國立台灣大學理學院物理學系

碩士論文

Department of Physics

College of Science

National Taiwan University

Master Thesis

比較不同張亮網路演算法應用在二微多體量子物理系統之優劣

Comparison between Tensor Network Algorithms for Two Dimensional Infinite Quantum Many-Body Systems

周昀萱

Yun-Hsuan Chou

指導教授：高英哲博士

Advisor: Ying-Jer Kao, Ph.D.

中華民國 105 年 4 月

April, 2016

誌謝

摘要

如何判斷多體量子系統的相變化，且從微觀系統來得到巨觀上的物理性質，在現代仍為凝態物理學中十分有趣的領域。

從 NRG 的為起點開始，多年來出現了許多突破性的演算法。其中 DMRG 在一維的系統的模擬中得到了相當好的結果。但在二微系統中，因為 Area law 的關係使其表現不如在一微系統中精確，不僅如此，在二維系統中，計算複雜度上升之速度也非一維系統可比擬。為了解決這些問題，因而出現了許許多多不同的建立在張亮網路理論上的演算法。

此篇論文，紀錄了幾個當今較為主流或新穎並用以模疑二維量子系統的張亮網路演算法。一開始將簡單解釋張亮網路的基本理論; 再來會介紹如何實做、優化演算法，以增加精確度和降低計算複雜度。章節中也附上偽代碼，來說明實作中應注意之細節。最後會比較它們計算二維易辛模型與海森堡模型的結果，來說明各演算法之優缺點。

Abstract

Determining the phase transition of many body systems and the physical properties of macroscopic systems from microscopic description are still challenging in condense matter physics.

Since the numerical renormalization group (NRG) came out, various algorithms sprang up like mushrooms for analyzing these problem . Among all, the density matrix renormalization group (DMRG) could be considered as the most remarkable outcome, which analyze accurately in one dimensional systems. However, it perform worse in two dimensional systems. Not only the physical reasons, such as the area law, but also the rapid increment of computational complexity which is much higher than one dimensional system.

In the thesis, we recorded some of popular tensor network algorithms which are developed for handling the problems in two-dimensional systems. First of all, we briefly introduce the tensor network theory. Then, in the following sections, we shown the network diagrams and simple pseudocode for explaining how to implement these algorithms.
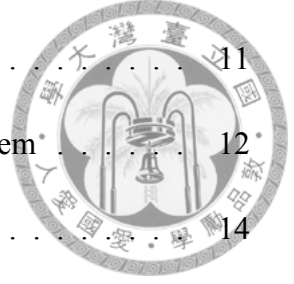
Key words— matrix product state(MPS), projected entangled pair state(PEPS), projected entangled simplex state, infinite time-evolveing block-decimation, corner transfer matrix, tensor renormalization group, Benchmarks, uni10.

Contents

Bibliography

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Understanding the phenomena of quantum many-body systems is one of most challenging problem in condense matter physics. Since, the coefficients required for describing entire systems grow exponentially with system size. For instance, If we desired to fully describe a N-site spin chain and each spin has d probable states, the requirement of coefficients is d^N. Due to the rapid increment of computational consumptions, it's impossible to simulate the system whose size is larger than 50 for classical computers.

For addressing that problems, various numerical methods have been developed. For instance, density matrix renormalization group (DMRG) [1] [2] acquire the accuracy ground state energy and provide a dominant tool to study properties of one dimensional systems. Furthermore, the theory of DMRG is related with matrix product states (MPS) [3] [4] , which could describe the wave-function of one dimensional system and be explicitly represented by *tensor diagrams*.

No doubt, DMRG is the most successful method in one-dimensional systems. However, it's failure in higher dimensional systems due to the insufficiency of dealing with the entanglement in systems from matrix product states. For two-dimensional lattices, the projected entangled pair state (PEPS) [5] [6] has been applied to deal with that problem

and many algorithms sprang up like mushrooms, such as time-evolving block decimation [7] [8] and multiscale entanglement renormalization ansatz [9].

Chapter 2

Tensor Network Theory

This section begins from a fundamental question: How to draw a tensor network diagrams? In tensor network theory [10] [11] [12], we used to represent tensors graphically instead of complicated equations, because *tensor diagrams* can map to quantum states and geometric lattices explicitly. Base on its clear representation, the implementation of tensor network algorithms become simply.

2.1   Representation of tensors in tensor Networks

In mathematical concept, a tensor is considered as a multi-dimensional array of scalers. The arrangement of the elements in a tensor is dependent on its *indices* and the *rank* of tensor is equivalent to the number of indices. Thus, a rank-0 tensor is a scaler (T), a rank-1 tensor is a vector (T_i), a rank-2 tensor is a matrix (T_{ij}) and so on.

Graphically, we usually use a node and few bonds to represent a tensor. Some specified examples are shown in figure 2.1, the number of bond is equal to the rank of tensor, which means that tensors without bonds, with a single bond, with two bonds and with three bonds can map to scalers, vectors, matrices and rank-3 tensor.

Then, we should determine the dimension of tensors. Explicitly, the dimension of
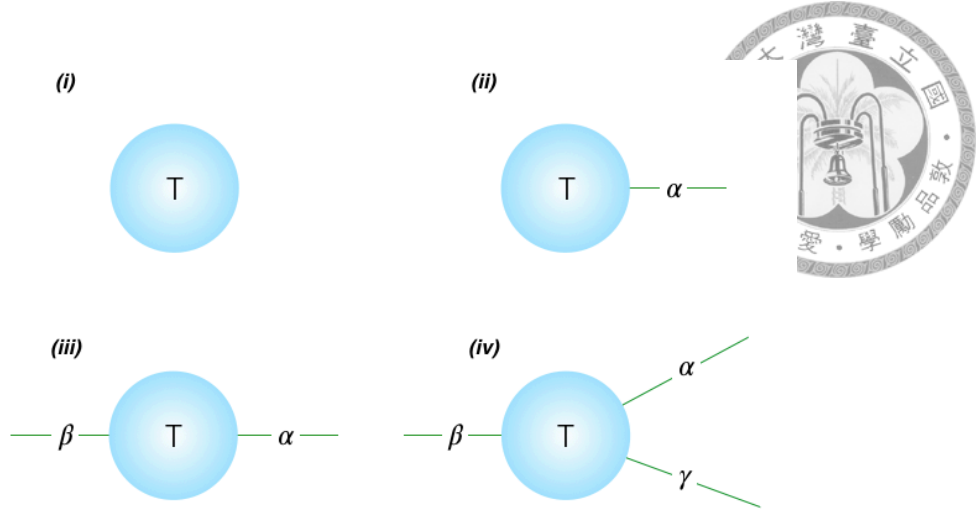
Figure 2.1: Usually we use a note and few bonds to compose a tensor and the numbers of bond depend on the rank of tensor. (i) A tensor without bonds is a scaler T, (ii) A tensor with one bond is a vector T_α, (iii) A tensor with two bonds is a Matrix $T_{\alpha\beta}$, (iv) A tensor with three bonds is a rank-3 tensor $T_{\alpha\beta\gamma}$.

rank=0 (scaler) is equal to one. However, the dimension of tensors higher than rank-0 depend on the bond dimensions. For instance, in Fig.2.1(iv), it's a rank-3 tensor and the dimensions of the indices are χ_α, χ_β, χ_γ and T contains $\chi_\alpha\chi_\beta\chi_\gamma$ components.

