SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY
INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics

An Optimal Symbolic Construction of Matrix Product Operators and Tree Tensor Network Operators

Hazar Çakır

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY
INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics

An Optimal Symbolic Construction of Matrix Product Operators and Tree Tensor Network Operators

Eine Optimale Symbolische Konstruktion von Matrix-Produkt-Operatoren und Baum-Tensor-Netzwerk-Operatoren

Author:	Hazar Çakır
Supervisor:	Prof. Dr. Christian B. Mendl
Advisor:	M.Sc. Richard Milbradt
Submission Date:	08.01.2025

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 08.01.2025 Hazar Çakır

Acknowledgments

I would like to express my deepest gratitude to my professor, Prof. Christian Mendl, for his invaluable guidance, insightful feedback, and unwavering support throughout the duration of this thesis. His expertise and encouragement have been instrumental in shaping the direction and quality of my work. I am equally grateful to my advisor, Richard Milbradt, also for his mentorship and constructive advice.

I am profoundly thankful to the German Academic Exchange Service, and the Turkish Education Foundation (DAAD - TEV) for their financial support, which allowed me to pursue my dreams, i.e., my master's degree in Germany. Their support allowed me to undertake this research and complete my studies at the Technical University of Munich like many others.

I am deeply grateful to my previous teachers, both at the Technical University of Munich (TUM) and Boğaziçi University. The knowledge, inspiration, and mentorship I received during my academic journey have been invaluable.

Additionally, I would like to acknowledge OpenAI's ChatGPT, which played a supportive role in the writing process. It helped enhance the clarity and coherence of this thesis. Importantly, all AI-generated content was carefully reviewed and tailored to meet the specific needs and standards of the research.

Lastly, I would like to extend my heartfelt thanks to my friends, who provided both academic and emotional support during this process and of course, I am deeply thankful to my family for their continuous love, encouragement, and support. Their belief in me has been a constant source of motivation, and their understanding and sacrifices made this achievement possible.

To all who have contributed to this endeavor in any way, I offer my sincere thanks.

Abstract

The construction of Matrix Product Operators (MPOs) and Tree Tensor Network Operators (TTNOs) plays a pivotal role in the simulation and analysis of quantum systems, particularly in capturing their complex interactions and structures. This thesis introduces an innovative framework that combines bipartite graph theory with a symbolic Gaussian elimination preprocessing step to address the challenges in the optimal construction of these operators. While previous methods have focused on linear MPO structures, this work extends the methodology to TTNOs, offering a novel approach to hierarchical operator representations.

The research begins by critically analyzing an established algorithm for MPO construction based on bipartite graphs. Through a detailed evaluation, the algorithm's strengths and limitations were identified, revealing its inability to handle certain edge cases. To overcome these shortcomings, a new algorithm was developed, incorporating symbolic Gaussian elimination to achieve greater flexibility and accuracy in bond dimension optimization. This enhanced approach preserves the symbolic nature of the computation, ensuring both numerical stability and general applicability.

Building on this foundation, the algorithm was adapted for TTNOs, which represent a more general and hierarchical class of tensor network operators. This transition introduced additional complexities. A systematic implementation strategy was devised to address these challenges, emphasizing modularity, efficiency, and reproducibility. Good programming practices were followed throughout the development process to ensure the resulting framework could be easily understood and extended.

The outcomes of this research demonstrate significant improvements in the construction of both MPOs and TTNOs, achieving reduced bond dimensions and better alignment with the underlying system symmetries. The findings represent a meaningful contribution to the field, providing a robust algorithmic solution for tensor network operator construction. This thesis details the theoretical background, algorithmic innovations, and practical implementations, offering a comprehensive approach to improving quantum simulation methodologies.

Contents

1. Introduction

The study of quantum computing bridges principles from quantum physics, computer science, and mathematics, offering profound insights into the behavior of quantum systems [1, 2]. This interdisciplinary field plays a pivotal role in advancing technologies such as quantum computing, cryptography [3], and the simulation of quantum phenomena [4–6]. Among its many aspects, this thesis focuses on quantum simulation, particularly the simulation of quantum systems on classical computers.

Simulating quantum systems on classical computers presents an inherent challenge: the representation of quantum systems. Due to the fundamentally different nature of quantum data compared to classical approaches, innovative methods have been developed over time. A cornerstone of these efforts is the concept of Matrix Product Operators (MPOs), which provide an efficient framework for representing quantum states and operations [7]. MPOs have been extensively studied for their utility in one-dimensional quantum systems, leading to the development and optimization of numerous algorithms [8–11]. However, the initial configuration and construction of MPOs have received comparatively less attention. But it is known that the efficiency of algorithms like Density Matrix Renormalization Group (DMRG) and Time-Evolving Block Decimation (TEBD) is significantly influenced by the initial construction of MPOs [12] [13]. This thesis focuses on addressing this gap by exploring methods for the optimal construction of such systems, which forms the core focus of this work.

Another key structure explored in this thesis is Tree Tensor Network Operators (TTNOs), an alternative representation of quantum systems that employs a hierarchical tree structure[14]. Unlike MPOs, which impose a linear representation regardless of the system's inherent structure, TTNOs offer a framework better suited for capturing long-range interactions. The linear nature of MPOs can introduce limitations when modeling such interactions, as no direct relationships are preserved within their structure. TTNOs address this by maintaining a tree-like relationship scheme, which is theoretically more accurate for certain scenarios.

This thesis focuses on the application of bipartite graph theory for the construction of MPOs and TTNOs[15]. Bipartite graph theory provides an efficient framework for

identifying interactions within quantum systems—such as virtual bonds—which holds great potential for developing algorithms that optimize tensor network operator construction. Initially, the study involved a comprehensive analysis and implementation of an existing bipartite graph algorithm for constructing MPOs to evaluate its effectiveness. During this process, limitations in the existing approach were identified, motivating the development of a new algorithm aimed at addressing these shortcomings.

The proposed novel approach incorporates symbolic Gaussian elimination as a preprocessing step for the bipartite graph algorithm. This enhancement preserves the symbolic nature of the original solution while introducing improvements to tackle previously observed limitations. The specifics of this algorithm are discussed in detail in Chapter 4

Subsequently, the algorithm was adapted for TTNOs, which represent a more generalized case compared to MPOs by incorporating tree-like structures rather than linear relationships. This adaptation presented significant challenges, particularly in transitioning from a linear framework to a hierarchical tree-based structure. While the core principles of bond optimization remained consistent, considerable effort was required at the implementation level to accommodate tree structures. Throughout this process, good programming practices were followed to ensure the resulting algorithm is both efficient and maintainable.

In summary, this thesis explores an efficient symbolic construction of MPOs and TTNOs through innovative applications of bipartite graph theory and symbolic Gaussian elimination. By addressing key limitations in existing methodologies and extending these approaches to tree tensor networks, the work provides a robust foundation for future advancements in quantum simulation techniques. This introduction provides an overview of the project's background, motivation, challenges, and objectives. It sets the stage for the detailed account that follows, covering the literature review, theory, methodology, and experiments, building up in a discussion of the project's contributions to the literature.

2. Related Work

The optimization and construction of Matrix Product Operators (MPOs) and Tree Tensor Network Operators (TTNOs) have been studied broadly in the literature. This section reviews key contributions to the field, highlighting their methodologies, strengths, and limitations, and situating this research within the wider context.

2.1. A General Automatic Method for Optimal Construction of Matrix Product Operators Using Bipartite Graph Theory

A significant starting point for this thesis was the detailed analysis and implementation of the methodologies proposed by Ren et al. (2020) [15]. Their work introduced an algorithm for the automatic construction of optimal MPOs leveraging bipartite graph theory. Their method utilizes the minimum vertex cover of a bipartite graph to achieve this construction. Their approach successfully eliminates numerical errors and produces globally optimal MPOs for nearly all kinds of Hamiltonians expressed in a sum-of-products form by employing symbolic computation. The algorithm demonstrates its effectiveness in reducing MPO bond dimensions and has shown success across various models, including the spin-boson and Holstein models and Ab initio electronic Hamiltonian.

Despite these advancements, our analysis uncovered cases where the algorithm fails to achieve optimal performance, challenging the claim of its universal applicability. Additionally, the algorithm lacks a generalization beyond linear MPO structures, underscoring the need for further exploration into hierarchical representations, such as Tree Tensor Network Operators (TTNOs).

2.2. Matrix Product Operators, Matrix Product States, and ab initio Density Matrix Renormalization Group algorithms

Chan et al. (2016) [16] provide a comprehensive analysis of the ab initio Density Matrix Renormalization Group (DMRG) algorithm, making the connections between

the traditional renormalization group framework and the modern matrix product state (MPS) and matrix product operator (MPO) formalism clear. They detail the translation between these two paradigms, demonstrating how the same algorithm can be expressed differently depending on the chosen language. The authors also introduce improvements to the ab initio DMRG sweep algorithm, such as Hamiltonian compression and a sum-over-operators representation, which enhance computational efficiency and parallelism.

Early in this thesis, the experimental focus was refined to study the ab initio electronic Hamiltonian system. A foundational understanding of fermionic algebra was critical to this endeavor, a challenge addressed through the foundational insights provided by Chan et al. (2016) [16]. Their work not only deepened my understanding of electronic systems but also illuminated their relationship with tensor networks, serving as a cornerstone for the applied research conducted in this thesis.

2.3. State Diagrams for Tree Tensor Network Operators

Milbradt et al. (2024) [17] introduced the concept of state diagrams as a framework for determining Tree Tensor Network Operators (TTNOs). Their work established a connection between tree topologies and operator representations, providing a novel blueprint for representing the hierarchical structure of TTNOs. When this thesis expanded its focus to include the construction of TTNOs, Milbradt et al. (2024) served as a pivotal reference point.

Although their methodology deviates significantly from bipartite graph theory and employs a suboptimal approach for forming TTNOs, the paper was instrumental in offering an initial conceptual framework. By leveraging the state diagram representation, this research embarked on developing a new algorithmic approach grounded in bipartite graph theory. This innovative method utilized the state diagram structure as a foundation while addressing its limitations to create an optimized construction process for TTNOs.

2.4. Optimal Tree Tensor Network Operators for Tensor Network Simulations: Applications to Open Quantum Systems

Li et al. (2024) [18] presented an algorithm for the automatic construction of optimal and exact Tree Tensor Network Operators (TTNOs) for quantum operators expressed in sum-of-product form. Building upon the work of Ren et al. (2020) [15], the authors—hailing from the same research group—extended the bipartite graph approach to the more generalized TTNO framework. Their algorithm was applied to simulate open quantum systems, addressing scenarios such as spin relaxation dynamics in the spin-boson model and charge transport in molecular junctions.

This paper, published during the final stages of the development of this thesis, operates in a closely related domain by extending the bipartite graph theory to hierarchical TTNO structures. However, despite this extension, the paper does not address the inherent limitations of the original approach, rendering the new method equally problematic in cases where the MPO approach previously fell short. This underscores the ongoing need for improvements in the generality and robustness of these algorithms.

2.5. The Density-Matrix Renormalization Group in the Age of Matrix Product States

Schollwöck (2011) [19] presented a comprehensive review of the Density Matrix Renormalization Group (DMRG) and its reformulation using Matrix Product States (MPS). The paper traces the historical evolution of DMRG, illustrating its transition from a numerical renormalization group method to a fully developed tensor network approach. Schollwöck highlights the extension of DMRG to tackle finite-temperature and time-dependent problems, as well as its integration with Matrix Product Operator (MPO) techniques for representing quantum operators. This review serves as a cornerstone in tensor network research, providing insights into the computational efficiency and versatility of these methods, which have significantly influenced subsequent advancements in the field.

In the context of this thesis, Schollwöck's review provided a crucial foundation for understanding the mathematical and computational frameworks underpinning MPS and MPOs. By delving into the integration of MPO techniques within DMRG, the paper informed our exploration of the initial configuration of MPOs and the role of

bond dimension optimization in algorithmic efficiency. The insights gained from this foundational work were instrumental in guiding our approach to constructing optimal MPOs and extending these methodologies to Tree Tensor Network Operators (TTNOs).

2.6. Generic Construction of Efficient Matrix Product Operators

Hubig et al. (2017) [9] introduced a generic scheme for constructing efficient Matrix Product Operators (MPOs). Their approach focuses on reducing bond dimensions through compression techniques, thereby enhancing computational performance while maintaining the accuracy of quantum simulations. By systematically compressing identical or redundant terms in the MPO representation, the authors demonstrated significant improvements in both the efficiency and scalability of tensor network-based algorithms.

In this thesis, the methods outlined by Hubig et al. were pivotal in shaping our understanding of MPO compression and its impact on bond dimension optimization. The paper's emphasis on eliminating redundancies resonated with our symbolic Gaussian elimination preprocessing step, offering a complementary perspective to the bipartite graph algorithm. By drawing on the principles established by Hubig et al., we were able to refine our approach, ensuring that our constructed MPOs maintained both efficiency and flexibility across various quantum systems.

3. Theory

3.1. Hamiltonian Representation with Operator Strings

In quantum mechanics, the Hamiltonian of a system is the key operator that defines its energy and governs its dynamics. Representing Hamiltonians efficiently and flexibly is crucial for analyzing and simulating quantum systems. In our framework, the Hamiltonian is expressed as a sum of terms, each consisting of a product of local operators and an associated coefficient, known as the **operator string representation**. This is mathematically described as:

$$H = \sum_i \gamma_i \hat{O}_1^{(i)} \otimes \hat{O}_2^{(i)} \otimes \ldots \otimes \hat{O}_N^{(i)}, \tag{3.1}$$

where:

- γ_i: Real or complex coefficients associated with each term, representing interaction strengths or coupling constants.
- $\hat{O}_j^{(i)}$: Local operators acting on site j for the i^{th} term. These operators can be Pauli matrices, annihilation/creation operators, or other quantum operators.
- $\otimes$: The tensor product, combining operators acting on different sites.

The operator string representation provides several advantages when dealing with complex quantum systems:

1. **Modularity**: Each term in the Hamiltonian is represented as a modular unit consisting of a coefficient and a sequence of local operators. This modularity allows for a clear and systematic way to construct and manipulate the Hamiltonian, making introducing or modifying new interactions easier.
2. **Flexibility**: This representation can accommodate a wide range of Hamiltonians, including those with nearest-neighbor interactions, long-range interactions, and even more complex multi-body interactions. The use of local operators $\hat{O}_j^{(i)}$ in each string allows for a concise encoding of the system's dynamics.

3. **Scalability**: As the system size increases, the operator string representation scales well by simply adding more terms or extending the length of the operator strings. This scalability is crucial for studying large quantum systems or systems with many degrees of freedom.

4. **Compatibility with Tensor Networks**: The operator string representation is particularly well-suited for tensor network methods such as Matrix Product Operators (MPOs) and Tree Tensor Network Operators (TTNOs). Each term in the Hamiltonian can be directly mapped onto a tensor network structure, where the local operators correspond to tensors and the coefficients γ_i are included as weights.

5. **Simplicity in Implementation**: The operator string representation offers a straightforward way to encode complex Hamiltonians. Its structured format makes it easier to translate into computational algorithms, facilitating numerical simulations and symbolic manipulations.

3.1.1. Power of the Representation

The power of the operator string representation lies in its ability to represent a wide variety of Hamiltonians compactly and efficiently. For example, consider the following typical forms:

- **Heisenberg Model**: The Hamiltonian can be written as a sum of operator strings involving spin operators $\hat{S}^x, \hat{S}^y, \hat{S}^z$ for different sites.

$$H = \sum_{\langle i,j \rangle} \gamma_{ij} \left(\hat{S}_i^x \otimes \hat{S}_j^x + \hat{S}_i^y \otimes \hat{S}_j^y + \hat{S}_i^z \otimes \hat{S}_j^z \right), \tag{3.2}$$

  where γ_{ij} are coefficients representing the coupling strength between spins on sites i and j. The Heisenberg model captures phenomena like ferromagnetism and antiferromagnetism depending on the sign of γ_{ij}. This model is particularly useful in studying quantum phase transitions and spin entanglement.[20]

- **Bosonic Hamiltonians**: In systems involving bosonic operators $\hat{a}, \hat{a}^\dagger$, the operator string representation can be used to encode creation and annihilation terms.

$$H = \sum_{i,j} \gamma_{ij} \hat{a}_i^\dagger \otimes \hat{a}_j + \sum_i \gamma_{ii} \hat{a}_i^\dagger \hat{a}_i, \tag{3.3}$$

  where γ_{ij} are the coefficients governing the hopping and on-site interactions. This Hamiltonian is often used to study systems like superfluidity, Bose-Einstein

condensation, and lattice models in cold atomic gases. The first term represents particle hopping between different sites, while the second term accounts for interactions within each site.

- **Multi-Body Interactions**: The operator string framework can also handle complex multi-body terms, such as those found in certain lattice gauge theories or models of superconductivity.

$$H = \sum_{i,j,k} \gamma_{ijk} \hat{O}_i \otimes \hat{O}_j \otimes \hat{O}_k, \tag{3.4}$$

where γ_{ijk} are the coefficients for the three-body interaction terms. Multi-body interactions are crucial for describing phenomena such as topological phases of matter, where interactions among multiple particles lead to emergent properties that cannot be understood through pairwise interactions alone. These models are computationally demanding but provide deep insights into strongly correlated systems.

3.1.2. Coefficients and Their Role

The coefficients γ_i are fundamental components in the Hamiltonian, as they define the strength and nature of the interactions within the quantum system. These coefficients serve as numerical weights for the terms in the operator string representation, directly influencing the physical properties and behavior of the system. In physical models, the coefficients γ_i often correspond to key parameters that characterize the system's dynamics, including:

Coupling Constants: These coefficients quantify the interaction strength between different components of the system, such as spins in a spin lattice or particles in a bosonic or fermionic system. For example, in the Heisenberg model, γ_{ij} represents the exchange interaction between spins on sites i and j.

External Field Strengths: In systems subjected to external fields, coefficients like γ_i determine the influence of these fields on the local operators. For instance, in a magnetic system, γ_i might represent the strength of a magnetic field acting on a particular spin.

Decay Factors: In models with long-range interactions, coefficients γ_{ij} can encode the distance-dependent decay of interaction strengths, such as $\gamma_{ij} = J/\|i-j\|^{\alpha}$, where $\|i-j\|$ is the distance between sites i and j and α is the decay exponent.

In summary, the coefficients γ_i not only define the strength of interactions in the Hamiltonian but also play a crucial role in shaping the overall behavior of the system, its energy landscape, and its dynamical evolution. Understanding and appropriately selecting γ_i is essential for accurately modeling and analyzing quantum systems.

3.2. Tensor Networks

Tensor networks (TNs) are a mathematical framework for efficiently representing and manipulating high-dimensional data, particularly in the context of quantum many-body systems. They provide a structured way to decompose complex tensors into a network of simpler, lower-dimensional tensors connected by shared indices. This decomposition is crucial for overcoming the exponential growth of computational resources required to represent large quantum systems. [21]

The concept of tensor networks originated from efforts to address the computational challenges of quantum mechanics. Early developments can be traced back to the work on *Matrix Product States (MPS)* in the context of the Density Matrix Renormalization Group (DMRG) method, introduced by White in the early 1990s [22][23]. DMRG became a groundbreaking algorithm for studying one-dimensional quantum systems, particularly in condensed matter physics.

Building on MPS, higher-dimensional extensions such as *Projected Entangled Pair States (PEPS)* and *Tree Tensor Networks (TTNs)* were developed to handle systems with more complex entanglement structures. The introduction of *Tensor Network States (TNS)* provided a unifying framework for these representations, allowing for systematic exploration of quantum systems with varied topologies and dimensions.

Recent advances have further extended tensor networks to applications beyond quantum mechanics, including machine learning, statistical physics, and quantum chemistry.

3.2.1. The Need for Tensor Networks

The primary motivation for tensor networks arises from the exponential complexity of representing quantum states and operators. In a system of N qubits, the state vector requires 2^N complex coefficients, making exact representations infeasible for large N. Tensor networks address this challenge by exploiting the fact that many physically relevant quantum states exhibit limited entanglement, which can be efficiently captured using low-rank tensor decompositions. [24]

Tensor networks offer several key advantages that make them indispensable tools in quantum many-body physics and beyond. One of their primary benefits is their **efficient representation** of high-dimensional data. By leveraging the entanglement structure of quantum systems, tensor networks significantly reduce both storage and

computational demands, enabling the study of systems that would otherwise be infeasible to model. Additionally, tensor networks are highly **scalable**, providing a robust framework for simulating large systems. This scalability is particularly valuable for addressing problems that are computationally intractable using conventional methods. Finally, tensor networks exhibit remarkable **flexibility**, as they can be adapted to a wide variety of system geometries, dimensions, and interaction types. This versatility allows researchers to study an extensive range of quantum systems, from simple linear chains to complex multi-dimensional lattices. [25]

Tensor networks have revolutionized the study of quantum many-body systems by providing a flexible and scalable representation framework. They balance accuracy and computational efficiency, enabling the exploration of large and complex systems that were previously beyond reach. The foundational concepts and algorithms of tensor networks set the stage for advanced methods like Tree Tensor Networks (TTNs) and their operator-based extensions, which will be discussed in the subsequent sections.

3.2.2. Historical Development of Tensor Networks

The development of tensor networks is deeply rooted in the need to efficiently study quantum many-body systems, where the complexity of the Hilbert space grows exponentially with system size. The history of tensor networks can be traced through several key milestones:

Density Matrix Renormalization Group (DMRG) and Matrix Product States (MPS)

The foundation of tensor networks was laid in the early 1990s with the introduction of the **Density Matrix Renormalization Group (DMRG)** by Steven R. White [22][23]. DMRG was developed as a variational method for studying the ground-state properties of one-dimensional quantum systems, such as spin chains. It achieved unprecedented accuracy for systems like the spin-1 Heisenberg chain and the Hubbard model, outperforming traditional renormalization group techniques.

Around the same time, it was recognized that DMRG could be reformulated in terms of a specific tensor network structure known as **Matrix Product States (MPS)**. MPS provide an efficient representation for quantum states with low entanglement, typical of gapped one-dimensional systems. An MPS expresses a quantum state as:

$$|\psi\rangle = \sum_{i_1, i_2, \ldots, i_N} A^{[1]}_{i_1} A^{[2]}_{i_2} \cdots A^{[N]}_{i_N} |i_1 i_2 \ldots i_N\rangle,$$

where $A_{i_n}^{[n]}$ are matrices representing tensors at each site, and $|i_n\rangle$ are basis states of the local Hilbert space.

