Master's thesis

Embedding of Tree Tensor Networks into Quantum Circuits of Two-Qubit Gates

(二量子ビットゲート量子回路への
ツリーテンソルネットワークの埋め込み)

January, 2025

Supervisor: Prof. Synge Todo

35236050 Shota Sugawara

Department of Physics, Graduate School of Science, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan.

Acknowledgments

This work is fully supervised and supported by Professor Synge Todo. He dedicated a significant amount of time to my research, from discussing physics to debugging code. Without his support, I would not have been able to complete my master's thesis. Additionally, he taught me many important life lessons, such as how to manage projects, handle failures, and think when faced with difficulties. What I learned from him will be a valuable guide for my future life. Thanks to him, the two years of my master's program became an invaluable and precious time.

I am profoundly thankful to Project Associate Professor Tsuyoshi Okubo. He also dedicated a significant amount of time to my research and made substantial contributions. Not only during the regular meetings but whenever I had questions, he would always use his deep knowledge of physics to accurately resolve my questions. It is thanks to him that I have gained a solid understanding of tensor networks.

I would also like to thank the other members of the Todo Group. Kazuki Inomata was responsible for part of the implementation of the research and made a significant contribution to its progress. Keisuke Murota and Takumi Kobori engaged in deep discussions about the relationship between tensor networks and quantum circuits. Tokinori Oe, as a fellow lab member, always helped me through difficult times. Additionally, I had regular discussions with Hidemaro Suwa, Masahiko Yamada, Sayan Mukherjee, Atsushi Iwaki, Tatsuya Sakashita, Keiichi Tamai, Tohru Mashiko, Yuma Nakanishi, Rihito Sakurai, Shinichiro Akiyama, Xun Zhao, Hopai Kwok, Sora Shiratani, Soichiro Imamura, Ruixiao Cao, Kazushige Ueda, and Ruqing Xu during our meetings. Their broad knowledge of physics was very stimulating for me. Emi Shimoshikiryo supported research activities and helped me as the secretary of the Todo Group. Their help always helped me a lot.

Finally, I would like to express my most profound appreciation to my

family. Thanks to my father, Mitsuru Sugawara, my mother, Yuka Sugawara, and my sister, Yumeka Sugawara, our home is always filled with laughter, allowing me to conduct my research in a mentally stable state. I was able to complete my master's thesis successfully because of them. I also want to express my deep gratitude to my grandmother, Miyoko Iwasaki, who passed away during my university years. She always supported me and celebrated my successes as if they were her own. The love she gave me continues to be a source of strength for me.

Shota Sugawara

Abstract

Variational Quantum Algorithms (VQAs) are highlighted as key algorithms for demonstrating a quantum advantage on Noisy Intermediate-Scale Quantum (NISQ) devices, which are limited to executing shallow quantum circuits because of noise. However, the barren plateau problem, where the amplitude of the loss function gradient becomes exponentially small with the number of qubits, hinders this goal. Recent studies suggest that embedding tensor networks into quantum circuits and initializing the parameters by them can avoid the barren plateau. Yet, embedding tensor networks into quantum circuits is generally difficult, and methods have been limited to the simplest structure, Matrix Product States (MPSs). This study proposes a method to embed Tree Tensor Networks (TTNs), characterized by their hierarchical structure, into shallow quantum circuits. TTNs are suitable for representing higher-dimensional systems and systems with long-range correlations, which MPSs are inadequate for representing. Our numerical results show that embedding TTNs provides better initial quantum circuits than MPS. Additionally, our method has a practical computational complexity, making it applicable to a wide range of TTNs. This study is expected to extend the application of VQAs to higher-dimensional systems and those with long-range correlations, which have been challenging to utilize.

Contents

Chapter 1

Introduction

Variational Quantum Algorithms (VQAs) are the foremost approach for achieving a quantum advantage with the current generation of quantum computing technologies. Quantum computers are anticipated to outperform classical ones, with some algorithms already proving more efficient [Shor, 1994, Grover, 1996]. However, the currently available Noisy Intermediate-Scale Quantum (NISQ) devices cannot execute most algorithms due to their limited number of qubits and susceptibility to noise [Preskill, 2018].

VQAs are hybrid methods in which classical computers optimize parameters of quantum circuits' ansatz to minimize the cost function evaluated by quantum computers. VQAs require only shallow circuits, making them notable as algorithms that can be executed on NISQ devices. VQAs come in many forms, such as Quantum Machine Learning (QML) [Biamonte et al., 2017, Mitarai et al., 2018, Schuld and Killoran, 2019], Variational Quantum Eigensolver (VQE) [Abrams and Lloyd, 1999, Aspuru-Guzik et al., 2005, Peruzzo et al., 2014], and Quantum Approximate Optimization Algorithm (QAOA) [Farhi et al., 2014, Wang et al., 2018, Zhou et al., 2020], with potential applications across diverse industries and fields.

However, a significant challenge known as the barren plateau stands in the way of realizing quantum advantage [McClean et al., 2018, Holmes et al., 2022]. The barren plateau phenomenon refers to the difficulty in VQAs where the amplitude of the cost function gradient decreases exponentially as the number of qubits increases. This phenomenon occurs regardless of whether the optimization method is gradient-based [Cerezo and Coles, 2021] or gradient-free [Arrasmith et al., 2021], and

it has been observed even in shallow quantum circuits [Cerezo et al., 2021]. Furthermore, it has been confirmed that this phenomenon also occurs in practical tasks using real-world data [Holmes et al., 2021, Sharma et al., 2022, Ortiz Marrero et al., 2021]. Avoiding the barren plateau is a critical challenge in demonstrating the superiority of quantum algorithms using NISQ devices. To avoid the barren plateau, appropriate parameter initialization in VQAs is crucial since randomly initializing the parameters can result in the algorithm starting far from the solution or near a local minimum [Zhou et al., 2020]. Although various initialization methods have been considered [Wiersema et al., 2020, Grant et al., 2019a, Friedrich and Maziero, 2022a], using tensor networks is natural due to their compatibility with quantum circuits.

Tensor networks are originally developed to represent quantum many-body wave functions efficiently. Any quantum circuit can be naturally regarded as a tensor network [Markov and Shi, 2008], and it is sometimes possible to simulate quantum computers with a practical amount of time using tensor networks [Liu et al., 2021]. Moreover, their utility has been recognized and applied in recent years across various fields, such as machine learning [Levine et al., 2019] and language models [Gallego and Orus, 2022].

In this study, we focus particularly on Matrix Product States (MPSs) and Tree Tensor Networks (TTNs), two of the various structures of tensor networks. MPSs are one-dimensional arrays of tensors. Its simplest and easiest-to-use structure, along with the presence of advanced algorithms such as Density Matrix Renormalization Group (DMRG) [White, 1992], and Time-Evolving Block Decimation (TEBD) [Vidal, 2003], has led to its application across a wide range of fields. These excellent algorithms have recently been applied to machine learning, continuing the exploration of new possibilities [Stoudenmire and Schwab, 2016, Han et al., 2018]. On the other hand, TTNs have tree-like structures of tensors. The procedures used in DMRG and TEBD have been applied to TTNs [Shi et al., 2006, Silvi et al., 2010]. Additionally, the distance between any pair of leaf nodes scales in logarithmic order in TTNs, while it scales in linear order in MPSs. As connected correlation functions decay exponentially with path length within a tensor network generally, TTNs are better suited to capture longer-range correlations and represent higher-dimensional objects than MPSs. TTNs are utilized in various fields such as chemical problems [Murg et al., 2010, Gunst et al., 2019], condensed matter physics [Nagaj et al., 2008, Tagliacozzo et al., 2009, Lin et al., 2017], and

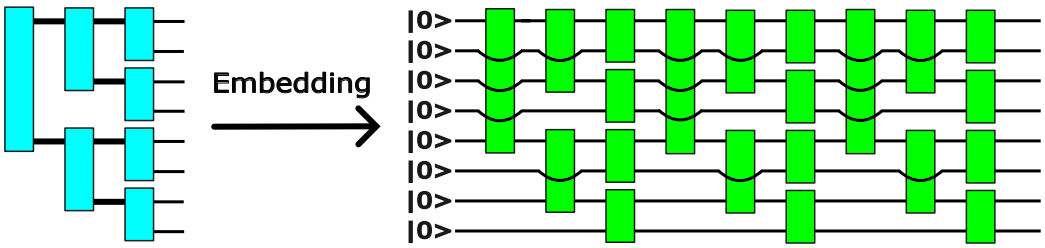

Figure 1.1: A schematic diagram illustrating the embedding of a TTN into a shallow quantum circuit composed solely of two-qubit gates. The aim is to ensure that the quantum state output by the quantum circuit closely approximates the quantum state represented by the TTN. The diagram depicts a case with three layers, and the approximation accuracy improves as the number of quantum circuit layers increases.

machine learning [Liu et al., 2019a, Cheng et al., 2019].

Rudolph et al. have successfully avoided the barren plateau problem by utilizing tensor networks[Rudolph et al., 2023b]. They proposed a method of first optimizing tensor networks, then mapping the optimized tensor networks to quantum circuits, and finally executing VQAs. While numerical results have shown that this method can indeed avoid the barren plateau, mapping tensor network states into shallow quantum circuits is a general challenge. The case where the tensor network structure is an MPS has already been well-studied [Ran, 2020, Rudolph et al., 2023a, Shirakawa et al., 2024, Malz et al., 2024]. However, effective methods for mapping tensor networks other than MPSs to shallow quantum circuits have not yet been devised.

In this thesis, we propose a method to embed TTNs into shallow quantum circuits composed of only two-qubit gates, as shown in Fig. 1.1. The primary obstacle to embedding TTNs into shallow quantum circuits has been the complexity of contractions arising from the intricate structure of TTNs. In general, the contraction of tensor networks requires an exponentially large memory footprint relative to the system size if performed naively. Furthermore, determining the optimal contraction order for tensor networks is an NP-complete problem [Chi-Chung et al., 1997]. Through the innovative design of a contraction method that balances minimal approximation error and computational efficiency, we successfully extend the embedding method of MPSs [Rudolph et al., 2023a] to TTNs. Additionally, by applying the proposed method to practical problems, we successfully prepare better initial

quantum circuits for VQAs than those provided by MPSs with a practical computational complexity. This study is expected to extend the application of VQAs to higher-dimensional systems and those with long-range correlations, which have previously been challenging to utilize.

Chapter 2

Variational quantum algorithms

2.1 Advantages of quantum computers

Quantum computers operate based on principles entirely different from classical computers, which are currently widespread globally. Although there are numerous challenges to be addressed in both software and hardware before quantum computers can be as ubiquitously used as classical computers, their immense potential has garnered significant global interest. It remains uncertain whether quantum computers will eventually solve problems that classical computers cannot, but many scientists are actively conducting extensive research on the subject.

There are two reasons to believe that quantum computers can perform tasks beyond classical computers' capabilities. The first is that classical algorithms capable of simulating quantum computers are still unknown. Generally, to accurately represent the quantum state of n qubits on a classical computer, it is necessary to maintain 2^n elements. Storing all the elements becomes fundamentally impossible as the number of qubits increases. For example, describing a quantum state with a few hundred qubits may require more bits than the number of atoms in the universe. Despite decades of effort by physicists and computer scientists, we still cannot simulate large-scale quantum computers using classical computers.

The second is that we know specific problems that are difficult to solve with classical computers but can be efficiently addressed by quantum computers. The most renowned example is Shor's algorithm [Shor, 1994], which can factorize huge composite numbers with remarkable speed. Factoring such

large numbers is challenging for classical computers and is used in cryptography due to this difficulty. However, it has been theoretically demonstrated that quantum computers can solve this problem in polynomial time relative to the number of bits of the integer being factored. Although current quantum computers have only achieved the factorization of small numbers, it is anticipated that as the number of usable qubits increases, they will be able to factorize the large numbers used in cryptography. This advancement is expected to have a significant societal impact.