2.2 Tensor operations and tensor network diagrams

Basically we can't calculate an fold tensors directly. So the first step, we should unfold tensors. On the other words, we must make their rank lower than 3 before operating. The process is also called *permuteation*.

2.2.1 Permutation

Fig2.2 is a simple permutation example which means that the tensor A permuted into tensor $\hat{A}$,

$$A_{\alpha\beta\gamma} \to \hat{A}_{\alpha\gamma\beta} \tag{2.1}$$
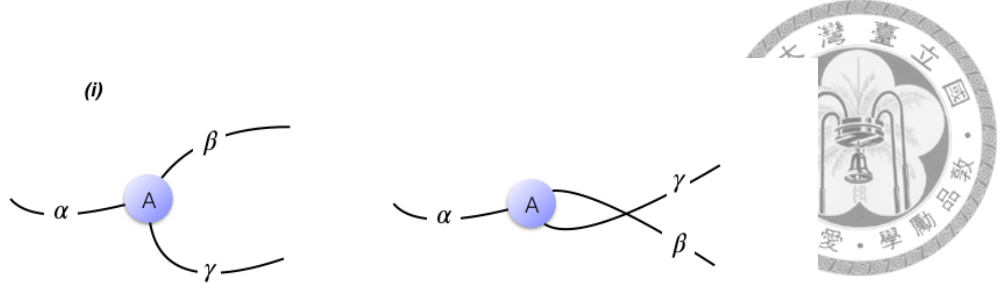
Figure 2.2: Simple example of a tensor permutation. Permute tensor A from indices $\alpha\beta\gamma$ to $\hat{A}_{\alpha\gamma\beta}$

The components of $\hat{A}$ and A are equivalence, but having different arrangements. In order to explaining clearly, I use two flags, incoming (BD_IN) and outgoing (BD_OUT) bonds which are also designed for distinguishing different types of *uni10::Bond* in *Uni10*, to show how to reduce the rank of a tensor. Further explanation, BD_IN and BD_OUT can be imagined as rows and columns of a matrix. For instance, if the indices of $T_{\alpha\beta\gamma}$ ordered like Fig2.3(i), it means that $T_{\alpha\beta\gamma}$ is a matrix $T_{\chi_\beta\chi_\gamma,\chi_\alpha}$. Similarity, If it's like Fig2.3(ii), it means that $T_{\alpha\beta\gamma}$ is a matrix $T_{\chi_\beta,\chi_\alpha\chi_\gamma}$.
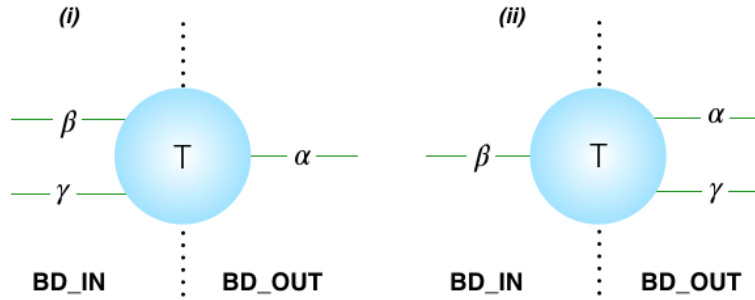
Figure 2.3: (i) Unfold a tensor to a matrix $T_{\chi_\beta\chi_\gamma,\chi_\alpha}$, (ii) Unfold a tensor to a matrix $T_{\chi_\beta,\chi_\alpha\chi_\gamma}$.

2.2.2 Tensor contraction

Tensor contraction is defined as the sum of all products of the shared indices of tensors. For instance, tensor contraction between two rank-2 tensors $A_{\alpha\beta}$ and $B_{\beta\gamma}$ which is equivalent to inner product between matrix, inner product between matrix $A_{\chi_\alpha,\chi_\beta}$ and

$$B_{\chi_\beta, \chi_\gamma}$$

$$C_{\alpha\gamma} = \sum_{\beta=1}^{\chi_\beta} A_{\alpha\beta} B_{\beta\gamma}, \tag{2.2}$$

and the tensor diagram is shown in Fig2.4(i),
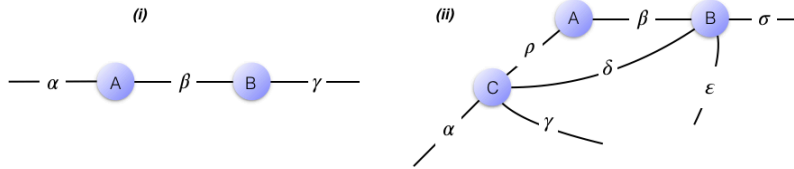
Figure 2.4: (i) Contraction between $A_{\alpha\beta}$ and $B_{\beta\gamma}$ is equivalent to inner product between matrix $A_{\chi_\alpha, \chi_\beta}$ and $B_{\chi_\beta, \chi_\gamma}$ (ii) A simple tensor network

Now we considered a more complexity network, in Fig2.4(ii). The equations is written as,

$$D_{\alpha\gamma\sigma\epsilon} = \sum_{\beta\rho\delta} A_{\rho\beta} B_{\beta\sigma\epsilon\delta} C_{\gamma\delta\rho\alpha}, \tag{2.3}$$

The contraction processes of this network can be separated to some steps.
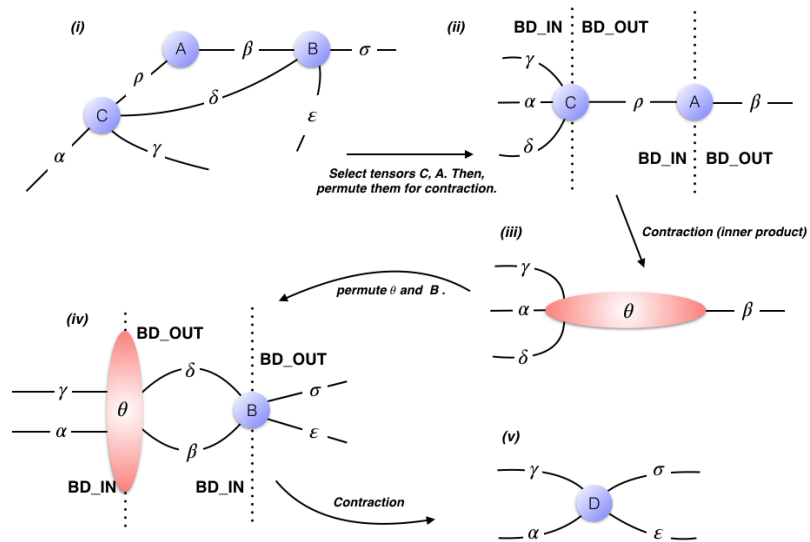
Figure 2.5: The contraction processes of the example shown in Fig2.4

6

(i) First, select a pair of tensors arbitrarily. In Fig2.5, we chose beginning from contracting tensors A and C . Actually, this step is very crucial. For example, If the dimensions of each bonds are the same and we select tensor B and C instead, it not only enlarge computational consumption, but also make algorithms inefficient. I'll discuss more detail in **??**.(ii) Second, A and C should permute into legal shape. In this case, A and C are permuted into $A_{\gamma\alpha\delta\rho}$ with one BD_OUT and $C_{\rho\beta}$ with one BD_IN, therefore the question evolves to inner product matrix $A_{\chi_\gamma\chi_\alpha\chi_\delta,\chi_\rho}$ and A_{χ_ρ,χ_β}.Repeated step (i) and (ii) again, after getting the tensor $\theta_{\gamma\alpha\delta\beta}$. Permute tensors θ and B into $\theta_{\gamma\alpha\delta\beta}$ having two BD_OUT and $B_{\delta\beta\sigma\epsilon}$ having two BD_IN and we will get tensor D, after contracting them by matrix multiplication.