Projected Entangled Pair States (PEPS)

In the mid-2000s, **Projected Entangled Pair States (PEPS)** were introduced as a generalization of MPS to higher dimensions[26] [27]. PEPS extend the tensor network representation to two-dimensional and three-dimensional systems, where tensors are connected in a lattice structure. While PEPS can encode complex correlations in higher-dimensional systems, their computational cost is significantly higher than that of MPS. [28]

PEPS became a crucial tool for studying two-dimensional quantum systems, such as the toric code and topological phases of matter. However, the increased complexity posed challenges for efficient optimization and contraction algorithms.

Tree Tensor Networks (TTNs)

Tree Tensor Networks (TTNs) emerged as a hierarchical alternative to MPS and PEPS. TTNs use a tree-like structure to represent quantum states, where tensors are organized hierarchically, with leaves corresponding to physical indices and internal nodes encoding correlations [29]. TTNs are particularly effective for systems with localized interactions and low entanglement, providing an efficient approximation of ground states and low-energy excitations.

Multiscale Entanglement Renormalization Ansatz (MERA)

In 2007, **MERA (Multiscale Entanglement Renormalization Ansatz)** was introduced by Guifre Vidal [30]. MERA incorporates a hierarchical structure similar to TTNs but includes disentangling layers to efficiently represent quantum states with critical or scale-invariant properties. This made MERA particularly powerful for studying conformal field theories and quantum systems at criticality.

The key feature of MERA is its ability to represent states with logarithmic scaling of entanglement entropy, which is characteristic of gapless systems. The disentangling layers reduce correlations at each scale, allowing for efficient representation and manipulation of the quantum state.[31]

Tensor Renormalization Group (TRG)

Tensor Renormalization Group (TRG), introduced in 2007 by Levin and Nave [32], provided a coarse-graining technique for studying partition functions in statistical mechanics. TRG uses tensor networks to represent the partition function and iteratively reduces the network by contracting tensors, preserving essential information about the system.

TRG inspired subsequent advancements in renormalization techniques, including **Higher-Order Tensor Renormalization Group (HOTRG)**[33] and **Tensor Network Renormalization (TNR)**[34], which improved accuracy and efficiency for systems in both quantum and classical regimes.

Insights from Schollwöck's 2011 Paper

In 2011, Ulrich Schollwöck published a comprehensive review article titled *The Density Matrix Renormalization Group in the Age of Matrix Product States* [19], which firmly established the connection between DMRG and MPS. This paper highlighted the underlying principles of MPS as a variational ansatz and its extensions to time evolution, finite-temperature states, and open quantum systems. Schollwöck's work was instrumental in transitioning DMRG from being viewed as a standalone algorithm to being understood as part of the broader tensor network framework.

Key contributions of this paper include:

- A formalization of DMRG as an optimization over MPS, providing a unified language for the method.
- Extensions of MPS to finite-temperature states using purification techniques.
- Applications of DMRG and MPS to dynamical properties and time evolution through methods like Time-Dependent DMRG (tDMRG).

Schollwöck's review has become a cornerstone reference in the field, guiding researchers in understanding the interplay between DMRG, MPS, and tensor network techniques.

Applications Beyond Quantum Mechanics

In recent years, tensor networks have found applications beyond their traditional domain of quantum many-body physics. For example:

- **Quantum Chemistry:** Tensor networks, such as MPS, are used to approximate molecular wavefunctions and compute ground-state energies in quantum chemistry.
- **Machine Learning:** Tensor networks have been adapted for machine learning tasks, providing compact representations of neural networks and probabilistic models.
- **Statistical Mechanics:** Tensor networks are employed to study phase transitions and critical phenomena in classical systems.

3.3. Matrix Product States (MPS) and Matrix Product Operators (MPO)

Matrix Product States (MPS) and Matrix Product Operators (MPO) are foundational concepts in the study of tensor networks. They provide efficient representations of quantum states and operators, particularly in one-dimensional quantum systems. This section delves into the details of their structure, mathematical formulation, and significance.

3.3.1. Matrix Product States (MPS)

Matrix Product States are tensor network representations of quantum states that leverage the entanglement structure of one-dimensional systems. An MPS expresses a quantum state as a product of matrices, significantly reducing the computational resources required to store and manipulate the state.[10]

Consider a one-dimensional quantum system with N sites, each having a local Hilbert space of dimension d. A general quantum state $|\psi\rangle$ can be written in the form:

$$|\psi\rangle = \sum_{i_1,i_2,\ldots,i_N} c_{i_1 i_2 \ldots i_N} |i_1 i_2 \ldots i_N\rangle,$$

where $c_{i_1 i_2 \ldots i_N}$ are the coefficients, and $|i_1 i_2 \ldots i_N\rangle$ represents the basis states of the composite system. For a large system, the number of coefficients grows exponentially, d^N, making this representation infeasible for large N.

In the MPS representation, the coefficients $c_{i_1 i_2 \ldots i_N}$ are decomposed into a product of tensors:

$$c_{i_1 i_2 \ldots i_N} = \sum_{\alpha_1,\alpha_2,\ldots,\alpha_{N-1}} A^{[1]}_{i_1,\alpha_1} A^{[2]}_{\alpha_1,i_2,\alpha_2} \cdots A^{[N]}_{\alpha_{N-1},i_N}.$$

Here:

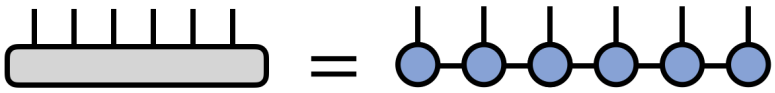

Figure 3.1.: A depiction of an MPS structure (adapted from tensornetwork.org[35]).

- $A^{[k]}$ is the local tensor at site k, where i_k is the physical index, and α_k and α_{k-1} are bond indices connecting neighboring tensors.
- The bond dimension χ of the indices α_k determines the amount of entanglement the MPS can capture. For low-entanglement states, χ can remain small even as N grows.

In compact form, the quantum state is expressed as:

$$|\psi\rangle = \sum_{i_1,i_2,\ldots,i_N} A^{[1]}_{i_1} A^{[2]}_{i_2} \cdots A^{[N]}_{i_N} |i_1 i_2 \ldots i_N\rangle.$$

Advantages of MPS

Matrix Product States (MPS) offer a highly **efficient representation** of quantum states with low entanglement. In particular, they excel at representing ground states of gapped one-dimensional systems, where the entanglement entropy scales logarithmically or remains constant. This compact representation significantly reduces the computational resources required to store and manipulate quantum states, making MPS a practical choice for simulating large quantum systems.

Another significant advantage of MPS is their **scalability**. The representation scales linearly with the system size for a fixed bond dimension χ, enabling efficient simulations of extensive systems without encountering exponential growth in computational demands. This property ensures that MPS remain computationally feasible even as the number of particles or degrees of freedom in the system increases.

MPS also serve as the foundation for many **powerful algorithms**, such as the Density Matrix Renormalization Group (DMRG) and Time-Evolving Block Decimation (TEBD). These algorithms leverage the MPS structure to efficiently compute ground states, simulate time evolution, and analyze other dynamical properties of quantum systems. The algorithmic utility of MPS has made them indispensable in the study of quantum many-body systems.

Operations on MPS

Matrix Product States also support a variety of efficient operations, making them practical tools for quantum simulations. The **norm** of an MPS can be computed by sequentially contracting tensors, allowing for a straightforward calculation. Similarly, **expectation values** of local observables can be obtained efficiently by exploiting the MPS structure, where the relevant contractions are performed locally. Furthermore, MPS are well-suited for **time evolution** simulations, as algorithms like TEBD leverage the MPS framework to propagate quantum states over time while maintaining computational efficiency. These capabilities underscore the versatility and utility of MPS in quantum many-body physics.

3.3.2. Matrix Product Operators (MPO)

Matrix Product Operators extend the MPS framework to represent operators instead of states. MPOs are crucial for efficiently encoding Hamiltonians, density matrices, and other operators in tensor network algorithms.[36]

Consider an operator $\hat{O}$ acting on a one-dimensional quantum system. In the full tensor representation, the operator is expressed as:

$$\hat{O} = \sum_{i_1,i_2,\ldots,i_N} \sum_{j_1,j_2,\ldots,j_N} O_{i_1 i_2 \ldots i_N, j_1 j_2 \ldots j_N} |i_1 i_2 \ldots i_N\rangle \langle j_1 j_2 \ldots j_N|.$$

Here, $O_{i_1 i_2 \ldots i_N, j_1 j_2 \ldots j_N}$ is a tensor containing the coefficients of the operator.

In the MPO representation, the coefficients are decomposed as:

$$O_{i_1 i_2 \ldots i_N, j_1 j_2 \ldots j_N} = \sum_{\alpha_1,\alpha_2,\ldots,\alpha_{N-1}} W^{[1]}_{i_1,j_1,\alpha_1} W^{[2]}_{\alpha_1,i_2,j_2,\alpha_2} \cdots W^{[N]}_{\alpha_{N-1},i_N,j_N}.$$

Here:

- $W^{[k]}$ is the local tensor at site k, with physical indices i_k and j_k, and bond indices α_k connecting neighboring tensors.
- The bond dimension χ controls the complexity of the MPO and determines the range of correlations it can encode.

Advantages of MPO

MPOs provide several critical advantages in the efficient representation and manipulation of quantum operators. One primary benefit is their **compact representation of operators**, which allows MPOs to efficiently encode operators with local or limited-range

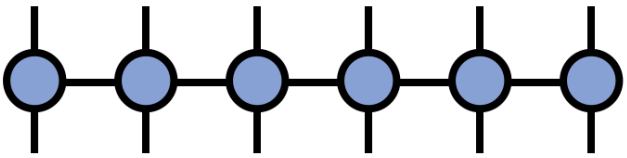

Figure 3.2.: A depiction of an MPO structure (adapted from tensornetwork.org[35]).

interactions, such as Hamiltonians for one-dimensional spin chains. This compactness significantly reduces the computational resources required for large-scale quantum simulations. Another advantage of MPOs is their **compatibility with MPS**, enabling seamless integration with Matrix Product States for tasks such as computing expectation values, simulating time evolution, and evaluating operator norms. Finally, MPOs exhibit excellent **scalability**, similar to MPS, as their representation scales efficiently with system size. This scalability makes them suitable for studying large systems with complex interactions.

Examples of MPOs

MPOs are versatile and can be used to represent a variety of operators in quantum systems. A common example is **local Hamiltonians**, such as those describing 1D spin chains with nearest-neighbor interactions. MPOs can also approximate **long-range interactions**, maintaining computational efficiency even for complex systems. Another straightforward example is the **identity operator** across all sites, which can be represented as an MPO with diagonal tensors $W^{[k]}$ at each site. These examples highlight the flexibility and utility of MPOs in quantum many-body simulations.

3.4. Tree Tensor Networks (TTNs) and Tree Tensor Network Operators (TTNOs)

Tree Tensor Networks (TTNs) and Tree Tensor Network Operators (TTNOs) are hierarchical tensor network structures designed to efficiently represent quantum states and operators, respectively. TTNs provide a compact representation of quantum states by leveraging their low entanglement, making them particularly effective for systems with localized interactions or ground states of gapped one-dimensional systems. Extending this concept, TTNOs encode quantum operators, such as Hamiltonians or density matrices, within a similar hierarchical framework. The tree-like structure of TTNs and TTNOs enables efficient encoding of correlations across multiple scales, making them

powerful tools for studying quantum many-body systems, quantum dynamics, and operator-based analyses. This section explores the theoretical foundations, mathematical formulations, and practical applications of TTNs and TTNOs, highlighting their advantages in computational efficiency and scalability.

3.4.1. Tree Tensor Networks (TTNs)

Tree Tensor Networks (TTNs) are a subclass of tensor networks distinguished by their hierarchical, tree-like structure. This structure makes TTNs particularly effective for representing and manipulating quantum states with low entanglement, such as ground states of gapped one-dimensional systems or systems with localized interactions. In TTNs, the nodes represent tensors, while the edges connecting these nodes correspond to shared indices, facilitating efficient encoding of correlations across the quantum system. [37]

The hierarchical nature of TTNs ensures a balanced distribution of information across the network. This structure is advantageous for approximating quantum states with reduced computational resources compared to more general tensor network structures, such as PEPS.

Mathematical Representation of TTNs

Mathematically, a TTN for a quantum state $|\psi\rangle$ can be expressed as:

$$|\psi\rangle = \sum_{i_1,i_2,\ldots,i_N} T^{(1)}_{i_1 i_2 \ldots i_k} T^{(2)}_{j_1 j_2 \ldots j_l} \ldots T^{(R)}_{r_1 r_2 \ldots r_m} |i_1, i_2, \ldots, i_N\rangle,$$

where:

- $T^{(1)}, T^{(2)}, \ldots, T^{(R)}$ are tensors representing the nodes of the network.
- $i_1, i_2, \ldots, i_N$ are the physical indices corresponding to the leaf nodes.
- $r_1, r_2, \ldots, r_m$ are bond indices connecting internal nodes.

Each tensor T_i in the TTN has a set of indices, typically including:

- **Physical Indices (p_i)**: Represent the local Hilbert space of a subsystem.
- **Bond Indices (b_{ij})**: Shared indices that connect tensors, encoding the correlations between subsystems.

The hierarchical structure allows TTNs to efficiently encode quantum correlations by contracting tensors along the edges of the tree, propagating information from the leaf nodes to the root.

Applications of TTNs

TTNs have several usage areas throughout the fields of quantum physics and computational science. One of their primary applications is in **ground-state approximation**, where TTNs are used to efficiently approximate the ground states of quantum systems, particularly in one-dimensional and quasi-one-dimensional geometries. The hierarchical structure of TTNs enables accurate and computationally efficient representations of these states.

In **quantum chemistry**, TTNs have proven valuable for modeling molecular systems. Their structure aligns well with the natural partitioning of electrons and nuclei, providing an efficient framework for representing electronic wavefunctions and other molecular properties.

TTNs also play a significant role in **quantum information theory**. By providing detailed insights into the entanglement structure of quantum systems, TTNs facilitate the study of entanglement entropy and related measures, which are critical for understanding quantum correlations and their implications.

Beyond quantum physics, TTNs have been adapted for **machine learning** tasks. Their hierarchical structure is particularly effective for encoding complex datasets, enabling efficient representation and analysis in machine learning applications. This cross-disciplinary utility highlights the broad impact of TTNs in both theoretical and applied settings.

Comparison to Other Tensor Networks

When compared to other tensor network representations, such as Matrix Product States (MPS) and Projected Entangled Pair States (PEPS), Tree Tensor Networks (TTNs) offer unique advantages. Unlike MPS, which are restricted to a linear topology, TTNs utilize a tree-like structure that enables more efficient encoding of correlations in non-linear systems. This flexibility allows TTNs to represent a wider range of quantum states effectively.

Furthermore, TTNs require fewer computational resources than PEPS, making them a more practical choice for systems with moderate dimensionality. While PEPS are suited for higher-dimensional systems, their computational cost can be prohibitive. TTNs strike a balance between efficiency and representational power, making them particularly advantageous for studying systems with hierarchical correlations or moderate

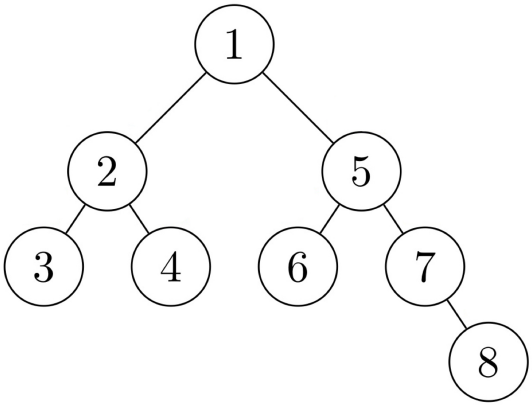

Figure 3.3.: Example of a Tree Tensor Network (TTN) for a 8-site quantum system. Leaf nodes correspond to physical indices, and internal nodes represent tensors connecting subsystems.

entanglement.

Graphical Representation of TTNs

Graphically, a TTN is depicted as a tree diagram, where:

- Nodes represent tensors.
- Edges represent shared indices, such as bond indices connecting two tensors.

An example of a TTN for a 8-site system is shown in Figure 3.3.

Mathematical Operations in TTNs

Tree Tensor Networks (TTNs) support several fundamental operations essential for their use in quantum many-body simulations. Similar to MPS and MPO, **tensor contraction** is a primary operation in TTNs, combining tensors by summing over shared indices to propagate information efficiently through the network. Another key operation is **optimization**, where tensor entries are iteratively adjusted to minimize the energy or match a target quantum state, ensuring the network effectively captures correlations. Additionally, **truncation** is performed to reduce bond dimensions, simplifying the TTN while retaining critical correlations. These operations collectively enable TTNs to provide efficient and scalable representatio

Summary

Tree Tensor Networks provide an efficient and flexible framework for representing quantum states with hierarchical correlations. Their structured design, combined with powerful mathematical operations, makes them an indispensable tool for studying quantum many-body systems, quantum chemistry, and related fields. In the following sections, we extend this framework to Tree Tensor Network Operators (TTNOs) for representing and manipulating quantum operators.

3.4.2. Tree Tensor Network Operators (TTNOs)

Tree Tensor Network Operators (TTNOs) are a natural extension of Tree Tensor Networks (TTNs), designed to represent and manipulate quantum operators rather than states. Like TTNs, TTNOs utilize a hierarchical tree structure, where each node corresponds to an operator tensor, and the edges represent shared indices. TTNOs are particularly advantageous for representing many-body operators, such as Hamiltonians, in systems with hierarchical correlations.

A TTNO for an operator $\hat{O}$ acting on a quantum system can be expressed as:

$$\hat{O} = \sum_{i_1,i_2,\ldots,i_N} \sum_{j_1,j_2,\ldots,j_N} \mathcal{O}^{(1)}_{i_1 j_1 i_2 j_2 \ldots} \mathcal{O}^{(2)}_{\ldots} \cdots \mathcal{O}^{(R)}_{\ldots} |i_1, i_2, \ldots, i_N\rangle \langle j_1, j_2, \ldots, j_N|,$$

where:

- $\mathcal{O}^{(1)}, \mathcal{O}^{(2)}, \ldots, \mathcal{O}^{(R)}$: Operator tensors representing the nodes in the TTNO.
- $i_1, i_2, \ldots, i_N$: Physical indices corresponding to the input states.
- $j_1, j_2, \ldots, j_N$: Physical indices corresponding to the output states.
- Bond indices b_{ij}: Shared indices connecting internal operator tensors.

Each node in the TTNO represents an operator tensor that acts on a subset of the system's Hilbert space. The root node encodes the global operator properties, while the hierarchical structure enables efficient encoding of local and non-local correlations.

Applications of TTNOs

The versatility of TTNOs has led to their adoption in various applications within quantum physics and beyond. A primary application is in **Hamiltonian representation**, where TTNOs are used to encode hierarchical Hamiltonians, including those with long-range or decaying interactions. This capability makes TTNOs particularly useful

for systems where operator complexity scales with interaction range.

TTNOs are also valuable for encoding **density matrices** in mixed-state simulations. By representing these matrices hierarchically, TTNOs enable efficient computation of thermal properties and other statistical measures in quantum systems. Another critical application is in **quantum dynamics**, where TTNOs serve as propagators for TTNs, facilitating the time evolution of quantum states in a computationally efficient manner. Furthermore, TTNOs have been explored in **quantum machine learning**, where their hierarchical structure enables the efficient representation and computation of quantum operators in machine learning models.

Comparison to Matrix Product Operators (MPOs)

While Matrix Product Operators (MPOs) are highly effective for representing operators in linear systems, TTNOs extend this capability to hierarchical systems with complex correlations. The key difference lies in their **topology**: MPOs are restricted to a linear structure, whereas TTNOs use a tree-like structure that allows for more efficient encoding of non-linear systems. Additionally, TTNOs demonstrate superior **scalability** for systems with hierarchical correlations or sparse operators, making them a powerful alternative to MPOs for applications requiring multi-scale representation.

3.5. State Diagrams for TTNOs

State diagrams serve as a graphical representation that maps the structure of Tree Tensor Network Operators (TTNOs) onto a tree topology. According to Milbradt et al. [17], a state diagram is a formalism that captures the connections and relationships between various operator tensors in a TTNO. This diagrammatic representation is crucial for visualizing the complex interrelations in multi-body quantum systems and for guiding the construction and optimization of TTNOs.

3.5.1. Definition and Structure

A state diagram consists of vertices and hyperedges. The vertices, denoted as $V = \{v_1, v_2, \ldots, v_n\}$, connect multiple hyperedges and represent the tensor contractions between these operators, corresponding to the shared indices in the tensor network. The hyperedges, denoted as $E = \{e_1, e_2, \ldots, e_m\}$, represent the operator tensors in the TTNO. Each hyperedge corresponds to an operator acting on a specific subset of sites in the quantum system.

We can basically form a state diagram $\mathcal{D}$ for a TTNO as:

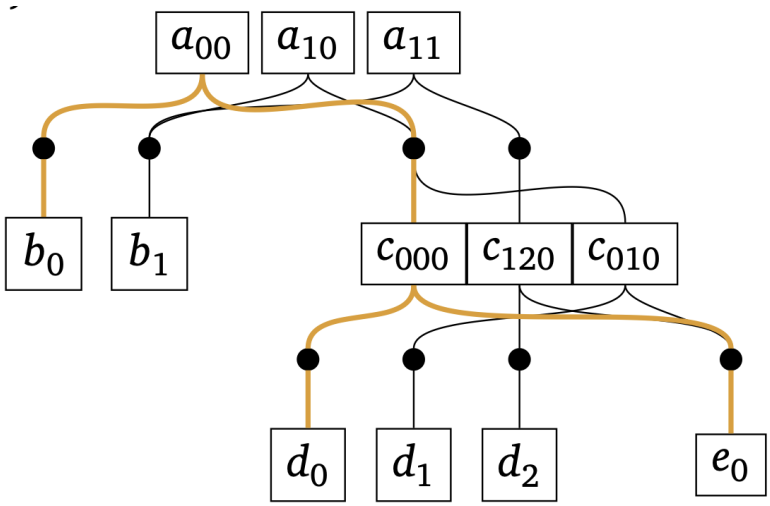

Figure 3.4.: Example of a state diagram for a TTNO. The vertices represent operator tensors, and the hyperedges represent the connections between these tensors. (from Milbradt et al[17])

- $\mathcal{D} = (V, E)$: The state diagram is defined by its set of vertices V and hyperedges E.
- $v_i \in V$: A vertex representing an operator tensor $\mathcal{O}_i$.
- $e_j \in E$: A hyperedge representing the contraction between operator tensors connected by e_j.
- p_i: A path through the state diagram, representing a sequence of tensor contractions.

