Tensor network methods for quantum simulation

van Eersel, H.

Award date:
2011

Tensor network methods
for quantum simulation

Harm van Eersel

December, 2010

Abstract

Quantum computing is a new paradigm in computer science, based on the phenomena of quantum physics. To classically represent an arbitrary quantum state, an exponential number of classical resources is necessary. Tensor network methods – based on a graph theoretic interpretation of tensors – do allow for an efficient classical representation of a limited class of quantum states. In this thesis several tensor network methods are reviewed, including matrix product states, tree tensor networks and projected entangled pair states their construction, transformation and measurement. Furthermore, some applications are discussed briefly.

Also, some original work is presented for projected entangled pair states: an update procedure for unitary quantum operators, a more efficient construction method in the teleportation picture and a comparison of the different geometries.

Contents

Introduction

Quantum computing is a new computing paradigm based on quantum physics instead of classical physics. Using the phenomena of quantum physics for computation, it is believed that a larger class of problems can be solved more efficiently than on classical computers. Quantum physics introduces the new concept of the quantum state. Compared with a classical state, the number of coefficients needed for the description of a state grows exponentially in the number of quantum bits, resulting in an exponential classical representation of these states. While this larger state space is essential for the additional power of quantum computers, it also makes the simulation of quantum systems intractable on classical computers.

It turns out, however, that some classes of quantum systems do allow for an efficient classical simulation. Several representation methods have been proposed in the literature. They differ in the classes of states that can be represented efficiently and the operations that can be applied to these states. An ideal representation method has the following properties:

- polynomial space complexity in the number of quantum bits for an interesting class of quantum states
- a construction method to represent an arbitrary quantum state
- an efficient update procedure of the representation after quantum operations on the state
- efficient (local) measurement of the state and an efficient update procedure to represent the post-measurement state

A number of the proposed formalisms that fulfill (some) of these requirements are based on so-called *tensor networks* (TN). *Tensors* are a generalization of the familiar concepts of scalars, vectors and matrices. Tensor networks are a graph-theoretic interpretation of tensors, expressing them in terms of vertices and edges.

It has turned out that the class of quantum systems that can be simulated efficiently by TN methods are also of interest in the field of solid state physics, for example the simulation of *spin systems*. This allows physicists to determine relevant properties with unprecedented precision. Only recently (April 2009), it was discovered that these methods could also be applied to two-dimensional *fermionic systems*, enlarging the simulatable class to include many systems of chemical interest. These systems were previously thought to be unsimulatable because of the so-called *sign problem* associated with fermions. This has sparked considerable interest for TN methods in the field of quantum chemistry.

In the field of computer science, tensor network methods are of interest because they can be used for the simulation of quantum computations. Quantum computations that can be simulated efficiently on a classical computer are in a sense not genuinely quantum. Sharpening the boundary between classes of computations that can and cannot be simulated efficiently, helps us to determine the genuine quantum elements.

There are some review articles available on tensor network methods [PVWC06, VMC08] and even a recent book on quantum simulation in general [VMH09]. These works either do not incorporate all tensor network based methods, or do not treat them from a computer science perspective. This thesis tries to provide an overview of the tensor network methods from a computer science perspective.

The thesis is organized as follows: after the preliminaries, we will first look at two closely related methods, namely matrix product states and tree tensor networks. The second chapter is the main topic of the thesis: projected entangled pair states. The chapter contains original work on update procedures, construction methods and on the question which classes of states can be represented efficiently. Chapter 3 contains an overview of other tensor network methods, including recent developments. Finally, we will look at some applications of tensor network methods in the field of solid states physics and quantum chemistry. This includes a description of the density-matrix renormalization group method.

The reader is assumed to have some background in (theoretical) computer science, graph theory and linear algebra. The preliminaries of tensor calculus, quantum computing and spin systems will be discussed briefly in the next chapter. Nevertheless, some background in quantum physics and chemistry, or quantum computing and information science will certainly help.

Chapter 1

Preliminaries

In this chapter several concepts will be introduced that lay the grounds for the rest of the thesis. We will start with the basics of *linear algebra* and *tensor algebra*, which will be used in the discussion of *quantum mechanics* and *quantum computing* that follows. The presentation is partly based on [NC00]. The chapter will be closed with an introduction to *spin systems*, physical models for which tensor network methods have proven invaluable. Since tensor network methods were initially developed for this type of simulation, some of the characteristics can be traced back to these systems. For a more thorough introduction to linear and tensor algebra the reader is referred to the book of Isham [Ish89] and for quantum computing to the book of Nielsen and Chuang [NC00] or, for a more concise version, to the paper of Rieffel and Polak [RP00].

1.1 Linear algebra

Linear algebra can be seen, amongst others, as the mathematical language of quantum mechanics. The field of linear algebra studies *vectors spaces* and *linear operations* on vector spaces. In this section we will briefly introduce the basic definitions and concepts of the field.

1.1.1 Vector spaces

Formally, a *vector space* over the field[1] of complex numbers $\mathbb{C}$ is a set V with two binary operators – *vector addition* and *scalar multiplication* – that satisfies a number of axioms. A *vector* is an element of a vector space. Using the *ket*

$$|v\rangle \in V$$

as notation for a vector in a vector space – the *Dirac notation* – is standard in quantum physics. Take for example the n-dimensional vector space $\mathbb{C}^n$. It consist of all n-tuples of complex numbers. Another common notation of vectors is the column matrix notation, with $z_k \in \mathbb{C}$:

$$\begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}.$$

The *addition* of two vectors is defined as

$$\begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} + \begin{bmatrix} z'_1 \\ \vdots \\ z'_n \end{bmatrix} = \begin{bmatrix} z_1 + z'_1 \\ \vdots \\ z_n + z'_n \end{bmatrix}.$$

[1] A *field* is algebraic structure that supports addition, subtraction, multiplication and division and satisfies certain axioms. Another example is the field of *real numbers* $\mathbb{R}$. If we refer to vector spaces, we will mean vector spaces over the field of complex numbers.

The *scalar multiplication* is defined as

$$z \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} zz_1 \\ \vdots \\ zz_n \end{bmatrix}.$$

On the right-hand side of these definitions $z_k + z'_k$ and zz_k are just ordinary addition and multiplication of complex numbers.

A vector space over the field of complex numbers satisfies the following axioms[2]:

- associativity of vector addition

$$\forall |u\rangle, |v\rangle, |w\rangle \in V : |u\rangle + (|v\rangle + |w\rangle) = (|u\rangle + |v\rangle) + |w\rangle$$

- commutativity of vector addition

$$\forall |u\rangle, |v\rangle \in V : |u\rangle + |v\rangle = |v\rangle + |u\rangle$$

- identity element[3] of vector addition

$$\exists 0 \in V : \forall |v\rangle \in V : |v\rangle + 0 = |v\rangle$$

- inverse element of vector addition

$$\forall |v\rangle \in V : \exists |w\rangle \in V : |v\rangle + |w\rangle = 0$$

- distributivity of scalar multiplication with respect to vector addition

$$\forall |v\rangle, |w\rangle \in V, \forall a \in \mathbb{C} : a(|v\rangle + |w\rangle) = a|v\rangle + a|w\rangle$$

- distributivity of scalar multiplication with respect to field addition

$$\forall |v\rangle \in V, \forall a, b \in \mathbb{C} : (a + b)|v\rangle = a|v\rangle + b|v\rangle$$

- compatibility of scalar multiplication with field multiplication

$$\forall |v\rangle \in V, \forall a, b \in \mathbb{C} : a(b|v\rangle) = (ab)|v\rangle$$

- identity element of scalar multiplication

$$\exists 1 \in \mathbb{C} : \forall |v\rangle \in V : 1|v\rangle = |v\rangle$$

A set of vectors $\{|v_1\rangle, \ldots, |v_n\rangle\}$ *span* the vector space V if any vector $|v\rangle \in V$ can be written as linear combination of the vectors in that set, i.e. for $a_i \in \mathbb{C}$ and $|v_i\rangle \in V$

$$|v\rangle = \sum_i a_i |v_i\rangle.$$

This set is called *linearly dependent* if there exists $a_1 \ldots a_n \in \mathbb{C}$ with at least $a_i \neq 0$ for some i, such that $\sum_i a_i |v_i\rangle = 0$. If this only holds if $a_i = 0$ for all i, then the set is *linearly independent*. Each vector space V is spanned by a set of linearly independent vectors: a *basis*. The number of elements in this set is called the *dimension* of V.

[2]Note that the first four axioms define an *Abelian group* under $+$.

[3]Note that the *zero vector* $0 \in V$ is not written as a ket. This is because the notation $|0\rangle$ is conventionally used for a basis vector that we will encounter later on.

1.1.2 Hilbert spaces

Another binary operator, the *inner product* $(\cdot, \cdot)$ that takes two vectors to a scalar, can be defined on a vector space to form an *inner product space*. A binary operator that maps $V \times V$ to $\mathbb{C}$ is an inner product if it has the following properties:

- right-linearity:
$$(|v\rangle, \sum_i \lambda_i |w_i\rangle) = \sum_i \lambda_i (|v\rangle, |w_i\rangle)$$
- conjugate-symmetry:
$$(|v\rangle, |w\rangle) = (|w\rangle, |v\rangle)^*$$
- positive-definiteness:
$$(|v\rangle, |v\rangle) \geq 0 \text{ and real, with equality only if } |v\rangle = 0$$

Here z^* is the complex conjugate of z.

In matrix column notation, the inner product on $\mathbb{C}^n$ is defined as

$$((y_1, \ldots, y_n), (z_1, \ldots, z_n)) = \sum_i y_i^* z_i = \begin{bmatrix} y_1^* & \ldots & y_n^* \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}.$$

The scalar that results from the inner product is independent of the chosen basis. In Dirac notation the inner product of $|v\rangle$ and $|w\rangle$ becomes

$$(|v\rangle, |w\rangle) = \langle v|w\rangle.$$

Two vectors are *orthogonal* if their inner product is zero. On $\mathbb{C}^n$, the inner product allows us to define the *norm* of a vector as

$$|||v\rangle|| \equiv \sqrt{\langle v|v\rangle}.$$

A vector $|v\rangle$ is called a *unit vector* or *normalized* if $|||v\rangle|| = 1$. A set of vectors is called *orthonormal* if each vector is normalized and all vectors are orthogonal.

A *Hilbert space* is a real or complex inner product space with some additional properties. However, for finite dimensional vector spaces, the only vector spaces we will encounter in this thesis, a Hilbert space is identical to an inner product space.

1.1.3 Linear operators

Any function $A : V \to W$ between two vector spaces V and W that is linear in its inputs, i.e.

$$A\left(\sum_i a_i |v_i\rangle\right) = \sum_i a_i A(|v_i\rangle),$$

is called a *linear operator*. A linear operator A can act *on* a single vector space V, i.e. $A : V \to V$. The *identity operator* I_V on vector space V is defined as

$$I_V |v\rangle \equiv |v\rangle$$

for all vectors $|v\rangle \in V$. We will drop V from the subscript if it is clear which vector space we are referring to. Linear operators can be represented by matrices relative to a chosen basis or as *outer product*. For $|v\rangle \in V$ and $|w\rangle \in W$, the outer product $|w\rangle\langle v|$ is defined by

$$(|w\rangle\langle v|)|v'\rangle \equiv |w\rangle(\langle v|v'\rangle) = \langle v|v'\rangle|w\rangle,$$

i.e. $|w\rangle\langle v|$ is a linear operator from V to W.

Eigenvectors and eigenvalues

Given a linear operator A on vector space V, the non-zero vector $|v\rangle \in V$ such that

$$A|v\rangle = \lambda|v\rangle,$$

is the *eigenvector* of A with a corresponding scalar *eigenvalue* λ. The eigenvalues and eigenvectors of an operator A can be found by solving the *characteristic equation*

$$\det(A - \lambda I) = 0$$

for λ, where $\det(B)$ is the *determinant* of matrix B. The different solutions for λ will be the eigenvalues of the operator. The *eigenspace* is the set of eigenvectors corresponding to a particular eigenvalue.

Hermitian and unitary operators

For each linear operator A on Hilbert space V there exists a unique operator $A^\dagger$ – the *adjoint* or *Hermitian conjugate* – such that for any $|v\rangle, |w\rangle \in V$:

$$(|v\rangle, A|w\rangle) = (A^\dagger|v\rangle, |w\rangle).$$

An operator A for which

$$A^\dagger = A$$

holds, is said to be *Hermitian* or *self-adjoint*. In the matrix representation the adjoint of a matrix is its conjugate transpose[4]:

$$A^\dagger \equiv (A^*)^T.$$

The adjoint is also defined on vectors. The adjoint of a ket $|\psi\rangle$ is defined to be a *bra*:

$$\langle\psi| \equiv (|\psi\rangle)^\dagger.$$

A matrix U is *unitary* if

$$UU^\dagger = U^\dagger U = I$$

holds.

Trace

The *trace* of a square matrix A is defined to be the sum of its diagonal elements, i.e.

$$\mathrm{tr}(A) \equiv \sum_i A_{ii},$$

where A_{ii} is the element of the matrix on row i and column i. The trace of a linear operator A is independent of the chosen basis for the matrix representation of the operator.

The following properties hold for the trace:

- cyclicity:
$$\mathrm{tr}(ABC) = \mathrm{tr}(CAB) = \mathrm{tr}(BCA)$$
- linearity:
$$\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(B)$$

[4]The conjugate transpose of an $l \times k$ matrix A is defined as $(A^*)^T \leftrightarrow \left(\left((a_{ij})_{i=1,j=1}^{k,l}\right)^*\right)^T = (a_{ji}^*)_{j=1,i=1}^{l,k}$, where a_{ij} is the element in matrix A on row i and column j.

1.1.4 Singular value decomposition

Matrices can be decomposed using the *singular value decomposition*. Let A be a square matrix. Then there exist unitary matrices U and V, and a diagonal matrix D with non-negative entries such that

$$A = UDV.$$

The diagonal elements of D are called the *singular values* of A.

1.2 Tensor algebra

The notions of *scalar*, *vector* and *matrix* are familiar concepts from linear algebra. A *tensor* is a generalization of these and is closely related to *multi-dimensional arrays* commonly used in programming languages. In this section we will formally define tensors in terms of vector spaces, but also introduce the *index notation* common in physics and the concept of *tensor contraction*. Furthermore, we will discuss a convenient graph theoretic interpretation of tensors in the form of *tensor networks*.

1.2.1 Tensor product spaces

Two n- and m-dimensional Hilbert spaces V and W can be combined to form a larger nm-dimensional vector space $V \otimes W$ using the *tensor product*[5] $\otimes$. By definition, the tensor product of two vectors satisfies the following properties for an arbitrary scalar z and $|v\rangle, |v_1\rangle, |v_2\rangle \in V$ and $|w\rangle, |w_1\rangle, |w_2\rangle \in W$:

- scalar product:
$$z(|v\rangle \otimes |w\rangle) = (z|v\rangle) \otimes |w\rangle = |v\rangle \otimes (z|w\rangle))$$
- right-distributivity over vector addition:
$$(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle$$
- left-distributivity over vector addition:
$$|v\rangle \otimes (|w_2\rangle + |w_2\rangle) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle$$

Elements of a tensor product space are called *tensors* and consist of linear combinations of the tensor product of the elements of vector spaces V and W:

$$V \otimes W = \{\sum_{i=1}^{k} c_i |v_i\rangle \otimes |w_i\rangle | v_1, \ldots, v_k \in V, w_1, \ldots, w_k \in W, c_1, \ldots, c_k \in \mathbb{C}\}.$$

Moreover, if $\{|e_i\rangle\}_i$ and $\{|f_j\rangle\}_j$ are orthonormal bases for V and W, then $\{|e_i\rangle \otimes |f_j\rangle\}_{ij}$ is a basis of $V \otimes W$. Note that the tensor product symbol $\otimes$ is often omitted leading to the abbreviated notations $|v\rangle \otimes |w\rangle \equiv |v\rangle|w\rangle \equiv |vw\rangle$.

Linear operators can also be defined on tensor product spaces. The linear operators A on V and B on W can be tensored to the linear operator $A \otimes B$ on $V \otimes W$, with:

$$(A \otimes B)(\sum_i c_i |v_i\rangle \otimes |w_i\rangle) \equiv \sum_i c_i A|v_i\rangle \otimes B|w_i\rangle.$$

[5] It can be convenient to interpret the tensor product as *Kronecker product*, i.e. for matrices M and N:

$$M \otimes N \leftrightarrow \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \otimes \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} = \begin{bmatrix} m_{11}\begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} & m_{12}\begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} \\ m_{21}\begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} & m_{22}\begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} m_{11}n_{11} & m_{11}n_{12} & m_{12}n_{11} & m_{12}n_{12} \\ m_{11}n_{21} & m_{11}n_{22} & m_{12}n_{21} & m_{12}n_{22} \\ m_{21}n_{11} & m_{21}n_{12} & m_{22}n_{11} & m_{22}n_{12} \\ m_{21}n_{21} & m_{21}n_{22} & m_{22}n_{21} & m_{22}n_{22} \end{bmatrix}$$

In this way the resulting tensor of two matrices can again be interpreted as a larger matrix.

Note that a linear operator does not have to be of this form, i.e. one can also define an arbitrary linear operator $C : V \otimes W \to V' \otimes W'$ that can be represented as

$$C = \sum_i c_i A_i \otimes B_i.$$

The inner product on $V \otimes W$ is defined as

$$\left(\sum_i a_i |v_i\rangle \otimes |w_i\rangle, \sum_j b_j |v'_j\rangle \otimes |w'_j\rangle \right) \equiv \sum_{ij} a_i^* b_j \langle v_i | v'_j \rangle \langle w_i | w'_j \rangle.$$

Finally, the notation

$$|\psi\rangle^{\otimes k} \equiv \underbrace{|\psi\rangle \otimes \ldots \otimes |\psi\rangle}_{k}$$

is used to denote k times the tensor product of $|\psi\rangle$ with itself.

1.2.2 Index notation

As described in the previous section, tensors are elements of a tensor product space. There is another notation for tensors that is common in physics: the *index notation*. Tensors can be represented using an index notation (with respect to a certain basis)

$$[T]^{i_1 \ldots i_n}_{j_1 \ldots j_m} \in V_1 \otimes \ldots \otimes V_n \otimes W_1 \otimes \ldots \otimes W_m,$$

where the indices $i_1 \ldots i_n \in I$ and $j_1 \ldots j_m \in J$ are elements from the index sets I and J. The cardinality of these sets is exactly the dimension of the corresponding vector spaces and is called the *dimension* of an index. If the indices are unbounded, e.g. $[T]^i_j$, we mean the complete tensor, if not, e.g. $[T]^2_7$, we mean a specific entry in the tensor. The index notation can be interpreted as a function $f : I \times \ldots \times I \to F$ that maps tuple of indices to a scalar.