Another well-known algorithm is Grover's algorithm [Grover, 1996], which enables rapid search of specific data within unsorted databases. When using classical computers, finding a target data point in an unsorted database requires $\mathcal{O}(N)$ queries, where N is the number of data points. In contrast, Grover's algorithm can locate the target with $\mathcal{O}(\sqrt{N})$ queries, achieving quadratic speedup. By leveraging Grover's algorithm, the exhaustive search components of various classical algorithms can be accelerated, making it applicable to problems such as satisfiability and searching for original values from specific hash values. The latter application has even been proposed for speeding up Bitcoin mining [Aggarwal et al., 2017].

2.2 Limitations of NISQ devices

While there are high expectations for the realization of quantum computers, achieving practical quantum computers will take a long time. The core of the problem stems from the fundamental requirements inherent in the realization of quantum computers. The first requirement is the isolation requirement. Quantum systems must be nearly perfectly isolated to store and process information reliably. The second requirement is the interaction requirement. Qubits need to interact with each other strongly to process information. With current quantum technologies, building a quantum system that meets both requirements is very challenging.

Eventually, we expect that quantum error correction will allow us to protect and scale up quantum computers. Quantum computation using quantum devices equipped with error correction capabilities is referred to as Fault-Tolerant Quantum Computing (FTQC). The principle of quantum error correction involves encoding quantum information in a highly entangled state, which prevents the environment from accessing and damaging the encoded information, thereby protecting the quantum system. Quantum error correc-

tion techniques have been extensively studied, and many promising methods exist. However, quantum error correction entails significant overhead, meaning that to protect quantum information, we need to use many more physical qubits than the logical qubits we want to safeguard. This extra requirement of physical qubits increases the complexity and resources needed, making it challenging to build reliable quantum computers.

NISQ devices refer to Noisy Intermediate-Scale Quantum devices, widely recognized as the current state of quantum computing technology. NISQ devices have two limitations compared to the ideal quantum computers. First, they have a limited number of qubits, typically in several hundreds. Consequently, they cannot execute general quantum algorithms that require many qubits. However, this number is still promising as it exceeds what can be simulated by brute force using the most powerful existing digital supercomputers. Second, NISQ devices are characterized by significant noise and high error rates. As a result, it is generally expected that these noisy devices cannot execute deep circuits, such as the circuit with one thousand fundamental two-qubit operations, since the noise will overwhelm the signal. Although NISQ devices are still in the developmental stages of quantum technology, they are garnering attention as a crucial step towards the more powerful quantum technologies that will be developed.

Given the high error rates and significant noise in NISQ devices, it might seem essential to incorporate quantum error correction. However, error correction cannot be implemented due to the limited number of qubits in NISQ devices. Consequently, most quantum algorithms cannot be executed on NISQ devices. For instance, practical applications of Grover's algorithm [Grover, 1996] or Shor's algorithm [Shor, 1994] are unlikely to be feasible. Nevertheless, there is considerable interest in quantum-classical hybrid algorithms, which collaborate with classical devices for computation. VQAs are a prominent example of such algorithms that can operate on NISQ devices.

2.3 Overview of VQAs

VQAs are the top proposal for achieving quantum advantage with NISQ devices. VQAs are similar to successful machine-learning methods like neural networks. Furthermore, VQAs utilize classical optimization techniques by employing parameterized quantum circuits on the quantum computer

and outsourcing the parameter optimization to a classical computer. Unlike quantum algorithms for the fault-tolerant era, this keeps the quantum circuit depth shallow, reducing noise.

One of the primary advantages of VQAs is their ability to provide a general framework applicable to a wide range of problems. Despite their different algorithmic structures and complexities, most VQAs share basic elements. The initial step is to define a cost function $C(\cdot)$ that encodes the problem's solution. Subsequently, an ansatz is proposed, a quantum operation dependent on a set of continuous or discrete parameters θ that can be optimized. This ansatz is then trained in a hybrid quantum-classical loop to solve the optimization task by minimizing the cost function,

$$\theta^* = \arg\min_{\theta} C(\theta). \tag{2.1}$$

VQAs are unique in using quantum computers to estimate the cost function $C(\theta)$ while leveraging classical optimizers to train the parameters θ.

A key part of a VQA is encoding the problem into a cost function. Like classical machine learning, the cost function maps the trainable parameters θ to real numbers. More abstractly, it creates a hypersurface, usually called the cost landscape, where the optimizer's job is to navigate this landscape to find the global minimum. The cost function of a VQA is generally expressed as

$$C(\theta) = f(\{\rho_k\}, \{O_k\}, U(\theta)), \tag{2.2}$$

where f is a function, $U(\theta)$ is a parameterized unitary, θ includes discrete and continuous parameters, $\{\rho_k\}$ are input states from a training set, and $\{O_k\}$ are a set of observables.

The cost function for VQAs should meet several key criteria: it must be faithful, be efficiently estimable, provide quantum advantages, be operationally meaningful, and be trainable. Specifically, the minimum of $C(\theta)$ should correspond to the problem's solution, ensuring it is faithful. It should be measurable on a quantum computer with possible classical post-processing, making it efficiently estimable. Additionally, it should not be efficiently computable with a classical computer to ensure quantum advantage. Smaller cost values should indicate better solution quality, making it operationally meaningful. Finally, it should be possible to optimize the parameters θ efficiently, ensuring the cost function is trainable. Designing a cost function that optimally satisfies these requirements is crucial in VQA.

Another key aspect of VQA is its ansatz. Generally, the ansatz's form determines the parameters θ and their training process to minimize costs. The specific structure of an ansatz typically depends on the task, as problem-specific information can often be used to customize the ansatz. These are referred to as problem-inspired ansatzes. Conversely, some ansatz architectures are problem-agnostic, meaning they can be applied even without relevant information.

Here, we review several significant ansatzes. The first is the hardware-efficient ansatz [Kandala et al., 2017]. The hardware-efficient ansatz is designed to reduce the circuit depth needed for implementation on specific quantum hardware. The second is the variable structure ansatz [Grimsley et al., 2019]. The variable structure ansatz optimizes not only the parameters of the circuit but also the structure of the circuit itself during training. The third is the hybrid ansatzes [Takeshita et al., 2020]. The hybrid ansatzes delegate part of the ansatz to classical hardware, reducing the complexity of quantum computations. By incorporating classical hardware, the computational burden on quantum systems is alleviated. Finally, the fourth is the tensor network-inspired ansatz [Yuan et al., 2021, Liu et al., 2019b]. The tensor network-inspired ansatz draws inspiration from tensor network theory, leveraging its extensive experience to enhance expressiveness and achieve efficient representations.

2.4 Examples of VQAs

The main benefit of the VQA paradigm is its support for task-oriented programming, providing a framework for various tasks. VQAs are used in many applications, such as finding molecular ground states, simulating quantum system dynamics, and solving linear equations. In this section, we will discuss two particularly significant applications: the Variational Quantum Eigensolver (VQE) and Quantum Machine Learning (QML).

2.4.1 Variational quantum eigensolver

In matrix-represented physical systems, finding the smallest eigenvalue is essential for many applications. In computational chemistry, for example, the smallest eigenvalue of a Hermitian matrix indicates the energy of a molecule's ground state. While the quantum phase estimation method can

find this eigenvalue, it is known that the quantum circuits required for practical problems are too long to be implemented on NISQ computers. Therefore, the Variational Quantum Eigensolver (VQE) was proposed to estimate the ground state energy of molecules using shallower quantum circuits.

The VQE is designed to determine the ground-state energy E of a Hamiltonian H. The cost function is expressed as

$$C(\theta) = \langle\psi(\theta)| H |\psi(\theta)\rangle . \tag{2.3}$$

This involves minimizing the expectation value of H over a trial state $|\psi(\theta)\rangle = U(\theta)|\psi_0\rangle$ for a given ansatz $U(\theta)$ and an initial state $|\psi_0\rangle$. The cost function is both meaningful and faithful, as $C(\theta) \geq E$, with equality if the trial state $\psi(\theta)$ is the ground state of H. Typically, the Hamiltonian H is represented as a linear combination of products of Pauli operators σ_k, denoted as $H = \sum_k c_k \sigma_k$. Consequently, the cost function $C(\theta)$ is derived from a linear combination of the expectation values of σ_k. Given that practical physical systems are generally described by sparse Hamiltonians, the cost function can be efficiently estimated on quantum computers, with computational costs that usually scale polynomially with the number of qubits.

2.4.2 Quantum machine learning

Quantum machine learning (QML) typically involves utilizing a quantum computer to discern patterns in quantum data to make accurate predictions on unknown and unseen data [Biamonte et al., 2017]. Many QML tasks are formulated as VQA, and various applications have been developed, including classifiers [Mitarai et al., 2018, Schuld and Killoran, 2019], autoencoders [Romero et al., 2017], generative models [Du et al., 2020, Benedetti et al., 2019], quantum Generative Adversarial Networks (GANs) [Romero and Aspuru-Guzik, 2021], and quantum neural network architectures [Beer et al., 2020].

In this thesis, we focus on generative models, which have garnered significant attention in recent years. Generative modeling is an unsupervised statistical learning task to learn a probability distribution that can generate a given dataset. The methodologies of quantum generative modeling are as follows. Let $\{x^{(i)}\}_{i=1}^{D}$ be a dataset of size D sampled from a probability distribution $q(x)$. The goal is to learn $q(x)$ as a parameterized probability distribution $p_\theta(x) = |\langle x| U(\theta)|\psi_0\rangle|^2$, obtained by applying $U(\theta)$ to an input

state and measuring in the computational basis. This corresponds to a quantum circuit Born machine [Benedetti et al., 2019]. In principle, the objective is to minimize the discrepancy between the two distributions. However, since $q(x)$ is not directly accessible, the cost function is defined using the negative log-likelihood defined by

$$C(\theta) = -\frac{1}{D}\sum_{=1}^{D} \log p_\theta(x^{(i)}). \tag{2.4}$$

The advancements in the study of quantum generative models are as follows. Benedetti et al. introduced a variational framework for training quantum circuit Born machines, demonstrated with classical and synthetic datasets, such as the bars-and-stripes dataset and synthetic datasets related to preparing GHZ states and coherent thermal states[Benedetti et al., 2019]. Liu and Wang investigated the generative model's representational power [Liu and Wang, 2018]. Coyle et al. showed that quantum circuit Born machines can simulate the restricted Boltzmann machine (RBM) and perform a sampling task difficult for classical computers [Coyle et al., 2020].

2.5 Critical challenges during the execution of VQAs

2.5.1 Barren plateau

Despite significant advancements in VQAs, numerous challenges remain that must be addressed to sustain the potential for quantum speedups. Particularly, the barren plateau phenomena in cost-function landscapes have emerged as a critical bottleneck for VQAs. When a cost function has a barren plateau, the amplitude of its partial derivatives diminishes exponentially with the number of qubits, making the landscape nearly flat. This requires exponentially high precision to overcome sampling noise and find a cost-minimizing direction, regardless of whether a gradient-based [Cerezo and Coles, 2021] or gradient-free [Arrasmith et al., 2021] optimization method is employed. Such precision scaling could nullify the quantum advantage of VQAs. Therefore, examining the barren plateau phenomenon in VQAs is vital to preserving their potential quantum advantage.

The barren plateau phenomenon was first discovered in [McClean et al., 2018], showing that deep, unstructured parameterized quantum circuits experience the barren plateau phenomenon with random initialization. This occurs because the ansatz is problem-agnostic and must search an exponentially large space, making the probability of finding the solution through random initialization exponentially small. The analysis of barren plateaus has been extended to include shallow random layered ansatzes, confirming that barren plateau phenomena can arise even in shallow circuits. Furthermore, it has been verified that this phenomenon occurs in practical tasks that involve real-world data [Holmes et al., 2021, Sharma et al., 2022, Ortiz Marrero et al., 2021].

