Numerical methods in Tensor Networks

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr. rer. nat.)

im Fachgebiet

Mathematik

vorgelegt

von Diplom-Mathematiker Stefan Handschuh

geboren am 13. August 1984 in Leipzig

Die Annahme der Dissertation wurde empfohlen von:

1. Professor Dr. Dr. h.c. Wolfgang Hackbusch (MPI MIS Leipzig)
2. Professor Dr. Daniel Kressner (EPF Lausanne, CH)

Die Verleihung des akademischen Grades erfolgt mit Bestehen

der Verteidigung am 14. Januar 2015 mit dem Gesamtprädikat magna cum laude

Bibliographische Daten

Numerical methods in Tensor Networks
Handschuh, Stefan
Universität Leipzig, Dissertation, 2014
142 Seiten, 43 Abbildungen, 48 Referenzen

Contents

Acknowledgements

This work has been created from 4/2010 till 10/2014 at the Max-Planck-Institute for Mathematics in the Sciences[1]. It was embedded in the research group *Scientific Computing*[2], which is lead by Prof. Dr. Dr. h.c. Wolfgang Hackbusch[3].

Most of all, I would like to express my gratefulness to Prof. Hackbusch for giving me the opportunity to perform the research that is the basis for this thesis. He has been always open to discuss issues which made research more joyful and easier.
The constructive atmosphere at the research group is also something that brought a relief of my daily work. Therefore, I would also like to thank all group members for numerous useful discussions.

Without the support of my family and friends I would not have been able to write this thesis, which makes me even more thankful towards them.

As this work heavily relies on the scientific libraries BLAS[4] and LAPACK[5] for the numerical experiments and on LaTeX, I also thank the numerous authors of these software packages.

Stefan Handschuh

Leipzig, October 2014

[1] Inselstr. 22, 04103 Leipzig, Germany
[2] `http://www.mis.mpg.de/scicomp/`
[3] `http://www.mis.mpg.de/scicomp/hackbusch.en.html`
[4] `http://www.netlib.org/blas/`
[5] `http://www.netlib.org/lapack/`

Chapter 1

Introduction

In this work we will – based on [1] – describe a generalization of commonly used approaches in representing a large amount of numerical data (that can be interpreted as a high dimensional object) using sums of *elementary tensors* (see Definition 1.1.5).

In physics, elements of a tensor space that are not able to be represented by one elementary tensor are called *entangled* states. We however will not only distinguish between elementary tensors and non-elementary tensors, but also classify the number of terms that is needed to represent an object of the *tensor space*. Within this classification we will describe a complex multi-graph based structure that will allow us to reach a level of flexibility, which is well suited to the currently available applications that use represented tensors.

To avoid misunderstandings, we give three very basic definitions in advance.

Definition (Lexicographical order)**.** *Let $\ell \in \mathbb{N}$ and $I_1, \dots, I_\ell$ be arbitrary strict totally ordered index sets with the relation symbol $\leqslant$. Then we define the* lexicographical order *of $I := (I_1, \dots, I_\ell)$ as the order that is induced by $\leqslant$ which is defined for $i, j \in I$ as*

$$i \leqslant j \Leftrightarrow \exists t \in \{1, \dots, \ell\} : i_k = j_k \ \forall k = 1, \dots, t-1, i_t \leqslant j_t.$$

Definition ($\mathbb{K}$)**.** *We will write $\mathbb{K}$ if we want to express that either $\mathbb{R}$ or $\mathbb{C}$ can be used.*

Definition (Matrix notation)**.** *Let A be a matrix. Then*

$$A_{i,j}$$

describes the entry at row i and column j of matrix A. In addition, we have

$$A = (A_{i,j})$$

which will only hold for the most outer brackets of a term such that for instance in case of writing

$$((B_{i,j})_{j,i})$$

we simply mean the transpose of matrix B. This will be the matrix notation *that we will use in this work.*
Note that i and j may also be multi-indices.

1.1 Tensor product and tensor spaces

Our main point of interest is the so called *tensor product* of elements of linear vector spaces as well as of the vector spaces themselves, which we call *tensor space*. We will derive the definition from *multilinear mappings* and describe properties that have an important meaning in practice.

As the introduction into tensor products and tensor spaces is a standard section, we are guided by the structure and contents of [2, Chapter I], [3, Chapter I] and [4, Chapter 1].