3.5.2. Benefits of State Diagrams

State diagrams offer a clear and intuitive way to visualize the complex tensor networks underlying TTNOs. By representing operators and their interactions graphically, state diagrams help in understanding the structure and behavior of the TTNO. They also facilitate the identification of optimal contraction sequences and the minimization of bond dimensions, making them a valuable tool for both theoretical analysis and practical implementation.

This foundational framework for state diagrams is crucial for developing efficient algorithms to construct and manipulate TTNOs, as detailed in subsequent sections.

3.6. How to Construct Initial MPOs and TTNOs

Constructing Matrix Product Operators (MPOs) and Tree Tensor Network Operators (TTNOs) is a fundamental step in representing quantum systems efficiently. When the input operators are given in a **Sum-of-Product (SOP)** form, an analytical expression for the MPO can often be derived without requiring approximations. This section explores different approaches to constructing MPOs and TTNOs, detailing their advantages, limitations, and practical implications.

3.6.1. Construction Methods

We can investigate the construction of MPOs and TTNOs through three primary methods, each with its own advantages and challenges:

Naïve Construction

The **naïve construction** method involves directly translating the SOP form of the Hamiltonian or operator into an MPO or TTNO structure. While this approach is straightforward, it is **far from optimal**. The resulting structure often has unnecessarily high bond dimensions, leading to inefficient computations and increased resource usage. This inefficiency arises because the method does not attempt to minimize redundancies or optimize the tensor structure, making it impractical for complex systems.

Manual Design

The **manual design** method focuses on carefully constructing the MPO or TTNO structure based on the specific operators and interactions within the system. Although this approach can yield more efficient representations compared to the naïve construction, it is inherently **labor-intensive** and prone to errors. This is particularly true for systems with many different types of operators, where ensuring both correctness and efficiency requires detailed knowledge of the system and significant effort. As the complexity of the system grows, so does the risk of mistakes, making this approach difficult to scale effectively.

In essence, manual design involves compressing the naïve construction of the Hamiltonian. As explained in Hubig et al. (2017) [9], this includes combining identical terms early in the process, as well as merging repeated operator chains, such as long sequences of identity operators. Such compression techniques are not only fundamental to manual design but also play a crucial role in our proposed approach.

SVD-Based Automated Construction

The **SVD-based automated construction** approach leverages Singular Value Decomposition (SVD) to automatically decompose the SOP form into an optimized MPO or TTNO structure. While this method is less prone to manual errors and can produce compact representations, it has significant drawbacks. The process is **mathematically unstable**, especially for operators with small singular values, and can be computationally expensive. This instability and cost make SVD-based construction less appealing for large-scale or highly intricate systems.

3.6.2. Comparison of Construction Methods

Each construction method offers trade-offs between ease of implementation, computational efficiency, and stability:

1. **Naïve Construction:** Easy to implement but results in inefficient structures.
2. **Manual Design:** Produces more optimized results but is time-consuming and error-prone.
3. **SVD-Based Construction:** Automates optimization but can be unstable and costly and not repeatable.

3.6.3. Constructing TTNOs

The construction of Tree Tensor Network Operators (TTNOs) extends the principles used for MPOs but adapts them to a hierarchical structure. Instead of a linear chain, TTNOs use a tree topology to represent operators, which is particularly effective for systems with long-range or multi-scale interactions. While the basic steps of construction (e.g., translating the SOP form into operator chains) remain the same, additional care must be taken to align the hierarchical structure with the system's interaction geometry.

3.6.4. Conclusion

The construction of MPOs and TTNOs is a critical step in efficiently representing quantum systems. Each method for constructing these structures presents unique advantages and challenges, with the choice of method largely depending on the specific requirements of the system and the available computational resources. While existing approaches, such as manual design and naïve construction, offer some utility, they fall short of providing a generalized and efficient solution. There is a pressing need for an automated design process that can generate compressed terms for any given

Hamiltonian. Ren et al. (2020) [15] proposed an automated solution, which we will analyze in detail, but their approach has limitations in addressing all possible cases. Bridging this gap is the primary goal of our research, as we aim to enhance the method proposed by Ren et al. (2020) and develop a more robust framework for automating the construction of optimized and compressed MPOs and TTNOs. By improving this approach, we seek to advance the efficiency and scalability of tensor network-based quantum simulations significantly.

4. Methodology - Algorithm

The primary goal of our algorithm is to construct Matrix Product Operators (MPOs) or Tree Tensor Network Operators (TTNOs) in an efficient and scalable manner. These constructions are critical for representing quantum systems with minimal computational overhead while maintaining accuracy. The focus is on developing methods that optimize the construction process underlying tensor structures. To achieve this, we explore techniques that minimize bond dimensions and adapt these methods to the hierarchical structures inherent in TTNOs.

The thesis is divided into two primary parts: the optimization of bond dimensions for any quantum state representation using bipartite graph theory with a symbolic Gaussian elimination preprocessing step, followed by the extension and adaptation of this method to Tree Tensor Network Operators (TTNOs). Each part addresses specific challenges and provides a novel contribution to the efficient construction and representation of quantum systems.

Since the core optimization in tree structures is fundamentally based on the optimization of a bond, we begin by thoroughly analyzing and understanding the optimization process for a given pair of sites. Once this is established, the remaining challenge lies in extending the same optimization procedure to the tree structure. The systematic methodology presented in this section ensures that the developed algorithm remains efficient and robust across various quantum system configurations.

4.1. Part I: Optimization of Bond Dimensions

The first part of our research focuses on optimizing bond dimensions, which is a key factor in constructing efficient tensor networks. We started by studying an existing method based on bipartite graph theory. This method aims to find the best bond dimensions for a quantum system, minimizing computational costs while keeping the representation accurate. The method uses the structure of bipartite graphs to simplify the problem, making it manageable for certain cases. However, after a detailed analysis, we found that this approach is limited. It can only handle a subset of the scenarios commonly encountered in quantum systems, leaving important cases non-optimal.

This limitation shows the need for a more flexible and complete solution.

To address these limitations, we present a new and robust solution designed to handle critical cases that the existing method cannot solve but are often encountered in practical applications. Our approach adds some complexity by including techniques such as a symbolic Gaussian elimination preprocessing step, enabling it to optimize cases where the bipartite graph algorithm falls short. Although the new method demands more computational resources and has a slower runtime in certain situations, it significantly improves efficiency by optimizing bond dimensions for these challenging cases. This improved algorithm provides a compact representation of quantum systems across a broader range of scenarios, ultimately reducing computational costs in subsequent tensor network operations.

This section begins with a detailed analysis of the bipartite graph algorithm, providing a re-interpretation of the underlying problem to explain its theoretical foundations and inherent limitations. We will systematically highlight the shortcomings of the existing approach, illustrating its failure in addressing certain classes of quantum systems. Finally, we will introduce our novel algorithm, demonstrating how it resolves these critical cases while maintaining a balance between computational complexity and efficiency.

4.1.1. Existing Method Implementation and Analysis

Our base algorithm was introduced by Ren et al. (2020) [15] and is both well-established and thoroughly documented, complete with detailed pseudocode. The original paper provides comprehensive insights for those seeking a more in-depth explanation. Here, we briefly outline the general process, as it serves as a foundational component of our work. To assess its functionality and reliability, we began by replicating the algorithm's implementation. As expected, the algorithm performed well, accurately identifying optimal bond dimensions in most cases. However, our experiments also highlighted certain edge cases where the bipartite graph approach fails, demonstrating that the method is not entirely comprehensive.

To understand how the algorithm works, it is important to first examine how the terms are structured. Here, we will focus on the MPO structure, as it forms the basis of the initial algorithm. While we will elaborate on the optimization of virtual bond dimensions for a single site — since the algorithm is iterative and applies the same technique to each site —understanding the overall MPO construction is key to grasping the problem. Each term in the MPO is represented as a chain of operators, with each

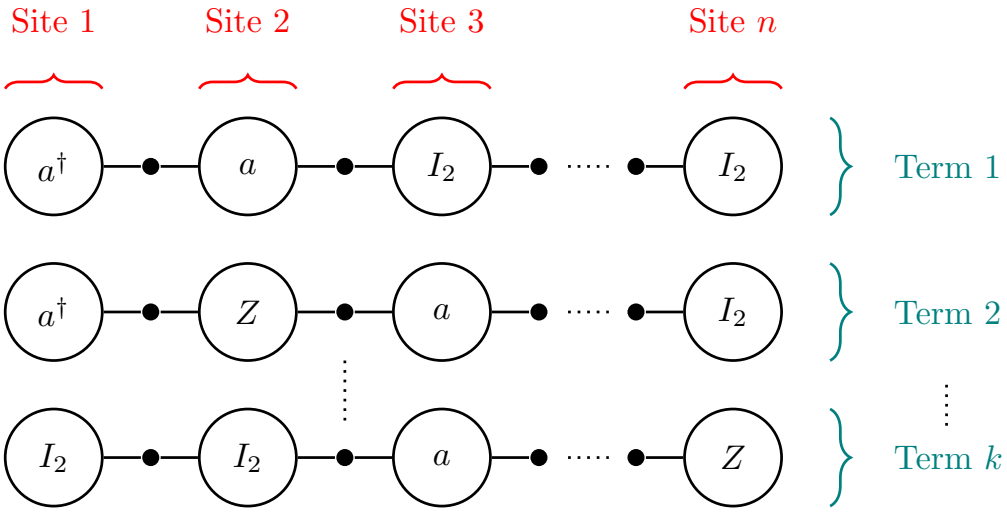

Figure 4.1.: The MPO structure is illustrated here, with each site marked in red and each chain represented in green. Operators are assigned randomly to serve as an example. For simplicity, the physical legs of the nodes are not shown in the diagram, though they are present in the actual structure.

local operator acting on a specific site. Suppose we have k terms and n sites; the edges between two adjacent sites play a critical role. The number of edges determines the virtual bond dimension between the MPO sites, and the algorithm's goal is to minimize these edges for every site-to-site connection as much as possible. Initially, there are k parallel chains, resulting in k edges between each of the $n-1$ connections. This initial structure lays the groundwork for the optimization process. We can see the explained structure in figure 4.1.

The algorithm begins with this initial trivial MPO construction. While this formation is a valid MPO representation, it features k dots between each pair of sites, indicating a virtual bond dimension of k. This is the maximum possible bond dimension and poses significant challenges for the algorithms that follow. Therefore, the goal is to minimize the bond dimensions as much as possible for each site. To achieve this, the algorithm optimizes the bond connections through a sweeping process, moving systematically from one end of the chain to the other, specifically from left to right.

We will refer to the left chains as U-chains and the right chains as V-chains. For each

edge in the network, the algorithm proceeds through the following steps to optimize the structure and reduce bond dimensions:

- **Create Non-Redundant Operator Sets (U and V):**
  The first step involves generating non-redundant sets of chains for the left (U) and right (V) sides. In this process, the algorithm compares entire chains on each side and removes duplicated chains, where a duplicated chain is defined as one in which every local operator matches exactly. After this step, the U-set contains only the unique chains from the $\tilde{U}$-chains, and similarly, the V-set contains the unique chains from the $\tilde{V}$-chains. This process ensures that the sets U and V are simplified by retaining only unique chains while eliminating redundancies, providing a clean and compact structure for further steps.

- **Preserve Connectivity Between U and V Sets:**
  Once the U and V sets are formed, the algorithm ensures that the connectivity between these sets is preserved. Specifically, connections are created between nodes in U and V based on the original edges in the $\tilde{U}$- and $\tilde{V}$-chains before their unification in step 1. This step guarantees that the structural integrity of the network is maintained, allowing the algorithm to operate on a consistent and accurate representation of the system.

- **Apply the Bipartite Algorithm:**
  In this step, the algorithm constructs a bipartite graph with the U and V sets as the two partitions. To optimize the connectivity between these sets, the algorithm first applies the Hopcroft-Karp algorithm to identify a **maximum matching**, which is a set of edges such that no two edges share a vertex. Once the maximum matching is determined, the algorithm uses it to calculate the **minimum vertex cover**, which is the smallest set of vertices that collectively covers all edges in the graph. The nodes included in this vertex cover are then selected to preserve the necessary connections while minimizing redundancy in the network. This process ensures an efficient and compact structure for the tensor network representation. The details of the Hopcroft-Karp algorithm and node selection process are explained in details in the following parts.

- **Form New U Chains for the Next Iteration:**
  In this step, the nodes identified in step 3 are used to update the U and V chains. As the optimization progresses from left to right, the newly optimized V-chains are prepared to serve as the U-chains for the next iteration. During this process, particular care is taken in assigning the γ coefficients, as they play a critical role in preserving the correct weights and relationships between the chains. Ensuring the

accurate assignment of these coefficients is essential to maintaining the integrity of the updated structure and accurately representing the original quantum system.

By systematically applying these steps to each edge, the algorithm incrementally optimizes the bond dimensions, ensuring an efficient and compact representation of the tensor network while maintaining the required connectivity.

4.1.2. Hopcroft-Karp Algorithm and Minimum Vertex Cover Detection

The Hopcroft-Karp algorithm is used to compute the **maximum cardinality matching** in the bipartite graph setting between sites. This matching forms the foundation for constructing an efficient tensor network by minimizing unnecessary connections. Once the maximum matching is identified, we leverage **König's theorem** to determine the **minimum vertex cover**, which is crucial for optimizing the structure of the tensor network. This subsection provides a detailed explanation of the Hopcroft-Karp algorithm and its role in detecting the minimum vertex cover.

Hopcroft-Karp Algorithm: Maximum Cardinality Matching

The Hopcroft-Karp algorithm is designed to efficiently find the maximum matching in a bipartite graph [38]. A matching in a graph is a set of edges such that no two edges share a vertex. The **maximum cardinality matching** is the largest possible matching in terms of the number of edges.

The algorithm operates in alternating phases of **breadth-first search (BFS)** and **depth-first search (DFS)**:

- **BFS Phase:** This phase builds a level graph by identifying the shortest augmenting paths, which are paths that start and end with unmatched vertices and alternate between edges not in the matching and edges in the matching.
- **DFS Phase:** Once the level graph is constructed, DFS is used to find and augment the matching along these shortest paths. Augmentation involves flipping the edges along the path, adding unmatched edges to the matching, and removing previously matched edges.

The algorithm alternates between these phases until no augmenting paths remain, at which point the matching is guaranteed to be maximal. Following pseudocode 1 illustrates the high-level flow of the algorithm that we implemented.

Algorithm 1: Hopcroft-Karp Algorithm for Maximum Cardinality Matching

Input: Bipartite graph $G = (U, V, E)$
Output: Maximum matching M

Function `HopcroftKarp(`G`)`**:**
  $M \leftarrow \emptyset$ `// Initialize an empty matching`
  while `connect_unmatched_vertices_BFS(`G, M`)` **do**
    foreach *unmatched vertex* $u \in U$ **do**
      `add_augmenting_path_DFS(`u, G, M`)` `// Augment along paths`
  return M
Function `connect_unmatched_vertices_BFS(`G, M`)`**:**
  $L \leftarrow$ `BuildLevelGraph(` G, M`)` `// Construct the level graph`
  return `HasAugmentingPaths(`L`)`
Function `add_augmenting_path_DFS(`u, G, M`)`**:**
  if u *is unmatched* **then**
    $v\_temp \leftarrow suitable\_v$ `// Find a suitable v to augment path`
    if `add_augmenting_path_DFS(`$pair\_of\_v, G, M$`)` **then**
      `AugmentMatching(`u, M`)` **return True**
    return False
  return True

Using König's Theorem to Find the Minimum Vertex Cover

Once the maximum cardinality matching is identified using the Hopcroft-Karp algorithm, we utilize **König's theorem** to find the minimum vertex cover. König's theorem states that in a bipartite graph, the size of the maximum matching is equal to the size of the minimum vertex cover. This property allows us to directly deduce the vertex cover from the matching.

The process involves:

- Identifying unmatched vertices in one partition of the bipartite graph.
- Performing a traversal to find the alternating paths relative to the matching.
- Determining which vertices belong to the vertex cover based on their involvement in these alternating paths.

The result is the smallest set of vertices collectively covering all edges in the graph, ensuring the minimum connections for the TN while preserving its structure.

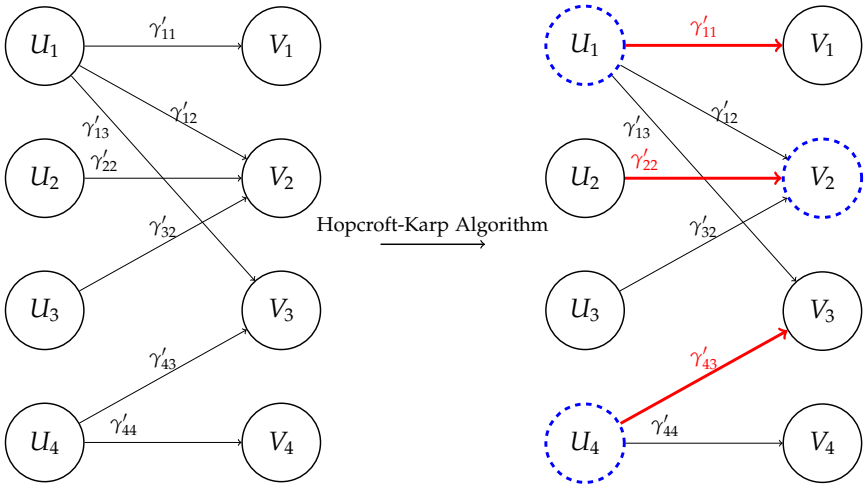

Figure 4.2.: Here we have an example bipartite graph, before and after the algorithm. The edges in the red form a maximum matching detected via Hopcroft-Karp algorithm. The blue dashed vertices form a minimum vertex cover.

Summary of Hopcroft-Karp Algorithm

In Figure 4.2, the process of applying the Hopcroft-Karp algorithm to a bipartite graph is illustrated. The algorithm begins by identifying the **maximum matching**, represented by the red edges in the graph. Once the maximum matching is established, the algorithm utilizes **König's theorem** to determine the **minimum vertex cover**, depicted by the blue nodes. The resulting blue nodes ensure all edges in the graph are covered with the minimal number of vertices, optimizing the graph's structure for further tensor network applications.

4.1.3. Selection of Nodes and Assignment of Gamma Coefficients

Selection of Nodes

Let us elaborate on why finding the minimum vertex cover is important and how to interpret the selection of nodes. In the bipartite graph, each edge represents a unique term, corresponding to the chain formed by the operators on either side of the edge. In the beginning, we ensure that no duplicate terms with different coefficients exist, as such terms can be easily combined into one by summing their coefficients. This guarantees that there will be no multiple edges between the same pair of vertices in our graph representation. With this understanding, it becomes essential to preserve information about all edges to retain every term of the Hamiltonian. Hence, spanning

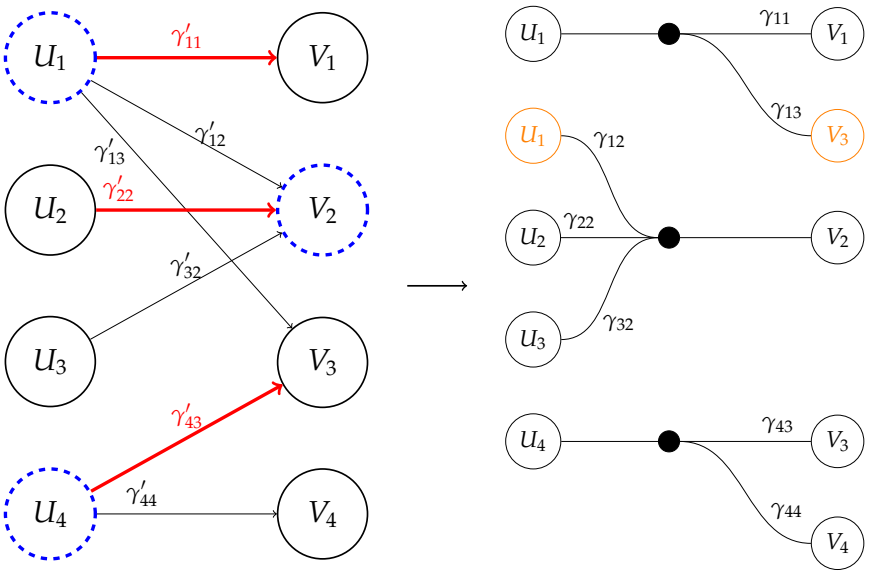

Figure 4.3.: The selection of nodes in the minimum vertex cover and assignment of the gamma coefficients.

all edges of the graph for each iteration is critical.

The key property of the minimum vertex cover is that it allows us to span all edges using only the vertices included in the vertex cover. This property drives our procedure. We group edges (i.e., terms) based on the nodes in the vertex cover. For each selected node, we create a connection dot, combining the edges associated with that node. The final result is that the number of connection dots corresponds to the number of vertices in the minimum vertex cover, which directly determines the virtual bond dimension.

As we iterate from left to right in our graph, the first nodes of the V-chains become the head of the U-chains in the subsequent iteration. When a U-node is selected, the corresponding V-chains are combined into a single dot, with each V-chain representing a different term. Conversely, when a V-node is selected, the U-chains are combined together, meaning the selected V-node now consists of multiple different terms.

This process results in the formation of what is referred to as a **complementary operation** or a **compressed operation**. The newly constructed node represents the summation of the combined terms, allowing it to serve as a compact representation for subsequent operations. This compression is a crucial step in reducing redundancies

and optimizing the structure for the following stages of the algorithm.

Gamma coefficients

When we say each edge represents a different term, we must consider the corresponding gamma coefficient of each term. Specifically, each edge carries the information of its associated gamma coefficient. During the construction process, our goal is to store these gamma coefficients within the nodes, as each node represents a matrix of local operators. For each term, it is crucial that only one local operator is multiplied by the corresponding gamma coefficient. Therefore, when grouping the U- and V-chains, we must also keep track of the positions of the gamma coefficients to ensure they are properly assigned.

Whenever we select a U-node, we push the gamma coefficients to the right, onto the V-nodes. Since there are distinct chains for each term on the right-hand side, the coefficients need to propagate through the next terms accordingly. On the other hand, when we create a complementary operation, the newly formed V-node will consist of multiple terms from the left-hand side. In this case, we cannot push different gamma coefficients into the same node. Instead, we store the coefficients in the U-nodes, and the new chain will have a coefficient of 1 for the remainder of the iteration to avoid recalculating the coefficients.

