state of the environment s_t;
- $R \in \mathcal{R}$, the reward received when the environment transitioning from s_t to s' after action a_t;
- $\gamma \in [0, 1]$, is the discount factor that represents the difference in the future and present reward.

If the dynamics are specific by Eq. (2.16), one can compute the expected reward $r(s, a)$ as

$$\sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(r, s'|s_t, a_t), \tag{2.17}$$

from the state-action pair. The state transition probability $p(s'|s_t, a_t)$ is expressed as

$$p(s'|s_t, a_t) = \sum_{r \in \mathcal{R}} p(r, s'|s_t, a_t), \tag{2.18}$$

and the expected rewards for the state–action–next-state $r(s, a, s')$ as

$$\sum_{r \in \mathcal{R}} \frac{rp(r, s'|s_t, a_t)}{p(s'|s_t, a_t)}. \tag{2.19}$$

Value function In RL the value function provides a measure of how good it is for an RL-agent to take a specific action at a particular state. It helps the agent to make decisions in an uncertain environment by quantifying the expected future reward that an agent can get for that particular state. The value functions are defined with respect to specific policies. Recalling that a policy, π, at time step t, is defined by a mapping from each state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$ to the probability $\pi(s_t, a_t)$ of taking the action a_t in state s_t. This inevitably gives rise to two main types of value functions in RL based on the state and the action as follows:

- **State-value function**, $v_\pi(s)$ is represented by the expected return starting from a particular state s and by following a particular policy π. The state value function encapsulates the inherent value associated with each s, irrespective of the action taken. For MDPs, one can define the state-value function as

$$v_\pi(s_t) = \mathbb{E}_\pi \left[\sum_{j=0}^{\infty} \gamma^j R_{t+j+1} \middle| s_t = s \right], \tag{2.20}$$

  where $\mathbb{E}_\pi[.]$ gives the expected value given that the agent follows policy π.

- **Action-value function**, $q_\pi(s, a)$ is represented by the expected return starting from a particular state s, taking an action $a \in \mathcal{A}$ and by following a particular policy π. It estimates the long-term value of taking a particular action in a given state. For MDPs, one can define the action-value function as

$$q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{j=0}^{\infty} \gamma^j R_{t+j+1} \middle| s_t = s, a_t = a \right]. \tag{2.21}$$

The value function in Eq. (2.20) can be decomposed into two parts: (1) the immediate reward and (2) the discounted value of the successor rate in the following way:

$$\begin{aligned}
v_\pi(s_t) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \dots \middle| s_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \dots \right) \middle| s_t = s \right] \\
&= \mathbb{E}_\pi \left[\underbrace{R_{t+1}}_{\substack{\text{Immediate} \\ \text{reward}}} + \gamma \underbrace{\sum_{j=0}^{\infty} \gamma^k R_{t+j+2}}_{\substack{\text{Discounted} \\ \text{successor rate}}} \middle| s_t = s \right] \\
&= \sum_a \pi(a|s) \sum_{s'=s_{t+1}} \sum_{r=R_{t+1}} p(s', r|s, a) \left[r + \gamma \mathbb{E}_\pi \left(\sum_{j=0}^{\infty} R_{t+k+2} \middle| s' \right) \right] \\
&= \sum_a \pi(a|s) \sum_{s', r} p_{ss'} \left[r + \gamma v_\pi(s') \right].
\end{aligned} \tag{2.22}$$

The Eq. (2.22) is known as the *Bellman expectation equation for the state-value function*. This is basically a sum of all values of the action, the current and the successor state of the environment. For each value of the variables, we compute the $\pi(s, a)p(s', r|s, a)$, multiply it by the linear weighted term $[r + \gamma v_\pi(s')]$ and sum over all possible values of the three variables to get expected value. In the same way using Eq. (2.21) we can find the *Bellman equation for the action-value function* as follows:

$$q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v(s_{t+1}, a_{t+1}) \middle| s_t = s, a_t = a \right]. \tag{2.23}$$

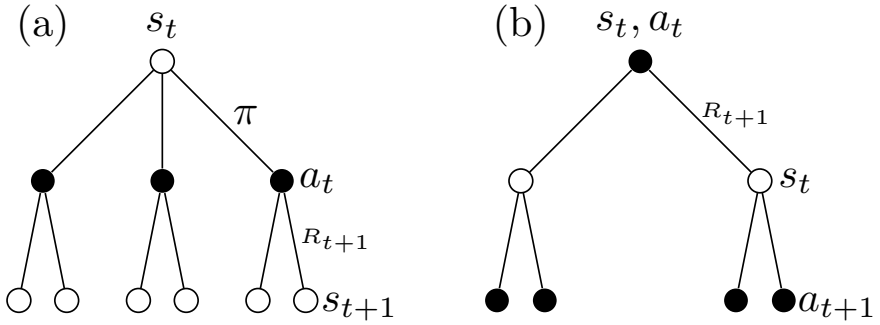


Figure 2.8: Simple illustration of the backup diagrams for (a) state-value function (v_π) and (b) action-value function (q_π). The time flows downwards.

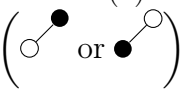
Here we define the *Bellman operator* T^π as

$$(T^\pi q)(s_t, a_t) = R_{t+1} + \gamma \mathbb{E}\left[q(s_{t+1}, a_{t+1}) \middle| s_{t+1}, a_{t+1}\right], \tag{2.24}$$

where s_{t+1} is samples with probability $p(s_{t+1}|s_t, a_t)$ and a_{t+1} is sampled from the policy $\pi(a|s)$. In the same manner, the *Bellman optimality operator* is defined by

$$(T^* q)(s_t, a_t) = R_{t+1} + \gamma \mathbb{E}\left[q^*(s_{t+1}, a_{t+1}) \middle| s_{t+1} \sim p(s_{t+1}|s_t, a_t)\right]. \tag{2.25}$$

The operator defined in Eq. (2.25) will be useful while we discuss the Deep Q-Network (DQN).

There are many possible ways to approximate, compute, and learn the value function for a particular policy, and all these methods are diagrammatically represented using *backup diagrams*. These diagrams visualize how the value function is updated based on new information received from the environment. In other words, backup diagrams depict the flow of information and the update process involved in processing the value function. This represents how the estimated values of state or state-action pair are modified after receiving feedback from the environment. In each *backup diagram* the states (○) and actions (●) are represented by vertices and the transaction among them are edges ( or)

For example, in Fig. (2.8) we illustrate the *backup diagrams* for v_π and q_π.

2.2.2 Optimal value function

The optimal value function is defined by the maximum expected return that an agent can achieve by following an optimal policy. A policy π is defined to be better

or equivalent to another policy π' i.e. $\pi \geq \pi'$ iff $v_\pi(s_t) \geq v_{\pi'}(s_t)\ \forall\ s_t \in \mathcal{S}$. There can be more than one *optimal policy*, which we denote by π^*. These policies share the same state-value function, which is called by *optimal state-value function*, v^* and is defined by

$$v^*(s_t) = \max_\pi v_\pi(s_t), \tag{2.26}$$

and in the same manner the *optimal action-value function*, q^*, is given by

$$v^*(s_t) = \max_\pi q_\pi(s_t, a_t), \tag{2.27}$$

where $a_t \in \mathcal{A}(s_t)$. Using Eq. (2.23) we can rewrite the above equation in terms of v^* as

$$q_*(s_t, a_t) = \mathbb{E}\left[R_{t+1} + \gamma v^*(s_{t+1}, a_{t+1}) \middle| s_t = s, a_t = a\right]. \tag{2.28}$$

As we saw in the previous section, one can obtain *Bellman equation* directly from either the state-value function or the action-value function. In the same way, from the optimal state-value and action-value function, we get *Bellman optimality equation*. The intuition behind the Bellman optimality equation lies in the principle of optimality, which states that an optimal policy can be divided into sub-problems and solved independently for each state. In other words, it decomposes the problem of finding the optimal value function into sub-problems for each state, allowing for a dynamic programming approach to solving MDPs. The Bellman optimality equation for the state-value function is given by

$$\begin{aligned}
v_*(s_t) &= \max_{a_t \in \mathcal{A}} \mathbb{E}_{\pi^*}\left[R_{t+1} + \gamma R_{t+2} + \ldots \middle| s_t = s, a_t = a\right] \\
&= \max_{a_t \in \mathcal{A}} \mathbb{E}_{\pi^*}\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \middle| s_t = s, a_t = a\right] \\
&= \max_{a_t \in \mathcal{A}} \mathbb{E}_{\pi^*}\left[R_{t+1} + \gamma \sum_{j=0}^{\infty} \gamma^k R_{t+j+2} \middle| s_t = s, a_t = a\right] \\
&= \max_{a_t \in \mathcal{A}} \mathbb{E}\left[R_{t+1} + \gamma v_*(s') \middle| s_t = s, a_t = a\right] \qquad (2.29) \\
&= \max_{a_t \in \mathcal{A}} \sum_{s', r} p(s', r|s, a)\left(r + \gamma v_*(s')\right). \qquad (2.30)
\end{aligned}$$

In the Eq. (2.29) and Eq. (2.30) is the Bellman optimality equation for v_*.

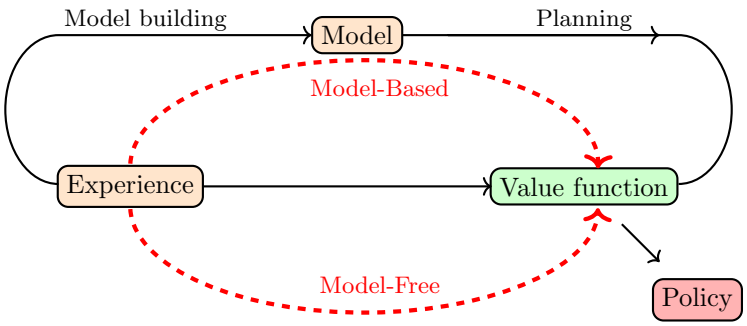


Figure 2.9: Illustration of model-based and model-free RL.

Following the same steps, we can obtain the Bellman optimality equation for q_* as

$$q_*(s_t, a_t) = \max_{a_t \in \mathcal{A}} \mathbb{E}\left[R_{t+1} + \gamma v_*(s') \middle| s_t = s, a_t = a \right], \tag{2.31}$$

$$= \max_{a_t \in \mathcal{A}} \sum_{s', r} p(s', r|s, a) \left(r + \gamma v_*(s') \right). \tag{2.32}$$

Having the (optimal) value functions at our disposal, we can now talk about how to solve a reinforcement learning problem. There are fundamentally two approaches used in solving reinforcement learning (RL) problems: (1) Model-Based RL [75, 107, 123] and (2) Model-Free RL. In the case of Model-Based RL, we have complete knowledge of the dynamics of the Markov decision process, which includes precise knowledge of the transition probabilities and rewards for each transition. Hence, to solve the Model-Free RL problem, *value iteration algorithm* [5, 13]can be utilized. It iteratively evaluates the optimal value of each state by taking the expected intermediate reward and the values for the successor states into account. For the sake of coherence regarding the thesis topic, we end the discussion on the *Model-Based RL* here. In the following, we briefly discuss the *Model-Free reinforcement learning* approach.

2.2.3 Model-Free learning

As previously discussed, the *value iteration algorithm* requires complete knowledge of the model, such as transition probabilities and rewards. In many scenarios, we encounter situations where the agent lacks knowledge of transition probabilities, necessitating alternative approaches to compute the policy function. The model-free approach is developed by keeping these encounters in mind. We compare the value- and policy-based approaches in Tab. 2.2. Meanwhile, a comparison of the model-based and model-free reinforcement learning is illustrated in Fig. (2.9).

Value-based	Policy-Based
Aims to learn the optimal state-value function or action-value function, i.e. Q-function directly.	Aims to learn the optimal policy directly, without explicitly estimating value functions.
These algorithms depends on the balance between exploration and exploitation. It explores new states and actions while also exploiting the value estimates.	These algorithms optimize the policy $\pi(s_t, a_t\|\vec{\theta})$ by modifying the $\vec{\theta}$ to maximize the expected cumulative reward.
A popular value-based algorithm that learns the Q-value through an iterative updating method is called *Q-learning* [162, 163]. This method updates the value based on observed rewards and transition probability.	In a policy-based algorithm gradient ascent method [141, 170] is used to update policy parameters.

Table 2.2: Comparison table for value-based and policy-based methods.

Model-based reinforcement learning involves a subroutine of creating an internal representation of how the environment behaves, allowing the agent to predict future states. Meanwhile, instead of learning from a predefined model, model-free reinforcement learning focuses on learning the best actions in different situations by estimating the value or policy directly from observed experiences. For the sake of this thesis, in the remaining sections, we will focus on value-based model-free algorithms.

Temporal difference learning In the value iteration process, the state-value function is calculated recursively using the Eq. (2.22). In model-free learning, the transition function is replaced by a sequence of the sample from the environment, and we can use Bellman's recursive computation to estimate the new updates to the value function based on the previous estimates.

Temporal difference learning [145] (TD) is a bootstrapping method that can be used to process and refine the samples to achieve an approximate final value of the state. As the name suggests, it refers to the difference in the values of the states at two different time steps. This information is then used to calculate the value at

the new time step. TD learning method is given by

$$v(s_t) \leftarrow v(s_t) + \alpha \left[R_{t+1} + \gamma v(s_{t+1}) - v(s_t) \right], \tag{2.33}$$

recalling s_t is the state of the environment at time step t, and s_{t+1} is the new state at the time $t+1$. The reward we receive for the transition from s_t to s_{t+1} is R_{t+1}. As we described before, γ is the discount factor, γ set closer to zero, represents a myopic agent, and if it is closer to 1, we get a far-sighted agent. The new variable α is the learning rate, which determines how fast the algorithm learns. The essence of temporal difference lies in the last term of Eq. (2.33), where we subtract "$-\, v(s_t)$" from the current state value to compute the temporal difference. The TD learning method revolutionized the use of model-free approaches in different RL scenarios. One remarkable achievement was TD-Gammon [151], a program that defeated human world champions in the game of Backgammon during the early 90s.

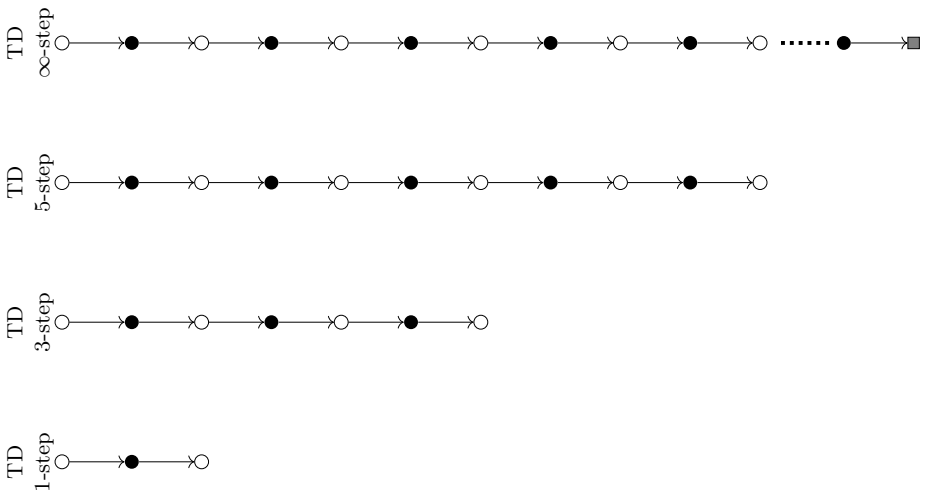


Figure 2.10: Illustration of 1, 5 and ∞-step TD learning. The ∞ step learning is equivalent to the Monte Carlo learning method.

Unlike the Monte Carlo learning method, which performs a full episode with various random actions before it utilizes the reward, the temporal difference bootstraps the Q-function with the help of the values that it gathers from the previous time steps. This helps to refine the value function with the cumulative rewards after each time step (1-step is defined by, ○⟶●⟶○, the arrow dented the flow of time). Based on this description, we can think of a middle ground with *n-steps*, which fundamentally points to the fact that we neither sample a single step like TD learning nor do we sample a full episode like Monte Carlo, but we sample a few steps (say, n steps) at a time before utilizing the reward values, which we

illustrate this in Fig. (2.10). This strategy allows for a more granular assessment of state values compared to TD learning, which may lead to faster convergence and reduced variance in the estimated values. Moreover, one of the advantages of the n-step approach is that it does not require the entire episode to be completed before updating, making the n-step TD learning computationally efficient over Monte Carlo methods, particularly in environments with long episodes. This middle-ground approach can thus be particularly advantageous in RL tasks where balancing the trade-offs between bias and variance is crucial for efficient learning and decision-making.

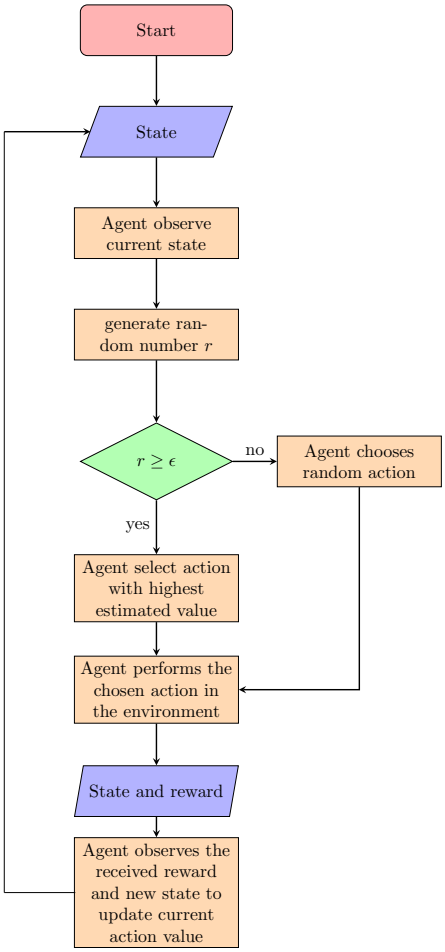


Figure 2.11: Block diagram of an ϵ-greedy algorithm. In this algorithm, a parameter ϵ is used to determine the probability of choosing a random exploration action versus selecting the action with the highest expected reward. When ϵ is set to a small value, the algorithm tends to exploit the current best action, while a higher value encourages more exploration.

Exploration and exploitation In the context of the model-free RL exploration/-exploitation are two primary concepts that are most frequently used in designing RL algorithms. The *exploration* part indicates the process of collecting environment information of the agent by taking random actions. The goal of this routine is to discover new states of the environment, actions, and their corresponding rewards. Meanwhile, *exploitation* is utilized to leverage the information that is gathered by maximizing the immediate rewards, and it focuses on the selection of actions that will provide the highest reward based on the existing knowledge of the environment. Hence, the *exploration* and *exploitation* complement each other.

One of the fundamental challenges in RL is to balance exploitation and exploitation because if the agent only focuses on exploration it never gets the chance to leverage the existing information whereas more exploitation of the immediate knowledge and the algorithm may get stuck in a non-optimal policy. This means the algorithm has failed to find a better strategy that leads to an optimal policy.

ϵ-greedy exploration It is a well-known strategy in RL to maintain a balance between the exploration and exploitation aspect. The ϵ-greedy method works by assigning a parameter ϵ, which encodes the information regarding the probability of exploration as

$$\text{Selection of action by agent} = \epsilon a_{\text{r}} + (1 - \epsilon) a_{\text{ev}}, \tag{2.34}$$

where a_{r} is the random action and a_{ev} is the action with highest estimate value. This algorithmic choice presented in Eq. (2.34) is called the *exploration/exploitation trade-off*. An illustration of the *epsilon*-greedy algorithm is presented in Fig. (2.11). There are other methods to define the exploration aspect, either by using the Thompson sampling [132, 139] or adding the Dirichlet-noise [81].

2.2.4 On-policy and off-policy learning

The RL deals with learning an action and a policy from the received rewards. That is, the agent selects an action to perform on the environment and learns from the reward that it receives after taking the action. The agent learns to select an action for the next time-step. Now, the dilemma arises if the agent either updates from its most recent action (*on-policy*) or learns from all the available information gathered (*off-policy*). A tabular representation of the difference between the two kinds of learning is presented in Tab. 2.3.

It should be noted that the on- and off-policy learning shows different behaviour in convergence to the optimal policy. Such as the off-policy method promises to converge to the optimal policy after sampling a sufficient number of states, so it uses a greedy reward approach. At the same time, the on-policy method for a fixed value of ϵ can not converge to the optimal policy as they keep on selecting the actions that are based on the current action. Meanwhile, when we use a policy where we keep on varying ϵ towards zero, then the on-policy method promises to converge.

In the following, we will briefly discuss two famous algorithms: (1) on-policy SARSA and (2) off-policy Q-learning.

SARSA Is an on-policy algorithm that was first proposed by Rummery and Niranjan [131]. The on-policy methods update the current policy by utilizing the

On-policy	Off-policy
The current policy determines the action to take and the value of that action is used to update the policy function	The determination of action takes place by backing up values of other action which is not necessarily selected by the behavior policy
This learning process is stable since the agent updates the policy based on experiences from the current policy, making the learning process more consistent	Off-policy learning although less stable compared to on-policy learning but here, the agent is open to learning from a diverse set of experiences, which has the potential to improve the exploration and discover better policies.
The agent's exploration might be biased, which causes difficulty in exploring actions that are not properly defined on policy, leading to sub-optimal policy.	It requires more data and computational resources compared to on-policy learning as it learns from all the available information gathered before.

Table 2.3: A comparison of on- and off-policy learning.

action value of the policy itself. Hence, the update rule is given by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]. \tag{2.35}$$

It should be noted that SARSA follows the same updating method as the TD method as presented through Eq. (2.33). The only difference is that the state-action value function replaces the state-value function.

SARSA updates the Q-values of the current state-action pair (s_t, a_t) by using the Q-values of the next state-action pair (s_{t+1}, a_{t+1}). To say it in a more elaborate manner, in the SARSA algorithm, we select an action and apply it to the environment, and then it follows the action that is guided by the behaviour policy. The behaviour policy is defined by either the ϵ-greedy approach. Hence, in this learning process, the state space sampling is done by following the behavioural policy and updating the current policy by updating the values of the actions based on the sampling.

Q-learning The Q-learning algorithm was first proposed by Watkins [38]. Unlike SARSA, Q-learning learns the Q-values using a different policy than the one it followed during the action selection process. The updated formula for Q-learning is given by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \tag{2.36}$$

The primary difference between SARSA and Q-learning update rule is the term $Q(s_{t+1}, a_{t+1})$ is replaced by $\max_a Q(s_{t+1}, a)$. This indicates that it now learns from the stored values of the best action instead of the action that was actually evaluated. The main reason Q-learning is called off-policy is that it updates the Q-values using the Q-value of the next state s_{t+1} and a greedy action which is not necessarily the action of the behaviour policy.

One of the drawbacks of Q-learning is that as the size of the Q-table grows exponentially with the increase in the number of states and actions, it becomes infeasible. To tackle this hindrance, we can combine Q-learning with the deep neural network, which is known as a *Deep Q-Network* (DQN) [110].

Deep Q-Network In this algorithm, a deep neural network in the form $Q_\beta : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is utilized to approximate the optimal Q function Q^*, where β is the parameter of the neural network. There are two essential tricks to be noted to achieve success with DQN, and those are [106]:

- The usage of *experience reply* [93] in DQN helps to obtain uncorrelated samples since the trajectory of MDP has a strong temporal correlation. Specifically, for each time step t, we tend to store the tuple (s_t, a_t, r_t, s_{t+1}) into the reply memory, which is followed by the sampling of a mini-batch of independent samples from the replay memory. This is used to train the deep neural network using a stochastic gradient descent method.

- The utilization of a target network Q_{θ^*} with parameter θ^* is another trick which we use in DQN. After collecting the independent samples (s_i, a_i, r_i, s_{i+1}) from the replay memory, to update the parameters of the Q-network, we evaluate the target network as follows:

$$E^i_{\text{targ}} = r_i + \gamma \times \max_{a \in \mathcal{A}} Q_{\theta^*}(s_{i+1}, a), \tag{2.37}$$

  and then updating θ using the gradient of

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[E_{\text{targ}} - Q_\theta \left(s_i, a_i \right) \right]^2. \tag{2.38}$$

Parameter θ^* is updated once in every T i.e. the target network is held fixed for T steps, and then we update it by the current weight of the Q-network.

In [106] the authors utilize a replay memory of size 10^6 on the other hand, in [116] the authors use a replay memory of 2×10^4. This indicates that the replay memory size is usually very large. Additionally, in DQN, we use ϵ-greedy policy to enable exploration over the state and action. In this scenario, when the replay memory is large, the experience replay is prone to sample independent transitions from an exploration-driven policy (as it is ϵ-greedy), which decreases the variance of the term $\Delta L(\theta)$. As the $L(\theta)$ plays a vital role in updating the parameters of a neural network, so having a low variance in $L(\theta)$ stabilizes the training of DQN.

Meanwhile, to understand the necessity of a target network, we first set $\theta^* = \theta$. The bias-variance decomposition gives us the expected value of the $L(\theta)$ as

$$\mathbb{E}[L(\theta)] = \underbrace{||Q_\theta - T^*Q_\theta||_\sigma^2}_{\substack{\text{mean-squared} \\ \text{Bellman equation}}} + \underbrace{\mathbb{E}[E^1_{\text{targ}} - (TQ_\theta)(s_1, a_1)]^2}_{\substack{\theta \text{ dependent} \\ \text{variance of} \\ E^1_{\text{targ}}}}, \tag{2.39}$$

where T^* is the *Bellman optimality operator* defined in Eq. (2.25). In Eq. (2.39) along with the mean-squared Bellman error, we have an additional bias of the form $\mathbb{E}[E^1_{\text{targ}} - (TQ_\theta)(s_1, a_1)]^2$, that is strictly dependent on the neural network parameter. This indicates that minimizing the $L(\theta)$ can drastically differ from minimizing the mean-squared Bellman equation. This issue is by using the target network in Eq. (2.38) what has an expectation value

$$\mathbb{E}[L(\theta)] = \underbrace{||Q_\theta - T^*Q_\theta||_\sigma^2}_{\substack{\text{mean-squared} \\ \text{Bellman equation}}} + \underbrace{\mathbb{E}[E^1_{\text{targ}} - (TQ_{\theta^*})(s_1, a_1)]^2}_{\substack{\theta \text{ independent} \\ \text{variance of} \\ E^1_{\text{targ}}}}, \tag{2.40}$$

as the second term is independent of θ so minimizing $L(\theta)$ is approximately equivalent to solving

$$\underset{\theta \in \Theta}{\text{minimize}} ||Q_\theta - T^*Q_\theta||_\sigma^2, \tag{2.41}$$

where Θ is the parameter space. In simple words, the ultimate aim of DQN is to solve the minimization problem defined through Eq. (2.41) with a fixed θ^* and it updates the θ^* by the minimizer parameter of the neural network θ.

An implementable version of the above-discussed DQN is known as the neural Fitted Q-Iteration (FQI) algorithm. This generates a sequence of value functions. Let us consider $\mathcal{F}$ is a class of functions on the state-action space. For the j-th iteration of the algorithm, we consider $\tilde{Q}_j$ is the present estimate of the Q^*. Hence,

Algorithm 1 FQI algorithm

MDP as tuple $(\mathcal{S}, \mathcal{A}, p, \mathcal{R}, \gamma)$, define $\mathcal{F}$, sampling distribution σ, total iterations J, sample number n, initial estimator $\tilde{Q}_0$. ▷ Input

for $j = 0, \ldots, J-1$ **do**

Sample i.i.d. $\{(s_j \in \mathcal{S}, a_j \in \mathcal{A}, R_j \in \mathcal{R}, s_{j+1})\}_{i\in[n]}$, with (s_i, a_i) sampled from σ.

Compute $E^j_{\text{targ}} = R_j + \gamma \times \max_{a\in\mathcal{A}} \tilde{Q}_j(s_{i+1}, a)$

Update the action-value function

$$\tilde{Q}_{k+1} \leftarrow \operatorname*{argmin}_{f\in\mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left[E^i_{\text{targ}} - f(s_i, a_i)\right]^2$$

end for

Define the policy π_J as a greedy policy in respect with $\tilde{Q}_J$

An estimator $\tilde{Q}_J$ of Q^* and policy π_J. ▷ Output

the update rule of $\tilde{Q}_j$ is defined as

$$\tilde{Q}_{k+1} = \operatorname*{argmin}_{f\in\mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left[E^i_{\text{targ}} - f(s_i, a_i)\right]^2. \tag{2.42}$$

We can replace the $\mathcal{F}$ by the neural networks, and then the algorithm is known as neural FQI [126]. Therefore, we can consider neural FQI as a variation of DQN, where we substitute experience replay with sampling from a stable distribution to understand the statistical characteristics.

Double Deep Q-Network Deep RL methods employ neural networks to adapt the agent's policy for optimizing the return

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

with the discount factor $\gamma \in [0, 1)$. Each state and action pair (s, a) can then be assigned an action-value that quantifies the expected return from state s in step t taking action a under policy π

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid s_t = s, a_t = a\right].$$

The aim is to find the optimal policy that maximizes the expected return. Such a policy can be derived from the optimal action-value function q_*, defined by the

Bellman optimality equation:

$$q_*(s,a) = \mathbb{E}\left[r_{t+1} + \max_{a'} q_*\left(s_{t+1}, a'\right) \mid s_t = s, a_t = a\right].$$

Instead of directly solving the Bellman optimality equation in value-based RL, the aim is to learn the optimal action-value function from data samples. One such prominent value-based RL algorithms is Q-learning, where each state-action pair (s,a) is assigned a so-called Q-value $Q(s,a)$ which is updated to approximate q_*. Starting from randomly initialized values, the Q-values are updated according to the following rule:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(r_{t+1} + \gamma \max_{a'} Q\left(s_{t+1},a'\right) - Q(s_t,a_t)\right),$$

where α is the learning rate, r_{t+1} is the reward at time $t+1$, and s_{t+1} is the next encountered state after taking action a_t in state s_t.