We start with a basic definition as of [3, Chapter I Subsection 1.1]:

Definition 1.1.1 (Bilinear mapping, img)**.** *Let A, B and C be linear vector spaces over $\mathbb{K}$. Then the mapping*

$$f : A \times B \to C$$

is called a bilinear mapping*, if it is linear in both components, i.e.*

$$f(\alpha_1 a_1 + \alpha_2 a_2, b) = \alpha_1 f(a_1, b) + \alpha_2 f(a_2, b)$$

and

$$f(a, \beta_1 b_1 + \beta_2 b_2) = \beta_1 f(a, b_1) + \beta_2 f(a, b_2)$$

for all $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathbb{K}, a, a_1, a_2 \in A$ and $b, b_1, b_2 \in B$. We further define the image of f as

$$\operatorname{img} f := \bigcup_{a \in A} \bigcup_{b \in B} \{f(a, b)\}.$$

A natural extension from a mapping that acts on two components, is one that acts on arbitrary (finitely) many components. Therefore, we state as of [3, Chapter I Subsection 1.3]

Definition 1.1.2 (Multilinear mapping)**.** *Let $d \in \mathbb{N}, \mathcal{V}_1, \ldots, \mathcal{V}_d$ and C be $\mathbb{K}$-linear vector spaces. Then the mapping*

$$f : \bigtimes_{\mu=1}^{d} \mathcal{V}_\mu \to C$$

is called a multilinear mapping *if it is linear in all d components, i.e. for every fixed $\mu \in \{1, \ldots, d\}$, we have*

$$\begin{aligned} f(a_1, \ldots, a_{\mu-1}, \alpha_1 a_{\mu,1} + \alpha_2 a_{\mu,2}, a_{\mu+1}, \ldots, a_d) =& \alpha_1 f(a_1, \ldots, a_{\mu-1}, a_{\mu,1}, a_{\mu+1}, \ldots, a_d) \\ &+ \alpha_2 f(a_1, \ldots, a_{\mu-1}, a_{\mu,2}, a_{\mu+1}, \ldots, a_d) \end{aligned}$$

for all $a_1 \in \mathcal{V}_1, \ldots, a_{\mu-1} \in \mathcal{V}_{\mu-1}, a_{\mu,1}, a_{\mu,2} \in \mathcal{V}_\mu, a_{\mu+1} \in \mathcal{V}_{\mu+1}, \ldots, a_d \in \mathcal{V}_d$ and for all $\alpha_1, \alpha_2 \in \mathbb{K}$. Analogously to Definition 1.1.1, we define the image of f as

$$\operatorname{img} f := \bigcup_{\mu=1}^{d} \bigcup_{a_\mu \in \mathcal{V}_\mu} \{f(a_1, \ldots, a_d)\}.$$

Based on this definition, we can define the object, that we previously described as one of our *main objects of interest*, which is the *tensor product*. This is, roughly speaking, a tuple of a multilinear mapping and its codomain that preserves certain commutativity properties. The tensor product is defined in [3, Definition 1.4 and Subsection 1.20 of Chapter I] as follows:

Definition 1.1.3 (Tensor product)**.** *Let $\mathcal{V}_1, \dots, \mathcal{V}_d$ and C, D be $\mathbb{K}$-linear vector spaces and f be a multilinear mapping*

$$f : \bigtimes_{\mu=1}^{d} \mathcal{V}_\mu \to C,$$

then the pair (f, C) is called tensor product *for $\mathcal{V}_1, \dots, \mathcal{V}_d$ if the following two properties hold*

1. *$C = \overline{\operatorname{span} \operatorname{img} f}$ (i.e. C is generated by the image of f)*
2. *for all multilinear mappings*

$$g : \bigtimes_{\mu=1}^{d} \mathcal{V}_\mu \to D$$

then there exists a linear mapping

$$h : D \to C$$

with $h \circ g = f$ (the connection between f, g and h is visualized in Figure 1.1).

Note that the closure of $\operatorname{span} \operatorname{img} f$ *makes only sense if we have a topology defined. Therefore, in [5, Subsection 3.2.1] there is a distinction into algebraic tensor products (where no topology is defined) and topological tensor products.*
In this work, we will treat only finite dimensional vector spaces such that the topological tensor space and the algebraic tensor space are equal.

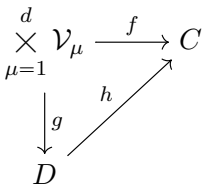

Figure 1.1: Commutativity between multilinear mappings of tensor products [1]

From [3, Definition 1.4] we state the following

[1] see [3, Chapter I Subsection 1.20] and [2, Theorem 1.1']

Lemma 1.1.4. *Let the notation of 1.1.3 hold, then (f, C) is a tensor product if and only if for all multilinear mappings*

$$g : \bigtimes_{\mu=1}^{d} \mathcal{V}_\mu \to D$$

there exists a unique linear mapping $h : D \to C$ such that $h \circ g = f$.