While previous studies have attributed the barren plateau phenomenon to the randomness of the ansatz, recent findings have identified a distinct phenomenon where noise can induce the barren plateau phenomenon, regardless of the ansatz used [Wang et al., 2021]. In this context, noise infiltrating the circuit gradually corrupts the state towards the fixed point of the noise model, typically the maximally mixed state. This noise-induced phenomenon emerges when the circuit depth is linear with or exceeds the number of qubits, affecting many commonly used ansatz.

2.5.2 Importance of parameter initialization

Randomly initializing an ansatz can cause the algorithm to start far from the solution, near a local minimum, or in a barren plateau region. The significance of parameter initialization has been demonstrated in various studies, such as [Zhou et al., 2020]. Therefore, selecting the optimal parameters at the beginning of optimization is crucial. Several initialization strategies have been proposed, and empirical evidence suggests that these methods outperform optimizations that begin with randomly initialized parameters.

One approach to initialization strategies involves limiting the randomness within the ansatz. Grant et al. proposed reducing circuit randomness by randomly selecting a subset of initial parameters and choosing the remaining parameters such that the entire circuit evaluates to the identity [Grant et al., 2019b]. Volkoff and Coles introduced an initialization method that effectively reduces the dimensionality of the hyperparameter space by correlating the ansatz parameters [Volkoff and Coles, 2021]. Skolik et al. suggested an optimization technique that incrementally adds layers to the circuit, optimizing shallow circuits first rather than optimizing all

parameters simultaneously [Skolik et al., 2021].

Another approach in initialization strategies involves utilizing classical computers for pre-training to select optimal initial parameters. Several studies suggest that pre-training with classical neural networks, such as Recurrent Neural Networks (RNNs) [Verdon et al., 2019] and Convolutional Neural Networks (CNNs) [Friedrich and Maziero, 2022b], can help avoid the barren plateau problem. A method for parameter initialization that leverages tensor networks, which are highly compatible with quantum computers, has also been proposed [Rudolph et al., 2023b]. This research suggests that optimizing tensor networks using classical resources and embedding these optimized networks into quantum circuits as initial parameters can circumvent the barren plateau problem. Numerical experiments have confirmed this method indeed avoids the barren plateau problem, regardless of the number of qubits or circuit depth. Numerical experiments have confirmed that the gradient magnitudes in VQAs remained stable, concluding that the barren plateau problem was successfully avoided.

Chapter 3

Tensor networks

Tensor networks are powerful mathematical frameworks that efficiently represent and manipulate high-dimensional data by decomposing tensors into interconnected lower-dimensional components [Bridgeman and Chubb, 2017, Orús, 2014]. They have been widely applied in various fields, including quantum many-body physics for approximating ground states of complex systems [White, 1992, Vidal, 2003, Xiang, 2023], machine learning for real data [Stoudenmire and Schwab, 2016, Han et al., 2018], and combinatorial optimization problems. Recently, increasing attention has been directed towards their compatibility with quantum computing, as their structure aligns well with quantum circuits and facilitates hybrid quantum-classical algorithms [Markov and Shi, 2008]. In this chapter, we begin by explaining the fundamentals of tensor network notation to fully leverage its advantages. Next, we introduce two of the most renowned tensor network forms, Matrix Product States (MPSs) and Tree Tensor Networks (TTNs), and discuss their useful property called canonical form. Finally, we provide an overview of previous research on embedding MPSs into quantum circuits.

3.1 Introduction to tensor network notation

The widespread application of tensor networks across various fields in recent years can be attributed to their simple and comprehensible notation. Tensor network notation provides a clear and intuitive way to represent complex tensor contractions and operations. They also facilitate the understanding of entanglement, correlation lengths, and other quantum state properties.

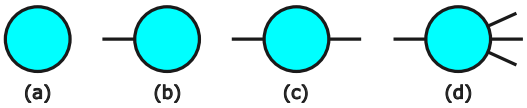

Figure 3.1: Some examples of tensors. (a) A scalar has no legs. (b) A vector has one leg. (c) A matrix has two legs. (d) A rank-4 tensor has four legs.

Moreover, the representation using tensor network notation is effective not only in quantum mechanics but also in all fields dealing with high-dimensional tensors, thus offering a more universal viewpoint. In this way, the utilization of tensor network notation offers numerous advantages.

Tensors are a generalization of vectors and matrices. In this thesis, we primarily address problems in quantum mechanics, assuming that the elements of tensors are complex numbers. A d-dimensional vector is an element of $\mathbb{C}^d$. A matrix with m rows and n columns is an element of $\mathbb{C}^{m\times n}$. Correspondingly, a rank-r tensor of dimensions $d_1 \times d_2 \times \cdots \times d_r$ is an element of $\mathbb{C}^{d_1\times d_2\times\cdots\times d_r}$. Scalars, vectors, and matrices correspond to rank 0, 1, and 2 tensors, respectively.

In tensor network notation, a tensor is depicted as a geometric shape with legs, each representing an index, much like the indices in Einstein notation. As illustrated in Fig. 3.1, a scalar has no legs, a vector has one leg, a matrix has two legs, a rank-4 tensor has four legs, and so on.

In some contexts, the geometric shape and the direction of the legs are determined by the properties of the tensor and its indices. When representing quantum states, the direction of the legs often indicates whether the vectors are in the Hilbert space for kets or its dual space. Following this convention easily prevents prohibited contractions, such as a contraction between two kets.

We will now explain the five key operations in tensor networks. First, we will explain the operations performed on a single tensor: partial trace, grouping, splitting, and singular value decomposition. Particularly, singular value decomposition is a crucial tool for harnessing the power of tensor networks in data compression. Then, we will explain the operations performed on multiple tensors: tensor product and contraction. Since the primary benefit of tensor network notation lies in its ability to create networks composed of multiple tensors, this will enable the full potential of tensor networks to be realized.

The first operation is the partial trace. For a tensor A with identical di-

Figure 3.2: Graphical examples of partial trace. When designating the left as the first axis, the upper right as the second axis, and the lower right as the third axis, (a) represents the equation $\sum_j A_{ijj}$, while (b) represents the equation $\sum_i A_{iik}$.

mensions for the x-th and y-th indices, the partial trace is a joint summation over that index:

$$\mathrm{Tr}_{x,y}(A)_{i_1,\ldots,i_{x-1},i_{x+1},\ldots,i_{y-1},i_{y+1},\ldots,i_r} = \sum_{\alpha=1}^{d_x} A_{i_1,\ldots,i_{x-1},\alpha,i_{x+1},\ldots,i_{y-1},\alpha,i_{y+1},\ldots,i_r}, \quad (3.1)$$

where r is the rank of A and d_x is the dimension of A's x-th index. In tensor network notation, this summation is shown by the corresponding legs being connected.

Figure 3.2 gives graphical examples of partial trace operation in tensor network notation. The advantage of partial trace notation in tensor network notation is that the summed-over indices do not need to be named. This simplifies the notation for large networks.

One example illustrating the convenience of partial trace notation is the cyclic property of the trace. As shown in Fig. 3.3, we can prove $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ by simply sliding the tensor to change its position in the network. Although this example is quite simple, the intuitive transformations of equations in tensor network notation become even more beneficial as the number of tensors increases.

The second and third operations we explain are grouping and splitting. The space of tensors $\mathbb{C}^{a_1 \times a_2 \times \cdots \times a_n}$ and $\mathbb{C}^b$ are isomorphic as vector spaces whenever $a_1 \times a_2 \times \cdots \times a_n = b$ holds. This allows us to apply concepts and techniques previously defined for vectors and matrices to all tensors. We can group indices to lower a tensor's rank or split them to raise it.

The definitions of grouping and splitting used in this thesis are as follows. Just as there are various ways to convert a matrix into a vector, there are multiple ways for grouping and splitting. We use an index convention that generalizes column-major ordering to higher dimensions. For simplicity, if we group all the indices of rank r tensor A, the resulting grouped tensor can

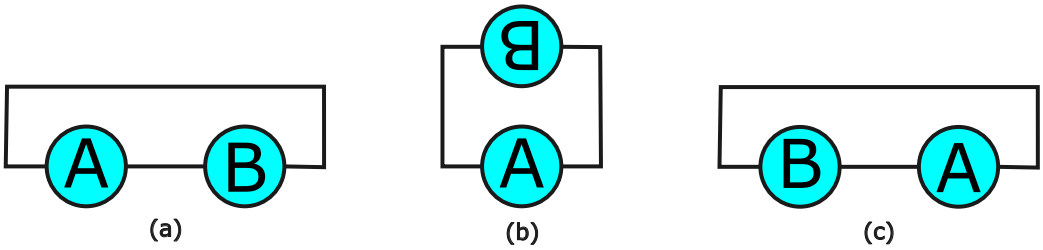

Figure 3.3: Diagrams illustrating the cyclic property of the trace using tensor networks. In (a), the equation $\mathrm{Tr}(AB)$ is represented in tensor network notation. Tensor network notation allows for free transformations as long as the information about which axes are connected is preserved. Thus, it can be transformed from (a) to (b) to (c). Since (c) represents the equation $\mathrm{Tr}(BA)$ in tensor network notation, the cyclic property of the trace is thereby demonstrated.

be written as

$$A_I \coloneqq A_{i_1,\ldots,i_n}, \tag{3.2}$$

where we have defined our grouped indices as

$$I \coloneqq i_1 + d_1 i_2 + d_1 d_2 i_3 + \cdots + (d_1 \times \cdots \times d_{n-1}) i_n, \tag{3.3}$$

and d_x is the dimension of the x-th index. According to this rule, we will refer to the process of combining indices as grouping and the process of separating indices as splitting.

In tensor network notation, grouping is represented by combining multiple indices into a single index, while splitting is represented by dividing a single index into multiple indices. Figure 3.4 depicts some examples of grouping and splitting. The resulting grouped index is often depicted with a bold line to distinguish it from the original indices.

The fourth operation we explain is the singular value decomposition (SVD). Any $M \times N$ matrix A can be decomposed as $A = USV^\dagger$ using SVD, where U is an $M \times M$ unitary matrix, $V^\dagger$ is an $N \times N$ unitary, matrix, and S is a real diagonal matrix with non-negative entries. SVD is a crucial algorithm for finding the best approximation. If we keep the largest k singular values in S to form $S^{(k)}$, then $A^{(k)} = US^{(k)}V^\dagger$ is the closest rank-k matrix to A in the Frobenius norm. This property is often used for dimensionality reduction while preserving as much information as possible.

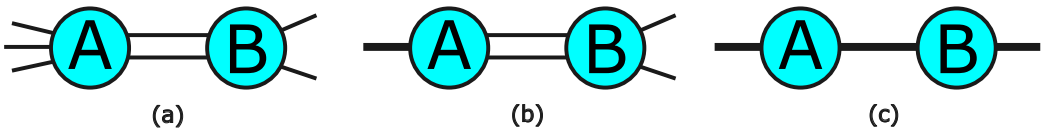

Figure 3.4: Graphical examples of grouping and splitting. By grouping the left indices in (a), we obtain (b). Further grouping the central and right indices in (b) results in (c). Since (c) represents a simple matrix product, the complex tensor network contraction is reduced to a matrix product. Conversely, the process of decomposing from (c) to (b) and then to (a) is known as splitting.