In summary, this process can be understood as an effort to push the gamma coefficients as far to the right as possible during the iterations. However, when a complementary operation is performed (i.e., combining left chains), the nodes involved in that operation take responsibility for storing the ongoing coefficients, and the iterations proceed with a coefficient value of 1.

Referring to Figure 4.3, we can observe the details of this assignment. In this example, the minimum vertex cover consists of the nodes U_1, V_2, and U_4. For the V_2 node, the coefficients are stored on the left-hand side nodes, while for U_1 and U_4, V-chains are created, and the coefficients are pushed to the right. The figure also illustrates how edges are grouped and combined during this process, highlighting the flow of gamma coefficients and the formation of new chains.

4.1.4. Gamma Matrix Interpretation

We can approach the bond optimization problem from an alternative perspective, which not only reveals why bipartite graph theory is insufficient for determining optimal

bond dimensions in certain cases but also guides the development of a natural follow-up algorithm. To illustrate this, let us represent the connection between two sites as a matrix, denoted by Γ. The Γ matrix can be seen as a modified adjacency matrix, reflecting the connectivity of the bipartite graph. Each row corresponds to the graph's left-hand side U-nodes, while each column represents the right-hand side V-nodes.

$$\Gamma = \begin{array}{c|cccccc} & \mathrm{V}_1 & \mathrm{V}_2 & \dots & \mathrm{V}_j & \dots & \mathrm{V}_m \\ \hline \mathrm{U}_1 & \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1j} & \cdots & \gamma_{1m} \\ \mathrm{U}_2 & \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2j} & \cdots & \gamma_{2m} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ \mathrm{U}_i & \gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{ij} & \cdots & \gamma_{im} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ \mathrm{U}_n & \gamma_{n1} & \gamma_{n2} & \cdots & \gamma_{nj} & \cdots & \gamma_{nm} \end{array}$$

An entry γ_{ij} in the matrix indicates the connectivity between the i-th U-node and the j-th V-node. The values of these entries correspond to the γ-coefficients of the terms, representing the strength of the connection between these nodes.

As explained previously, preserving the connectivity described by the Γ matrix is essential to accurately represent the Hamiltonian. Each coefficient in the matrix corresponds to an operation chain, representing a Hamiltonian term. Thus, each γ-coefficient plays a crucial role in maintaining the system's structure. We can again think about retaining each connection exactly as it is and follow the structure of the Γ matrix. While this approach is valid, it is still far from optimal and our primary objective was to construct an efficient representation of the Hamiltonian, allowing us to run algorithms on the resulting structure more effectively.

We can analyze the rank k of the Γ matrix, which represents the number of linearly independent rows or columns. This approach is motivated by the Singular Value Decomposition (SVD) of the matrix. The rank k provides insight into the minimal number of terms and complementary operations necessary to accurately represent the Hamiltonian. By determining the rank, we ensure that the essential structure of the

Hamiltonian is captured using the optimal number of terms, as outlined below:

$$\Gamma = \begin{array}{c|cccccc} & V_1 & V_2 & \dots & V_j & \dots & V_m \\ \hline U_1 & \gamma_{11} & \gamma_{12} & \dots & \gamma_{1j} & \dots & \gamma_{1m} \\ U_2 & \gamma_{21} & \gamma_{22} & \dots & \gamma_{2j} & \dots & \gamma_{2m} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ U_i & \gamma_{i1} & \gamma_{i2} & \dots & \gamma_{ij} & \dots & \gamma_{im} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ U_n & \gamma_{n1} & \gamma_{n2} & \dots & \gamma_{nj} & \dots & \gamma_{nm} \end{array} = \begin{bmatrix} \alpha_{11} \\ \alpha_{12} \\ \vdots \\ \alpha_{1n} \end{bmatrix} \begin{bmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1m} \end{bmatrix} + \begin{bmatrix} \alpha_{21} \\ \alpha_{22} \\ \vdots \\ \alpha_{2n} \end{bmatrix} \begin{bmatrix} \beta_{21} & \beta_{22} & \dots & \beta_{2m} \end{bmatrix} + \dots + \begin{bmatrix} \alpha_{k1} \\ \alpha_{k2} \\ \vdots \\ \alpha_{kn} \end{bmatrix} \begin{bmatrix} \beta_{k1} & \beta_{k2} & \dots & \beta_{km} \end{bmatrix}$$

As expected, deriving such a decomposition can be computationally intensive. The bipartite graph algorithm addresses this challenge by simplifying the problem through additional constraints on the terms. Specifically, it requires that one of the vectors (either a row or a column) must be a unit vector. In practical terms, this means the algorithm selects certain rows or columns directly from the Γ matrix to construct the decomposition. To explore this operation further, let us examine the following Γ_e matrix in detail:

$$\Gamma_e = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ 0 & \gamma_{22} & 0 & 0 \\ 0 & \gamma_{32} & 0 & 0 \\ 0 & 0 & \gamma_{43} & \gamma_{44} \end{bmatrix}$$

This is the same example with the figure 4.2. For this toy example, it is trivial to see that selecting Row_1, $Column_2$ and Row_4 is enough to decompose the Γ_e as:

$$\Gamma_e = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ 0 & \gamma_{22} & 0 & 0 \\ 0 & \gamma_{32} & 0 & 0 \\ 0 & 0 & \gamma_{43} & \gamma_{44} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \gamma_{22} \\ \gamma_{32} \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & \gamma_{43} & \gamma_{44} \end{bmatrix}$$

As observed, we achieve a rank-3 solution by selecting specific rows and columns from the Γ_e matrix. This result is derived by applying the Hopcroft-Karp algorithm to the bipartite graph representation, as previously discussed. The selected rows and columns correspond to the elements of the minimum vertex cover of the Γ_e matrix. This demonstrates how the decomposition process facilitated by the Hopcroft-Karp algorithm translates into mathematical terms, providing a clear connection between the graph-based method and the matrix representation.

4.1.5. Identified Limitations in Bipartite Graph Algorithm

While the existing method effectively reduces bond dimensions in most cases, we identified some configurations where it fails to achieve the true minimal bond dimension. Although Ren et al. (2020) [15] provide a scientific proof asserting the completeness of the algorithm, the assumptions underlying this proof are insufficient to guarantee its applicability in all scenarios. To address these issues, let us first examine the problematic aspects of the algorithm's approach.

The bipartite algorithm selects edges from only one side of the bond by choosing vertices from either the U or V set, resulting in connections that are either on the left or the right side. To better illustrate this, consider Figure 4.3. In the diagram, the algorithm selects $\hat{U}_1, \hat{V}_2, \hat{U}_4$. Selecting $\hat{U}_1$ involves combining the connected vertices on the left side. Similarly, selecting $\hat{V}_2$ combines three connected vertices on the right. However, this approach is limited and fails when combining both sides, i.e., when dealing with doubly connected sites.

Rephrasing the problem in terms of the Γ matrix, the bipartite graph algorithm is limited to selecting either a single column or row when forming a sum term in the decomposition. However, in some cases, it may be more efficient to combine both rows and columns simultaneously rather than restricting the operation to one side. More specifically, when we have a uniform value for the term coefficients, this kind of combination becomes fairly common and required.

Let's analyze the Hamiltonian H_f, which is a minimal example where we can see the non-optimality of the approach:

$$H_f = \sum_{j=1}^{4} h_j = X_1Y_2 + X_1X_2 + Y_1Y_2 + Y_1X_2$$

In Figure 4.4, we compare the solution proposed by the algorithm with the actual optimal bond configuration. The Hopcroft-Karp algorithm results in a virtual bond dimension of 2, as it selects nodes only from one side of the bipartite graph. In this example, we assume it chooses the left side, although the opposite is equally possible. Regardless of the choice, the solution will consist of two dots, limiting the representation to have an non-optimal virtual bond dimensions.

In contrast, the optimal configuration involves creating a fully connected node, where every node is connected to all others. This fully connected structure allows for capturing all 4 terms accurately. To understand this in detail, consider how chains are

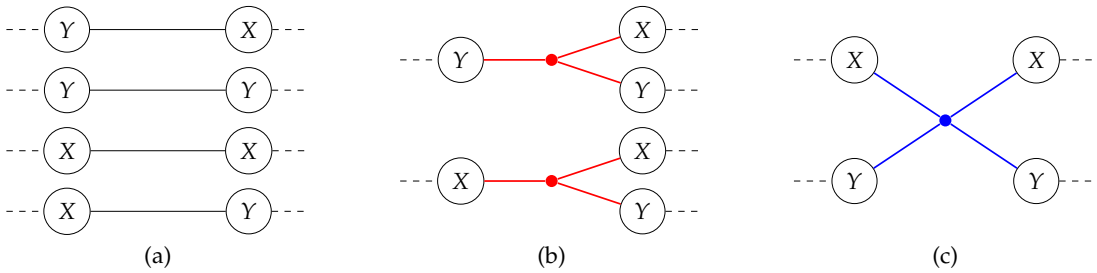

Figure 4.4.: a) The initial configuration of the chains. b) The result of the bipartite-graph algorithm with virtual dimension 2. c) Actual optimal solution with a both side connected node.

processed: for each line traversing from left to right in the structure, a distinct term is produced. In the fully connected case, all 4 terms can be represented, demonstrating the limitations of the algorithm in achieving the true minimal bond configuration.

Let's check the problem in terms of Gamma matrices to understand more analytically:

$$\Gamma_f = \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} \gamma_{11} & \gamma_{12} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} \gamma_{21} & \gamma_{22} \end{bmatrix}$$

Examining H_f, we observe that the coefficients are identical, with a value of 1 for all 4 terms. More generally, this can be expressed as $\gamma_{11} = \gamma_{12} = \gamma_{21} = \gamma_{22} = \gamma$. With this generalization, the decomposition can be rewritten as:

$$\Gamma_f = \begin{bmatrix} \gamma & \gamma \\ \gamma & \gamma \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} \gamma & \gamma \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} \gamma & \gamma \end{bmatrix}$$

As one can see easily, two terms can be combined into one as they have the same row vector. With this basic operation, we can represent Γ with just one term as follows, which corresponds to the fully connected graph in the figure 4.4:

$$\Gamma_f = \begin{bmatrix} \gamma & \gamma \\ \gamma & \gamma \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} \gamma & \gamma \end{bmatrix}$$

We were able to construct a counterexample that challenges the generality of the initial algorithm with relative ease. This phenomenon, as demonstrated in this case, can also arise in more complex systems. The bipartite graph algorithm is inherently limited in its ability to handle many-to-many connections, specifically cases where row or column vectors in the decomposition are not unit vectors. Consequently, an enhancement to the algorithm is necessary to address these situations and provide a

more comprehensive and robust solution.

As can be seen from the example, to create such a case, we set the coefficients all equal to each other. While such patterns may not be commonly encountered in chemical models, they are crucial in lattice models often studied in physics, where many Hamiltonian terms share the same coefficient.[18] Ignoring this pattern could lead to sub-optimal representations. Therefore, it becomes essential to develop methods that specifically address this configuration.

4.1.6. Proposed Improvement - Symbolic Gaussian Elimination

A straightforward approach for achieving optimal bond dimensions is the application of Singular Value Decomposition (SVD). While SVD effectively identifies the rank of a matrix and optimizes its representation, it comes with notable drawbacks. Firstly, as discussed earlier, it is computationally expensive and can be mathematically unstable. Secondly, its outcome highly depends on the specific values of the γ-coefficients. As a result, any change in the γ-coefficients, even a minor one, requires re-running the optimization process.

In contrast, the bipartite graph algorithm offers a significant advantage: it operates independently of the specific values of the γ-coefficients. Instead of relying on these values, the algorithm focuses on selecting rows or columns and determining the vertex cover of the Γ-matrix based solely on its structural properties. This independence from coefficient values is a key feature we sought to preserve in our work.

To address the encountered issues, we decided to retain the Hopcroft-Karp algorithm and introduce a pre-processing step to overcome its limitations. Specifically, our goal is to minimize the connectivity matrix (Γ) while preserving all essential information. Singular Value Decomposition (SVD) is one way to achieve this, but a more stable and symbolic solution is required due to explained concerns. To start from a simpler point of view, we can pre-process the Γ-matrix with a Gaussian elimination process to eliminate linearly dependent rows or columns. In that way, we can again determine the matrix's rank. However, as we want to stay symbolic, our Gaussian elimination method should also be working symbolically. That's where we come up with our variant of Gaussian Elimination:

Symbolic Gaussian Elimination

We propose a symbolic adaptation of Gaussian elimination, referred to as **Symbolic Gaussian Elimination**. This is a symbolic method in nature. In this method, the entries of the input matrix are stored and manipulated symbolically, represented as terms such as "3 times γ_1". During the row and column elimination steps, symbolic values are preserved to ensure that the process remains symbolic throughout. To maintain this representation, strict constraints are enforced.

Firstly, we do not allow combinations of symbolic values. Each entry of the Γ-matrix must remain in the form of "**a real coefficient** multiplied by **a symbolic gamma**". For instance, operations that result in terms like $\gamma_1 + \gamma_2$ are not permitted. This restriction simplifies the implementation for subsequent iterations, ensuring that the symbolic nature of the process is preserved and keeping the method practical and easily implementable. These operations are called **Restricted Row and Column Operations**, as they follow the standard principles of Gaussian elimination but with these additional constraints.

Secondly, we constrain the real coefficient in "a real coefficient multiplied by a symbolic gamma" to be **a fractional real number**. This constraint ensures mathematical stability while offering sufficient representation power. Initially, we attempted to limit coefficients to integers, but this approach severely restricted the representation capabilities of the matrix, leaving some trivial cases unsolvable by the algorithm. By allowing fractional real numbers, we significantly increase the representation scope while maintaining stable calculations. Although this approach imposes some limitations, it provides a practical balance, covering most cases effectively and ensuring the process remains stable and robust.

After applying the **Symbolic Gaussian Elimination** to update the Γ-matrix, the bipartite graph method is still required to determine the minimal virtual bond dimension. However, this step becomes unnecessary if all the entries in the Γ-matrix share uniform symbolic value. In such cases, having one identical symbolic gamma essentially reduces the problem to traditional Gaussian elimination, where the resulting matrix naturally reflects its rank.

On the other hand, when there is only partial uniformity among the coefficients or when all coefficients are symbolically distinct, the bipartite graph algorithm becomes indispensable. In these cases, the algorithm is required to determine the rank because, with coefficients being distinct, no simplification is possible due to the restriction

against combining symbolic values. Consequently, the process once again relies on the Hopcroft-Karp algorithm to select the appropriate row or column unit vectors, ensuring that the matrix is reduced in compliance with the symbolic constraints while preserving its essential structure.

As previously mentioned, "Symbolic Gaussian Elimination" serves as a preprocessing step and should not increase the overall complexity of the problem. Therefore, we apply the bipartite graph algorithm to both the initial Γ and the processed $\tilde{\Gamma}$ and compare the results. We proceed with Gaussian elimination only if the new matrix results in a lower virtual bond dimension. If the pre-processing step does not yield an improvement, we retain the initial configuration. This ensures that the performance of the previous algorithm is at least matched, with potential improvements in the best-case scenario.

Pre-processing Implementation details

Now, we can deep dive into the details of the algorithm. We introduce two operator matrices, O_l and O_r, both initially defined as identity matrices. These matrices are used to keep track of the transformations (operations) applied to the Γ-matrix. As we progress, these matrices will be updated to reflect the operations performed. Throughout the process, we maintain the following equality:

$$\Gamma = O_l \cdot \tilde{\Gamma} \cdot O_r$$

where $\tilde{\Gamma}$ is the updated Γ-matrix.

The matrix O_l, referred to as the **left operator matrix**, keeps track of the row operations applied to Γ. To ensure the equality above remains valid, the **columns** of O_l are updated with the **inverse of the operations** applied to the **rows** of Γ. By inverse, we mean that for addition, we subtract the same value, and for multiplication, we divide (or multiply by the reciprocal of the value).Additionally, the indices of columns and rows are reversed. For example, if the i-th row of Γ is added to the j-th row, this corresponds to subtracting the j-th column from the i-th column in O_l. This inverse operation ensures the overall multiplication remains consistent with the equation.

Similarly, O_r tracks the column operations applied to Γ. The updates to O_r follow the same principles as O_l, applying inverse operations. For instance, if we multiply the i-th column of Γ by a scalar c, we divide the corresponding i-th row of O_r by c.

If a zero row is encountered in the $\tilde{\Gamma}$-matrix at the j-th row, both the j-th row of $\tilde{\Gamma}$ and the corresponding j-th column of O_l can be removed. Similarly, if a zero column

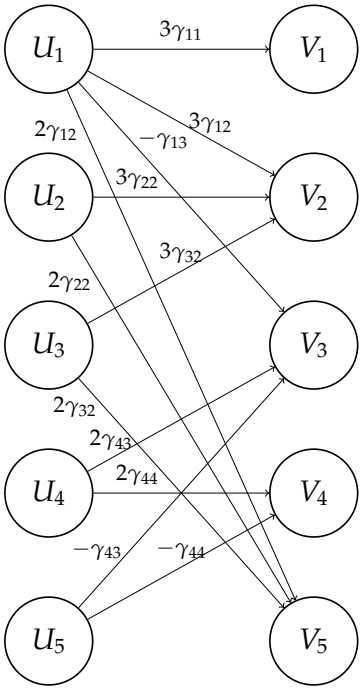

Figure 4.5.: A simple example bipartite graph setting to apply symbolic Gaussian elimination

is detected in the $\tilde{\Gamma}$-matrix at the j-th column, we eliminate both the j-th column of $\tilde{\Gamma}$ and the corresponding j-th row of O_r. These steps simplify the matrix and operator structures by removing redundant components.

By systematically applying these principles, we ensure that O_l, $\tilde{\Gamma}$, and O_r accurately represent the transformations applied to the Γ-matrix while maintaining the given equality. Once the "Symbolic Gaussian Elimination" process is complete, the resulting $\tilde{\Gamma}$-matrix will be ready for the application of the bipartite graph algorithm.

We will construct two virtual layers using the newly generated matrices. Leveraging the preserved equality $\Gamma = O_l \cdot \tilde{\Gamma} \cdot O_r$, the initial connectivity matrix can now be expressed as the product of these three matrices. Each of these matrices can also be interpreted as representing connectivity matrices. Consequently, the problem can be reformulated with two additional virtual layers, where the connectivities of these layers are described by O_l, $\tilde{\Gamma}$, and O_r.

To understand how this is done, let us illustrate and analyze this approach with a simple example shown in figure 4.5 corresponds to the matrix Γ_h:

$$\Gamma_h = \begin{bmatrix} 3\gamma_{11} & 3\gamma_{12} & -\gamma_{13} & 0 & 2\gamma_{12} \\ 0 & 3\gamma_{22} & 0 & 0 & 2\gamma_{22} \\ 0 & 3\gamma_{32} & 0 & 0 & 2\gamma_{32} \\ 0 & 0 & 2\gamma_{43} & 2\gamma_{44} & 0 \\ 0 & 0 & -\gamma_{43} & -\gamma_{44} & 0 \end{bmatrix}$$

If we were to apply the Hopcroft-Karp algorithm directly, the resulting virtual dimension would be 5, as it is not possible to cover the entire graph by selecting rows or columns. If we run the algorithm, it would select 5 nodes at either one of the sides. However, it becomes evident that some rows or columns can be actually eliminated, as they are linear combinations of others (especially some rows or columns are parallel to each other for this specific example). Therefore, we first employ the "Symbolic Gaussian Elimination" to obtain a reduced matrix $\tilde{\Gamma}$, and then proceed with the bipartite graph method.

After implementing the "Symbolic Gaussian Elimination", we can reach the following operator and Gamma and matrices. The details of the implementation and how derived these matrices can be seen in the Appendix A:

$$\Gamma = O_l \cdot \tilde{\Gamma} \cdot O_r$$

$$\begin{bmatrix} 3\gamma_{11} & 3\gamma_{12} & -\gamma_{13} & 0 & 2\gamma_{12} \\ 0 & 3\gamma_{22} & 0 & 0 & 2\gamma_{22} \\ 0 & 3\gamma_{32} & 0 & 0 & 2\gamma_{32} \\ 0 & 0 & 2\gamma_{43} & 2\gamma_{44} & 0 \\ 0 & 0 & -\gamma_{43} & -\gamma_{44} & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 3\gamma_{11} & 3\gamma_{12} & -\gamma_{13} & 0 \\ 0 & 3\gamma_{22} & 0 & 0 \\ 0 & 3\gamma_{32} & 0 & 0 \\ 0 & 0 & 2\gamma_{43} & 2\gamma_{44} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{2}{3} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

This new decomposition allows us to interpret the matrices as adjacency matrices with creation of 2 virtual layers in the middle of the sites which we call $\tilde{U}$ and $\tilde{V}$, shown as follows:

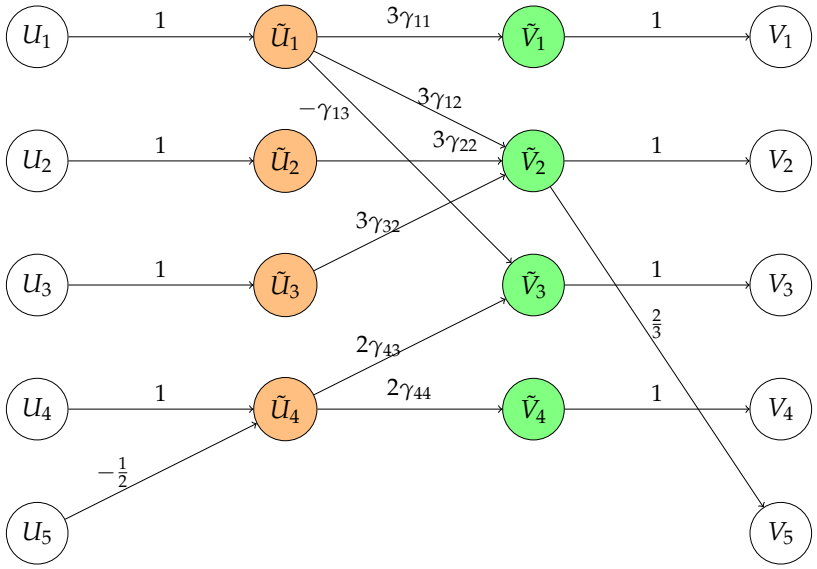

Figure 4.6.: The orange nodes represent the virtual U-nodes on the left-hand side, while the green nodes represent the V-nodes on the right-hand side.