Proof. See [3, Definition 1.4 and Section 1.20 of Chapter I]. □

By combining [2, Definition 1.4] and [5, Definition 3.9], we get

Definition 1.1.5 ($\otimes$, elementary tensor)**.** *Let the notation of Definition 1.1.3 hold. Then we define the symbol $\otimes$ as follows:*

$$\bigotimes_{\mu=1}^{d} a_\mu := f(a_1, \ldots, a_d)$$

and

$$\bigotimes_{\mu=1}^{d} \mathcal{V}_\mu := \overline{\operatorname{span} \operatorname{img} f}.$$

We further denote elements of $\bigotimes_{\mu=1}^{d} a_\mu$ as an elementary tensor *of $\bigotimes_{\mu=1}^{d} \mathcal{V}_\mu$.*

So we always have a multilinear mapping f in the background that acts on the vector spaces $\mathcal{V}_\mu$. The mapping itself does not matter, but only the properties of Definition 1.1.3 and Lemma 1.1.4, respectively.

As of [5, Equation (1.1) of Subsection 1.1.1] and [6, Equation (3.3) of Subsection 3.2.1] we state

Example 1.1.6. *If the vector spaces $\mathcal{V}_1, \ldots, \mathcal{V}_d$ are all $\mathbb{K}$ vector spaces, then we can define the multilinear mapping f of Definition 1.1.5 pointwise by*

$$f(a_1, \ldots, a_d)_{i_1,\ldots,i_d} := \prod_{\mu=1}^{d} (a_\mu)_{i_\mu}$$

where $(a_\mu)_{i_\mu}$ denotes the i_μth position of vector a_μ. This is the standard definition of the tensor product of vectors over $\mathbb{K}$.

An important property of the tensor product is the isomorphism between different orderings of tensor products of vector spaces and their elements.

Combining [2, Proposition 1.5 and Proposition 1.7] leads to

Lemma 1.1.7. *Let $\mathcal{V}_1, \mathcal{V}_2$ and $\mathcal{V}_3$ be vector spaces over $\mathbb{K}$. Then the following properties hold:*

1. $\mathcal{V}_1 \otimes \mathcal{V}_2 \cong \mathcal{V}_2 \otimes \mathcal{V}_1$
2. $\mathcal{V}_1 \otimes \mathcal{V}_2 \otimes \mathcal{V}_3 \cong \mathcal{V}_1 \otimes (\mathcal{V}_2 \otimes \mathcal{V}_3)$.

Proof. See [2, Proposition 1.5 and Proposition 1.7]. □

Consequently, we obtain as of [2, Proposition 1.8]

Corollary 1.1.8. *Let $\mathcal{V}_1, \mathcal{V}_2$ and $\mathcal{V}_3$ be vector spaces over the body $\mathbb{K}$, then*

$$(\mathcal{V}_1 \otimes \mathcal{V}_2) \otimes \mathcal{V}_3 \cong \mathcal{V}_1 \otimes (\mathcal{V}_2 \otimes \mathcal{V}_3).$$

For vector spaces, we will now state a lemma that we will use implicitly through the whole work which is given in [2, Proposition 1.2]:

Lemma 1.1.9. *Let $\mathcal{V}_1, \dots, \mathcal{V}_d$ be finite dimensional $\mathbb{K}$ vector spaces. Then*

$$\dim \bigotimes_{\mu=1}^{d} \mathcal{V}_\mu = \prod_{\mu=1}^{d} \dim \mathcal{V}_\mu.$$

Proof. See [2, proof of Theorem 1.1]. □

1.2 Tensors as general vectors

In this work, all vector spaces $\mathcal{V}_1, \dots, \mathcal{V}_d$ are vectors spaces over $\mathbb{K}$ if not defined otherwise. With the help of Lemma 1.1.9, we can formulate a corollary whose content is stated in [5, Section 5.3]:

Corollary 1.2.1. *Let $\mathcal{V}_1, \dots, \mathcal{V}_d$ be $\mathbb{K}$ finite dimensional vector spaces. Then*

$$\bigotimes_{\mu=1}^{d} \mathcal{V}_\mu \cong \mathbb{K}^{\prod_{\mu=1}^{d} \dim \mathcal{V}_\mu}$$

since $\mathbb{K}$ is a field.

This corollary is of major importance as it allows us to rewrite tensors as vectors or matrices. In our algorithms and applications we will often use this corollary. Let us give a simple example, that is based on [5, Subsection 5.1.1].