2.3   Describe Quantum states with tensor network

Before drawing a many-body system with TN, we should discuss how to describe a chain of N particles, with each particle having d different states and why using TN. Hence, we considered a many-body systems is composed of many local particles and have studied that a pure state corresponds to a vector in Hilbert space, so we intuitively imagined that the wave-function of systems can be written as,

$$|\Psi_N\rangle = \sum_{i_1,i_2,...,i_N} C_{i_1,i_2,i_3,...,i_N} |i_1\rangle \otimes |i_2\rangle \otimes \ldots \otimes |i_N\rangle, \tag{2.4}$$

each individual $|i_1\rangle, |i_2\rangle, \ldots, |i_N\rangle$ spanned by an orthogonal basis which the degree of freedom is d. After writing down the eq.2.4, we are able to build a TN for quantum states, each bond of tensor corresponds to the local Hilbert space and the dimensions of each bond is equivalent to the states of particle on i site. The geometric notation of $|\Psi\rangle$ and some fundamental operations are shown in Fig.2.6.

From 2.4, we are aware of the number of coefficients, $C_{i_1,i_2,i_3,...,N}$, is proportional to d^N and fully describing a many-body system which is too expensive. Fortunately, the wave-function can be decomposed to two subsystem by Schmidt Decomposition and there are
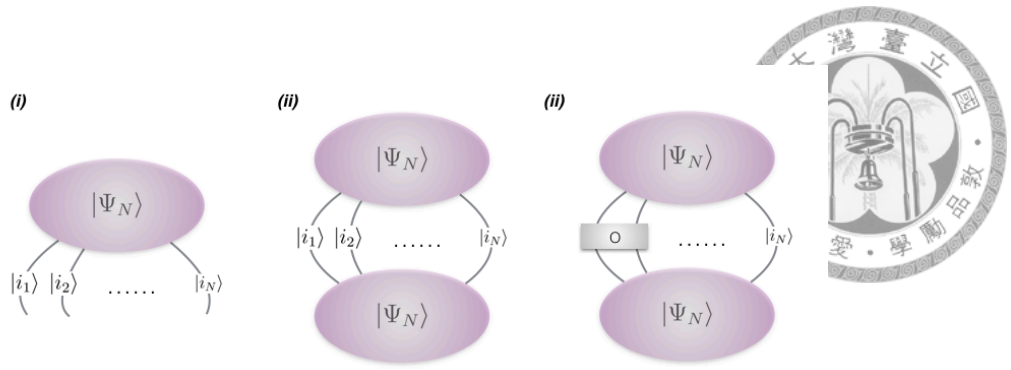
Figure 2.6: (i) Wave-function: $|\Psi_N\rangle$ (ii) Norm of $|\Psi_N\rangle$; $\langle\Psi_N\,|\,\Psi_N\rangle$ (iii) Expectation value of observable O; $\langle\Psi_N|\,O\,|\Psi_N\rangle$

more details in chapter.3.

Chapter 3

2-D Imaginary Time Evolving Block Decimation

In this chapter we introduce some different ways to implement two-dimensional imaginary time-evolving block decimation and apply them in calculation of ground state of Heisenberg and transverse Ising model on two-dimensional square lattice. In section 3.1 and 3.2, we briefly review the idea of imaginary time evolution (TEBD) [7] [8] and explain how to extend it to an infinite two-dimensional system [13]. Second, we briefly review another implementation which is able to make 2D-iTEBD more stable[14]. In the last section 3.4, we record various particulars which are helpful optimizing algorithms.

3.1 Imaginary Time Evolution

Theoretically, if having the imaginary time evolution operator $e^{-\tau H}$, we could project any random states to the ground state, as long as the wave-function can be written as,

$$|\psi_0\rangle = \frac{e^{\tau H}|\Psi\rangle}{\| e^{\tau H}|\Psi\rangle \|} \tag{3.1}$$

but according to the eq.3.1, we may found that the number of coefficients in an origin evolution operator $e^{-\tau H}$ is proportional to $2^N \times 2^N$. On the other words, it is impossible

to update entire system directly. In order to restricting the rapid dimensional growth, we apply *Suzuki Trotter decomposition* to approximate. The main idea of *Suzuki Trotter* is decomposing the whole system to lots of units cell and using some smaller operations to update the wave-function.

$$e^{\delta A+B} = e^{\delta A}e^{\delta B} + O(\delta^2) \tag{3.2}$$

eq.3.2 means the first-order Suzuki Trotter decomposition, and A and B are non-commutative with each other.

Now that the dimension of the evolution operator is reduced to a n-site operator and in chapter **??**, we have shown that a many-body system can map to a MPS [3][4] or PEPS [5] structure, so we can draw the process of updating a ground state like Fig.3.1,
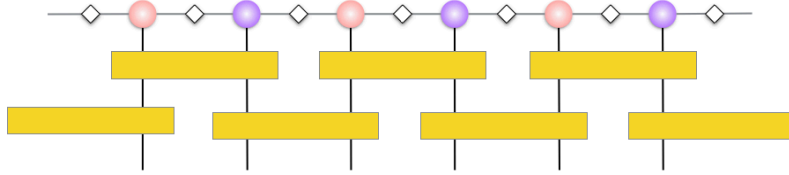
Figure 3.1: The red and blue tensor denotes on *odd* and *even* sites. The yellow one are time evolution operators $e^{-\tau H_{k,k+1}}$, $e^{-\tau H_{k+1,k}}$

On the other work, contract the tensors in Fig.3.1 repeatedly until the ground state energy to the minimum. The remained tensor is considered as the ground state of the system. So the next question: How can we contract them and preserve the structure like Fig3.2? This answer is iTEBD.

Figure 3.2: The simple form of a matrix product state.

3.2 Simple Infinite Imaginary-Time Evolving Block Decimation for 2-D system
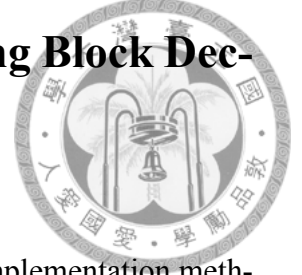
In this section, we apply the TN diagrams to introduce various implementation methods. More theoretical details are included in the references [15] [16] [17].