The position of an index of the tensor (i.e. sub- or superscript) has significance when one considers coordinate transformations. However, we will have no need to make this distinction, so the position of the indices will be immaterial.

The *rank* or *order* of a tensor is the number of indices of a tensor. So, a scalar $[c]$ has rank 0, a vector $[v]^i$ rank 1 and a matrix $[M]^i_j$ rank 2. A rank g tensor where each of the indices has dimension D will have D^g entries.

We will illustrate this notation using a rank two tensor $M \in V \otimes W$ in matrix notation:

$$\begin{bmatrix} m_{00} & m_{01} \\ m_{10} & m_{11} \end{bmatrix}.$$

In index notation M is written as $[M]^i_j$ with $i \in I$ and $j \in J$. Using the function interpretation, we see that the upper left entry of the matrix is $[M]^0_0 = m_{00}$. Note that in the notation of the previous section, M corresponds to

$$\sum_{i,j=0}^{1} [M]^i_j |ij\rangle = m_{00}|e_0 f_0\rangle + m_{01}|e_0 f_1\rangle + m_{10}|e_1 f_0\rangle + m_{11}|e_1 f_1\rangle,$$

using the assumption that $\{|e_i f_j\rangle\}_{ij}$ is the basis of the matrix notation.

1.2.3 Tensor contraction

If two tensors have indices with the same dimension, the two tensors can be *contracted* over these indices. Usually this will be indicated by *shared indices*, i.e. two tensors have indices with the same label. The result of the contraction is again a tensor, with rank decreased by 2:

$$\mathcal{C}([A]^{i_1 \ldots i_l}_{j_1 \ldots j_m}, [B]^{j_1 \ldots j_m}_{k_1 \ldots k_n}) \equiv \sum_{j_1 \ldots j_m} [A]^{i_1 \ldots i_l}_{j_1 \ldots j_m} [B]^{j_1 \ldots j_m}_{k_1 \ldots k_n} = [C]^{i_1 \ldots i_l}_{k_1 \ldots k_n}.$$

The *trace* and *matrix product* operations can be expressed in terms of contractions. For the trace we simply contract a rank two tensor over its own indices:

$$\mathrm{tr}(A) = \sum_i [A]^i_i.$$

Compare the notation with the earlier definition of the trace in Section 1.1.3. The matrix product contracts two rank 2 tensors to one rank 2 tensor:

$$[A]^i_j \cdot [B]^j_k = \sum_{j=1} [A]^i_j [B]^j_k = [C]^i_k.$$

When a sequence of more than two tensors has to be contracted, e.g.

$$[D]^i_l = \mathcal{C}([A]^i_j, [B]^j_k, [C]^k_l),$$

the order in which a sequence of tensors is contracted has no influence on the final result. The size of the intermediate tensors – and hence the efficiency of the contraction operation – does.

1.2.4 Tensor networks

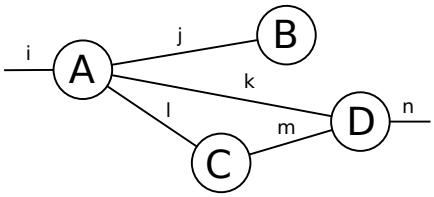

Figure 1.1: The tensor network representation of $[A]^i_{jkl}[B]^j[C]^l_m[D]^{km}_n$.

Tensor networks are a representation of tensors and tensor algebra using the language of graph theory, first introduced by Penrose [Pen71]. A collection of tensors, sharing a number of indices, can be mapped to a graph. Each tensor is represented by a labeled *vertex* or *node* and each index by a labeled *edge*. Indices shared by two tensors are connected by an edge labeled by that index. Indices that are not shared are represented by *open edges* and are again labeled by the index. For example, the tensors

$$[A]^i_{jkl}[B]^j[C]^l_m[D]^{km}_n$$

are represented by the tensor network shown in Figure 1.1.

In the network, contraction of two tensors results in the merge of two nodes along a shared edge. The resulting node inherits the uncontracted edges of the original nodes. See Figure 1.2 for an example of a contraction along index l.

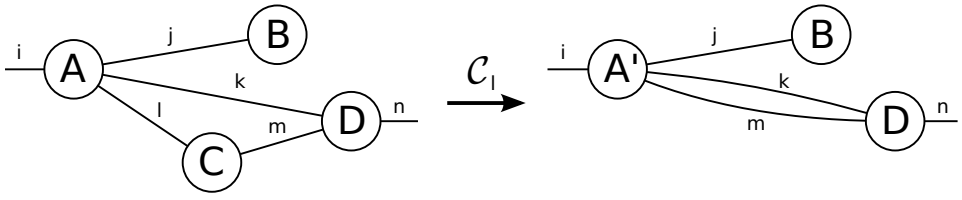

Figure 1.2: Tensor contraction of two nodes in the tensor network.

Mappings

It is possible to map a tensor network with a higher dimensional geometry (e.g. a grid or a tree) to one with a lower dimensional (e.g. a line). Since the number of neighbors is reduced in this map, the resulting graph will have nodes that share indices with non-neighbors. These long-range connections will be distributed over the short-range indices of the intermediate, neighboring nodes. See Figure 1.3 for an example.

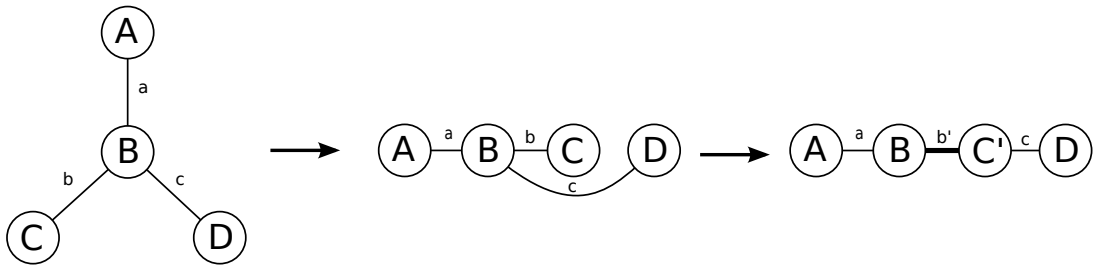

Figure 1.3: The mapping of a tree-like tensor network to a linear tensor network. The long-range index c is distributed over two short-range indices. The thicker line indicates an index with a higher dimension.

To see how this works, consider the following tensors, representing a tree tensor network:

$$[A]_a[B]^a_{bc}[C]^b[D]^c.$$

An additional Kronecker delta[6] can be introduced to separate two indices:

$$[A]_a[B]^a_{b\sigma}\delta^\sigma_c[C]^b[D]^c.$$

The delta-function can be combined with the third tensor :

$$[A]_a[B]^a_{b\sigma}[C']^{b\sigma}_c[D]^c,$$

where $[C']^{b\sigma}_c \equiv \delta^\sigma_c[C]^b$. Combining two indices, the rank of the tensors can be reduced to two:

$$[A]_a[B]^a_{(b\sigma)}[C']^{(b\sigma)}_c[D]^c.$$

The dimension of the combined index will be the product of the dimension of the original indices. Hence, the original tree-like tensor network has been mapped to a linear tensor network, consisting only of matrices and vectors. This mapping does come at a cost. The original tensor network consisted of three rank one tensors and one rank three tensor, each with an index dimension of D. This results in a total number of entries of $3D + D^3$. The linear version of the tensor network, however, consist of two rank one tensors with index dimension D and two rank two tensors, with index dimensions D and D^2, leading to a total number of $2D + 2D^3$ entries.

1.3 Quantum computing

Now that all the necessary mathematical concepts have been introduced, we will introduce the basics of quantum mechanics, the foundation of quantum computing. Moreover, we will introduce *quantum circuits*, a convenient formalism to describe quantum calculations, describe the *quantum teleportation* protocol and introduce the *Schmidt decomposition*, an important tool for the representation methods discussed in the rest of this thesis.

[6] The Kronecker delta tensor δ^i_j is defined as $\delta^i_j = 1$ if $i = j$ and $\delta^i_j = 0$ otherwise. This can be interpreted as a matrix with 1 on the diagonal and 0 elsewhere, i.e. an identity matrix.

1.3.1 Quantum states

Postulate 1. Any isolated physical system is associated with a Hilbert space $\mathcal{H}$, the *state space* of the system. The state of the system is completely described by a unit vector in the state space known as the *state vector* of the system.

The simplest quantum mechanical system is the *qubit*, the quantum analogue to the bit. It has a two-dimensional state space $\mathcal{H}$, spanned by for example the orthonormal basis vectors $|0\rangle$ and $|1\rangle$, the *computational basis*. An arbitrary state vector can be written as

$$|\psi\rangle = a|0\rangle + b|1\rangle,$$

with $a, b \in \mathbb{C}$ and $|a|^2 + |b|^2 = 1$, i.e. the vector is *normalized*.

Postulate 2. The state space of a composite physical system consists of the tensor product of the state spaces of the individual systems. Given n systems in the states $|\psi_1\rangle, \ldots |, \psi_n\rangle$, the state of the composite system will be $|\psi_1\rangle \otimes \ldots \otimes |\psi_n\rangle$.

The state of a composite system consisting of n qubits is now a unit vector in the tensor product of the Hilbert spaces

$$\mathcal{H}_{comp} = \mathcal{H}^{\otimes n}.$$

Since this includes all the *linear combinations* or *superpositions* of the individual qubits, this results in a number of coefficients exponential in n needed to describe this state, i.e.:

$$|\Psi\rangle = \sum_{s_1 \ldots s_n \in \{0,1\}} c_{s_1 \ldots s_n} |s_1 \ldots s_n\rangle.$$

Note that not all composite states are tensor products of single-qubit states. Consider the following state of two qubits that form an *EPR pair*[7]:

$$|\Psi_{\text{EPR}}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

This state cannot be written as the tensor product of two single-qubit states, i.e.:

$$|\Psi_{\text{EPR}}\rangle \neq |\psi_1\rangle \otimes |\psi_2\rangle,$$

for any $|\psi_1\rangle$ and $|\psi_2\rangle$. The qubits are said to be *entangled*. States that *can* be written as a tensor product of single-qubit states are called *product states*, e.g.

$$|\Psi_{prod}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle.$$

Density matrices

An alternative formalism to describe quantum states is the *density operator* or *density matrix* formalism. A quantum state $|\Psi\rangle$ can be described as density matrix ρ using the relation

$$\rho = |\Psi\rangle\langle\Psi|.$$

This formalism is in some cases more convenient to describe concepts in quantum computing, in particular when we don't have full information about the system. A *pure state* is simply an element of $\mathcal{H}$, i.e. we have all information on the state. For the resulting density matrix the property $\text{tr}(\rho^2) = 1$ will hold.

We do not always have full information on the state of a quantum system, but we know that the system is in one of several possible pure states with a certain probability. This *mixed state*

[7] This state is called the *EPR state* or *EPR pair*, after the authors of the paper that studied its strange properties: Einstein, Podolsky and Rosen [EPR35].

can be described by an *ensemble* $\{p_i, |\psi_i\rangle\}$ where the system is in pure state $|\psi_i\rangle$ with probability p_i. The ensemble is expressed in terms of density operators, with pure states ρ_i and probabilities p_i, as

$$\rho = \sum_i p_i \rho_i.$$

For mixed states $\mathrm{tr}(\rho^2) < 1$ holds.

Sometimes we will consider a small part of a larger pure state in isolation. The *reduced density matrix* of subsystem A of a larger system described by ρ^{AB} is defined to be

$$\rho^A \equiv tr_B(\rho^{AB}).$$

Here tr_B is the *partial trace* over subsystem B. The partial trace over a system B of a composite system AB is defined as $\mathrm{tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) \equiv |a_1\rangle\langle a_2|\mathrm{tr}(|b_1\rangle\langle b_2|)$, with $|a_1\rangle, |a_2\rangle$ and $|b_1\rangle, |b_2\rangle$ any two vectors in the state space of A and B respectively. Note that $\mathrm{tr}(|b_1\rangle\langle b_2|) = \mathrm{tr}(\langle b_2|b_1\rangle) = \mathrm{tr}(c_{12})$ with c_{12} a scalar, so the resulting matrix will be of the form $c_{12}|a_1\rangle\langle a_2|$. When the subsystem is entangled with the full system, the subsystem will be in a mixed state.

Qubits vs qu*d*its

So far we have only considered qubits, states in a two dimensional Hilbert space. This is the simplest quantum physical system and the quantum equivalent to the classical bit. There is, however, no reason not to consider states in a three dimensional (*qutrits*) or even d-dimensional Hilbert space: qu*d*its. Most of the results in this thesis can be generalized from qubits to qudits.

1.3.2 Quantum operations

Unitary quantum operations

Postulate 3. The change of the state of a *closed* quantum system in time, i.e. its *evolution*, is described by a *unitary transformation*: if the state of a system is $|\psi\rangle$ at t_1 and $|\psi'\rangle$ at t_2, then $|\psi\rangle$ and $|\psi'\rangle$ are related by a unitary operator U depending only on t_1 and t_2.

Any unitary quantum operation on a n-qubit system can be decomposed into one-qubit and two-qubit operations. Furthermore, it suffices to describe two-qubit operations only on nearest neighbors. The idea is to use the SWAP operation to move the two qubits until they are neighbors. After the operation, the SWAP operation can again be used to move the qubits to their original location. The number of swap operations needed is at most $2(n-1)$, i.e. linear in in the number of qubits.

Quantum measurement

Postulate 4. Measurements on a quantum system are described by a collection of *measurement operators* M_m acting on the state space of the system being measured, where index m indicates the measurement outcome. The probability of a measurement outcome m in a system with state $|\psi\rangle$ is

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle.$$

After the measurement, the system will be in the state

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}.$$

The measurement operators satisfy the *completeness equation*

$$\sum_m M_m^\dagger M_m = I.$$

A *measurement of a qubit in the computational basis* is described by the measurement operators

$$M_0 = |0\rangle\langle 0|; M_1 = |1\rangle\langle 1|.$$

Projective measurements form a special class of measurements that are related to a Hermitian operator $\hat{O}$, the *observable*. The observable can be written as

$$\hat{O} = \sum_m m P_m,$$

with m the eigenvalues of $\hat{O}$ and P_m a projector onto the eigenspace. The *expectation value* of $\hat{O}$ for quantum state $|\psi\rangle$ is

$$\langle \hat{O} \rangle \equiv \langle \psi | \hat{O} | \psi \rangle = \sum_m m \langle \psi | P_m | \psi \rangle.$$

It turns out that any measurement on a quantum system can be described in terms of projective measurements and unitary operations.

Quantum circuits

A convenient formalism to describe quantum operations is the *quantum circuit*. It is related to the classical formalism of the *logical circuit*. Unitary operations are described using *quantum gates* on one or multiple qubits, analogously to the classical *logical gate*. The gates are connected via *quantum wires*, visualizing the transport of quantum information. The final element is the projective measurement in the computational basis, visualized with a *meter symbol*. As discussed in the previous section, any general measurement can be expressed using unitary transformations and a projective measurement in the computational basis.

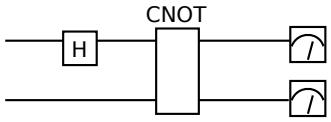

Figure 1.4: An example of a quantum circuit, showing several quantum gates and measurement.

The *Pauli gates* form an important group of quantum operators:

- $\sigma_0 = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ is the *identity* operator, i.e. $I(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle + \beta|1\rangle$
- $\sigma_1 = X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ is the *bit flip* operator, i.e. $X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle$
- $\sigma_2 = Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ is the *imaginary bit-phase flip* operator, i.e. $Y(\alpha|0\rangle + \beta|1\rangle) = \alpha i|1\rangle - \beta i|0\rangle$
- $\sigma_3 = Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ is the *phase flip* operator, i.e. $Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle$

The class of decision problems that can be solved by a quantum computer in polynomial time with an error probability of 1/3 is called *BQP* (Bounded error Quantum Polynomial time). In terms of quantum circuits, this means that there exists a quantum circuit with a number of gates polynomial in the size of the input that solves the problem, if and only if the problem is in BQP.

1.3.3 Quantum teleportation

Quantum teleportation is a procedure to transport a qubit in absence of a quantum channel, using two bits over a classical channel and an EPR-pair. For an illustration of the procedure imagine the following scenario. Alice and Bob each own half of a shared EPR-pair, that is we have a composite two qubit system in the state

$$\frac{|0\rangle_A|0\rangle_B + |1\rangle_A|1\rangle_B}{\sqrt{2}},$$

where the first qubit, labeled with subscript A, is in possession of Alice and Bob has possession of the second qubit, labeled with B. Furthermore, they are able to communicate classically. Now Alice is given a qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and has the task to communicate this state to Bob. That is, we start in the initial situation

$$|\psi\rangle_A \left(\frac{|0\rangle_A|0\rangle_B + |1\rangle_A|1_B\rangle}{\sqrt{2}} \right).$$

Alice now performs a measurement in the *Bell basis*

$$\begin{aligned}
\beta_{00} &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
\beta_{01} &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\
\beta_{10} &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\
\beta_{11} &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle),
\end{aligned}$$

on the qubit that is to be sent and her half of the EPR pair. This results in one of four possible post-measurement states:

- $(\beta_{00})_A \otimes (\alpha|0\rangle + \beta|1\rangle)_B$
- $(\beta_{01})_A \otimes (\beta|0\rangle + \alpha|1\rangle)_B$
- $(\beta_{10})_A \otimes (\alpha|0\rangle - \beta|1\rangle)_B$
- $(\beta_{11})_A \otimes (\beta|0\rangle - \alpha|1\rangle)_B$.

Note how the measurement on Alice's qubits has influenced the state of Bob's qubit, even while they are at different locations. Depending on the measurement outcome, Bob has to do one of four operations on his qubit to obtain state $|\psi\rangle$:

- $(\beta_{00})_A \to I$
- $(\beta_{01})_A \to X$
- $(\beta_{10})_A \to Z$
- $(\beta_{11})_A \to ZX$.

For Bob to know which operation he has to apply, Alice has to communicate her measurement outcome to Bob using classical communication. The final result of the procedure will be

$$(\beta_m)_A \otimes |\psi\rangle_B,$$

where m represents Alice's measurement outcome. Indeed, Alice has transported $|\psi\rangle$ to Bob using only a shared EPR-pair and two bits of information over a classical communication channel.