Example 1.2.2. *For simplicity, we start with the tensor product of two vectors $v \in \mathbb{K}^n$ and $w \in \mathbb{K}^m$ where $n, m \in \mathbb{N}$. Due to Corollary 1.2.1, we can interpret the space $\mathbb{K}^n \otimes \mathbb{K}^m$ as $\mathbb{K}^{n \cdot m}$ such that we can describe $v \otimes w$ in the following way:*

$$v \otimes w = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \otimes \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} \cong \begin{pmatrix} v_1 \cdot w_1 \\ \vdots \\ v_1 \cdot w_m \\ \vdots \\ v_n \cdot w_1 \\ \vdots \\ v_n \cdot v_m \end{pmatrix} \in \mathbb{K}^{n \cdot m}.$$

Since the space $\mathbb{K}^n \otimes \mathbb{K}^m$ is only defined up to an isomorphism and $\mathbb{K}^{n\cdot m} \cong \mathbb{K}^{n\times m}$, we can also define the tensor product of v and w as a matrix

$$\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \otimes \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} \cong \begin{pmatrix} v_1 \cdot w_1 & \dots & v_1 \cdot w_m \\ v_2 \cdot w_1 & \dots & v_2 \cdot w_m \\ \vdots & & \vdots \\ v_n \cdot w_1 & \dots & v_n \cdot w_m \end{pmatrix} \in \mathbb{K}^{n\times m}.$$

However, the interpretation of the tensor product of two vectors depends on the application and has a huge influence on the numerical treatment and complexity.

In later sections, we will see that it is very important to interpret the tensor product of vectors differently. In fact, a lot of algorithms that we describe in this work, depend on the ability to re-order or re-interpret the given data such that it can be treated efficiently. Some basic concepts, such as vectorization (see [5, Section 5.1]), matricization (see [5, Section 5.2]) and tensorization (see [5, Section 5.3 and Chapter 14]), also rely on the re-ordering of the entries.

1.3 Tensor product of infinite dimensional spaces

Everything that we stated until now besides Lemma 1.1.9 and Corollary 1.2.1 did not require finiteness of the dimensions of the vector spaces $\mathcal{V}_\mu$. From the theoretical point of view, it is important to treat the infinite dimensional case slightly differently from the finite dimensional one.

In the infinite dimensional case, we can express the tensor product for each element of the vector space. As of [5, Remark 3.59] we have

Example 1.3.1. *Let us assume that the vector spaces $\mathcal{V}_1, \dots, \mathcal{V}_d$ of Definition 1.1.3 are all $L^2([a,b])$ with $a, b \in \mathbb{R} : a < b$. Then the tensor product can be defined pointwise. That is, for $f_i \in L^2([a,b])$ with $i = 1, \dots, d$, we have*

$$\left(\bigotimes_{\mu=1}^{d} f_\mu\right)(x_1, \dots, x_d) := \prod_{\mu=1}^{d} f_\mu(x_\mu)$$

for all $x_1, \dots, x_d \in [a,b]$.

Analogously to this example, we can also define the tensor product pointwise for finite dimensional vector spaces (see Example 1.1.6). For the numerical treatment, we have to convert the infinite dimensional spaces into finite ones. This is usually done by a projection to a finite dimensional subspace. This justifies

Remark 1.3.2. *For all numerical treatment and the description of the algorithms, we will assume the finiteness of the tensor spaces dimension and therefore the finiteness of the underlying vector spaces' dimension.*

1.4 Motivation

Keeping in mind Remark 1.3.2, we will now motivate the dealing with tensor networks on a very basic level. We want to point out the reason for treating high dimensional data as tensor network representations by giving a simple artificially constructed example. At first, [3, Chapter I Section 1.5] contains the content of

Theorem 1.4.1. *Let $\mathcal{V}_1, \dots, \mathcal{V}_d$ be finite dimensional vector spaces. Then each element of*

$$\mathcal{V} := \bigotimes_{\mu=1}^{n} \mathcal{V}_\mu$$

can be represented as a finite sum of elementary tensors of $\mathcal{V}$. That is

$$\forall a \in \mathcal{V}\ \exists r \in \mathbb{N}, a_{\mu,1}, \dots, a_{\mu,r} \in \mathcal{V}_\mu \forall \mu \in \{1, \dots, d\} : a = \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} a_{\mu,i}.$$

Proof. Follows directly from the finite dimension of $\mathcal{V}, \mathcal{V}_1, \dots, \mathcal{V}_d$ with the help of Corollary 1.2.1. □

The structure of the sum of elementary tensors is important and this is also the objective that we want to address: finding a *clever* structure of the sum can help reduce the computational and storage complexity of the element of the tensor space $\mathcal{V}$ that we want to represent. This also helps to deal with the so called *curse of dimensionality* which is the exponential cost w.r.t. d. We will give a short example that will emphasize the importance of the representation structure.