	$\tilde{U}_1$	$\tilde{U}_2$	$\tilde{U}_3$	$\tilde{U}_4$
U_1	1	0	0	0
U_2	0	1	0	0
U_3	0	0	1	0
U_4	0	0	0	1
U_5	0	0	0	$-\frac{1}{2}$

	$\tilde{V}_1$	$\tilde{V}_2$	$\tilde{V}_3$	$\tilde{V}_4$
$\tilde{U}_1$	$3\gamma_{11}$	$3\gamma_{12}$	$-\gamma_{13}$	0
$\tilde{U}_2$	0	$3\gamma_{22}$	0	0
$\tilde{U}_3$	0	$3\gamma_{32}$	0	0
$\tilde{U}_4$	0	0	$2\gamma_{43}$	$2\gamma_{44}$

	V_1	V_2	V_3	V_4	V_5
$\tilde{V}_1$	1	0	0	0	0
$\tilde{V}_2$	0	1	0	0	$\frac{2}{3}$
$\tilde{V}_3$	0	0	1	0	0
$\tilde{V}_4$	0	0	0	1	0

Using these adjacency matrices, we can now identify the cut sites, as illustrated in Figure 4.6. The O_l matrix defines the first virtual cut site between U and $\tilde{U}$, while $\tilde{\Gamma}$ and O_r define the second and third cut sites, respectively. The $\tilde{U}$ and $\tilde{V}$ layers serve as representative layers, where each $\tilde{U}$ node represents a combination of real U nodes, and each $\tilde{V}$ node similarly represents a combination of real V nodes. Essentially, this approach creates a virtual layer with simplified connectivity, represented by the matrix $\tilde{\Gamma}$, while preserving the original connectivity of the system.

We can now apply bipartite graph theory to the inner cut site to determine the minimum vertex cover using the virtual nodes. The Hopcroft-Karp algorithm again does its magic to identify the maximum edge matching and the corresponding minimum vertex cover. Once this operation is complete, we obtain the minimal set of nodes required to

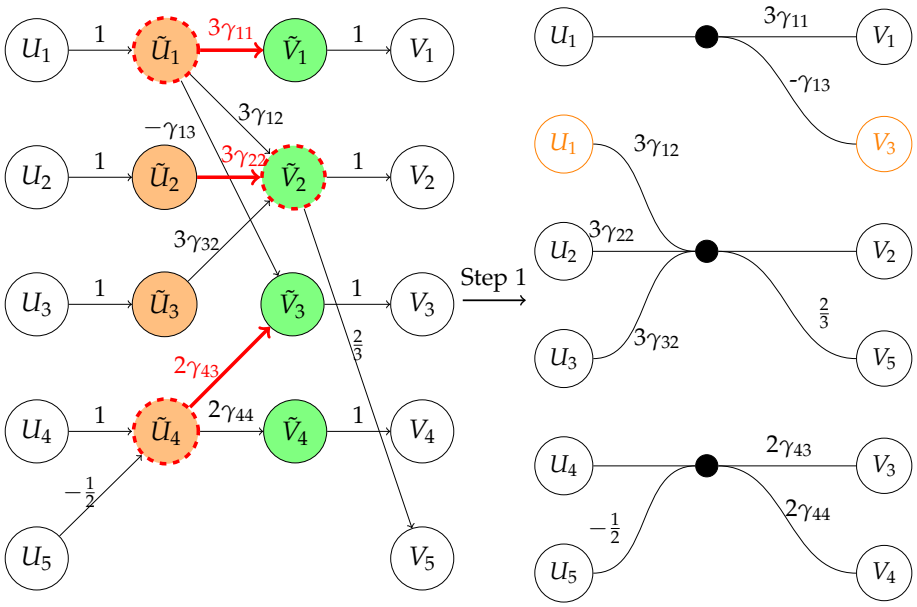

Figure 4.7.: On the left, we can see the red dashed nodes form the minimal span of the connectivity, with the red edges indicating the maximum matching, identified using the Hopcroft–Karp algorithm. The second figure is the realisation of the cut site.

represent the cut site with virtual nodes.

The final step is to replace the virtual $\tilde{U}$ and $\tilde{V}$ nodes with their corresponding real U and V nodes. The weights associated with the first and third cut sites must be carefully handled during this replacement. The coefficients generated during the operation are assigned to the local operators of the corresponding real nodes. Additionally, the gamma coefficients are pushed to the left or right, as described earlier in the Hopcroft-Karp section. We can see the details of the realisation in the figure 4.7.

Here we observe the result with a virtual bond dimension of 3, corresponding to the creation of 3 dots as determined by the minimal vertex cover, which consists of 3 nodes. The process begins with the $\tilde{V}_2$ node, located in the middle of the diagram. On its right-hand side, $\tilde{V}_2$ connects to two real nodes, V_2 and V_5, forming the right side of the second dot. On the left-hand side, $\tilde{V}_2$ connects to the virtual nodes $\tilde{U}_1$, $\tilde{U}_2$, and $\tilde{U}_3$. Each of these virtual nodes corresponds to only one real node, maintaining the simplicity of the representation.

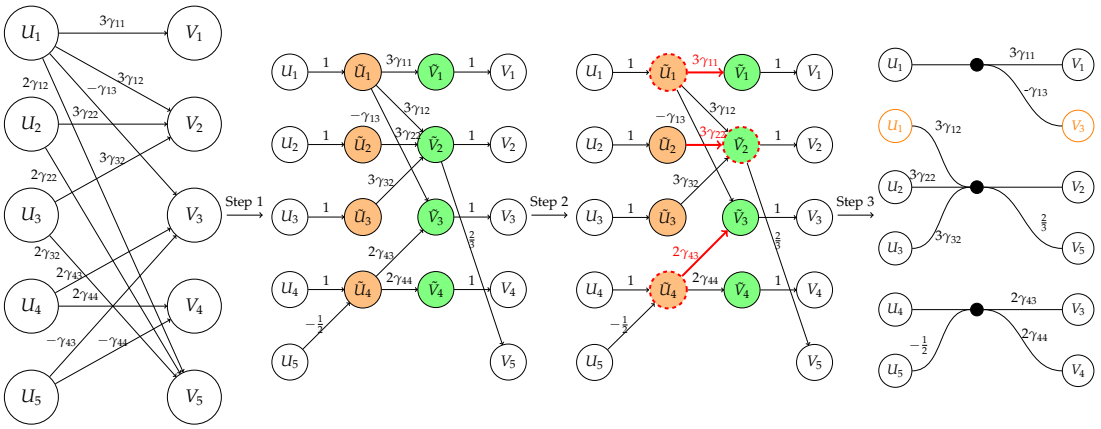

Figure 4.8.: The figure illustrates the creation of virtual layers and the outcome of the bipartite graph algorithm. In the final step, we can see the realization of optimization with virtual bond dimension 3 as the result.

Additionally, for the last dot, we replace the virtual node $\tilde{U}_4$ with the real nodes U_4 and U_5. This final replacement step ensures that all virtual nodes are correctly substituted with their corresponding real nodes while preserving the optimal virtual bond dimension of 3.

This process ultimately results in a more optimal virtual bond dimension for the given site, improving the efficiency of the tensor network representation. The whole process for this specific example summarized in the figure 4.8.

4.1.7. Algorithm Summary

The symbolic implementation of the algorithm is observed to be non-trivial. A partial pivoting strategy was adapted from the method described by Heath (2002, p. 73) [39], with modifications to adhere to the symbolic constraints. We can see the details of the algorithm outlined in the pseudocode "Algorithm 2". Here, we can see an additional de-parallelization step, which serves as a preprocessing stage to identify and handle parallel rows and columns. This step is crucial for reducing the complexity of the Gaussian elimination and addressing the limitations inherent in the symbolic algorithm. Also in the end, we check if there is an actual improvement to decide whether to keep the symbolic Gaussian elimination or not.

This algorithm offers significant advantages, particularly in its ability to preserve

the symbolic representation of the γ coefficients throughout the computation. By maintaining the symbolic form of the coefficients, we avoid numerical dependencies that typically arise when directly manipulating the values. While this symbolic representation may initially seem to distance us from an optimal solution, it is a minor sacrifice to have a more stable implementation.

Another key strength of this approach is its ability to handle uniform coefficient cases commonly encountered in lattice models. In these cases, where multiple or all of the terms in the Hamiltonian share identical coefficients, the bipartite graph approach struggles to provide an accurate representation. However, our algorithm is specifically designed to address this scenario, ensuring that uniform coefficients are properly represented without sacrificing accuracy. This makes the algorithm well-suited for applications in physics, where such uniformity in the coefficients often occurs in models like the Heisenberg or Hubbard models or lattice models.

Algorithm 2: Symbolic Gaussian Elimination and Bipartite Graph Optimization

Input : Matrix Γ with coefficients γ_{ij}
Output: Updated matrix $\tilde{\Gamma}$

Initialize two identity matrices O_l and O_r matching the row and column dimensions of Γ;
Detect parallel rows and columns and **De-parallelize** them.
foreach *row i in* Γ **do**
    foreach *column j in* Γ **do**
        if γ_{ij} *is non-zero* **then**
            Apply *restricted* row elimination on row i;
            Update O_l to track row operations;
            Apply *restricted* column elimination on column j;
            Update O_r to track column operations;
    if *row i in* Γ *is zero* **then**
        Remove row i from Γ and the corresponding row i from O_r;
    if *column j in* Γ *is zero* **then**
        Remove column j from Γ and the corresponding column j from O_l;
Apply bipartite graph algorithm to the reduced $\tilde{\Gamma}$ to determine minimum vertex cover;
if *minimum vertex cover is bigger than* Γ **then**
    Replace $\tilde{\Gamma}$ with Γ and continue with Γ;

Output: $\tilde{\Gamma}$, O_l, and O_r matrices

4.2. Part II: Application to TTNOs

In the previous section, we provided a broad explanation of the optimization process for a single bond site. It is important to note that the optimization of a bond site is independent of the overall system structure, whether it is an MPO, a TTNO, or another representation. However, in our explanation, we primarily referred to the MPO structure. This focus stems from the fact that MPOs are both the most commonly used representation method and the basis of the previous research upon which we built our approach.

In this section, we extend the improved optimization approach to the more general case of TTNOs. Unlike MPOs, TTNOs are organized in a hierarchical tree structure, presenting additional challenges and opportunities for optimization. This extension represents the second part of our research, broadening the applicability of the improved method to systems with complex and non-linear topologies.

4.2.1. Adapting the Optimization Algorithm to Tree Structure

We utilize the state diagram representation for TTNO structures, as proposed by Milbradt et al. (2024) [17]. The details of state diagrams were briefly introduced in Section 3.5 - State Diagrams. In this context, we refer to a state diagram representation consisting of multiple terms as a **compound state diagram**.

The algorithm operates on the compound state diagram with the objective of optimizing virtual bond dimensions by minimizing the number of vertices intersected by hyperedges. The starting configuration and the optimized result are illustrated in Figure 4.9. Unlike the MPO case, where the structure is linear and traversal is straightforward—progressing from one side to the other—TTNOs require a more intricate navigation strategy due to their hierarchical and non-linear tree structure.

In our approach, we utilize a Breadth-First Search (BFS) strategy, beginning at the root of the tree and systematically traversing its branches. The optimization of a vertex site is referred to as a *cut site*, as the process essentially involves "cutting" the tree at that vertex to optimize the connectivity and reduce the virtual bond dimensions.

The core of the algorithm remains unchanged: we apply bipartite graph theory, augmented with our pre-processing step using symbolic Gaussian elimination, to optimize a cut site. The previously proposed algorithm, tailored for MPO structures, processes sites in a linear fashion by accumulating or shifting the coefficients of terms from left

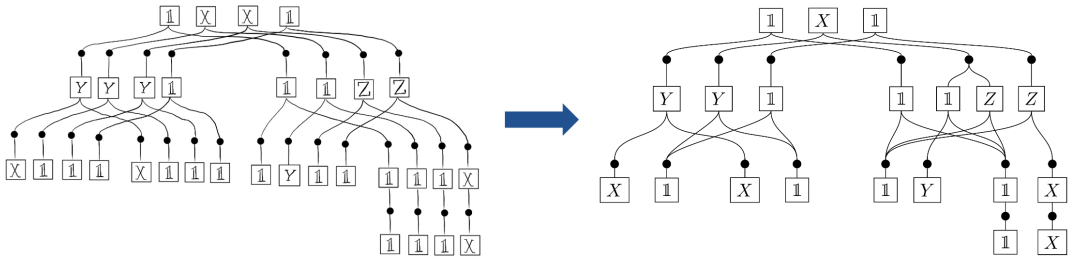

Figure 4.9.: The objective of the algorithm is to determine the optimal virtual bond dimensions between sites. This involves minimizing the number of vertices (represented as black dots) as the algorithm transitions from the initial tree structure on the left to the optimized structure on the right.

to right. At first glance, this approach may appear incompatible with the hierarchical nature of a tree structure, where there is no absolute "left" or "right." Nevertheless, the algorithm can be adapted to navigate and process tree structure

To understand this, let us revisit the "pushing" interpretation of the original algorithm: when optimizing a bond dimension, nodes are selected on either the left or right side after applying the Hopcroft–Karp algorithm. Selecting a left node pushes the γ-coefficients to the right chains, while selecting a right node leaves them in the left chains. Previously, we described this process as proceeding linearly from left to right, pushing coefficients as far right as possible. When a node from the right-hand side is selected, the coefficients are left in the left U-nodes. Extending this interpretation, we can also say that selecting a right-side node effectively pushes coefficients to the left.

This perspective reveals that even in the linear case, the MPO structure can be processed starting from any middle node. The key requirement is to traverse the structure without skipping nodes, ensuring that coefficients are pushed incrementally and gently. This flexible interpretation allows us to extend the approach to the tree structure as well. In the tree setting, we can start from any node, and as long as the coefficients are pushed methodically without skipping, the optimization result will remain the same and optimal, regardless of the starting node or traversal order.

In designing the implementation for the tree structure, our focus was on achieving the most computationally efficient solution. To this end, we explored approaches inspired by dynamic programming. While the order of processing does not influence the final optimization result, a strategically designed implementation can significantly

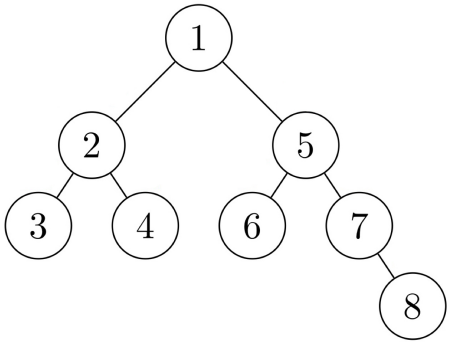

Figure 4.10.: Tree structure

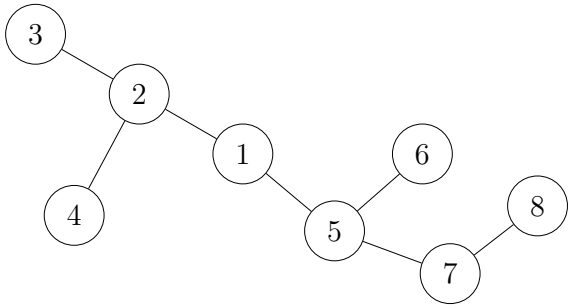

Figure 4.11.: Graph structure

Figure 4.12.: Graph and Tree interpretations side by side

improve the speed and efficiency of the algorithm.

4.2.2. Decision of the Tree Root and Orientation

Another important aspect to consider is how to determine a suitable tree configuration. This issue was not the primary focus of our research, as we assumed that the tree structure details would be provided to the algorithm and our implementation operates on the given configuration. However, in real-world problems, it is often necessary to convert a grid-like 2D system into a tree structure to represent it in classical computers. While it is not strictly required to use a tree structure—many systems are currently modeled as MPOs due to the availability of several well-established algorithms—we want to explore TTNOs more and for that it is essential to construct a tree structure.

It is important to recognize that the choice of the root node in our graph is largely arbitrary, as the nature of most systems does not inherently dictate a specific root. However, certain nodes may be more suitable as a root, depending on the system's characteristics, and constructing the tree structure from such a node could offer advantages. This aspect requires further investigation, which we partially explored during our experiments.

One might then question whether this arbitrary root choice or the flexibility in the tree structure could impact the algorithm's performance. Specifically, since we start from the root and process the tree toward its leaves, would changing the root alter the order of processing? The answer is yes, it would change the order. However, as discussed in the previous section, our traversal order is designed solely to ensure computational efficiency and has no effect on the optimization outcome. The algorithm's

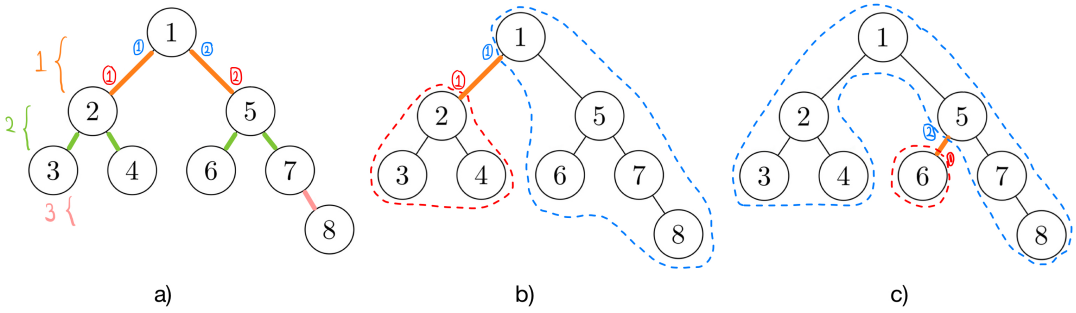

Figure 4.13.: **a)** Tree is traversed in a BFS manner but double-passing each level. The order of the traverse is first the 1. level (orange), then the second level (green), and finally the third level (pink). For each layer, we process each vertex site twice. For example, in the first layer, firstly, vertices are processed in the red numbering order, then blue. **b)** For each vertex site, we split the tree into two parts. The red part is *U-subtree*, and the blue part is *V-subtree*. **c)** Another example cut for the bond between sites 5 and 6.

results remain consistent regardless of the starting point. For instance, we could just as easily begin from a leaf node and traverse the tree while performing site optimizations, and still achieve the same optimal tree structure.

Throughout this section, we alternately use a tree structure and a graph representation to describe the given system. As illustrated in Figure 4.12, the two representations actually depict the same system. In one representation, we maintain the tree structure to facilitate the explanation of the algorithm, as the algorithm assumes the tree structure is already provided. However, to emphasize the arbitrary nature of the root choice, we also present the system using a graph representation. This highlights that while the algorithm relies on a tree structure for its operation, the underlying system can equally be viewed as a graph.

4.2.3. Tree Structure Traversal

For each site optimization, the tree is effectively divided into two parts, which we refer to as *U-subtrees* and *V-subtrees*. The naming convention for U and V is derived from the bipartite algorithm discussed in the previous sections. A *U-subtree* consists of the subtree rooted at a specific node (or the lower node of a given cut site), including all its children and the complete subtree structure beneath it. Conversely, the *V-subtree* represents the remainder of the tree after the *U-subtree* has been extracted.

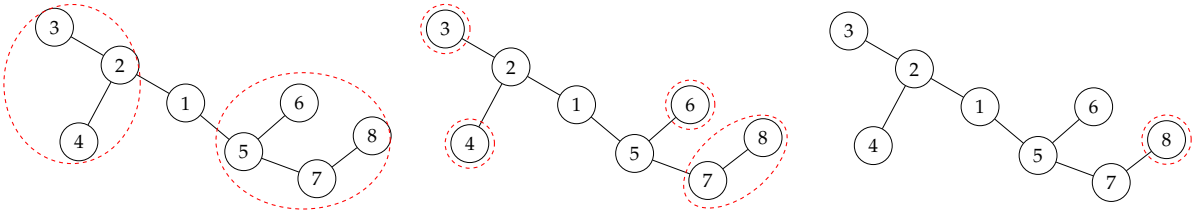

Figure 4.14.: The traverse of U-subtrees for the first, second and third level. The resulting red dotted subtrees are unique set of subtrees

Figure 4.13 illustrates this partitioning. In parts (b) and (c), the division into *U-subtrees* and *V-subtrees* is shown for the vertex sites 1-2 and 5-6, respectively. It is notable that the *U-subtrees* retain a tree structure including all children from the new root node, while the *V-subtrees* is not like that, reflecting the asymmetry introduced by the partitioning process.

As previously mentioned, we utilize a Breadth-First Search (BFS) strategy, starting at the root of the tree and traversing through its branches. A key feature of our approach is the dual traversal at each level: rather than processing each vertex site consecutively, the algorithm performs a complete pass through all vertex sites at any level before initiating a second pass. This traversal order is illustrated in Figure 4.13 a), providing a clear representation of the process.

One of the primary challenges in this approach is constructing non-redundant sets for each cut. In the linear case, this task involved identifying unique left and right chains to form the initial non-redundant sets of U and V nodes, which served as the algorithm's starting point. This process was relatively straightforward. However, in the tree structure, the focus shifts to comparing subtrees instead of chains, significantly increasing the computational complexity. This is the main motivation of our double-pass strategy.

During the first pass, all U-subtrees are processed, and redundant U-subtree sets are identified and formed. Since each U-subtree is independent and does not share any nodes with others, we can precalculate and update the compound state diagram accordingly. This approach eliminates the need to repeatedly recalculate the same tree parts, improving computational efficiency. The process of handling U-subtrees is illustrated in Figure 4.14 for each level. Let us examine how this contributes to the overall optimization.

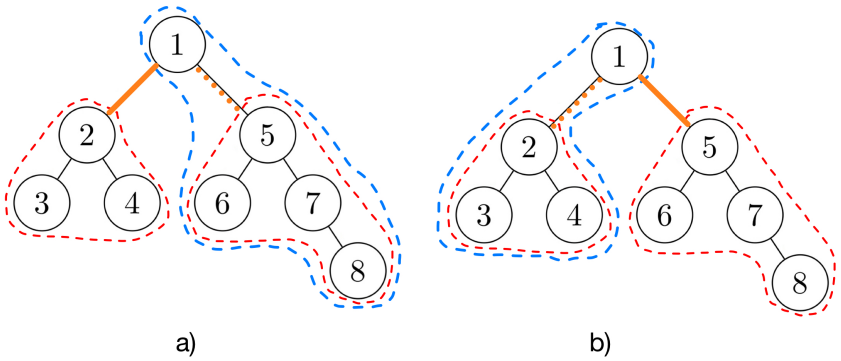

Figure 4.15.: Red dashed lines are U-subtrees that are already optimized, and Blue dashed lines correspond to the V-subtrees.