3.2.1 Simple Description of iTEBD for 1-D system

The algorithm start from 2 random states and 2 random diagonal matrices which are considered as entanglement between particles. In the TN diagrams, Fig.3.3, the states and entanglement between neighbor sites are represented by the nodes and bonds with different colors.
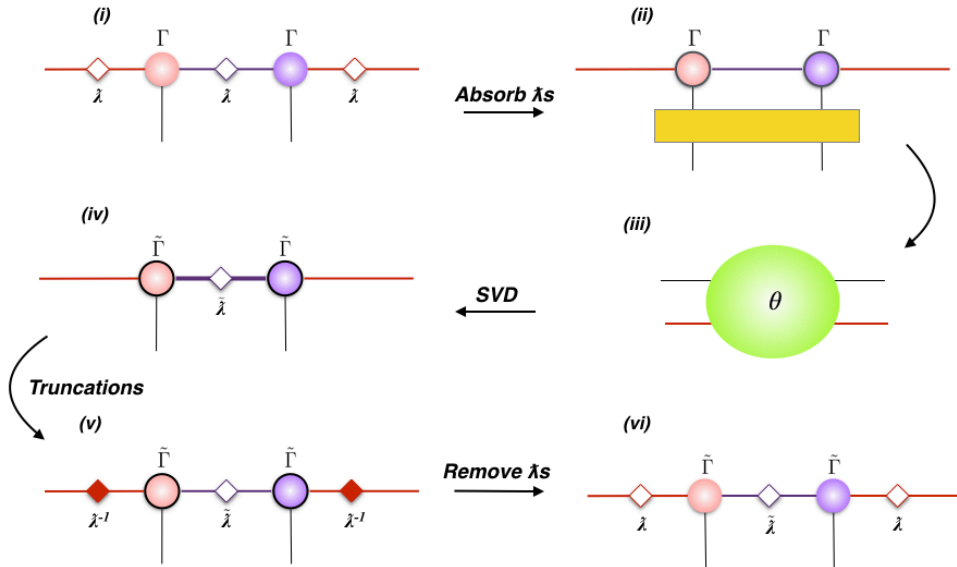
Figure 3.3: (i)Absorb all λ to Γ. (ii) Contract an evolution operator $e^{-\delta H}$ for evolving the system. (iii) Decompose the tensor θ by SVD. (iv) Truncate and Update the states and λ on the green bond.(v) Remove λ for obtaining the states. (iv) After updating the states and λ on the purple bond, apply the way to update the red bond and repeat all the steps until the ground state convergence.

The processes shown in Fig.3.3 is a standard strategy for implementation of iTEBD and also called *Simple Update*.

11

In one dimensional system, the performance of Simple Update is pretty well, because 1-D systems obey the canonical form and have less influences of environment. However, in 2-D systems, due to the area law, we need consider the environment more restrictively when measuring the local observable. Moreover, the computational consumption is another serious problem, owing to the growth of a state's dimension which is proportional to dD^4.

In order to solving that obstacles, optimizations of 2-D algorithms became an important part in condense matter physics. This chapter we focus on obtaining good enough projected entangled pair states from 2-D iTEBD and the strategies of improving measurement would be shown in following chapters.

3.2.2   Description and Pseudocode of iTEBD for 2-D system

Now that we stated to extend it to a two dimensional system. In chapter.**??**, we have known that a two dimensional many-body system is able to be represented by PEPS. Due to impossibly drawing an network of infinite sites, the structure of infinite PEPS (iPEPS) is decided by the geometry of the lattice and the unit cell we chosen. In usual, the size of unit cell depend on the n-site evolution operator. For instance, if the target is updating iPEPS of a square lattice with 2-site operator, the tensor diagram would be drawn as Fig.3.4.
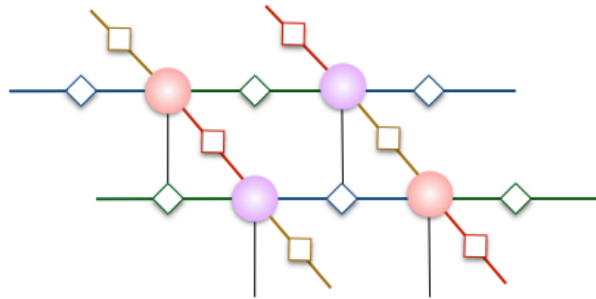

Figure 3.4: Four sites unit cell in iPEPS.

After setting the form of iPEPS, we start to deal the question of updating states. The most intuitive scheme is to apply the scheme of *Simple Update* which is shown in Fig.3.5.

The steps are similar to the iTEBD on 1-D systems. However, there are some differences. Firstly, the projected entangled pair states is a rank-5 tensor. Secondly, there are more entangled should be considered, due to increased neighbor sites.
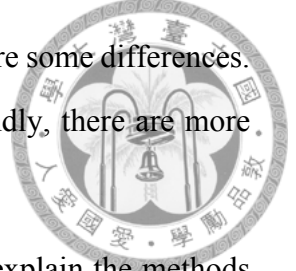
More theoretical descriptions are written in [**??**][**??**]. Here, we explain the methods with TNs and some simple pseudocodes. The basic idea of *2-D iTEBD* is to update the states from four directions by *Simple Update*. The example starting from updating the green bond is shown in Fig.3.5 and Fig.3.6,
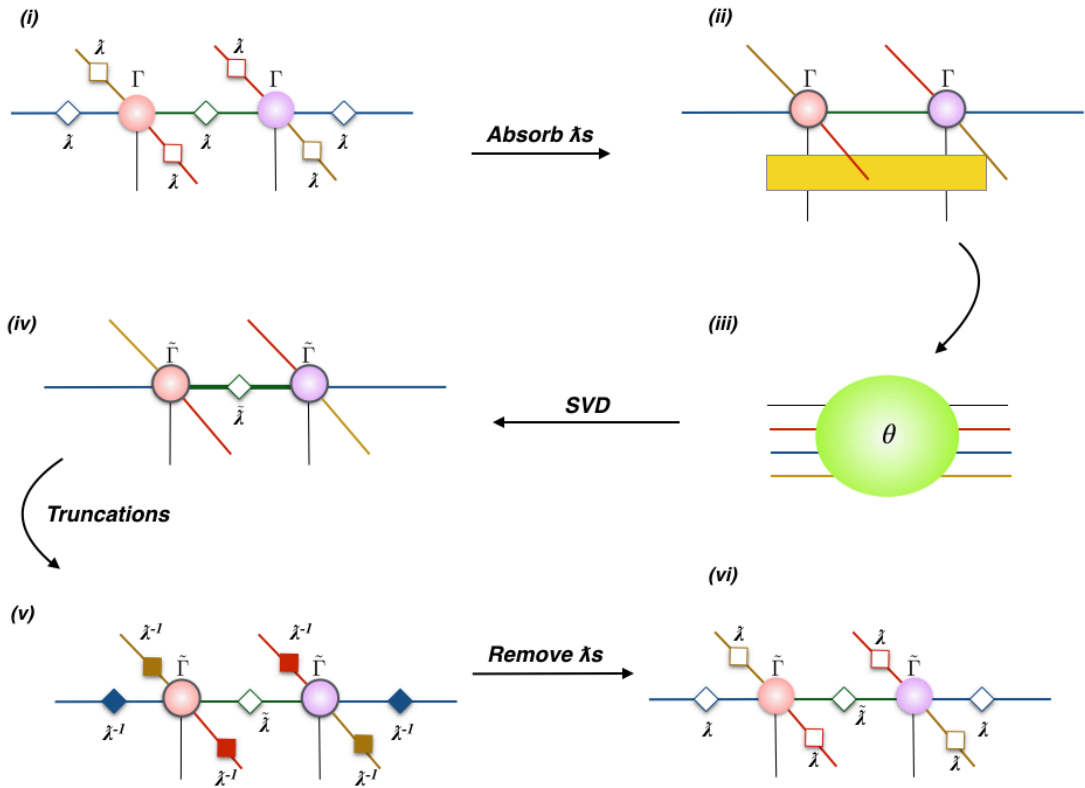