SVD for matrices can be extended to general tensors using the grouping and splitting algorithm. Let T be a tensor of rank $n+m$. By grouping n indices as rows and m indices as columns and then performing SVD, we get a higher dimensional version of SVD:

$$T_{i_1,\ldots,i_m;j_1,\ldots,j_n} = \sum_{\alpha} U_{i_1,\ldots,i_m,\alpha} S_{\alpha,\alpha} (V_{j_1,\ldots,j_n,\alpha})^*, \tag{3.4}$$

where U and V are isometries ($U^\dagger U = V^\dagger V = I$) across the indicated partitioning. Similar to the matrix case, if $S^{(k)}$ is the $k \times k$ matrix formed by retaining the largest k singular values in S, then

$$T^{(k)}_{i_1,\ldots,i_m;j_1,\ldots,j_n} = \sum_{\alpha}^{k} U_{i_1,\ldots,i_m,\alpha} S^{(k)}_{\alpha,\alpha} (V_{j_1,\ldots,j_n,\alpha})^* \tag{3.5}$$

is the rank-k tensor closest to T in terms of the Frobenius norm. This approach allows for dimensionality reduction in tensor networks while preserving essential information. Figure 3.5 represents the diagram of SVD for tensors.

Next, we consider operations involving multiple tensors. The fifth operation is the tensor product, which is a generalization of the outer product of vectors. The tensor product's value for a given set of indices is the elementwise product of the constituent tensor's values. Explicitly expressed in index notation, the binary tensor product takes the form:

$$[A \otimes B]_{i_1,\ldots,i_r,j_1,\ldots,j_s} \coloneqq A_{i_1,\ldots,i_r} \cdot B_{j_1,\ldots,j_s}. \tag{3.6}$$

Diagrammatically, the tensor product is represented by placing two tensors adjacent, as shown in Fig. 3.6. Therefore, the value of a network with disjoint tensors is simply the product of their values.

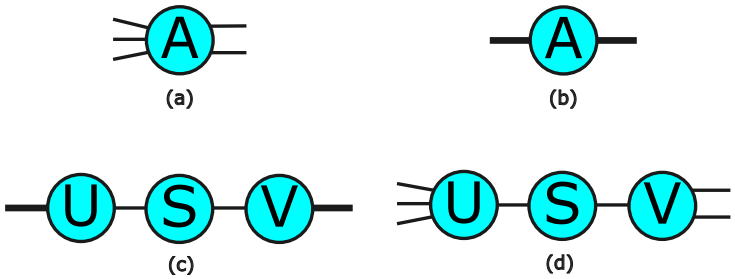

Figure 3.5: Diagrams for tensor SVD. For a general tensor A (a), by grouping the left and right axes respectively, it can be regarded as a matrix, as shown in (b). Applying SVD to the matrix results in its decomposition into unitary matrices U and V and a diagonal matrix S containing the singular values (c). By splitting indices, the SVD of the tensor is obtained while preserving the original tensor's axis information (d).

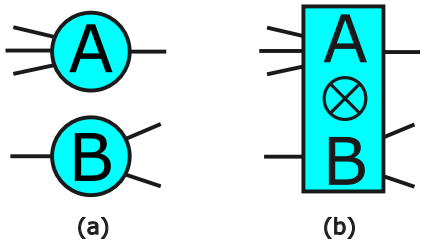

Figure 3.6: Diagrams representing the tensor product of two tensors. As shown in (a), two unconnected tensors are combined using the tensor product. This can also be explicitly denoted as a tensor product, as illustrated in (b).

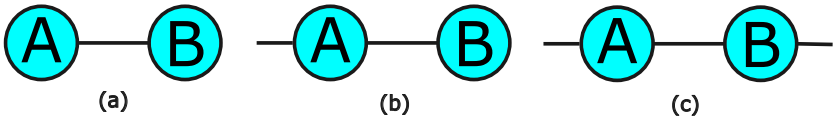

Figure 3.7: Some examples of the contraction. In (a), we observe the dot product of vectors, in (b), the product of a matrix and a vector, and in (c), the matrix product. Each of these represents a well-known special case of contraction.

The most frequently utilized operation is contraction, a tensor product followed by a partial trace over the indices of two tensors. The contraction of A's x-th index and B's y-th index is defined as

$$C_{i_1,\ldots,i_{x-1},i_{x+1},\ldots,i_r,j_1,j_{y-1},j_{y+1},\ldots,j_s} = \sum_{\alpha} A_{i_1,\ldots,i_{x-1},\alpha,i_{x+1},i_r} B_{j_1,\ldots,j_{y-1},\alpha,j_{y+1},\ldots,j_s}, \tag{3.7}$$

where C is the result of the contraction. The inner product of vectors, matrix-vector multiplications, and matrix-matrix multiplications are all contractions, as in Fig. 3.7.

By integrating the aforementioned tensor operations, we can define a tensor network as a diagram that specifies how to combine multiple tensors into a single composite tensor. The rank of this tensor is determined by the number of unmatched legs in the diagram. The value for a given set of external indices is the product of the constituent tensor's values, summed over all internal index labelings consistent with the contractions.

Although tensor networks are designed so their values do not depend on the order of tensor contractions, the sequence affects computational complexity and practicality. One can contract tensor networks by starting with one tensor and sequentially contracting it with others. This sequence, called bubbling, continues until the network is fully absorbed into the stored tensor, leaving only the final result. Many networks can be bubbled efficiently or inefficiently. We need to be careful when arranging the order of contractions.

As an example of how the order of contractions can significantly affect computational complexity, consider the bubbling of a ladder-shaped network that appears in calculations, such as the inner product of MPSs. One approach is to contract along the top of the ladder, then back along the bottom, illustrated in Fig. 3.8. This method is inefficient since, for a ladder of length n, the tensor tracked at the midpoint has rank n, causing the number of en-

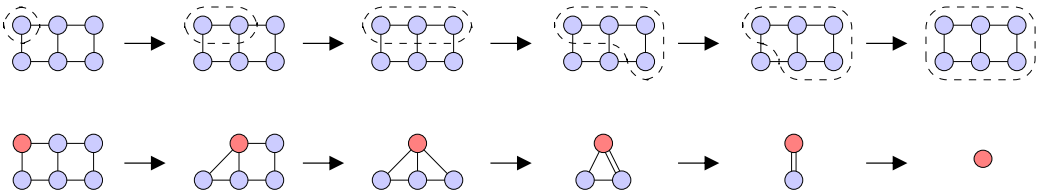

Figure 3.8: An example of inefficient bubbling in a ladder-shaped network. In this example, the contraction proceeds sequentially from the tensors in the upper row. However, the spatial computational complexity increases exponentially with the ladder length at the intermediate point. Consequently, for long ladders, current computers are unable to complete the contraction calculations.

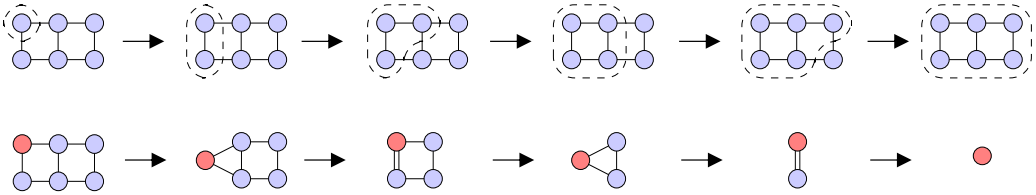

Figure 3.9: An example of efficient bubbling in a ladder-shaped network. For a ladder-shaped network, as in this example, sequentially contracting from the left achieves a computational complexity that is linear with respect to the ladder length.

tries to scale exponentially. As a result, the memory and time needed for this contraction are exponential, making it impractical for large n. However, by contracting each rung in sequence, the tracked tensor's rank stays at three or below, leading to constant memory and linear time costs depicted in Fig. 3.9.

Although our example shows bubbling where one tensor contracts others one by one, this is not always the most efficient method; a multi-bubbling approach is often faster. It is crucial to determine the appropriate grouping of parts, the starting tensor, and the contraction sequence to reduce computational complexity.

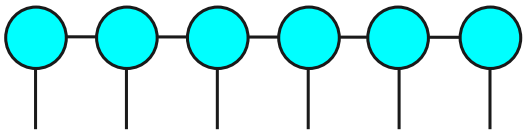

Figure 3.10: MPS represented in tensor network notation, specifically for a system size of six.

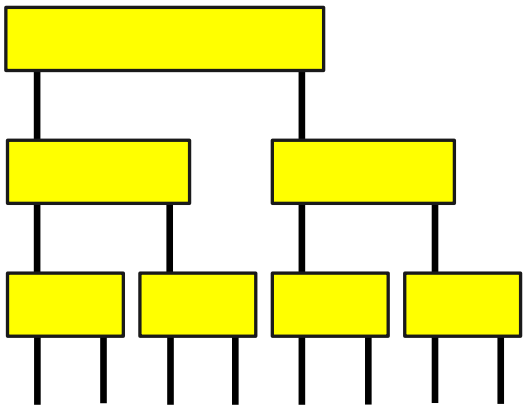

Figure 3.11: TTN represented in tensor network notation, specifically for a binary tree with a system size of eight.

3.2 MPSs and TTNs

As illustrated in Fig. 3.10, Matrix Product States (MPSs) represent the simplest form of tensor networks, with tensors arranged in a single line. Advanced algorithms such as Density Matrix Renormalization Group (DMRG) and Time-Evolving Block Decimation (TEBD) have been developed, facilitating their application across various fields. In contrast, Tree Tensor Networks (TTNs), depicted in Fig. 3.11, have a more complex tree-like structure. It is evident that MPSs can be considered a subset of TTNs. One notable advantage of TTNs is that the distance between any pair of leaf nodes scales logarithmically with the number of nodes, in contrast to the linear scaling observed in MPSs. Since connected correlation functions in tensor networks typically decay exponentially with path length, TTNs are inherently more effective at capturing long-range correlations and are particularly well-suited for representing higher-dimensional systems. As a result, TTNs have found applications in diverse domains, including quantum chemistry [Murg et al., 2010, Gunst et al., 2019], condensed matter physics [Nagaj et al., 2008, Tagliacozzo et al., 2009, Lin et al., 2017], and machine learning [Liu et al., 2019a, Cheng et al., 2019].

The coefficients representing the state vector as a linear combination of a given basis form a tensor, which can be expressed using a tensor network. Let $|\psi\rangle$ be a tensor network in the form of either an MPS or a TTN. Number the tensors in $|\psi\rangle$ from left to right for MPS, and in breadth-first search (BFS) order from the root node for TTN, denoting the i-th tensor as $A^{(i)}(i = 1, \ldots, N)$. By adding a leg with bond dimension one, a two-legged tensor can be converted into a three-legged tensor, thus all tensors in $|\psi\rangle$ can be considered three-legged. $|\psi\rangle$ is in canonical form if there exists a node $A^{(i)}$, called the canonical center, such that for all other nodes $A^{(i')}$ the following holds

$$\sum_{l,m} A^{(i')}_{l,m,n} A^{(i')*}_{l,m,n'} = I_{n,n'}, \tag{3.8}$$

where the leg denoted by index n is the unique leg of $A^{(i')}$ pointing towards $A^{(i)}$. A tensor that satisfies this equation is referred to as an isometric tensor. Any $|\psi\rangle$ can be transformed into this canonical form, and the position of the canonical center can be freely moved without changing the quantum state.

The canonical form offers numerous advantages. In this thesis, the key benefit is that at the canonical center, the local SVD becomes equivalent

to the global SVD, allowing for precise bond dimension reduction through truncation while appropriately moving the canonical center. Additionally, in the canonical form, each tensor is an isometry, facilitating easy conversion to unitary form and embedding into quantum circuits. Unless otherwise specified, we assume that any MPSs and TTNs are converted to the canonical form with $A^{(0)}$ as the canonical center.

3.3 Embedding of MPSs

Although embedding general tensor networks into shallow quantum circuits is challenging, several workable methods have been proposed for MPSs. In this subsection, we overview the technique for seamlessly embedding MPSs into shallow quantum circuits.