Example 1.4.2. *Let us consider a* 27×3 *matrix* A *with real valued entries:*

$$A := \begin{pmatrix} 5.0749 & 4.1027 & 3.0527 \\ 6.7196 & 5.5868 & 5.0496 \\ 6.2299 & 5.0837 & 3.7979 \\ 3.6702 & 2.9646 & 2.2221 \\ 4.3273 & 3.6029 & 3.3677 \\ 4.2061 & 3.4233 & 2.6278 \\ 4.8387 & 3.9141 & 2.8971 \\ 6.4658 & 5.3754 & 4.8326 \\ 5.9407 & 4.8581 & 3.6509 \\ 2.3474 & 1.8982 & 1.4092 \\ 3.1186 & 2.5928 & 2.3384 \\ 2.8806 & 2.3528 & 1.7626 \\ 1.5195 & 1.2285 & 0.9135 \\ 1.815 & 1.511 & 1.4002 \\ 1.7387 & 1.4201 & 1.1013 \\ 2.0438 & 1.6534 & 1.2229 \\ 2.6417 & 2.1971 & 1.9899 \\ 2.4503 & 2.0049 & 1.5266 \\ 3.1872 & 2.5776 & 1.9116 \\ 4.1868 & 3.4814 & 3.1462 \\ 3.8762 & 3.1676 & 2.3868 \\ 2.1159 & 1.7097 & 1.2777 \\ 2.5016 & 2.0828 & 1.9416 \\ 2.4201 & 1.9723 & 1.5209 \\ 2.869 & 2.321 & 1.7165 \\ 3.7805 & 3.1435 & 2.8339 \\ 3.4851 & 2.8513 & 2.1566 \end{pmatrix}$$

which has rank 3*. Therefore, its storage cost is* 81*. We can represent this matrix by*

$$\tilde{A} = \sum_{j_1,j_2,j_3=1}^{3,3,3} v_1(j_1) \otimes v_2(j_1, j_2) \otimes v_3(j_2, j_3) \otimes v_4(j_3),$$

where

$$
\begin{aligned}
v_1(1) &:= (0.8\ 0.4\ 0.6)^T \\
v_1(2) &:= (0.8\ 0.5\ 0.5)^T \\
v_1(3) &:= (0.9\ 0.2\ 0.4)^T \\
v_2(1,1) &:= (1.0\ 0.2\ 0.7)^T \\
v_2(2,1) &:= (0.6\ 0.5\ 0.4)^T \\
v_2(3,1) &:= (0.3\ 0.0\ 0.9)^T \\
v_2(1,2) &:= (0.8\ 0.5\ 0.5)^T \\
v_2(2,2) &:= (0.3\ 0.4\ 0.6)^T \\
v_2(3,2) &:= (0.4\ 0.9\ 0.2)^T \\
v_2(1,3) &:= (0.3\ 0.2\ 0.4)^T \\
v_2(2,3) &:= (0.8\ 0.2\ 0.3)^T \\
v_2(3,3) &:= (0.6\ 0.5\ 0.8)^T \\
v_3(1,1) &:= (0.3\ 0.6\ 0.1)^T \\
v_3(2,1) &:= (0.8\ 0.9\ 0.9)^T \\
v_3(3,1) &:= (0.9\ 0.9\ 0.8)^T \\
v_3(1,2) &:= (0.4\ 0.8\ 0.8)^T \\
v_3(2,2) &:= (0.4\ 0.7\ 0.5)^T \\
v_3(3,2) &:= (0.2\ 0.8\ 0.2)^T \\
v_3(1,3) &:= (0.8\ 0.8\ 1.0)^T \\
v_3(2,3) &:= (0.6\ 0.2\ 0.4)^T \\
v_3(3,3) &:= (0.1\ 0.6\ 0.9)^T \\
v_4(1) &:= (0.9\ 0.7\ 0.5)^T \\
v_4(2) &:= (0.5\ 0.5\ 0.9)^T \\
v_4(3) &:= (0.9\ 0.7\ 0.2)^T,
\end{aligned}
$$

and the connection

$$A = \Big(\mathrm{reshape}(\tilde{A}, 3, 27)\Big)^T$$

(with the reshape *notation of matlab and similar products). Storing $v_1, \ldots, v_4$ only requires* 72 *entries. If we increase the dimension of tensor $\tilde{A}$ this advantage may even get larger.*

This little example gives us enough motivation to find *clever* structures for the sums of elementary tensors for higher dimensions. By considering structured sums, we can not only reduce the storage cost, but also the computational effort that is needed to

compute one entry or perform basic linear algebra operations, i.e. abstract matrix vector multiplications (see [5, Section 4.6]).