1.3.4 Schmidt decomposition

Every pure quantum state can be decomposed into a sum of orthogonal bipartite states using the *Schmidt decomposition*. Suppose $|\Psi^{AB}\rangle$ is a pure state of a composite system, AB. Then there exist orthonormal states $|\psi_i^A\rangle$ for system A, and orthonormal states $|\psi_i^B\rangle$ of system B such that

$$|\Psi^{AB}\rangle = \sum_i^{\chi} \lambda_i |\psi_i^A\rangle |\psi_i^B\rangle,$$

where λ_i are non-negative real numbers satisfying $\sum_i \lambda_i^2 = 1$ known as the *Schmidt coefficients*. The number of non-zero coefficients χ is called the *Schmidt rank* or *Schmidt number* of $|\Psi^{AB}\rangle$; the bases $|\psi_i^A\rangle$ and $|\psi_i^B\rangle$ are called *Schmidt vectors* or *Schmidt bases*.
The Schmidt decomposition has several properties that make it useful:

1. The Schmidt coefficients are the eigenvalues of the reduced density matrices of both subsystems.

2. The Schmidt decomposition is invariant under local operations on subsystems A and B, i.e. an operation on qubits within subsystem A or B does not effect the Schmidt rank nor the Schmidt coefficients of a decomposition.

3. The Schmidt rank is a measure for the amount of entanglement between the two subsystems.

1.4 Spin systems

Spin systems are simple physical models used to study the behavior of *spins* of particles. Spin is an intrinsic degree of freedom of elementary particles in physics. For our purposes, we can take spin to be described by an n-level quantum system. In many models the position of the particle is taken to be discrete, i.e. there is a lattice of possible sites. Depending on the model, the particles can migrate to different sites, interact with each other, change their spin and can be influenced by an external force.

In this section we will first look at the properties of spin systems; next, we will distinguish two classes of elementary particles that have spin and illustrate a spin system that has been simulated using tensor network methods in the literature.

1.4.1 Properties of spin systems

A spin system is in a certain *configuration*, that is, all the spins are at a certain site and have a certain value. For each configuration several properties can be calculated, for example the *energy* of the system. The configuration with the lowest energy (the *ground state*) is of special interest. Furthermore the transitions between the configurations can show certain behavior, that is the system can have *phases* and *phase transitions*. All these concepts will be discussed next.

Energy

One of the most important properties of a physical system is its energy, also for spin systems. The energy of a certain configuration of a system is described by the *Hamiltonian*. The Hamiltonian H models the different forces working on a system, including neighbor and environment interactions, as well as the intrinsic energies of the system, like the kinetic energy. The Hamiltonian in combination with the topology fully describes the physics of a spin system.

For physicists, it is interesting to understand the relation between possible configurations and the energy of a system. This includes questions like whether the possible energies are discrete or continuous, whether they are evenly distributed or gapped and how the transition between different configurations occur. Things get even more interesting when certain other physical properties like magnetism and conductivity depend on the configurations of the spins. Determining the relation

between the configuration and the energy is not an easy problem for non-trivial systems, and systems consisting only of a few qubits can already be non-trivial.

Ground state

Since all systems in nature seek to minimize their energy, spin systems will tend toward the configuration in which their energy is minimal. This lowest energy state is called the *ground state*. Determining the ground state of a system is in general a hard problem.

Not all systems have a simple ground state, i.e. a ground state that can be described in terms of the states of the individual spins. Imagine for example a triangular system, with a spin on each corner of the triangle. In many spin models, neighboring spins need to be anti-parallel, e.g. have the opposite values, to avoid an energy penalty. However, in this triangular system, it is not possible to satisfy this condition for all spins, so the simple ground state in which all spins are anti-parallel is not possible. These systems are called *frustrated spin systems*.

Phase transitions

A *phase* is a certain homogeneous configuration of the system for a certain energy regime. The phase of a system can change abruptly at a certain energy threshold. This sudden change configuration is called a *phase transition*.

Systems that are close to a phases transition are said to be *critical*. The behavior of these systems is more difficult to describe than systems in a non-critical situation, i.e. well away from a phase transition.

1.4.2 Bosons and fermions

Based on the values the spin can take, there are several classes of particles in physics, including *bosons* and *fermions*. The most important example of fermions are electrons. The main difference between the two classes is that fermions have to obey the *Pauli exclusion principle*: no two fermions can be in exactly the same state. In other words, there can be no two identical fermions with the same spin on one site. This can be expressed by

$$|\Psi(\alpha, \beta)\rangle = -|\Psi(\beta, \alpha)\rangle,$$

whereas for bosons

$$|\Psi(\alpha, \beta)\rangle = |\Psi(\beta, \alpha)\rangle.$$

Here Ψ is a certain *wave function*[8] describing the state of the system, and α and β are two particles in that wave function. In other words, the wave function has to be anti-symmetric under the exchange of fermions α and β. This gives rise to the so-called *sign problem* when simulating spin systems: for many simulation methods this restriction is difficult to take into account and the minus sign can lead to incorrect results.

1.4.3 Example of a spin system

To illustrate which kind of physical systems can be simulated with tensor network methods, the Ising model is shown below. Other spin systems for which tensor network methods have been applied in the literature include the Hubbard, Bose-Hubbard and XY models.

[8] A wave function is function from several variables, that specify the system, to the probability amplitude. It is the function that describes the state in the now possibly infinite Hilbert space.

Quantum Ising model

The Hamiltonian of the Quantum Ising model is given by

$$H = -\tfrac{1}{2} \sum_{<i,j>} J_{i,j}\sigma_i\sigma_j - h_i\sigma_i,$$

where

$< i, j >$	are all nearest neighboring spins i and j,
$J_{i,j}$	is the *coupling constant* of spin i and j, e.g. the interaction between the spins,
σ_i	is the *spin operator* of spin i, that corresponds to a Pauli matrix
h_i	is the interaction of an external magnetic field on spin i.

In other words, the energy of the spin system is determined by a nearest neighbor interaction (a spin-spin coupling) and an interaction with an external field. The *coupling constant* can be both positive and negative, leading to different behavior of the spins. If the constant is positive, than the configuration where neighboring spins have the same value (i.e. *parallel*) will have a lower energy, leading to a *ferromagnetic* phase. If the constant is negative, the configuration with where neighboring spins have opposite values (i.e. *antiparallel*) will have a lower energy, leading to an *antiferromagnetic* phase. Of course, this behavior can be influenced by the interaction with the external field, which can force the spins to a certain value if large enough.

The Ising model was originally developed as a model for magnetism, but has also been applied to model atoms in the gas phase, neural networks and spin glasses (a system where the spins are "frozen"). Note that there is no term for changing the position of the spins, so the spin location is fixed in this model.

Chapter 2

Matrix product states and tree tensor networks

In this chapter we describe two tensor network methods that provide a way to classically represent a class of quantum systems efficiently and also allow for the classical simulation of unitary evolution and measurement: the matrix product state (MPS) formalism, and their extension to trees in the form of tree tensor networks (TTNs).

2.1 Matrix product states

In the *matrix product state* formalism a quantum state is expressed using a product of matrices. The idea is to associate each qudit of the state with a single matrix.

The MPS formalism was first introduced by Fannes, Nachtergaele and Werner as "finitely correlated states" [FNW92] in 1992. They were further investigated and extended by Östlund and Rommer [ÖR95]. There is a historic relation with the AKLT model by Affleck, Kennedy, Lieb and Tasaki [AKLT88] published in 1988, where these states occur naturally as the ground state of the spin-1 spin chain Hamiltonian. Since then, their use as a general representation method – the Matrix Product Representation (MPR) – has been further investigated by Vidal [Vid03], Jozsa [Joz06] and many more.

First the structure of matrix product states will be discussed and a method to construct an MPS for an arbitrary quantum state will be presented. The procedure to update an MPS after unitary operations and measurement will be discussed next, followed by an interpretation of the MPS formalism as a tensor network.

2.1.1 Structure

Product states of n qu*d*its can be efficiently represented classically with at most dn scalar coefficients:

$$|\Phi\rangle = \sum_{s_1 \dots s_n}^{d} [c_1]^{s_1} \cdot \ldots \cdot [c_n]^{s_n} |s_1 \dots s_n\rangle,$$

where $|s_1 \dots s_n\rangle$ is an n-qu*d*it computational basis vector and $[c_k]^{s_k}$ is the vector with the d scalar coefficients of qubit k. In the matrix product formalism, these scalar coefficients are generalized to matrices:

$$|\Psi\rangle = \sum_{s_1 \dots s_n}^{d} [A_1]^{s_1} \dots [A_n]^{s_n} |s_1 \dots s_n\rangle,$$

where $[A_k]^{s_k}$ are the d matrices for qubit k.

Contrary to product states, any state can be represented as matrix product state, as long as the size of the matrices is large enough, in general exponential in the number of qudits. The advantage is, however, that an interesting class of entangled states *can* be represented by matrices of only polynomial size. Note that the class of product states is recovered for matrices of size 1×1.

For each qu*d*it, the matrix product representation consists of d matrices of dimension $D \times D$. The dimension of the matrices depends on the amount of entanglement within the quantum state, as well as on the order in which the qubits are represented. From the construction method described below it will follow that for a particular order

$$D = \chi_{max},$$

where χ_{max} is the maximum *Schmidt rank* of all the sequential *Schmidt decompositions* (see Section 1.3.4) along the ordering of the quantum system. Note that choosing the optimal ordering is not an easy problem.

In fact, the efficiency of the representation can be further improved by allowing a different dimension for each matrix, depending on the entanglement of that particular qu*d*it. The only condition is imposed by the matrix product: the dimension of adjacent matrices has to be compatible.

Representation costs Using the assumption that all the matrices have the same dimension $D \times D$, a n-qu*d*it state can be represented by ndD^2 entries.

2.1.2 Construction

In order for a representation method to be useful, we need an efficient method to construct the representation from an arbitrary quantum state. A procedure that uses an iterative Schmidt decomposition has been proposed by Vidal in [Vid03]. This procedure allows any quantum state to be described as matrix product state, given large enough matrices. Note that in general these matrices will be of exponential size, so not all quantum states have an efficient classical representation.

The procedure consists of the following steps:

1. Schmidt decompose the state according to $1|2...n$[1]

$$|\Psi\rangle = \sum_{\alpha_1}^{\chi_1} \lambda_{\alpha_1} |\psi_{\alpha_1}^{[1]}\rangle |\psi_{\alpha_1}^{[2...n]}\rangle$$

2. Expand the first part in the computational basis

$$|\Psi\rangle = \sum_{\alpha_1}^{\chi_1} \sum_{s_1}^{d} \lambda_{\alpha_1} \Gamma_{\alpha_1}^{[1]s_1} |s_1\rangle |\psi^{[2...n]}\rangle$$

3. Expand $|\psi_{\alpha_1}^{[2...n]}\rangle$ in a local basis for qudit 2

$$|\psi_{\alpha_1}^{[2...n]}\rangle = \sum_{s_2}^{d} |s_2\rangle |\tau_{\alpha_1 s_2}^{[3...n]}\rangle$$

4. Express $|\tau_{\alpha_1 s_2}^{[3...n]}\rangle$ in terms of $|\psi_{\alpha_2}^{[3...n]}\rangle$

$$|\tau_{\alpha_1 s_2}^{[3...n]}\rangle = \sum_{\alpha_2}^{\chi_2} \Gamma_{\alpha_1 \alpha_2}^{[2]s_2} \lambda_{\alpha_2}^{[2]} |\psi_{\alpha_2}^{[3...n]}\rangle$$

[1] This notation indicates a split between the first qubit and the rest of the system.

5. Substitute in the previous equations

$$|\Psi\rangle = \sum_{\alpha_1\alpha_2}^{\chi_1\chi_2} \sum_{s_1 s_2}^{d} \Gamma^{[1]s_1}_{\alpha_1} \lambda^{[1]}_{\alpha_1} \Gamma^{[2]s_2}_{\alpha_1\alpha_2} \lambda^{[2]}_{\alpha_2} |s_1 s_2\rangle |\psi^{[3\ldots n]}_{\alpha_2}\rangle$$

6. Repeat steps 3, 4 and 5 for all Schmidt bases $|\psi^{[l\ldots n]}_{\alpha_{l-1}}\rangle$

$$|\Psi\rangle = \sum_{\alpha_1\ldots\alpha_{n-1}}^{\chi_1\cdots\chi_{n-1}} \sum_{s_1\ldots s_n}^{d} \Gamma^{[1]s_1}_{\alpha_1} \lambda^{[1]}_{\alpha_1} \ldots \Gamma^{[n]s_n}_{\alpha_{n-1}} |s_1 \ldots s_n\rangle$$

A detailed explanation of the steps:

Step 1: Prior to the first step, the qudits are ordered. This numbering defines the order in which the qudits are processed in the construction procedure. While the ordering is arbitrary, it influences the size of the generated matrices and hence on efficiency of the representation. Choosing the optimal ordering is in general a hard problem, but a good guideline is to minimize the distance between two qudits that are entangled.

According to the *Schmidt decomposition* (Section 1.3.4), every state can be described as a sum of the tensor product of two bases times a coefficient. In the first step the system is decomposed in the first qudit and the rest of the system, resulting in χ_1 bases and coefficients. Note that for this first decomposition the Schmidt rank is at most d.

Step 2: Every state can be expanded in a basis such as the computational basis. This is done for all $|\psi^{[1]}_{\alpha_1}\rangle$ and results in d vectors of coefficients $\Gamma^{[1]s_1}_{\alpha_1}$, where α_1 is the index of the different Schmidt bases and s_1 the index of the computational basis.

Step 3: The bases of the second system are expanded in a local basis of qudit 2, i.e. each vector $|\psi^{[2\ldots n]}_{\alpha_1}\rangle$ is split in the tensor product of $|s_2\rangle$ and a vector of the other qudits $|\tau^{[3\ldots n]}_{\alpha_1 s_2}\rangle$, for every value of s_2. This can be achieved by expanding the state in the computational basis and then regrouping the qudits. In general this will result in an unnormalized vector $|\tau^{[3\ldots n]}_{\alpha_1 s_2}\rangle$.

Step 4: $|\tau^{[3\ldots n]}_{\alpha_1 s_2}\rangle$ is again expressed in Schmidt vectors and the corresponding Schmidt coefficients. These vectors can be obtained from the reduced density matrix $\rho^{[3\ldots n]}$. An additional coefficient matrix $\Gamma^{[2]s_2}_{\alpha_1\alpha_2}$ is introduced to map the Schmidt vectors to the local basis.

2.1.3 Quantum operations

Another desired property of a representation method is the ability to update the representation directly after unitary operations and measurement. As stated earlier (Section 1.3.2), it suffices to only describe procedures for one-qubit and two-qubit operations on nearest neighbors.

One qubit

Because of the properties of the Schmidt decomposition, the coefficients of the decomposition will not change under a local operation (Section 1.3.4). So, only the matrix of the local qubit has to be updated. The new matrix can be updated in $\mathcal{O}(\chi^2)$ steps, as explained in [Vid03].

Two qubits

As shown in [Vid03], an MPS can be updated in $\mathcal{O}(\chi^3)$ steps. Only the matrices corresponding to both qubits involved in the operation have to be updated. The update procedure consists of a new Schmidt decomposition between qubits $1\ldots k|l\ldots n$, where k and l are the two neighboring qubits the quantum operation has acted on.

Measurement

Again, due to the properties of the Schmidt decomposition, the probability of a certain measurement outcome of a qubit can be calculated easily from the MPR. Due to the Schmidt decomposition, the following property always holds:

$$|\Psi\rangle = \sum_{\alpha\beta} [A]^{s_l}_{\alpha\beta} |\psi_\alpha^{[1\ldots k]}\rangle |s_l\rangle |\psi_\beta^{[m\ldots n]}\rangle,$$

with $[A]^{s_l}_{\alpha\beta} = \lambda_\alpha^{[l-1]} \Gamma^{[l]s_l} \lambda_\beta^{[l]}$. Because $|\psi_\alpha^{[1\ldots k]}\rangle$ and $|\psi_\beta^{[m\ldots n]}\rangle$ are orthogonal, it is easy to see that the probability of outcome s_l is equal to:

$$p(s_l) = \sum_{\alpha\beta} |[A]^{s_l}_{\alpha\beta}|^2,$$

and can be calculated in $\mathcal{O}(\chi^2)$ steps.

Note This only holds for MPSs constructed and maintained using the Schmidt decomposition, i.e. using the procedures describe above. For other construction and update methods, the stated property does not necessary hold and the procedure will take more steps (see Section 2.1.4).

To calculate the post-measurement state, all the matrices of the MPS have to be updated. This can be done in $\mathcal{O}(n\chi^4)$ steps, as shown in [YS06].

2.1.4 Matrix product states as tensor networks

The MPS can be interpreted as a tensor network. For that consider the tensor network depicted in Figure 2.1a. Each qu*d*it of a quantum state can be associated with a rank 3 tensor (except for the first and last qudit, which are rank 2), with two D-dimensional indices shared with the neighboring tensors, the *virtual indices*, and one d-dimensional open index, the *physical index* (see Figure 2.1b).

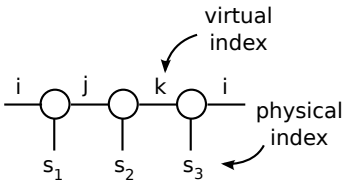

Figure 2.1: A tensor network interpretation of an MPS.

Operations on the MPS can also be interpreted in the tensor network picture. A one-qubit operation consists of the contraction of a tensor representing a unitary matrix and the qubit tensor along the physical index. A two-qubit operation contracts the two qubit tensors with a two-qubit operation tensor. To restore the original structure of the MPS, the resulting tensor has to be split in two separate tensors. This can be achieved using the *singular value decomposition*.

Since $p(i) = \langle\Psi|M_i^\dagger M_i|\Psi\rangle$, the probability $p(i)$ for outcome i of measurement operator M_i can be determined by the contraction of the corresponding tensor network. The idea is to first update the MPS with operation M_i and to the take the inner product of the MPS with itself. See the next section for a method to calculate the inner product of an MPS.

Inner product The procedure to calculate the *inner product* of two MPSs can also be interpreted in the tensor network representation: contraction of the tensor network resulting from connecting the tensor network of the adjoint MPS with the original MPS on the physical indices. See Figure 2.2 for an illustration.

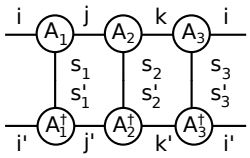

Figure 2.2: The tensor network resulting by the connection of the adjoint MPS with the original MPS on the physical indices. The contraction of this network results in the inner product of the state represented by the MPS with itself.

2.2 Tree tensor networks

In the last section above, the MPS was interpreted as tensor network. Due to the matrix structure – a rank two tensor – this resulted in a linear network. In this tensor network picture it is easy to imagine a generalization to more complex tensor networks. The most straightforward generalization of the linear structure of MPSs are trees. Several versions of these *tree tensor networks* (TTNs) were introduced in the literature, amongst others: Shi *et al.* in [SDV05], Tagliacozzo *et al.* in [TEV09] and Murg *et al.* in [MLNV10].

First the structure of TTNs will be described, after which the construction of and operations on TTNs will be discussed. The relation between TTN and MPS will be examined next, and the section will be completed with brief look at trees with a small number of cycles.