Algorithm 2 Double Q-Learning

Initial network $\leftarrow Q^\theta$, target network $\leftarrow Q^{\theta'}$, replay buffer $\leftarrow \mathcal{D}$, $\tau << 1$ ▷ Input
for each number of iterations **do**
  for each step **do**
    Observe state s_t and choose $a_t \sim \pi(a_t, s_t)$
    Apply a_t, observe s_{t+1} and $r_t = R(a_t, s_t)$
    Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
  end for
  for each update step **do**
    Sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$
    Compute Q-value:

$$Q^*(s_t,a_t) \approx r_t + \gamma Q^\theta\left(s_{t+1}, \text{argmin}_a Q^{\theta'}(s_{t+1},a)\right) \tag{2.43}$$

    Perform gradient descent on $\left[Q^*(s_t,a_t) - Q^\theta(s_t,a_t)\right]^2$
    Update the Q-network parameter:

$$\theta' \leftarrow \tau \times \theta + (1-\tau) \times \theta' \tag{2.44}$$

  end for
end for

In the limit of visiting all (s,a) pairs infinitely often, this update rule converges to the optimal Q-values in the tabular case [102]. In practice, a so-called ϵ-greedy

policy is used to ensure sufficient exploration in a Q-learning setting. Formally, stated as,

$$\pi(a \mid s) = \begin{cases} 1 - \epsilon_t \text{ for } a = \max_{a'} Q(s, a') \\ \epsilon_t \text{ otherwise} \end{cases}$$

The ϵ-greedy policy is only used to introduce randomness to the actions selected by the agent during training, but once training is finished, a deterministic policy follows. We employ neural networks (NN) as function approximators to extend Q-learning to large state and action spaces. NN training typically requires independently and identically distributed data, which isn't naturally available in the sequential RL data. This problem is circumvented by experience replay. This method divides past experiences into single-episode updates, creating batches randomly sampled from memory. To stabilize training, two NNs are employed: a policy network that is continuously updated and a target network that is an earlier copy of the policy network. The policy network estimates the current value, while the target network provides a more stable target value represented by Y:

$$Y_{\text{DQN}} = r_{t+1} + \gamma \max_{a'} Q_{\text{target}}(s_{t+1}, a')$$

In the Double deep Q-network (DDQN) algorithm, the action for the target value is sampled from the policy network to reduce the overestimation bias inherent in standard DQN. The corresponding target is defined as:

$$Y_{\text{DDQN}} = r_{t+1} + \gamma Q_{\text{target}}\left(s_{t+1}, \arg\max_{a'} Q_{\text{policy}}\ (s_{t+1}, a')\right).$$

This target value is approximated using a selected loss function, in this case, a smooth L1-norm loss.

2.3 RL-based quantum architecture search algorithm

Through the discussions in the previous sections, we notice that learning an RL-agent primarily depends on the following ingredients. The environment specifications, a proper description of the RL-state, the formulation of the reward function and the action space encoding. Hence, while constructing an RL-based quantum architecture algorithm to solve variational quantum algorithms, it is crucial to define the environment, the state of the RL, the action, and the reward function in an agent-friendly way.

In the Fig. (2.12), we illustrate an effective yet simple way to define the

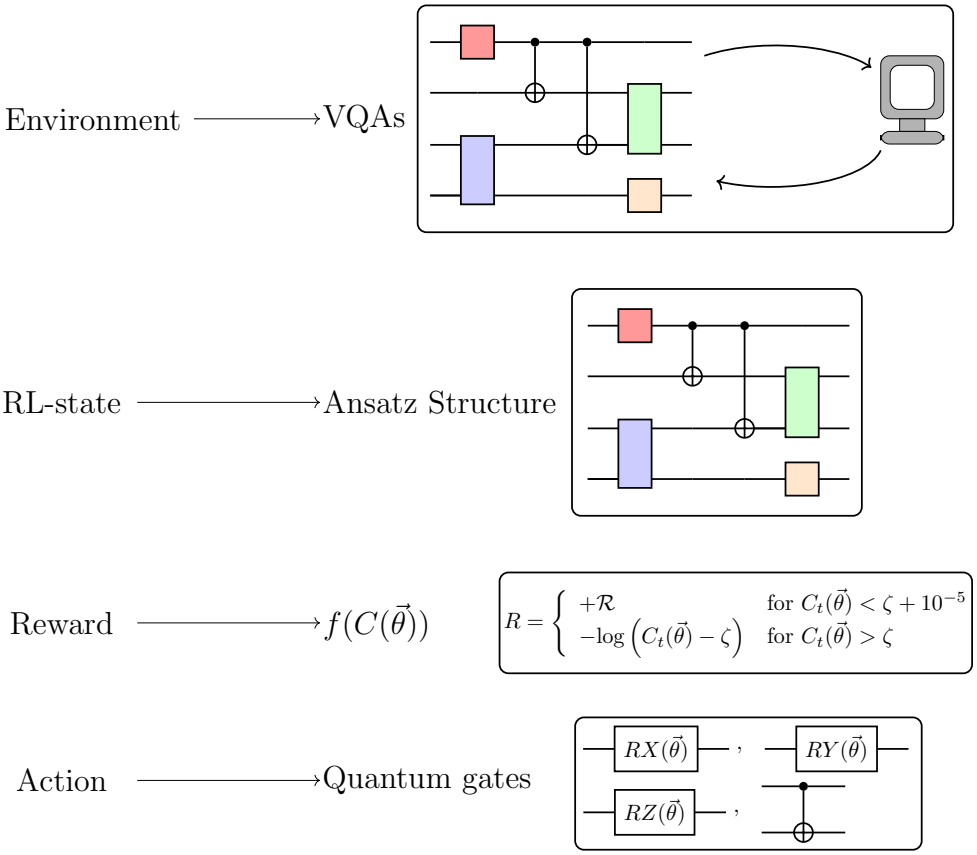


Figure 2.12: The crucial ingredients to cook a successful RL-based QAS algorithm. Here, the environment is defined through the hybrid quantum-classical algorithm. The RL-state is represented through the variational quantum circuit, the reward function is a function of the cost function that encodes the problem, and the actions are defined by one- and two-qubit quantum gates.

ingredients in order to construct an RL-based QAS algorithm. The environment is specified through the quantum-classical algorithm, where the variational quantum circuit, i.e. the ansatz, is considered RL-state. It should be noted that there are several ways to encode the ansatz in an RL-agent readable way [50, 116], but we (which is elaborated in the upcoming chapters) utilize a novel 3D tensor-based binary encoding scheme. Meanwhile, the reward function can be defined in a sparse or dense manner. In the illustration, the reward is dense as the cost function is calculated, and then, based on its value, the reward is defined in each step t of an episode. If the cost function approaches the predefined threshold value ζ, the agent receives a highly positive response ($+\mathcal{R}$). In other scenarios, instead of punishing the agent, we define the cost so that the agent reaches closer to the goal (here, the

predefined threshold ζ), and the reward becomes more positive. This makes the learning curve of the agent smoother and not stricter than a sparse (punishing) reward function. This will be more elaborately discussed in the upcoming section when we present the RL-VQSD algorithm in chapter 3. It should be noted that we also make use of a sparse reward function in the case of the CRL-based VQE algorithm presented in chapter 4 in order to compare its performance with the state-of-the-art RL methods. Finally, after each step t, the agent decides on an action to take from the action space. The action space is represented by a continuous set of parameterized one-qubit rotation gates, and for two-qubit gates throughout the article, we utilize the `CX` gate. The action space is defined using one-hot encoding, which will be elaborated in the upcoming chapter.

Having all the ingredients for constructing an RL-based QAS algorithm, the upcoming chapters focus on specific applications of this design.

Chapter 3

Reinforcement Learning assisted Variational Quantum State Diagonalization: RL-VQSD

In this chapter, we provide an overview of the results concerning the utilization of Reinforcement Learning in the task of Variational Quantum State Diagonalization which we call the RL-VQSD method. We start with a review of the state-of-the-art approach for quantum state diagonalization, followed by an investigation of the efficiency of various classical optimization techniques and ansatz structure. Next, we introduce a novel binary encoding scheme for quantum circuits that improves the *efficient search for a diagonalizing unitary and provides an enhancement in the accuracy for the VQSD scheme*. Moreover, a carefully constructed dense reward function makes the RL-VQSD more efficient in terms of the number of gates and depth of the diagonalizing unitary. In the last part, we demonstrate the example where the proposed techniques lead to a significant improvement of the VQSD algorithm.

3.1 Introduction

One of the most prominent variational quantum algorithms is called the Variational Quantum State Diagonalization (VQSD) [87]. This procedure utilizes a quantum-classical hybrid procedure to identify the unitary rotation under which the given quantum state becomes diagonal in the computational basis, i.e. it diagonalizes a quantum state. This has several applications, including quantum state fidelity estimation [31], device certification [85], Hamiltonian diagonalization [168], and extracting the entanglement properties of a system [87]. VQSD generalizes the well-studied problem of quantum state preparation, which can be understood as

quantum state tomography for pure states. Considering it has applications that range from quantum information to condensed matter physics, an efficient way to deal with quantum state diagonalization may lead to interesting insights in these fields.

It is worth mentioning that classical methods of diagonalization scale polynomially with the size of the matrix [43] and in Tab. 3.1 we list a few well-known classical diagonalization algorithms and their complexity. Meanwhile, quantum principal component analysis [97] (qPCA) proposes an exact algorithmic method for quantum state diagonalization.

Methods	Asymptotic Complexity
QR iteration [164]	$\mathrm{O}(n^2)$
BI iterations [144]	$\mathrm{O}(nm)$
Divide & conquer [58]	$\mathrm{O}(n^3)$
MR method [121]	$\mathrm{O}(n^2)$

Table 3.1: Complexity comparison of the classical methods for diagonalization.

Nevertheless, qPCA often results in complex circuits, which variational approaches could potentially surpass. However, a significant challenge in VQSD lies in devising an *efficient* ansatz capable of diagonalizing a specific quantum state.

Many factors can be used as a pointer for an efficient ansatz ($\mathcal{A}_{\text{eff}}$) but for the sake of the thesis we primarily focus on the following definition to define an efficient ansatz [82]:

Definition 3.1.1: *An ansatz is efficient if the depth and the total number of gates are smaller compared to the state-of-the-art ansatz structures, and which returns a lower error in solving the problem.*

Here, we can formulate the task of finding an efficient ansatz that can be expressed (ideally) in the following way

$$\mathcal{A}_{\text{eff}} = \min_{\mathcal{D},\mathcal{N}_g,\Delta} f(\mathcal{D},\mathcal{N}_g,\Delta), \tag{3.1}$$

where $\mathcal{D}$ is the total depth, $\mathcal{N}_g$ is the total number of gates in the ansatz. Meanwhile, Δ is the error we receive using the ansatz. Based on the fact that the parameters $\mathcal{D}$, $\mathcal{N}_g$, and Δ increases with the increase in number of qubits the overall $\mathcal{A}_{\text{eff}}$ is a monotonically increasing function of these parameters. And the task of finding the efficient ansatz boils down to decreasing the rate of the growth of this function

with the number of qubits. It should be noted that based on the problem, the definition of Δ varies, as we will show in the upcoming sections.

In the standard VQSD methods, a Layered Hardware Efficient Ansatz (LHEA) is utilized as shown in Fig. (2.5).

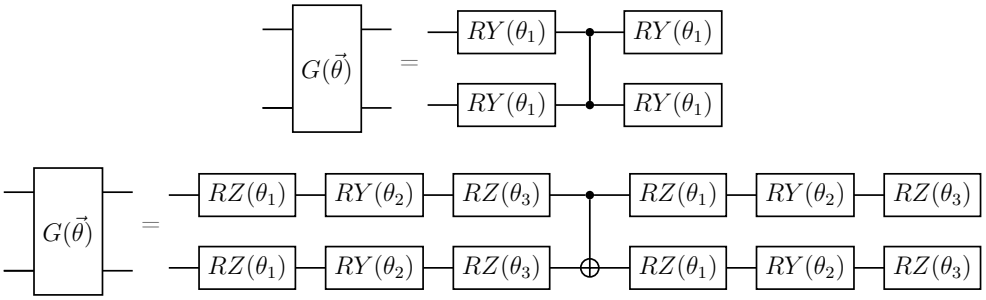


Figure 3.1: Two possible decompositions of the two-qubit rotations in each layer-wise unitary $U_i(\vec{\theta}_i)$ of the layer-ansatz in Fig. (2.5).

A single layer of the ansatz contains two-qubit gates acting on neighboring qubits. These gates are decomposed in two ways illustrated in Fig. 3.1. Although in the LHEA, the parameter count increases linearly with the number of layers and qubits, it has trainability issues and often encounters local minima. To tackle the trainability issue, instead of using a fixed structure of LHEA, the authors allow additional updates (i.e. changes in the ansatz structure) during the classical optimization process. In this process, every optimization step involves the minimization of the cost function with a small random change to the ansatz structure. The new structure is approved or rejected based on a simulated annealing scheme [37]. Although the varying structure LHEA outperforms fixed structure LHEA, the number of gates in the quantum circuit increases rapidly as we scale the size of the quantum state. Hence, the problem of finding a method to construct an ansatz that satisfies all efficiency criteria is still an open problem.

To tackle the problem, we incorporate a Reinforcement Learning (RL) agent to automate the search for an efficient ansatz for VQAs. In the past several novel approaches have been introduced in the light of machine learning techniques to address the challenge of finding a new architecture of ansatz [26, 44, 63, 86, 112, 116, 118, 166, 169]. In a nutshell, the RL-based algorithm, i.e. RL-VQSD we introduce in [82] utilizes a state-of-the-art encoding method for RL-state space, representing the variational circuit, along with a dense reward function to show that our approach can be successfully used for diagonalizing arbitrary mixed quantum states. Moreover, we also demonstrate that compared to LHEA, the ansatz proposed by the RL-agent is more efficient (Def. 3.1 for the definition of efficient ansatz). The methods we discuss are algorithm-independent, so they can be easily adopted

to tackle any VQAs.

In the following, we first briefly discuss the VQSD algorithm and the ways to implement it. Using LHEA we analyze its performance for random quantum states. Next, we give a brief discussion on how we can incorporate Reinforcement Learning (RL) to automate the VQSD process, which throughout the article we call the RL-VQSD method.

3.2 Previous work

In this section, we briefly discuss the variational quantum state diagonalization algorithm and analyze its performance. We will use the described procedure as the starting point for the utilization of Reinforcement Learning to improve efficiency.

3.2.1 The VQSD algorithm

The variational quantum state diagonalization algorithms, introduced in [87], aim to identify the unitary rotation under which the given quantum state becomes diagonal on the computational basis. Hence, for a given state ρ, VQSD composed of the three following subroutines (see Fig. (3.2))

- Training In this subroutine, for a given state ρ, one optimizes the parameters $\vec{\theta}$ of a quantum gate sequence $U(\vec{\theta})$, which (ideally) after optimization satisfies

$$\tilde{\rho} = U(\vec{\theta}_{\text{opt}})\rho U(\vec{\theta}_{\text{opt}})^{\dagger} = \rho_{\text{diag}}, \tag{3.2}$$

  where ρ_{diag} is the diagonalized ρ in its eigenbasis and $\vec{\theta}_{\text{opt}}$ are the optimal angles. One can utilize classical gradient-based methods such as SPSA [142] and Gradient-Descent [130], or gradient-free optimization methods such as COBYLA and POWELL [124] in the training process.

- Eigenvalue Readout In this subroutine, using the optimized unitary $U(\vec{\theta}_{\text{opt}})$ and one copy of state ρ, one can extract – for low-rank states – all the eigenvalues or – for full-rank state – the largest eigenvalues. This is achieved by measuring the $\tilde{\rho}$ in the computational basis, $\mathbf{b} = b_1 b_2 \dots b_n$, as follows

$$\tilde{\lambda} = \langle \mathbf{b}|\tilde{\rho}|\mathbf{b}\rangle, \tag{3.3}$$

  where $\tilde{\lambda}$ are inferred eigenvalues.

- Eigenvector Preparation In the final step one can prepare the eigenvectors associated with the largest eigenvalues. If $\tilde{\mathbf{b}}$ is a bit string associated

with $\tilde{\lambda}$ then one can get the inferred eigenvectors $|\tilde{v}_{\tilde{\mathbf{b}}}\rangle$ as follows

$$|\tilde{v}_{\tilde{\mathbf{b}}}\rangle = U(\theta_{\text{opt}})^{\dagger}|\tilde{\mathbf{b}}\rangle = U(\theta_{\text{opt}})^{\dagger}\left(X^{b_1}\otimes\ldots\otimes X^{b_n}\right)|\mathbf{0}\rangle. \tag{3.4}$$

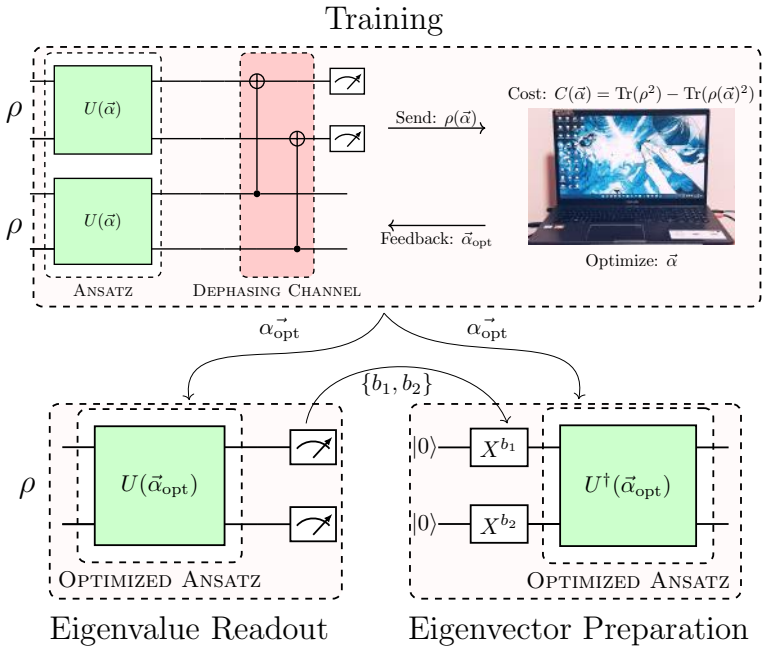


Figure 3.2: Elements of Variational Quantum State Diagonalization (VQSD) algorithm. In the presented example, we consider the diagonalization for a two-qubit input state.

3.2.2 The cost function

The VQSD algorithm focuses on finding a unitary that diagonalizes a quantum state; hence, the cost function should encode the information about how far a state ρ is from being diagonal. Meanwhile, we need to ensure that the cost function choice follows the three golden rules of cost function construction that are briefly discussed in Sec. 2.1.1. One such cost-efficiently computable cost function is

$$C(\vec{\theta}) = \text{Tr}(\rho^2) - \text{Tr}(\mathcal{D}(\tilde{\rho})^2), \tag{3.5}$$

where $\mathcal{D}$ denotes the dephasing channel to minimize the off-diagonal terms in $\tilde{\rho}$. The cost function Eq. (3.5) is upper bounded by the error in the eigenvalue of ρ.

Where we define the eigenvalue error by

$$\Delta_i = \sum_{i=1}^{m} \left(\lambda_i - \tilde{\lambda}_i\right)^2, \tag{3.6}$$

where m represents the number of the largest eigenvalues, λ_i is the true eigenvalue and $\tilde{\lambda}_i$ is the inferred eigenvalue obtained from the EIGENVALUE READOUT subroutine. In the ideal case, where the state is completely diagonalized, $m = 2^n$ indicates all the eigenvalues have been considered.

It should be noted that in the cost function Eq. (3.5), the 1st term on the RHS can be computed outside the optimization loop, and we left out with the main task of evaluating the 2nd term. Also, the cost function landscape for the cost function turns insensitive to the changes in the ansatz when we scale up the size of the quantum state, but as we constrain ourselves to $n \leq 6$ qubits, we can efficiently utilize the cost Eq. (3.5).

In the following, we first analyze the performance of different ansatz and optimizers to diagonalize a two-qubit mixed quantum state.

3.2.3 Benchmarking the performance of LHEA

In the following, we rigorously investigate the state-of-the-art VQSD method for two- and three-qubit uniformly distributed random quantum states that are generated by using the `random_density_matrix` generator of `qiskit.quantum_info` module[1] of `qiskit` [4]. We do the investigation for two-qubit random mixed quantum states and focus on the performance of LHEA. Further, we benchmark classical optimization methods in the task of diagonalizing a two-qubit mixed quantum state.

Performance of LHEA In Fig. (3.3), we explore the possible constructions of HEA and briefly discuss their performance. For name convention, we call the ansatz in LHS of the first row as `RYCZ` and the RHS one as `RZRXCX` ansatz where we call the ansatz in the 2nd row as `RYRZRYCX` ansatz. It can be seen that for the `RYCZ` ansatz, the convergence with the number of parameters saturates at 10^{-1} for all kinds of optimizers, whereas the other two ansatz shows good performance in minimizing the cost.

The convergence of the `RYRZRYCX` and `RZRXCX` ansatz shows comparable performance. Both ansatzes have their advantages and disadvantages which are discussed in the following points.

[1] https://qiskit.org/documentation/stubs/qiskit.quantum_info.random_density_matrix.html

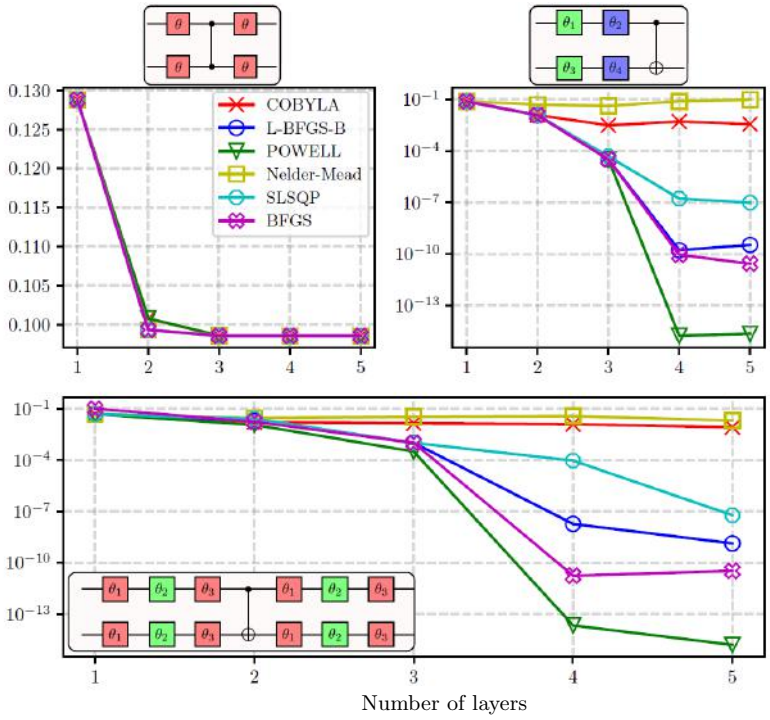


Figure 3.3: Three kinds of LHEA in diagonalizing a two-qubit quantum state using VQSD. We can see from the results that the `RYCZ` structure performs worse than the other two structures in terms of finding the optimal cost function for the task. In the pictures ■=`RY`, ■=`RZ` and ■=`RX` gate.

- In the case of the `RZRXCX` ansatz, the number of parametrized gates is very small and relatively susceptible to noise. On the other hand, due to a higher number of different parameters, i.e. $\theta_1, \theta_2, \theta_3$ and θ_4 for larger quantum states, it fails to optimize the cost function.

- In the case of the `RYRZRYCX` ansatz, the number of different parameters in gates is very small, making it more suitable for optimization. On the other hand, due to the increase in number of gates, it is not susceptible to noise.

In Fig. (3.4) we further investigate the performance of `RYRZRYCX` and `RZRXCX` ansatz for a three quantum mixed quantum state, and we can clearly see that `RYRZRYCX` outperforms the other so we use the `RYRZRYCX` construction of LHEA for VQSD throughout the paper, if not stated otherwise.

Benchmarking optimizers In Fig. (3.5) `RYRZRYCX` construction of LHEA to diagonalize 100 Haar-random mixed quantum states is presented.

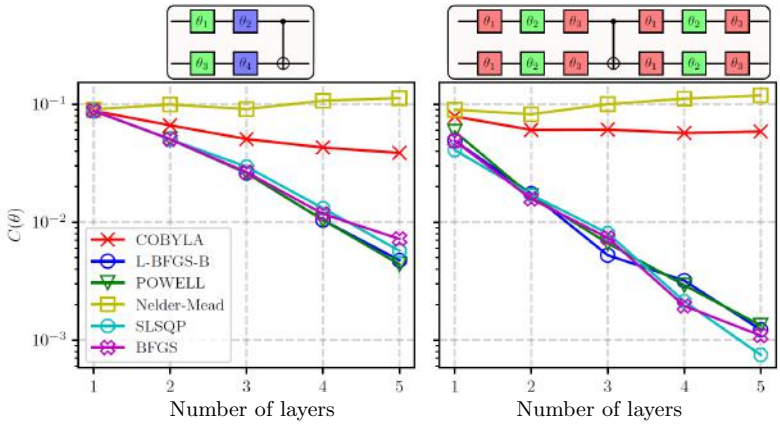


Figure 3.4: Two distinct structures of LHEA. In Fig. (3.3), we saw that the `RZRXCX` and the `RYRZRYCX` ansatz performs equivalently for two-qubits in optimizing the cost. By considering these two structures of ansatz, we investigate the performance of these structures in cost function minimization. The figure shows that `RYRZRYCX` structure outperforms in terms of accuracy compared to `RZRXCX`.

From the left-hand side of Fig. (3.5) we can see that we need at least 3 layers to get a good approximation to the diagonal state of an arbitrary two-qubit mixed state. This indicates we need at least 4 `CX` and $12 \times 4 = 48$ parameterized gates with a total depth of $7 \times 4 = 28$ to get a good estimate on the diagonalizing unitary. Among all the optimizers, the POWELL optimizer gives us the best outcome; meanwhile, on the right-hand side of Fig. (3.5) we notice that the T_{opt}, i.e. the time (in minutes) it takes to finish a complete round of optimization[2] grows rapidly as we increase the number of layers. On the other, COBYLA finishes the complete optimization process in just 0.1 minutes. Still, the cost function does not go below 10^{-3} and hence fails to give us a good approximation to the diagonal state.

In the following, we primarily use COBYLA in the classical optimization subroutine for reinforcement learning-assisted VQSD, i.e., the RL-VQSD method. The main motivation behind this decision is the following. While utilizing a learning-based method like RL *we start from an empty quantum circuit, and using a state-of-the-art encoding method, we encode the circuit into an RL-state*. After each application of a gate to the quantum circuit, the RL-state updates, and the circuit passes through a classical optimization method. Using the outcome of the optimization, we evaluate a reward function. *Based on the reward, the RL-agent decides on an action that encodes the information of a particular quantum gate*. This process repeats until a predefined threshold in cost estimation is reached, and

[2]Each round of optimization contains 10 runs of the optimizer with uniform random initialization and we choose the best parameters among the 10 runs.

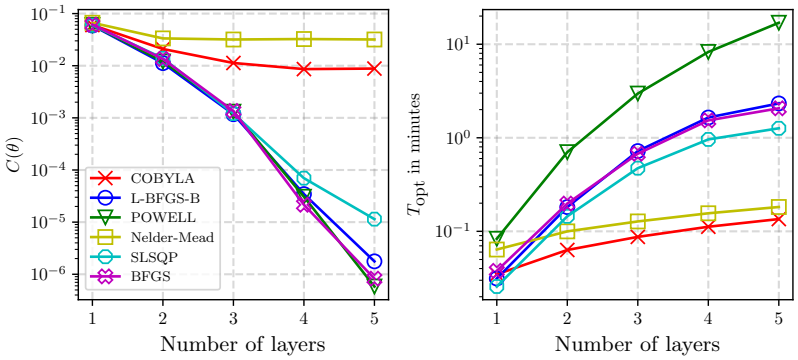


Figure 3.5: Different optimizers' performance in minimizing the cost function in diagonalizing 100 random two-qubit mixed states. It can be seen that among the gradient-free optimizers, `POWELL` performs optimally, whereas the time it takes to provide the optimized result is higher than other optimizers. In the case of the gradient-based optimizer, the `L-BFGS-B` and the `BFGS` outperform other similar optimizers.

the total number of episodes is exhausted. After the application of each action, we harness the power of classical optimization to make sure that the optimization process does not consume most of the resources in the RL-VQSD.

3.3 Components of RL-VQSD algorithm

Before jumping into RL-VQSD, in this section, we provide a detailed description of the building blocks used in the proposed algorithm. As in any RL method, the proposed methods included the specification of the state, the actions, and the reward function.

3.3.1 RL-State

Learning-based quantum architecture search algorithms require a concise circuit representation that is commonly referred to as encoding. Through the encoding scheme, we can modify, compare, and explore the quantum circuit. This helps us to navigate the search space of all possible alternatives efficiently and discover effective and innovative quantum circuits. In the following, we will elaborate on a tensor-based binary encoding approach for the quantum circuit, which can efficiently be utilized as an RL-state. The encoding was first introduced in [119]. In this scheme, the gate structure of the ansatz is expressed as a tensor of dimension $[T \times ((N+3) \times N)]$, where N represents the size of the problem and T is the

considered maximum depth of the ansatz. For VQSD, N represents the number of qubits in the quantum state that need to be diagonalized. The proposed encoding can be explained through the following two points:

1. **Freedom in connectivity** The encoding enables *all-to-all* qubit connectivity, but it can be restricted by considering *unidirectional nearest neighbour* connections only. In this scenario, the matrix dimension $((N+3) \times N)$ is reduced to $(4 \times N)$. One should note that in the case of a two-qubit gate, one is not required to keep track of the control and target simultaneously. Hence, defining one argument of the two-qubit gate implicitly provides information about the other argument due to its nearest neighbor and unidirectional nature. A similar encoding scheme is described in [50].

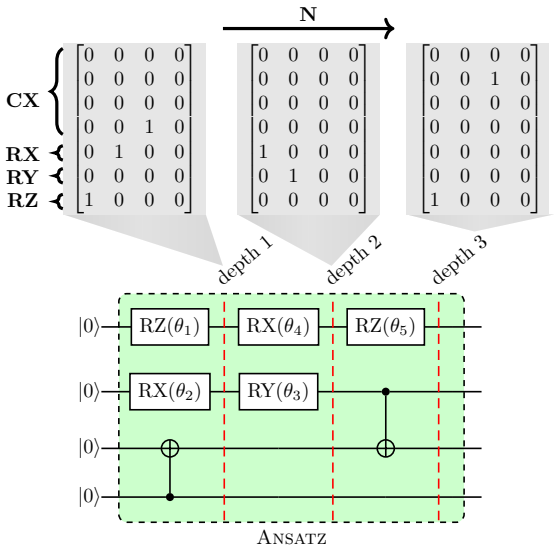


Figure 3.6: Example of the proposed encoding for 4-qubit ansatz. The first $(N \times N)$ square matrix is reserved for the `CX` connectivity. The columns of the square matrix encode the *target qubit*, and the rows represent *control qubits*. The remaining $((N+j) \times N)$ elements encode arbitrary rotation towards j direction where $j = 1$, 2, and 3, for `X`, `Y` and `Z` rotations, respectively.