Example 1.4.2 fits in the general framework that we want to describe in this work. To be precise, the way of representing the tensor $\tilde{A}$ is called *Matrix Product State* or *Tensor Train*, see Definition 2.1.8.

The upcoming Chapter 3 will introduce an approach that generalizes summation structures for finitely arbitrarily many dimensions. We will try to be as general as possible in order to include many major generalization attempts for the sum of elementary tensors as of Theorem 1.4.1.

1.5 Structure of this work

After this introductory chapter, we will make a brief excursion into the history of the current state of the art tensor representations in Chapter 2. There, we will also describe the connection between these different types of tensor representations and give algorithms for the conversion of one type of representation into the other.

Chapter 3 will finally introduce the generalized tensor representation that will cover all representations that are introduced in Chapter 2. We will also state the strong connection to the r-term representation (sum of elementary tensors). This chapter will be the foundation of the whole work. Having understood the concept behind the generalization, we can make interesting statements about represented tensors without worrying about the precise structure.

Subsequently, we are going to describe approximation algorithms for the tensor network format (i.e. algorithms that deal with tensor network representations). Non-linear block Gauss-Seidel methods and an approach about finding an initial guess. Due to the complex structure of the approach, this chapter will be of very technical nature. It should be sufficient though to understand the general idea behind the algorithms, as it is more easy to write the algorithms in detail than to read them.

Chapter 5 will focus on the description of an algorithm that helps to convert different tensor network representations into each other.

An important chapter will be Chapter 6, where we will describe general problems in dealing with represented tensors. Due to the complexity of the tensor network representation, we will stumble upon a lot of difficulties and numerical problems. As soon as such a problem arises, we will emphasize it and try to precisely describe the origin.

Since the tensor network approach does not cover all kinds of existing tensor representations, Chapter 7 will give a short overview of selected tensor representations that do not fit in our scheme. The list however, will by no means be complete.

For numerical experiments, the implementation is always also an important factor when it comes to measuring efficiency and runtime. Therefore, we will dedicate a whole chapter to the implementation, which is Appendix A, where we also will try to justify several peculiarities of the created source code. The interested reader is encouraged to try out the examples since they are useful in helping to understand the theory.

Wherever we introduce or describe a numerical method, we will give numerical examples in order to emphasize the statements that we make during the description. This should help the reader to get an impression about where the method can be applied in practice.

At this point we want to explicitly point out that every application is special. Therefore the structure of the tensor representation should be chosen such that it fits the setting where the data comes from. So one should always try to ask the question of whether the numerical structure is reasonable with respect to the real world. Long story short: There is no *one best* way to generally represent tensors. Everything depends on the application. There is a broad set of applications equipped with well working methods of representing tensors. The known working tensor based solutions may serve as an indicator on what structure may be used for a specific class of problems. [7] gives a very extensive overview about currently used methods.

Chapter 2

Overview over existing tensor formats

In order to properly justify the new approach that is used in this thesis, we will explain different ways of representing a tensor. This chapter's main part is the description of various popular tensor formats and the explanation of the difference between a tensor format and a tensor representation, which is important (see introduction into Section 2.1). We will also describe, how to convert the representation of a tensor into another representation of the same tensor (i.e. the tensor itself is not changed, but its representation).

2.0 Notation

The upcoming notation is valid for the rest of this work.

We will denote $\mathbb{K}$ vector spaces with dimension $n_\mu \in \mathbb{N}$ as

$$\mathcal{V}_\mu$$

for $\mu \in \{1, \dots, d\}$ with $d \in \mathbb{N}$. Furthermore, we define

$$\mathcal{V} := \bigotimes_{\mu=1}^{d} \mathcal{V}_\mu$$

as of Definition 1.1.5. There will be also mappings

$$f : \mathbb{N}^j \to \mathcal{V}_\mu$$

and

$$f : \mathbb{N}^i \to \mathbb{K}$$

for some $j, i \in \mathbb{N}$, which we define to have finite support, compare [1, Notation 2.1].