Ran introduced a systematic decomposition method for an MPS into several layers of two-qubit gates with a linear next-neighbor topology [Ran, 2020]. First, the MPS $|\psi^{(k)}\rangle$ with bond dimension χ is truncated to a bond dimension two MPS $|\psi^{(k)}_{\chi=2}\rangle$. Next, the isometric tensors in $|\psi^{(k)}_{\chi=2}\rangle$ are converted into unitary tensors. The resulting set of unitary tensors, $L[U]^{(k)}$, is referred to as a layer and can be embedded into a quantum circuit composed of two-qubit gates. Additionally, since

$$|\psi^{(k)}_{\chi=2}\rangle = \mathrm{L}[U]^{(k)} |0\rangle \tag{3.9}$$

and

$$\mathrm{L}[U]^{(k)\dagger} |\psi^{(k)}_{\chi=2}\rangle = |0\rangle \tag{3.10}$$

are hold, $L[U]^{(k)\dagger}$ can be considered a disentangler, transforming the quantum state into a product state. Finally, a new MPS $|\psi^{(k+1)}\rangle = \mathrm{L}[U]^{(k)\dagger} |\psi^{(k)}\rangle$ can be obtained. Since $L[U]^{(k)\dagger}$ acts as a disentangler, $|\psi^{(k+1)}\rangle$ should have reduced entanglement compared to $|\psi^{(k)}\rangle$. However, this operation involves the transformation of the tensor network, which introduces approximation errors in practical calculations. By repeating this process starting from the original MPS $|\psi^{(0)}\rangle$, a quantum circuit $\prod_{k=1}^{K} \mathrm{L}[U]^{(k)} |0\rangle$ with multiple layers can be generated.

Optimization-based methods are employed as an alternative approach to embedding tensor networks into quantum circuits. This method sequentially optimizes the unitary operators within the quantum circuit to maximize the magnitude of the inner product between the quantum circuit

and the tensor network. Evenbly and Vidal proposed an iterative optimization technique that utilizes the calculation of environment tensors and SVD [Evenbly and Vidal, 2009]. Similarly, Shirakawa et al. employed this iterative optimization method to embed quantum states into quantum circuits [Shirakawa et al., 2024]. An environment tensor is calculated to update a unitary by removing the unitary from the circuit and contracting the remaining tensor network. We perform the SVD of the environment tensor and utilize the fact that the resulting unitary matrix serves as the optimal operator to increase the fidelity between the original tensor network and the constructed quantum circuit. For a more detailed explanation of the optimization algorithm, please refer to [Rudolph et al., 2023a, Shirakawa et al., 2024] or the chapter on optimization algorithms for TTNs described later.

Rudolph et al. investigated the optimal sequence for combining these two methods to achieve the highest accuracy [Rudolph et al., 2023a]. The study concludes that the best results are obtained by adding new layers through a systematic decomposition step and optimizing all layers with each addition. This method has achieved exceptional accuracy in embedding a wide range of MPSs in fields such as physics, machine learning, and random systems. Therefore, it can be considered one of the best current techniques for embedding MPSs into shallow circuits. However, this method does not support embedding tensor networks other than MPSs. Consequently, embedding tensor networks such as TTNs and other non-MPS structures remains an unresolved issue.

Chapter 4

Embedding of TTNs to shallow quantum circuits

In this section, we propose a method for embedding TTNs into shallow quantum circuits. The proposed method can be generalized to embed TTNs of any shape since any shape of TTN can be converted into a binary tree using SVD. Below, we assume a binary tree for simplicity. The fundamental approach of the proposed method is similar to that of MPSs [Rudolph et al., 2023a]. By repeatedly adding layers using the systematic decomposition algorithm and optimizing the entire circuit, we generate highly accurate embedded quantum circuits. This chapter first introduces the systematic decomposition algorithm for TTNs. Due to the increased structural complexity of TTNs compared to MPSs, the contraction methods become non-trivial. We propose a method that balances approximation error and computational cost. Next, we describe the optimization algorithm for TTNs and integrate it with the systematic decomposition. Finally, we discuss the computational cost and demonstrate that embedding TTNs is feasible within practical computational limits.

4.1 Overview of this method

4.1.1 Systematic decomposition by disentangling

Our systematic decomposition algorithm for TTNs is an extension of the algorithm for MPSs introduced in [Ran, 2020]. We denote the original TTN

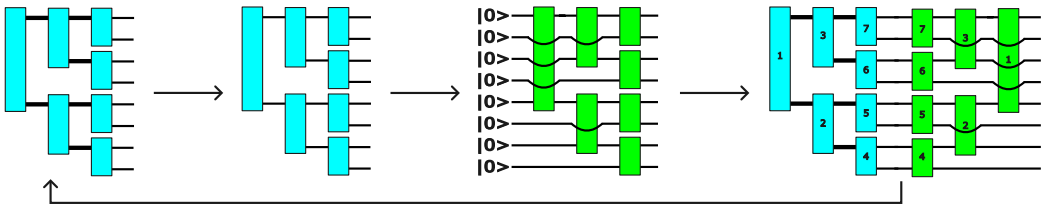

Figure 4.1: A schematic diagram illustrating the systematic decomposition for a TTN. The TTN is truncated to a bond dimension of two using SVD. This simplified TTN is then embedded into a quantum circuit by converting tensors into unitaries via Gram-Schmidt orthogonalization. These unitaries act as disentanglers and are applied to the original TTN. By repeating these steps, a multi-layer quantum circuit is formed.

Input: TTN $|\psi_0\rangle$, Maximum number of layers K
Output: Quantum Circuit $\prod_{k=1}^{K} \mathrm{L}[U]^{(k)} |0\rangle$
$|\psi^{(1)}\rangle \leftarrow |\psi_0\rangle$;
for $k = 1$ *to* K **do**
  Truncate $|\psi^{(k)}\rangle$ to $|\psi^{(k)}_{\chi=2}\rangle$ via SVD;
  Convert $|\psi^{(k)}_{\chi=2}\rangle$ to $\mathrm{L}[U]^{(k)}$;
  $|\psi^{(k+1)}\rangle \leftarrow \mathrm{L}[U]^{(k)\dagger} |\psi^{(k)}\rangle$;
end

Algorithm 1: Systematic decomposition of TTN.

as $|\psi_0\rangle$, the k-th layer of the quantum circuit as $\mathrm{L}[U]^{(k)}$, the number of layers in resulting quantum circuit as K, and the resulting quantum circuit as $\prod_{k=1}^{K} \mathrm{L}[U]^{(k)} |0\rangle$. Algorithm 1 details the systematic decomposition process for TTNs, as depicted in Fig. 4.1.

In this algorithm, a copy of the original TTN is truncated to a lower dimension using SVD. Converting a TTN with bond dimension χ to bond dimension two can be done accurately, similar to MPSs, by shifting the canonical center so that the local SVD matches the global SVD. This truncated TTN is then transformed into a single layer of two-qubit gates by converting the isometric tensors in the layer into unitary tensors using the Gram-Schmidt orthogonalization process. The inverse of this layer is applied to the original TTN, resulting in a partially disentangled state with potentially reduced entanglement and bond dimensions. This process can be iteratively repeated to generate multiple layers, which are indexed in reverse order to

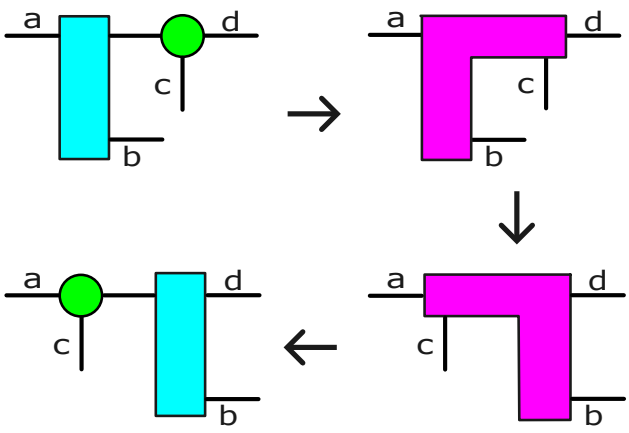

Figure 4.2: Diagrams of the penetration algorithm. It contracts two connected tensors, reorders axes, and separates them using SVD, making it seem like their positions have swapped.

form a circuit that approximates the target TTN. Notably, the final layer of the disentangling circuit $\mathrm{L}[U]^{(k)}$ is used as the initial layer of the quantum circuit for approximation.

While this algorithm appears to function effectively at first glance, the computation of $|\psi^{(k+1)}\rangle \leftarrow \mathrm{L}[U]^{(k)\dagger} |\psi^{(k)}\rangle$ is exceedingly challenging from the perspective of tensor networks. The structure of $|\psi^{(k+1)}\rangle$ is assumed to be TTN, whereas tensor network $\mathrm{L}[U]^{(k)\dagger} |\psi^{(k)}\rangle$ has a more complex structure, necessitating its transformation into the shape of TTN. As previously mentioned, the naive transformation of a general tensor network requires exponentially large memory relative to the number of qubits. Additionally, although limiting the bond dimension during the transformation can facilitate the process, the sequence of transformations can lead to substantial approximation errors.

To address the issue of tensor network transformations arising from the complexity of TTN structures, we propose a penetration algorithm as a submodule. As illustrated in Fig. 4.2, the penetration algorithm operates by contracting two tensors connected by a single edge along the connected axis, appropriately reordering the axes, and then separating them using SVD. This makes it appear as if the positions of the two tensors have been swapped. Additionally, by adjusting the number of singular values retained during SVD, we can balance the approximation accuracy and the computational cost.

Input: $\mathrm{L}[U]^{(k)\dagger}\,|\psi^{(k)}\rangle$
Output: TTN $|\psi^{(k+1)}\rangle$
for $i = N-1$ *to* 1 **do**
  Split $\mathrm{L}[U]_i^{(k)\dagger}$ into U_p and L_o via SVD;
  while U_p *is not connected to* $A^{(i)}$ **do**
    $A \leftarrow U_p$'s left tensor;
    Make U_p penetrate A;
  end
  while L_o *is not conntected to* $A^{(i)}$ **do**
    $A \leftarrow L_o$'s left tensor;
    Make L_o penetrate A;
  end
  $A^{(i)} \leftarrow$ Contract $A^{(i)}$, U_p, and L_o;
end

Algorithm 2: Transformation of tensor networks using the penetration algorithm.

Algorithm 2 details the transformation of tensor networks using the penetration algorithm, as depicted in Fig. 4.3. We denote i-th tensor of $|\psi^{(k)}\rangle$ as $A^{(i)}$, where the numbers are assigned to the tensors of $|\psi^{(k)}\rangle$ in breadth-first search (BFS) order from the root. We also assign numbers to the tensors of $\mathrm{L}[U]^{(k)\dagger}$ based on its origin in the TTN, denoting the tensor with number i as $\mathrm{L}[U]_i^{(k)\dagger}$. For each tensor $\mathrm{L}[U]_i^{(k)\dagger}$ in $\mathrm{L}[U]^{(k)\dagger}$, first use SVD to split it into upper tensor U_p and lower tensor L_o to reduce the computational complexity in the penetration algorithm. Then, apply the penetration algorithm iteratively until the upper tensor is connected with $A^{(i)}$. Repeat this for the lower tensor, too. Finally, contract the upper and lower tensors with $A^{(i)}$ to form a new TTN's i-th tensor. Perform this process sequentially from the highest-numbered tensor in $\mathrm{L}[U]^{(k)\dagger}$. This algorithm allows contraction with each tensor in $\mathrm{L}[U]^{(k)\dagger}$ without changing the TTN structure of $|\psi^{(k)}\rangle$. Consequently, despite minor approximation errors during penetration, $\mathrm{L}[U]^{(k)\dagger}\,|\psi^{(k)}\rangle$ can be transformed into a TTN structure $|\psi^{(k+1)}\rangle$.