2. **Depth-based encoding** In previous work [116] each $((N+3) \times N)$ matrix carries information corresponding to each action taken by the agent, where each action represents either a single or a two-qubit gate. Additionally, the information was integer-based, in the range 0 to N.

   On the contrary, In our work, the encoding is binary and depth-based. For example, if $T = 3$, then the encoding initiates by filling up the $[i \times ((N+3) \times N)]$

for $i = 1$ until a depth of RL-ansatz is encoded. Which is described as follows:

$$\left[\underbrace{((N+3)\times N)}_{\text{depth}=1}, \underbrace{((N+3)\times N)}_{\text{all zeros}}, \underbrace{((N+3)\times N)}_{\text{all zeros}}\right]. \tag{3.7}$$

Then, as $i = 1$ is filled up, we move to $i = 2$ to encode depth $= 2$ of the RL-ansatz, which yields

$$\left[\underbrace{((N+3)\times N)}_{\text{depth}=1}, \underbrace{((N+3)\times N)}_{\text{depth}=2}, \underbrace{((N+3)\times N)}_{\text{all zeros}}\right]. \tag{3.8}$$

Finally, the depth $= 3$ is encoded in $i = 3$ resulting in

$$\left[\underbrace{((N+3)\times N)}_{\text{depth}=1}, \underbrace{((N+3)\times N)}_{\text{depth}=2}, \underbrace{((N+3)\times N)}_{\text{depth}=3}\right]. \tag{3.9}$$

Each depth encoding follows the scheme shown in 3.6. In the following, we give a simple example of $T = 3$ encoding. In the Appendix D.1, we provide a code that is used for the RL-state encoding.

3.3.2 RL-action

For constructing the quantum circuits, we use the scheme developed in [116] with `CX` and one-qubit rotation gates, which are feasible on currently available quantum devices. The encoding of the action space can be defined as follows. The `CX` gates are represented by a pair of values that indicate the positions of the control and target qubits, with enumeration starting from 0. As for the rotation gates, they are encoded using two integers, also starting from 0. The first integer identifies the qubit register, while the second integer specifies the rotation axis. For an N-size quantum state, the agent can choose from $3 \times N$ single-qubit gates and from $2 \times \binom{N}{2}$ two-qubit gates. Additionally, we utilize a one-hot encoding for the action. We provide a simple example to make it more clear.

Each action is represented by a list of 4 numbers where

$$\mathcal{A}_t = \begin{cases} [N, N, e_3, e_4]_t, & \text{if gate} = \texttt{ROT} \\ [e_1, e_2, N, N]_t, & \text{if gate} = \texttt{CX} \end{cases}$$

where $t \in T$ denotes the time step and $e_i \in \{0, 1, \ldots, N-1\}$. In the case of one qubit parameterized gates defined through `ROT` encoding, say $[N, N, e_3, e_4]$ the *qubit*

position is defined by e_3 and the *rotation axis* is encoded through by e_4. Meanwhile for two-qubit `CX` gates the encoding $[e_1, e_2, N, N]$ represents that the *control* is on e_1 qubit whereas the target is on $(e_1 + e_2) \pmod N$ qubit.

An example Here, we give a simple example of the encoding and the action scheme for $T = 3$. In the Fig. (3.7), we take a three-qubit system. The empty circuit is denoted by a tensor of size $3 \times (6 \times 3)$ tensor. As we can see from the

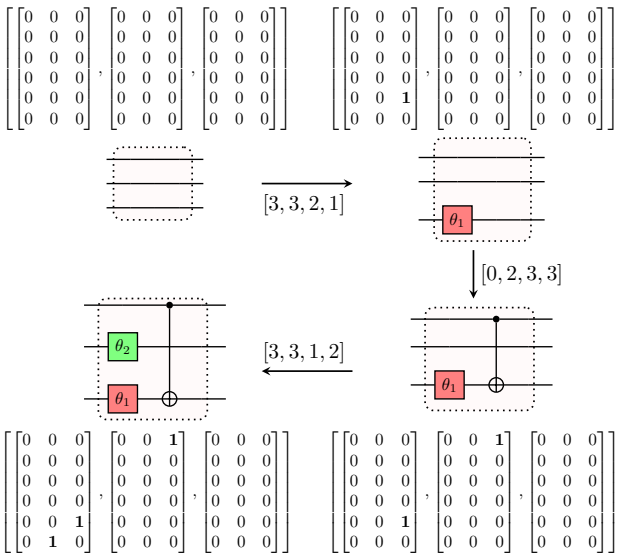


Figure 3.7: A toy example of action and the encoding scheme. The actions are represented with a list of four numbers. The first two elements in the digit carry the information about the controls and the target of a two-qubit gate that the RL-agent suggests adding in the next step to the ansatz. Meanwhile, the last two digits tell us about a one-qubit gate, on which qubit the gate should be added, and in which direction the rotation is.

example, the first (6×3) matrix is filled when the `RY` gate is applied on the 3rd qubit. But the information is encoded in the next empty tensor for the next gate, which is `CX` with a control on the first and a target on 3rd qubit. This is because each (6×3) matrix encodes the complete information corresponding to a depth. That is why when the next `RZ` gate is applied on the second qubit, the information corresponding to it was encoded in the previous (6×3) matrix because it does not increase the depth of the circuit.

The encoding presented in the paper is an RL-agent-friendly representation of the action space. Many such encoding schemes can be adopted as a representation of the action space. Meanwhile, as we constrain the action space to only one-qubit

rotations and `CX` gate, it would also be interesting to investigate the inclusion of other gate-sets in the action space.

3.3.3 RL-reward

To guide the agent quickly towards the goal, we introduce a reward that is dense in time at each time step t. The reward used in this work is given as

$$R = \begin{cases} +\mathcal{R} & \text{for } C_t(\vec{\theta}) < \zeta + 10^{-5}, \\ -\log\Big(C_t(\vec{\theta}) - \zeta\Big) & \text{for } C_t(\vec{\theta}) > \zeta. \end{cases} \tag{3.10}$$

where the goal of the agent is to reach the minimum error for a predefined threshold ζ, *ie.*the tolerance for cost function minimization. The ζ is a hyperparameter of the model. The cost function at each step t is calculated for the ansatz which outputs a state $\rho_t(\vec{\theta})$ as

$$C_t(\vec{\theta}) = \mathrm{Tr}(\rho^2) - \mathrm{Tr}(\rho_t(\vec{\theta})^2). \tag{3.11}$$

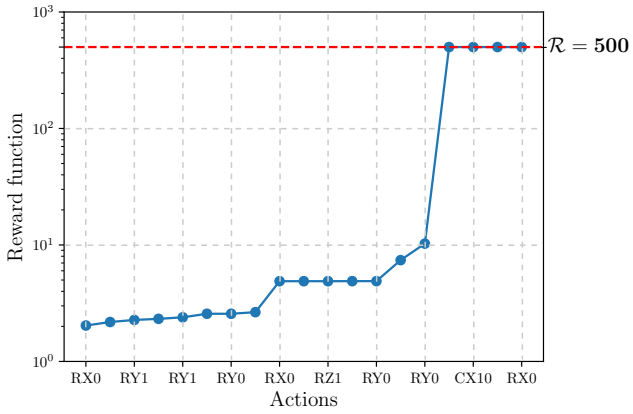


Figure 3.8: The variation of reward function with actions taken by the RL-agent. In this illustration, we use RL-VQSD (which is discussed in the upcoming section) to diagonalize a two-qubit random quantum state. The agent receives a high reward of value $\mathcal{R} = 500$ if the RL-ansatz passes a predefined threshold (which is $\zeta = 10^{-5}$) in any step of an episode. The total number of steps considered for this task is 20. The labels in the x-axis represent each action in terms of a gate where for one qubit gate, the label is denoted as `Gj` where `G` is the rotation gate on `j`-th qubit. For the two-qubit gate, we use the notation `CXij` where `CX` is the controlled-NOT gate with control on `i`-th and the target on the `j`-th qubit.

Through the illustration in Fig. (3.8), we show how, for a successful episode,

the reward function defined in Eq. (3.10) changes after the application of each action. The illustration shows the learning curve of the RL-ansatz where after taking 15 actions, the cumulative reward reaches the maximum (i.e. $\mathcal{R} = 500$), which means the RL-agent at this point is able to pass the predefined threshold (which $\zeta = 10^{-5}$).

3.3.4 RL-VQSD

In this section, we utilize the previously described approaches, such as ansatz encoding, action formulation, and various approaches to reinforcement learning (see Sec. 2.2.3 for details

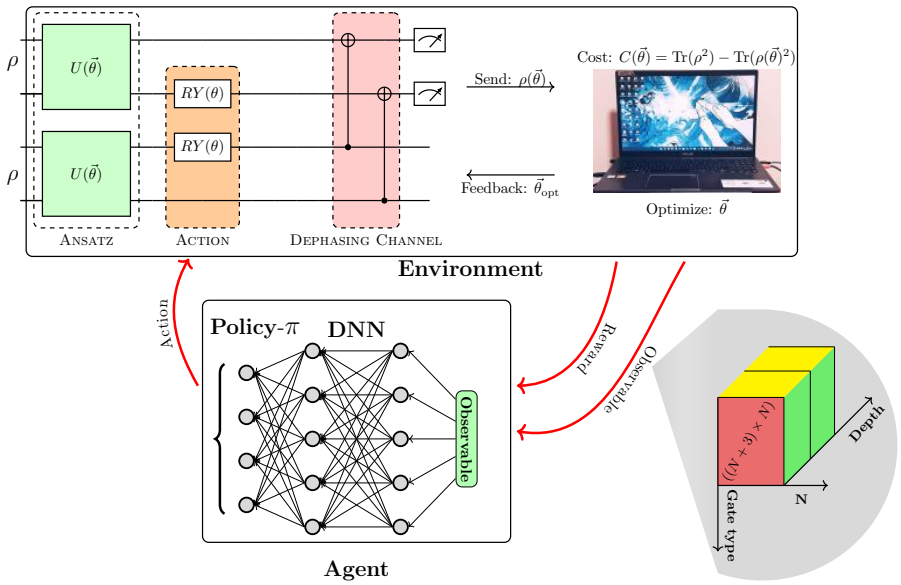


Figure 3.9: Demonstration of the RL-VQSD procedure: In this approach, the VQA task is linked to an environment where ansatz serves as the RL-state in Reinforcement Learning (RL). The RL-agent receives a reward in the form of an optimized cost function from the environment, along with the current RL-state. Employing an ϵ-greedy policy, the agent selects an action (a quantum gate), which updates the RL-state for the subsequent step. Using this updated RL-state, the VQA optimizes the cost function, generating a new reward for the agent. This iterative process continues until all episode steps are completed or the cost function reaches a predefined threshold. Throughout our study, we initiate RL-VQSD with an empty circuit, and at each step, the agent's action constructs the RL-ansatz, symbolized by $U(\vec{\alpha}) = \mathbb{I}$.

on the various approaches) to enhance the performance of the VQSD algo-

rithm. We primarily focus on the number of gates, the depth, and the accuracy of diagonalization of an ansatz. In Fig. (3.9), we illustrate the RL-VQSD algorithm.

To achieve the results, we use a Double Deep-Q network [105] (DDQN) for better stability with an ϵ-greedy policy and the ADAM optimizer [78]. We start with the parameter specifications given in [116], which uses the n-step DDQN algorithm with a discount factor of $\gamma = 0.88$ and an ϵ-greedy policy for selecting random actions. The value of ϵ is gradually decreased from 1 to a minimum value of 0.05 by a factor of 0.99995 at each step. The size of the memory replay buffer is set to 2×10^4, and the target network in the DDQN training is updated with every 500 action. Following each training episode, we conduct a testing phase where the probability of selecting a random action is set to 0, and the experience replay procedure is turned off. Experiences obtained during the testing phase are not added to the memory replay buffer.

To obtain a reward $\mathcal{R}$ for the circuit (*ie.*for each environmental state), an optimization subroutine needs to be applied to determine the values of the rotation gate angles. We use well-developed methods for continuous optimization, such as Constrained Optimization By Linear Approximation [124] (COBYLA), which has been shown to be among the best performing when there is no noise in the system. In this chapter, we set $\mathcal{R} = 500$.

3.4 Diagonalizing quantum state with RL-VQSD

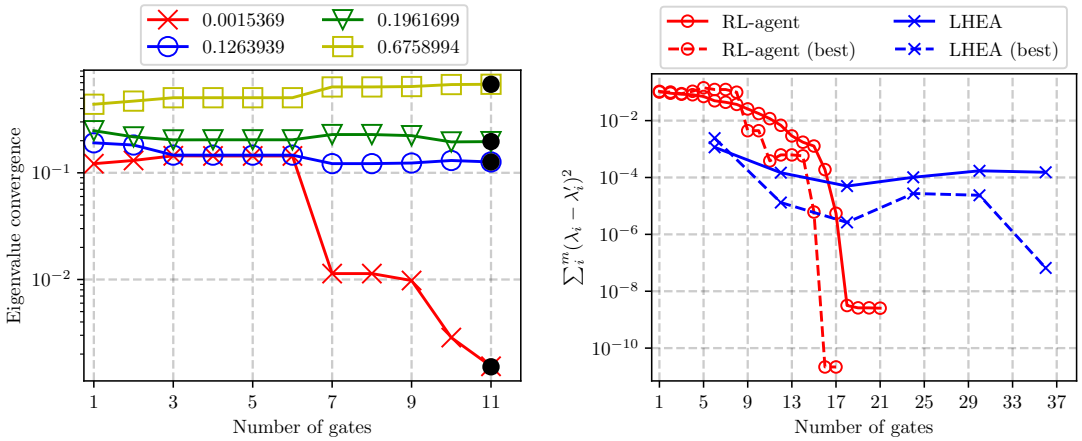
In this section, we briefly investigate the performance of RL-VQSD on diagonalizing two-, three- and four-qubit quantum states. We start with diagonalizing two-qubit randomly sampled quantum states and show that the RL-VQSD outperforms the state-of-the-art VQSD method using the `RYRZRYCX` construction of LHEA. Due to the fast optimization time throughout the chapter, we utilize `COBYLA` optimizer with 400, 500, and 1000 iterations for two-, three- and four-qubit states. In the case of diagonalizing three- and four-qubit states, we consider the reduced ground state of the six- and right-qubit Heisenberg model and show that the RL-agent proposed ansatz (so-called the RL-ansatz) provides us with a smaller quantum circuit with lesser gates, depth and better accuracy compared to state-of-the-art ansatz structures.

3.4.1 Two-qubit states

In this section, we aim to utilize a quantum computer to diagonalize (1) a fixed mixed quantum state and (2) 50 random quantum states to get the average eigenvalue approximation error and count the gates in RL-ansatz. We utilized the `random_density_matrix` of the module `quantum_info` of `qiskit` [4] to obtain

the random density matrices. The states are sampled from the Haar measure. By (1), we argue that RL-VQSD can exactly diagonalize a quantum state. The results of (2) demonstrate that the average performance of RL-VQSD is better than state-of-the-art ansatz.

It can be seen from 3.10a we show that the agent is able to propose an ansatz that provides us with the exact eigenvalues for a two-qubit random quantum state with 12 quantum logic gates containing 10 rotations and 2 `CX` gates. In 3.11, we illustrate the quantum circuit proposed by the agent.



(a) (b)

Figure 3.10: Diagonalization of a two-qubit random density matrix of full rank. In (a), we illustrate the convergence of eigenvalues of a mixed quantum state. Meanwhile, in (b), we compare the performance of the RL-agent proposed ansatz with the LHEA. It can be seen that the RL-agent ansatz gives us a better approximation of the eigenvalues. Additionally, the RL-based methods can achieve the accuracy of the LHEA using the circuit with significantly reduced depth of the resulting circuit.

In Fig. (3.10b), we benchmark the performance of RL-ansatz against LHEA. In the illustration, we show that the agent not only gives us a small ansatz to diagonalize with a specific predefined threshold ζ, but it also helps us achieve a very low error in eigenvalue estimation compared to LHEA.

Furthermore, we explore the possibility of utilizing the ansatz proposed by the RL agent, learning on a fixed quantum state, for the diagonalization of other states. We can confirm that this is indeed possible in the case of the two-qubit state. The corresponding results are presented in Fig. (3.12). One can argue that, in this case, the diagonalization task is relatively easy. However, one should note that the

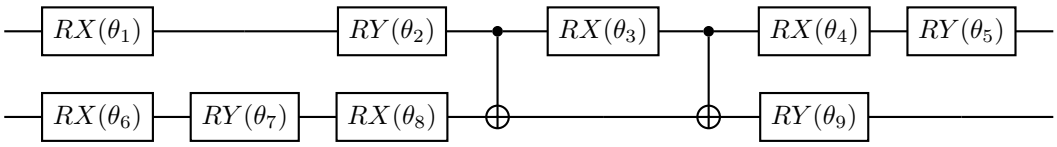


Figure 3.11: The ansatz proposed by RL-agent the state with eigenvalues convergence illustrated in Fig. (3.10a).

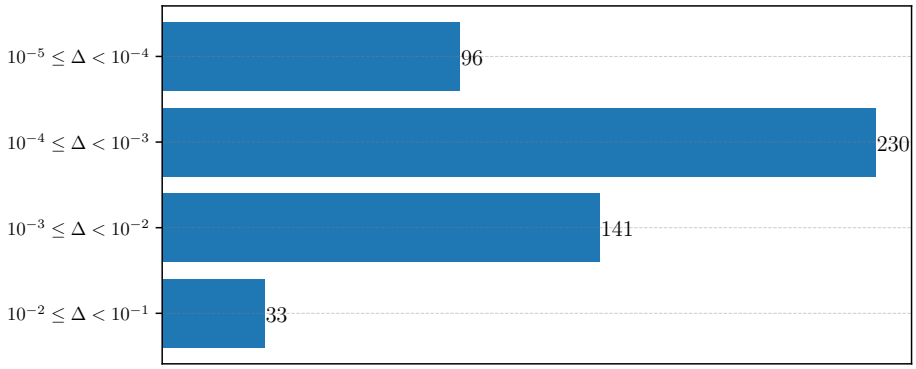


Figure 3.12: Statistics of error in eigenvalue estimation for 500 arbitrary quantum states. As an ansatz to diagonalize all the quantum states, we consider the fixed RL-ansatz in Fig. (3.11).

ansatz proposed by the RL agent gives an average error in eigenvalue estimation as in the case of using a standard approach based on LHEA (cf. Fig. (3.10b)) while enabling the utilization of a shorter quantum circuit, and reducing the potential influence of errors.

3.4.2 Three-qubit reduced Heisenberg model

One of the important applications of VQSD is to study the entanglement in condensed matter systems [91]. Hence, in this experiment, to get a better understanding of the efficacy of our method in this regard, we consider a three-qubit reduced state of the ground state ($|\psi_{S_1,S_2}\rangle$) of the one-dimensional Heisenberg model defined on six qubits which have the following form

$$H = \sum_{j=1}^{2n} \vec{S}^{(j)} \cdot \vec{S}^{(j+1)}, \tag{3.12}$$

where $\vec{S}^{(j)} = \frac{1}{\sqrt{3}}\left(X^{(j)}\hat{x} + Y^{(j)}\hat{y} + Z^{(j)}\hat{z}\right)$ with periodic boundary condition $\vec{S}^{(2n+1)} = \vec{S}^{(1)}$, where X, Y, and Z are the Pauli operators. To perform entanglement spectroscopy on the ground state of the 6-spin Heisenberg model (*ie.*$2n = 6$), we diagonalize the reduced state $\rho_{\text{red}} = \text{Tr}_{S_2}\left[|\psi_{S_1,S_2}\rangle\langle\psi_{S_1,S_2}|\right]$. We consider choosing the threshold $\zeta = 10^{-4}$ for 500 iterations of the global COBYLA method.

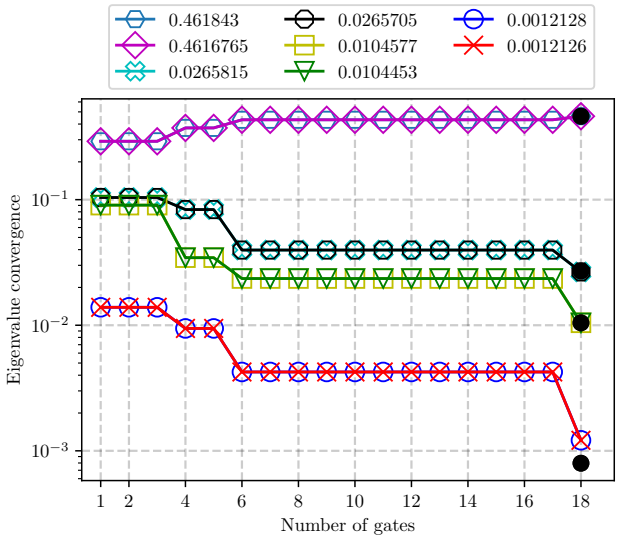


Figure 3.13: Convergence of the eigenvalues of three-qubit, reduced Heisenberg model. The labels on the top of the figure correspond to the eigenvalues. The black dots represent the true eigenvalues. Due to degeneracy in energy level, some dots are overlapped.

The results presented in Fig. (3.13) confirm that the RL-agent can learn to construct an ansatz that can find all the eigenvalues with good accuracy with a very small number of gates and depth. The ● represents the true eigenvalues. We can see that the ansatz takes 18 gates to give us 6 out of 8 exact eigenvalues of a three-qubit Heisenberg model. Additionally, the RL-ansatz finds the remaining two smallest eigenvalues with $\Delta_7 = \Delta_8 = 1.73 \times 10^{-7}$ accuracy. In Fig. (3.14) we present the RL-ansatz that contains 10 rotations and 8 CX gates, proposed by our methods.

It should be noted from circuits in Fig. (3.11) and in Fig. (3.14) that the rotation in the Z direction, *ie.*`RZ` quantum logic gate, does not play a crucial part in the diagonalizing unitary. Thus, one might attempt to diagonalize a random quantum state of two and three qubits, excluding `RZ` rotation from the list of quantum gates. This gives us a hint concerning the action space that could be significantly reduced in these examples.

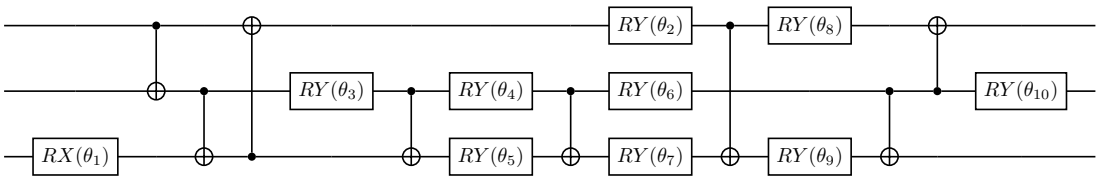


Figure 3.14: The ansatz proposed by the RL-agent for diagonalizing a state in the 3three-qubit reduced Heisenberg model. The circuit contains 10 rotations and 8 CX gates.

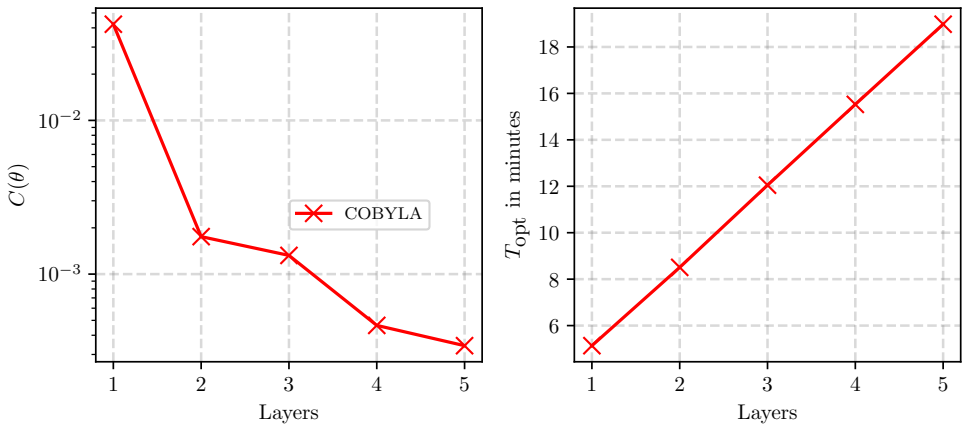


Figure 3.15: Performance of LHEA with the number of layers on (see LHS) and on the RHS, we show the time it takes to complete layers.

Performance of LHEA Here we compare the performance of the LHEA as shown in Fig. (3.15) with the RL-ansatz 3.14 for diagonalizing a three-qubit reduced Heisenberg model. In the case of LHEA, each layer is composed of all-to-all connected two-qubit rotations containing a sandwiched `CX` gate in between `RZRYRZ` rotations of the form presented in 3.1(with 3 rotation gates). Hence, each layer contains 12 parameters. To achieve a cost function in the order of 10^{-4}, it takes more than 60 parameters and at least 5 two-qubit gates. At the same time, the RL-ansatz achieves an accuracy lower than 10^{-4} in just 10 rotations and 8 two-qubit gates. We get a 6 times improvement compared to LHEA in terms of parameters.

RL-ansatz scaling with accuracy Here, we briefly investigate the scaling of the minimum number of gates and the depth of the ansatz to diagonalize the three-qubit ground state of the reduced three-qubit Heisenberg model using the RL-VQSD. The results are illustrated in the Fig. (3.16) where we can see that the number of gates and the depth both scale linearly with the increase in ζ, the

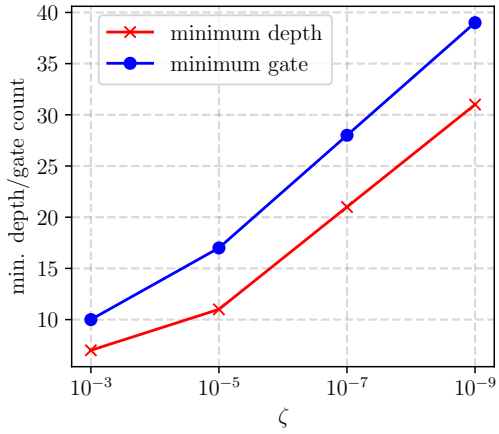


Figure 3.16: In this illustration, we show the minimum number of gates and depth required to diagonalize the ground state of the reduced 6-spin Heisenberg model using an RL-VQSD.

pre-defined tolerance provided to the RL-agent.

ζ	min. ROT	min. CX	ave. depth	ave. num gate
10^{-3}	5	4	16.63	23.084
10^{-5}	8	5	19.67	27.863
10^{-7}	14	10	31.63	41.13
10^{-9}	18	15	36.8	46.15

Table 3.2: In this table, we summarize the scaling of various important RL-ansatz components averaged over 3000 episodes of RL-VQSD.

3.4.3 Four-qubit reduced Heisenberg model

We extend the results of the previous section for the ground state of the 8-spin Heisenberg model (*ie.* $2n = 8$). We diagonalize the four-qubit reduced state of the ground state of the 8-spin Heisenberg model.

It takes 53 gates to find the first 6 largest eigenvalues, with an error below 10^{-5}. Out of 53 gates, 16 are `CX` gates, and the remaining are one-qubit rotations. We consider choosing the threshold $\zeta = 10^{-3}$ for 1000 iterations of the global COBYLA method.

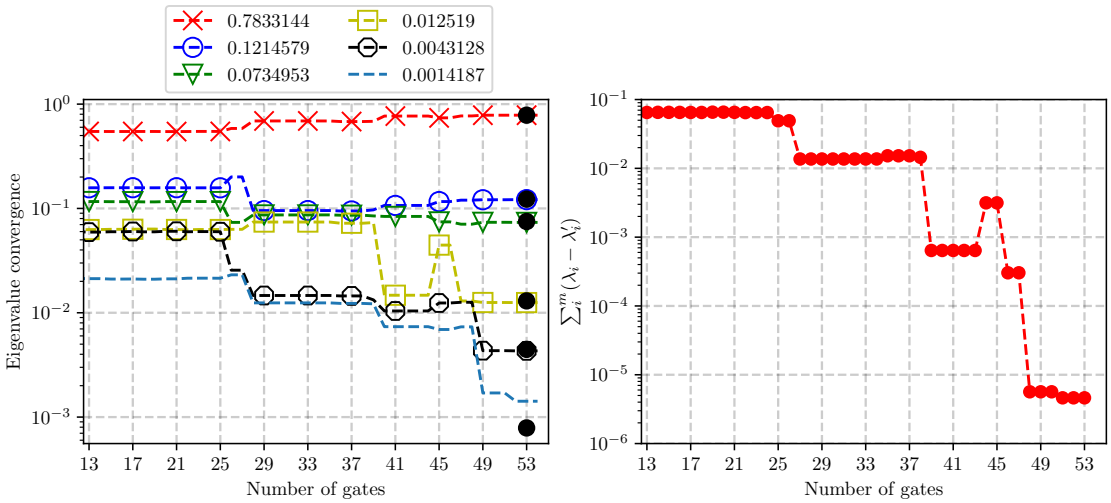


Figure 3.17: The convergence of individual (left panel) and the overall error (right panel) in eigenvalues for 4-qubit reduced Heisenberg model. This provides a significant improvement in terms of gate count and depth compared to the result reported in [87].

Qubits	Minimum depth	Minimum # of rotations	Minimum # of CX
2	8	9	2
3	12	10	8
4	33	28	16

Table 3.3: A summary of the minimum number of one- and two-qubit gates required in RL-ansatz to diagonalize two-, three- and four-qubit systems.

The summary of our results is provided in Table 3.3. One can notice that there is a relation between the number of `CX`s and the dimension of the state that we want to diagonalize. The number of `CX`s grows exponentially with the number of qubits. As for the two-qubit case, we find all the eigenvalues with 10^{-10} error with just two `CX`s. Whereas for three qubits, we are able to find the first 6 eigenvalues with an error below 10^{-8} but the smallest two eigenvalues we find with 1.73×10^{-7} error with 8 `CX`s. Finally, for four-qubit, we find the first 6 eigenvalues with an error below 10^{-8} and the remaining eigenvalues with an error in the range $10^{-4} \leq \Delta \leq 10^{-6}$ with 16 `CX`s. This observation suggests that for a full-rank quantum state of $N \geq 3$, we require at least as many `CX`s as the rank of the quantum state to get a good approximation of the largest eigenvalues. It should be noted that to find the first 5 largest eigenvalues with error 10^{-5}, the ansatz proposed by the RL-agent is of

depth 18 and a total of 30 gates, among which 12 are `CX` gates and the remaining are rotations. This significantly improves the depth, and the gate count in the diagonalizing ansatz compared to the results in [32] and [87].