2.2.1 Structure

As the name suggests, the formalism is based on tree-like tensor networks. In this context a *tree* is a *connected* tensor network without *cycles*. For simplicity, we will mainly discuss regular trees where each node – except for the *leaves* – has the same number of neighbors, i.e. the same *degree* or *coordination number*. See Figure 2.3 for an example of these so-called *Cayley trees*.

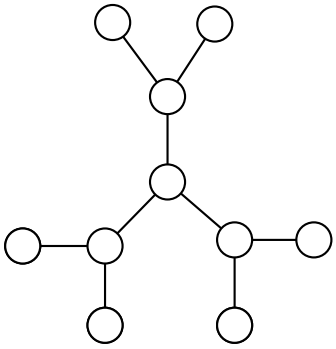

Figure 2.3: An example of a Cayley tree, that can be used as tree tensor network.

A TTN state can be expressed as a state-vector using the following relation:

$$|\Psi\rangle = \sum_{s_1 \dots s_n}^{d} \mathcal{C}([T_1]^{s_1}, \dots, [T_n]^{s_n}) |s_1 \dots s_n\rangle,$$

where $\mathcal{C}$ is the contraction function over the virtual indices of the tensors $[T_k]^{s_k}$.

Note that the rank of the tensor node depends on the degree of the network, i.e. a TTN where each node has three neighbors uses tensors of rank at least 3. Depending on the implementation of the TTN, all nodes have the same degree [SDV05], all nodes have a physical index, increasing the rank by one [MLNV10], or only the leaves have physical indices [TEV09]. Nodes without physical index are called *ancilla nodes*.

Representation costs To store all the coefficients in this structure for n qu*d*its, ndD^g entries have to be stored.

2.2.2 Construction

There is no construction method known from literature for tree tensor networks. However, a state represented as an MPS can be reshaped to a tree (see Section 2.2.4) and our proposed construction method of the PEPS formalism, as discussed in Section 3, can also be applied to trees.

2.2.3 Quantum operations

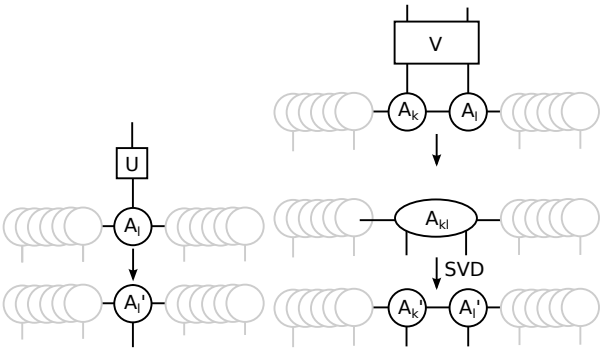

Figure 2.4: The application of a one-qubit operator (a) and a two-qubit operator (b) on a TTN.

One-qubit

A one-qubit operation can be directly applied to a node, absorbing the tensor describing the quantum operation (see Figure 2.4a). Only the tensor of that node has to be updated. This can be interpreted as contraction of the operation tensor with the qudit tensor node.

Two-qubit

To update the TTN after a two-qubit operation, two qudit tensors have to be contracted with a tensor node describing the operation (see Figure 2.4b). The resulting tensor node has to be split again to restore the original tree structure. This can be done using singular value decomposition, resulting in a total $\mathcal{O}(d^3\chi^3)$ steps [SDV05].

Measurement

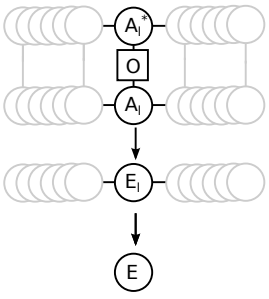

Figure 2.5: Calculating the expectation value of an observable using the contraction of a TTN.

To calculate expectation values on the tree, the tree has to be connected with its adjoint via the tensor nodes of local observables and contracted (see Figure 2.5). Note that the result of the contraction will not depend on the order in which the nodes are contracted, however, the efficiency of the contraction procedure does. The first step is to contract the tree with its adjoint along the physical indices, resulting in a tree without physical indices. The next step is to contract this tree. The most efficient way to do this, is starting at the leaves, working inwards to the root of the tree. In this way, the rank and dimensions of the tensors will remain constant, so the expectation value can be calculated exactly and efficiently. To see this, note that the leaves are rank 1 tensors. Contracting g of these tensors (depending on the degree g of the tree) with the rank $g+1$ parent node results in rank 1 tensor.

2.2.4 Relation with MPS

TTNs are a straightforward extension of MPSs and they can easily be mapped to each other, as described by Verstraete *et al.* [VMC08]. The costs of the mappings give an indication of the different classes of quantum states that can be represented efficiently by either method.

Mapping an MPS to a tree

Using the procedure described in [VMC08], it is possible to map an MPS to a TTN. For convenience, we start from an MPS with *cyclic boundary conditions*, i.e. the first node is connected to the last node[2]. The virtual bonds can be reshaped in a tree-like form in two ways, as illustrated in Figure 2.7. Using the mapping method for tensor networks (Section 1.2.4), the bonds can be redistributed over the nodes. Note that in the b method, ancilla nodes are introduced.

Using this method, the virtual dimension D will be squared. This method can also be used to build trees of a higher degree. By adding virtual tensors at the correct locations, we obtain a TTN with virtual dimension D^g, with g the degree of the nodes in the tree.

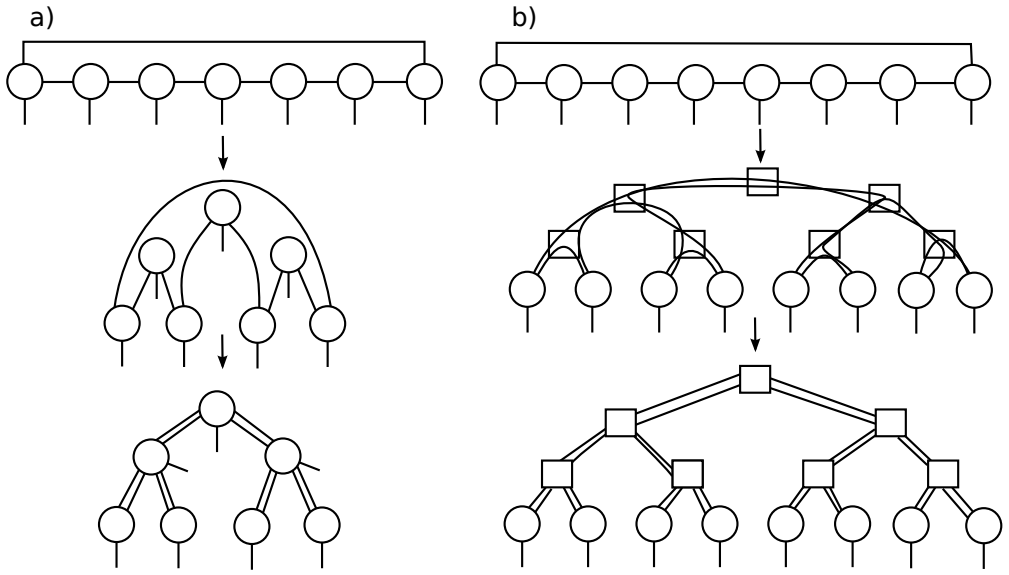

Figure 2.6: Mapping an MPS to two types of TTNs, without (a) and with ancilla nodes (b). The ancilla nodes (indicated by squares) do not have a physical index and hence do not correspond with a physical degree of freedom.

[2]Note that for the non-cyclic case, the resulting tree would both have bonds with different virtual dimension (D and D^2).

Mapping a TTN to an MPS

To map a TTN to an MPS, the tree has to be flattened, as illustrated in Figure 2.7. Again, using the mapping of tensor networks (Section 1.2.4) the virtual bonds can be redistributed over the nodes.

In a TTN of n nodes of degree g the maximum distance between two nodes is $2\log_{(g-1)} n + 1$. For an MPS, the maximum distance between two nodes is n. This imposes an upper bound on the virtual bond dimension of $D^{\log_{(g-1)} n} = n^{\log_{(g-1)} D}$, which is still polynomial.

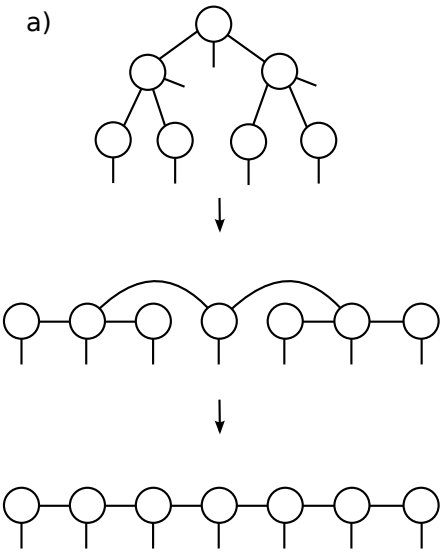

Figure 2.7: A TTN can be flattened to an MPS.

2.2.5 Enhancements

Trees with a small number of cycles

It turns out that tree-like tensor networks with a small number of cycles can still be simulated efficiently. However, for the methods described above to be applicable, the network has to be converted to an equivalent network without cycles. The key idea is to contract the nodes that are part of the cycle in an efficient way. The problem of mapping a graph to a tree is closely related to the graph-theoretic concept of *tree width*. This measure indicates how close a graph is to a tree. It will return later on in the context of *projected entangled pairs states* and the *contracting tensor network formalism*.

Chapter 3

Projected entangled pair states

While MPSs and TTNs prove to be useful representation methods which describe an interesting class of states and operations efficiently, they have limitations. Only systems with a linear or tree-like geometry[1] will have a polynomial small representation. However, states with for example grid-like geometry cannot be described efficiently using this method, as we will see in Section 3.4.

The problem was solved by Verstraete *et al.* with the concept of *projected entangled pair states* (PEPSs) [VC04a]. This formalism allows for an easy generalization to arbitrary geometries and can represent any quantum state. The idea is to use a method based on a virtual layer of entangled pairs between sites and projectors that map the virtual layer to a physical state. This finer underlying structure is the key to the power of PEPS.

First the structure of PEPSs will be discussed in detail. Furthermore, we will describe how to update a PEPS after quantum operations and how to perform measurements on qudits. Next, another perspective on PEPS will be presented: the teleportation picture. Based on an earlier argument to show that any state can be represented as a PEPS, we will show a novel, more efficient construction method and express quantum operations in this picture. PEPSs allow for efficient representations for a larger class of states than MPSs and TTNs. The relation between the classes of states that can be represented efficiently and the geometry will be investigated, as well as the change in space and time complexity of the data structure and operations. We will end the chapter with several enhancements of the formalism.

3.1 Basics

3.1.1 Structure

Like the name suggests, PEPSs represent states in terms of *projections* or, more precisely, *maps* from entangled pairs to a physical quantum state. The entangled pairs are ordered in a certain geometry, best visualized using the graph-like structure of a tensor network.

Our main interest is the class of states that cannot be efficiently represented by an MPS. The most common example of such a state is a 2D grid of qubits, where each qubit is entangled with its nearest neighbours. To view this structure as a PEPS, imagine a 2D grid graph with a node for each qubit (see Figure 3.1a). Each node or *site* represents a *projector* mapping from the underlying *virtual* structure to the *physical* quantum state. Each site has an open *physical index* of dimension d that represents the *physical degree of freedom*, for example a qubit for $d = 2$. The entanglement between the sites is represented by *virtual indices* or *bonds* with *virtual bond dimension* D. These bonds consists of *entangled pairs* $|\phi\rangle = \frac{1}{\sqrt{D}} \sum_{i=0}^{D-1} |i\rangle|i\rangle$ shared by two neighbouring sites. The different elements of the node are illustrated in Figure 3.1b.

[1] The term *geometry* in this context expresses the structure of the entanglement in the quantum system. A linear system with nearest neighbor entanglement is said to have a *linear* or *1D* geometry. A system consisting of a 2D grid with nearest neighbor entanglement will have a *2D* geometry, since every qudit now has four nearest neighbors. Usually, this geometry also corresponds with the structure of a physical system, e.g. an array of physical qubits.

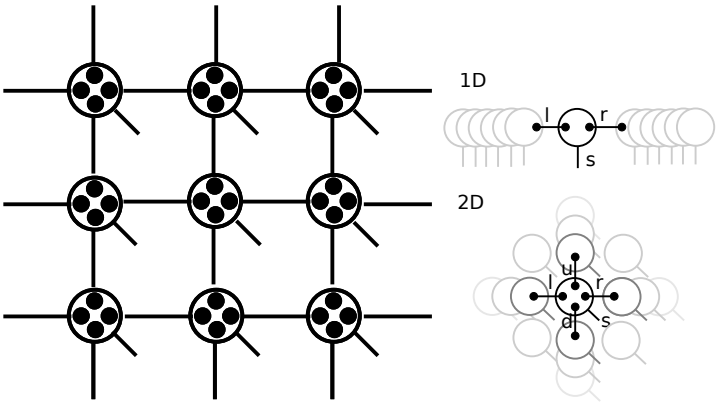

Figure 3.1: (a) The structure of 2D grid PEPS; (b) one node in the PEPS in the 1D linear and 2D grid case.

This construction can be made more formal using the following equation:

$$|\Psi\rangle = P_1 \otimes \ldots \otimes P_n |\phi\rangle^{\otimes n}.$$

Here, $|\Psi\rangle$ is the quantum state, $\{P_i\}_{i=1}^n$ are the projectors and $|\phi\rangle$ is an entangled pair. While not directly clear from the formula, the projectors act on two halves of two distinct entangled pairs, not on one complete pair. Analogous to MPSs, the original state can also be retrieved by contraction of the coefficient tensors of the projectors:

$$|\Psi\rangle = \sum_{s_1 \ldots s_n}^{d} \mathcal{C}([A_1]^{s_1}, \ldots, [A_n]^{s_n}) |s_1 \ldots s_n\rangle.$$

Note Any geometry can be built using entangled pairs. However, unless stated otherwise, the 2D grid geometry is assumed from now on.

Entangled pairs

The entangled pairs are chosen to be *maximally entangled* pairs:

$$\frac{1}{\sqrt{D}} \sum_{i=0}^{D-1} |i\rangle|i\rangle.$$

The dimension D of the entangled pair can be implemented as either an entangled qu$D$it pair or, equivalently, $\log_2 D$ entangled qubit pairs (Section 1.3.1). The virtual dimension D depends on the amount of entanglement between the parts of the system, as we will see later on.

How many pairs are necessary to encode an n-qud*i*t state? That, of course, depends on the geometry of the PEPS. Furthermore, the boundary conditions have to be taken into account. For the 1D case at most $d^{\lfloor \frac{n}{2} \rfloor}$ pairs are necessary to describe an arbitrary state. This follows from the teleportation construction method described below (Section 3.2). It turns out that this is the worst-case scenario 3.2.3 for an example.

Note In general each virtual bond can have an individual dimension. However, for the sake of simplicity, we assume that all virtual bond dimensions are equal to D.

Projectors

The projectors map the virtual space of the entangled pairs to the physical space of the qu*d*its. The projector acting on site x is given by:

$$P_x = \sum_{s_x}^{d} \sum_{\alpha\beta}^{D} [A_x]^{s_x}_{\alpha\beta} |s_x\rangle\langle\alpha\beta|,$$

in which d and D are the physical and virtual bond dimension respectively; $[A_x]^{s_x}_{\alpha\beta}$ is a tensor and $|s_x\rangle$ is a basis of the physical state of site x. The subscript inside the square brackets is just a label and indicates the site the tensor corresponds to; the indices outside square brackets are the actual tensor indices. The bra $\langle\alpha\beta|$ acts on the underlying halves of entangled pairs. Note that the projector has two virtual indices, α and β. This means that this is a projector for a 1D linear PEPS or a corner of a 2D grid PEPS. Since the number of virtual indices depends on the geometry of the graph, we usually summarize all virtual indices with the set $\mathcal{V}$:

$$P_x = \sum_{s_x}^{d} \sum_{\mathcal{V}}^{D} [A_x]^{s_x}_{\mathcal{V}} |s_x\rangle\langle\mathcal{V}|.$$

Note They are not projectors in the sense of projective measurements, e.g. $P_x P_x \neq P_x$. However, they can be interpreted as element of a complete measurement with *post-selection*. Post-selection is the principle that one can do a measurement and select a specific measurement outcome, in contrast to regular measurements where the outcome is probabilistic. It has been shown that adding measurements with post-selection result in more powerful quantum computers.

Representation costs

Representing an n-qu*d*it state as a regular[2] PEPS, n tensors of dimension dD^g have to be stored. For each of the d states of the qu*d*it, a tensor of D^g has to be stored. Here, g is the number of virtual indices of the tensor and hence depends on the geometry of the graph (e.g. it is the *degree* or *coordination number* of the nodes: 4 for a 2D grid, excluding the boundary nodes). The dimension of the virtual indices D depends on the amount and distribution of entanglement in the quantum system being represented.

3.1.2 Construction

Every MPS can be interpreted as a PEPS by adding the required substructure, i.e. entangled pairs and projectors. It turns out that the matrices of the MPS are exactly the matrices of the projectors. For example, for the MPS

$$\sum_{s_1 \ldots s_n} [A]^{s_1}_{\alpha} [A]^{s_2}_{\alpha\beta} \ldots [A]^{s_n}_{\zeta} |s_1 \ldots s_n\rangle$$

the projectors of the corresponding PEPS will have the form

$$P_i = \sum_{s_x} \sum_{\mathcal{V}} [A]^{s_x}_{\mathcal{V}} |s_x\rangle\langle\mathcal{V}|.$$

This relation between MPS and PEPS provides a construction method to convert an arbitrary quantum state to a PEPS by first constructing the MPS and then adding the substructure. However, there exists a more direct way using teleportation, as described in [VPC04b] and [VPC04a]. In Section 3.2 we will describe this method in detail and propose a number of extensions.

[2] In this context, *regular* means that all the nodes in the PEPS have the same degree and the same virtual dimension.

3.1.3 Quantum operations

Here we will describe how quantum operations can be interpreted in the PEPS picture, making it possible to update the representation. We will discuss the procedure to update a PEPS after one-qubit and two-qubit operations on nearest neighbors, since all unitary operations can be described in terms of these. Furthermore, we will discuss measurements on PEPSs.

One-qubit operations

Suppose we want to do a unitary operation on qubit l. How can we update the PEPS representation after the operation? The answer is to update a projector with a unitary operation. To simplify the derivation of the new projector, note that any one-qubit unitary operator can be written as follows:

$$U = \sum_{ss'} [U]_s^{s'} |s'\rangle\langle s|,$$

where s is the row and s' the column of the matrix interpretation of U. The new projector now becomes:

$$\begin{aligned} P' &= UP \\ &= U \sum_{s}^{d} \sum_{\mathcal{V}}^{D} [A]_{\mathcal{V}}^{s} |s\rangle\langle\mathcal{V}| \\ &= \sum_{s'}^{d} \sum_{\mathcal{V}}^{D} [A']_{\mathcal{V}}^{s'} |s'\rangle\langle\mathcal{V}| \end{aligned},$$

with $[A']_{\mathcal{V}}^{s'} = \sum_s^d U_s^{s'} [A]_{\mathcal{V}}^{s}$.