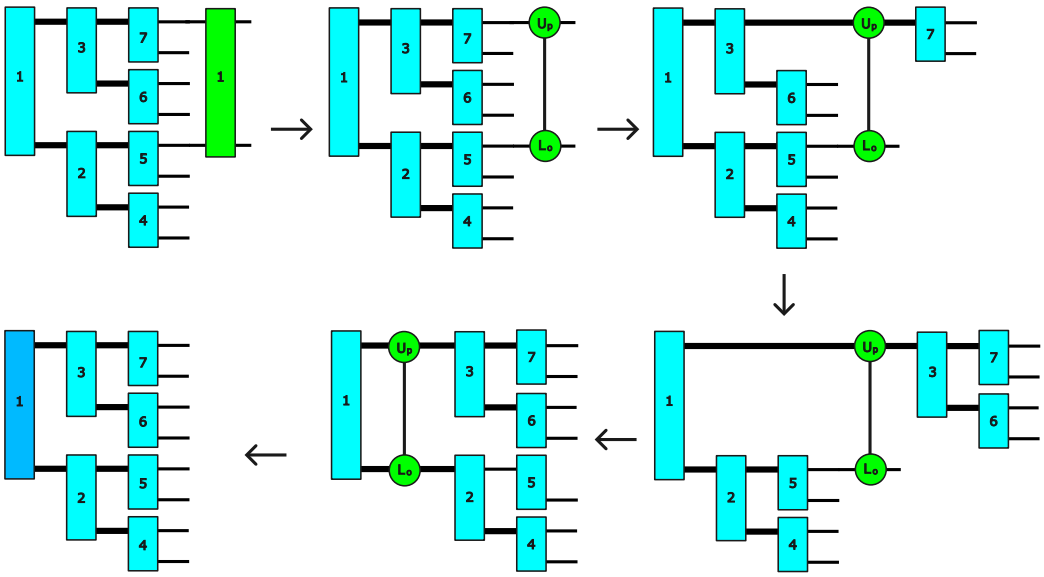

Figure 4.3: A schematic diagram explaining the transformation from a complex tensor network into a TTN. We number the tensors based on their positions in the network. We first use SVD to split it into an upper and a lower tensor. Then, we apply the penetration algorithm iteratively until the upper tensor is connected with the corresponding tensor. This process is repeated for the lower tensor as well. Finally, we contract the upper and lower tensors with the corresponding tensor to form a new tensor in the TTN. This process is performed sequentially from the highest-numbered component of the disentangler.

Input: Quantum Circuit $\prod_{k=1}^{K} \mathrm{L}[U]^{(k)} |0\rangle$, number of sweeps T, learning rate $r \in [0, 1]$
Output: Optimized Quantum Circuit $\prod_{k=1}^{K} \mathrm{L}[U]^{(k)} |0\rangle$
for $t = 1$ *to* T **do**
  for i *to* $N - 1$ **do**
    for $k = 1$ *to* K **do**
      $U_{old} \leftarrow \mathrm{L}[U]_i^{(k)}$;
      Calculate environment tensor E;
      SVD $E = USV^\dagger$;
      $U_{new} \leftarrow UV^\dagger$;
      $\mathrm{L}[U]_i^{(k)} \leftarrow U_{old}(U_{old}^\dagger U_{new})^r$;
    end
  end
end

Algorithm 3: Optimization.

4.1.2 Combination of systematic decomposition and optimization

As demonstrated in [Rudolph et al., 2023a], the systematic decomposition algorithm achieves the highest embedding accuracy when appropriately combined with the optimization algorithm. In this study, we also integrate the systematic decomposition with the optimization algorithm. The fundamental concept of the optimization algorithm for quantum circuits embedded with TTNs is analogous to that for quantum circuits embedded with MPS. The primary difference lies in the ansatz of the quantum circuits; however, the optimization algorithm can be executed similarly.

Algorithm 3 details the optimization process for TTNs, as depicted in Fig. 4.4. The environment tensor E is obtained by contracting all tensors except for the one of interest. In this algorithm, the tensor of interest is $\mathrm{L}[U]_i^{(k)}$ and E becomes a rank-4 tensor with two legs on the left and two on the right. We compute the SVD of E and utilize the fact that the product $UV^\dagger$ is the unitary matrix that maximizes the magnitude of the inner product between the original TTN and the generated quantum circuit [Shirakawa et al., 2024]. Given the strength of this local update, we introduce a learning rate r, which modifies the unitary update rule via $U_{old}(U_{old}^\dagger U_{new})^r$. Replacing $\mathrm{L}[U]_i^{(k)}$ with

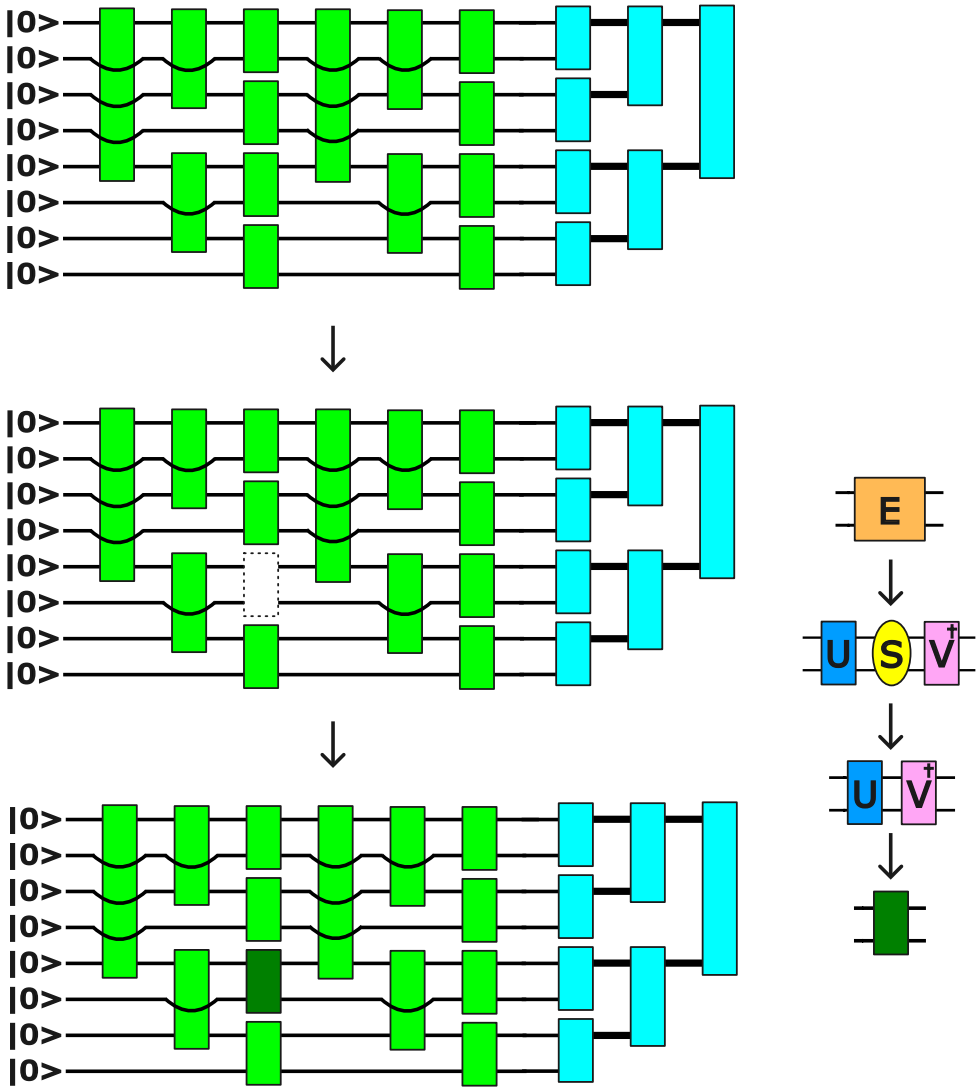

Figure 4.4: Optimization procedure of tensors. Consider the tensor network formed by taking the inner product between the generated quantum circuit and the original TTN. The original TTN is depicted in light blue, while the generated quantum circuit is shown in light green. The environment tensor E is obtained by contracting all tensors except the one of interest. We compute its SVD $E = USV^\dagger$ and calculate $UV^\dagger$ to find the unitary operator closest to the environment tensor. The generated unitary operator is positioned at the location of the removed unitary operator.

Input: TTN ψ_0, Maximum number of layers K
Output: Quantum Circuit $\prod_{k=1}^{K} \mathrm{L}[U]^{(k)} |0\rangle$
$|\psi\rangle \leftarrow |\psi_0\rangle$;
for $k = 1$ *to* K **do**
  Truncate $|\psi\rangle$ to $|\psi_{\chi=2}\rangle$ via SVD;
  Convert $|\psi_{\chi=2}\rangle$ to $\mathrm{L}[U]^{(k)}$;
  Optimize $\prod_{k'=1}^{k} \mathrm{L}[U]^{(k')}$;
  for $j = 1$ *to* k **do**
    Absorb $\mathrm{L}[U]^{(j)\dagger}$ into $|\psi\rangle$;
  end
end

Algorithm 4: $[D_i, O_{all}]$ method for TTN.

the unitary operator calculated in this manner for all operators constitutes one step, and repeating this process for T steps completes the optimization algorithm.

The integration of the systematic decomposition and optimization algorithms involves a method where a new layer is added using the systematic decomposition algorithm, followed by optimizing the entire quantum circuit with the optimization algorithm. This process is repeated iteratively. Rudolph et al. refers to this method as Iter$[D_i, O_{all}]$ [Rudolph et al., 2023a], confirming that it achieves the highest accuracy regardless of the type of MPSs. When creating the $(k+1)$-th layer using the systematic decomposition algorithm, the layers up to k are first absorbed into the original TTN using the penetration algorithm before executing the systematic decomposition algorithm. It should be noted that since we are using the penetration algorithm, the $|\psi\rangle$ that has absorbed $L[U]^{j\dagger}$ retains its TTN structure. The integrated algorithm is presented as Algorithm 4.

4.2 Computational complexity

For a TTN with a maximum bond dimension χ, the memory requirements scale as $\mathcal{O}(N\chi^3)$. However, the computational complexity of transforming the TTN into its canonical form scale as $\mathcal{O}(N\chi^4)$. Most of the algorithms related to TTNs require transformation into the canonical form. Therefore, it is reasonable to assume that we set χ such that computations of $\mathcal{O}(N\chi^4)$

can be performed efficiently.

In Algorithm 4, truncating $|\psi\rangle$ to $|\psi_{\chi=2}\rangle$ requires $\mathcal{O}(N\chi^3)$. Optimizing the entire circuit is significantly more challenging compared to MPS due to the complexity of the TTN structure. Generally, exponential memory is required with respect to N, but by using an appropriate contraction order and caching, the computation can be performed in $\mathcal{O}(N\chi^3 4^K)$, which is linear in N. Although the computational complexity increases exponentially with the number of layers, this algorithm is designed for embedding into shallow quantum circuits, and it operates efficiently for $K \approx 7$, as used in our experiments. Details on the techniques explained here are provided in [Inomata et al., 2025]. Furthermore, when embedding TTNs into a larger number of layers, it is possible to achieve $\mathcal{O}(N \log N\chi^4 KT)$, where T is the number of sweeps, linear computational complexity with respect to the number of layers, bond dimension by introducing approximations that prevent the bond dimension from exceeding χ during the contraction of environment tensors, as used in the embedding of MPS [Rudolph et al., 2023a].