3.4.4 Performance of random search

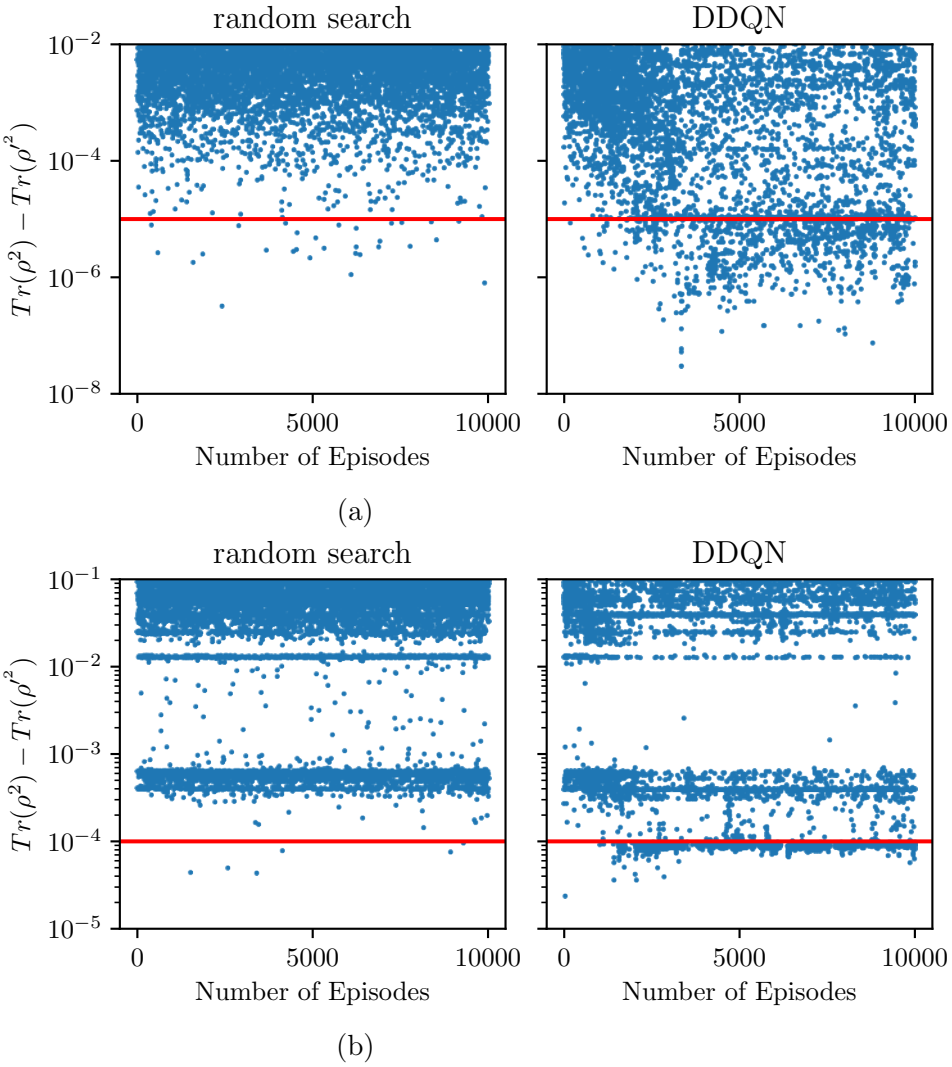


Figure 3.18: The RL-agent can give us more frequent solutions, whereas the random search can hardly solve the problem. Comparison of accuracy obtained using random search and RL-based method. Illustration of 10^4 episodes to solve the full rank random quantum state of (a) two qubits and (b) three qubits. The red line denotes the pre-defined tolerance for the approximation of the cost function.

To demonstrate the hardness of the variational diagonalization task, we utilize random search to find an efficient ansatz in this section. Unlike the previous examples where an RL-agent selects an action based on a policy, here, the action at each step is chosen randomly from a uniform distribution.

In Fig. (3.18) (in the first column), we show the results for random search to diagonalize a two- and three-qubit quantum state. It can be seen that the number of successful episodes (the episodes that pass the predefined tolerance of cost function) drastically reduces as we scale the number of qubits in the state. At the same time, the RL-agent (in the second column) provides us with a more consistent outcome.

3.5 Takeaways

The goal of this chapter was to investigate the VQSD algorithm, to identify its weak points, and to provide a novel method that can significantly enhance this technique. To summarize, our result indicated that the following aspects are crucial for the efficiency of the VQSD technique.

- **Choice of best fixed-structure ansatz** In the Sec. 3.2.3 we briefly investigate the state-of-the-art VQSD algorithm. For the sake of comparison, we choose 6 different optimization methods and 3 different generic structures of hardware efficient ansatz (HEA). Our results first show that The `RYRZRYCX` of Fig. (3.1) overall outperforms the other structures of HEA in terms of accuracy and performs well with the scaling in the number of qubits in the diagonalizing state for all the optimizers.

- **Time vs. accuracy trade-off in classical optimizer** Once again in the Sec. 3.2.3, using the LHEA, and `RYRZRYCX` as ansatz we consider `COBYLA`, `L-BFGS-B`, `POWELL`, `Nelder-Mead`, `SLSQP` and `BFGS` as classical optimizer. And through Fig. (3.5) we see that `Powell` gives us the best accuracy towards the diagonalizing unitary but it has a significant time overhead. Whereas `COBYLA` takes 100 times less time compared to `Powell`, and it fails to return a good approximation to the diagonalizing unitary. Hence, there is a time vs. accuracy overhead in the choice of classical optimizers. Due to the impressive time efficiency, we utilize `COBYLA` optimizer in RL-VQSD; because if we have $\mathcal{A}_N$ number of actions per episode and N is the number of episodes, then we need $\mathcal{A}_N \times N$ query of the classical optimizer. Hence, to minimize the time spent on each query of the classical optimizer, it is ideal to use an optimizer that takes less time, pointing towards `COBYLA`.

- **Tensor-based binary encoding scheme for quantum circuit** In **Section 3.3.1** we present a qubit efficient encoding scheme for quantum circuits

that scales polynomially with the number of qubits. Each depth of the circuit is encoded in a block of dimension $((N+3) \times N)$. We show that the encoding scheme provides more degree of freedom in qubit connectivity compared to previously proposed schemes, and the depth-based nature of the encoding ensures that it is more efficient. Each depth is filled up by the action scheme briefly described in **Section 3.3.2**. For further clearance, please see the example in the same section.

- **RL-VQSD outperforms existing algorithms** Through the numerical simulations in **Section 3.4.1**, **3.4.2** and **3.4.3** we show that RL-VQSD outperforms the state-of-the-art VQSD methods and provides us with a diagonalizing unitary with a minimal number of parameters, depth and with very high accuracy. This claim becomes more prominent when we show that in Fig. (3.15), the state-of-the-art VQSD method fails to reach the accuracy reached by RL-VQSD with the same (or even more) number of gates for the same classical optimizer. Later in Fig. (3.16), we show that using RL-VQSD, the number of gates and the depth of the diagonalizing unitary scales linearly with increasing accuracy.

- **Random search is inefficient for diagonalizing task** In **Section 3.4.4** we replace the RL-agent by random search where the actions per step are chosen from a uniformly random distribution. Our investigation shows that (1) The random search fails to achieve a better accuracy compared to the RL-agent and (2) the number of solutions after the same number of episodes decreases drastically for the random search. This not only gives us an idea about the hardness of the diagonalizing problem but also provides insight into the efficiency of the novel RL-VQSD algorithm.

Chapter 4

Ansatze synthesis using curriculum reinforcement learning for variational quantum eigensolver

This chapter presents a novel curriculum-based reinforcement learning (CRL) based quantum architecture search algorithm tailored to address the challenges inherent in deploying variational quantum algorithms in realistic noisy scenarios. This approach integrates three key elements: (i) the tensor-based ansatz encoding scheme presented in chapter 3, an *illegal action* scheme to constrain the search in the action space, enabling efficient exploration of potential circuits, an episode-halting scheme guiding the agent towards discovering shorter circuits, and a variant of the simultaneous perturbation stochastic approximation algorithm, fostering faster convergence for optimization. Through a series of numerical experiments focusing on quantum chemistry problems, we showcase our methods' performance compared to existing QAS algorithms in noiseless and noisy environments. Through the investigation, we show that our algorithm provides a more efficient ansatz than state-of-the-art algorithms. The influence of noise on the architecture search of ansatz is poorly understood. This chapter addresses this issue by showing that our CRL-based QAS algorithm can efficiently solve quantum chemistry problems under realistic quantum noises and constrained connectivity.

4.1 Introduction

The Quantum Phase Estimation (QPE) [2,3,79] is a quantum algorithm introduced to extract the eigenvalues of a unitary operator utilizing the Inverse Quantum Fourier Transform (IQFT) and phase kickback. It is shown that QPE can achieve exponential speedup in obtaining the eigeninformation of unitaries as long as the

trial state is appropriately prepared. The promise of achieving quantum advantage is evident with QPE if a big enough fault-tolerant quantum computer is available. But its subroutine with IQFT requires a lot of resources, in terms of gates and qubits, for even relatively small quantum systems. Although recent developments in QPE are explored to minimize depth [113] and computational resources [77], exploring the true potential of QPE is beyond the capabilities of present NISQ devices.

Keeping the hardware constraints in mind a hybrid quantum-classical algorithm, Variational Quantum Eigensolver (VQE) is introduced [101, 120, 129]. At its core, VQE utilizes both quantum devices and classical optimization techniques to find the ground state energy of a molecular Hamiltonian. Finding the ground state is a crucial problem in quantum chemistry and is essential for predicting chemical properties and reactions by understanding the electronic structure of molecules. This has applications from material science [98] to engineering [27]. In VQE a trial wave function or an ansatz using a Parametrized Quantum Circuit (PQC), $U(\vec{\theta})$, is prepared as follows

$$|\psi(\vec{\theta})\rangle = U(\vec{\theta})|\psi_0\rangle. \tag{4.1}$$

$|\psi_0\rangle$ is the initial state usually chosen as $|0\ldots0\rangle$. The $U(\vec{\theta})$ can be decomposed into a series of parametrized and non-parametrized using Eq. (2.6). If the electronic Hamiltonian of a molecule is defined as H_{mol}, then VQE aims to minimize the cost

$$C(\vec{\theta}) = \langle\psi(\vec{\theta})|H_{\text{mol}}|\psi(\vec{\theta})\rangle, \tag{4.2}$$

using a classical optimizer. The variational principle guarantees that $E_{\text{ground}} \leq C(\vec{\theta})$ where E_{ground} is the ground energy of H_{mol}. Meanwhile, $C(\vec{\theta})$ is the energy expectation value of H_{mol}. The electronic Hamiltonian H_{mol} is constructed (for an overview of the construction of molecular Hamiltonian see [65, 103]) with fermions and in order to evaluate the energy one can use indistinguishable fermions to distinguishable qubit mapping using the Jordan-Wigner [74], Parity [23] and Bravyi-Kiteav [24, 134] encodings. In recent years a Bravyi-Kitaev superfast [135] and a qubit-efficient encoding [137] for Hamiltonian is introduced. For a comparison QPE requires $\mathcal{O}(1)$ repetitions with circuit depth scaling in precision $\mathcal{O}(\frac{1}{\epsilon})$ whereas VQE requires $\mathcal{O}(\frac{1}{\epsilon^2})$ shots with circuit depth scaling in precision $\mathcal{O}(1)$ [159].

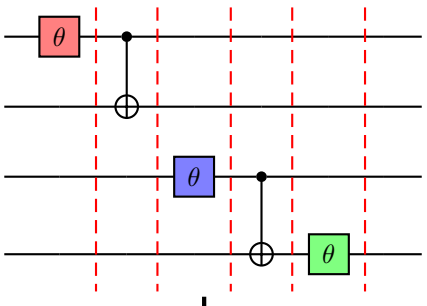
The performance of VQE can be influenced by the structure of the ansatz [56, 95, 148] due to the fact that the accuracy of the energy depends on the state manifold accessible by the PQC. So, finding new methods to construct PQC can lead to breakthroughs in VQAs for chemistry problems. In the Sec. 2.1.2, we briefly discuss the various ansatz constructions depending on how much knowledge of the Hamiltonian we have at our disposal. The number of gates and depth increases exponentially in a problem-inspired PQC as we scale up the size of the molecule.

Meanwhile, the hardware-efficient and problem-agnostic ansatz reflect trainability issues. To address these challenges, new methods have been introduced that draw on the insight and techniques of machine learning [14, 63, 128]. In recent times, numerous explored Reinforcement Learning (RL) methods to find an efficient PQC for VQE [44, 50, 116] problem.

In this chapter, we introduce a novel approach towards efficient ansatz construction using RL, which exploits a Double Deep-Q-Network [157] (see pseudocode **2** for the pseudocode). Notably, our approach outperforms the existing state-of-the-art VQE algorithm in the noiseless scenario. We also benchmark the performance of our RL-based approach under quantum noise and show that even under hardware connectivity constraints and decoherence noise, the introduced approach shows impressive performance.

4.1.1 Previous works

As for the groundwork, we are going to mainly focus on the research [116] where the authors introduce an RL method for quantum circuit construction, utilizing the Double Deep-Q network (DDQN) with an ϵ-greedy policy and an ADAM optimizer. They primarily focus on finding the ground state of the four- and six-qubit `LiH` molecule with bond distances 1.2, 2.2 and 3.4, respectively. In order to achieve a solution, they make use of a reward function of the following form.



ctrl qubit	4	0	4	2	4	4
targ qubit	0	1	0	1	0	0
rot qubit	0	4	2	4	3	4
rot axis	1	0	1	0	2	0

Figure 4.1: Example of state representation. In this example, the maximum length of the circuit L is set to 6 for four-qubit. Since we count qubits from 0, the lack of a particular gate at each layer is represented by the maximum number of qubits, i.e., in this case, it is 4, which can be seen in the last column where the `ctrl` and the `rot` qubit are both set to 4.

$$R = \begin{cases} 5 & \text{if } C_t < \zeta, \\ -5 & \text{if } t \geq L \text{ and } C_t \geq \zeta, \\ \max\left(\frac{C_{t-1}-C_t}{C_{t-1}-E_{\min}}, -1\right) & \text{otherwise.} \end{cases} \tag{4.3}$$

The C_t is calculated at each time step t according to the Eq. (4.2). Here, the main goal of the RL-agent is to achieve the $E_{\min}$ within a predefined threshold ϵ, where L is the maximum number of circuit layers. To encode the quantum circuit the authors introduce an ordered list of layers that are composed of single quantum gates. As `CX`, `RX`, `RY` and `RZ` are considered as the building blocks for quantum circuits, the environment state is represented by a list that fully describes the circuit in terms of the `CX` and one qubit gates. The parameterized rotations are encoded using two integers: the first number indicates the registered qubit, and the second is the axis of rotation. Meanwhile, `CX` gate is also represented using two integers to indicate the position of control and target qubit. To get rid of the continuous parameter of rotation angles, the estimated energy by the circuit is appended to the state representation. In Fig. (4.1), we illustrate the circuit encoding scheme the authors utilize to prepare the state. In both four- and six-qubit `LiH` problem using a global strategy[1] the authors outperform the hardware efficient and the UCCSD ansatz in terms of depth and minimum gate count.

Meanwhile, in article [44], the authors introduce a quantum architecture search method to improve the learning performance of VQAs by enhancing trainability. To do so, the work considers a pool of all possible ansatz, say $\mathcal{P}_{\mathcal{A}}$ to build the ansatz for the VQA, where

$$|\mathcal{P}_{\mathcal{A}}| = f(\mathcal{G}^{N\times L}) \tag{4.4}$$

where $\mathcal{G}$ is the set of different types of quantum gates, N is the number of qubits and L is the maximum depth of the circuit. To incorporate the realistic noisy scenario, we consider a quantum noise channel $\mathcal{C}_{a_i}$ for the a_i-th ansatz. Now, if the problem is defined through a Hamiltonian H, then the VQA objective is redefined as

$$(\vec{\theta}^*, a_i^*) = \arg\min_{\vec{\theta}, a_i \in \mathcal{P}_{\mathcal{A}}} \mathcal{L}(\vec{\theta}, a_i, H, \mathcal{C}_{a_i}), \tag{4.5}$$

The authors make use of *supernet* and *weight sharing strategy* to get a good estimation of the Eq. (4.5) in a runtime comparable with the runtime of VQAs and minimal memory usage. On the one hand, the weight-sharing strategy helps to correlate the parameters among different analyses, helping reduce the parameter space during optimization. Meanwhile, the supernet is utilized as an indicator for the ansatz in the pool $\mathcal{P}_{\mathcal{A}}$ and parametrizes each ansatz in the pool.

[1] A global strategy corresponds to optimizing all the angles of the PQC after application of each action in terms of quantum gates to the quantum circuit.

Utilizing this approach, the authors tackle the ground state finding problem of the four-qubit $\mathtt{H}_2$ molecule, showing that the energy converges to the true energy in a few iterations. However, the performance of this method is comparable to the conventional VQE in the noiseless scenario. In the case of a noisy scenario, the authors consider running in real superconducting quantum hardware, i.e., Ibmq_ourense, and they show that their method gives a better approximation to the ground state compared to a hardware-efficient ansatz-driven VQE. The estimated ground energy of the method with $W = 1$ and $W = 5$ achieves -0.93 and -1.05Ha[2], respectively. Where W is the number of supernets, the energies are better than the conventional VQE algorithm with a Hardware Efficient Ansatz (HEA), which achieves an energy of -0.4Ha.

A very recent development based on differentiable Quantum Architecture Search (QAS) to automate the design of Parameterized Quantum Circuits (PQC) is introduced in ref. [165]. Before this work in ref. [169], the authors used the differentiable search for QAS using Monte Carlo sampling to estimate the gradient by sampling multiple circuits at each epoch. This helps to get the approximation of the continuous distribution of quantum circuit architecture weights. But to achieve higher efficiency in the work ref. [165] the authors use the Gumbel-Softmax [15,59,72] technique to sample quantum circuits instead of Monte Carlo. Right after the sampling, the circuit architecture weights are updated by the gradient descent method. In the paper, the authors propose *micro* and *macro* search methods where the *micro* search focuses on searching for the sub-circuits of an architecture, and these sub-structures are later stacked to form the whole circuit. Meanwhile, Macro search directly searches for the whole circuit not focusing on the sub-structures. Using this differentiable search QAS method and $\{\mathtt{RX}, \mathtt{RY}, \mathtt{RZ}, \mathtt{P}, \mathtt{CX}\}$ as the candidate gate set, where $\mathtt{P}$ is a phase-shift gate, they find the ground state of the $\mathtt{H}_2$, $\mathtt{LiH}$ and $\mathtt{H_2O}$ problem of configuration given in Tab. 4.1.

4.2 Groundwork

We divide this section into subsections where we (1) briefly compare the tensor-based quantum circuit encoding that we use as the RL-state with previously proposed encoding schemes for the VQE task. (2) Next, we introduce a simple mechanism, namely *illegal actions*, which helps narrow down the search space significantly, and finally (3) to facilitate the agent's ability to discover more compact ansatz in early successful episodes, we introduce a technique called *random halting*. Before briefing the subroutines, we outline the RL-agent and environment specifications in the upcoming section.

[2] 1 Ha (Hartree) is 27.211 electron volt.

Molecule	Fermion to qubit mapping	Configuration	Number of qubits
H_2	Jordan-Wigner	H $(0, 0, -0.35)$; H $(0, 0, 0.35)$	4
LiH	Parity	Li $(0, 0, 0)$; H $(0, 0, 2.2)$	4
	Jordan-Wigner	Li $(0, 0, 0)$; H $(0, 0, 3.4)$	6
H_2O	Jordan-Wigner	H $(-0.021, -0.002, 0)$; O $(0.835, 0.452, 0)$; H $(1.477, -0.273, 0)$	8

Table 4.1: List of molecules considered for noisy and noiseless simulation.

4.3 Agent and environment specification

We utilize the double Deep-Q Network algorithm in the noiseless experiments with H_2, LiH four-qubit. Meanwhile, the noisy simulations and the noiseless simulation of harder molecules such as six-qubit LiH and eight-qubit H_2O, we make use of the Double Deep-Q Network step algorithm, where we utilize differing step sizes in the n-step trajectory roll-out update [146]. In these settings, we set the discount a factor set to $\gamma = 0.88$, and the probability of a random action being selected is set by an ϵ-greedy policy, with ϵ decayed in each step by a factor of 0.99995 from its initial value $\epsilon = 1$, down to a minimal value $\epsilon = 0.05$. The memory replay buffer size is set to 20000, and the target network in the DQN training procedure is updated after every 500 action. After each training episode, we included a testing phase where the probability of random action is set to $\epsilon = 0$, and the experiences obtained during the testing phase are not included in the memory replay buffer. In the curriculum learning approach, the threshold is changed greedily after 500 episodes for two-, three-, and four-qubit problems, whereas the threshold is changed after every 2000 episode for six- and eight-qubit problems with an amortization radius of 0.0001. After 50 successfully solved episodes, the amortization radius is decreased by 0.00001. The initial threshold value is set to $\varepsilon = 0.005$. Simulations of quantum circuits were performed using the Qulacs library [147]. The hyperparameters were selected through coarse grain search. The employed network is a fully connected network with 5 hidden layers with 1000 neurons each for the 4-qubit case, 2000 neurons each for the six-qubit case, and 5000 neurons each for 8-qubit. The maximum number of gates is set to 40 for four-qubit, 70 for six-qubit and 250 for eight-qubit. All experiments were performed on three computing clusters with an NVIDIA-A100 GPU.

In the case of noiseless scenario we used the global strategy with COBYLA optimizer whereas for the noiseless case we use the same strategy with the introduced

multistage Adam-SPSA optimizer with PTM formalism.

4.3.1 The tensor-based vs integer encoding

Recalling the Tensor-Based Encoding (TBE) presented in the Sec. 3.3.1 where we encode the quantum circuit depth-wise. Each depth is encoded into a 2-D grid of size $[T \times ((N+3) \times N)]$ where T is predefined as the maximum depth of the circuit and N is the number of qubits. We refer to the Sec. 3.3.1 for an elaboration of the encoding scheme.

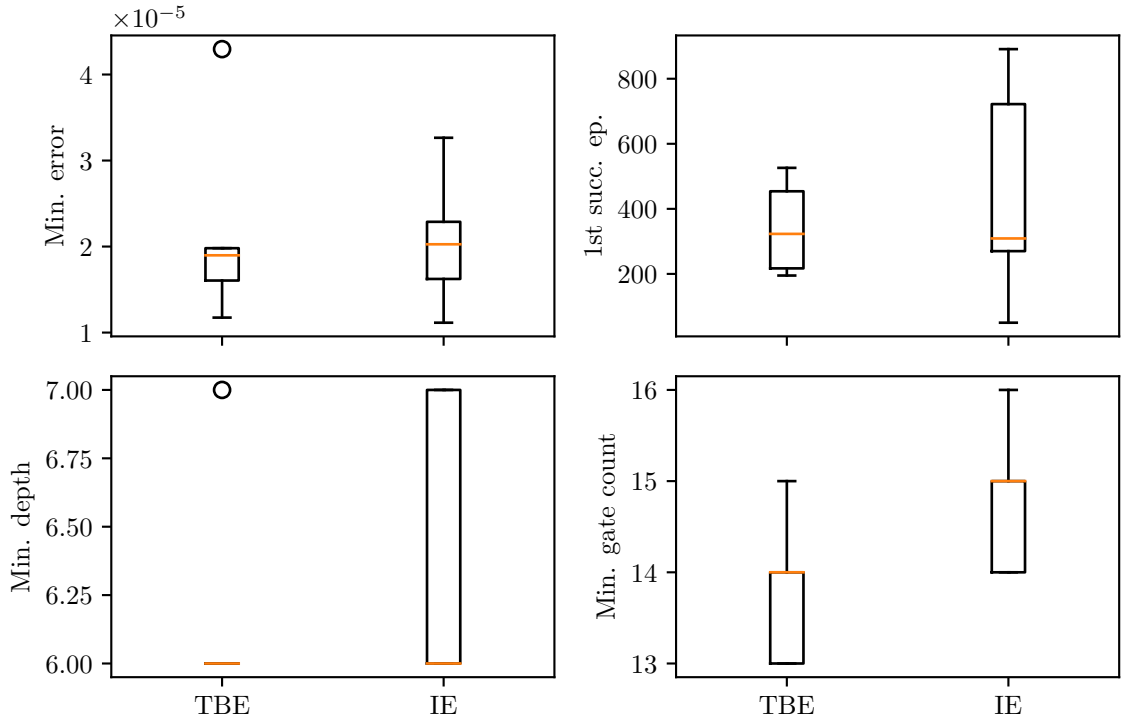


Figure 4.2: The Tensor-Based Encoding (TBE) outperforms the Integer Encoding (IE) and returns a lower depth and smaller number of gates circuit with lower error. To conduct this experiment, we consider `LiH` molecules with a bond distance of 3.4 with parity encoding. The results are averaged over 5 seeds, and for each seed, we initialize the double deep-Q network with different values. For comparison, we consider the minimum error (Min. error), the first successful episode (1st succ. ep.), the minimum depth (Min. depth) and the minimum number of gates (Min. gate count) for both the TBE and IE. Additionally, the TBE makes the agent more stable compared to IE by preventing the spread of the deviation from the median.

Recalling the Tensor-Based Encoding (TBE) presented in the Sec. 3.3.1 where we encode the quantum circuit depth-wise. Each depth is encoded into a 2-D grid of size $[T \times ((N+3) \times N)]$ where T is predefined as the maximum depth of the circuit and N is the number of qubits. We refer to the Sec. 3.3.1 for an elaboration of the encoding scheme.

In one of the very first works in [116], the authors introduce an integer-based encoding scheme where each block of the RL-state carries information about each gate applied. In order to show improvement over this state-of-the-art encoding scheme, here we compare the performance of the Integer Encoding (IE) with the encoding presented in this thesis, namely TBE. In a nutshell, the investigation shows that TBE outperforms IE in all aspects. To do this experiment, we consider the four-qubit LiH molecule with parity (fermion to qubit) encoding. For a detailed geometry of the configuration of LiH, see Tab. 4.1.

In the Fig. (4.2) we illustrate the results and as a measure of performance, we consider the minimum error in energy (*min error*), the first successful episode (*1st succ. ep.*), the minimum depth of the ansatz (*min. depth*), and the minimum number of gates (*min. gate count*). The first thing to notice in Fig. (4.2) is that the TBE is more stable in IE, which can be seen through the spread of the quartiles.

Another remarkable aspect of the TBE is that it gives the minimum error in the ground state energy lower than the IE with a smaller number of gates. This is ideal for the NISQ era to mitigate the negative impact of gate errors and decoherence effectively. It is important to ensure the circuits are both gate-efficient and have minimal depth.

4.3.2 Illegal actions: The reduction of search space

The QAS algorithms present a challenging combinatorial problem, which is characterized by an extensive search space. In order to narrow down this search space proves advantageous for the RL-agent. This not only helps the agent to discover a quantum circuit with diverse structures but also improves the learning by the agent.

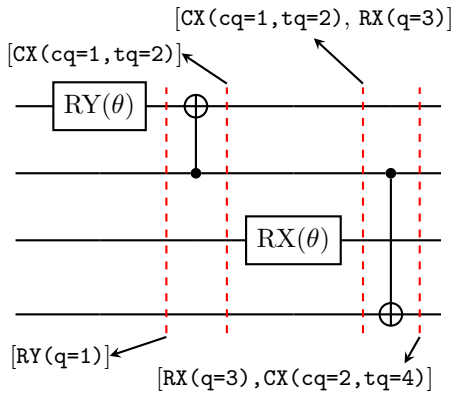


Figure 4.3: The illegal action scheme on four-qubits. Here, `cq` and `tq` represent the control and the target qubit. This approach encourages the RL-agent to avoid selecting the same action in two consecutive steps, effectively reducing the action space.

Therefore, we present a straightforward approach referred to as *illegal actions* to greatly reduce the search space. At the core of the technique, it leverages the inherent property of quantum gates being unitary, which results in the cancellation of two similar gates when applied to the same qubit. We employ this mechanism to reduce the search space for the RL-agent. In Fig. (4.3) illustrate the `illegal action` mechanism on a four-qubit system, and an elaborated discussion of the technique is provided in Appendix C. Meanwhile, in the Appendix D.2, we provide a code that is used to implement the illegal actions technique.

Although illegal action mechanism is straightforward to implement for any ansatz construction, one can consider more complex pruning of quantum circuits as provided in [50].

4.3.3 Investigation of reward function

A well-defined reward function can accelerate the rate of convergence of an RL-agent towards the target. There are many ways to formulate a reward based on the task under consideration. In previous work [116], for solving chemistry problems using Deep neural networks, a reward function of type 4.3 is considered, which is quite sparse in nature.

In this section, we keenly investigate and compare the reward presented through Eq. (4.3) and the reward that is used in Chapter 3 of the form

$$R(\mathcal{R}) = \begin{cases} +\mathcal{R} & \text{for } C_t(\vec{\theta}) < \zeta + 10^{-5} \\ -\log\left(C_t(\vec{\theta}) - \zeta\right) & \text{for } C_t(\vec{\theta}) > \zeta \end{cases} \tag{4.6}$$

where the $\mathcal{R}$ is a positive large number. For the sake of investigation, we compare the performance of the reward function 4.3 with $R(\mathcal{R}=0)$, $R(\mathcal{R}=50)$, $R(\mathcal{R}=100)$, $R(\mathcal{R}=500)$ and $R(\mathcal{R}=1000)$.