Figure 3.5: Absorb all λ to Γ. (ii) Contract an evolution operator $e^{-\delta H}$ for evolving the system. (iii) Decompose the tensor θ by SVD. (iv) Truncate and Update the states and λ on the green bond.(v) Remove λ for obtaining the states. (iv) Obtain a original form of iPEPS. Repeat all the step to update the other bonds until the ground state energy convergence

It's the same as one dimensional case, The first step is to absorb all λ around the sites. The tensor with gray bold means the state have absorbed all entanglements. Secondly, contract the gate, $e^{-\delta H}$, for getting the tensor θ. Thirdly, apply singular value decomposition to update states and the entanglement on the green bond. After decomposing θ, we found that the dimension of the green bond increase to dD^4. Therefor, truncation plays

a significant roles for keeping the dimension in the algorithm. In the end of the updating processes, multiply pseudo inverse of all λ to the tensors for reducing to original form.
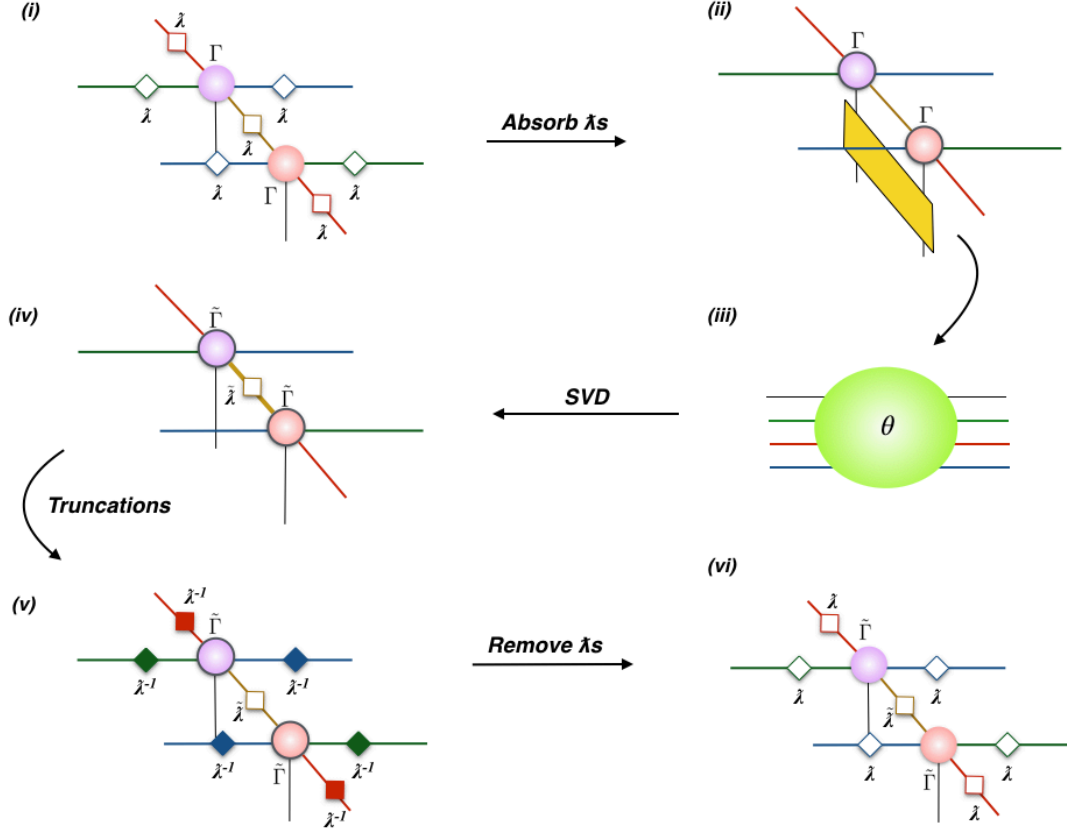

Figure 3.6: Update the yellow bond and the steps are similar to Fig.3.5

For easily to imagine how to updating the others directions. The updating steps of yellow bond are shown In Fig.3.6.

3.3 Ameliorate two-dimensional iTEBD

This method which make the algorithm more stable was published by *M. B. Hastings* [14]. Although *Simple Update* shown in section.3.2.2 can obtain pretty good states, it's not stable and efficient enough. The reason is that too many multiplications of pseudo inverse λ at the step Fig.3.5(v). In numerical methods, it's hard to deal the problem of dividing the value which is equal or approach to zero. On the other words, the more inverse operations, the more the probability of bring about divergence or destroying algorithms.

For reducing the risk of breaking algorithms. Firstly, declare the states Γ which include two entanglements among them. For instance, In Fig.3.7(i), the red tensor is considered as multiplication of A and the λ on the yellow. The purple one is multiplication of B and remained λ. Secondly, in Fig.3.7(ii), because the entanglements on red and blue bonds are included in tensor B, we just need absorb the yellow one and contract the evolution operation for obtaining tensor C. Thirdly, we contract the red and blue λ to the red and blue bonds which belong to the original tensor A in tensor C for getting θ. Fourthly, getting the $\tilde{B}$ from decomposing and truncating θ. Owing to avoid multiplying inverse matrices, we get $\tilde{A}$ by contracting tensor C and $\tilde{B}$ and multiply an inverse λ of yellow bond for removing the entanglement and reducing to original form.
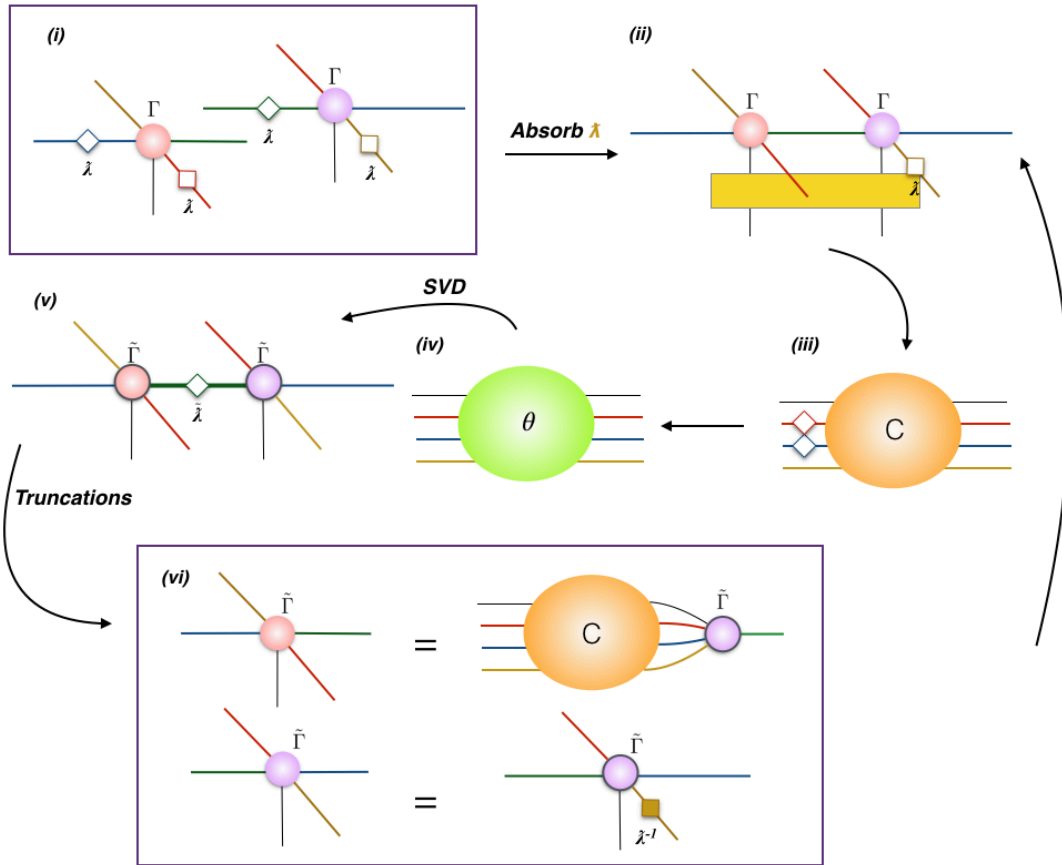

Figure 3.7: The tensor network diagrams for the ameliorated 2-D iTEBD.