To increase the readability, we will support the definition of the tensor formats and the explanation of the tensor representation conversions by diagrams, which are different from the diagrams introduced by [8]. These diagrams will also have a sort of notation that would allow us to define the formats just by drawing a picture (we will make proper definitions however). Since our approach is (multi)graph-based, we have edges and vertices. We will use the graphical notation of [1] in a sightly modified way (the modification is only with respect to the vertex symbols):
Each vertex represents a mapping into a vector space. As in [1, Example 2.2], we will distinguish between spatial vertices and scalar vertices. The spatial vertices represent mappings into $\mathbb{K}$ vector spaces with dimension $\geqslant 1$ whereas the scalar vertices represent mappings into $\mathbb{K}$. This difference is also visualized in the graphs, see Figure 2.1. In most cases, we will not distinguish between the vertex and the mapping that is represented by the vertex.

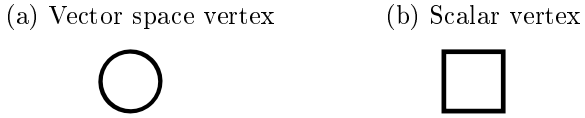

Figure 2.1: Symbols for different vertex types

Since we want to explain summation structures of elementary tensors, we have to visualize summations. In the diagrams, we have an edge between vertices, where there is a summation. Each edge will be labeled with the summation index although the summation index only exists within the summation. This way, the diagrams are supposed to be as close as possible to the formulas. In [1], the edges are labeled in the same way as well as the vertices.

A short and very simple example is Figure 2.2 where we have a vector space vertex (labeled v) and a scalar vertex (labeled w). The connection between both vertices is equivalent to a common summation index of the mappings (that are represented by the vertices). So if we name the mappings equivalent to their corresponding vertices, we get the following formula

$$\sum_j v(j)w(j)$$

where $v : \mathbb{N} \to \mathcal{V}$ and $w : \mathbb{N} \to \mathbb{K}$ have a finite support.

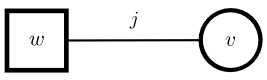

Figure 2.2: Example graph representation [1]

[1] compare [1, Fig. 1]

2.1 Formats

In each subsection, we are going to give a short introduction into the format, state its main properties and name advantages and disadvantages where necessary.

A tensor format is - roughly speaking - a summation structure for the sum of elementary tensors without limiting the number of terms of the sum. We impose however, the finiteness of the support, such that there is only a finite number of non-zero terms. So basically a tensor format describes a summation structure. Thus, a tensor does not *have* a format, it just may be *represented* in a certain format, which means that one tensor can be represented in different formats (i.e. the tensor itself does not change). For a general definition of the term *tensor format*, we refer to Definition 3.2.3.

If we say *v is an XYZ formatted tensor* or *v is an XYZ tensor*, we want to express that v is given in the form of the tensor format XYZ. Analogously, if we say v is a XYZ representation, we mean that v is a tensor represented in the XYZ format.

2.1.1 r-term format

As a tensor format is a structure of *some* sum of elementary tensors in our context, one of the most simple tensor formats that we can describe is defined by a sum of independent elementary tensors.

The idea for the r-term format has been introduced in [9].

Definition 2.1.1 (r-term format)**.** *Let $d \in \mathbb{N}$, then the r-term format is defined as*

$$\bigtimes_{\mu=1}^{d} \{f : \mathbb{N} \to \mathcal{V}_\mu\} \to \mathcal{V}$$

$$(v_1, \dots, v_d) \mapsto \sum_j \bigotimes_{\mu=1}^{d} v_\mu(j).$$

The name r-term format therefore is well-founded as we are only considering mappings v_i which have finite support such that a tensor in the r-term format can be represented with $r \in \mathbb{N}$ (finitely many) elementary tensors.

We say, a tensor v is an r-term tensor (or CP tensor) (or tensor in r-term/CP representation) with representation rank $\tilde{r} \in \mathbb{N}$ if we represent v by

$$\sum_{j=1}^{\tilde{r}} \bigotimes_{\mu=1}^{d} v_\mu(j)$$

with $v_i : \{1, \dots, \tilde{r}\} \to \mathcal{V}_i \ \forall i \in \{1, \dots, d\}$.

Visualizing the r-term format leads us to Figure 2.3 which illustrates a graph that has d vertices that are connected by one edge (which we labeled j).

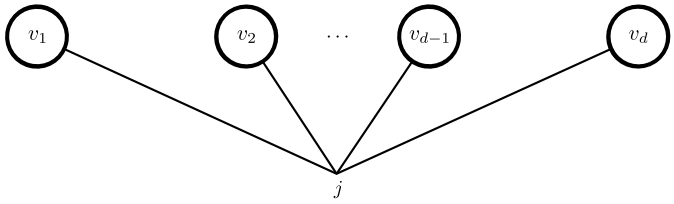

Figure 2.3: r-term tensor of order d

A overview of the r-term format can be found in [5, Chapter 7]. For application related use of this format, [4] is recommended. Note that this format is often also called *CP format* which stands for *canonical polyadic*. In the psychometrics and phonetics community, r-term tensors are often called CANDECOMP (**can**onical **decomp**osition, see [10]) and PARAFAC (**para**llel **fac**tors, see [11]), respectively. However, we will not use these alternative names.