Two-qubit operations

The update of a PEPS after a two-qubit operator V is a bit more involved, since it now acts on two projectors $P_A \otimes P_B$. Again, the operator can be rewritten as

$$V = \sum_{sts't'} V_{st}^{s't'} |s't'\rangle\langle st|.$$

The tensor product of the two projectors can be written as

$$P_A \otimes P_B = \sum_{st}^{d} \sum_{\mathcal{VW}}^{D} [A]_{\mathcal{V}}^{s} [B]_{\mathcal{W}}^{t} |st\rangle\langle\mathcal{VW}|.$$

Now the unitary can be incorporated in the combined two-site projector:

$$P'_{AB} = UP_{AB} = \sum_{s't'}^{d} \sum_{\mathcal{VW}}^{D} [C']_{\mathcal{VW}}^{s't'} |s't'\rangle\langle\mathcal{VW}|,$$

with $[C']_{\mathcal{VW}}^{s't'} = \sum_{st}^{d} V_{st}^{s't'} [A]_{\mathcal{V}}^{s} [B]_{\mathcal{W}}^{t}$.

The final step is to reexpress this in terms of the tensor product of two projectors. This can be achieved by *singular value decomposition* and leads to the updated PEPS:

$$P'_A \otimes P'_B = (\sum_{s'}^{d} \sum_{\mathcal{V}'}^{D'} [A']_{\mathcal{V}'}^{s'} |s'\rangle\langle\mathcal{V}'|) \otimes (\sum_{t'}^{d} \sum_{\mathcal{W}'}^{D'} [B']_{\mathcal{W}'}^{t'} |t'\rangle\langle\mathcal{W}'|).$$

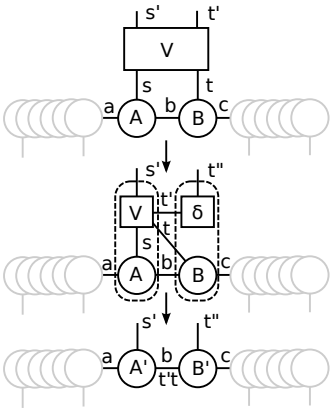

Figure 3.2: A two-qubit gate V acting on two neighboring qubits can increase the virtual dimension of the shared bond. This can be shown by introducing a Kronecker δ tensor and mapping of the tensor network.

Maximal increase in virtual dimension What is the maximum growth of the virtual dimension after a two-qubit operation? Since a one-qubit operation can only affect the qubit itself, no entanglement is generated or lost by this operation and the virtual dimension is unaltered. A two-qubit operation, however, can generate or dissolve entanglement and can, for example, swap two qubits, hence influencing the virtual dimension.

Theorem 3.1.1. *In a PEPS, a two-qudit operation can increase the virtual dimension D between two neighboring sites k and l by at most a factor of d^2, where d is the physical dimension.*

Proof. This can be illustrated by applying a two-qudit gate on two neighboring unentangled qudits, e.g. $D = 1$. The resulting tensors will be

$$[A]_\alpha [B]^{\alpha s}_\beta [C]^{\beta t}_\gamma [D]^\gamma$$

Adding a two qudit operator V in tensor form results in

$$[V]^{s't'}_{st} [A]_\alpha [B]^{\alpha s}_\beta [C]^{\beta t}_\gamma [D]^\gamma.$$

By adding an additional δ function

$$[V]^{s't'}_{st} \delta^{t''}_{t'} [A]_\alpha [B]^{\alpha s}_\beta [C]^{\beta t}_\gamma [D]^\gamma,$$

the tensors can be rearranged in

$$[A]_\alpha ([V]^{s't'}_{st} [B]^{\alpha s}_\beta)(\delta^{t''}_{t'} [C]^{\beta t}_\gamma)[D]^\gamma$$

and merged to

$$[A]_\alpha [B']^{s't'\alpha}_{t\beta} [C']^{\beta t t''}_{\gamma t'} [D]^\gamma.$$

The total dimension indices shared between B' and C' is

$$\dim t \cdot \dim t' \cdot \dim \beta = d^2 D,$$

i.e. an increase by a factor of d^2. □

Measurement on PEPSs

For MPSs we know – by construction – that the measurement value of a single qubit l can be calculated efficiently using the l-th Schmidt decomposition and squaring of the corresponding

matrix. Using a similar method for PEPS does work in the 1D case, as long as the same MPS construction method is used. For higher dimensions, however, the correctness of this method can no longer be guaranteed, since the Schmidt decomposition is no longer well defined. There is, however, a method to calculate expectation values for the complete system. Unfortunately, this can only be done exactly and efficiently for the 1D case.

Exact expectation values The expectation value $\langle\Psi|\hat{O}|\Psi\rangle$ of a local observable $\hat{O}$ in system $|\Psi\rangle$ can also be determined for a PEPS. Note that $\hat{O}$ can be written as a tensor product of local observables on the different sites, and that the calculation is equivalent to a contraction of the tensor network.

To write observable $\hat{O}$ as a tensor product[3] of local observables $\hat{O}_i$ on site i:

$$\hat{O} = \bigotimes_i \hat{O}_i.$$

The expression $\langle\Psi|\hat{O}|\Psi\rangle$ can now be written per site j:

$$[E_j]_{v'v} = \sum_{k'k}^{d} \langle k'|\hat{O}_j|k\rangle [A_j^*]_{v'}^{k'} [A_j]_v^k.$$

Here, $[E_j]_{v'v}$ is the new tensor of site j with virtual indices[4] v and v', while $[A_j]_v^k$ is the original tensor with physical index k virtual index v and $[A_j^*]_{v'}^{k'}$ with indices k' and v' its complex conjugate. The tensors are contracted with the observable over k and k' by summation over the physical dimension d. For a 2D grid PEPS, this will result in a grid as illustrated in Figure 3.3.

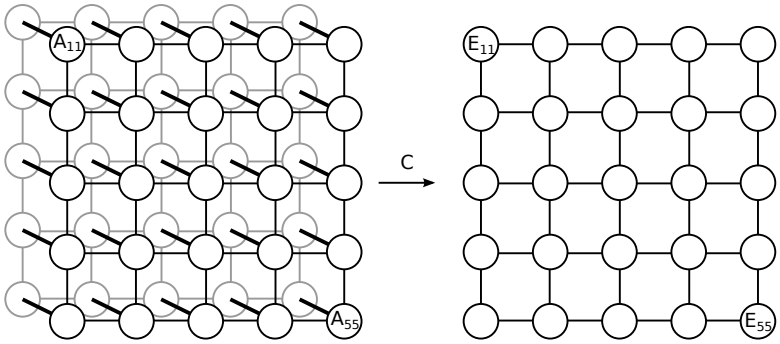

Figure 3.3: After contraction with the observable and the adjoint PEPS, the original PEPS will be transformed to a new tensor grid without physical indices.

To calculate the expectation value, all tensors have to be contracted. For an arbitrary geometry the procedure becomes exponential. This can be easily seen in the case of a $L \times L$ 2D grid PEPS (see Figure 3.4). For example, by contracting first in the physical dimension, the virtual dimension D of all the virtual indices of the tensors will increase by another factor of D, resulting in D^2. Consider a row-by-row contraction as next step. This will result factor of D^2 increase in all the virtual dimensions for each row. For L rows, this results in virtual dimensions of D^{2L} for the final row, making this procedure intractable.

Approximate expectation values Using the procedure shown above, calculating the exact value of expectation values in the 2D PEPS takes exponential time. There is no method known to calculate these values efficiently and exactly. Fortunately, there is a straightforward approximation

[3]The $\bigotimes$ notation works as expected, i.e. $\bigotimes_{i=1}^{3} a_i = a_1 \otimes a_2 \otimes a_3$.

[4]Here v' (and v) is used as a joined index consisting of all the virtual indices present in the geometry. This index has dimension $\dim(v) = \prod_{i\in\mathcal{V}} \dim(i)$, where $\mathcal{V}$ is the set of indices.

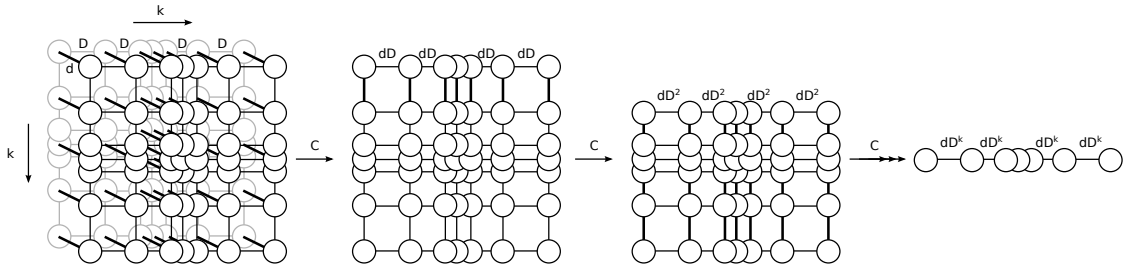

Figure 3.4: The contraction of a 2D grid will result in indices with a dimension exponential in the number of qubits.

method. The idea is to limit the growth of the virtual dimension D of the tensors, say up to a dimension of $\tilde{D}$.

To calculate the expectation value of an observable $\hat{O}$, it is first written as a tensor product of local ones, like above. The main difference is in the order of the contraction and in the additional truncation step. The PEPS is first contracted in the physical dimension, resulting in a new PEPS. This PEPS in contracted row-by-row. The key observation is that the first and last row can be interpreted as an MPS.

In principle, we start with the MPS of the first row and contract the second row. This, however, can result in an increase of the virtual dimension of the MPS, eventually leading to an exponential calculation. To avoid this, a truncation step is now applied to the resulting MPS, reducing the dimension to the preset maximum dimension $\tilde{D}$. This truncation consists of the removal of the smallest Schmidt coefficients in the matrices that are oversize.

This procedure is repeated until the contraction of the MPSs of the last two rows. As a final step, the resulting MPS can be contracted to a rank 0 tensor node, yielding the expectation value of the observable.

Error The error made in every truncation step of this approximation scheme is just the sum of all truncated Schmidt coefficients:

$$\epsilon = \sum_{i>\tilde{D}} \lambda_i,$$

where λ_i are the Schmidt coefficients and $i > \tilde{D}$ are only the values of i that where discarded.

3.1.4 Extensions

There are two straightforward extensions imaginable to the standard linear and grid PEPS: other building blocks and other geometries. They will be illustrated below, including some remarks. More elaborate extensions and enhancements will be discussed in Section 4.2.

Other states than EPR as basis

The formalism is specified on entangled pairs as basis for the virtual structure. However, one could look at using a more complicated building block like GHZ states. See Figure 3.5 for an example for the 3-qubit GHZ state:

$$|GHZ\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}.$$

Other geometries

While in the 2D grid PEPS all nodes are only entangled with their nearest neighbors, this is not an inherent limitation of the formalism. If fact, using the building blocks presented, any graph-like

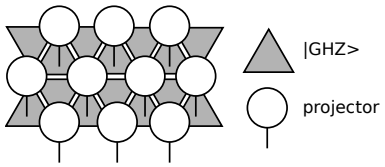

Figure 3.5: Using other building blocks than the entangled pair, potentially other classes of states can be represented efficiently. Here a PEPS based on the 3-qubit GHZ state, visualized by a triangle, is illustrated.

geometry can be built. This opens up the questions what classes of states can be described by which geometries and whether there exist an optimal geometry for every state. Furthermore, we can consider more complex geometries like an infinite, cyclic or periodic grid.

3.2 The teleportation picture

3.2.1 Construction

Imagine a grid of entangled pairs without projectors. The claim is that any n qu*d*it state can be constructed by applying projectors on such grid, consisting of at most $d^{\lfloor \frac{n}{2} \rfloor}$ entangled pairs per two sites.

Theorem 3.2.1 (Verstraete *et al.* [VPC04a, VPC04b])**.** *Any arbitrary n qu*d*it quantum state $|\Psi\rangle$ can be represented as a PEPS, using a virtual dimension of at most $d^{\lfloor \frac{n}{2} \rfloor}$.*

Proof. See the example below. It is easy to see that this can be generalized to an arbitrary state. □

The idea is to use the entangled pairs in the PEPS to teleport the qu*d*its to the right location. The following example will illustrate the procedure.

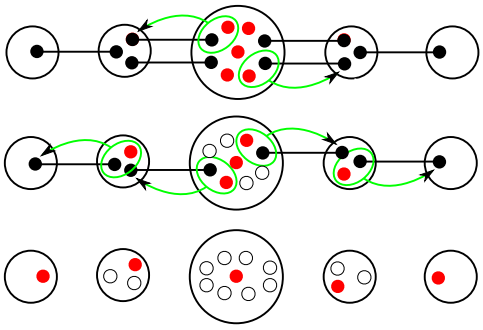

Figure 3.6: Constructing a PEPS using teleportation steps.

Example 3.2.1. Imagine a 5 qubit state $|\Psi\rangle$ and a PEPS as illustrated in Figure 3.2.1. Start now with state $|\Psi\rangle$ in the middle position. It is easy to see that we can use the entangled pairs to teleport the first and second qubit of $|\Psi\rangle$ to the second position of the PEPS. Furthermore, the entangled pair between the first and second position can be used to teleport the first qubit of $|\Psi\rangle$ to the first position. The fourth and fifth qubit of $|\Psi\rangle$ can be teleported in an analogous way to positions 4 and 5, resulting in a PEPS representation of state $|\Psi\rangle$. These teleportation steps result in a virtual dimension of at most $D \leq 2^{\lfloor \frac{5}{2} \rfloor} = 2$.

To start with the 5-qubit quantum state $|\Psi\rangle$ in the middle position, the following projector can be used:

$$P^I = |\Psi\rangle\langle 00000|.$$

This projector acts on five ancilla qubits already present at the middle position.

The teleportation from A to B can be implemented by a teleportation projector $T_{X\to Y}$ that is defined as follows:

$$T_{X\to Y} = \langle 0_A 0_B| + \langle 1_A 1_B|.$$

This projector acts on qubit A and half an entangled pair B on site X, teleporting qubit A to qubit C on site Y, the other half of the entangled pair.

The final projector P_L on site L will consist of several teleportation projectors and an identity projector on qubit l:

$$P_L = T_{L\to M}^{\otimes k} \otimes I_l.$$

3.2.2 Quantum operations

Operations on the PEPS can also be viewed from a teleportation perspective. A one-qubit operation can simply be incorporated in the projector for the corresponding site. This results in a projector of the form

$$\hat{P}_X = UP_X,$$

where the unitary is applied before the qubit is teleported.

For two-qubit operations, the qubits have to be on the same physical site. To solve this the qubit has to be teleported to the other site the operation is acting on, and teleported back after the operation has been applied. This can be accomplished by the following projectors:

$$\hat{P}_X = T_{X\to Y} P_X$$

$$\hat{P}_Y = T_{Y\to X} V_B P_Y,$$

where the qubit is first teleported to site Y, then the unitary V_B is applied on both qubits, after which the qubit is teleported back to site X again. See Figure 3.7 for an illustration.

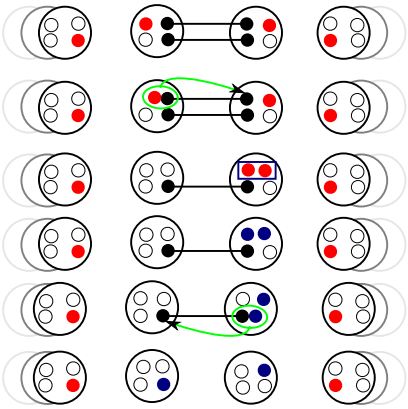

Figure 3.7: The teleportation picture can also be used for quantum operations.

This requires two additional entangled pairs between the two sites the operator acts on. This exactly reflects the maximal increase of the virtual dimension after a two-qudit operation as discussed earlier, i.e. d^2. Note that this procedure can be applied to any PEPS with additional entangled pairs, not only the PEPSs constructed using the teleportation methods.

3.2.3 A more efficient variant

Since the method described above leads to an exponential virtual dimension ($d^{\lfloor \frac{n}{2} \rfloor}$ for a n-qu*d*it state), this method is not feasible for large numbers of qubits. However, it may be possible to construct a PEPS representation of the same state with fewer teleportations. We propose a quantum compression scheme, reducing the number of qubits that have to be teleported.

The scheme works as follows. The first step is to decompose the state $|\Psi\rangle$ on site A using the Schmidt decomposition:

$$|\Psi\rangle = \sum_{i=1}^{\chi} \lambda_i |\psi_i^{[1]}\rangle_A |\psi_i^{[2..n]}\rangle_A.$$

The next step is to rotate the second part of the state to a convenient basis. This can be accomplished by a unitary U:

$$U \sum_{i}^{\chi} |\psi_i^{[2\ldots n]}\rangle_A = \sum_{i}^{\chi} |i0\ldots0\rangle_A.$$

Note that this is equivalent to labeling each basis state. The resulting state is:

$$|\Psi'\rangle = \sum_{i=1}^{\chi} \lambda_i |\psi_i^{[1]}\rangle_A |i0\ldots0\rangle_A.$$

The only data that has to be teleported to the second site are the states

$$\sum_{i=1}^{\chi} \lambda_i |i\rangle,$$

since the additional zeros can be reconstructed. This state can be teleported using $\log \chi$ entangled pairs. After the state has been teleported, it has to be decoded:

$$|\Psi\rangle = \sum_{i=1}^{\chi} \lambda_i |\psi_i^{[1]}\rangle_A |i0\ldots0\rangle_B$$

and rotated back again, using the inverse of U. This results in the second half of the bipartition being at site B:

$$|\Psi\rangle = \sum_{i=1}^{\chi} \lambda_i |\psi_i^{[1]}\rangle_A |\psi_i^{[2\ldots n]}\rangle_B$$

The teleportation to the next site proceeds in a similar way, using the second Schmidt decomposition. In this fashion, all qubits are teleported to the correct sites, e.g. the Schmidt decomposition for site l is:

$$|\Psi\rangle = \sum_{i=1}^{\chi} \lambda_i |\psi_i^{[1\ldots l]}\rangle_L |\psi_i^{[l+1\ldots n]}\rangle_L.$$

Note how the efficiency of this scheme depends on the Schmidt rank of the decomposition, analogue to the construction method for MPS. A highly entangled state will still need an exponential number of teleporations, as expected. For a 1D linear PEPS, the representation will be equivalent to the conventional construction method of MPS (Section 2.1.2).