Finally, repeat all the processes shown in Fig.3.7 to update different bonds until the convergence of the ground state energy.

3.4 Optimizations

3.4.1 Initialization

Intuitively, the initialization of states should not affect the result. However, it's a serious misunderstanding. Actually, stating from a awful initial sates may break the algorithms or hardly converge.

From the viewpoint of physics, translational invariance is one of essential properties in many-body system, So we can assume that the group state on two sites should be similar. For instance, if the TN diagram of the states is shown as Fig 3.8(i), Fig 3.8(ii) might be the better way to initialize the states.
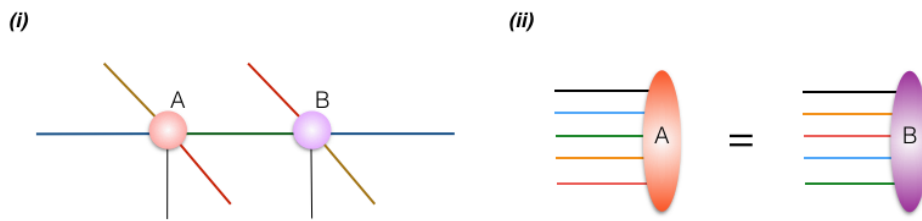

Figure 3.8: (i) The structure of PEPS, (ii) The initialization of states

3.4.2 Truncattion Error

3.4.3 QR decomposition

Though the strategy described in previous sections improve the stability, it's not efficient enough. The reason why is that the dimension of tensor θ, in Fig.3.5(iii) and Fig.3.7(iii), is proportional to $d^2 D^6$. In addition to that, the time complexity of singular value decomposition is proportional to $O(NM^2)$. In conclusion, those steps are expensive and the dimension of tensor θ must be redued for accelerating.
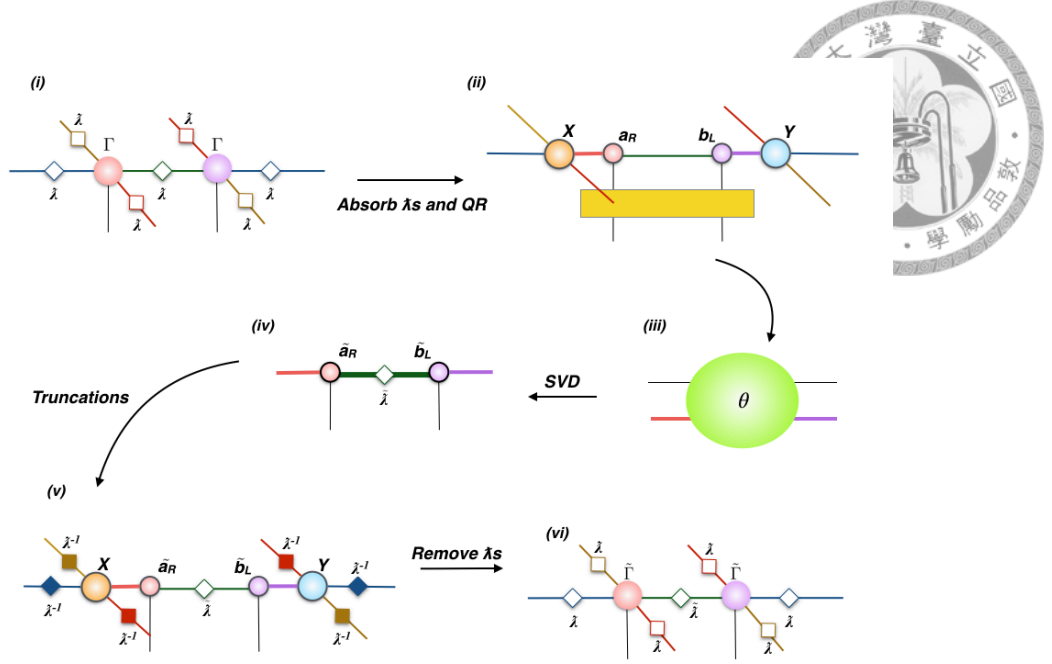
Figure 3.9: The tensor network diagrams for the ameliorated 2-D iTEBD.

To achieve the goal, the projected pair state must be decomposed by QR decomposition [13] [?]. The processes making *Simple Update* more efficient is illustrated in Fig 3.9. Most of the steps shown in Fig 3.9 are like in Fig 3.5. The only difference is that after absorbing all the λ, we decompose the state to an orthogonal matrix and an upper triangular matrix by QR. For instance, in Fig 3.9 (ii), the state A is decomposed to an orthogonal matrix X and upper triangular matrix a_R. Due to the columns of X are orthonormal, $XX^\dagger$ is equal to I. In the other word, the tensor X can be ignored and we just need consider the tensor a_R by QR. Similarity, the state B can be decomposed to a lower triangular b_L and an orthogonal matrix Y by LQ which is equivalence to operate QR decomposition after transpose the matrix. Next, (iii) we can obtain the tensor θ whose dimension is d^2D from a_R, b_L and an evolution operator.