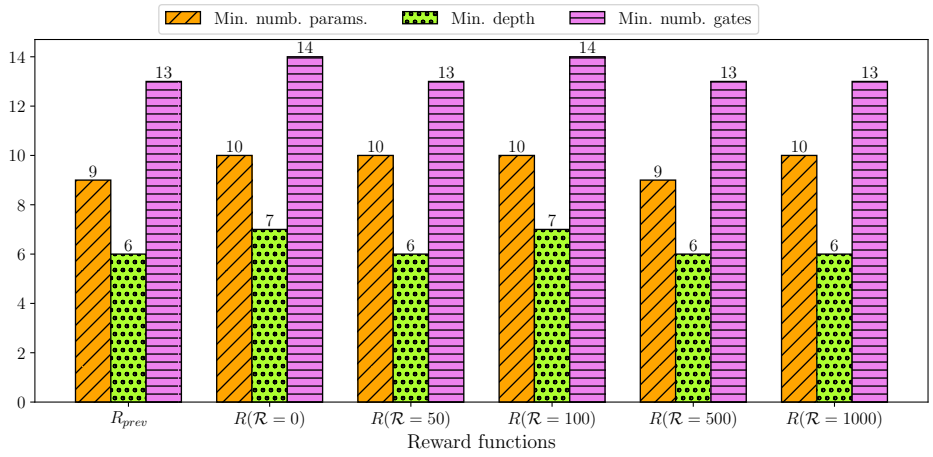


Figure 4.4: Comparison of different reward functions where using $R_{\textbf{prev.}}$ and $R(\mathcal{R}=500)$ we get the minimum number of parameters (Min. numb. params.), minimum depth (Min. depth) and number of gates (Min. numb. gates) for four-qubits `LiH` molecule. We evaluate the models based on 5 different seeds where in each seed the neural network is initialized with different input. It should be noted that both the rewards i.e. $R_{\textbf{prev.}}$ and $R(\mathcal{R}=500)$ give us the same averaged error (9.99×10^{-4} for $R_{\textbf{prev.}}$ and 1.06×10^{-3} for $R(\mathcal{R}=500)$) over the 5 seeds.

It can be seen from Fig. (4.4) that the performance of the RL-agent with $R_{\text{prev.}}$ is similar to the performance with $R(\mathcal{R}=500)$ and the reward Eq. (3.10) with $\mathcal{R}=500$ outperforms the other variants. An in-depth investigation of the $R_{\text{prev.}}$ and $R(\mathcal{R}=500)$ unveils that the average number of successful episodes over 5 different seeds is 16567 for $R_{\text{prev.}}$ and for $R(\mathcal{R}=500)$ it is just 2301 but the later reward function helps us to achieve the first successful episode faster. For $R_{\text{prev.}}$, the first successful episode on average appears at episode 343 whereas, for $R(\mathcal{R}=500)$, it is at 181-th episode.

In this chapter, we primarily focus on finding a very compact ansatz. Hence it is significant to have a higher number of successful episodes because it will yield a greater array of ansatz options for our investigation and to pick the best one among them. That's why we decided on utilizing the $R_{\text{prev.}}$ instead of $R(\mathcal{R}=500)$.

In the upcoming section, we introduce a straightforward technique aimed at enabling the RL-agent to accommodate shorter-length episodes in the first few successful episodes.

4.3.4 Random halting: quickly discovering compact ansatz

In the case of the previous works, such as in [116], a full-length episode can be decomposed into a constant number of time steps T_s. Each time when noise is applied to a quantum circuit, a CPTP channel is applied to the circuit which not only reduces the performance of the circuit to achieve a task but increases the computation time by many times compared to the noiseless scenario.

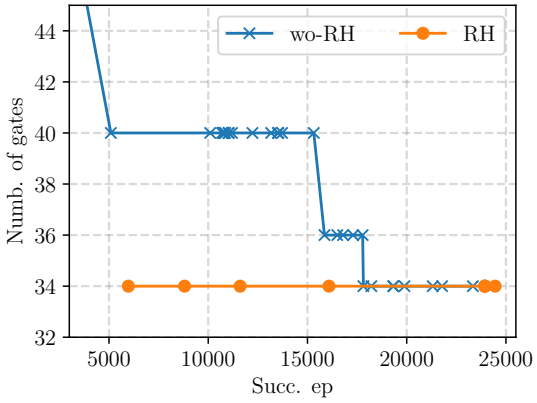


Figure 4.5: The *Random halting* (RH) gives a more compact ansatz compared to without RH settings in very early successful episodes (Succ. ep) with six-qubit `LiH` molecule. It can be seen with RH in around $5,000$ episodes. The minimum number of gates (Numb. of gates) we require to solve the six-qubit `LiH` problem reaches 34. Meanwhile, reaching the same number of gates without RH settings requires around $17,000$ episodes. Hence, RH helps the RL-agent learn 3 times faster, the optimal quantum circuit compared to without RH settings.

Hence, in a realistic scenario, in the midst of quantum noise, the RL-agent begins by proposing a lengthy ansatz in terms of the number of gates and depth that achieves chemical accuracy in the initial few successful episodes. However, over thousands of subsequent episodes, it gradually shifts towards a shorter ansatz. This situation is suboptimal due to the significantly extended duration of noisy simulation.

To address this challenge, we propose a method referred to as *Random Halting* (RH), where the value of T_s is no longer a constant parameter. Instead, it varies from one episode to another according to a particular probability distribution that is dependent on the number of qubits. To be more precise, we sample the episode-specific number of step s, denoted as T_s, from the following negative-binomial distribution:

$$T_s \sim \binom{n_f + n_s - 1}{n_f} p^{n_f} (1-p)^{n_s}, \tag{4.7}$$

In this context, n_s represents the count of successes. Meanwhile, n_f denotes the count of failures. The sum of successes and failures determines the total number of trials, represented as $n_f + n_s$, and p signifies the probability associated with each success.

The primary motivation for incorporating RH into the algorithm is to empower the RL agent to accommodate shorter episode lengths. This, in turn, enhances the agent's capability to uncover more concise ansatz in the early stages of successful episodes, even if it occasionally delays achieving the first successful episode. We observe in Fig. (4.5) that within approximately $5,000$ episodes, the minimum number of gates needed to solve the six-qubit LiH problem (see Tab. 4.1 for details of the molecule geometry) decreases to 34. In contrast, reaching the same gate count without RH settings necessitates approximately $3\times$ more episodes.

4.3.5 Multistage ADAM-SPSA algorithm

In the case of VQE, the budget for measurement samples is restricted. To exhibit robustness against finite sampling noise, several versions of Simultaneous Perturbation Stochastic Approximation (SPSA) are utilized (Cade et al., 2020; Bonet et al., 2023). Among these variants, multi-stage SPSA adjusts the decaying parameters while tuning the permitted measurement sample budget between stages. Moreover, incorporating a moment adaptation subroutine from classical machine learning, like Adam (Kingma & Ba, 2014), alongside standard gradient descent helps us enhance the robustness and accelerates the convergence of the algorithm.

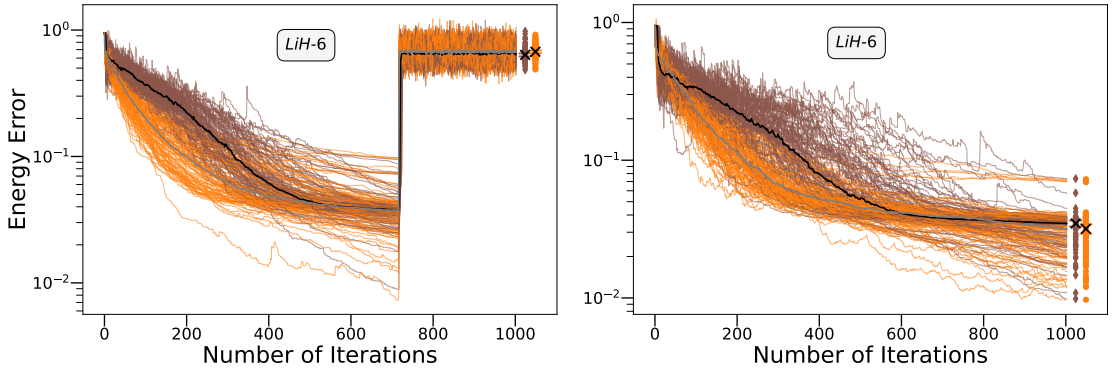


Figure 4.6: Optimization traces of the 3-stage sampling strategy of SPSA (brown and black) and Adam-SPSA (orange and grey) on the sex-qubit `LiH` (`LiH`-6) molecules using the hyperparameters outlined in Appendix D.3. The individual traces are represented by thin lines, while the thick line on top indicates the median of 100 independent runs. The left and right panels showcase the resetting and continuous evolution of SPSA (Adam-SPSA) hyperparameters, respectively.

As we discussed in the previous section, noise increases the computational time per episode, and it is very crucial to enhance the robustness and faster convergence rates of VQAs under realistic scenarios. For this reason, we leverage a 3-stage Adam-SPSA (whose pseudocode is provided in the Appendix D.3). For a rigorous illustration of the 3-stage Adam-SPSA we investigate six-qubit `LiH` (`LiH`-6) molecule where the hyperparameters of the algorithm are set according to the Tab. D.1 in the Appendix D.3. We see that in 3-stage Adam-SPSA, unlike the vanilla SPSA without Adam momentum, the convergence towards the minima is qualitatively much faster which is qualitatively shown in Fig. (4.6).

Utilizing these insights from our analysis of various SPSA variants, we employ 1- and 3-stage Adam-SPSA in our noisy experiments. This helps cut down the total number of function evaluations by half, thereby doubling the speed of our RL training. This improvement at the algorithm level helped us simulate noisy systems that suffer from computational complexity and large run times.

4.3.6 Pauli-transfer matrix formalism on GPU

Restating the fact that QAS demands a significant number of noisy function assessments unless a training-free approach is adopted. However, executing the steps poses enormous challenges within state-of-the-art simulation framework. The noisy simulation process not only encounter difficulty due to the curse of dimensionality related to dense matrix operations but also due to the exponential increase in the number of noise channels and their corresponding Kraus operators.

To address this challenge, a Pauli-transfer matrix (PTM) formalism is utilized, enabling the precomputation of noise channel fusion with respective gates offline. This eliminates the need for recalculations at each step. Alongside PTM formalism we integrate GPU computation along with just-in-time (JIT) compiled functions in JAX, resulting in up to a 6× enhancement in RL-agent training efficiency while simulating noisy quantum circuits.

4.4 Curriculum reinforcement learning

The moving threshold technique (see Fig. 4.7) is a feedback-driven curriculum learning method introduced in [116]. During the learning process, the agent pursues a parameter ξ_2 that marks the lowest energy known by the agent so far and updates a threshold parameter with respect to this parameter based on some rules. In the beginning, the ξ_2 parameter is set to a hyperparameter ξ_1. If the agent finds an energy value lower than the current one, it updates ξ_2 to this new energy value. Another hyperparameter “fake minimum energy" μ, a proxy to the lower bound

of attainable ground state energy, is set as a target for the agent.[3] We compute this proxy by taking the summation of absolute values of Pauli string coefficients stemming from the Hamiltonian.

In the absence of amortization, the algorithm shifts the threshold to $|\mu - \xi_2|$ for the new ξ_2. In the presence of amortization, however, it adds a parameter to that threshold as $|\mu - \xi_2| + \delta$, where δ is the amortization hyperparameter. In the meantime, the agent continues its exploration with subsequent actions and episodes and records the number of successful actions. Here, there are two rules at play. The first rule greedily shifts the threshold to $|\mu - \xi_2|$ after G episodes. Here G is a hyperparameter as well. The second rule slowly decreases the threshold parameter each time there is a successful episode by subtracting a factor of δ/κ. Here κ is the radius of shifts, also a hyperparameter. Upon setting the threshold to $|\mu - \xi_2|$, if the agent fails to improve the energy value in consecutive episodes, the threshold is increased back to $|\mu - \xi_2| + \delta$, as demonstrated in Fig. (4.7). This way, the agent is given an opportunity to trace its steps back if it was stuck in a local minimum.

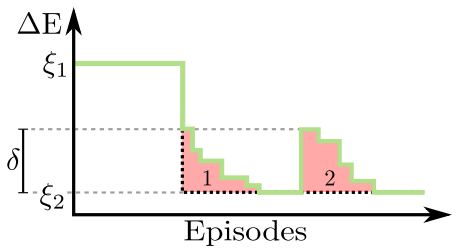


Figure 4.7: Demonstration of the feedback-driven (green) process, depicting two amortization occurrences (pink), δ. The initial occurrence adjusts the threshold from ξ_1 to ξ_2, signifying the improvement. The subsequent event occurs when the agent fails to surpass ξ_2 or the gain is marginal. This prompts a sudden threshold increase due to amortization reset. Note that the final threshold, after the second amortization, may be less than ξ_2.

Notably, this method does not require any prior knowledge regarding the true value of the ground state energy and does not impose any specific constraints on the initial threshold value, unlike existing QAS methods in the literature.

4.5 Results

In this section, we in detail present the results of finding the ground state of H_2, LiH, and H_2O molecules of two-, three-, four- and eight-qubit. The structure of

[3]One can set the target of the agent to such a value for VQE because, from Rayleigh's variational principle, the agent theoretically can never attain energy below the true ground state energy.

the molecules is provided in Tab. 4.1. We initiate the section by simulating the molecules in a noiseless scenario and comparing its results with the state-of-the-art QAS algorithms. Through the rigorous comparison, we show that our CRL-based VQE algorithm outperforms the state-of-the-art and the existing learning-based QAS algorithm in tackling the same optimization task. Later on, we run our algorithm to find the ground in a realistic noisy scenario obtained from IBM devices such as `ibmq_mumbai` and `ibmq_ourense`.

4.5.1 Noiseless case

This section primarily focuses on the noiseless simulation of the molecules listed in the Tab. 4.1. We compare the performance of our algorithm to the existing algorithms such as *RL-VQE* introduced in [116] and *quantumDARTS* described in [165] while showing that our algorithm outperforms them.

To obtain the results, we consider the six-qubit `LiH` and right-qubit H_2O molecule. The detailed configurations are in the Tab. 4.1. The results are summarized in Tab. 4.3. It can be seen from the table that our algorithm outperforms the UCCSD, RL-VQE and the quantumDARTS ansatz and provides a quantum circuit with a smaller number of gates (N_G) and parameters (N_P) for six-qubit `LiH` but the UCCSD ansatz provides smaller error in the ground state with the trade-off having 18 times more gates, which is really costly.

On the other hand, for eight-qubit H_2O, we compare our algorithm's performance with the quantumDARTS algorithm and show that we achieve 1.68 times lower error in ground energy with 79 fewer quantum gates and with 4.31 times less parameters.

4.5.2 Noisy case

Let us now discuss the effect of quantum noise, such as shot noise and real device noise with connectivity constraints, in the proposed algorithm. To see the performance of the CRL-based VQE method, we find the ground state of H_2 molecule with two-, three-qubit and `LiH` with four-qubit under different amplitude of shot noise. Furthermore, we take the maximum noise from the `ibmq_mumbai` and `ibmq_ourense` device of `IBM quantum` and uniformly apply it to all the qubits for three- and four-qubit H_2 molecule. The noise includes the single, two-qubit depolarizing noise, readout error, thermal relaxation, and single and two-qubit gate time without connectivity constraints for `ibmq_mumbai` and with connectivity constraints for `ibmq_ourense`.

Table 4.2: Tabular representation of the maximum noise of `ibmq_mumbai` device. Additionally, the qubit frequency and the anharmonicity are the same for maximum noise settings and are set to 4.896 GHz and −0.33 GHz, respectively.

Model/Noise	1q dep.	2q dep.	Read. error	Therm. rel. noise (μs)	1q gate time (s)	2q gate time (s)
Max	1.45×10^{-3}	2.30×10^{-2}	8.7×10^{-2}	$T_1 = 122.286$ $T_2 = 167.2$	35×10^{-9}	739.5×10^{-8}

Methods \ Molecule	six-qubit `LiH`				eight-qubit H_2O			
	ε	N_P	N_D	N_G	ε	N_P	N_D	N_G
Ours (RH)	1.22×10^{-3}	15	14	25				
Ours (wo-RH)	1.98×10^{-4}	25	22	42	1.84×10^{-4}	35	75	140
UCCSD	4.0×10^{-5}	224	347	464				
RL-VQE	CA	17	11	27				
quantumDARTS	2.9×10^{-4}	80	54	132	3.1×10^{-4}	151	64	219

Table 4.3: Our algorithm outperforms the existing ansatz such as UCCSD, the ansatz proposed in RL-VQE and the quantumDARTS algorithm in terms of the number of parameters (N_P) and number of gates (N_G) for six-qubit `LiH` molecule and in terms of the error in energy (ε), N_P and N_G for eight-qubit H_2O molecule. In the table, the empty cells correspond to the results that are not relevant or unavailable for that particular algorithm. The *CA* corresponds to chemical accuracy. For six-qubit `LiH`, we run our algorithm with the random halting technique and without it and present both results. On the other hand, for H_2O, we only conduct the simulation without random halting settings.

In [44], the authors consider the four-qubit H_2 molecule under `ibmq_ourense` device noise and constrained connectivity. Hence, we compare the performance of our algorithm with the one introduced in [44] in Tab. 4.4, highlighted with blue colour) and show that our algorithm outperforms in terms of error in energy estimation (ε), number of parameters (N_P), depth (N_D), and number of gates (N_G). Using the QAS algorithm [44], the minimum error in energy recorded is 1.88×10^{-2} with the number of parameters 10, depth 9 and the number of gates 16, but using our algorithm, we achieve an error in energy 2.98×10^{-4} with ansatz containing 6 parameters, 6 depth and with just 10 gates.

Additionally, from the results, we conclude that the algorithm we present is susceptible to shot noise, and we solve almost every instance of the neural network for two-, three-, four-qubit H_2 and `LiH` molecules. For two- and three-qubits, the error in the ground energy estimation goes well below 10^4, and for the four-qubit, we get an error below 10^{-3} (below chemical accuracy). The parameter `SN` corresponds to the amplitude of shot noise applied for the molecule. It can be seen that even with 10^3 (two-qubit H_2) and 10^4 (three-qubit H_2) shots our algorithm finds an error below 10^{-4} with just 8 and 5 gates respectively.

Finally, to show the diversity of our algorithm in Tab. 4.4 we solve the ground state of the three-qubit H_2 molecule under maximum noise of `ibmq_mumbai` as presented in Tab. 5.1. We show that in all the seeds, we are able to solve the molecule with an error in energy in the order of 10^{-4} with a minimum number of parameters 2, depth 7 and a total number of gates 8.

Table 4.4: Our algorithm solves the two-, three-, four-qubits $\mathtt{H_2}$ in all initialization of the neural network and four-qubit `LiH` problem in 2 out of 3 seeds in the presence of different amplitude of shot noise. Our algorithm outperforms the QAS algorithm presented in [44] under `ibmq_ourense` noise and connectivity in terms of the number of parameters (N_P) and number of gates (N_G) for six-qubit `LiH` molecule and in terms of the error in energy (ε). Unlike in QAS [44] where the algorithm could not achieve the chemical accuracy, we show that using our algorithm, we can go $10\times$ below chemical accuracy using an ansatz with 6 parameters and 4 rotations for four-qubit `LiH` problem. The `SN` corresponds to the number of shots that are considered for the molecule.

Molecules \ Methods	Our algorithm				
	seed	ε	N_P	N_D	N_G
$\mathtt{H_2}$ $(\mathtt{2qubit}, \mathtt{SN} = 10^3, \mathtt{RH})$	100	3.63×10^{-6}	4	4	5
	101	1.16×10^{-4}	14	15	16
	102	9.25×10^{-6}	38	24	40
$\mathtt{H_2}$ $(\mathtt{3qubit}, \mathtt{SN} = 10^4, \mathtt{RH})$	100	2.81×10^{-5}	6	7	9
	101	4.31×10^{-5}	7	4	9
	102	7.94×10^{-5}	5	5	8
$\mathtt{H_2}$ $(\mathtt{4qubit}, \mathtt{QAS}, \mathtt{RH})$	100	3.30×10^{-4}	7	8	13
	101	2.98×10^{-4}	6	6	10
	102	3.30×10^{-4}	7	8	13
$\mathtt{H_2}$ $(\mathtt{3qubit}, \mathtt{ibmq\_mumbai\ max}, \mathtt{RH})$	100	4.38×10^{-4}	2	8	8
	101	3.38×10^{-4}	3	7	8
	102	3.94×10^{-4}	2	7	8
$\mathtt{LiH}$ $(\mathtt{4qubit}, \mathtt{SN} = 10^6, \mathtt{RH})$	100	1.32×10^{-3}	23	16	31
	101	1.19×10^{-3}	25	15	35
	102	NS	NS	NS	NS

4.6 Takeaways

This chapter introduced a vanilla and curriculum reinforcement learning-based quantum architecture search algorithm for variational quantum algorithms. The algorithm is benchmarked for under noiseless and realistic noisy scenarios based on `IBM` hardware. The crucial takeaways from the chapter are as follows

- **Tensor-based ansatz encoding provides efficient data representation for RL** In Sec. 4.3.1 we introduce a depth-based binary encoding, namely TBE (tensor-based encoding) for quantum circuits that we utilize as an RL-state. In the very heart of the encoding lies a 3D grid structure where each dimension carries information about the depth, the type of the gate, and the position of the gate (i.e. on which qubit the gate is to be placed), respectively. The grid is of size $[T \times ((N+3) \times N)]$ where T is a predefined number corresponding to maximum depth and N is the number of qubits.

  We benchmark the TBE with previously proposed integer encoding, namely IE, in the task of finding the ground state of molecules. Through Fig. (4.2), we simulate a four-qubit `LiH` molecule and show that the TBE is more stable than IE, and it gives the minimum error in the ground state energy lower than the IE with a smaller number of gates. This is beneficial for the NISQ era, as it helps effectively mitigate the negative impact of gate errors and decoherence.

- **Enhanced insight on the reward function** In Sec. 4.3.3, we extend our understanding of a dense and sparse reward based on two formulations of the reward function in finding the ground state of the four-qubit `LiH` molecule. The first kind of reward we consider is introduced in the RL-VQSD chapter by Eq. (3.10), namely log reward, which depends on a large positive integer $\mathcal{R}$. For the sake of understanding the performance of the reward function, we choose $\mathcal{R} = 0, 50, 100, 500, 1000$ and compare it with the reward function proposed in [116] (see Eq. (4.3)), namely R_{prev}.

  The results are illustrated in Fig. (4.4) where we can clearly see that the log reward improves as the $\mathcal{R}$ increases up to $\mathcal{R} = 500$, and after that the improvement diminishes. Interestingly, the performance of the log reward at $\mathcal{R} = 500$ mimics the performance of the R_{prev} in terms of the minimum number of gates, depth, number of parameters in ansatz and even in the accuracy of estimating the ground energy. But as the number of successful episodes with the R_{prev} is larger than the log reward at $\mathcal{R} = 500$, we utilize the R_{prev} as the reward function for larger molecules. The main motivation behind this is that the higher the number of successful episodes, the greater

the array of ansatz. This helps us investigate a wide arrangement of gates in an ansatz and pick the best one among them.

- **Random halting helps discover compact ansatz quickly** Throughout the Sec. 4.3.4, we elaborate on a simple technique called the *random halting* (RH), which is introduced by keeping the realistic noisy scenario in mind. In the midst of quantum noise, predictably, the RL-agent starts by proposing a lengthy ansatz in early successful episodes. However, over thousands of subsequent episodes, it gradually shifts towards a shorter ansatz. This is inefficient in terms of the computational time of our algorithm when we compare it with the noiseless case. Hence, to address this challenge, we introduce this method where the number of time steps per episode is a variable and changes from one episode to another based on the probability distribution provided in Eq. (4.7).

  In the Fig. (4.5), we illustrate the number of gates it requires to solve a six-qubit `LiH` molecule in the presence and absence of RH. We clearly noted that in the presence of RH, the RL-agent learns 3× faster than the optimal quantum circuit without the RH setting. However, it should be noted that the number of successful episodes drastically decreases with RH.

- **Solving molecules under physical noise and connectivity constrained** In the Sec. 4.5.2, we utilize our algorithm to find the ground state of H_2 and `LiH` molecules with two-, three- and four-qubit. For our simulation, we consider shot and physical device noise. The noise is imported from the `IBM quantum` hardware `ibmq_ourense` (we consider the maximum noise among all the qubits from the noise model is provided in ref. [44] and uniformly applied to all qubits taking the qubit connectivity into account) and `ibmq_mumbai` (we consider the maximum noise among all the qubits from the noise model in Tab. 4.2 and uniformly apply it all qubits and does not take the qubit connectivity into account). In the case of shot noise, we show that for two- and three-qubit H_2 and four-qubit `LiH` molecule, we can solve the problem with 10^3, 10^4 and 10^6 shots with 5, 9 and 31 gates respectively. Under `ibmq_ourense` noise, we solve four-qubit H_2 in 10 gates, and it takes 8 gates to solve three-qubits H_2 with `ibmq_mumbai` noise.

  This shows that our algorithm is susceptible to shot noise and can solve `LiH` problem with ease. Meanwhile, for device noise and constrained connectivity, we can solve the four-qubit H_2 with a very small ansatz.

- **Introduced algorithm outperforms existing QAS algorithms** Through the Tab. 4.3 and Tab. 4.4 we compare the performance of our algorithm with the RL-VQE [116], quantumDARTS [165] and the net based QAS [44]

algorithms. We show that in the case of the noiseless scenario, our algorithm outperforms the RL-VQE and the quantumDARTS in terms of the number of gates, accuracy in energy estimation, parameter number and depth of the ansatz. Meanwhile, for the noisy scenario, the authors in using the QAS algorithm [44] could not find the chemical accuracy but using our algorithm we not only provide a shorted ansatz and find the ground energy.

Chapter 5

Variational certification of quantum channels: An application of RL-VQSD

The goal of this chapter is to describe an application of VQSD techniques in the area of quantum technologies. To this end, we focus on a protocol for quantum channel certification based on the variational approach. We demonstrate the building blocks of the protocol, and we demonstrate the implementation of the proposed algorithm on a near-term quantum computer. The results in the chapter are based on [85], and the accompanying source code can be found in [84]. After introducing the quantum channel certification algorithm, we elaborate on how the RL-based quantum architecture search method can be used to enhance the performance of the certification algorithm.

5.1 Introduction

One of the primary applications of quantum state diagonalization is the certification of quantum devices. However, certification tasks pose a significant challenge in quantum computing applications. The certification of the characteristics of a quantum system is similar to trying to recreate the results we can get from a regular classical simulation. However, this task is computationally complex, which aligns with the essence of quantum supremacy [9, 19, 60, 99, 150].

The challenge in the certification of a quantum device primarily arises from the inherent computational advantage of quantum computers. Hence, it is better to explore the potential use of quantum computers to certify quantum devices. Here we dive into the scenario where we present a certification approach that is based on the structure of the space of quantum operations. The inherent link

between states and operations in quantum mechanics i.e. the Choi-Jamiołkowski isomorphism [36, 71] leads to novel techniques of quantum information processing that has the potential to go beyond the possibilities of classical mechanics. If the quantum device is represented through a quantum channel Φ then through Choi-Jamiołkowski isomorphism we get the corresponding state as

$$\rho_\Phi = \mathcal{J}(\Phi) = (\mathbb{I} \otimes \Phi) \sum_{i=1}^{n} |i\rangle \otimes |i\rangle, \tag{5.1}$$

where $\sum_{i=1}^{n} |i\rangle \otimes |i\rangle$ is the maximally entangled states.

The problem of distinguishing between two or more quantum devices is equivalent to defining the distance in the space of density matrices. The fascination with the physical implementation of quantum information processing has led to the introduction of a class of distance/similarity measures, as evident from the substantial work carried out in this area [45,73,122,160]. Notably, with a concentrated emphasis on assessment of the practical viability of the suggested methodologies [133].

One of the well-known measures of similarity between two quantum states is quantum state fidelity which is defined as follows [156]

$$F(\rho, \sigma) = ||\sqrt{\rho}\sqrt{\sigma}|| = \mathrm{tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}, \tag{5.2}$$

it gives us the quantum counterpart of the Bhattacharyya coefficient [17] which measures the similarity between two probability distributions and it reduces to the scalar product for rank-1 operators.

A significant research effort has been devoted to finding methods for approximating fidelity [92]. Hence, for ease of calculation, the bounds for the values of fidelity using the functional are introduced [104]. The bounds are defined by the sub- and super-fidelity bounds (SSFB)

$$F_{\mathrm{sub}}(\rho, \sigma) = \mathrm{tr}(\rho\sigma) + \sqrt{2\left[\mathrm{tr}(\rho\sigma) - \mathrm{tr}(\rho\sigma)^2\right]}, \tag{5.3}$$

$$F_{\mathrm{sup}}(\rho, \sigma) = \mathrm{tr}(\rho\sigma) + \sqrt{\left(1 - \mathrm{tr}\rho^2\right)\left(1 - \mathrm{tr}\sigma^2\right)}], \tag{5.4}$$

which satisfies the property

$$F_{\mathrm{sub}}(\rho, \sigma) \leq F(\rho, \sigma) \leq F_{\mathrm{sup}}(\rho, \sigma). \tag{5.5}$$

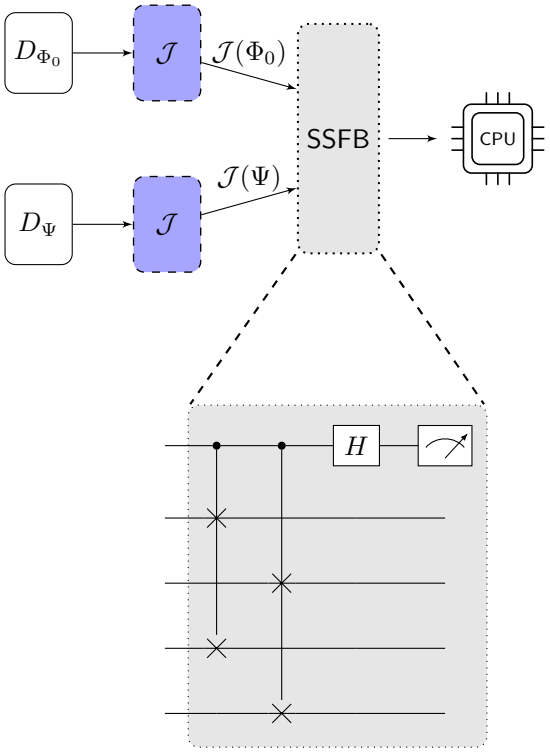


Figure 5.1: The super and sub-fidelity bound-based quantum device certification procedure. In this scheme, the goal is to certify a quantum device Ψ against an ideal device Φ_0 using the calculable bounds for fidelity. The input of the procedure is given in the form of physical devices, D_{Φ_0} and D_Ψ for the ideal and the unknown quantum channel respectively. In the CPU part of the scheme, we do classical post-processing of the results obtained using the quantum subroutine.