Scaling to higher dimensions and more complex topologies

While it is easy to see that this procedure works in 1D, in two and more dimensions an additional ingredient is necessary to actually make use of the additional degrees of freedom. For example, applying the procedure above to a 2D grid state will result in a tree-like sequential traversal of the grid and the resulting PEPS will not be a genuine 2D PEPS, but a tree tensor network. There is no construction method known that does use all of the additional degrees of freedom of a two and higher dimensional PEPS.

Representing the result of a BQP computations

Using the teleportation picture, it can be shown that the final state of any BQP computation has an efficient 2D grid PEPS representation. This is done by showing that any quantum circuit computation can be mimicked in a 2D PEPS. The contraction of the PEPS is now equivalent to simulating the result of the circuit.

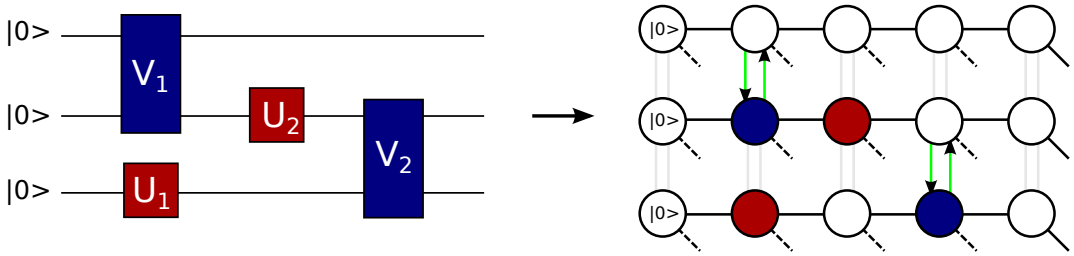

Figure 3.8: Any circuit can be mapped to a 2D grid PEPS.

An arbitrary quantum circuit can be mapped to a 2D PEPS as illustrated in Figure 3.8. The qubit lines are mapped to the horizontal lines of the grid, with the vertical index as the qubit index and the horizontal index as a time index. We use the general assumption that the circuit only consists of one-qubit and two-qubit operations on neighbours (Section 1.3.2).

The input of the circuit – without loss of generality assumed to be the all $|0\rangle$ state – can be easily mapped to the first column using the projectors

$$P^I = |0\rangle\langle 0|.$$

Of course, this can be done for an arbitrary product input using the projector

$$\hat{P}^I = (\alpha|0\rangle + \beta|1\rangle)\langle 0|.$$

The next step is the evolution of the qubits in time. This is modeled by a teleportation step to the right. This can be implemented in a projector as follows:

$$T = |0\rangle(\langle 00| + \langle 11|).$$

The "virtual Bell measurement" $\langle 00| + \langle 11|$ effectively teleports qubit A the site to the right. While the value is irrelevant for the map of the circuit, in a PEPS each projector maps to a physical space. In this case, the physical index is set to $|0\rangle$. This teleporations results in the need for one entangled pair in the horizontal time direction of the grid.

The application of the unitary gates on the qubits can be implemented using the projectors discussed in the previous section. Note that this results in the need for two entangled pairs in the vertical qubit direction of the grid.

The total cost in number of entangled pairs of implementing a n qubit quantum circuit of depth d is $\mathcal{O}(2nd)$.

Even a PEPS with a single entangled pair between every node is enough for this mapping, using only polynomial additional nodes. To see this, note that the only a unitary operation on two qubits needs two entangled pairs. An additional row between every two rows of the original PEPS can be used to "reroute" the teleportation, reducing the number of entangled pairs needed. The procedure is illustrated in Figure 3.9. Starting with qubit A and B in their original positions, qubit B is first teleported to new in-between row. In the next step, both qubits are teleported a column to the right. In the same column, now qubit A is teleported to the node on in-between row and a unitary two-qubit operator is applied on both qubits. Next, qubit B is teleported a row down again and both qubits are teleported to the right. In the final step, qubit A is teleported up to its original row again.

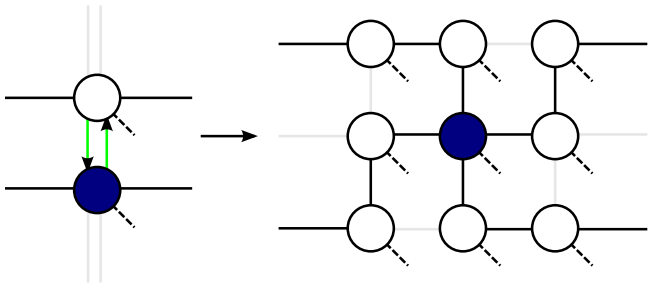

Figure 3.9: A two qubit gate can be implemented using additional nodes.

Theorem 3.2.2. *The contraction of a 2D PEPS is at least as hard as simulating an arbitrary quantum circuit, i.e. BQP-hard.*

Proof. Using the mapping from a arbitrary quantum circuit to a 2D PEPS described earlier, contracting the complete network would be equivalent to calculating the resulting state of the circuit. Since the class of problems solvable by a quantum circuit in polynomial time is BQP, the contraction of the circuit is BQP-hard. □

Corollary. *The exists no general efficient classical method to approximate the expectation value of a single qubit measurement in a 2D grid PEPS, unless $P = BQP$.*

3.3 Complexity of PEPS

As can be seen from the construction methods, every quantum state can be encoded as a PEPS. In the teleporation picture it was already shown that there exist no general methods to efficiently calculate local expectation values and that contraction of a 2D PEPS is BQP-hard. In this section we fill further study the complexity of PEPS. We will show the relation between measurement based quantum computing and PEPS and the resulting implications for the complexity of PEPS. Furthermore, we will present a sharp bound on the complexity, as described by Schuch *et al.*.

3.3.1 2D PEPS as cluster state

Measurement based quantum computing is a quantum computing paradigm which involves only measurements on an earlier prepared state. It turns out that, for the right class of prepared states, any quantum computation can be done using only measurements.

The classes of states used by measurement based quantum are *graph states*, e.g. states that can be interpreted as vertices for qubits and edges for entanglement relations between the vertices. A subset of these graph states are the *cluster states*, states represented by a regular cluster of qubits, for example a grid. A 2D cluster state turns out to be universal for quantum computing. It was shown by Verstraete *et al.* [VC04b] that these cluster states can be mapped to a PEPS and actually form a subset of the 2D grid PEPS with a virtual dimension of $D = 2$.

Theorem 3.3.1. *Calculation of the post-measurement state in a 2D PEPS is BQP-hard.*

Proof. Cluster states are a subset of 2D PEPS. This implies that a 2D PEPS combined with measurement enables universal quantum computations. Since the class of quantum computations that can be implemented with a polynomial number of gates forms the class BQP, calculation of the post-measurement state on a 2D grid PEPS is BQP-hard. □

3.3.2 A sharp bound

Given a black box method that creates any PEPS and given the fact that we can perform arbitrary local measurements on the PEPS, what kind of computational problems would we be able to solve

efficiently? This question has been briefly looked into by Verstraete *et al.* in [VWPC06] and in detail by Schuch *et al.* in [SWVC07]. The proofs will not be shown here, but are based on the duality of PEPS and post-selection.

Two different types of computational problems are considered. First of all, the class of problems we could solve by merely creating a specific PEPS and performing a measurement.

Lemma 3.3.2 (Schuch *et al.* [SWVC07])**.** *Given a method to create any PEPS, any PP-complete problem can be solved.*

The PP complexity class is the class of problems that can be solved by a probabilistic Turing machine with an error rate smaller than 0.5. That is, the Turing machine will give the correct answer in more than half of the cases.

The second class of problems considered is the complexity of classically simulating a PEPS.

Lemma 3.3.3 (Schuch *et al.* [SWVC07])**.** *Classically computing local expectation values of a PEPS is a #P-complete problem.*

The #P complexity class is the class of counting problems corresponding decision problems of the class NP. That is, if a decision problem asks whether there exists a solution, than the counting problem asks how many solutions there are.

Furthermore, they extend the result to the contraction of arbitrary tensor networks.

Corollary (Schuch *et al.* [SWVC07])**.** *The contraction of an arbitrary tensor network is #P-complete.*

3.4 Capacity of PEPS

It was already shown that any quantum state can be encoded using a PEPS. The question of interest for classical simulation, however, is which classes of quantum states can be represented *efficiently* using a PEPS. Furthermore, it is interesting to see how these classes depend on the geometry of the PEPS. In this section the 1D linear and 2D grid PEPS will be compared.

To convert a 2D grid PEPS to a linear 1D PEPS, a linear ordering of the nodes in the grid has to be chosen. There are several orderings possible. In the linear ordering of the nodes, the entangled pairs will no longer be shared between nearest neighbors only. This indicates the presence of long-range entanglement. To convert this representation to a real one-dimensional system, the long-range interactions have to be incorporated in the virtual dimension of the intermediate nodes (Section 1.2.4). The virtual dimension needed in the 1D PEPS is the dimension of the bond between two nearest neighbors multiplied with the dimension of all the long-range bonds parallel to the original bond. For simplicity, we assume that the virtual bond dimension is D for all the bonds. The problem is now to determine the number of parallel bonds, i.e. the number of edges that cross a bipartition perpendicular to the linear graph. Note that this number depends on the order and that determining the optimal order that leads to the smallest D, is an NP-hard problem.

It turns out that the number edges that cross a bipartition perpendicular to a linear ordering is actually related to a known concept in graph theory: the *cutwidth* of a graph.

Definition 3.4.1 (Bodlaender [Bod86])**.** Let $G = (V, E)$ be a graph, with $n = |V|$:

- A linear ordering is a bijective mapping $f : V \to \{1, \ldots, n\}$.
- A linear ordering f of G is said to have cutwidth k if $k = \max_{1 \leq i \leq n} |\{(u, v) \in E | f(u) \leq i \leq f(v)\}|$.
- The cutwidth of G, denoted $cw(G)$, is the minimum cutwidth of a linear ordering f over all possible linear orderings of G.

The concept is perhaps best illustrated by an example.

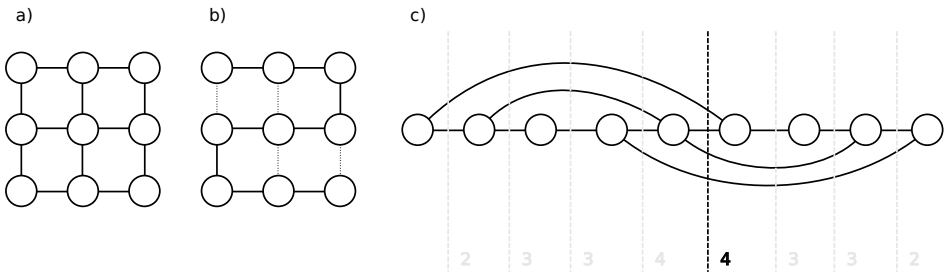

Figure 3.10: Given a 3×3 grid graph (a), a linear ordering can be chosen (b) that can be used to rearrange the graph to a line (c). Now, the cutwidth of the linear ordering is exactly the maximum of edges crossed for any bipartition perpendicular to the line.

Example 3.4.1. Imagine a 3×3 grid graph, as illustrated in Figure 3.10a. To determine the cutwidth of this graph, a linear ordering has to be chosen. For example, the "snake-like" ordering (Figure 3.10b): order the nodes starting from left-to-right on the first row and continue on second row from right-to-left and on the next row from left-to-right again, and so on switching the direction on each row. Rearranging the graph to a line according to the ordering, gives the result in Figure 3.10c. Now, the cutwidth of this linear ordering is exactly the maximum number of edges crossed considering all bipartitions perpendicular to the line.

First we will provide an upper bound for the cutwidth of a grid graph.

Lemma 3.4.1. *A $k \times k$ grid graph has a cutwidth of at most $k + 1$.*

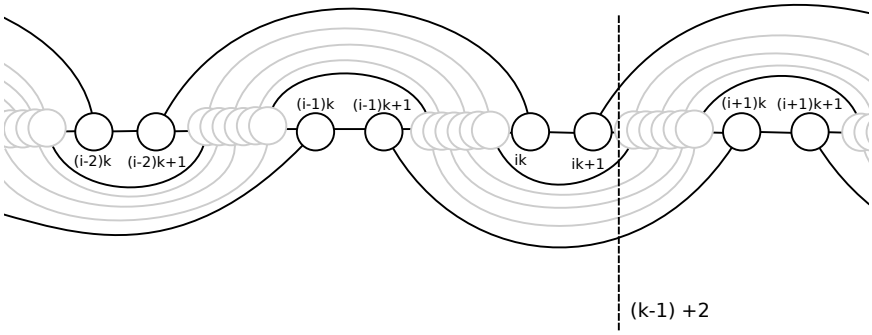

Figure 3.11: Rearranging a $k \times k$ grid graph on a line using a "snake"-like linear ordering results in a cutwidth of $k + 1$.

Proof. Using the "snake"-like linear ordering presented earlier, any $k \times k$ grid graph can be rearranged on a line. This will result in the structure that is partly illustrated in Figure 3.11. The cutwidth of this linear ordering is the maximum number of edges we cross considering all bipartitions perpendicular to the line. Counting the number of edges we cross – $k - 1$ edges for all the vertices between vertex $ik + 1$ and vertex $(i + 1)k + 1$, and additionally an edge between vertices $ik + 1$ and $ik + 2$ and an edge between vertices $ik + 1$ and $(i + 2)k$ – this leads to a cutwidth of $(k - 1) + 2 = k + 1$.

Since the cutwidth of the graph is the minimum of the cutwidths of all possible linear orderings, $k + 1$ is an upper bound for the cutwidth of a grid graph. □

To provide a lower bound for the cutwidth of a 2D grid graph, we use some results from the literature. Bodlaender has investigated the relation between the cutwidth and some other graph-theoretic notions, including the notion of *tree-width*.

Definition 3.4.2 (Bodlaender [Bod86])**.** Let $G = (V, E)$ be a graph. A tree-decomposition of G is a pair $(\{X_i | i \in I\}, T = (I, F))$, with $\{X_i | i \in I\}$ a family of subsets of V, and T a tree, with the following properties:

- $\bigcup_{i \in I} X_i = V$
- For every edge $e = (v, w) \in E$, there is a subset $X_i, i \in I$ with $v \in X_i$ and $w \in X_i$.
- For all $i, j, k \in I$, if j lies on the path in T from i to k, then $X_i \cap X_k \subseteq X_j$.

The tree-width of a tree-decomposition $(\{X_i | i \in I\}, T)$ is $\max_{i \in I} |X_i| - 1$.

The tree-width of G, denoted by $tw(G)$, is the minimum tree-width of a tree-decomposition of G, taken over all possible tree-decompositions of G.

Lemma 3.4.2 (Bodlaender [Bod86])**.** *For every graph G, $cw(G) \geq tw(G)$ holds.*

The tree-width was first introduced by Robertson and Seymour and they determined the value for different kinds of graphs, including grid graphs.

Lemma 3.4.3 (Robertson and Seymour [RS86])**.** *The tree-width of a $k \times k$ grid graph is k.*

Combining these two lemmas, we can state the following:

Lemma 3.4.4. *The cutwidth of a $k \times k$ grid graph is greater than or equal to k.*

Proof. This follows trivially from Lemma 3.4.3 and 3.4.2. □

Combining this lower bound with the upper bound above, we can state the following theorem:

Theorem 3.4.5. *The cutwidth of a $k \times k$ grid-graph G is bounded by $k \leq cw(G) \leq k + 1$.*

Proof. Lemma 3.4.4 provides a lower bound for the cutwidth. Lemma 3.4.1 provides an upper bound. Combining these bounds results in $k \leq cw(G) \leq k + 1$. □

Now back to PEPS. Using the bounds on the cutwidth of a grid graph and the assumption of a uniform virtual dimension D in the original PEPS, we can state the following theorem:

Theorem 3.4.6. *Encoding a $k \times k$ 2D grid PEPS with a uniform virtual dimension D using a 1D linear PEPS leads to a maximal new virtual dimension $D' = D^k$ or $D' = D^{k+1}$.*

Proof. Interpreting the $k \times k$ 2D grid PEPS with a uniform virtual dimension D as a grid graph, we can reorder the 2D grid PEPS to a linear 1D PEPS. The new virtual dimension depends on the cutwidth of the graph. By Theorem 3.4.5, the cutwidth of a $k \times k$ grid graph G is $k \leq cw(G) \leq k+1$. By the mapping procedure for tensor networks described earlier (Section 1.2.4), the maximal new virtual dimension becomes D^k or D^{k+1}. □

Chapter 4

Other tensor network methods

In this chapter we will briefly discuss a number of other tensor network methods. First we will look at the *contracting tensor network* formalism, a convenient formalism developed specifically for the simulation of quantum circuits. Next, several enhanced tensor network methods will be described, including recent developments. Finally, for comparison, we will describe the matchgate formalism and stabilizer states, but that are *not* based on tensor networks.

4.1 Contracting Tensor Network formalism

The methods discussed thus far all originated from the simulation of physical systems and find their main applications there. The *contracting tensor network* (CTN) formalism, however, was developed specifically for the simulation of quantum circuits. In this formalism, a quantum circuit and its input is directly mapped to a tensor network. The application of the circuit turns out to be equivalent to the contraction of the graph. As long as the contraction can be done efficiently, the circuit can be classically simulated. The complexity of the contraction procedure turns out to be related with the *tree-width* of the network.

Contrary to the other methods, the CTN formalism represents a quantum circuit, not a state as such. However, one can interpret the outcome of the quantum circuit as the state that is represented by the network. This allows all states that can be build using a circuit of only polynomial size to be represented efficiently. Additional unitary operations on the state and measurement can also be easily be calculated by adding the corresponding elements to the circuit.

The idea of using the contraction of tensor networks to simulate quantum circuits was developed by Markov and Chi [MS05]. A simplified version, based on *linear circuits*, was proposed by Jozsa [Joz06].

An interesting thought is that some quantum calculations, including the *quantum Fourier transform* – a key part in many quantum algorithms – can be approximated by a circuit of only limited tree-width. Unfortunately, this does not allow us to for example calculate integer factorization efficiently, because there is no known method to perform another key step of *Shor's factoring algorithm*, the exponentiation, in a circuit with limited tree-width [ALM06].

4.1.1 Structure

An arbitrary quantum circuit can be converted to a tensor network using the relations described next. The conversion was originally defined on density operators and superoperators (e.g. operators that linear map density operators to density operators).

Input The input is assumed to be a product state. In the original approach the state was represented as density matrix and was decomposed in a smaller set of matrices of a specific class.

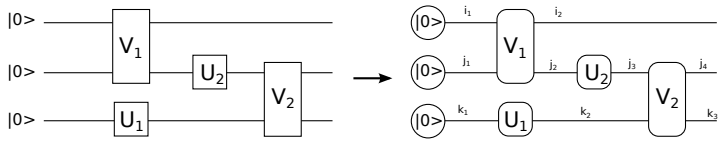

Figure 4.1: A quantum circuit represented as a tensor network.