The strategy to accelerate *Ameliorate Simple Update* is shown in the Fig 3.10 and its main idea is also to reduce the dimension of θ.
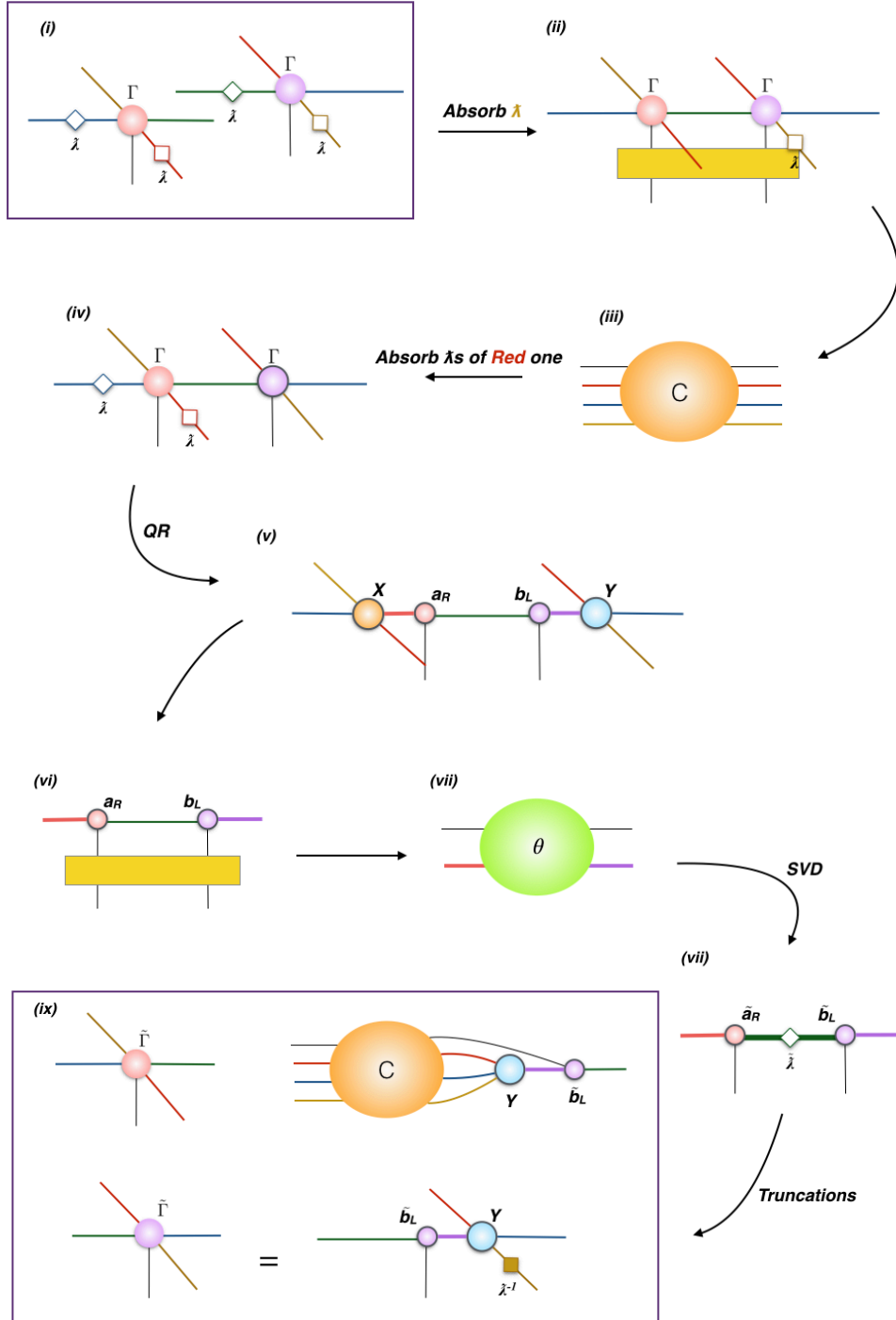
Figure 3.10: The tensor network diagrams for the ameliorated 2-D iTEBD.

3.5 Comparison

So far, we have benchmarked the improved iTEBD.

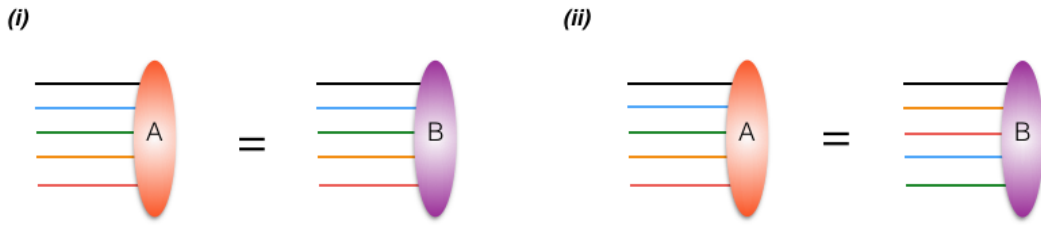### 3.5.1 Different Initializations

Figure 3.11: (i) Type 1, (ii) Type 2
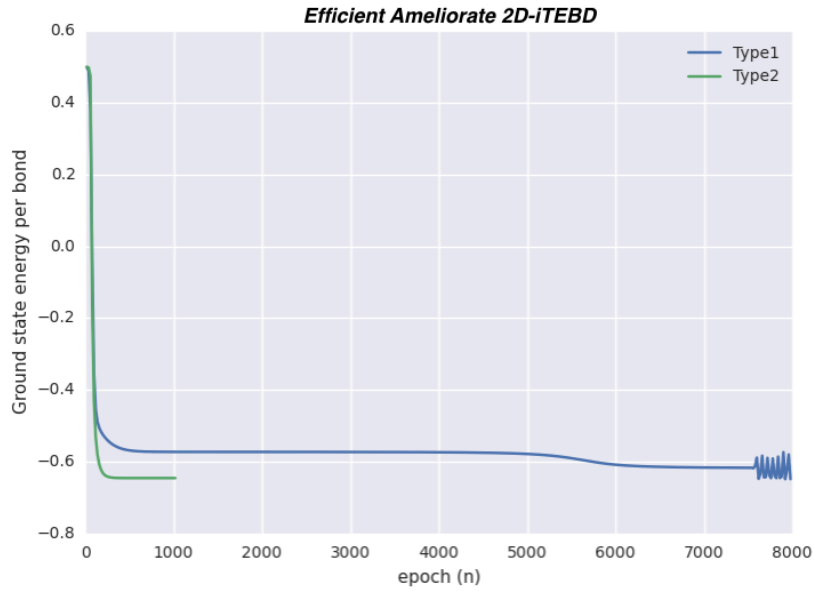
See Fig 3.12, the both cases are