It should be noted that there is no exact quantum algorithm that can be used to calculate the fidelity. This is because the calculation of the fidelity requires the non-integer powers of the quantum states. In [31] the authors introduce the Variational Quantum State Fidelity (VQFE) algorithm which is a variational quantum-classical algorithm to find the bounds of fidelity. The VQFE computes upper and lower bounds on fidelity. The bounds are based on the *truncated fidelity*, which can be evaluated using Eq. (5.2) for state σ and a state ρ_m. The state ρ_m is called the truncated state of ρ that can be obtained by projecting the stateρ onto the subspace associated with its m-largest eigenvalues. The bounds can be tightened monotonically with the increment in m, and finally, the bounds will converge to the true fidelity when $m = \text{rank}(\rho)$. The bounds on the fidelity between states ρ and σ are expressed by

$$F(\rho_m, \sigma_m^\rho) \leq F(\rho, \sigma) \leq F_*(\rho_m, \sigma_m^\rho), \tag{5.6}$$

where

$$F_*(\rho_m, \sigma_m^\rho) = ||\sqrt{\rho_m}\sqrt{\sigma_m^\rho}|| + \sqrt{(1 - \mathrm{tr}\rho_m)\,(1 - \mathrm{tr}\sigma_m^\rho)}], \tag{5.7}$$

where $\sigma_m^\rho = \Pi_m^\rho \sigma \Pi_m^\rho$ is the operator obtained as the projection of σ onto the subspace spanned by m largest eigenvectors of ρ. The $F_*(\rho_m, \sigma_\rho^m)$ is the *truncated fidelity*. The F_* is also utilized to compute quantum Fisher information [140].

Following these developments, we present a novel algorithm that utilizes the super and sub-fidelity bounds and VQFE procedures as building blocks for quantum device certification. We achieve this by combining the procedures for the estimation of bounds on the fidelity with the resulting density matrix obtained by using Choi-Jamiołkowski isomorphism given in Eq. (5.1). In Fig. (5.1) we illustrate an algorithm based on the bounds given in Eq. (5.3) and in Eq. (5.4).

The procedure takes two devices as input – the standard device (Ψ) with the operational capacity already confirmed, and the device for which its conformation with the standard device is to be confirmed.

One should note that in this scheme classical data processing is required only at the final step of the procedure. This step is required to compute the bounds for the fidelity based on the measurement results.

In the following, we first briefly discuss the problem statement and then describe the novel quantum device certification algorithm. Next, we briefly discuss the results of the algorithm which is followed by a brief investigation of the results. Finally, we give conclusive remarks.

5.2 Groundwork

5.2.1 Problem statement

Let's consider a scenario where a quantum start-up has successfully developed a quantum device that can address critical optimization problems or can generate valuable states essential for quantum communication protocols. In this context, it becomes crucial to provide some testing procedures that will reassure the buyers about the device's actual functionality. Hence, the main object of the buyer would be to verify the quantum device, if it performs as advertised by the seller.

In the general context of differentiating between quantum channels, it is customary to assume that we have for our disposal a set of N quantum devices that are denoted by the quantum channels $\Psi_1, \Psi_1, \ldots, \Psi_N$. Each device operates as a black box, which directly indicates that we are not aware of the Kraus representation of the channels. In such a situation, it is impossible to determine whether the input devices can be perfectly distinguished [45].

However, in our situation, the task is straightforward. All we intend to do is to convince the buyer that the device we would like to sell emulates the operation of

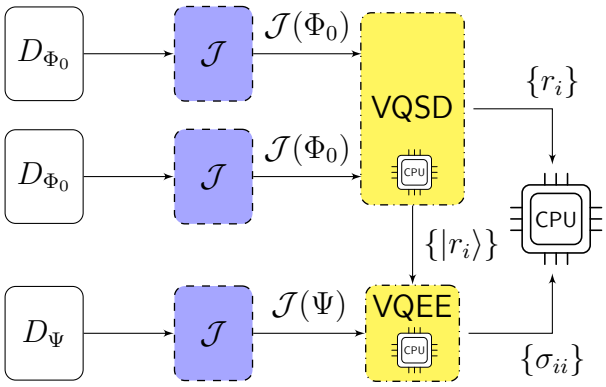


Figure 5.2: The VQFE certification algorithm.

an ideal device, denoted by D_{Φ_0} which in turn characterized by a quantum channel Φ_0, operating on the space of n qubits. Moreover, we have the second device, D_Ψ, which is claimed to be indistinguishable compared with D_{Φ_0}.

Furthermore, as the start-up specializes in quantum technology, the board decided that the certification procedures should also benefit from the quantum advantage. Such a decision yields two benefits. Firstly, it supports the claims concerning the ubiquitous applications of quantum computing. Secondly, it provides an opportunity to develop a unique certification service that can be offered to other quantum start-ups [108].

5.2.2 The algorithm

Apart from the process described through Fig. (5.1) we introduce an alternative certification approach based on variational quantum fidelity estimation [85]. The primary goal of the algorithm is to find the truncated fidelity of $\mathcal{J}(\Psi)$ on the basis of m largest eigenvalues of $\mathcal{J}(\Phi_0)$. In Fig. (5.2) we illustrate the VQFE-based certification scheme. In this scheme to certify a quantum device Ψ against the ideal device Φ_0, the VQFE procedure is used. The input of the procedure is given in the form of physical devices, D_{Φ_0} and D_Ψ, implementing channels Φ_0 and Ψ respectively. The first step is to apply Choi-Jamiołkowski isomorphism to obtain dynamical matrices for the input devices. Next, two copies of the ideal device $(\mathcal{J}(\Phi_0))$ are used as an input for the variational quantum state diagonalization (VQSD) procedure, which is briefly discussed in the Sec. 3.2.1. At the same time, $\mathcal{J}(\Psi)$ is processed by a quantum-classical algorithm to obtain its matrix elements in the eigenbasis of $\mathcal{J}(\Phi_0)$. Finally, classical processing of the obtained eigenvalues is used to calculate the approximation of the fidelity between quantum operations. Note that in this procedure, the CPU part is utilized at several steps – as a part of VQSD used for calculating eigenvalues r_i of $\mathcal{J}(\Phi_0)$ and matrix elements σ_{ij} of $\mathcal{J}(\Psi)$.

The yellow blocks in Fig. (5.2) indicate hybrid quantum-classical sub-procedures.

In a nutshell, the variational quantum fidelity-based certification procedure consists of the following steps.

- First we prepare two copies of $\mathcal{J}(\Phi_0)$ to process in the VQSD algorithm and one copy of $\mathcal{J}(\Psi)$.

- The two copies of the ideal device are utilized to diagonalize $\mathcal{J}(\Phi_0)$. After this process, we get the m-largest eigenvalues $\{r_i\}$, which can be subsequently stored on a classical CPU, and the eigenvectors of $\mathcal{J}(\Phi_0)$, which will be useful in the upcoming step.

- We now make use of the VQFE procedure with the $\mathcal{J}(\Psi)$ and provide the eigenvectors of the ideal device obtained from the previous VQSD sub-procedure to obtain matrix elements $\sigma_{ii} = \langle r_i|\mathcal{J}(\Psi)|r_j\rangle$ in the eigenbasis of $\mathcal{J}(\Phi_0)$, if the cost function in VQSD process is non-zero then we get $\sigma'_{ii} = \langle r'_i|\mathcal{J}(\Psi)|r'_j\rangle$, where r'_i is the inferred eigenbasis.

- The resulting matrix elements of $\mathcal{J}(\Psi)$ in the eigenbasis of $\mathcal{J}(\Phi_0)$, and eigenvalues of $\mathcal{J}(\Phi_0)$ are used to calculate truncated fidelity bounds according to Eq. (5.6). To obtain these bounds one needs to first compute the RHS of Eq. (5.7).

$$||\sqrt{\rho_m}\sqrt{\sigma_m^\rho}|| = \mathrm{tr}\sqrt{\sum_{i,j} T_{i,j}|r_i\rangle\langle r_j|}, \tag{5.8}$$

  where $T_{i,j}$ is a matrix whose dimension is dependent on the number of largest eigenvalues we can retrieve from the VQSD process. If we have m largest eigenvalues then $T_{i,j}$ is if $m \times m$ which elements are computed by

$$T_{i,j} = \sqrt{r_i r_j}\langle r_i|\mathcal{J}(\Psi)|r_j\rangle, \quad \text{such that } \; T \geq 0. \tag{5.9}$$

  Explicitly the truncated fidelity bounds are computed as follows

$$\begin{aligned} F_*(\rho_m, \sigma_m^\rho) &= \sum_i \sqrt{\lambda_i} + \sqrt{\left(1 - \sum_i r_i\right)\left(1 - \sum_i \sigma_{ii}\right)}, \\ F(\rho_m, \sigma_m^\rho) &= \sum_i \sqrt{\lambda_i}, \end{aligned} \tag{5.10}$$

  with λ_i are the eigenvalues of T where $i = 1, 2, \ldots, m$.

5.2.3 Noise models

This section briefly describes a class of noise models that we utilize to investigate the variational device certification process. The noise models are constructed using the Kraus operator representation [114]. For a brief introduction to quantum channels, we refer the reader to Appendix A.

Depolarizing noise For a single *depolarized* qubit with probability γ, the term *depolarized* means that the single qubit state is replaced by a completely mixed state i.e. $\mathbb{1}/2$ and with $(1-\gamma)$ probability the qubit is completely left untouched. Hence the state of the system after getting affected by the noise is

$$\Delta_\gamma = (1-\gamma)\rho + \frac{\gamma}{2}\mathbb{1}. \tag{5.11}$$

The circuit model of simulating depolarizing noise contains three qubits where one of the qubits contains the input quantum state and the remaining two lines are an environment to simulate the channel.

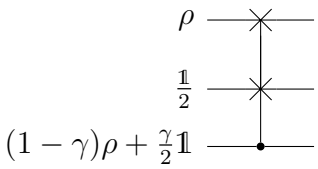


Figure 5.3: Illustration of a circuit that simulates depolarizing noise.

In the Fig. (5.3) the main idea behind the circuit is that the controlled qubit is the mixture of $|0\rangle$ with $(1-\gamma)$ probability and state $|1\rangle$ with γ probability and this decides whether or not the state $\mathbb{1}/2$ is swapped into the first qubit.

Amplitude damping noise The amplitude-damping channels lead to a decay of energy from an excited state to the ground state depending on the probability γ. Hence, the channel's action on a state is given as

$$A_\gamma = \mathcal{K}_0 \rho \mathcal{K}_0^\dagger + \mathcal{K}_1 \rho \mathcal{K}_1^\dagger, \tag{5.12}$$

where

$$\mathcal{K}_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \quad \mathcal{K}_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}. \tag{5.13}$$

In Fig. (5.4) we illustrate the circuit to simulate depolarizing noise.

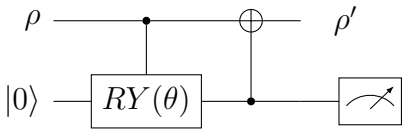


Figure 5.4: Illustration of a circuit that simulates amplitude damping noise.

The intuition behind the circuit is. Let us consider the ρ state is $a|0\rangle + b|1\rangle$ if the probability corresponding to $|1\rangle$ is higher then it is more probable that the ground state of the environment will be $|1\rangle$, which activates the `CX` gate and makes $\rho' = a|1\rangle + b|0\rangle$. So depending on the time the channel is applied, it is possible that the information corresponding to the qubit can be nearly completely dissipated.

Random X noise In the case of the random X quantum noise model the effect of the noise can be described as follows

$$R_\gamma = \gamma X + (1 - \gamma)\mathbb{1}, \tag{5.14}$$

where with probability γ we apply X-gate and with $(1 - \gamma)$ no gate is applied.

It should be noted that unlike the previous two noise models, which directly affect the state of a quantum system, random X noise is a gate-based noise model where during the construction of the ansatz for variation algorithms after each successive gate with γ probability, an X-gate might be applied depending on the probability distribution.

Real device noise By utilizing the `IBMQ.get_privder`(*device*), we can access the IBM Quantum real device backends, which simulate the exact noise model for that particular IBMQ device. During noisy simulation, we use the noise models of `IBMQ_lima` and `IBMQ_manila`. A tabular representation of various noise amplitudes is provided in Tab. 5.1 and Tab. 5.2 for `IBMQ_lima` and `IBMQ_manila` respectively.

Qubit	T1 (us)	T2 (us)	1q gate error	EX error
0	87.497	195.117	3.769e-4	0_1:0.00657
1	113.244	115.320	4.574e-4	1_0:0.00657
2	111.164	134.323	3.521e-4	2_1:0.00657
3	98.681	81.855	2.283e-4	3_4:0.0143
4	23.512	26.554	7.011e-4	4_3:0.0143

Table 5.1: Parameters of various noises in `IBMQ_lima` device.

Qubit	T1 (us)	T2 (us)	1q gate error	CX error
0	148.164	56.529	3.113e-4	0_1:0.00813
1	308.163	80.769	2.217e-4	1_2:0.00892
2	94.381	21.952	2.149e-4	2_3:0.00716
3	141.151	73.101	2.464e-4	3_4:0.00744
4	91.733	41.979	4.804e-4	4_3:0.00744

Table 5.2: Representation of various noises in `IBMQ_manila` device.

It should be noted that in the tables the *1q gate error* decomposes in the error in $\mathbb{1}$, SX (square root of X gate), and the X gates because the basis gates for `IBMQ_manila` and `IBMQ_manila` are `CX`, $\mathbb{1}$, `RZ`, `SX`, and `X` so it decomposes any gate into the following basis gates and apply noise.

5.2.4 Error quantification

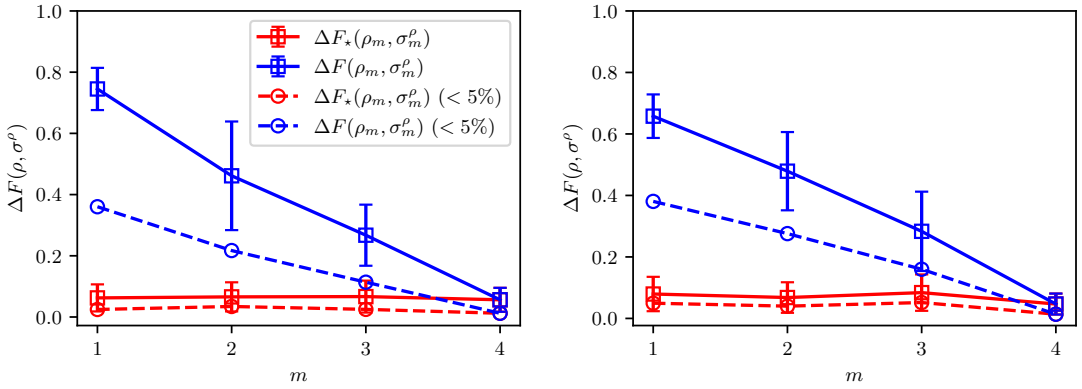
We quantify the error in the VQFE-based certification process calculating the difference in the truncated fidelity and the true fidelity defined as

$$\Delta F(\rho, \sigma^{\rho}) = F(\rho_m, \sigma_m^{\rho}) - F(\rho, \sigma). \tag{5.15}$$

Throughout this chapter, we use the above quantifier mentioned in 5.15 if not stated otherwise.

5.3 Results

In this section, we demonstrate the performance of the VQFE-based device certification algorithm where we consider (1) random 1-qubit quantum channels and then we scale up the system to consider (2) two-qubit quantum channels.



(a) `CZ` as entangler.

(b) `CX` as entangler.

Figure 5.5: Error in fidelity estimation with respect to m. The results were obtained by taking an average over 1000 random 1-qubit quantum channels ($n = 1$) of rank 4, using IBM quantum computer simulator from `Qiskit Aer` package. Here $\Delta F(\rho, \sigma^\rho) = F(\rho_m, \sigma_m^\rho) - F(\rho, \sigma)$. The dashed lines depict the average over channels with less than 5% error in the estimation of fidelity. The source code for the implementation can be obtained from [84].

In the case of a one-qubit channel, we do a rigorous investigation of the certification algorithm in the presence and absence of noise. While for two-qubit we investigate the noiseless scenario briefly.

5.3.1 One-qubit quantum channel

Noiseless scenario The results for the 1-qubit random quantum channels are illustrated in Fig. (5.5). We sample the quantum channels from Haar distribution using the `random_quantum_channel` module of `qiskit.quantum_info`.

The results are averaged over 1000 random quantum channels. For the purpose of illustration, we explicitly showcase the case where the ansatz contains `CZ` and `CX` as entangler in Fig. (5.5a) and in Fig. (5.5b) respectively. This helps us to note that for random 1-qubit quantum channels, it is better to consider `CX` gates in the ansatz than other entangling gates. Additionally, in both cases, we observe that the introduced certification procedure provides a very good approximation to fidelity for low-rank quantum channels. At the same time, SSFB certification can provide a useful lower bound for fidelity between operations. However, the upper bound obtained in the SSFB case is unsuitable for providing a viable approximation of fidelity.

We also observe that the approximation given by the certification procedure can be significantly improved by increasing the number of eigenvalues estimated in

the variational diagonalization subroutine. This is evident in the

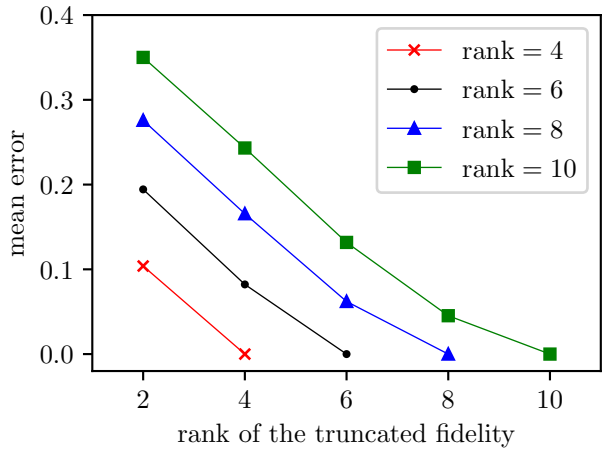


Figure 5.6: Mean error for the approximation of fidelity by the truncated fidelities. Values are plotted for random quantum operations as a function of the rank of the truncated fidelity. Each combination of color and shape corresponds to density matrices with a fixed rank. Each point was obtained by averaging the difference between the fidelity and the truncated fidelity on the sample of 10^5 pairs of random dynamical matrices.

In Fig. (5.6), where the dependency of the mean approximation error of the truncated fidelity is plotted for dynamical matrices with different ranks. As one can see, the bound obtained using the truncated fidelity can be easily tightened. Moreover, the mean for the given rank of the truncated fidelity decreases with the increasing rank of random dynamical matrices. The theoretical interpretation of this observation can be clearly seen through the Eq. (5.10) where the bounds in TFB reply on the eigenvalues of the matrix T. In Eq. (5.9) we also see that the primary building blocks of the T matrix are the eigenvalues of the quantum channel that is to be diagonalized in the VQSD subroutine. Hence higher the rank of the quantum channel the more eigenvalues the VQSD subroutine can approximate which in turn gives a better approximation to the eigenvalues of the T matrix, resulting in tighter truncated fidelity bound.

The exact ansatz construction that we use is a Hardware Efficient Ansatz of the form

$$U(\vec{\theta}) = U_{\text{ent}} \times \prod_{i=1}^{N} \texttt{RY}(\theta_i)^{\otimes i} \texttt{RZ}(\theta_i)^{\otimes i}, \tag{5.16}$$

where n is the size of the quantum channel.

Noisy scenario In this part, we illustrate through Fig. (5.7) the effect of depolarizing, amplitude damping, and random X noise.

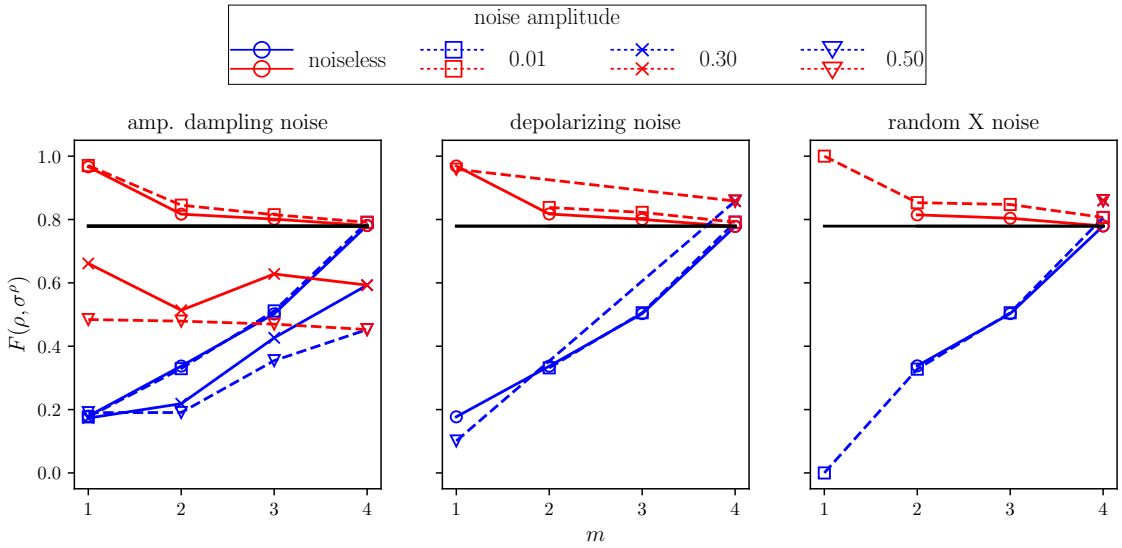


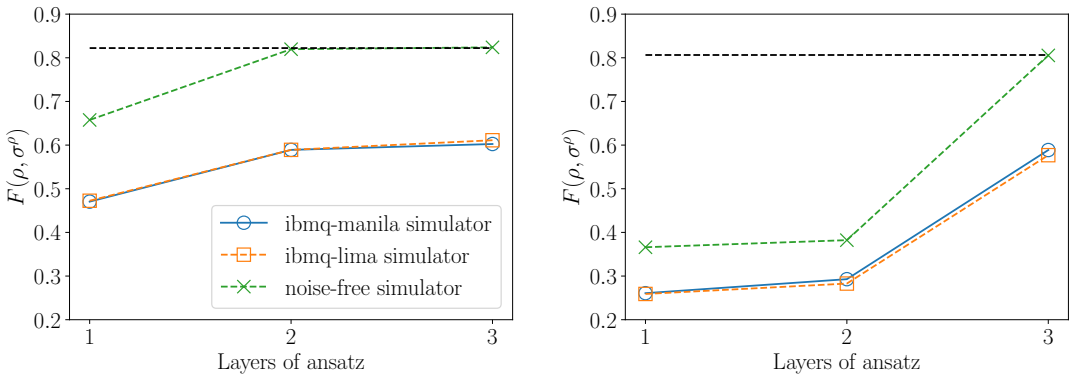
Figure 5.7: Effect of noise in the estimation of truncated fidelity bound under amplitude damping, depolarizing, and random X noise. It can be seen that the effect of amplitude damping noise is more on fidelity estimation when we increase the noise above 30%. Additionally, for amplitude damping noise when the noise amplitude is within 10%, we do not observe any notable changes in the fidelity estimation, but in retrieving lower ranks, it performs poorly.

A brief discussion of the three kinds of noise models is given in Sec. 5.2.3. Through our illustration, we see that the fidelity bound fails to converge to the true value as the noise amplitude increases more than 10%. The depolarizing and random X noise elevates the fidelity at full rank compared to the true value, while due to the effect of amplitude damping noise, the estimated fidelity decreases. It is due to the fact that the amplitude damping noise reduces the value of the eigenvalues that are collected during the channel diagonalization process. As we already saw the fidelity calculation directly depends on the eigenvalues i.e. more exact the eigenvalues the better the estimation of fidelity hence a decrease in eigenvalue decreases the estimated fidelity. When the amplitude of noise increases more than 10%, the deviation of fidelity estimation from its true value becomes more prominent.

We can also notice that the higher the noise value the higher the error in the fidelity estimation due to failure in retrieval of eigenvalue for lower rank. This is more prominent in the case of random X noise where we only retrieve the fidelity eigenvalue corresponding to the highest rank i.e. rank 4.

In Fig.(5.8), we depict the results for fidelity estimation after running the algorithm in `Qiskit Aer` with real device backends provided by IBM. In the presence of noise, we expect a degradation of the performance of the algorithm which is evident from the figure. The fidelity achieved by using `CX` and `CZ` for the entangling gate is similar at the 3rd layer but the `CZ` gate helps to obtain a very close to optimal fidelity (which is less than the true fidelity) in just one layer. Whereas the ansatz with `CX` keeps on improving at each step and finally achieves a similar fidelity as the ansatz with `CZ` at the 3rd layer. This helps us infer that although `CX` gate represents better learnability, hence beneficial for optimization in a noiseless scenario. But in a noisy case, it is profitable to use `CZ` to achieve higher accuracy with very few gates and depth.

As the noise model for both devices is similar hence we observe similar characteristics in the variation of fidelity with respect to layers for the two kinds of noise. Additionally, as the amplitude-damping noise is more prominent in the real device we see a decrease in the fidelity value. As the noise model for both devices is similar hence we observe similar characteristics in the variation of fidelity with respect to layers for the two kinds of noise. Additionally, as the amplitude-damping noise is more prominent in the real device we see a decrease in the fidelity value.



(a) `CZ` as entangler.

(b) `CX` as entangler.

Figure 5.8: The convergence of the truncated fidelity for a single random one-qubit (rank 4) channel in IBM's noise-free and `AerSimulator` from Qiskit package with real device backends. The black dashed line depicts the true value of fidelity. See [84] for the implementation details.

noteI added labels in Fig. (5.8a) and Fig. (5.8b)

5.3.2 Two-qubit quantum channel

In this section, we show the performance of the VQFE-based device certification with the scaling of the quantum channel. The result is illustrated in Fig. (5.9) where we see that the average error in the estimation of the fidelity for one-qubit quantum channels is around 15% whereas 100 channels are below 5% error, which is lower compared to the one-qubit case where we saw 500 channels goes below 5% error.

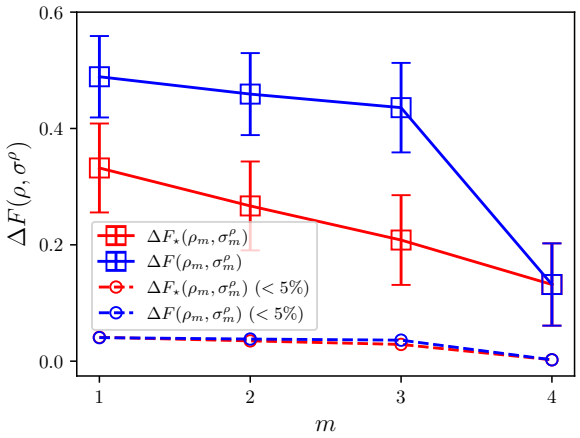


Figure 5.9: The average error in fidelity estimation with respect to m, obtained by averaging over 1000 random two-qubit quantum channels ($n = 2$) of rank 4, using IBM quantum computer simulator from `Qiskit Aer` package.

5.4 RL assisted variational channel certification

In the previous chapter in Sec. 3.4, we elaborately discuss how we can harness a reinforcement learning agent to efficiently diagonalize a quantum state. From our elaborated investigation, it was evident that a carefully chosen encoding scheme for the quantum circuit, a dense reward function, and an ϵ-greedy policy for the agent can help us achieve a diagonalizing unitary with very low depth and with a small number of gates. Additionally, with the efficient circuit, we were able to achieve an accuracy better than the existing algorithm.

Using the Jamiołkowski-Choi isomorphism in Eq. (5.1) we can extend the applicability of RL-VQSD to RL-based quantum device certification. Finding the exact eigendecomposition of a quantum channel is an indispensable part of the VQFE-based certification process. A near-exact approximation of the eigenvalues and eigenvectors of the channel led us to the efficient certification of quantum devices. As we saw the RL can enhance the performance of the diagonalization and

give us a better approximation to the eigendecomposition. This in turn enhances the VQFE-based device certification process.

Meanwhile, till now the ansatz we were using in the VQFE-based certification process were all fixed structures LHEA that contain at least 4-8 parameters per layer and its depth grows rapidly as we increase the layers of the ansatz. This has two issues (1) A large number of parameters can cause trainability issues as we scale up the size of the quantum channel and (2) due to high depth and gate various kinds of quantum noise can impact the accuracy of the certification process. An RL-based certification will reduce the number of parameters in the quantum circuit by minimizing the number of gates and the depth of the circuit. On the other hand, the minimized ansatz proposed by an RL-agent will be more noise-resilient than the fixed structure LHEA.

5.5 Conclusion

In this chapter, as an application of the RL-VQSD method, discussed in the previous chapter, we present a novel approach to certify quantum devices based on the variational quantum fidelity estimation method. The introduced algorithm is primarily based on the combination of Jamiołkowski-Choi isomorphism and variational quantum state diagonalization method. The algorithm is suitable for real-life cases when one needs to provide a convincing argument supporting their claim about a quantum device, keeping in mind the certified quantum device operates on a small number of qubits. This is exactly the case for the NISQ era.

The intrinsic benefit of the VQFE-based channel certification comes from the principle of variational paradigm [109]. The variational structure of quantum circuits provides us with the advantage of harnessing the strengths of a given quantum architecture. Which benefits avoiding some classes of quantum noises [101], making it more appealing for the NISQ devices.

This is more apparent during the noisy simulations of the VQFE-based certification where we show that amplitude-damping, depolarizing, and random X noise with 1% intensity do not affect the performance of the algorithm. We also simulate our protocol on a real quantum device and show that the performance of the certification process decreases drastically compared to the noiseless scenario.

5.6 Takeways

This chapter introduced a protocol for quantum channel discrimination based on the variational quantum algorithm for fidelity estimation.

- **Choi-Jamiołkowski isomorphism driven quantum device certification** In this chapter, we utilize the variational quantum state diagonalization (that is discussed in the previous chapter) with the Choi-Jamiołkowski isomorphism as described in Eq. (5.1) to certify quantum channels. Where the approximation of the similarity measure, i.e. fidelity (see Eq. (5.2)), plays a very important role. During the construction of the certification scheme, we see that the state-of-the-art fidelity bounds in Eq. (5.4) are replaced by the truncated fidelity bounds as given in Eq. (5.6) and in Eq. (5.7). Hence, for two distinct quantum channels, if the truncated fidelity bounds converge to 1, we can say that the channels are the same. The algorithm is elaborated in the Sec. 5.2.2.