Following the first pass, we proceed with the second pass, where V-subtrees are calculated and cut optimizations are performed. The optimization process itself remains unchanged, following exactly the same steps discussed in the previous section. The key enhancement lies in the calculation of V-subtrees, where we leverage the precomputed U-subtrees.

Refer to Figure 4.15, which depicts two cuts at the first level. Notice that a V-subtree typically includes already computed U-subtrees, except for the cut site node. To identify the redundant set of V-subtrees, it suffices to compare the cut site node with the corresponding subtrees. This significantly reduces the computational burden, as much of the work was already performed during the first pass.

The same principle extends to consecutive levels. As shown in Figure 4.16, there are three cuts at the next level. Here, we observe that V-subtrees are composed not only of the precomputed U-subtrees but also of the precomputed V-subtrees from the previous iteration.

For the comparison process, it is unnecessary to compare all the nodes within the blue dashed region. Instead, we can once again leverage the precalculated data and limit the comparison to the cut site nodes and their immediate connections. This further reduces the computational overhead, making the algorithm more efficient as it progresses through the tree.

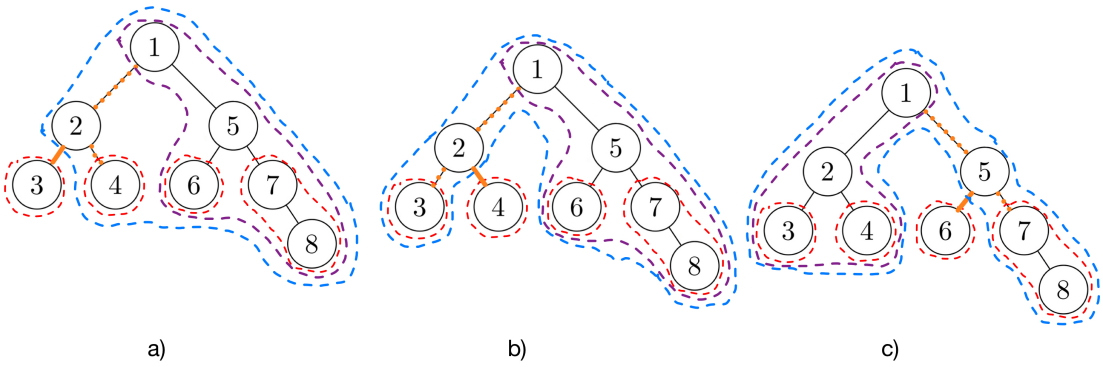

Figure 4.16.: Red dashed lines are U-subtrees that are already optimized. Purple dashed lines are previously optimized V-subtrees. Blue dashed lines represent V-subtrees that are currently being optimised. To detect identical V-subtrees, it is enough to check *cut_site* and orange dashed connections with U-subtrees and optimized V-subtrees. **a)** Cut through sites 2-3. **b)** Cut through sites 2-4. **c)** Cut through sites 5-6.

One could argue that even with the precomputation of subtrees, we still need to meticulously compare all the nodes of each subtree at least once, which demands significant computational resources. This observation leads us to the next clever implementation of our algorithm for subtree comparisons: **hashing**.

4.2.4. Efficiency in Comparison - Hash Values

To perform comparisons within the tree structure more efficiently, we employ hashing. Each node is assigned a unique hash value, computed based on its internal content and the hash values of its child nodes. These hash values are calculated once during the tree's construction, using a bottom-up approach.

With this method, comparing two subtrees is reduced to a straightforward comparison of their precomputed hash values. This significantly lowers the computational cost, as it eliminates the need to examine potentially hundreds or thousands of nodes for each comparison attempt. By utilizing hashes, the algorithm becomes far more efficient, enabling rapid subtree comparisons even in large and complex tree structures.

These hash values cannot be directly applied to V-subtrees, as these structures do not form complete subtrees encompassing all children nodes. However, V-subtrees can be uniquely identified through the cut-site node and its connections to other subtrees. As

demonstrated in Figures 4.15 and 4.16, each V-subtree is composed of either U-subtrees or previously calculated V-subtrees.

To address this, the comparison of V-subtrees is simplified by employing a specialized hash function. This function integrates the internal information of the cut-site node along with the hash values of its connected subtrees. By using this approach, we ensure that V-subtrees can be efficiently and uniquely identified without needing to examine their entire structure, maintaining computational efficiency.

The previously described double-traversed BFS enables efficient computation of U-subtrees and V-subtrees for earlier depths before the cut, allowing the pre-computed information to be reused effectively. As mentioned earlier, the order of processing does not affect the optimization result but impacts the computational complexity as explained here.

4.2.5. Algorithm Summary and Analysis

The TTNO optimization algorithm is a comprehensive approach designed to minimize virtual bond dimensions while maintaining computational efficiency. It integrates symbolic Gaussian elimination, bipartite graph theory, and subtree hashing to achieve its objectives. The details of the overall TTNO optimization process are outlined in "Algorithm 3", and the specific steps for cut optimization are presented in "Algorithm 4".

One of the notable features of the algorithm is its two-pass traversal strategy within each tree level, as illustrated in Algorithm 3. During the first pass, U-subtrees are processed to form non-redundant sets using precomputed hash values, which significantly reduces computational redundancy. The second pass focuses on V-subtrees, leveraging the precomputed U-subtrees to simplify the identification of redundant V-subtrees. This dual-pass strategy ensures that the algorithm efficiently traverses the tree structure while maintaining optimal computational performance.

A critical component of the algorithm is the cut optimization process, detailed in Algorithm 4. For each cut site, a Γ-matrix is constructed based on the initial connectivity. Symbolic Gaussian elimination is applied to simplify the matrix while adhering to symbolic constraints. The bipartite graph algorithm is then employed to identify the optimal configuration, comparing the results of the initial and updated Γ-matrices. This step determines whether to proceed with the updated configuration or revert to the initial one. Finally, the cut site is reconnected, and coefficients are carefully assigned to preserve the system's integrity.

This algorithm offers several key advantages. By leveraging symbolic Gaussian elimination, it preserves the symbolic representation of γ-coefficients, avoiding numerical dependencies and ensuring stability. Additionally, the use of hashing for subtree comparisons drastically reduces computational overhead, particularly for large and complex tree structures. Furthermore, the algorithm's adaptability allows it to handle both uniform and non-uniform coefficient cases effectively, making it suitable for a wide range of physical models.

In summary, the TTNO optimization algorithm provides a robust framework for optimizing hierarchical tensor networks. By combining efficient traversal strategies, symbolic computations, and advanced graph-theoretical techniques, it achieves both accuracy and efficiency, making it a valuable tool for quantum many-body simulations and related applications.

Algorithm 3: TTNO Optimal Constrcution Algorithm

Input: Tree structure T, Hamiltonian H
Output: Optimized TTNO with minimized virtual bond dimensions
Initialization: Represent T as a hierarchical tree structure and initialize compound state diagram S using given Hamiltonian; Precompute hash values for all nodes in T using a bottom-up approach;
Breadth-First Traversal:;
foreach *level l in T (processed in BFS order)* **do**
    First Pass: Process U-Subtrees;
    foreach *U-subtree at level l* **do**
        Identify redundant U-subtrees using precomputed hash values;
        Form non-redundant sets of U-subtrees;
        Update compound state diagram S with optimized U-subtree information;
    end
    Second Pass: Process V-Subtrees and Perform Cut Optimization;
    foreach *V-subtree at level l* **do**
        Identify V-subtrees using precomputed U- and V-subtrees;
        Use specialized hash function to identify redundant V-subtrees;
        Call Cut Optimization Algorithm: Optimize the cut site using the **Cut Optimization Algorithm**;
        Update compound state diagram S and the cut site with the results of the optimization.;
    end
end
return Optimized TTNO compound state diagram S;

Algorithm 4: Cut Optimization Algorithm

Input: Compound state diagram S with a focus on cut site C
Output: Updated S, with optimized connectivity for cut site C

Step 1: Create Initial Γ-Matrix;
Construct the Γ-matrix using the initial connectivity of the cut site C;

Step 2: Apply Symbolic Gaussian Elimination;
Update Γ-matrix by performing symbolic Gaussian elimination;
Preserve symbolic constraints during the elimination process;

Step 3: Apply Bipartite Graph Algorithm;
Run the bipartite graph algorithm on the initial Γ-matrix to determine minimal virtual bond dimensions;
Run the bipartite graph algorithm on the updated Γ-matrix (from symbolic Gaussian elimination);

Step 4: Compare Results;
if *Updated Γ-matrix improves virtual bond dimensions* **then**
    Use the updated Γ-matrix for further processing;
end
else
    Revert to the initial Γ-matrix;
end

Step 5: Reconnect Cut Site;
foreach *node in cut site C* **do**
    Reconnect nodes based on the result of the bipartite graph algorithm;
    Create new nodes if required;
    Assign γ-coefficients carefully to ensure correctness;
    Push coefficients to left or right subtrees as needed;
end

return Optimized connectivity for cut site C;

5. Physical Model

As previously outlined, this research centers on constructing representations of physical systems for simulation. Up to this point, we have discussed the theoretical foundations and the algorithmic framework. The remaining critical step involves conducting experiments on real-world systems to evaluate the implementation's effectiveness. Before presenting the experiments and results, it is important to provide an overview of the target systems. Understanding these systems is vital for developing accurate and efficient representations or analyzing the results better.

The study begins with the ab initio electronic Hamiltonian, a relatively straightforward example encompassing both short- and long-range interactions. This choice is motivated by its inclusion in one of the reference papers underpinning this research. Subsequently, our attention shifts to a more general and fundamental Hamiltonian representing a two-dimensional lattice model. Additionally, experiments are conducted on randomly generated Hamiltonians expressed in sum-of-products form, utilizing a limited set of local operators. The details of these systems are explored in the following sections.

5.1. Ab Initio Electronic Hamiltonian

In the field of quantum chemistry, the *ab initio* electronic Hamiltonian is a fundamental framework used to describe the quantum mechanical interactions between electrons within molecules. This Hamiltonian is constructed entirely from the first principle of quantum mechanics without relying on empirical parameters. The mathematical representation of the electronic Hamiltonian, $\hat{H}$, is given as follows:

$$\hat{H} = \sum_{pq} t_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} v_{pqrs} a_p^\dagger a_q^\dagger a_r a_s \tag{5.1}$$

To fully understand the equation, it is essential to first become familiar with the fundamentals of fermionic algebra. This mathematical framework governs the behavior and properties of fermions, the particles that obey the Pauli exclusion principle. Fermionic algebra is characterized by anticommutation relations, which are central to describing quantum systems involving electrons:

$$a = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad a^\dagger = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

The operators a_p and $a_p^\dagger$ are fundamental components of fermionic algebra and play a crucial role in the mathematical description of quantum systems involving fermions, such as electrons. These operators are used to modify the occupation number of a given quantum state, such as adding or removing particles from that state. Their properties are defined as follows:

- a_p: The **annihilation operator**, which removes a fermion (electron) from the quantum state p. When applied to a quantum state, a_p reduces the occupation number of state p by one, provided that state p is already occupied. This can be written as:
$$a_p|n_p\rangle = |n_p - 1\rangle$$
If the state is unoccupied, the operation yields zero, reflecting the Pauli exclusion principle:
$$a_p|0_p\rangle = 0$$
This reflects the fact that you cannot remove a particle from an empty state.

- $a_p^\dagger$: The **creation operator**, which adds a fermion (electron) to the quantum state p. When applied to a quantum state, $a_p^\dagger$ increases the occupation number of state p by one, provided that state p is unoccupied. Mathematically, it can be expressed as:
$$a_p^\dagger|n_p\rangle = |n_p + 1\rangle$$
This operator is called the creation operator because it 'creates' a particle in the state p.

The operators obey the following **anticommutation relations**, which are fundamental to the behavior of fermionic systems:

$$\{a_p, a_q^\dagger\} = a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}$$

$$\{a_p, a_q\} = a_p a_q + a_q a_p = 0$$

$$\{a_p^\dagger, a_q^\dagger\} = a_p^\dagger a_q^\dagger + a_q^\dagger a_p^\dagger = 0$$

These relations ensure that no two fermions can occupy the same quantum state simultaneously, encapsulating the Pauli exclusion principle. Additionally, the operators a_p and $a_p^\dagger$ allow for the construction and manipulation of many-body quantum states, making them indispensable tools in the study of quantum mechanics and quantum

chemistry. For the purposes of this thesis, these operators are used to construct the electronic Hamiltonian, which describes interactions among electrons in a given system.

Mastering all aspects of fermionic algebra is beyond the scope of this thesis. However, a foundational understanding is necessary to interpret the provided Hamiltonian. To this end, we can simplify the ab inito electronic Hamiltonian by explaining the two types of interactions it encompasses:

$$\sum_{pq} t_{pq} a_p^\dagger a_q$$

This term represents the one-body interactions within the system. It sums over all pairs of quantum states p and q. The coefficients t_{pq}, referred to as one-electron integrals, typically account for the kinetic energy of electrons moving in the field of the nuclei as well as the potential energy arising from the attraction between the electrons and the fixed nuclei. This term captures the behavior and contributions of single electrons within the molecular system.

$$\frac{1}{2} \sum_{pqrs} v_{pqrs} a_p^\dagger a_q^\dagger a_r a_s$$

This term describes the two-body interactions, specifically the repulsion between pairs of electrons. The summation extends over all combinations of quantum states p, q, r, and s. The coefficients v_{pqrs}, known as two-electron integrals, quantify the repulsion energy between electrons in states p and q. The factor $\frac{1}{2}$ accounts for the double counting of electron pair interactions, as the repulsion between two specific electrons is counted once for states p and q and again for states r and s. The sequence of creation and annihilation operators, $a_p^\dagger a_q^\dagger a_r a_s$, adheres to the fermionic anticommutation rules, ensuring compliance with the Pauli exclusion principle. This structure correctly captures the antisymmetric nature of the wave function under the exchange of any two electrons.

5.2. Effective Lattice Hamiltonian with Local and Non-Local Terms

Even though the previous Hamiltonian captures both short- and long-range interactions within a quantum system, an additional feature we aim to explore is the scenario where

coefficients are shared across different terms. To simplify the system under consideration while retaining both local and long-range relations, we adopt the following Hamiltonian. In this representation, the strength of the interaction decays with distance:

$$H = \sum_{i,j} \frac{J}{\|i-j\|} X_i X_j + \sum_i g Z_i$$

In this case, the Hamiltonian involves basic Pauli matrices, specifically *Pauli-X* and *Pauli-Z*, operating on sites i and j. The *Pauli-X* operator (X_i) corresponds to a spin flip on site i, while the *Pauli-Z* operator (Z_i) measures the spin along the z-axis. Using Pauli matrices offers several advantages: they form a complete basis for single-qubit operations, allowing efficient representation and manipulation of quantum states. Additionally, their simple algebraic properties make them particularly suitable for testing and validating the interaction strengths and decay patterns within quantum systems.

For this system, we can again explain the two types of interactions to understand system better:

$$\sum_{i,j} \frac{J}{\|i-j\|} X_i X_j$$

This term represents the *long-range interaction* between pairs of elements (e.g., qubits, spins, or particles) at positions i and j. The interaction strength is governed by the coefficient J, which is scaled by the inverse of the distance between i and j, denoted as $\|i-j\|$. Such scaling reflects the physical principle that interactions between elements typically weaken as their separation increases. The operators X_i and X_j are spin-flip operators in spin systems or qubit state rotations in quantum computing. This term is essential for capturing *correlation effects* in a quantum many-body system, as it introduces coupling between distant elements, making the system non-local. The inverse-distance dependency $\frac{1}{\|i-j\|}$ is common in systems where interactions decay with separation, such as Coulomb interactions in physics or certain graph-based problems in quantum optimization.

$$\sum_i g Z_i$$

The second term describes a *local field effect* acting on each element i independently. The coefficient g represents the strength of this local field, and Z_i is Pauli Z-operator. This term can be thought of as a local "bias" or "energy shift" that influences individual elements based on their state. For instance, in a spin-1/2 system, this term would differentiate between the energy of spin-up and spin-down states. In the context of

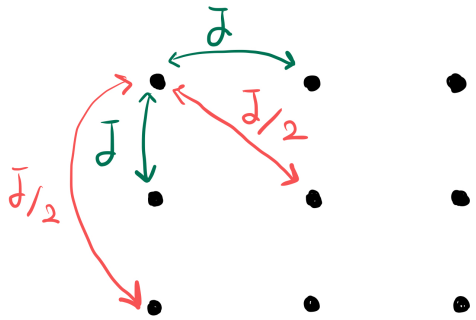

Figure 5.1.: Above, we present a grid consisting of 9 sites, illustrating only the long-range interactions alongside their corresponding coefficients. Using the Manhattan distance, we observe that the distance between node $(0, 0)$ and node $(1, 1)$ is the same as the distance between node $(0, 0)$ and node $(0, 2)$. This symmetry arises from the definition of the Manhattan distance, which sums the absolute differences in the coordinates of two nodes. As a result, interactions at equal Manhattan distances will have identical coefficients in the Hamiltonian.

quantum annealing or optimization problems, this term can encode *external constraints or penalties* that guide the system toward specific solutions. The local field term is crucial for breaking symmetry in the system and introducing tunable control over individual elements.

Together, these two terms form a versatile Hamiltonian that combines long-range correlations (via the first term) with local control (via the second term). This type of Hamiltonian is particularly useful in quantum simulations, quantum annealing, and optimization problems, where both global interactions and local effects must be balanced to explore or solve complex systems efficiently.

For the distance between two sites i and j, $\|i - j\|$, we decided to use the Manhattan distance instead of the Euclidean distance. The Manhattan distance, also known as the L_1 norm, is defined as the sum of the absolute differences of their coordinates. Mathematically, for two points $i = (i_x, i_y)$ and $j = (j_x, j_y)$ on a 2D grid, the Manhattan distance is given by:

$$\|i - j\| = |i_x - j_x| + |i_y - j_y|$$

This distance metric measures the total "grid-based" distance. The choice of Manhattan distance in our implementation stems from its computational and representational advantages. If the Euclidean distance (or L_2 norm) were used instead, the coefficient term $\frac{J}{\|i-j\|}$ would often result in irrational numbers. In contrast, the Manhattan distance ensures that the coefficient remains a rational number, aligning with the constraints of our implementation, which is designed to operate with rational coefficients. This choice simplifies computations and maintains consistency across the Hamiltonian terms while retaining meaningful distance-based interactions.

5.3. Randomly Generated Hamiltonians in Sum-of-Products Form

In addition to the structured Hamiltonians discussed previously, we also explore randomly generated Hamiltonians expressed in the form of a sum of products of local operators. This approach provides a flexible and versatile framework for testing our implementation across a diverse range of scenarios, as the Hamiltonian parameters can be freely varied to simulate different system configurations.

The general form of the randomly generated Hamiltonian is given as:

$$H = \sum_{k=1}^{N} \gamma_k \prod_{i=1}^{M} O_i^{(k)}$$

where N is the total number of terms in the Hamiltonian, and M represents the number of local operators in each term which is equal to the number of sites in our system. The coefficients γ_k are assigned to each term and can be configured in one of the following ways:

- All coefficients are distinct, ensuring each term has a unique weight.
- All coefficients are uniform, providing equal contribution from each term.
- Coefficients are selected from a discrete set of predefined values, allowing for partial uniformity.

For the local operators $O_i^{(k)}$ used in the product, we constrain the set of operators to the Identity matrix (I) and the Pauli matrices (X, Y, Z). These operators form a complete basis for single-qubit operations, enabling efficient representation and manipulation of quantum states. The constraint to these operators ensures computational tractability while preserving the generality of the random Hamiltonian's structure.

5.3.1. Key Features of the Randomly Generated Hamiltonian

The randomly generated Hamiltonian incorporates several important features that make it versatile and adaptable for testing:

1. **Term Randomization**: Each term in the Hamiltonian is generated randomly, allowing for an arbitrary number of terms (N) and random combinations of the local operators ($O_i^{(k)}$).

2. **Coefficient Variability**: By adjusting the configuration of the γ_k coefficients, we can simulate different physical scenarios, such as fully distinct interactions, uniform interaction strengths, or discrete interaction patterns.

3. **Operator Constraints**: The restriction to Identity and Pauli matrices simplifies the generation process while maintaining a wide range of possible interactions. These operators are well-suited for capturing the fundamental dynamics of many-body quantum systems.

5.3.2. Applications of Randomly Generated Hamiltonians

The random Hamiltonians are particularly useful for stress-testing the implementation and evaluating its robustness across different scenarios:

- **Validation of Algorithm Performance**: By varying the number of terms (N), number of the sites (M) and the complexity of the operator products ($O_i^{(k)}$), we can analyze the scalability and efficiency of our construction methods.

- **Simulation of Generic Systems**: Random Hamiltonians simulate generic quantum systems without relying on specific physical models, enabling broader applicability.

- **Exploration of Edge Cases**: The randomization allows us to generate rare or extreme configurations, which are valuable for identifying potential limitations in the algorithm.

This approach ensures a comprehensive evaluation of the methods developed in this thesis, providing insights into their applicability to real-world systems and their general performance across diverse Hamiltonian structures.

6. Experiments, Results and Evaluations

This chapter presents the experiments conducted to evaluate the proposed algorithms and discusses the results obtained. The experiments are designed to test the algorithm's performance and its applicability to various quantum systems. Additionally, for each system, we test different features, such as the impact of varying tree structures and the effect of diverse coefficient assignments, including uniform, distinct, and discrete values.

To assess the performance of our proposed method, we compare the results for each experimental set with optimal results calculated using the mathematically accurate Singular Value Decomposition (SVD), with the outcomes of previously implemented algorithm which is the Bipartite Graph Algorithm and with non-optimized trivial construction. These comparisons provide a comprehensive evaluation of our method's accuracy and efficiency relative to existing techniques and ground state.

The use of Singular Value Decomposition (SVD) as a baseline method for determining the optimal virtual bond dimensions offers a mathematically rigorous reference. However, this approach comes with significant computational limitations. The computational cost of SVD for dense matrices is $O(n^3)$, where n is the dimension of the matrix. This cost scales exponentially with the size of the system, making SVD infeasible for larger quantum systems, particularly those with more than 7–8 sites in our set-up.

For the experiments, wherever SVD is used to establish baseline reference values, we will focus on smaller systems, typically limited to those with fewer than 7–8 sites. This enables a direct comparison between the optimal results produced by SVD and the outputs of the proposed algorithms. For larger systems, where SVD computations are impractical, the analysis will rely on comparisons among the algorithms themselves, excluding SVD from consideration. By adhering to these constraints, we balance computational feasibility with the need for a reliable baseline in small-scale scenarios.