Figure 3.12: The Blue line represents updating tensors from the initial state shown in Fig 3.11 (i) and the green one represents updating from Fig 3.11 (ii)
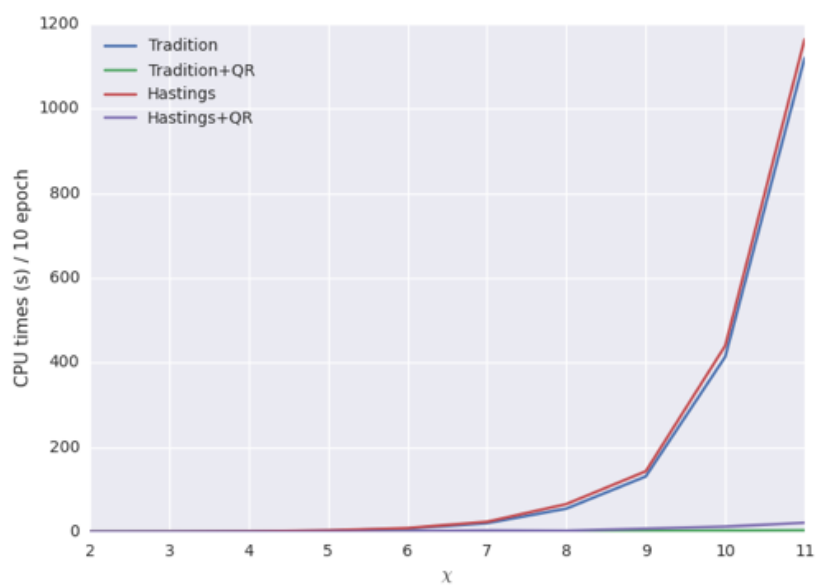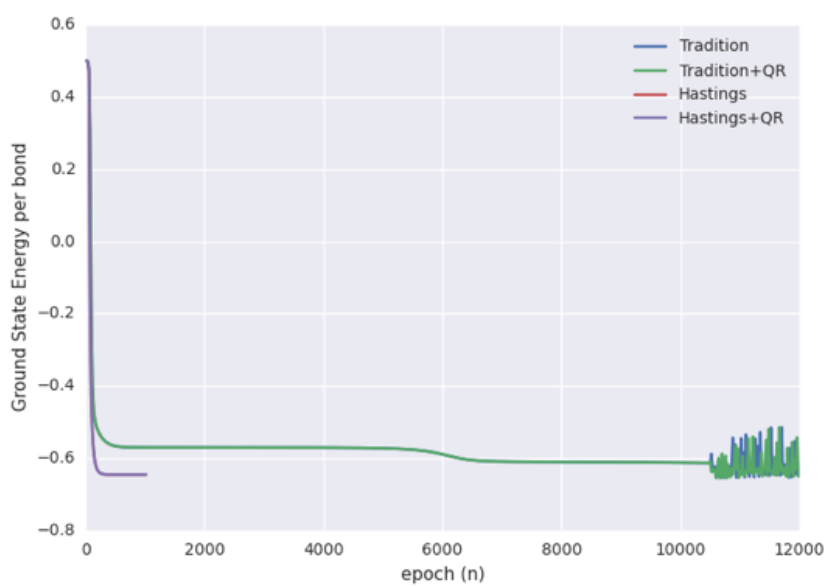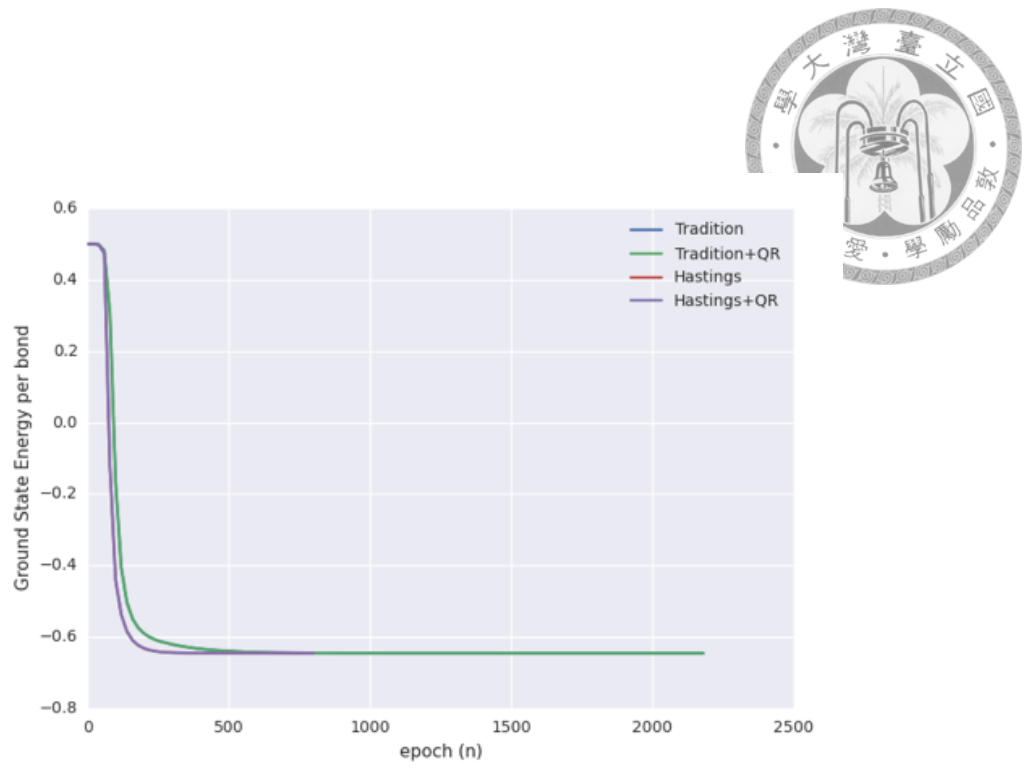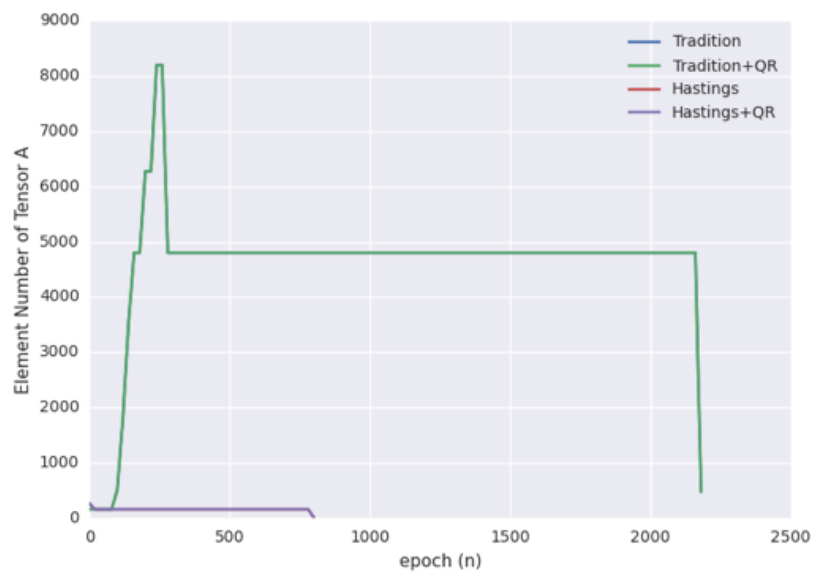
Figure 3.13

Figure 3.14

Figure 3.15

Figure 3.16

Chapter 4

Infinite Projected Entangled Simplex State

Introduce to PESS ansatz.

4.1   Simplex-solid State

The theroy of simplex-solid state.

4.2   Infinite Kagome Lattice

The algorithm of PESS.

4.2.1   3-PESS

The example on Kagome lattice.

4.2.2 5-PESS

4.3 Infinite Square Lattice

The example on Kagome lattice.

4.3.1 4-PESS (Rank-3 local tensors)

4.3.2 4-PESS (Rank-5 local tensors)

Chapter 5

Corner Transfer Matrix

Introduction of Corner Transfer Matrix.

5.1 Obtain States from PEPS

CTM with iPEPS.

5.2 Obtain States from PESS

CTM with PESS.

Chapter 6

Summary

Bibliography

[1] S. R. White, **69**, 2863 (1992).

[2] S. R. White, **48**, 10345 (1993).

[3] F. Verstraete and J. I. Cirac, **73**, 10.1103/PhysRevB.73.094423.

[4] S. Östlund and S. Rommer, **75**, 3537.

[5] V. Murg, F. Verstraete, and J. I. Cirac, **75**, 10.1103/PhysRevA.75.033605.

[6] F. Verstraete, V. Murg, and J. Cirac, **57**, 143.

[7] G. Vidal, **91** (2003), 10.1103/PhysRevLett.91.147902.

[8] G. Vidal, **93** (2004), 10.1103/PhysRevLett.93.040502.

[9] G. Vidal, **99** (2007), 10.1103/PhysRevLett.99.220405.

[10] J. Jordan, "Studies of infinite two-dimensional quantum lattice systems with projected entangled pair states," .

[11] R. Orús, **349**, 117.

[12] R. B. Bauer, "Tensor network states," .

[13] W. Li, J. von Delft, and T. Xiang, **86**, 10.1103/PhysRevB.86.195137.

[14] M. B. Hastings, **50**, 095207.

[15] G. Vidal, **98** (2007), 10.1103/PhysRevLett.98.070201.

[16] H. C. Jiang, Z. Y. Weng, and T. Xiang, **101**, 10.1103/PhysRevLett.101.090603.

[17] R. Orús and G. Vidal, **78**, 10.1103/PhysRevB.78.155117.