- **Better the approximation of VQSD, the better certification** In the Fig. (5.6), we illustrate the dependency of the approximation error in the certification process with respect to the rank of dynamical matrices of quantum channels. As it can be seen from the illustration for 4, 6, 8, and 10 rank dynamical matrices the mean error in truncated fidelity decreases gradually and becomes tighter as we increase the rank in the dynamical matrix. This indicates that during the channel diagonalization process, if we can retrieve all the eigenvalues and eigenvectors of the channel, then the certification efficiency maximizes.

- **Little effect of noise and scaling** In the Fig. (5.5) and **5.9** we present the performance of the VQFE-based certification process for one- and two-qubit quantum channels averaged over 1000 random quantum channels. It can be seen that the error in fidelity does not get affected by a lot as we scale up the size of the quantum channel. But the number of states with less than 5% error decreases from 480 for one-qubit to 99 in the case of two-qubit.

  On the other hand in Fig. (5.7) we show that depolarizing, amplitude-damping, and random X noise do not affect the performance of the certification scheme if the noise value is below 10% but due to the decoherence error and connectivity constraints, the performance of the certification process underwhelms compared to a single type of noise scenario.

- **RL assisted channel certification** In the concluding section of the chapter, we briefly describe how a reinforcement learning agent can be utilized to enhance the performance of the VQFE-based certification process. To do this we propose to replace the fixed-structured quantum circuit given in Eq. (5.16) by a variable ansatz, where each element (i.e. quantum gate and we refer to Sec. 3.3.2 for details) is decided by a policy and a dense reward function (see Sec. 3.3.2 for details). This promises to give us a short-depth ansatz containing comparatively very few one and two-qubit gates.

Due to the compactness of the RL-agent proposed ansatz we guarantee to observe better performance of the certification scheme in the noisy scenario and also while running the scheme on the real quantum device.

Chapter 6

Discussions

Current quantum hardware is limited in terms of the number of qubits and connectivity among them. Due to very small decoherence time, the qubits are noise-prone and their performance degrades rapidly as we evolve a quantum state through a unitary. Hence it is important to find small quantum circuits that can solve a problem with low error. To address this, in this thesis, we discuss a general framework that automates the search for efficient quantum circuits for variational quantum algorithms (VQAs) based on reinforcement learning (RL) techniques. A lot of research is dedicated to providing a definitive structure of the variational quantum circuit in VQAs. Due to the mystery behind the structure of these variations circuits, they are famously known as *ansatz*. The problem of automating the search for an efficient ansatz is known as the quantum architecture (QAS) problem.

Before discussing the framework, we give a brief introduction to quantum states, gates, and quantum channels in the Appendix A which is followed by a brief discussion of variational quantum algorithms (VQAs) in Chapter 2.1. The description of VQAs includes a brief introduction to two of the most crucial subroutines of VQAs (1) the construction of cost function and how its landscape varies with the number of qubits in Sec. 2.1.1, which is followed by a brief discussion of (2) the problem inspired and problem agnostic ways of constructing an ansatz in Sec. 2.1.2. Afterwards, in Sec. 2.2 we provide an introduction to reinforcement learning. Keeping the vastness of RL we keep our discussion compact and introduce the concepts relevant to the thesis. Finally, in the Sec. 2.12 we introduce the necessary ingredients for building the RL-assisted quantum architecture search framework that we use in the following chapters of the thesis.

In the framework, the *environment* is defined by the quantum-classical optimization loop of VQAs, the *RL-state* is represented through the ansatz structure, the *reward function* is itself a function of the cost function (that encodes the problem) and the *action space* is composed of quantum gates. Utilizing this framework we

consider two problems based on their relevancy in quantum physics, quantum chemistry, and condensed matter physics. These two problems are variational quantum state diagonalization (VQSD) and variational quantum eigensolver (VQE).

As the first application of the framework, in the Chapter 3 we consider the reinforcement learning assisted variational quantum state diagonalization (VQSD) which we term as RL-VQSD [82] in a noiseless scenario. The algorithm focuses on identifying the unitary rotation under which a given quantum state becomes diagonal in the computational basis. We first benchmark the performance of linear hardware efficient ansatz (LHEA) in diagonalizing a two- and three-qubit quantum state. Later we use the RL-based diagonalization method to find the eigenvalues of two-, three-, and four-qubit quantum states. In the case of two-qubit states, we sample arbitrary quantum states from the Haar distribution. Whereas, for three-, and four-qubit cases we choose the reduced ground state of a six-, and eight-qubit Heisenberg model respectively, and diagonalize it. Finally, to demonstrate the hardness of the problem, we run a random-agent method where the RL-agent does not choose the action at a step from the cumulative reward collected from the previous steps but randomly.

In the next application, in Chapter 4, we use our framework to construct a curriculum reinforcement learning-based variational quantum eigensolver (VQE) CRL-VQE based on ref [119] algorithm where we find the ground state of molecules in a noiseless and noisy scenario. In the noiseless scenario, we consider a `LiH` molecule of four- and six-qubit and an eight-qubit $\mathtt{H_2O}$. Next, we compare the performance of the CRL-VQE with the previously proposed QAS algorithms such as RL-VQE in ref. [116] and quantumDARTS in ref [165]. The majority of research in QAS is focused on a noiseless scenario and the impact of noise on the QAS remains inadequately explored. So in this chapter, we extend the results in realistic noisy scenarios as well where the noise models are based on IBM quantum devices such as `ibmq_mumbai` (with all-to-all connectivity) and `ibmq_ourense` (with constrained connectivity). Under these noise models, we solve the two-, three-, and four-qubit $\mathtt{H_2}$ molecule and the four-qubit `LiH` molecule.

While solving these two problems we consider the following settings:

1. **The RL methods:** While dealing with VQSD the RL method we use is a *vanilla curriculum* approach where the user needs to provide a predefined threshold. In the ideal case for the VQSD, the cost function should approach 0, but in the realistic scenario, it is important to define a predefined threshold (ζ) arbitrarily and completely depends on the number of qubits. But while dealing with ground state finding problems using VQE, the exact value of the ground state of molecules is not known; hence, the idea of fixing a predefined threshold is out of scope. That is why we utilize a *feedback-driven curriculum RL* (namely CRL) in Sec. 4, which is independent of the prior knowledge

regarding the true ground state and does not impose any specific constraints on the initial threshold value.

2. **The RL-state representation:** The RL-state is defined through a tensor-based binary (TBE) encoding scheme elaborated in **Section 3.3.1** in dealing with both the VQSD and VQE. In **Section 4.3.1**, we also show that for VQE problems, the TBE outperforms the previously proposed encodings for similar problems [116].

3. **The RL-action space encoding:** The action space is encoded in a one-hot-encoding manner where each action represents a quantum gate defined by a list of four elements. The complete encoding scheme is elaborated in Sec. 3.3.2 with an example. The encoding of the action space remains invariant in the VQSD and VQE. Meanwhile,

4. **The reward function construction:** For the VQSD algorithm the reward function is defined densely as given in Eq. (3.10). Whereas, for VQE we utilize a sparse reward function Eq. (4.3). In the Sec. 4.3.3 we show that for the VQE problem both the reward functions provide similar results in terms of ansatz depth, parameters, and accuracy but as the number of successful episodes using Eq. (4.3) is higher than Eq. (3.10) the former is more reliable in finding smaller ansatz.

5. **The action space pruning:** Throughout the thesis we call the action space pruning method as *illegal actions*. When adding a unitary to a qubit at a certain step s, if we append the same unitary to the same qubit in step $s+1$, these two operations negate, and the cumulative result is an identity matrix or an idling operation. To restrict redundant such operations and enhance the RL-agent's exploration efficiency while dealing with VQE we utilize the *illegal actions* technique, elaborated in Sec. 4.3.2 with an example on four-qubits.

6. **The random halting technique:** This technique is specifically designed to deal with the noise in quantum devices. As it is discussed in Sec. 4.5, whenever noise is applied to a quantum circuit, a CPTP channel is applied to the circuit, which not only reduces the performance of the circuit to achieve a task but increases the computation time by many times compared to the noiseless scenario. Hence to keep up with this, we make the total number of steps in an episode a variable by sampling it from a negative binomial distribution. This method is particularly used while finding the ground state of molecules using the VQE algorithm. In Fig. (4.5) we show that the *random halting* searches the optimal ansatz 3 times faster.

Finally, in the Chapter 5 we discuss a novel quantum channel certification [85] process based on the VQSD algorithm. The algorithm takes two devices as input – the standard device with the operational capacity already confirmed, and the device for which its conformation with the standard device to be confirmed. We benchmark the variational quantum channel algorithm in the noiseless scenario for one-, and two-qubit Haar random quantum channels. Next, we investigate how the algorithm is influenced under amplitude damping, depolarizing, and random `X` noise models. Later on, we benchmark the performance of the performance of the algorithm under real device noise imported from `ibmq_manila` and `ibmq_lima`. Finally, we discuss as a future work the possibility of constructing an RL-based quantum channel certification process.

Chapter 7

Conclusions

In this thesis, we introduce a simple yet effective infrastructure for a reinforcement learning-based quantum architecture search algorithm. To show the performance of the thesis, we propose an RL-VQSD [82] and CRL-VQE [7] algorithm that is built upon the framework introduced in this thesis. We solve these problems in the absence and the presence of noise and gather valuable insights. The results are summarized through the following points:

1. **In noiseless scenario:** For both the RL-VQSD and the CRL-VQE, the RL-agent provides us with an ansatz (that we call RL-ansatz), which contains a smaller number of parameters and depth while achieving lower error in cost function evaluation compared to state-of-the-art methods. For example, in the Chapter 3 we compare the performance of the LHEA ansatz proposed in state-of-the-art VQSD algorithm with the RL-anatz provided by the agent in RL-VQSD. We show that the RL-ansatz is not only of smaller depth and number of gates, but it also helps us achieve a lower error in eigenvalue estimation which is represented through the Fig. (3.10). Meanwhile, in the Chater 4, we compare the performance of our algorithm with previously proposed learning-based methods for quantum chemistry problems such as the RL-VQE [116], quantumDARTS [165] and the net-based QAS [44] algorithms. Our results in Tab. 4.3 show that the CRL-VQE algorithm outperforms the RL-VQE and the quantumDARTS and provides an ansatz with a smaller number of gates, depth, and parameters also keeping a very high accuracy in ground state energy estimation proving the **Hypothesis 1**.

   On the other hand,

2. **In noise scenario:** The performance of the CRL-VQE algorithm is investigated under real device noise. Where the noises are modeled from IBM devices such as `ibmq_mumbai` (with all-to-all connectivity) and `ibmq_ourense`

(with constrained connectivity). Through Tab. 4.4, We show that under the framework of the CRL-VQE in the presence of such noise and constrained connectivity, the RL-agent solves various quantum chemistry problems while utilizing a novel action space pruning (which we call *illegal actions*) and variable episode length (calling it *random halting*) techniques with curriculum reinforcement learning. This proves the **Hypothesis 2** introduced in the introduction section.

In the final chapter of the thesis, we discuss a variational quantum channel certification algorithm that uses the VQSD algorithm as a subroutine. The results on the Fig. (5.5) and Fig. (5.9) benchmark the performance of the algorithm for one- and two-qubit quantum channels. Whereas in Fig. (5.7) and Fig. (5.8) we show the performance of the algorithm under various simulated noise models. In the following, we discuss broadly first the strengths and advantages of the discussed framework which is followed by an elaboration of the limitations and accompanying future work.

7.1 Strengths and Advantages

In this section, we outline the merits of the introduced framework, specifically emphasizing the benefits of utilizing the existing infrastructure as a foundation for future research.

A straightforward framework of QAS for VQAs In this thesis, we introduce a very simple and efficiently implementable framework for quantum architecture search in VQAs. The strength of the framework lies in the fact that it can perform efficiently in the absence and presence of real device noise. The primary advantage of the framework is that it can be readily adapted to address a wide range of VQAs in deploying quantum architecture search methods.

Binary depth-based encoding scheme for ansatz The quantum circuit encoding scheme that is presented in the scope of a subroutine of RL scales quadratically with the number of qubits. In this scheme, each depth of the circuit is encoded in a block of dimension $((N + 3) \times N)$, where the presence or absence of a gate is represented by either 1 or 0 respectively. The strength of this approach becomes evident when contrasted with the previously suggested integer gate-based encoding [116], surpassing the performance of the prior encoding scheme. The advantage of the presented encoding is the fact that it can be easily modified (based on connectivity among qubits and the limitation of the connectivity in two-qubit gates) and adapted to any quantum architecture search method.

Action-space pruning The *illegal actions* is an add-on technique presented in the thesis and is a very simple yet effective way to narrow the action space for the RL-ansatz. Which optimizes the search in the action space and provides a significant advantage in minimizing the time per episode while dealing with our framework. The primary strength of the technique lies in its ability to be easily toggled on or off, and seamlessly adapted to any quantum architecture search methods without difficulty.

Accelerating the discovery of efficient ansatz Noise in quantum devices increases the simulation longer. To cope with this increasing computational time, in this thesis, we introduce *random halting* scheme and we show that using this method we can search an efficient ansatz three times faster, showcasing its strength. The advantage of the technique lies in the fact it can be easily adaptable to any quantum architecture search method and real quantum hardware.

Curriculum reinforcement learning In VQAs, it is not possible to have prior knowledge of the solution. In the vanilla curriculum of RL, it is required to define prior a threshold value for the cost function. In the case of some VQAs, the threshold can be arbitrary (as can be seen in the RL-VQSD) but for quantum chemistry problems, it is necessary to set the threshold in such a way that the ansatz proposed by the agent can provide the ground state energy. In this thesis, we utilize curriculum reinforcement learning. The strength and advantage of this RL is that, without having prior knowledge of the ground energy, the algorithm iteratively shifts the threshold value after each episode and leads the agent to find the ground state energy.

7.2 Limitations and Future Work

Progress in RL methods Despite the promise of RL methods, they encounter challenges concerning sample efficiency, stability, and sensitivity regarding the learning ability of the agent. It is crucial to acknowledge these aspects in the scope of the constantly evolving RL field and mitigate such limitations in future work.

Computational requirements The computational demands of training the agent are substantial, primarily in the presence of real device noise. This imposes challenges in both evaluating quantum circuits on a quantum computer and training the algorithm on classical devices. A careful investigation is needed to find more efficient computational strategies.

Scalability to complex problems As a follow-up to the discussion regarding the computational challenges, it is crucial to address the scalability of the proposed framework to larger and highly correlated quantum chemistry problems, and in case of diagonalizing many-body Hamiltonians or diverse noise models. The training of the RL-agent is done from scratch, which promotes the need for exploration to enhance scalability.

Real quantum hardware validation The thesis is constrained by the simulation of real device noise and its connectivity which lacks the lack of validation on real quantum hardware, primarily due to existing cost limitations. This can be seen as a motivation for future research endeavors.

Broadening the application scenario The application scenario of the proposed framework in the thesis is limited to just three possible problems that have an impact on quantum physics, chemistry, condensed matter physics, and quantum information theory. A straightforward future work in this line can be broadening the scope of this framework to additional applicable scenarios. For example, in variational quantum linear solver [22] the authors propose a fixed structure layered hardware efficient ansatz which contains a large number of one- and two-qubit gates. Our framework has the potential to automate the search for an efficient ansatz that is of shorter depth and gate. Following this motivation it is recently shown that using an Schrödinger-Heisenberg variational quantum algorithm [136] it is possible to scale find the expectation value of a complex chemistry or condense matter Hamiltonian with shallow ansatz. It is possible to automate the search for the Schrödinger and the Heisenberg circuits to further optimize the gates and depth of the overall ansatz.

Bibliography

[1] Circuits | Cirq | Google Quantum AI. https://quantumai.google/cirq/build/circuits. Accessed: 2024-01-16.

[2] D. S. Abrams and S. Lloyd. Simulation of many-body fermi systems on a universal quantum computer. *Physical Review Letters*, 79(13):2586, 1997. doi:10.1103/PhysRevLett.79.2586.

[3] D. S. Abrams and S. Lloyd. Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors. *Physical Review Letters*, 83(24):5162, 1999. doi:10.1103/PhysRevLett.83.5162.

[4] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernández, J. Carballo-Franquis, A. Chen, C.-F. Chen, et al. Qiskit: An open-source framework for quantum computing. Accessed on: 16/03/2023. URL: https://qiskit.org/.

[5] E. Alpaydin. *Introduction to machine learning*. MIT Press, 2020.

[6] P. G. Anastasiou, Y. Chen, N. J. Mayhall, E. Barnes, and S. E. Economou. Tetris-adapt-vqe: An adaptive algorithm that yields shallower, denser circuit ansätze. *arXiv*, 2022. doi:10.48550/arXiv.2209.10562.

[7] Anonymous. Curriculum reinforcement learning for quantum architecture search under hardware errors. In *Submitted to The Twelfth International Conference on Learning Representations*, 2023. under review. URL: https://openreview.net/forum?id=rINBD8jPoP.

[8] A. Arrasmith, M. Cerezo, P. Czarnik, L. Cincio, and P. J. Coles. Effect of barren plateaus on gradient-free optimization. *Quantum*, 5:558, 2021. doi:10.22331/q-2021-10-05-558.

[9] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. doi:10.1038/s41586-019-1666-5.

[10] B. Bakó, A. Glos, Ö. Salehi, and Z. Zimborás. Near-optimal circuit design for variational quantum optimization. *arXiv*, 2022. doi:arXiv:2209.03386.

[11] H. B. Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989. doi:10.1162/neco.1989.1.3.295.

[12] J. Bausch. Recurrent quantum neural networks. *Advances in neural information processing systems*, 33:1368–1379, 2020.

[13] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966. doi:10.2307/j.ctv1nxcw0f.

[14] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019. doi:10.1088/2058-9565/ab4eb5.

[15] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv*, 2013. doi:10.48550/arXiv.1308.3432.

[16] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, et al. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics*, 94(1):015004, 2022. doi:10.1103/revmodphys.94.015004.

[17] A. Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhyā: the indian journal of statistics*, pages 401–406, 1946.

[18] E. Bianchi, L. Hackl, M. Kieburg, M. Rigol, and L. Vidmar. Volume-law entanglement entropy of typical pure quantum states. *PRX Quantum*, 3(3):030201, 2022. doi:10.1103/prxquantum.3.030201.

[19] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595–600, 2018. doi:10.1038/s41567-018-0124-x.

[20] M. Born and R. Oppenheimer. Zur quantentheorie der molekeln. *Annalen der Physik*, 389(20):457–484, 1927. doi:10.1002/andp.19273892002.

[21] L. Botelho, A. Glos, A. Kundu, J. A. Miszczak, Ö. Salehi, and Z. Zimborás. Error mitigation for variational quantum algorithms through mid-circuit measurements. *Physical Review A*, 105(2):022441, 2022.

[22] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles. Variational quantum linear solver. *Quantum*, 7:1188, 2023.

[23] S. Bravyi, J. M. Gambetta, A. Mezzacapo, and K. Temme. Tapering off qubits to simulate fermionic hamiltonians. *arXiv*, 2017. doi:10.48550/arXiv.1701.08213.

[24] S. B. Bravyi and A. Y. Kitaev. Fermionic quantum computation. *Annals of Physics*, 298(1):210–226, 2002. doi:10.1006/aphy.2002.6254.

[25] H. Buhrman, R. Cleve, J. Watrous, and R. De Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16):167902, 2001. doi:10.1103/physrevlett.87.167902.

[26] C. Cao, Z. An, S.-Y. Hou, D. Zhou, and B. Zeng. Quantum imaginary time evolution steered by reinforcement learning. *Communications Physics*, 5(1):57, 2022. doi:10.1038/s42005-022-00837-y.

[27] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical reviews*, 119(19):10856–10915, 2019. doi:10.1021/acs.chemrev.8b00803.

[28] D. Castelvecchi. Quantum computers ready to leap out of the lab in 2017. *Nature*, 541(7635), 2017. doi:10.1038/541009a.

[29] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021. doi:10.1038/s42254-021-00348-9.

[30] M. Cerezo and P. J. Coles. Higher order derivatives of quantum neural networks with barren plateaus. *Quantum Science and Technology*, 6(3):035006, 2021. doi:10.1088/2058-9565/abf51a.

[31] M. Cerezo, A. Poremba, L. Cincio, and P. J. Coles. Variational quantum fidelity estimation. *Quantum*, 4:248, 2020. doi:10.22331/q-2020-03-26-248.

[32] M. Cerezo, K. Sharma, A. Arrasmith, and P. J. Coles. Variational quantum state eigensolver. *npj Quantum Information*, 8(1):113, 2022. doi:10.1038/s41534-022-00611-6.

[33] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature communications*, 12(1):1791, 2021. doi:10.1038/s41467-021-21728-w.

[34] R. Chen, Z. Song, X. Zhao, and X. Wang. Variational quantum algorithms for trace distance and fidelity estimation. *Quantum Science and Technology*, 7(1):015019, 2021. doi:10.1088/2058-9565/ac38ba.

[35] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang. Quantum long short-term memory. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8622–8626. IEEE, 2022. doi:10.1109/icassp43922.2022.9747369.

[36] M.-D. Choi. Completely positive linear maps on complex matrices. *Linear algebra and its applications*, 10(3):285–290, 1975. doi:10.1016/0024-3795(75)90075-0.

[37] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles. Learning the quantum algorithm for state overlap. *New Journal of Physics*, 20(11):113022, 2018. doi:10.1088/1367-2630/aae94a.

[38] H. W. CJC. Learning from delayed rewards (phd thesis). *University of Vambridge, England*, 1989.

[39] I. Cong, S. Choi, and M. D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019. doi:10.1038/s41567-019-0648-8.

[40] P. Cunningham, M. Cord, and S. J. Delany. Supervised learning. In *Machine learning techniques for multimedia: case studies on organization and retrieval*, pages 21–49. Springer, 2008.

[41] A. Das and B. K. Chakrabarti. Colloquium: Quantum annealing and analog quantum computation. *Reviews of Modern Physics*, 80(3):1061, 2008.

[42] P. Dayan, M. Sahani, and G. Deback. Unsupervised learning. *The MIT encyclopedia of the cognitive sciences*, pages 857–859, 1999. doi:10.1093/acprof:oso/9780198749783.003.0009.

[43] J. W. Demmel, O. A. Marques, B. N. Parlett, and C. Vömel. Performance and accuracy of lapack's symmetric tridiagonal eigensolvers. *SIAM Journal on Scientific Computing*, 30(3):1508–1526, 2008. doi:10.1137/070688778.

[44] Y. Du, T. Huang, S. You, M.-H. Hsieh, and D. Tao. Quantum circuit architecture search for variational quantum algorithms. *npj Quantum Information*, 8(1):62, 2022. doi:10.1038/s41534-022-00570-y.

[45] R. Duan, Y. Feng, and M. Ying. Perfect distinguishability of quantum operations. *Physical Review Letters*, 103(21):210501, 2009. doi:10.1103/physrevlett.103.210501.

[46] DWave corporation. DWave Quantum Inc. https://www.dwavesys.com/learn/quantum-computing/, 2023. Accessed on September 16, 2023.

[47] J. Eisert, M. Cramer, and M. B. Plenio. Colloquium: Area laws for the entanglement entropy. *Reviews of modern physics*, 82(1):277, 2010.

[48] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv*, 2014. doi:10.48550/arXiv.1411.4028.

[49] E. Fontana, M. Cerezo, A. Arrasmith, I. Rungger, and P. J. Coles. Non-trivial symmetries in quantum landscapes and their resilience to quantum noise. *Quantum*, 6:804, 2022. doi:10.22331/q-2022-09-15-804.

[50] T. Fösel, M. Y. Niu, F. Marquardt, and L. Li. Quantum circuit optimization with deep reinforcement learning, 2021. doi:10.48550/arXiv.2103.07585.

[51] B. T. Gard, L. Zhu, G. S. Barron, N. J. Mayhall, S. E. Economou, and E. Barnes. Efficient symmetry-preserving state preparation circuits for the variational quantum eigensolver algorithm. *npj Quantum Information*, 6(1):10, 2020. doi:10.1021/scimeetings.0c00536.

[52] Z. Ghahramani. Unsupervised learning. In *Summer school on machine learning*, pages 72–112. Springer, 2003. doi:10.1007/978-3-540-28650-9_5.

[53] A. Glos, A. Krawiec, and Z. Zimborás. Space-efficient binary optimization for variational quantum computing. *npj Quantum Information*, 8(1):39, 2022.

[54] D. Gottesman and I. Chuang. Quantum digital signatures. *arXiv*, 2001. doi:10.48550/arXiv.quant-ph/0105032.

[55] H. R. Grimsley, D. Claudino, S. E. Economou, E. Barnes, and N. J. Mayhall. Is the trotterized uccsd ansatz chemically well-defined? *Journal of chemical theory and computation*, 16(1):1–6, 2019. doi:10.1021/acs.jctc.9b01083.

[56] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature communications*, 10(1):3007, 2019. doi:10.1038/s41467-019-10988-2.

[57] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996. doi:10.1145/237814.237866.

[58] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 16(1):172–191, 1995. doi:10.1137/s0895479892241287.

[59] E. J. Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1948.

[60] A. W. Harrow and A. Montanaro. Quantum computational supremacy. *Nature*, 549(7671):203–209, 2017. doi:10.1038/nature23458.

[61] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, R. Tibshirani, and J. Friedman. Overview of supervised learning. *The elements of statistical learning: Data mining, inference, and prediction*, pages 9–41, 2009.

[62] M. B. Hastings. Classical and quantum bounded depth approximation algorithms. *arXiv*, 2019. doi:10.48550/arXiv.1905.07047.

[63] Z. He, X. Zhang, C. Chen, Z. Huang, Y. Zhou, and H. Situ. A gnn-based predictor for quantum architecture search. *Quantum Information Processing*, 22(2):128, 2023. doi:10.1007/s11128-023-03881-x.

[64] J. Helm and W. T. Strunz. Quantum decoherence of two qubits. *Physical Review A*, 80(4):042108, 2009. doi:10.1103/physreva.80.042108.

[65] B. Howard and R. Moss. The molecular hamiltonian: I. non-linear molecules. *Molecular Physics*, 19(4):433–450, 1970. doi:10.1080/00268977000101471.

[66] R. A. Howard. Dynamic programming and markov processes., 1960. doi:10.2307/1266484.

[67] IBM Corporation. IBM Quantum. https://quantum-computing.ibm.com/services/resources, 2023. Accessed on September 16, 2023.

[68] Intel Corporation. Honeywell Quantum Solutions. https://www.honeywell.com/us/en/company/quantum, 2023. Accessed on September 16, 2023.

[69] Intel Corporation. Intel Quantum Computing. https://www.intel.com/content/www/us/en/research/quantum-computing.html, 2023. Accessed on September 16, 2023.

[70] IonQ Corporation. IonQ. https://ionq.com/, 2023. Accessed on September 16, 2023.

[71] A. Jamiołkowski. Linear transformations which preserve trace and positive semidefiniteness of operators. *Reports on Mathematical Physics*, 3(4):275–278, 1972. doi:10.1016/0034-4877(72)90011-0.

[72] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax, 2016. doi:10.48550/arXiv.1611.01144.

[73] Z. Ji, Y. Feng, R. Duan, and M. Ying. Identification and distance measures of measurement apparatus. *Physical Review Letters*, 96(20):200401, 2006. doi:10.1103/physrevlett.96.200401.

[74] P. Jordan and E. P. Wigner. *Über das paulische äquivalenzverbot*. Springer, 1993. doi:10.1007/978-3-662-02781-3_9.

[75] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al. Model-based reinforcement learning for atari. *arXiv*, 2019. doi:10.48550/arXiv.1903.00374.

[76] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *nature*, 549(7671):242–246, 2017. doi:10.1038/nature23879.

[77] C. Kang, N. P. Bauman, S. Krishnamoorthy, and K. Kowalski. Optimized quantum phase estimation for simulating electronic states in various energy regimes. *Journal of Chemical Theory and Computation*, 18(11):6567–6576, 2022. doi:10.1021/acs.jctc.2c00577.

[78] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. doi:10.48550/arXiv.1412.6980.

[79] A. Y. Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv*, 1995. doi:10.48550/arXiv.quant-ph/9511026.

[80] C. Kokail, C. Maier, R. van Bijnen, T. Brydges, M. K. Joshi, P. Jurcevic, C. A. Muschik, P. Silvi, R. Blatt, C. F. Roos, et al. Self-verifying variational quantum simulation of lattice models. *Nature*, 569(7756):355–360, 2019. doi:10.1038/s41586-019-1177-4.

[81] S. Kotz, N. Balakrishnan, and N. L. Johnson. *Continuous multivariate distributions, Volume 1: Models and applications*, volume 1. John Wiley & Sons, 2004.

[82] A. Kundu, P. Bedełek, M. Ostaszewski, O. Danaci, Y. J. Patel, V. Dunjko, and J. Miszczak. Enhancing variational quantum state diagonalization using reinforcement learning techniques. *New Journal of Physics*, 2024. URL: http://iopscience.iop.org/article/10.1088/1367-2630/ad1b7f, doi:10.1088/1367-2630/ad1b7f.

[83] A. Kundu, L. Botelho, and A. Glos. Hamiltonian-oriented homotopy qaoa. *arXiv*, 2023. doi:arXiv:2301.13170.

[84] A. Kundu and J. A. Miszczak. Qiskit source code for variational fidelity estimation for quantum channels., 2021. doi:10.5281/zenodo.5804364.

[85] A. Kundu and J. A. Miszczak. Variational certification of quantum devices. *Quantum Science and Technology*, 7(4):045017, 2022. doi:10.1088/2058-9565/ac8572.

[86] E.-J. Kuo, Y.-L. L. Fang, and S. Y.-C. Chen. Quantum architecture search via deep reinforcement learning. *arXiv*, 2021. doi:10.48550/arXiv.2104.07715.

[87] R. LaRose, A. Tikku, É. O'Neel-Judy, L. Cincio, and P. J. Coles. Variational quantum state diagonalization. *npj Quantum Information*, 5(1):57, 2019. doi:10.1038/s41534-019-0167-6.

[88] E. G. Learned-Miller. Introduction to supervised learning. *I: Department of Computer Science, University of Massachusetts*, 3, 2014.

[89] S. M. Lee, J. Lee, and J. Bang. Learning unknown pure quantum states. *Physical Review A*, 98(5):052302, 2018. doi:10.1103/physreva.98.052302.

[90] L. Leone, S. F. Oliviero, L. Cincio, and M. Cerezo. On the practical usefulness of the hardware efficient ansatz. *arXiv*, 2022. doi:10.48550/arXiv.2211.01477.

[91] H. Li and F. D. M. Haldane. Entanglement spectrum as a generalization of entanglement entropy: Identification of topological order in non-abelian fractional quantum hall effect states. *Physical review letters*, 101(1):010504, 2008. doi:10.1103/physrevlett.101.010504.