The computational complexity of absorbing K layers into the original TTN is $\mathcal{O}(N \log N\chi^4 K)$. In the penetration algorithm, SVD of a matrix with row dimension χ^2 and column dimension χ has a complexity of $\mathcal{O}(\chi^4)$. Given that the depth of the TTN is $\mathcal{O}(\log N)$, the penetration operation is performed $\mathcal{O}(\log N)$ times per tensor. Since each layer contains $\mathcal{O}(N)$ tensors, the complexity of absorbing one layer is $\mathcal{O}(N \log N\chi^4)$, leading to a total complexity of $\mathcal{O}(N \log N\chi^4 K)$ for K layers. In summary, the computational complexity to generate one layer is $\mathcal{O}(\max(N \log N\chi^4 K, N\chi^3 4^K))$, and it is $\mathcal{O}(\max(N \log N\chi^4 K^2, N\chi^3 4^K))$ for K layers. It scales polynomially with the number of qubits. Additionally, the bond dimension is constrained to χ^4, making it suitable for embedding TTNs with large bond dimensions. Furthermore, by allowing approximations in optimization, the complexity can be reduced to $\mathcal{O}(N \log N\chi^4 K^2 T)$ for K layers.

Chapter 5

Numerical Results

We conducted numerical experiments using two distinct state vectors from the fields of machine learning and physics. The first state vector represents a uniform superposition over the binary data samples in the 4×4 bars and stripes (BAS) dataset, which has become a canonical benchmark for generative modeling tasks in quantum machine learning. The second state vector represents the ground state of the J_1-J_2 Heisenberg model, a model that characterizes competing interactions in quantum spin systems. The Hamiltonian for this model is given by

$$H = J_1 \sum_{<i,j>} \boldsymbol{S}_i \cdot \boldsymbol{S}_j + J_2 \sum_{<<i,j>>} \boldsymbol{S}_i \cdot \boldsymbol{S}_j, \tag{5.1}$$

where $J_1(J_2)$ represents the nearest (next-nearest) neighbor interactions. By varying the ratio of the first and second nearest-neighbor interactions, the J_1-J_2 Heisenberg model exhibits complex quantum many-body phenomena and has been widely studied. In this thesis, we utilized $J_2/J_1 = 0.5$. Both state vectors have two-dimensional correlations, making them more suitably represented by TTNs rather than MPSs.

The state vector of the BAS was manually prepared, while the ground state of the J_1-J_2 Heisenberg model was generated using a numerical solver package $\mathcal{H}\Phi$ [Kawamura et al., 2017], which is designed for a wide range of quantum lattice models. MPSs and TTNs were constructed by iteratively applying SVD to the state vector from the edges.

The conversion from MPSs and TTNs to quantum circuits employed the Iter[D_i, O_{all}] method, which is also utilized in the proposed method. Additionally, to compare with the proposed method, we also used D_{all}, O_{all}, and

Iter[I_i, O_{all}] from [Rudolph et al., 2023a]. The D_{all} method generates circuits solely through systematic decomposition. The O_{all} method optimizes circuits starting from an initial state composed only of identity gates. The Iter[I_i, O_{all}] method sequentially adds identity layers, optimizing the entire circuit at each step.

To measure the accuracy of embedding into quantum circuits, we utilized infidelity as the evaluation metric. The infidelity I_f between two quantum states, $|\Psi\rangle$ and $|\Phi\rangle$, is expressed as

$$I_f = 1 - |\langle\Phi|\Psi\rangle|, \tag{5.2}$$

and a smaller infidelity indicates that the two quantum states are closer. In this study, infidelity quantifies the success of our transformations.

Figure 5.1 illustrates the infidelity between the original state vector and the quantum circuits embedded with either TTN or MPS. Despite the greater difficulty in embedding TTNs into shallow quantum circuits due to their hierarchical structure, the infidelities between original state vectors and generated quantum circuits from TTNs are sufficiently low, indicating the successful embedding of the TTNs into shallow quantum circuits. Notably, in the BAS dataset, the TTN's superior representational capacity enables more accurate embeddings than MPS.

The result of the J_1-J_2 Heisenberg model reveals an intriguing pattern: TTN outperforms in both shallow and deep quantum circuits, while MPS excels in the intermediate region. In deep quantum circuits, the superior representational capacity of TTN results in enhanced convergence accuracy. For shallow and intermediate quantum circuits, this result can be interpreted as a trade-off between representational capacity and approximation error. Generally, TTN surpasses MPS in representational capacity, enabling the depiction of states that are inadequate for MPS. However, embedding TTN introduces larger approximation errors than MPS. Consequently, in the shallow region, TTN prepares a better initial circuit due to its representational advantage outweighing the approximation error. Conversely, in the intermediate region, MPS prepares a better initial circuit as the approximation error outweighs the representational capacity. As the number of qubits in quantum computers increases, the difference in representational capacity between TTN and MPS will become more pronounced, and the impact of approximation errors will relatively diminish.

Figure 5.2 highlights the importance of the systematic decomposition algorithm for embedding TTNs into quantum circuits. As demonstrated in

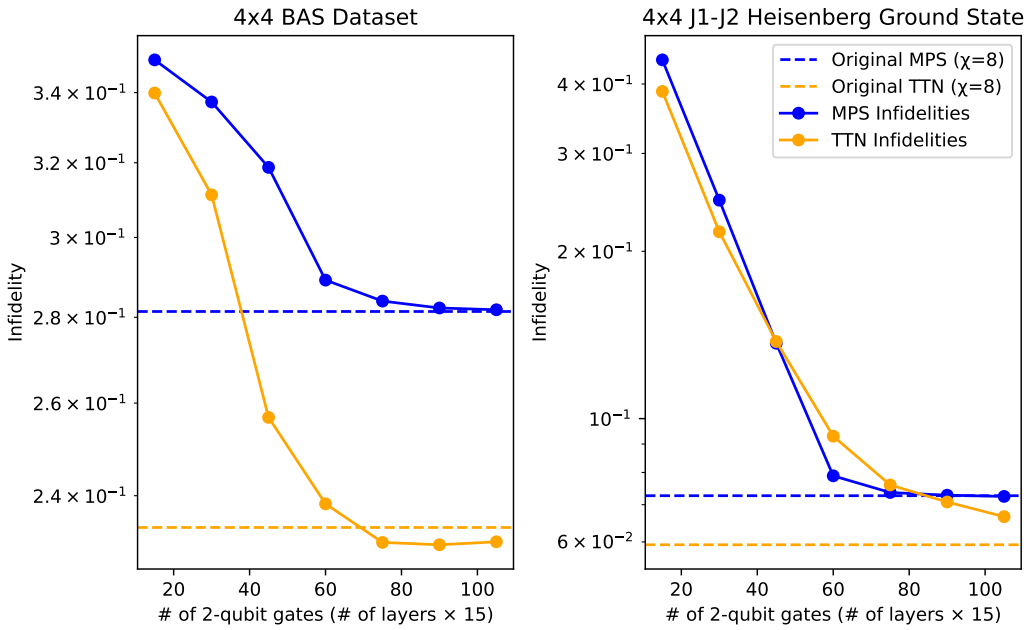

Figure 5.1: Infidelities between original state vectors and generated quantum circuits. The bond dimensions of tensor networks are 8, and the number of optimization steps T is $1,000$. The learning rate r for optimization is set to 0.65 for the BAS dataset and 0.6 for the J_1-J_2 Heisenberg model. The MPS results are represented by the blue lines, while the TTN results are depicted by the orange lines. Additionally, the infidelity between the original tensor network and the state vector is indicated by the dotted lines.

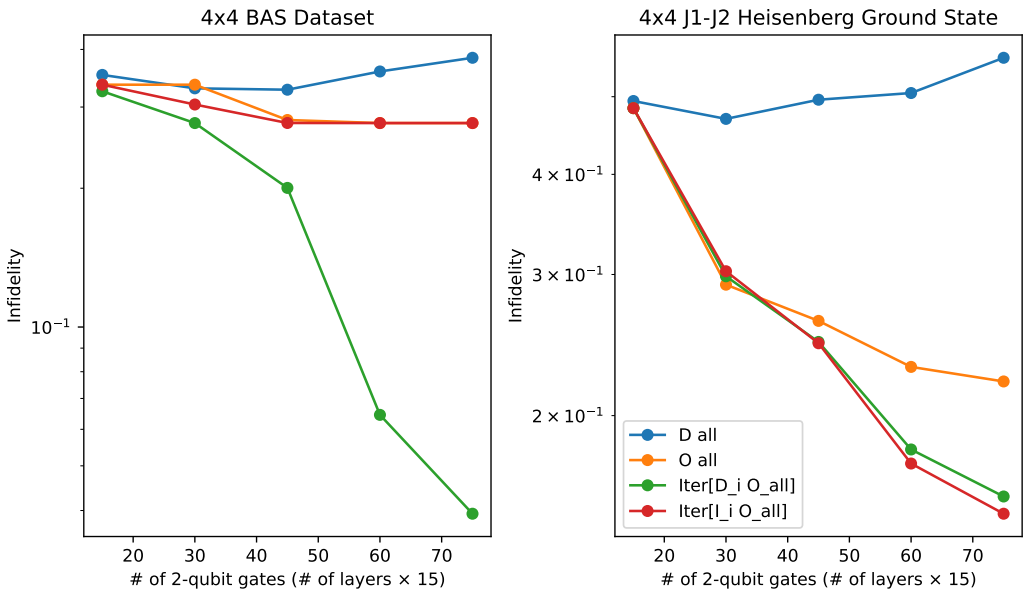

Figure 5.2: Infidelities between original TTNs and generated quantum circuits. The bond dimensions of TTNs are 16, and the number of optimization steps is 1,000. The learning rate for optimization is set to 0.6.

previous research on MPS [Rudolph et al., 2023a], methods such as Iter[D_i, O_{all}] and Iter[I_i, O_{all}], which sequentially add layers and optimize all circuits, achieve high accuracy. Furthermore, as suggested in previous studies, it was confirmed that Iter[I_i, O_{all}] struggles with the BAS dataset even in TTN embeddings, where the singular values at the bond become discontinuous. This indicates that Iter[D_i, O_{all}] is the most effective and robust embedding method, achieving high accuracy in multiple fields, including machine learning and physics, thus affirming the critical importance of the systematic decomposition algorithm. This result also verifies the effective performance of the proposed systematic decomposition method.

Chapter 6

Conclusion

To avoid the barren plateau issue in VQAs, there is increasing interest in using tensor networks to initialize quantum circuits. However, embedding tensor networks into shallow quantum circuits is generally difficult, and prior research has been limited to embedding MPSs. In this study, we proposed a method for embedding TTNs, which have a more complex structure than MPSs and can efficiently represent higher-dimensional systems and systems with long-range correlations into shallow quantum circuits composed solely of two-qubit gates. We applied our proposed method to different types of TTNs and confirmed that it prepares quantum circuit parameters with better accuracy than embedding MPSs. Additionally, the computational complexity is $\mathcal{O}(\max(N \log N \chi^4 K^2, N \chi^3 4^K))$, or $\mathcal{O}(N \log N \chi^4 K^2 T)$ with approximation, making it applicable to practical problems. As the number of qubits increases in the future, the difference in representational power between TTN and MPS will become more pronounced, thereby enhancing the importance of methods for embedding TTN into quantum circuits. This study will be an important bridge for implementing hybrid algorithms combining tree tensor networks and quantum computing.

Bibliography

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

[Aggarwal et al., 2017] Aggarwal, D., Brennen, G. K., Lee, T., Santha, M., and Tomamichel, M. (2017). Quantum attacks on bitcoin, and how to protect against them. *arXiv preprint arXiv:1710.10377*.

[Arrasmith et al., 2021] Arrasmith, A., Cerezo, M., Czarnik, P., Cincio, L., and Coles, P. J. (2021). Effect of barren plateaus on gradient-free optimization. *Quantum*, 5:558.

[Aspuru-Guzik et al., 2005] Aspuru-Guzik, A., Dutoi, A. D., Love, P. J., and Head-Gordon, M. (2005). Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707.

[Beer et al., 2020] Beer, K., Bondarenko, D., Farrelly, T., Osborne, T. J., Salzmann, R., Scheiermann, D., and Wolf, R. (2020). Training deep quantum neural networks. *Nature communications*, 11(1):808.