The simplified version uses a rank 1 tensor of dimension d (i.e. a state vector). Note that any desired input state can be generated using additional quantum gates.

Wires The wires between the different components of the quantum circuit are modelled as indices shared between the components. Each segment of a wire has a unique index.

Gates Quantum gates can be either directly interpreted as rank 2 tensors or converted to superoperators on density matrices, depending on the input representation.

Output The output of the circuit is represented as open indices in the tensor network.

4.1.2 Contraction

The evolution of time in the circuit is equivalent to the contraction of the operator nodes with the nodes representing the initial state. In this case the quantum state is step-wise updated.

The order of the tensor contraction does not matter for the final result. This enables the freedom of choosing an optimal order of contraction in which the dimension of the indices of the intermediate tensors is small. It turns out that the complexity of the contraction of a tensor network is related to the tree-width (Section 3.4) of the circuit. Circuits with a logarithmic tree-width can be efficiently simulated classically.

Another metric often used in this context is the *contraction complexity*, the maximum rank of a tensor during the contraction. It was shown by Markov and Shi [MS05] that the contraction complexity is related to the tree-width by $cc(G) = tw(G^*)$, with G^* the *line graph*[1] of G.

4.1.3 Measurement

Measurements in the CTN formalism are modelled with *measurement scenarios*.

Definition 4.1.1 (Markov and Shi [MS05])**.** Let $m \geq 1$ be an integer. A *measurement scenario* on m qubits is a function $\tau : [m] \to \mathbf{L}(\mathbb{C}^2)$, such that $\tau(i)$ is a single-qubit POVM measurement element.

Contraction of the circuit with input and the measurement scenario gives the rank 0 tensor with the probability of that scenario.

In the simplified version of Jozsa, measurements are modelled with projectors $|k\rangle\langle k|$. They can be interpreted as rank two tensors. The probability of measurement outcome k can now be determined in the following situations:

Single-qubit at the end Add a projector for the desired outcome between the circuit and its adjoint. Contraction of the network will result in a rank 0 tensor yielding the probability.

Multi-qubit at the end First perform the measurement as described above on the first qubit. Replace the projector of the first qubit with the renormalized projector $\Pi(k_a)/\sqrt{prob(k_a)}$. Now calculate the probability for the desired outcome of the second procedure, given the previous outcome of the first qubit. Repeat this procedure for all qubits that are to be measured.

[1] The *line graph* G^* of graph G is defined as $V(G^*) \equiv E(G)$ and $E(G^*) \equiv \{\{e_1, e_2\} \subseteq E(G) : e_1 \neq e_2, \exists v \in V(G^*)$ such that e_1 and e_2 both are incident to $v\}$.

Single-qubit with dependencies Simulate the circuit up to the first measurement. Perform the measurement as described before. Replace the projector with the renormalized projector as in the previous case and fix the dependent operations based on the measurement outcome. Repeat the procedure for all measurements with dependencies.

4.2 Enhanced methods

The tensor network methods discussed so far – MPS, TTN, PEPS, and CTN – already enable classical simulation of an interesting class of quantum systems. These methods are, however, not without drawbacks. In the recent years several enhancements over these conventional methods have been developed.

The first method we will describe is the *multi-scale entangelement renormalization ansatz*, which adds a virtual dimension to the tensor network that allows for more efficient operations. The second group of methods is closely related, but "complementary rather than overlapping" as stated by one of the authors. These include *string-bond states*, *entangled-plaquette states* and *correlator product states*. The third group concentrates on *graph states* and extensions thereof, including *weighted graph states* and *weighted graph matrix product states*. Finally some other enhancements over the conventional tensor network methods will be discussed, namely *sequential generated states* and *complete-graph tensor network states*.

4.2.1 Multi-scale entanglement renormalization ansatz

The *multi-scale entanglement renormalization ansatz* (MERA) is another extension of the MPS picture, developed by Vidal [Vid06, Vid05]. The idea is to add an additional dimension to an existing tensor network. The main advantage of this methods is that single qubit measurements can be done efficiently, contrary to (2D) PEPS.

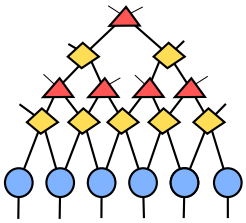

Figure 4.2: The structure of MERA in 1D. Note the different types of tensors.

MERA consists of a D+1 tensor network where the +1 is a tree-like circuit on the D dimensional state (Figure 4.2). This additional dimension of MERA can be interpreted in two ways:

- as a circuit that starts with a product state $|0\rangle^{\otimes N}$ and constructs the state $|\Psi\rangle$ using unitary gates
- as a coarse graining of state $|\Psi\rangle$

The basic building blocks of the tree-like structure are two types of tensors:

- disentanglers
- isometries

The *disentanglers* transform a state $|\Psi\rangle$ into a less entangled state $|\tilde{\Psi}\rangle$. The *isometries* combine states into smaller states, essential performing the rescaling operation.

Due to the form of these two tensors, a large group of tensors cancel out during the contraction of the graph. This allows local properties to be calculated efficiently by contraction of the *light cone* or *causal cone* of the states. See Figure 4.3 for an example.

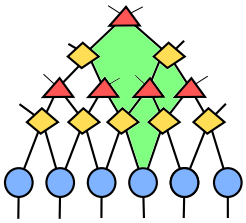

Figure 4.3: The contraction needed for the calculation of local properties is limited to the light cone.

4.2.2 String-Bond states

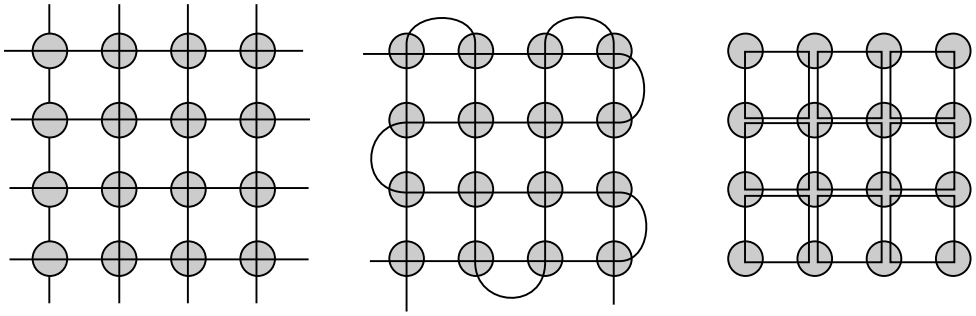

Figure 4.4: Several string-bond states as tensor networks.

The class of string-bond states (SBS) [SWVC08] uses strings of operators on a lattice.
A string-bond state is defined as follows:

$$\langle n|\psi\rangle = \prod_{s\in S} Tr[\prod_{x\in s} M^{s,x}_{n_x}],$$

with $|n\rangle = |n_1\rangle \dots |n_N\rangle$ a local basis, $s \in S$ a set of strings and $M^{s,x}_{n_x}$ are some complex $D \times D$ matrices.

In other words, the strings model a sequential traversal through the graph, combining multiple traversals to fill the complete graph.

Interestingly, the states form a subset of PEPS incorporating the states of interest in quantum information theory, including *toric codes* and *cluster states*. Using Monte-Carlo sampling, the expectation value of local observables can be calculated efficiently. This makes this class of states suitable as ansatz for variational algorithms.

4.2.3 Entangled-plaquette states

The idea of entangled-plaquette states (EPS) proposed by Mezzacapo *et al.* [MSBC09] is to divide a 2D grid PEPS in sub-blocks and calculate the parameters for each block separately. Contrary to a previous method discussed in [IOD09], overlap of the plaquettes is allowed. By tuning the size and overlap of the sub-blocks, this method can be optimized. Applying larger blocks in an iterative fashion provides more accurate results. The expectation values of observables is calculated using Variational Monte Carlo.

4.2.4 Correlator product states

Contrary to the conventional methods, correlator product states (CPS) [CKUC09] do not use virtual dimensions to model the correlations between the physical sites, but model them explicitly

using *correlators*. This concept is actually equivalent to the plaquettes used in the previous methods.

For example, the nearest neighbour correlation is modelled as:

$$|\Psi\rangle = \sum_{\{q\}} \prod_{\langle ij \rangle} C^{q_i q_j} |q_1 \dots q_L\rangle,$$

with $\langle ij \rangle$ nearest neighbors and q_i the qubits.

Expectation values can be calculated by Monte-Carlo sampling or by deterministic means. The correlators can be augmented with virtual indices. Adding one virtual index results in states similar to SBS.

4.2.5 Weighted graph states

The weighed graph states (WGS) were introduced as an alternative to the MPS and its derivatives by Anders *et al.* [ABD07]. Contrary to MPSs, in which the efficiency of the representation scales with the amount of entanglement, the WGS does allow for an efficient representation of highly entangled states. Furthermore, the geometry of the graph is not related to the actual structure of the represented state and as such is equally suitable for two and higher dimensional states.

While the performance is not as good as DMRG for 1D, WGSs can be used to calculate reduced density matrices of the state efficiently for any dimension.

4.2.6 Renormalization algorithm with graph enhancement

In response to the WGS discussed earlier, Hübener *et al.* [HKH+09] enhanced the MPS by combining them with WGSs. The idea is to start with an MPS and apply commuting unitaries to any two sites, enabling long-range entanglement while local properties still can be calculated efficiently. Based on these states a new renormalization algorithm was developed, outperforming DMRG in several cases.

4.2.7 Sequentially generated states

One of the main differences between MPSs and PEPSs is that the first can be contracted efficiently and the latter not. Furthermore, the first can be generated efficiently using sequential methods, whereas such methods are not known for (2D) PEPS and may not exist.

The question rises which is the broadest class of states that can be generated sequentially and contracted efficiently. The class of sequential generated states was studied for two-dimensional systems by Bañuls *et al.* [BPW+08]. The class combines the best of MPSs – expectation values can be determined efficiently – with the possibility to describe 2D systems. The class is shown to be a subclass of PEPSs and suitable as an ansatz for variational algorithms.

4.2.8 Concatenated tensor network states

Observing the trouble of simulating time evolution with MPS, PEPS and MERA, Hübener *et al.* developed concatenated tensor network states (CTS) [HND10]. The idea is to replace tensors with a high rank with a network of tensors of low rank. Repeating this procedure, e.g. *concatenating* these tensor networks, results in a PEPS-like structure where only some of the nodes have a physical index. It has been shown that in some cases, CTS are more efficient than the conventional tensor network methods.

4.2.9 Complete-graph tensor network states

The complete-graph tensor network states (CGTN) of Marti *et al.* [MBR+10] were specifically developed to capture the electronic structure of molecules. They build on on the family of SBS, EPS and CPS.

A n-qud*i*t state has the following CGTN representation:

$$|\Psi_{\text{CGTN}}\rangle = \sum_{s_1 \ldots s_n}^{d} \prod_{\alpha}^{n} \prod_{\beta \leq \alpha} A_{\alpha\beta}^{s_\alpha s_\beta} |s_1 \ldots s_n\rangle,$$

with A a tensor shared between the two sites α and β, s_α and s_β the physical indices and $s_1 \ldots s_n$ the corresponding basis.

In its basic form, the CGTN formalism incorporates all the two orbital interactions, capturing the full configuration of the molecule. This can be interpreted as a subset of the CPS formalism, consisting of only two-site correlators. It can be extended to include three orbital and higher-order interactions using higher-order correlators.

Expectation values can be calculated using a variational Monte Carlo scheme on CGTN states.

4.3 Other methods

This section introduces a number of classical simulation methods for quantum systems not based on tensor networks. The reader is referred to the book by Viamontes, Markov and Hayes [VMH09] for an in-depth review of quantum circuit simulation.

4.3.1 Matchgates

Matchgates were introduced by Valiant [Val02] as an interesting class of quantum gates, based on the graph-theoretical concept of matchgates. They are two-qubit unitary operations of the form

$$G_{AB} \begin{pmatrix} a_{11} & 0 & 0 & a_{12} \\ 0 & b_{11} & b_{12} & 0 \\ 0 & b_{21} & b_{22} & 0 \\ a_{21} & 0 & 0 & a_{22} \end{pmatrix},$$

composed out of the matrices

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}; B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}.$$

The matrices A and B have the additional property that they are unitary and that $\det(A) = \det(B)$. They are unitaries that act on the even- and odd-parity two-qubit subspaces respectively.

It was shown that a circuit consisting of matchgates – a *matchcircuit* – acting on nearest neighbors, with a product state as input and a measurement in the computational basis on the output, can be simulated in polynomial time. Furthermore, the class of matchgates acting on next nearest neighbors (or, equivalently, matchgates on nearest neighbors and the SWAP gate) were shown to be universal for quantum computing.

The class of quantum computations simulatable with matchcircuits is neither contained by nor contained in the class simulatable by tensor network methods.

4.3.2 Stabilizer formalism

The *stabilizer formalism* originates from quantum error-correcting codes, where they were used to construct stabilizer codes. However, they found interesting applications in a broader scope. One of the applications is in the simulation of quantum circuits [AG04]. The key idea is that some states are invariant under a set of operations. In some cases the set of operations uniquely defines such a *stabilizer state*. Furthermore, the representation of this set of operations could be smaller than the representation of the state itself. This makes the stabilizer formalism a candidate for the efficient classical representation of quantum states.

These states can be efficiently updated as long as the applied unitaries are in the *Clifford group*, the group generated by the Hadamard, phase and controlled-NOT gates. A measurement on a stabilizer state can also be done in polynomial time.

Chapter 5

Applications

In this chapter we will look two main applications of tensor network methods, besides the simulation of quantum circuits. First of all we will discusse applications in the field of solid state physics. Methods to calculate the energy of spin system configurations, in particular the ground states, will be discussed. Most tensor network methods were specifically developed for this application and it is arguably the focus of most current research in the field.

Secondly, some more recent applications in the field of quantum chemistry will be discussed. Since the discovery that tensor network methods could also be applied to fermions, quantum chemists took also an interest in tensor networks and not only adapted the existing methods to make them suitable for molecules, but developed some new methods specifically for quantum chemistry as well.

5.1 Physics

We will first discuss two common concepts in physics: *variational methods* and the *renormalization group*. These will both be used in the *density matrix renormalization group* procedure to calculate the ground state of a spin system.

5.1.1 Variational methods

Variational methods are based on the *variational principle* and *variational calculus*. Basically one chooses a set of parameters for a certain problem that has to be minimized/maximized and tries to iteratively optimize the set of parameters. Based on the outcome of one iteration, the set of input parameters is adjusted. The procedure is repeated till either a fixed point or the desired accuracy is reached.

The key idea is that the ground state energy is a lower bound on the energy and that any other input than the ground state wave function will result in a higher energy as output from the algorithm.

5.1.2 Renormalization group

In physics, it is not uncommon to look at a system or process at different scales. Even more generally, scale has not necessarily be length, but can also be energy for example. The *renormalization group* (RG) is a method in physics to deal with the relations between these different length scales.

A simple example of a RG is the use of blocking of a spin grid (see Figure 5.1). Imagine a grid of spins, each with individual properties and interactions. Now, at a slightly different scale we can look a blocks of 2×2 spins. This reduces the number of sites and can reduce the complexity of the problem. The interactions and properties of the blocks can now be described in a similar way as the individual spins.

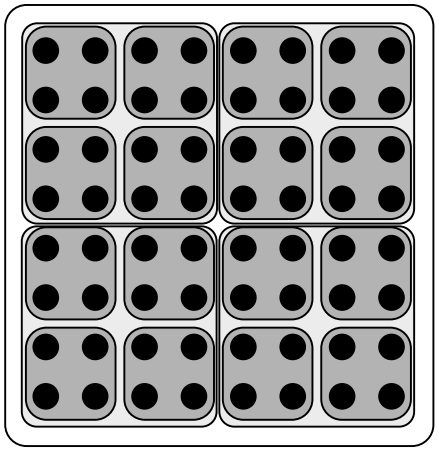

Figure 5.1: An illustration of the blocking process.

This procedure can be repeated, leading to a smaller number of larger and larger blocks. The RG transformation describes how the properties of the system change with the scale. The procedure can be repeated until one large block remains, but usually a fixed point is reached earlier in which the properties and interactions are invariant under the upscaling.

More formally, let say that a certain property of the system can be describe as a function of the state of the system and an interaction term. Now, consider a blocking of the system leading to a smaller state description. The aim is to rewrite the function in terms of the new state, using a map form the original interaction term to a new set of interaction terms. If this is possible, then the property is said to be *renormalizable*.

5.1.3 Density matrix renormalization group

The amazing precision enabled by the density matrix renormalization group (DMRG) method is what really sparked the interest in the matrix product states. Based on Wilson's numerical Renormalization Group [Wil75], DMRG is a true variational method on density matrices. The method was first introduced by White [Whi92]; the method described below is mainly based on [CRRM06].

The ansatz and data structure

DMRG is a variational method on matrix product states. The initial state for the algorithm or *ansatz* is an MPS reasonably close enough to the ground state. Usually this state is prepared using the results of a conventional method. However, one can also start with a random MPS with a fixed virtual dimension.

The Hamiltonian

In the steps described below, the Hamiltonian quantum operator is used to calculate the ground state of the system. In principle this procedure can be used for any quantum operator.

Key ideas

The algorithm is based on the observation that, due to the structure of the MPS, it is possible to choose a site on the MPS, fix the value and optimize the structure. Note than in general the evolution of a quantum state will result in a state with an exponential virtual dimension. We can however approximate this state by adding a truncation step to the algorithm that limits the virtual dimension of the MPS.

Building blocks

A block $\square = B(L, m_L)$ is defined to be a chain n_L of states, with chain length L. The Hamiltonian of block B is:

$$\hat{H}_B = \sum_i \sum_q J(q)\hat{S}_i(q)\hat{T}_{i+1}(q) + \hat{B}(q)\hat{V}_i(q).$$

Here $J(q)$ and $B(q)$ are coupling constants and $\hat{S}_i(q)_q$, $\hat{T}_i(q)_q$ and $\hat{V}_i(q)_q$ are sets of operators acting on the i-th site.

The steps

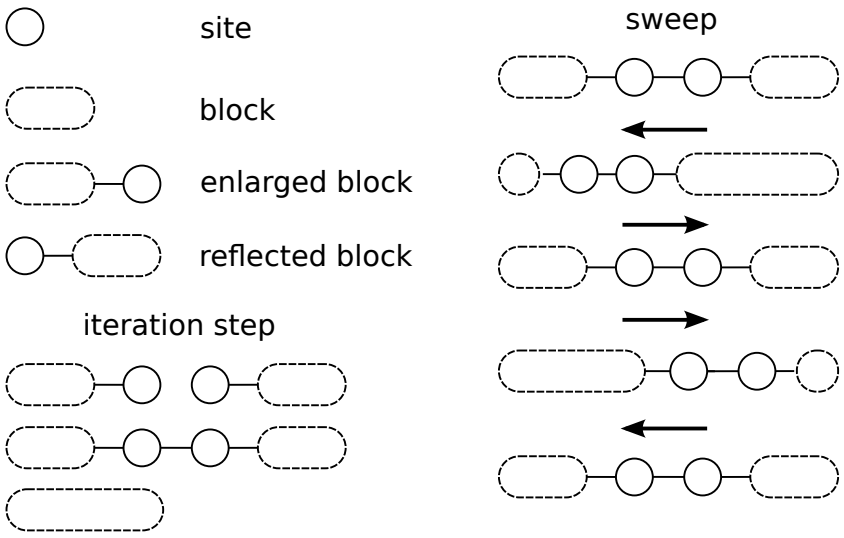

Figure 5.2: The DMRG procedure step-by-step.