The chapter is organized as follows: We provide an analysis of the three main Hamiltonian structures considered in this study: the *ab initio* electronic Hamiltonian, the effective lattice Hamiltonian, and randomly generated Hamiltonians. For each system, we evaluate specific aspects, such as the suitability of different tree structures

or the role of coefficient variability. Finally, we compare the performance of our method to existing approaches and discuss the implications of the results within each section.

6.1. Experimental Setup

The experiments were conducted in a controlled computational environment using an Arm M1 processor with 16 GB RAM. The algorithms were implemented in vanilla Python without extra numerical libraries even like NumPy. For symbolic computations, we again stick with pure Python implementation, refraining from usage of the SymPy library.

Key performance metrics considered in the experiments include:

- **Virtual Bond Dimension**: A measure of the efficiency of the constructed MPOs and TTNOs.
- **Accuracy**: The ability to reproduce the original Hamiltonian after reconstruction. (Which we always reach with any method)

The results for each Hamiltonian configuration are discussed in the subsequent sections.

6.2. *Ab Initio* Electronic Hamiltonian

Although considerable effort was invested in understanding the *ab initio* electronic Hamiltonian during the initial phase of this thesis, it was ultimately not the focus of the experimental phase. The decision was based on the observation that testing with randomly generated Hamiltonians effectively encompasses all scenarios that could occur with the *ab initio* electronic Hamiltonian. Additionally, the lattice Hamiltonian was determined to be a more suitable system for evaluating coefficient variability and different tree constructions. As a result, experiments with the *ab initio* electronic Hamiltonian have been deferred to future work.

6.3. Randomly Generated Hamiltonians

The initial set of experiments is randomly generated Hamiltonians, providing a general evaluation of the algorithm's performance. As explained in the previous section, these

Hamiltonians were constructed with varying numbers of terms, different sets of coefficients, and combinations of local operators (I, X, Y, Z).

These Hamiltonians are composed of terms varying in number from 1 to 30. For each level (i.e., each number of terms), 1,000 distinct Hamiltonians were tested, resulting in a total of 30,000 Hamiltonians for one set of coefficients. This broad dataset ensures a comprehensive evaluation of the optimization methods across a wide range of scenarios.

For this set of experiments, we utilize a fixed tree structure as depicted in Figure 4.12. The tree structure is kept constant throughout all tests to isolate the effects of the optimization methods on bond dimensions. By standardizing the tree structure, we ensure that any variations in performance are attributable to the optimization methods rather than differences in tree topology.

The optimization methods under consideration are as follows:

- **SVD (Singular Value Decomposition):** This method provides the reference solution with the optimal bond dimensions. Although computationally expensive, SVD serves as a benchmark for comparing the other approaches.
- **Basic:** The naive construction of the TTNO without any form of compression. This represents the worst-case scenario with maximal bond dimensions.
- **Bipartite Theory:** This approach applies the bipartite graph optimization technique (using the Hopcroft-Karp algorithm) to minimize the bond dimensions, without any additional preprocessing.
- **SGE + Bipartite:** This enhanced method combines Symbolic Gaussian Elimination (SGE) with the bipartite graph theory to preprocess and optimize the construction further.

Each optimization method was tested with three distinct sets of coefficients, as described in the previous section:

- **Uniform Coefficients:** All terms in the Hamiltonian share the same coefficient value.
- **Partially Uniform Coefficients:** A mix of terms with identical and distinct coefficients.
- **Distinct Coefficients:** Each term in the Hamiltonian has a unique coefficient.

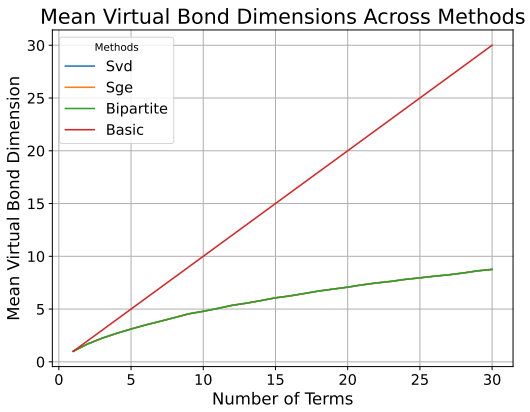

Figure 6.1.: Mean bond dimensions for different methods, with basic configuration

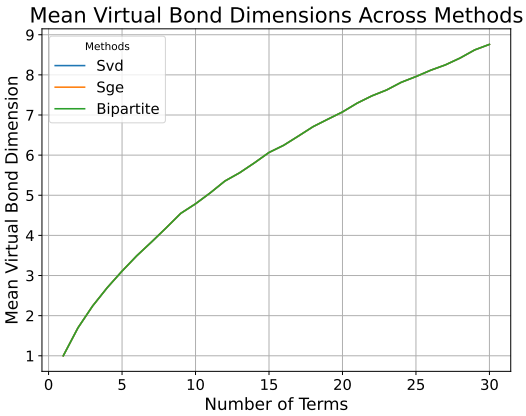

Figure 6.2.: Mean bond dimensions for different methods without basic configuration.

These coefficient configurations allow us to test how the optimization methods perform under varying levels of complexity in the Hamiltonian. For instance, the uniform coefficients case is expected to provide simpler connectivity structures, while the distinct coefficients case introduces higher complexity due to the lack of redundancy.

The results obtained align well with our theoretical expectations, highlighting the strengths and weaknesses of each method under different scenarios. The following sections present a detailed analysis of the results, accompanied by visualizations of the data. These include comparisons of the bond dimensions achieved by each method and an exploration of the effects of varying coefficient configurations on optimization performance.

6.3.1. Uniform Coefficients:

This scenario underscores the inherent limitation of the Bipartite Graph Theory, where the addition of Symbolic Gaussian Elimination (SGE) was expected to yield significant improvements. As illustrated in Figures 6.1 and 6.2, the basic system—with the naïve construction approach—shows substantially higher mean bond dimensions compared to the other methods. This strong difference highlights the efficiency of the optimisations. Even when the basic system is excluded, as shown in Figure 6.2, the mean bond dimensions for the remaining methods appear closely aligned, reflecting the general effectiveness of the Bipartite Graph approach in most cases. However, this overall success also masks subtle yet critical differences between the methods.

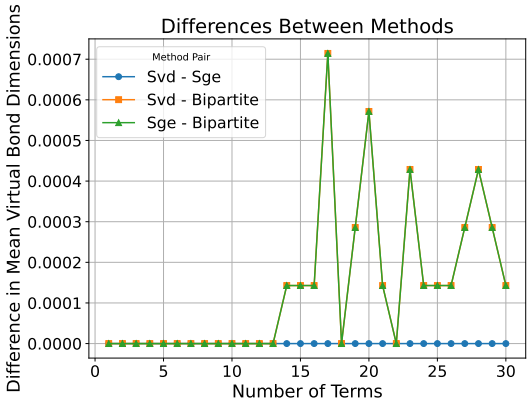

Figure 6.3.: Differences between Bipartite and SVD-SGE

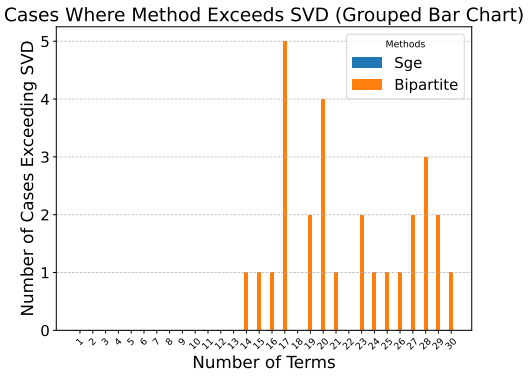

Figure 6.4.: Number of cases where bond dimension is suboptimal compare to SVD

A more detailed examination, presented in Figures 6.3 and 6.4, reveals a clear trend: the deviation from the SVD solution increases with the use of the Bipartite Graph Theory alone. This trend indicates the method's inability to fully optimize bond dimensions in certain cases, particularly as the system complexity grows. By contrast, the combination of SGE with Bipartite Graph Theory consistently achieves the optimal bond dimensions, matching the SVD results in all tested scenarios. This success underscores the strength of the SGE preprocessing step in addressing the shortcomings of the standalone Bipartite Graph approach, particularly in handling complex uniform coefficient cases.

Despite these findings, it would be unjust to disregard the contributions of the Bipartite Graph approach entirely. As shown in Figure 6.5, the naïve construction performs significantly worse than the Bipartite Graph method, even when viewed on a logarithmic scale. The disparity in performance between these methods underscores the considerable advancements provided by the Bipartite Graph approach over basic construction methods. However, the SGE-enhanced algorithm stands out as the most robust and reliable, delivering consistently optimal results across all cases. This makes it a pivotal step forward in the optimization of bond dimensions for tensor network operators.

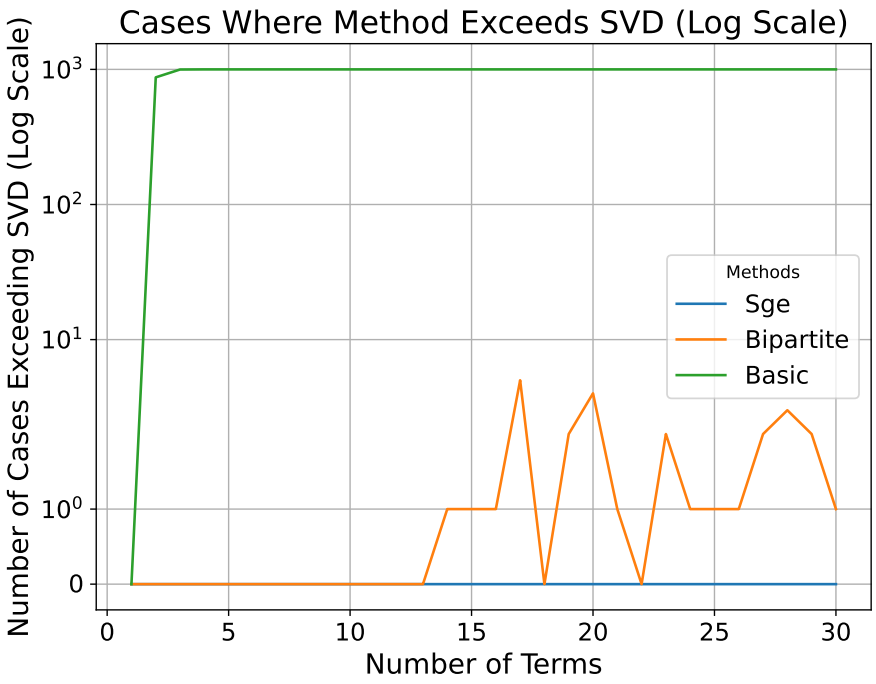

Figure 6.5.: Deviation from SVD for all methods.

6.3.2. Partially Uniform and Distinct Coefficients:

For a unique set of coefficients, the theoretical proof of the Bipartite Graph Theory provided by Ren et al. [15] holds true, demonstrating that the Bipartite Graph method is, in theory, capable of achieving the optimal bond dimension. This expectation is satisfied with our experimental results, where both the Bipartite Graph approach and our approach SGE consistently reach the optimal configuration for all tested cases. Figure 6.6 illustrates this equivalence, showing that the bond distributions for these methods are identical across the experiments, regardless of the number of terms in the Hamiltonian.

These results are also valid for both partially uniform and completely distinct coefficient sets. While partially uniform coefficients might theoretically introduce inefficiencies in the Bipartite Graph approach—particularly in cases where the overlap between coefficients leads to inefficiency in the optimization process—our random experiment set revealed no such issues. A likely explanation is that the number of unique coefficient terms in the partially uniform set remained sufficiently high to avoid problematic configurations. Additionally, the randomness of our experiment set may

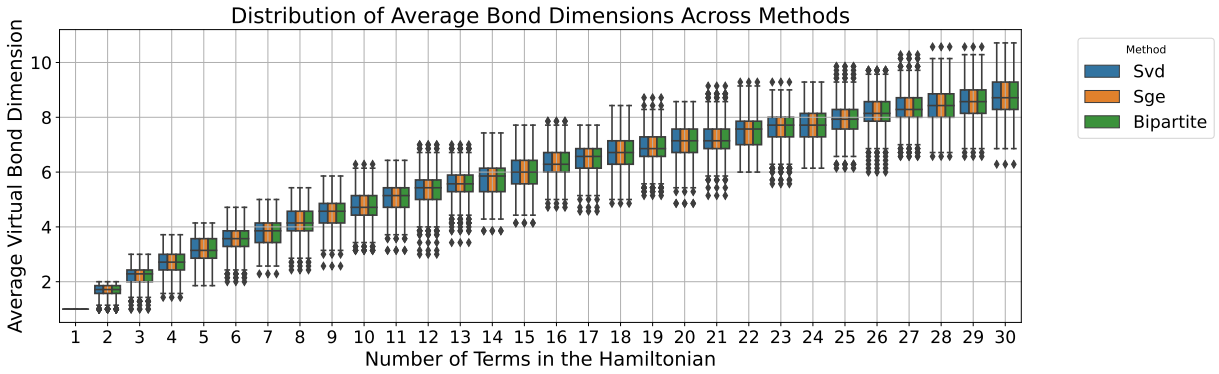

Figure 6.6.: Virtual Bond distribution for each method

have naturally excluded edge cases where partially uniform coefficients would cause inefficiencies.

It is worth noting that while these results suggest broad applicability of the Bipartite Graph method, they do not entirely rule out the possibility of inefficiencies in edge cases with more carefully curated partially uniform coefficients. Future research could explore this possibility by deliberately constructing test cases designed to challenge the Bipartite Graph method under partially uniform coefficient conditions. Nevertheless, for the configurations tested in this study, both the Bipartite Graph and SGE methods performed as expected, consistently yielding optimal bond dimensions.

6.4. Effective Lattice Hamiltonian

The second set of experiments focused on the effective lattice Hamiltonian, which features both local and long-range interactions. The algorithm's capability to manage decaying interaction strengths was thoroughly tested using a two-dimensional lattice model. Particular attention was given to the influence of different tree structures on bond dimensions. To this end, various methods for traversing the lattice grid were employed, enabling a comprehensive analysis of how tree configurations affect the optimization process and the resulting bond dimensions.

The traversal methods explored in the experiments are as follows and can be shown in the figure 6.7:

- **Snake**: This method performs a linear traversal, iterating over each node in a snake-like pattern, ultimately resulting in a Matrix Product State (MPS) structure.

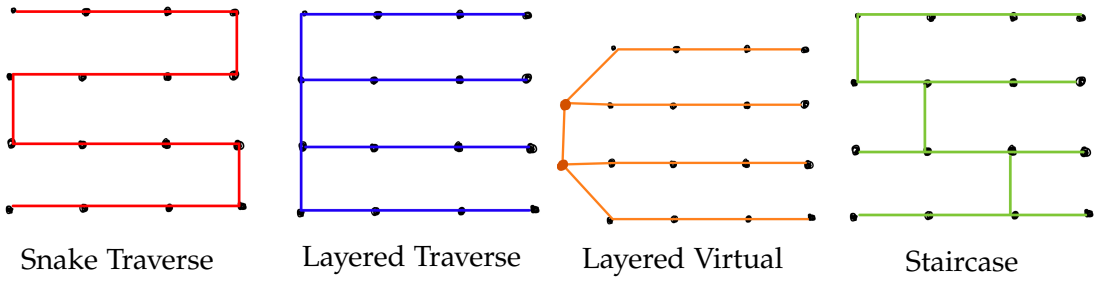

Figure 6.7.: Traverse Methods

- **Layered**: A tree structure where the first node of each line is connected directly to the other first nodes, creating a layered hierarchy across the grid.
- **Layered with Virtual Nodes**: A variation of the layered tree structure, where the connections between the first nodes are established through additional virtual nodes, providing an intermediary layer.
- **Staircase**: A more balanced tree structure that traverses the grid along the diagonal, creating a hierarchical tree configuration stemming from the diagonal connections.

6.4.1. Experiment Results

We performed experiments on the lattice Hamiltonian by progressively increasing the lattice size L in an $L \times L$ grid, carefully observing the virtual bond dimensions generated across different tree structures. This experimental design allowed us to evaluate how various tree structures impact the efficiency of TTNO representations.

For these tests, we employed Symbolic Gaussian Elimination (SGE) to construct the Hamiltonian. The choice of SGE was informed by its demonstrated superiority in prior experiments with randomly generated Hamiltonians, where it consistently achieved optimal bond dimensions more effectively than other methods. As with examining the lattice Hamiltonian, we observe that it involves only two symbolic coefficients, accompanied by varying scalar multipliers for interaction terms, which is a scenario where the bipartite graph algorithm may exhibit inefficiencies. By leveraging SGE, we ensured a precise and efficient initialization of the lattice Hamiltonian, isolating the performance of tree structures from any variability in the construction process.

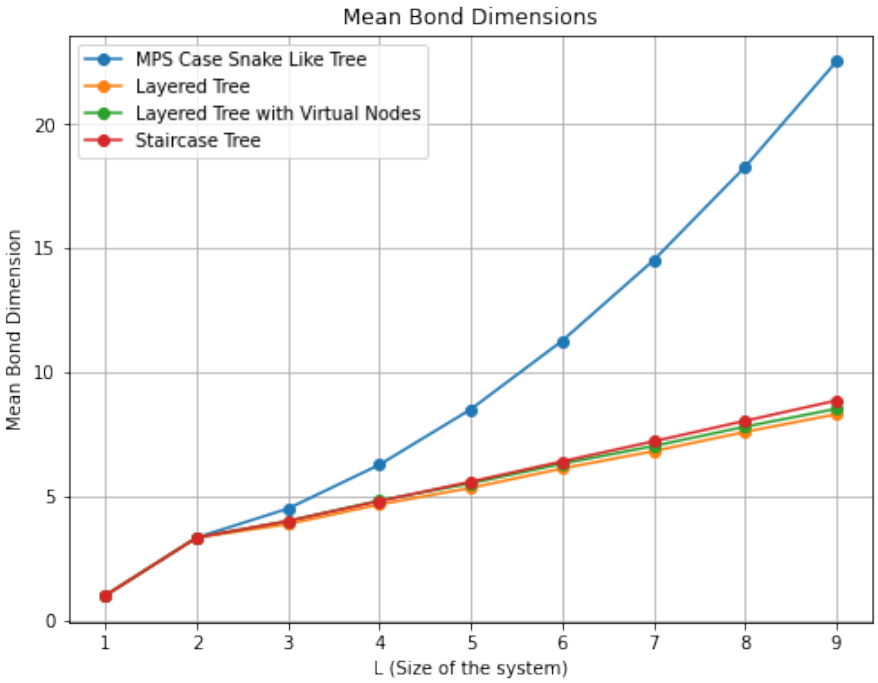

Figure 6.8.: Mean Virtual bond Dimensionns

In our experiments, we focus on analyzing two critical metrics: the **mean virtual bond dimension** and the **maximum virtual bond dimension**. The mean virtual bond dimension serves as an indicator of the overall success of the optimization process. A lower mean value reflects better compression and efficiency in the system's representation. On the other hand, the maximum virtual bond dimension highlights the *bottleneck* in the representation. Even a single site with an excessively large bond dimension can render the representation computationally infeasible.

It is important to note that all tree construction considered in our experiments are **accurate**, meaning they represent the system without any errors or approximations. Thus, the differences in performance metrics are solely attributed to the optimization quality of the tree structures, rather than any compromise in representational fidelity.

Mean Virtual Bond Dimensions

The graph in Figure 6.8 illustrates the mean virtual bond dimensions as a function of the lattice size L for various tree structures. The four tree structures—MPS Case Snake-Like Tree, Layered Tree, Layered Tree with Virtual Nodes, and Staircase Tree—are

compared to evaluate their efficiency in representing the lattice Hamiltonian.

The Snake-Like MPS Tree shows a rapid increase in the mean bond dimensions with increasing L. This behavior is expected because the linear structure of the MPS case inherently lacks the ability to represent long-range interactions efficiently. As L grows, the linear topology forces the bond dimensions to expand significantly to capture correlations, leading to higher computational costs.

Other tree methods demonstrate significantly better scalability compared to the Snake-Like Tree. By incorporating a layered structure, these topologies effectively balance local and global correlations, leading to a much slower growth in bond dimensions as the system size increases. This improvement underscores the advantages of tree structures, particularly for systems with long-range interactions, where linear representations like the Snake-Like Tree become inefficient.While there are slight differences among the tree structures tested, the Layered Tree slightly achieves the best bond dimensions overall.

Maximum Virtual Bond Dimensions

The graph in Figure 6.9 illustrates the maximum virtual bond dimension for each tree structure across varying grid sizes. This metric highlights the bottleneck in the tensor network representation, as the maximum bond dimension directly impacts the computational complexity and resource requirements for simulations.

As observed, the Snake-Like MPS Tree consistently exhibits a higher maximum bond dimension, indicating a significant bottleneck in its representation. This result aligns with expectations, as the Snake-Like structure is inherently limited in its ability to distribute correlations efficiently across the system, resulting in larger peak bond dimensions.

What is less expected, however, is the performance of the Staircase Tree, which shows bottlenecks comparable to the MPS Tree in terms of maximum bond dimension. This suggests that the Staircase Tree may not effectively mitigate correlation bottlenecks in smaller grids, potentially due to its less balanced hierarchical structure. It is possible that this performance issue may only manifest in smaller grids, as larger grids were beyond the computational limits of this study.