[Benedetti et al., 2019] Benedetti, M., Garcia-Pintos, D., Perdomo, O., Leyton-Ortega, V., Nam, Y., and Perdomo-Ortiz, A. (2019). A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information*, 5(1):45.

[Biamonte et al., 2017] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671):195–202.

[Bridgeman and Chubb, 2017] Bridgeman, J. C. and Chubb, C. T. (2017). Hand-waving and interpretive dance: an introductory course on tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 50(22):223001.

[Cerezo and Coles, 2021] Cerezo, M. and Coles, P. J. (2021). Higher order derivatives of quantum neural networks with barren plateaus. *Quantum Science and Technology*, 6(3):035006.

[Cerezo et al., 2021] Cerezo, M., Sone, A., Volkoff, T., Cincio, L., and Coles, P. J. (2021). Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature communications*, 12(1):1791.

[Cheng et al., 2019] Cheng, S., Wang, L., Xiang, T., and Zhang, P. (2019). Tree tensor networks for generative modeling. *Physical Review B*, 99(15):155131.

[Chi-Chung et al., 1997] Chi-Chung, L., Sadayappan, P., and Wenger, R. (1997). On optimizing a class of multi-dimensional loops with reduction for parallel execution. *Parallel Processing Letters*, 7(02):157–168.

[Coyle et al., 2020] Coyle, B., Mills, D., Danos, V., and Kashefi, E. (2020). The Born supremacy: quantum advantage and training of an Ising Born machine. *npj Quantum Information*, 6(1):60.

[Du et al., 2020] Du, Y., Hsieh, M.-H., Liu, T., and Tao, D. (2020). Expressive power of parametrized quantum circuits. *Physical Review Research*, 2(3):033125.

[Evenbly and Vidal, 2009] Evenbly, G. and Vidal, G. (2009). Algorithms for entanglement renormalization. *Physical Review B*, 79(14):144108.

[Farhi et al., 2014] Farhi, E., Goldstone, J., and Gutmann, S. (2014). A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*.

[Friedrich and Maziero, 2022a] Friedrich, L. and Maziero, J. (2022a). Avoiding barren plateaus with classical deep neural networks. *Physical Review A*, 106(4):042433.

[Friedrich and Maziero, 2022b] Friedrich, L. and Maziero, J. (2022b). Avoiding barren plateaus with classical deep neural networks. *Physical Review A*, 106(4):042433.

[Gallego and Orus, 2022] Gallego, A. J. and Orus, R. (2022). Language design as information renormalization. *SN Computer Science*, 3(2):140.

[Grant et al., 2019a] Grant, E., Wossnig, L., Ostaszewski, M., and Benedetti, M. (2019a). An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214.

[Grant et al., 2019b] Grant, E., Wossnig, L., Ostaszewski, M., and Benedetti, M. (2019b). An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214.

[Grimsley et al., 2019] Grimsley, H. R., Economou, S. E., Barnes, E., and Mayhall, N. J. (2019). An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature communications*, 10(1):3007.

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

[Gunst et al., 2019] Gunst, K., Verstraete, F., and Van Neck, D. (2019). Three-legged tree tensor networks with SU(2) and molecular point group symmetry. *Journal of Chemical Theory and Computation*, 15(5):2996–3007.

[Han et al., 2018] Han, Z.-Y., Wang, J., Fan, H., Wang, L., and Zhang, P. (2018). Unsupervised generative modeling using matrix product states. *Physical Review X*, 8(3):031012.

[Holmes et al., 2021] Holmes, Z., Arrasmith, A., Yan, B., Coles, P. J., Albrecht, A., and Sornborger, A. T. (2021). Barren plateaus preclude learning scramblers. *Physical Review Letters*, 126(19):190501.

[Holmes et al., 2022] Holmes, Z., Sharma, K., Cerezo, M., and Coles, P. J. (2022). Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3(1):010313.

[Inomata et al., 2025] Inomata, K., Sugawara, S., Okubo, T., and Todo, S. (2025). In preparation.

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

[Kawamura et al., 2017] Kawamura, M., Yoshimi, K., Misawa, T., Yamaji, Y., Todo, S., and Kawashima, N. (2017). Quantum lattice model solver hϕ. *Computer Physics Communications*, 217:180–192.

[Levine et al., 2019] Levine, Y., Sharir, O., Cohen, N., and Shashua, A. (2019). Quantum entanglement in deep learning architectures. *Physical Review Letters*, 122(6):065301.

[Lin et al., 2017] Lin, Y.-P., Kao, Y.-J., Chen, P., and Lin, Y.-C. (2017). Griffiths singularities in the random quantum Ising antiferromagnet: A tree tensor network renormalization group study. *Physical Review B*, 96(6):064427.

[Liu et al., 2019a] Liu, D., Ran, S.-J., Wittek, P., Peng, C., García, R. B., Su, G., and Lewenstein, M. (2019a). Machine learning by unitary tensor network of hierarchical tree structure. *New Journal of Physics*, 21(7):073059.

[Liu and Wang, 2018] Liu, J.-G. and Wang, L. (2018). Differentiable learning of quantum circuit born machines. *Physical Review A*, 98(6):062324.

[Liu et al., 2019b] Liu, J.-G., Zhang, Y.-H., Wan, Y., and Wang, L. (2019b). Variational quantum eigensolver with fewer qubits. *Physical Review Research*, 1(2):023025.

[Liu et al., 2021] Liu, Y., Liu, X., Li, F., Fu, H., Yang, Y., Song, J., Zhao, P., Wang, Z., Peng, D., Chen, H., et al. (2021). Closing the "quantum supremacy" gap: achieving real-time simulation of a random quantum circuit using a new sunway supercomputer. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12.

[Malz et al., 2024] Malz, D., Styliaris, G., Wei, Z.-Y., and Cirac, J. I. (2024). Preparation of matrix product states with log-depth quantum circuits. *Physical Review Letters*, 132(4):040404.

[Markov and Shi, 2008] Markov, I. L. and Shi, Y. (2008). Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981.

[McClean et al., 2018] McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., and Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812.

[Mitarai et al., 2018] Mitarai, K., Negoro, M., Kitagawa, M., and Fujii, K. (2018). Quantum circuit learning. *Physical Review A*, 98(3):032309.

[Murg et al., 2010] Murg, V., Verstraete, F., Legeza, Ö., and Noack, R. M. (2010). Simulating strongly correlated quantum systems with tree tensor networks. *Physical Review B*, 82(20):205105.

[Nagaj et al., 2008] Nagaj, D., Farhi, E., Goldstone, J., Shor, P., and Sylvester, I. (2008). Quantum transverse-field Ising model on an infinite tree from matrix product states. *Physical Review B*, 77(21):214431.

[Ortiz Marrero et al., 2021] Ortiz Marrero, C., Kieferová, M., and Wiebe, N. (2021). Entanglement-induced barren plateaus. *PRX Quantum*, 2(4):040316.

[Orús, 2014] Orús, R. (2014). A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of physics*, 349:117–158.

[Peruzzo et al., 2014] Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., Aspuru-Guzik, A., and O'brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213.

[Preskill, 2018] Preskill, J. (2018). Quantum computing in the nisq era and beyond. *Quantum*, 2:79.

[Ran, 2020] Ran, S.-J. (2020). Encoding of matrix product states into quantum circuits of one-and two-qubit gates. *Physical Review A*, 101(3):032310.

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

[Romero et al., 2017] Romero, J., Olson, J. P., and Aspuru-Guzik, A. (2017). Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2(4):045001.

[Rudolph et al., 2023a] Rudolph, M. S., Chen, J., Miller, J., Acharya, A., and Perdomo-Ortiz, A. (2023a). Decomposition of matrix product states into shallow quantum circuits. *Quantum Science and Technology*, 9(1):015012.

[Rudolph et al., 2023b] Rudolph, M. S., Miller, J., Motlagh, D., Chen, J., Acharya, A., and Perdomo-Ortiz, A. (2023b). Synergistic pretraining of parametrized quantum circuits via tensor networks. *Nature Communications*, 14(1):8367.

[Schuld and Killoran, 2019] Schuld, M. and Killoran, N. (2019). Quantum machine learning in feature Hilbert spaces. *Physical Review Letters*, 122(4):040504.

[Sharma et al., 2022] Sharma, K., Cerezo, M., Cincio, L., and Coles, P. J. (2022). Trainability of dissipative perceptron-based quantum neural networks. *Physical Review Letters*, 128(18):180505.

[Shi et al., 2006] Shi, Y.-Y., Duan, L.-M., and Vidal, G. (2006). Classical simulation of quantum many-body systems with a tree tensor network. *Physical Review A*, 74(2):022320.

[Shirakawa et al., 2024] Shirakawa, T., Ueda, H., and Yunoki, S. (2024). Automatic quantum circuit encoding of a given arbitrary quantum state. *Physical Review Research*, 6(4):043008.

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

[Silvi et al., 2010] Silvi, P., Giovannetti, V., Montangero, S., Rizzi, M., Cirac, J. I., and Fazio, R. (2010). Homogeneous binary trees as ground states of quantum critical Hamiltonians. *Physical Review A*, 81(6):062335.

[Skolik et al., 2021] Skolik, A., McClean, J. R., Mohseni, M., Van Der Smagt, P., and Leib, M. (2021). Layerwise learning for quantum neural networks. *Quantum Machine Intelligence*, 3:1–11.

[Stoudenmire and Schwab, 2016] Stoudenmire, E. and Schwab, D. J. (2016). Supervised learning with tensor networks. *Advances in neural information processing systems*, 29.

[Tagliacozzo et al., 2009] Tagliacozzo, L., Evenbly, G., and Vidal, G. (2009). Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law. *Physical Review B*, 80(23):235127.

[Takeshita et al., 2020] Takeshita, T., Rubin, N. C., Jiang, Z., Lee, E., Babbush, R., and McClean, J. R. (2020). Increasing the representation accuracy of quantum simulations of chemistry without extra quantum resources. *Physical Review X*, 10(1):011004.

[Verdon et al., 2019] Verdon, G., Broughton, M., McClean, J. R., Sung, K. J., Babbush, R., Jiang, Z., Neven, H., and Mohseni, M. (2019). Learning to learn with quantum neural networks via classical neural networks. *arXiv preprint arXiv:1907.05415*.

[Vidal, 2003] Vidal, G. (2003). Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters*, 91(14):147902.

[Volkoff and Coles, 2021] Volkoff, T. and Coles, P. J. (2021). Large gradients via correlation in random parameterized quantum circuits. *Quantum Science and Technology*, 6(2):025008.

[Wang et al., 2021] Wang, S., Fontana, E., Cerezo, M., Sharma, K., Sone, A., Cincio, L., and Coles, P. J. (2021). Noise-induced barren plateaus in variational quantum algorithms. *Nature communications*, 12(1):6961.

[Wang et al., 2018] Wang, Z., Hadfield, S., Jiang, Z., and Rieffel, E. G. (2018). Quantum approximate optimization algorithm for maxcut: A fermionic view. *Physical Review A*, 97(2):022304.

[White, 1992] White, S. R. (1992). Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863.

[Wiersema et al., 2020] Wiersema, R., Zhou, C., de Sereville, Y., Carrasquilla, J. F., Kim, Y. B., and Yuen, H. (2020). Exploring entanglement and optimization within the Hamiltonian variational ansatz. *PRX quantum*, 1(2):020319.

[Xiang, 2023] Xiang, T. (2023). *Density Matrix and Tensor Network Renormalization*. Cambridge University Press.

[Yuan et al., 2021] Yuan, X., Sun, J., Liu, J., Zhao, Q., and Zhou, Y. (2021). Quantum simulation with hybrid tensor networks. *Physical Review Letters*, 127(4):040501.

[Zhou et al., 2020] Zhou, L., Wang, S.-T., Choi, S., Pichler, H., and Lukin, M. D. (2020). Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067.