[92] Y.-C. Liang, Y.-H. Yeh, P. E. Mendonça, R. Y. Teh, M. D. Reid, and P. D. Drummond. Quantum fidelity measures for mixed states. *Reports on Progress in Physics*, 82(7):076001, 2019. doi:10.1088/1361-6633/ab1ca4.

[93] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992. doi:10.1007/978-1-4615-3618-5_5.

[94] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994. doi:10.1016/b978-1-55860-335-6.50027-1.

[95] X. Liu, A. Angone, R. Shaydulin, I. Safro, Y. Alexeev, and L. Cincio. Layer vqe: A variational approach for combinatorial optimization on noisy quantum computers. *IEEE Transactions on Quantum Engineering*, 3:1–20, 2022. doi:10.1109/tqe.2021.3140190.

[96] S. Lloyd. Quantum approximate optimization is computationally universal. *arXiv*, 2018. doi:10.48550/arXiv.1812.11075.

[97] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014. doi:10.1038/nphys3029.

[98] V. Lordi and J. M. Nichol. Advances and opportunities in materials science for scalable quantum computing. *MRS Bulletin*, 46:589–595, 2021. doi:10.1557/s43577-021-00133-0.

[99] A. P. Lund, M. J. Bremner, and T. C. Ralph. Quantum sampling problems, bosonsampling and quantum supremacy. *npj Quantum Information*, 3(1):15, 2017. doi:10.1038/s41534-017-0018-2.

[100] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018. doi:10.1038/s41467-018-07090-4.

[101] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016. doi:10.1088/1367-2630/18/2/023023.

[102] F. S. Melo. Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4, 2001.

[103] H. Meyer. The molecular hamiltonian. *Annual review of physical chemistry*, 53(1):141–172, 2002.

[104] J. A. Miszczak, Z. Puchała, P. Horodecki, A. Uhlmann, and K. Życzkowski. Sub–and super–fidelity as bounds for quantum fidelity. *Quantum Information and Computation*, 9(1&2):0103–0130, 2009. doi:10.26421/qic9.1-2-7.

[105] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv*, 2013. doi:10.48550/arXiv.1312.5602.

[106] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. doi:10.1038/nature14236.

[107] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023. doi:10.1561/9781638280576.

[108] M. Mohseni, P. Read, H. Neven, S. Boixo, V. Denchev, R. Babbush, A. Fowler, V. Smelyanskiy, and J. Martinis. Commercialize quantum technologies in five years. *Nature*, 543(7644):171–174, 2017. doi:10.1038/543171a.

[109] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, et al. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, 2018. doi:10.1088/2058-9565/aab822.

[110] G. Montavon, W. Samek, and K.-R. Müller. Methods for interpreting and understanding deep neural networks. *Digital signal processing*, 73:1–15, 2018. doi:10.1016/j.dsp.2017.10.011.

[111] M. E. Morales, J. D. Biamonte, and Z. Zimborás. On the universality of the quantum approximate optimization algorithm. *Quantum Information Processing*, 19:1–26, 2020. doi:10.1007/s11128-020-02748-9.

[112] L. Moro, M. G. Paris, M. Restelli, and E. Prati. Quantum compiling by deep reinforcement learning. *Communications Physics*, 4(1):178, 2021. doi:10.1038/s42005-021-00684-3.

[113] H. Ni, H. Li, and L. Ying. On low-depth algorithms for quantum phase estimation. *arXiv*, 2023. doi:10.48550/arXiv.2302.02454.

[114] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010. doi:10.1017/cbo9780511976667.

[115] R. O'Donnell and J. Wright. Efficient quantum tomography. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 899–912, 2016. doi:10.1145/2897518.2897544.

[116] M. Ostaszewski, L. M. Trenkwalder, W. Masarczyk, E. Scerri, and V. Dunjko. Reinforcement learning for optimization of variational quantum circuit architectures. *Advances in Neural Information Processing Systems*, 34:18182–18194, 2021.

[117] M. Otten, C. L. Cortes, and S. K. Gray. Noise-resilient quantum dynamics using symmetry-preserving ansatzes. *arXiv*, 2019. doi:10.48550/arXiv.1910.06284.

[118] Y. J. Patel, S. Jerbi, T. Bäck, and V. Dunjko. Reinforcement learning assisted recursive qaoa. *arXiv*, 2022. doi:10.48550/arXiv.2207.06294.

[119] Y. J. Patel, A. Kundu, M. Ostaszewski, X. Bonet-Monroig, V. Dunjko, and O. Danaci. Curriculum reinforcement learning for quantum architecture search under hardware errors. In *The Twelfth International Conference on Learning Representations*, 2024. URL: https://openreview.net/forum?id=rINBD8jPoP.

[120] 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 photonic quantum processor. *Nature communications*, 5(1):4213, 2014. doi:10.1038/ncomms5213.

[121] M. Petschow and P. Bientinesi. The algorithm of multiple relatively robust representations for multi-core processors. In *Applied Parallel and Scientific Computing: 10th International Conference, PARA 2010, Reykjavík, Iceland, June 6-9, 2010, Revised Selected Papers, Part I 10*, pages 152–161. Springer, 2012. doi:10.1007/978-3-642-28151-8_15.

[122] M. Piani and J. Watrous. All entangled states are useful for channel discrimination. *Physical Review Letters*, 102(25):250501, 2009. doi:10.1103/physrevlett.102.250501.

[123] A. S. Polydoros and L. Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017. doi:10.1007/s10846-017-0468-y.

[124] M. J. Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994. doi:10.1007/978-94-015-8330-5_4.

[125] J. Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018. doi:10.22331/q-2018-08-06-79.

[126] M. Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16*, pages 317–328. Springer, 2005. doi:10.1007/11564096_32.

[127] Rigetti Corporation and J. Russell. Rigetti Launches Quantum Cloud Services, Announces $1Million Challenge, 2023. Accessed on January 1, 2024. URL: https://medium.com/rigetti/introducing-rigetti-quantum-cloud-services-c6005729768c.

[128] J. Romero and A. Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *Advanced Quantum Technologies*, 4(1):2000003, 2021.

[129] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. J. Love, and A. Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Science and Technology*, 4(1):014008, 2018. doi:10.1088/2058-9565/aad3e4.

[130] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv*, 2016. doi:10.48550/arXiv.1609.04747.

[131] G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[132] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and A. Wen. A tutorial on thompson sampling. arxiv. *arXiv*, 2017. doi:10.48550/arXiv.1707.02038.

[133] M. Sedlák and M. Ziman. Unambiguous comparison of unitary channels. *Physical Review A*, 79(1):012303, 2009. doi:10.1103/physreva.79.012303.

[134] J. T. Seeley, M. J. Richard, and P. J. Love. The bravyi-kitaev transformation for quantum computation of electronic structure. *The Journal of chemical physics*, 137(22), 2012. doi:10.1063/1.4768229.

[135] K. Setia and J. D. Whitfield. Bravyi-kitaev superfast simulation of electronic structure on a quantum computer. *The Journal of chemical physics*, 148(16), 2018. doi:10.1063/1.5019371.

[136] Z.-X. Shang, M.-C. Chen, X. Yuan, C.-Y. Lu, and J.-W. Pan. Schrödinger-heisenberg variational quantum algorithms. *Physical Review Letters*, 131(6):060406, 2023.

[137] Y. Shee, P.-K. Tsai, C.-L. Hong, H.-C. Cheng, and H.-S. Goan. Qubit-efficient encoding scheme for quantum simulations of electronic structure. *Physical Review Research*, 4(2):023154, 2022. doi:10.1103/physrevresearch.4.023154.

[138] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[139] R. C. Sickles and V. Zelenyuk. *Measurement of productivity and efficiency*. Cambridge University Press, 2019. doi:10.1017/9781139565981.

[140] A. Sone, M. Cerezo, J. L. Beckey, and P. J. Coles. Generalized measure of quantum fisher information. *Physical Review A*, 104(6):062602, 2021. doi:10.1103/physreva.104.062602.

[141] J. Sorg, R. L. Lewis, and S. Singh. Reward design via online gradient ascent. *Advances in Neural Information Processing Systems*, 23, 2010.

[142] J. C. Spall. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on aerospace and electronic systems*, 34(3):817–823, 1998. doi:10.1109/7.705889.

[143] M. Steudtner and S. Wehner. Fermion-to-qubit mappings with varying resource requirements for quantum simulation. *New Journal of Physics*, 20(6):063010, 2018. doi:10.1088/1367-2630/aac54f.

[144] P. Strobach. Bi-iteration svd subspace tracking algorithms. *IEEE Transactions on signal processing*, 45(5):1222–1240, 1997. doi:10.1109/78.575696.

[145] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988. doi:10.1007/bf00115009.

[146] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[147] Y. Suzuki, Y. Kawase, Y. Masumura, Y. Hiraga, M. Nakadai, J. Chen, K. M. Nakanishi, K. Mitarai, R. Imai, S. Tamiya, et al. Qulacs: a fast and versatile quantum circuit simulator for research purpose. *Quantum*, 5:559, 2021. doi:10.22331/q-2021-10-06-559.

[148] H. L. Tang, V. Shkolnikov, G. S. Barron, H. R. Grimsley, N. J. Mayhall, E. Barnes, and S. E. Economou. qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor. *PRX Quantum*, 2(2):020310, 2021. doi:10.1103/prxquantum.2.020310.

[149] A. G. Taube and R. J. Bartlett. New perspectives on unitary coupled-cluster theory. *International journal of quantum chemistry*, 106(15):3393–3401, 2006. doi:10.1002/qua.21198.

[150] B. M. Terhal. Quantum supremacy, here we come. *Nature Physics*, 14(6):530–531, 2018. doi:10.1038/s41567-018-0131-y.

[151] G. Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995. doi:10.1145/203330.203343.

[152] N. V. Tkachenko, J. Sud, Y. Zhang, S. Tretiak, P. M. Anisimov, A. T. Arrasmith, P. J. Coles, L. Cincio, and P. A. Dub. Correlation-informed permutation of qubits for reducing ansatz depth in the variational quantum eigensolver. *PRX Quantum*, 2(2):020337, 2021. doi:10.1103/prxquantum.2.020337.

[153] M. Tomamichel. *Quantum information processing with finite resources: mathematical foundations*, volume 5. Springer, 2015.

[154] M. Tomamichel, R. Colbeck, and R. Renner. Duality between smooth min-and max-entropies. *IEEE Transactions on information theory*, 56(9):4674–4681, 2010. doi:10.1109/tit.2010.2054130.

[155] A. Tranter, P. J. Love, F. Mintert, and P. V. Coveney. A comparison of the bravyi–kitaev and jordan–wigner transformations for the quantum simulation of quantum chemistry. *Journal of chemical theory and computation*, 14(11):5617–5630, 2018. doi:10.1021/acs.jctc.8b00450.

[156] A. Uhlmann. The "transition probability" in the state space of a*-algebra. *Reports on Mathematical Physics*, 9(2):273–279, 1976. doi:10.1016/0034-4877(76)90060-4.

[157] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. doi:10.1609/aaai.v30i1.10295.

[158] G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidary. Quantum graph neural networks. *arXiv*, 2019. doi:10.48550/arXiv.1909.12264.

[159] D. Wang, O. Higgott, and S. Brierley. Accelerated variational quantum eigensolver. *Physical review letters*, 122(14):140504, 2019. doi:10.1103/physrevlett.122.140504.

[160] G. Wang and M. Ying. Unambiguous discrimination among quantum operations. *Physical Review A*, 73(4):042301, 2006. doi:10.1103/physreva.73.042301.

[161] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature communications*, 12(1):6961, 2021. doi:10.1038/s41467-021-27045-6.

[162] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992. doi:10.1007/bf00992698.

[163] C. J. C. H. Watkins. Learning from delayed rewards. *arXiv*, 1989.

[164] D. S. Watkins. Understanding the qr algorithm. *SIAM review*, 24(4):427–440, 1982. doi:10.1137/1024100.

[165] W. Wu, G. Yan, X. Lu, K. Pan, and J. Yan. Quantumdarts: Differentiable quantum architecture search for variational quantum algorithms. *International Conference on Machine Learning. PMLR*, 2023.

[166] E. Ye and S. Y.-C. Chen. Quantum architecture search via continual reinforcement learning. *arXiv*, 2021. doi:10.48550/arXiv.2112.05779.

[167] Y. S. Yordanov, V. Armaos, C. H. Barnes, and D. R. Arvidsson-Shukur. Qubit-excitation-based adaptive variational quantum eigensolver. *Communications Physics*, 4(1):228, 2021. doi:10.1038/s42005-021-00730-0.

[168] J. Zeng, C. Cao, C. Zhang, P. Xu, and B. Zeng. A variational quantum algorithm for hamiltonian diagonalization. *Quantum Science and Technology*, 6(4):045009, 2021. doi:10.1088/2058-9565/ac11a7.

[169] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao. Differentiable quantum architecture search. *Quantum Science and Technology*, 7(4):045023, 2022. doi:10.1088/2058-9565/ac87cd.

[170] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.

Appendix A

Basics of quantum computing

A.1 Quantum states

A quantum state encapsulates all the information about a quantum system, including its position, momentum, spin, and other relevant properties. Mathematically, quantum states are represented as vectors in a complex vector space known as a Hilbert space. For one qubit quantum state can be represented by a linear combination of the $|0\rangle$ and $|1\rangle$ as follows

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{A.1}$$

where α and β are complex numbers known as probability amplitudes and $|0\rangle$, $|1\rangle$ are orthonormal ($\langle 0|1\rangle = 0$) are known as basis states. We can do a projective measurement on the basis and find the probability of finding the $|0\rangle$ to be $\langle\psi|0\rangle = |\alpha|^2$ and finding the $|1\rangle$ to be $\langle\psi|1\rangle = |\beta|^2$. If the quantum state is a linear combination of j basis vectors, then the quantum state looks as follows

$$|\chi\rangle = \sum_j c_j|\phi\rangle_j. \tag{A.2}$$

Another way to express a quantum state is through density matrices, where we take the outer product of the state Eq. (A.2), which results into

$$\rho = |\psi\rangle\langle\psi| = \sum p_j|\phi\rangle\langle\phi|_j, \tag{A.3}$$

where $p_j = |c_j|^2$. It is important to define states in this way because if we do not have the whole information, only a statistical mixture of states, then we can not get access to the probability amplitudes of corresponding basis states; hence, it is not possible to represent it with a single ket vector. As such, we see mixed states.

Pure state If the ρ can not be expressed as a mixture/convex combination of other states, it can be defined as a single ket vector, i.e. $\rho = |\psi\rangle\langle\psi|$. The purity of a pure state, i.e. $\mathrm{Tr}\rho^2 = 1$.

Mixed state If the ρ can be expressed as a statistical mixture/convex combination of other states, it is known as a mixed state. The purity of a mixed state, i.e. $\mathrm{Tr}\rho^2 < 1$.

The purity mathematically helps us distinguish between a pure and a mixed state. It tells us that a pure state exhibits maximal coherence and lacks inherent uncertainty in its properties. However, under realistic consideration when, the environment can influence the quantum systems, leading to interactions that cause decoherence and introduce noise. Which results in a mixed state—a statistical ensemble of pure states.

A.2 Quantum gates

Quantum gates are fundamental elements in quantum computing that perform operations on qubits. These gates are represented by unitary operators in quantum mechanics, transforming the quantum state of one or more qubits. Each gate corresponds to a specific operation, such as changing the state of a qubit, entangling multiple qubits, or performing computations on quantum data. For the sake of the thesis, we only define three kinds of one-qubit parameterized quantum gates that rotate a qubit in X (`RX`), Y (`RY`) and Z (`RZ`) direction as follows

$$\texttt{RX} = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix}, \quad \texttt{RY} = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}, \tag{A.4}$$

$$\texttt{RZ} = \begin{bmatrix} \exp(-i\theta/2) & -0 \\ 0 & \exp(i\theta/2) \end{bmatrix}. \tag{A.5}$$

Additionally, for a two-qubit gate, we use the `CX` gate, where one of the qubits works as a control and modifies the target qubit depending on the input in the control. Without influencing the control qubit, the gate performs a Pauli X gate on the target qubit when the control qubit is in $|1\rangle$.

A.3 Quantum channels

Quantum channels represent the mathematical formalism used to describe the evolution of quantum systems under the influence of various physical processes,

including interactions with an external environment. A quantum channel is a completely positive and trace-preserving map that describes the evolution of a quantum state. It models how quantum information interacts with its surroundings, resulting in system state changes.

A quantum channel $\mathcal{E}$ can be expressed as a linear, completely positive trace-preserving (CPTP) map that acts on the density operators of the quantum states. Mathematically, the action of the quantum channel $\mathcal{E}$ on a quantum state ρ is represented by

$$\rho' = \mathcal{E}(\rho),$$

where ρ' is the density matrix after passing through the channel $\mathcal{E}$. To adhere to the complete positivity requirement, the channel $\mathcal{E}$ should leave all positive semidefinite operators invariant under its action, ensuring the preservation of the positivity of the density matrix. A fundamental property of quantum channels is their linearity, characterized by

$$\mathcal{E}(\alpha\rho_1 + \beta\rho_2) = \alpha\mathcal{E}(\rho_1) + \beta\mathcal{E}(\rho_2),$$

for any scalar λ, μ and input states ρ_1, and ρ_2. This property ensures that the transformation remains consistent and proportional to the input states.

Additionally, the trace-preserving property of the quantum channel dictates that the output state maintains a unit trace, capturing the conservation of probability:

$$\mathrm{Tr}(\mathcal{E}(\rho_j)) = \mathrm{Tr}(\rho_j) = 1,$$

ensuring the preservation of the normalization of quantum states.

Quantum channels are pivotal in various quantum information processing tasks, including quantum communication, error correction, and computation. Understanding the properties and characteristics of quantum channels is crucial for designing efficient communication protocols, error-correcting codes, and quantum algorithms.

In this thesis, we utilize specific types of quantum channels, such as unitary (random `X`) channels and depolarizing channels. Each chapter where these channels are applied will include a brief explanation to aid conceptual clearance.

Appendix B

Proof of Truncated Fidelity Bound 5.6

Proposition 1: *The truncated fidelity bounds are as follows:*

$$F(\rho_m, \sigma_m^\rho) \leq F(\rho, \sigma) \leq F_*(\rho_m, \sigma_m^\rho). \tag{B.1}$$

Proof. The F_* represents the generalized fidelity and the RHS of the Eq. (B.1) follows from the monotonous nature of F_* under the completely positive trace non-increasing maps [153, 154]. Such a map can be defined as

$$M_m(\rho) = \Pi_m^\rho \rho \Pi_m^\rho, \tag{B.2}$$

where Π_m^ρ is the projector on the subspace spanned by the eigenvectors for the m-largest eigenvalues. This helps us write

$$F(\rho, \sigma) = F_*(\rho, \sigma) \leq F_*(M_m(\rho), M_m(\sigma)) = F_*(\rho_m, \sigma_m^\rho). \tag{B.3}$$

Meanwhile, the lower bound i.e. $F(\rho, \sigma) \geq F(\rho_m, \sigma_m^\rho)$ can be derived using the strong concavity of fidelity [114] as follows

$$\begin{aligned} F(\rho, \sigma) = F(M_m(\rho) + \bar{M}_m(\rho), \sigma) &\geq \sqrt{p_m} F\left(\frac{M_m(\rho)}{p_m}, \rho\right) + \sqrt{1 - p_m} F\left(\frac{\bar{M}_m(\rho)}{1 - p_m}, \sigma\right) \\ &\geq \sqrt{p_m} F\left(\frac{\rho_m}{p_m}, \rho\right) = F(\rho_m, \sigma) = F(\rho_m, \sigma_m^\rho), \end{aligned} \tag{B.4}$$

where $p_m = \mathrm{Tr}\rho_m$ and we have expressed

$$\rho = M_m(\rho) + \bar{M}_m(\rho) \tag{B.5}$$

where $\bar{M}_m(\rho) = \bar{\Pi}^\rho_m \rho \bar{\Pi}^\rho_m$ and $\bar{\Pi}^\rho_m$ is the orthogonal compliment of Π^ρ_m. ■

Appendix C

Illegal actions elaboration

Here we delve into a comprehensive discussion of the concept of illegal actions. For every N-qubit system, there can be a maximum of N actions applied to individual qubit(s). We start with an empty list, representing an initially empty circuit, which can accommodate up to four sub-lists, each corresponding to a specific illegal action.

$$A_{\text{illegal}} = [\underbrace{[]}_{\text{N times}}]. \tag{C.1}$$

If the agent selects an action represented as $[i, j, N, N]$, signifying a `CX` operation with control on qubit i and target on qubit $(i + j) \pmod N$, the illegal action list is adjusted as follows:

$$A_{\text{illegal}} = [[i, j, N, N], \underbrace{[]}_{(N-1)\text{ times}}]. \tag{C.2}$$

For the selection of the next action, we employ a selection rule that encourages the agent to opt for the action with the highest estimated action value. To prevent the agent from repeating the action $[i, j, N, N]$ in the next action, we assign a corresponding estimate of $-\infty$ to the illegal action. Therefore, during the greedy action selection process, the agent automatically excludes the illegal action from the available list of actions, pruning the action space.

In the next time step if the agent decides on an action $[N, N, l, m]$ which corresponds to a rotation towards m direction and the rotation to be applied on l-th qubit then if $i \neq l$, and $(i + j) \pmod N \neq l$, then the illegal actions updated as:

$$A_{\text{illegal}} = [[i, j, N, N], [N, N, l, m], \underbrace{[]}_{(N-2)\text{ times}}], \tag{C.3}$$

else if $i = l$ or $(i + j) \pmod N = l$ then the update of illegal actions follows

$$A_{\text{illegal}} = [[N, N, l, m], \underbrace{[]}_{(N-1) \text{ times}}]. \tag{C.4}$$

In either case, we set the estimate corresponding to the illegal actions to $-\infty$, and the agent does not choose the action or the list of actions.

Appendix D

Implementation of components of RLQAS

D.1 RL-state implementation

Here, we give the code for the tensor-based encoding for the ansatz that is used as the observable for the RL-agent. The state is represented as a `torch tensor` of dimension $T \times ((N+3) \times N)$. Where in the code block `qubits` $= N$, `num_step` $= T$. To keep track of the gates and the operations, we use the notion of `moments` in a quantum circuit. The moment of a quantum circuit is defined by a collection of quantum gates that all act during the same abstract time slice [1]. The RL-state gets filled up by the rule presented through Fig. (3.6) for each action. Meanwhile, the action space encoding is straightforward and is presented in Fig. (3.7). In 22, we provide a python code example for RL-state (presented by `state`) encoding where the action space is denoted by the `action_list` and for each action, the `state` gets updated. In order to keep track of the gates and the qubits, we make use of the notion of `moments` [1].

Algorithm 3 RL-state construction and update

Require: $qubits$ (the number of qubits in the problem)
Require: num_step (number of steps in an episode)
Require: $action\_list$ (list of possible actions)
Ensure: Updated RL-state ($state$)
Ensure: Updated ansatz moments ($moments$)

$state \leftarrow$ torch.zeros$((num_step, qubits + 3 + 3, qubits))$
$moments \leftarrow [0] * qubits$
for each $action$ in $action\_list$ **do**
    Extract $ctrl$, $targ$, rot_qubit, and rot_axis from $action$
    if $rot\_qubit < qubits$ **then**
        $gate\_tensor \leftarrow moments[rot\_qubit]$
    else if $ctrl < qubits$ **then**
        $gate\_tensor \leftarrow$ np.max$(moments[ctrl], moments[targ])$
    end if
    if $ctrl < qubits$ **then**
        $state[gate\_tensor][targ][ctrl] \leftarrow 1$
    else if $rot\_qubit < qubits$ **then**
        $state[gate\_tensor][qubits + rot\_axis - 1][rot\_qubit] \leftarrow 1$
    end if
    if $rot\_qubit < qubits$ **then**
        $moments[rot\_qubit] \leftarrow moments[rot\_qubit] + 1$
    else if $ctrl < qubits$ **then**
        $max\_of\_two\_moments \leftarrow$ np.max$(moments[ctrl], moments[targ])$
        $moments[ctrl] \leftarrow max\_of\_two\_moments + 1$
        $moments[targ] \leftarrow max\_of\_two\_moments + 1$
    end if
end for

D.2 Illegal actions implementation

The following code block provides an elaborated implementation of the *illegal actions* technique. As a quantum physicist, I must concede that the code is entangled to a degree surpassing even a maximally mixed state, and I am confident that some can render it significantly and make it more decoherent.

```
qubits = #the number of qubits in the problem
current_action = [qubits]*4
illegal_actions = [[]]*qubits
action_list = #list of actions

```

```
for action in action_list:
    action=current_action
    ctrl,targ = action[0],(action[0]+action[1])%qubits
    rot_qubit,rot_axis = action[2],action[3]

    if ctrl < qubits:
        are_you_empty=sum([sum(l) for l in illegal_actions])
        if are_you_empty!=0:
            for ill_ac_no,ill_ac in enumerate(illegal_actions):
                if len(ill_ac) != 0:
                    ill_ac_targ=(ill_ac[0]+ill_ac[1])%qubits
                    if ill_ac[2]==qubits:

                        if ctrl==ill_ac[0] or ctrl==ill_ac_targ:
                            illegal_actions[ill_ac_no]=[]
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
                        elif targ==ill_ac[0] or targ==ill_ac_targ:
                            illegal_actions[ill_ac_no]=[]
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
                        else:
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break

                    else:
                        if ctrl==ill_ac[2]:
                            illegal_actions[ill_ac_no]=[]
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
                        elif targ==ill_ac[2]:
                            illegal_actions[ill_ac_no]=[]
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
                        else:
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
```

```
        else:
            illegal_actions[0]=action

    if rot_qubit<qubits:
        are_you_empty=sum([sum(l) for l in illegal_actions])
        if are_you_empty!=0:
            for iac_no, iac in enumerate(illegal_actions):

                if len(iac)!=0:
                    ill_ac_targ=(iac[0]+iac[1])%qubits
                    if iac[0]==qubits:

                        if rot_qubit==iac[2] and rot_axis!=iac[3]:
                            illegal_actions[iac_no]=[]
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
                        elif rot_qubit!=iac[2]:
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
                    else:
                        if rot_qubit==iac[0]:
                            illegal_actions[iac_no]=[]
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
                        elif rot_qubit==iac_targ:
                            illegal_actions[iac_no]=[]
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
                        else:
                            for i in range(1, qubits):
                                if len(illegal_actions[i])==0:
                                    illegal_actions[i]=action
                                    break
        else:
            illegal_actions[0]=action

    for indx in range(qubits):
        for jndx in range(indx+1, qubits):
            if illegal_actions[indx]==illegal_actions[jndx]:
                if jndx!=indx +1:
                    illegal_actions[indx]=[]
```

```
                else:
                    illegal_actions[jndx]=[]
                break

    for indx in range(qubits-1):
        if len(illegal_actions[indx])==0:
            illegal_actions[indx]=illegal_actions[indx+1]
            illegal_actions[indx+1]=[]

    illac_decode=[]
    for key, contain in dictionary_of_actions(qubits).items():
        for ill_action in illegal_actions:
            if ill_action==contain:
                illac_decode.append(key)
```

Listing D.1: A python code example for illegal action (presented by `illegal_actions`) technique where the action space is denoted by the `action_list` and for each action the `illegal_actions` list gets updated. As a final product we get `illac_decode`.

D.3 3-stage Adam-SPSA pseudocode and hyperparameters setting

In this section we provide the pseudocode for multistage Adam-SPSA optimization algorithm and the parameters that are fixed while running the algorithm.

Algorithm 1: Simultaneous Perturbation Stochastic Approximation with Adam (Adam-SPSA)

Data: Initial parameter vector θ, Objective function $f(\theta)$, Number of iterations K
Result: Optimal parameter vector θ^*
Hyperparameters: a, α, c, γ_{sp}, λ, β_1, β_2
Initialize momentum parameters to zero: $m, v \leftarrow 0$;
for $k = 1$ **to** K **do**
  Compute scaling parameters: $a_k \leftarrow \frac{a}{(k+1)^{\alpha}}, \quad c_k \leftarrow \frac{c}{(k+1)^{\gamma_{sp}}}$;
  Compute hyperparameters: $\beta_{1,k} \leftarrow \frac{\beta_1}{(k+1)^{\lambda}}$;
  Randomly choose a perturbation vector Δ_k with elements ± 1;
  Evaluate objective function gradients: $g_k^+ \leftarrow f(\theta + c_k\Delta_k)$ and $g_k^- \leftarrow f(\theta - c_k\Delta_k)$;
  Compute gradient estimate: $\nabla J_k \leftarrow \frac{g_k^+ - g_k^-}{2c_k\Delta_k}$;
  Biased update of moment parameters $\hat{m}$ and $\hat{v}$:
  $\hat{m} \leftarrow \beta_{1,k}m + (1-\beta_{1,k})\nabla J_k, \quad \hat{v} \leftarrow \beta_2 v + (1-\beta_2)(\nabla J_k)^2$;
  Unbiased computation of moment parameters $\hat{m}$ and $\hat{v}$:
  $\hat{m} \leftarrow \frac{m}{1-\beta_{1,k}^{k+1}}, \quad \hat{v} \leftarrow \frac{v}{1-\beta_2^{k+1}}$;
  Update gradient estimate: $\nabla J_k \leftarrow \frac{\hat{m}}{\sqrt{\hat{v}}+k}$;
  Update parameters: $\theta \leftarrow \theta - a_k\nabla J_k$;
end

Table D.1: The hyperparameters of Adam-SPSA optimizer used during the noisy simulations. In the noisy simulation of 2-, and 3-qubit problems we used 1-stage version of the algorithm, therefore only single maximum function evaluation hyperparameters are given. The parameters within the curly brackets denote the maximum number of function evaluations in the 3-stage version of the algorithm. We provide Max fevals both for 1-stage and their 3-stage equivalents.

Molecule	a	α	β_1	β_2	c	γ_{sp}	λ	Max fevals	Shots
H_2-2	1.2104	0.9531	0.9414	0.9983	0.1039	0.0984	0.9277	500	10^3
H_2-3	0.5188	0.9859	0.716	0.6265	0.0938	0.0974	0.6483	500	10^4
LiH-4	1.2324	0.9709	0.6114	0.9326	0.2215	0.1485	0.9772	1600 {1191, 357, 119}	3300 {2383, 715, 238}	10^6
LiH-6	1.7564	0.8365	0.6841	0.9048	0.1068	0.1549	0.1223	2000 {1430, 429, 143}		10^8