These findings highlight the importance of carefully considering tree structure when

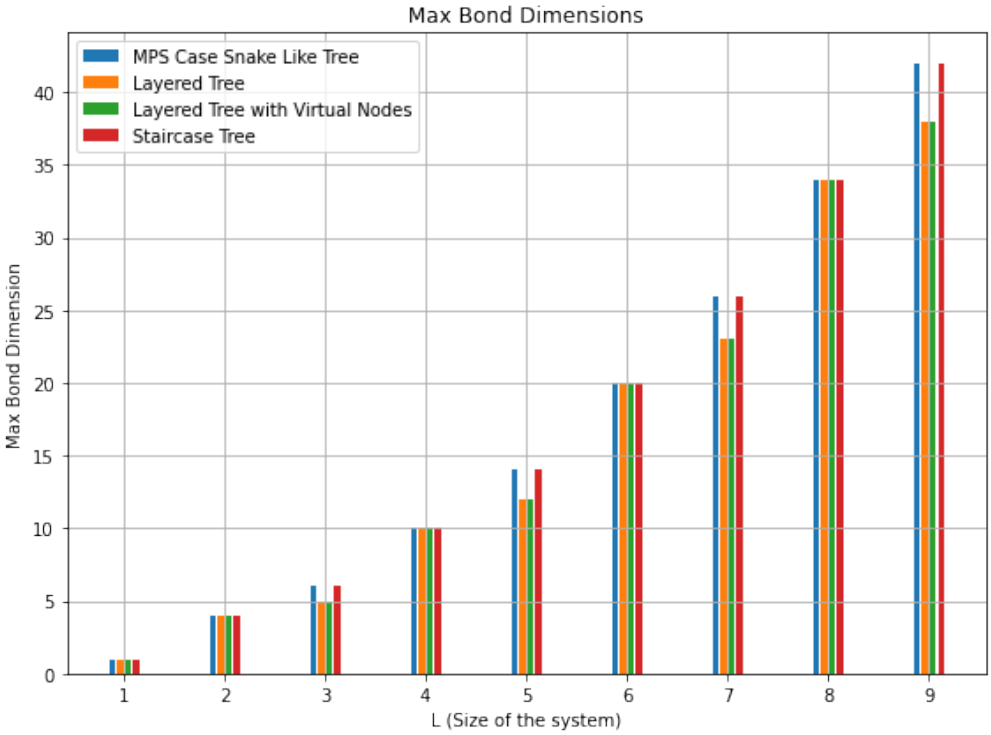

Figure 6.9.: Maximum Virtual Bond Dimension exist in the grid

designing tensor network representations, as suboptimal configurations can lead to significant bottlenecks that hinder scalability and computational efficiency.

6.5. Discussion

The experimental results presented in this chapter highlight the effectiveness of the proposed methods for optimizing bond dimensions in tensor network operators. By systematically testing various optimization methods across different Hamiltonian structures and coefficient configurations, we gained valuable insights into the strengths and limitations of each approach.

One of the most significant findings is the consistent superiority of the Symbolic Gaussian Elimination (SGE) combined with the Bipartite Graph Theory. This hybrid approach consistently achieved optimal bond dimensions across all tested cases, matching the results obtained using the mathematically rigorous but computationally expensive

SVD method. However, it is important to note that, due to its symbolic nature, the SGE + Bipartite approach can theoretically deviate from the optimal solution, particularly in cases involving unique or partially uniform coefficients. Despite this possibility, the experiments demonstrated that such deviations are rare, indicating that the method performs reliably across a broad range of configurations.

The Bipartite Graph Theory, while effective in many cases, demonstrated limitations when dealing with uniform or highly correlated coefficients. These scenarios resulted in suboptimal bond dimensions, as reflected in the increasing deviations from the SVD baseline. Nevertheless, the Bipartite Graph approach significantly outperformed the naïve construction method, showcasing its value as a practical and computationally efficient alternative for many applications.

The naïve construction method serves as a baseline to demonstrate the representation of the system without any form of compression or optimization on the TTNO. This approach highlights the worst-case scenario, underscoring the inefficiency and computational challenges posed by unoptimized representations. The powerfull contrast between the naïve construction and the results achieved using optimization methods such as Bipartite-SGE underscores the necessity of implementing more sophisticated techniques for tensor network operator construction.

The experiments also underscored the role of coefficient configurations in influencing the performance of the optimization methods. While all methods effectively handled unique and distinct coefficient sets, uniform configurations posed challenges, particularly for the Bipartite Graph Theory alone. These findings emphasize the need for robust algorithms that can adapt to diverse coefficient patterns and ensure optimal performance across a wide range of systems.

The results from the lattice experiments -on the other hand- provide additional insights into the performance of different tree structures and their implications for optimizing tensor network representations. One of the most striking observations is the sharp difference in scalability between the Snake-Like MPS Tree and other hierarchical structures. The MPS structure consistently produces higher mean bond dimensions, showcasing its inefficiency in capturing correlations for larger systems. This trend shows itself when considering the maximum bond dimensions. The inability of the Snake-Like MPS to distribute correlations effectively highlights the inherent limitations of linear structures for systems with long-range interactions.

These findings reinforce the importance of selecting appropriate tree structures for

tensor network representations, particularly when dealing with systems characterized by long-range interactions. While hierarchical structures generally outperform linear ones, the specific design of the tree can significantly impact scalability and efficiency.

In conclusion, the lattice experiments demonstrate that the choice of tree structure plays a pivotal role in optimizing tensor network representations. While the MPS - Tree may suffice for simpler systems, it is inadequate for capturing the complexities of systems with long-range interactions. Hierarchical structures, particularly the Layered Tree and its variants, offer a more robust framework for such systems, enabling efficient representation and reduced computational overhead. These findings complement the results from the random Hamiltonian experiments and provide a comprehensive understanding of the algorithm's behavior across diverse quantum systems.

7. Conclusions

This thesis explored an efficient symbolic construction of Matrix Product Operators (MPOs) and Tree Tensor Network Operators (TTNOs), focusing on leveraging bipartite graph theory and symbolic Gaussian elimination to address existing limitations. The research began by analyzing an established algorithm by Ren et al (2020) for MPO construction and identifying scenarios where the method underperformed. To overcome these challenges, a novel approach incorporating symbolic Gaussian elimination was developed, enhancing the bipartite graph algorithm's capabilities. This method was then extended to TTNOs, introducing additional complexity due to the hierarchical nature of tree structures.

The symbolic Gaussian elimination (SGE) preprocessing step proved to be a pivotal addition, enabling the algorithm to handle cases involving compressions more effectively. The algorithm avoided numerical instability by maintaining the symbolic representation of coefficients while preserving accuracy in most configurations. The extension to TTNOs, facilitated by state diagram representations, marked a significant step toward more generalized tensor network constructions.

Through extensive experiments, the following key findings were observed:

- The SGE-enhanced bipartite graph algorithm consistently achieved optimal or near-optimal bond dimensions, particularly in scenarios where the standalone bipartite graph algorithm struggled due to linear dependencies.
- The method demonstrated robustness across different Hamiltonian structures, including random configurations and lattice systems validating its versatility.
- While SVD provided the theoretical optimal solution, its computational limitations restricted its applicability to small systems. The SGE approach offered a practical alternative with competitive results.
- It has seen that TTNOs are significantly more efficient than linear MPO structures for systems with long-range interactions. Also, it revealed notable performance differences between various tree structures, highlighting the importance of selecting an optimal tree configuration for specific systems.

- The transition from MPOs to TTNOs, enabled by the hierarchical processing strategy and state diagram representation, extended the applicability of the method to non-linear systems while preserving its effectiveness.

7.1. Limitations and Challenges

Despite its successes, the proposed method is not without limitations:

- The symbolic approach, while effective in maintaining coefficient accuracy, does introduce additional computational overhead, particularly for larger systems. However, this overhead is deemed acceptable because the construction is performed only once at the start of the simulation. Given this, we prioritized accuracy and efficiency over computational cost, ensuring a reliable and optimized foundation for the subsequent simulation processes.
- The SGE-enhanced algorithm may deviate from the optimal solution in unique or partially uniform coefficient scenarios, although this was rare in the conducted experiments.
- The hierarchical extension to TTNOs, though successful, relies on predefined tree structures. Determining the most efficient tree configuration remains an open question, warranting further research.

7.2. Future Work

This research opens several avenues for future exploration:

- Extending the SGE-enhanced algorithm to other tensor network structures, such as Projected Entangled Pair States (PEPS) or higher-dimensional systems, could broaden its applicability.
- Developing more computationally efficient symbolic techniques or hybrid symbolic-numerical methods could address the overhead associated with the current approach.
- Exploring the integration of the method into practical quantum simulation frameworks to evaluate its performance on real-world quantum problems.

7.3. Concluding Remarks

Symbolic and efficient construction of tensor network operators, particularly MPOs and TTNOs, remains a critical challenge in quantum simulation. This thesis has made significant contributions to this field by introducing a novel algorithmic framework that addresses existing limitations and broadens the applicability of tensor network methods to hierarchical structures. By utilizing bipartite graph theory and symbolic techniques, this work establishes a robust and versatile approach to tensor network constructions, providing a solid foundation for future advancements in quantum computing and simulation.

A. Appendix

A.1. Symbolic Gaussian Elimination Implementation steps

To elaborate the implementation of the Symbolic Gaussian Elimination, we can check the given specific example, to observe how we preserve the operations while reducing the matrix to a 4x4 format and the rank to 4 without creating combinations of elements. In each step, the updates are highlighted with red, and we remove the zero rows-columns in the end as a final step:

$$
\rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3\gamma_{11} & 3\gamma_{12} & -\gamma_{13} & 0 & 2\gamma_{12} \\ 0 & 3\gamma_{22} & 0 & 0 & 2\gamma_{22} \\ 0 & 3\gamma_{32} & 0 & 0 & 2\gamma_{32} \\ 0 & 0 & 2\gamma_{43} & 2\gamma_{44} & 0 \\ 0 & 0 & -\gamma_{43} & -\gamma_{44} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 3\gamma_{11} & 3\gamma_{12} & -\gamma_{13} & 0 & 2\gamma_{12} \\ 0 & 3\gamma_{22} & 0 & 0 & 2\gamma_{22} \\ 0 & 3\gamma_{32} & 0 & 0 & 2\gamma_{32} \\ 0 & 0 & 2\gamma_{43} & 2\gamma_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 3\gamma_{11} & 3\gamma_{12} & -\gamma_{13} & 0 & 0 \\ 0 & 3\gamma_{22} & 0 & 0 & 0 \\ 0 & 3\gamma_{32} & 0 & 0 & 0 \\ 0 & 0 & 2\gamma_{43} & 2\gamma_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{2}{3} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 3\gamma_{11} & 3\gamma_{12} & -\gamma_{13} & 0 \\ 0 & 3\gamma_{22} & 0 & 0 \\ 0 & 3\gamma_{32} & 0 & 0 \\ 0 & 0 & 2\gamma_{43} & 2\gamma_{44} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{2}{3} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}
$$

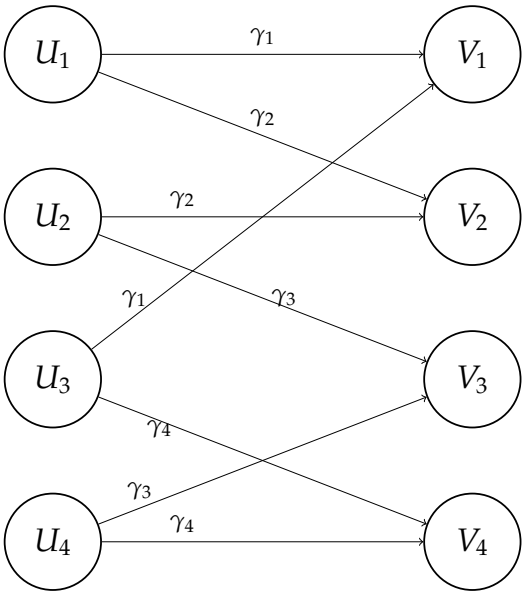

Figure A.1.: Another more complex example bipartite graph setting to apply symbolic Gaussian elimination

A.2. Another example where Symbolic Gaussian Elimination is required

Here, we can see a more complex example where the bipartite graph algorithm is unable to find an optimal virtual bond dimension. The configuration presented has a rank of 3, which could be expressed as a sum of three terms. Unlike simpler cases, no single row or column is directly parallel to another. Instead, the last row is a linear combination of the preceding rows. The operations performed in this case, explicitly outlined here, involve only row eliminations.

$$\Gamma_h = O \cdot \tilde{\Gamma}_h$$

$$\begin{bmatrix} \gamma_1 & \gamma_2 & 0 & 0 \\ 0 & \gamma_2 & \gamma_3 & 0 \\ \gamma_1 & 0 & 0 & \gamma_4 \\ 0 & 0 & \gamma_3 & \gamma_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \gamma_1 & \gamma_2 & 0 & 0 \\ 0 & \gamma_2 & \gamma_3 & 0 \\ \gamma_1 & 0 & 0 & \gamma_4 \\ 0 & 0 & \gamma_3 & \gamma_4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(R_3 = R_3 - R_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \gamma_1 & \gamma_2 & 0 & 0 \\ 0 & \gamma_2 & \gamma_3 & 0 \\ 0 & -\gamma_2 & 0 & \gamma_4 \\ 0 & 0 & \gamma_3 & \gamma_4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(R_3 = R_3 + R_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \gamma_1 & \gamma_2 & 0 & 0 \\ 0 & \gamma_2 & \gamma_3 & 0 \\ 0 & 0 & \gamma_3 & \gamma_4 \\ 0 & 0 & \gamma_3 & \gamma_4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(R_4 = R_4 - R_3) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \gamma_1 & \gamma_2 & 0 & 0 \\ 0 & \gamma_2 & \gamma_3 & 0 \\ 0 & 0 & \gamma_3 & \gamma_4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(\text{Zero Row erased}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \gamma_1 & \gamma_2 & 0 & 0 \\ 0 & \gamma_2 & \gamma_3 & 0 \\ 0 & 0 & \gamma_3 & \gamma_4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A.3. Appendix: Use of AI Tools

During the preparation of this thesis, OpenAI's ChatGPT was utilized to assist in the writing process. Its contributions included:

- Rephrasing and refining textual content in various sections.
- Offering suggestions for organizing sections and improving the flow of ideas.

All content generated by ChatGPT was critically reviewed, verified, and, where necessary, modified to meet the academic standards of the thesis. The use of this tool was limited to textual assistance and did not involve any analytical or computational aspects of the research.

Bibliography

[1] E. National Academies of Sciences and Medicine. *Quantum Computing: Progress and Prospects*. Ed. by E. Grumbling and M. Horowitz. Washington, DC: The National Academies Press, 2019. ISBN: 978-0-309-47969-1. DOI: 10.17226/25196.

[2] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[3] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. S. Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden. "Advances in quantum cryptography." In: *Adv. Opt. Photon.* 12.4 (Dec. 2020), pp. 1012–1236. DOI: 10.1364/AOP.361502.

[4] A. J. Daley, I. Bloch, C. Kokail, S. Flannigan, N. Pearson, M. Troyer, and P. Zoller. "Practical quantum advantage in quantum simulation." In: *Nature* 7920 (2022), pp. 667–676.

[5] I. M. Georgescu, S. Ashhab, and F. Nori. "Quantum simulation." In: *Rev. Mod. Phys.* 86 (1 Mar. 2014), pp. 153–185. DOI: 10.1103/RevModPhys.86.153.

[6] I. Buluta and F. Nori. "Quantum Simulators." In: *Science* 326.5949 (2009), pp. 108–111. DOI: 10.1126/science.1177838. eprint: https://www.science.org/doi/pdf/10.1126/science.1177838.

[7] F. Verstraete, J. J. Garcıa-Ripoll, and J. I. Cirac. "Matrix Product Density Operators: Simulation of Finite-Temperature and Dissipative Systems." In: *Phys. Rev. Lett.* 93 (20 Nov. 2004), p. 207204. DOI: 10.1103/PhysRevLett.93.207204.

[8] B. Pirvu, V. Murg, J. I. Cirac, and F. Verstraete. "Matrix product operator representations." In: *New Journal of Physics* 12.2 (Feb. 2010), p. 025012. DOI: 10.1088/1367-2630/12/2/025012.

[9] C. Hubig, I. P. McCulloch, and U. Schollwöck. "Generic construction of efficient matrix product operators." In: *Phys. Rev. B* 95 (3 Jan. 2017), p. 035129. DOI: 10.1103/PhysRevB.95.035129.

[10] I. P. McCulloch. "From density-matrix renormalization group to matrix product states." In: *Journal of Statistical Mechanics: Theory and Experiment* 2007.10 (Oct. 2007), P10014. DOI: 10.1088/1742-5468/2007/10/P10014.

[11] S. Keller, M. Dolfi, M. Troyer, and M. Reiher. "An efficient matrix product operator representation of the quantum chemical Hamiltonian." In: *The Journal of Chemical Physics* 143.24 (Dec. 2015), p. 244118. ISSN: 0021-9606. DOI: 10.1063/1.4939000. eprint: https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.4939000/15513208/244118_1_online.pdf.

[12] A. J. Daley, C. Kollath, U. Schollwöck, and G. Vidal. "Time-dependent density-matrix renormalization-group using adaptive effective Hilbert spaces." In: *Journal of Statistical Mechanics: Theory and Experiment* 2004.04 (Apr. 2004), P04005. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2004/04/p04005.

[13] S. R. White and A. E. Feiguin. "Real-Time Evolution Using the Density Matrix Renormalization Group." In: *Physical Review Letters* 93.7 (Aug. 2004). ISSN: 1079-7114. DOI: 10.1103/physrevlett.93.076401.

[14] Y.-Y. Shi, L.-M. Duan, and G. Vidal. "Classical simulation of quantum many-body systems with a tree tensor network." In: *Phys. Rev. A* 74 (2 Aug. 2006), p. 022320. DOI: 10.1103/PhysRevA.74.022320.

[15] J. Ren, W. Li, T. Jiang, and Z. Shuai. "A general automatic method for optimal construction of matrix product operators using bipartite graph theory." In: *The Journal of Chemical Physics* 153.8 (Aug. 2020). ISSN: 1089-7690. DOI: 10.1063/5.0018149.

[16] G. K.-L. Chan, A. Keselman, N. Nakatani, Z. Li, and S. R. White. *Matrix Product Operators, Matrix Product States, and ab initio Density Matrix Renormalization Group algorithms*. 2016. arXiv: 1605.02611 [physics.chem-ph].

[17] R. M. Milbradt, Q. Huang, and C. B. Mendl. *State Diagrams to determine Tree Tensor Network Operators*. 2024. arXiv: 2311.13433 [quant-ph].

[18] W. Li, J. Ren, H. Yang, H. Wang, and Z. Shuai. "Optimal tree tensor network operators for tensor network simulations: Applications to open quantum systems." In: *The Journal of Chemical Physics* 161.5 (Aug. 2024), p. 054116. ISSN: 0021-9606. DOI: 10.1063/5.0218773. eprint: https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0218773/20096727/054116_1_5.0218773.pdf.

[19] U. Schollwöck. "The density-matrix renormalization group in the age of matrix product states." In: *Annals of Physics* 326.1 (2011). January 2011 Special Issue, pp. 96–192. ISSN: 0003-4916. DOI: https://doi.org/10.1016/j.aop.2010.09.012.

[20] G. S. Joyce. "Classical Heisenberg Model." In: *Phys. Rev.* 155 (2 Mar. 1967), pp. 478–491. DOI: 10.1103/PhysRev.155.478.

[21] J. C. Bridgeman and C. T. Chubb. "Hand-waving and interpretive dance: an introductory course on tensor networks." In: *Journal of Physics A: Mathematical and Theoretical* 50.22 (May 2017), p. 223001. DOI: 10.1088/1751-8121/aa6dc3.

[22] S. R. White. "Density matrix formulation for quantum renormalization groups." In: *Phys. Rev. Lett.* 69 (19 Nov. 1992), pp. 2863–2866. DOI: 10.1103/PhysRevLett.69.2863.

[23] S. R. White. "Density-matrix algorithms for quantum renormalization groups." In: *Phys. Rev. B* 48 (14 Oct. 1993), pp. 10345–10356. DOI: 10.1103/PhysRevB.48.10345.

[24] J. Biamonte and V. Bergholm. *Tensor Networks in a Nutshell*. 2017. arXiv: 1708.00006 [quant-ph].

[25] J. Biamonte. *Lectures on Quantum Tensor Networks*. 2020. arXiv: 1912.10049 [quant-ph].

[26] F. Verstraete and J. I. Cirac. *Renormalization algorithms for Quantum-Many Body Systems in two and higher dimensions*. 2004. arXiv: cond-mat/0407066 [cond-mat.str-el].

[27] F. Verstraete, M. M. Wolf, D. Perez-Garcia, and J. I. Cirac. "Criticality, the Area Law, and the Computational Power of Projected Entangled Pair States." In: *Physical Review Letters* 96.22 (June 2006). ISSN: 1079-7114. DOI: 10.1103/physrevlett.96.220601.

[28] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac. "Computational Complexity of Projected Entangled Pair States." In: *Physical Review Letters* 98.14 (Apr. 2007). ISSN: 1079-7114. DOI: 10.1103/physrevlett.98.140506.

[29] Y.-Y. Shi, L.-M. Duan, and G. Vidal. "Classical simulation of quantum many-body systems with a tree tensor network." In: *Physical Review A* 74.2 (Aug. 2006). ISSN: 1094-1622. DOI: 10.1103/physreva.74.022320.

[30] G. Vidal. "Class of Quantum Many-Body States That Can Be Efficiently Simulated." In: *Physical Review Letters* 101.11 (Sept. 2008). ISSN: 1079-7114. DOI: 10.1103/physrevlett.101.110501.

[31] G. Evenbly and G. Vidal. "Algorithms for entanglement renormalization." In: *Physical Review B* 79.14 (Apr. 2009). ISSN: 1550-235X. DOI: 10.1103/physrevb.79.144108.

[32] M. Levin and C. P. Nave. "Tensor Renormalization Group Approach to Two-Dimensional Classical Lattice Models." In: *Physical Review Letters* 99.12 (Sept. 2007). ISSN: 1079-7114. DOI: 10.1103/physrevlett.99.120601.

[33] K. Nakayama. *Randomized higher-order tensor renormalization group*. 2023. arXiv: 2307.14191 [cond-mat.stat-mech].

[34] G. Evenbly and G. Vidal. "Tensor Network Renormalization." In: *Physical Review Letters* 115.18 (Oct. 2015). ISSN: 1079-7114. DOI: 10.1103/physrevlett.115.180405.

[35] TensorNetwork. *TensorNetwork.org*. Accessed: 2025-01-05. 2025. URL: https://www.tensornetwork.org.

[36] G. M. Crosswhite, A. C. Doherty, and G. Vidal. "Applying matrix product operators to model systems with long-range interactions." In: *Phys. Rev. B* 78 (3 July 2008), p. 035116. DOI: 10.1103/PhysRevB.78.035116.

[37] N. Nakatani and G. K.-L. Chan. "Efficient tree tensor network states (TTNS) for quantum chemistry: Generalizations of the density matrix renormalization group algorithm." In: *The Journal of Chemical Physics* 138.13 (Apr. 2013). ISSN: 1089-7690. DOI: 10.1063/1.4798639.

[38] J. E. Hopcroft and R. M. Karp. "An $n^{5/2}$ *Algorithm for Maximum Matchings in Bipartite Graphs*." In: *SIAM Journal on Computing* 2.4 (1973), pp. 225–231. DOI: 10.1137/0202019. eprint: https://doi.org/10.1137/0202019.

[39] M. T. Heath. *Scientific Computing*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2018. DOI: 10.1137/1.9781611975581. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611975581.