The procedure starts with a block consisting of one site $B(1, D)$, where D is the virtual dimension of the site.

Step 1 Adding a site to the right
Block B is extended by a site S on the right. The new Hamiltonian becomes $\hat{H}_E = \hat{H}_B + \hat{H}_S + \hat{H}_{BS}$, with $\hat{H}_B$ the Hamiltonian of block B, $\hat{H}_S$ the Hamiltonian of site S and $\hat{H}_{BS}$ the Hamiltonian of the interaction between block B and site S.

Step 2 Reflecting the enlarged block
A reflected block B' is also extended with one site S', now on the left site. Again, the new Hamiltonian $\hat{H}_{E'}$ consists of the sum of the Hamiltonians of the parts, plus an additional interaction term:

$$\hat{H}_{E'} = \hat{H}_B + \hat{H}_{S'} + \hat{H}_{BS'}$$

Step 3 Build the super-block
Now both parts can be connected. The Hamiltonian of this *super-block* consists of the Hamiltonian of both enlarged blocks and an additional interaction term:

$$\hat{H}_{supB} = \hat{H}_E + \hat{H}_{E'} + \hat{H}_{SS'}.$$

Step 4a Find the ground state
The ground state of super-block can be calculated by diagonalization.

Step 4b Calculating the reduced density matrix
Given the ground state $|\psi_G\rangle$, the reduced density matrix of the right enlarged block can be constructed:

$$\hat{\rho}_L = Tr_R |\psi_G\rangle\langle\psi_G|$$

Step 5 Renormalization of the enlarged block
The next step is to represent the new block using a reduced basis. This basis corresponds to the eigenstates with the largest eigenvalues of the reduced density matrix of the right enlarged block. This operation can be expressed in a rectangular matrix $\hat{O}_{L\to L+1}$ consisting of the eigenstates.

Step 6 Output
The new block for the next iteration of the algorithm is the block $B(L+1, m_{L+1})$

Step 7 Calculating the updated Hamiltonian
For next iteration of the algorithm, the Hamiltonians are updated using the same operator $\hat{O}_{L\to L+1}$.

Truncation error

It is easy to see that the error generated by the truncation step is:

$$\epsilon = \sum_{i>m} \lambda_i$$

Finite and infinite systems

In infinite systems this procedure can be repeated up to convergence. To account for finite systems, a "sweeping" procedure is necessary. This consists of repeating the above procedure in alternating directions, however, now reducing one block while enlarging the other. The sweep is repeated up to convergence.

5.2 Quantum chemistry

One of the main topics in quantum chemistry in the calculation of the energy of molecules, in particular the ground state energy. Since the spin system associated with a molecule is electron-based, tensor network methods were long thought not to be applicable because of the sign problem. The development of methods that do allow for the simulation of fermionic spins sparked a huge interest in the field by quantum chemists. We will look at the interpretation of molecules in the tensor network formalism.

5.2.1 Encoding molecules as tensor networks

There are several ways to encode a molecule to a tensor network. A molecule is usually interpreted in terms of its orbitals. Orbitals can be defined on several levels, but to limit the complexity of the description, the description in terms of one-electron orbitals in the most common in quantum chemistry. These orbitals can either contain an electron or be vacant. They are usually correlated to other orbitals, for example in the form of bonds: a shared electron pair between two orbitals.

One can imagine that a linear molecule can mapped to an MPS in a straightforward way. The assumption of low correlation (i.e. entanglement) between the orbitals that lay further apart is acceptable in most cases. Similarly, branched molecules can be mapped to TTNs. DMRG has been reimplemented for quantum chemical purposed by Chan *et al.* [CDG+08] and Hachmann *et al.* [HCC06], so the properties of interest can be calculated efficiently for these types of molecules.

Not all molecules have a structure that be approximated by a line or a tree. Furthermore, for high-precision results the correlations between all the orbitals have to be taken into account. Therefore, more complex tensor network structures are needed. An example is the interpretation of the complete-graph tensor network as the representation of a molecule. A CGTN representation of a molecule with k one-electron orbitals results is

$$|\Psi_{\mathrm{CGTN}}\rangle = \sum_{n_1 \dots n_k}^{2} \prod_{\alpha}^{k} \prod_{\beta \leq \alpha} A_{\alpha\beta}^{n_\alpha n_\beta} |n_1 \dots n_k\rangle,$$

where the orbitals n_α, n_β can either be occupied ($|1\rangle$) or unoccupied ($|0\rangle$).

Chapter 6

Discussion and conclusion

In the previous chapters, different tensor network methods have been discussed. Comparing MPS, TTN and PEPS it is easy to see a hierarchical relation. An MPS can be interpreted as a TTN of degree two, and both MPS and TTN can be interpreted as PEPS. Vice versa a 2D PEPS can be mapped to an MPS at exponential cost, while a TTN can be mapped to an MPS at polynomial costs as shown in Section 2.2.4.

Two procedures have been shown for the conventional methods to calculate local expectation values: either the value can be calculated directly from the tensor by construction or the complete tensor network has to be contracted. The mentioned construction method depends on the Schmidt decomposition. For the decomposition to be well-defined, the graph has to be bipartite, i.e. it has to be tree-like. Furthermore, the contraction complexity was shown to be closely related to the tree-width, linking the other formalisms to the CTN formalism. This provides a strong indication that tree-like tensor networks are actually the broadest class of tensor networks for which the local expectation values still can be efficiently calculated exactly classically.

Comparing the suitability of the different methods for the purpose of efficient classical simulation of quantum computations, the CTN formalism is the most convenient method. This is not surprising since the formalism was specifically developed for this purpose. The circuit can be directly mapped to the tensor network, without any additional steps. If the main purpose is the efficient representation of a quantum state, with only a small number of updates after quantum operations and measurements, TTN is probably the best choice. An arbitrary quantum state without long-range correlations can be efficient represented in this state, while updates after quantum operations and local expectation values can still be calculated exactly.

In conclusion, several representations for quantum states based on tensor networks have been discussed in this thesis, including matrix product states, tree tensor networks and projected entangled-pairs states. These allow for the efficient classical representation of and operation on an interesting class of quantum states. The main factor of influence on the efficiency of all these methods is the amount and structure of the entanglement in the states.

In turned out that for tensor network structures up to the tree, operations can be simulated exactly and efficiently on the representations. Also, efficient construction methods for these tensor networks have been shown. For the 2D grid structure it was shown that the representation can be updated efficiently after unitary quantum operations. Local expectation values, however, can only be simulated approximately. Furthermore, no efficient construction method is known for this tensor network that construct a real grid. New developments have further expanded our understanding of the classes of states that can be simulated efficiently.

In this thesis several novel concepts were introduced. These include an update procedure of a 2D PEPS after unitary operations, the interpretation of unitary operations and a more efficient construction method in the teleportation picture and the result that 2D grid PEPS are strictly more expressive that linear PEPS.

Other than for classical simulation of quantum computations, the tensor network formalisms

mainly have applications in solid state physics, due to the exceptional precision of DMRG methods. Most of the formalisms have been developed as representations of spin systems. The adaptation of the methods to support fermions, has led to applications in the field of quantum chemistry and new TN methods specifically designed for chemical systems.

Nevertheless, a large number of open questions still remains. Most of all, it is still unclear exactly which class of systems can be simulated efficiently. While the efficiency of the methods discussed here are mainly dependent of the amount and structure of the entanglement of the state, there are methods known that do not depend on the entanglement, for example weighted graph states.

The construction methods considered in this thesis allow for the creation of a tree-like structure at most. More complex geometries, like a 2D grid can be constructed, but do not lead to the most efficient representation of the state. A construction method that truly uses the 2D character of the grid would give much insight in the true nature of the 2D grid state.

The approximate calculation of expectation values possible with 2D PEPS could be used to develop a method capable of simulating circuits that would otherwise be completely unsimulatable. One could also limit the allowed operations on a 2D PEPS to maintain a "canonical form" in which the local expectation values can be determined directly from the tensors. This would allow a limited class of quantum circuits to be simulated on a 2D grid PEPS.

For the enhanced formalisms, like MERA and SBS, procedures should be developed to efficiently update these representations after quantum operations.

Acknowledgements

Mijn dank gaat uit naar Harry voor het mogelijk maken van dit project – ondanks mijn onconventionele achtergrond –, zijn inbreng in het project, ideeën, kritische vragen, feedback en de mogelijkheid om naar de workshop *Tensor network methods for quantum chemistry* in Zürich te gaan. Ik heb heel veel geleerd.

Erik wil ik bedanken voor het regelen en mogelijk maken van dit project, zijn advies, enthousiasme en interesse, ook al was het onderwerp wat ver van zijn bed en voor zijn inzichten in de wereld van informatica-academici.

I'd like to thank Vid for his collaboration during this project. Your ideas, questions, feedback and especially your physical background have been invaluable.

Herman Haverkort wil bedanken voor het plaatsnemen in de commissie.

Jos, ook al was het maar even, wil ik bedanken voor mijn eerste informaticacollege al weer zoveel jaar geleden. Met het vak *Automaten en Formele talen* wist ik meteen dat ik de juiste minor had gekozen.

Ik heb tijdens mijn afstuderenproject maar liefst zes verschillende kamers mogen bezetten. Mijn kamergenoten op de TU/e (Paul, Helle, Harsh en Dennis) en mijn kamergenoten op het CWI (Jop, Bruno, David) wil ik bedanken voor hun gezelschap. Jop wil ik ook nog bedanken voor zijn hulp bij de tensor variant van de SVD en het Choi-Jamiolkowski isomorphisme. For this last bit, I'd also like to thank Christian for his insights. De andere leden van de FM/TOC/MDSE en PNA6 groep bedank ik ook voor hun gezelschap en andere inbreng tijdens het project.

Bibliography

[ABD07] S. Anders, H. J. Briegel, and W. Dür. A variational method based on weighted graph states. *New Journal of Physics*, 9(10):361–361, 2007.

[AG04] S. Aaronson and D. Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, November 2004.

[AKLT88] I. Affleck, T. Kennedy, E. H. Lieb, and H. Tasaki. Valence bond ground states in isotropic quantum antiferromagnets. *Communications in Mathematical Physics*, 115(3):477–528, 1988.

[ALM06] Dorit Aharonov, Zeph Landau, and Johann Makowsky. The quantum FFT can be classically simulated. *quant-ph/0611156*, November 2006.

[Bod86] H. L. Bodlaender. Classes of graphs with bounded tree-width. Technical Report RUU-CS-86-22, Department of Information and Computing Sciences, Utrecht University, 1986.

[BPW+08] M. C. Bañuls, D. Pérez-García, M. M. Wolf, F. Verstraete, and J. I. Cirac. Sequentially generated states for the study of two-dimensional systems. *Physical Review A*, 77(5):052306, May 2008.

[CDG+08] G. Kin-Lic Chan, J. J. Dorando, D. Ghosh, J. Hachmann, E. Neuscamman, H. Wang, and T. Yanai. An introduction to the density matrix renormalization group ansatz in quantum chemistry. In Jean Maruani, Christian Minot, Roy McWeeny, Yves G Smeyers, Stephen Wilson, W. N Lipscomb, I. Prigogine, H. gren, D. Avnir, J. Cioslowski, R. Daudel, E. K.U Gross, W. F.van Gunsteren, K. Hirao, I. Huba, M. P Levy, G. L Malli, P. G Mezey, M. A.C Nascimento, J. Rychlewski, S. D Schwartz, S. Suhai, O. Tapia, P. R Taylor, R. G Woolley, Stephen Wilson, Peter J Grout, Jean Maruani, Gerardo Delgado-Barrio, and Piotr Piecuch, editors, *Frontiers in Quantum Systems in Chemistry and Physics*, volume 18 of *Progress in Theoretical Chemistry and Physics*, pages 49–65. Springer Netherlands, 2008. 10.1007/978-1-4020-8707-3_4.

[CKUC09] H. J. Changlani, J. M. Kinder, C. J. Umrigar, and G. Kin-Lic Chan. Approximating strongly correlated spin and fermion wavefunctions with correlator product states. *0907.4646*, July 2009.

[CRRM06] G. De Chiara, M. Rizzi, D. Rossini, and S. Montangero. Density matrix renormalization group for dummies. *cond-mat/0603842*, March 2006. J. Comput. Theor. Nanosci. 5, 1277-1288 (2008).

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can Quantum-Mechanical description of physical reality be considered complete? *Physical Review*, 47(10):777, May 1935.

[FNW92] M. Fannes, B. Nachtergaele, and R. Werner. Finitely correlated states on quantum spin chains. *Communications in Mathematical Physics*, 144(3):443–490, March 1992.

[HCC06] Johannes Hachmann, Wim Cardoen, and Garnet Kin-Lic Chan. Multireference correlation in long molecules with the quadratic scaling density matrix renormalization group. *The Journal of Chemical Physics*, 125(14):144101, 2006.

[HKH+09] R. Hübener, C. Kruszynska, L. Hartmann, W. Dür, F. Verstraete, J. Eisert, and M. B. Plenio. Renormalization algorithm with graph enhancement. *Physical Review A*, 79(2):022317, February 2009.

[HND10] R. Hübener, V. Nebendahl, and W. Dür. Concatenated tensor network states. *New Journal of Physics*, 12(2):025004, 2010.

[IOD09] L. Isaev, G. Ortiz, and J. Dukelsky. Hierarchical mean-field approach to the J_1-J_2 heisenberg model on a square lattice. *Physical Review B*, 79(2):024409, January 2009.

[Ish89] C. J. Isham. *Lectures on groups and vector spaces for physicists.* World Scientific, 1989.

[Joz06] R. Jozsa. On the simulation of quantum circuits. *quant-ph/0603163*, March 2006.

[MBR+10] K. H. Marti, B. Bauer, M. Reiher, M. Troyer, and F. Verstraete. Complete-Graph tensor network states: A new fermionic wave function ansatz for molecules. *1004.5303*, April 2010.

[MLNV10] V. Murg, Ö. Legeza, R. M. Noack, and F. Verstraete. Simulating strongly correlated quantum systems with tree tensor networks. *1006.3095*, June 2010.

[MS05] I. L. Markov and Y. Shi. Simulating quantum computation by contracting tensor networks. *quant-ph/0511069*, November 2005. SIAM Journal on Computing, 38(3):963-981, 2008.

[MSBC09] F. Mezzacapo, N. Schuch, M. Boninsegni, and J. I. Cirac. Ground-State properties of quantum Many-Body systems: Entangled-Plaquette states and variational monte carlo. *0905.3898*, May 2009. New J. Phys. 11 (2009) 083026.

[NC00] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information.* Cambridge University Press, 2000.

[ÖR95] S. Östlund and S. Rommer. Thermodynamic limit of density matrix renormalization. *Physical Review Letters*, 75(19):3537, November 1995.

[Pen71] Roger Penrose. Applications of negative dimensional tensors. In *Combinatorial mathematics and its applications.* Academic Press, 1971. edited by D.J.A. Welsh.

[PVWC06] D. Perez-Garcia, F. Verstraete, M. M Wolf, and J. I Cirac. Matrix product state representations. *quant-ph/0608197*, August 2006. Quantum Inf. Comput. 7, 401 (2007).

[RP00] E. G. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Comput. Surv.*, 32(3):300–335, 2000.

[RS86] Neil Robertson and P. D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, September 1986.

[SDV05] Y. Shi, L. Duan, and G. Vidal. Classical simulation of quantum many-body systems with a tree tensor network. *quant-ph/0511070*, November 2005. Phys. Rev. A 74, 022320 (2006).

[SWVC07] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac. Computational complexity of projected entangled pair states. *Physical Review Letters*, 98(14):140506, April 2007.

[SWVC08] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac. Simulation of quantum Many-Body systems with strings of operators and monte carlo tensor contractions. *Physical Review Letters*, 100(4):040501, January 2008.

[TEV09] L. Tagliacozzo, G. Evenbly, and G. Vidal. Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law. *0903.5017*, March 2009. Phys. Rev. B 80, 235127 (2009).

[Val02] L. G. Valiant. Quantum circuits that can be simulated classically in polynomial time. *SIAM Journal on Computing*, 31(4):1229–1254, January 2002.

[VC04a] F. Verstraete and J. I Cirac. Renormalization algorithms for Quantum-Many body systems in two and higher dimensions. *cond-mat/0407066*, July 2004.

[VC04b] F. Verstraete and J. I. Cirac. Valence-bond states for quantum computation. *Physical Review A*, 70(6):060302, December 2004.

[Vid03] G. Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters*, 91(14):147902, October 2003.

[Vid05] G. Vidal. Entanglement renormalization. *cond-mat/0512165*, December 2005. Phys. Rev. Lett. 99, 220405 (2007).

[Vid06] G. Vidal. A class of quantum many-body states that can be efficiently simulated. *quant-ph/0610099*, October 2006. Phys. Rev. Lett. 101, 110501 (2008).

[VMC08] F. Verstraete, V. Murg, and J. I. Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143, 2008.

[VMH09] G. F. Viamontes, I. Markov, and J. P. Hayes. *Quantum Circuit Simulation*. Springer, 2009.

[VPC04a] F. Verstraete, D. Porras, and J. I. Cirac. Density matrix renormalization group and periodic boundary conditions: A quantum information perspective. *Physical Review Letters*, 93(22):227205, November 2004.

[VPC04b] F. Verstraete, D. Porras, and J. I Cirac. DMRG and periodic boundary conditions: a quantum information perspective. *cond-mat/0404706*, April 2004. Phys. Rev. Lett. 93, 227205 (2004).

[VWPC06] F. Verstraete, M. M Wolf, D. Perez-Garcia, and J. I Cirac. Criticality, the area law, and the computational power of PEPS. *quant-ph/0601075*, January 2006. Phys. Rev. Lett. 96, 220601 (2006).

[Whi92] Steven R. White. Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863, November 1992.

[Wil75] Kenneth G. Wilson. The renormalization group: Critical phenomena and the kondo problem. *Reviews of Modern Physics*, 47(4):773, October 1975.

[YS06] N. Yoran and A. J. Short. Classical simulation of Limited-Width Cluster-State quantum computation. *Physical Review Letters*, 96(17):170503, May 2006.