The main advantage of this format is the linear storage dependency with respect to the dimension d. Since all addends are independent from each other, we can change and adjust them separately without taking care of the other addends. This will become a very important property later on (see Chapter 4). In contrast to [1], we include the r-term format into the framework of tensor networks.

Unfortunately, the manifold of r-term tensors with a certain fixed rank is not closed (see [12]). That is an important fact as it leads to numerical instability which has to be taken care of.

Finding an r-term representation that has the smallest possible value for r for a given tensor is an *ill-posed* problem, as shown in [12].

2.1.2 Tucker format / Subspace format

In the previous subsection, we mentioned the difficulties and numerical problems that appear when using the r-term format. We can overcome these by changing the structure of the representation. Separating the dimensions from each other (such that they do not share a summation index) by introducing a so called *core tensor* leads to the Tucker format, which has been introduced in [13]:

Definition 2.1.2 (Tucker format, Subspace format, Tucker legs, Tucker core, Tucker tensor)**.** *Let $d \in \mathbb{N}$, then the* Tucker format *is defined as*

$$\left(\bigtimes_{\mu=1}^{d} \{f : \mathbb{N} \to \mathcal{V}_\mu\}\right) \times \left\{f : \mathbb{N}^d \to \mathbb{K}\right\} \to \mathcal{V}$$

$$(v_1, \ldots, v_d, w) \mapsto \sum_{j_1,\ldots,j_d} w(j_1, \ldots, j_d) \bigotimes_{\mu=1}^{d} v_\mu(j_\mu)$$

where w is the so called core tensor or Tucker core *and the v_i are the so called* Tucker legs*. This format is also referred to as the* Subspace format*, see [5, Chapter 8]. The graphical representation is as of Figure 2.4.*

We say, a tensor v is a Tucker tensor *(or a tensor in Tucker representation) with representation rank $(r_1, \dots, r_d) \in \mathbb{N}^d$, if we represent v by*

$$\sum_{j_1,\dots,j_d=1}^{r_1,\dots,r_d} w(j_1, \dots, j_d) \bigotimes_{\mu=1}^{d} v_\mu(j_\mu)$$

with $v_1, \dots, v_d, w$ as defined above.

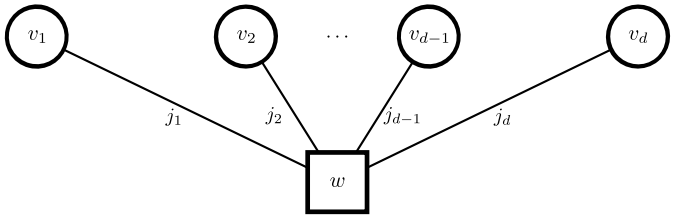

Figure 2.4: Tucker tensor of order d

From the technical point of view, this format seems to be not very feasible as the Tucker core w depends on all d summation indices and therefore has a storage cost which is exponentially in d (i.e. w can be considered to be a full order d tensor). In practice however, it turns out that the representation rank may be very small such that the support of w is also relatively small, compared to n^d. In [14] the Tucker core has been approximated by a different structure (to wit the TT format). See the next subsection for details. In contrary to the r-term format, the Tucker format is stable.

2.1.3 Tree-like formats

The problems of the r-term format have been solved with the Tucker format whereas new problems have been introduced such as possibly large ranks $r_1, \dots, r_d$ such that the storage cost for the Tucker core is exponentially in d. Representing the core tensor w of the Tucker format in a different structure, such that tree like shapes will be created, is an option to avoid the Tucker format's main disadvantage.

Remark 2.1.3. *Technically, the Tucker format as of Subsection 2.1.2 is also a tree structured format. However, due to its importance, we separated the description of this format.*

Just as the name suggests, the formats that we want to briefly describe in this subsection are based on a tree in the sense of graph theory. In contrast to the r-Term format, tree-like formats are closed as shown in [1, Theorem 3.2] (this also includes the Tucker format).

In [15], the hierarchical format has been introduced. We want to define a simplification of that format as follows:

Definition 2.1.4 (Hierarchical format (balanced))**.** *For simplicity, we assume that* $\mathbb{N} \ni d = 2^k$ *for some* $k \in \mathbb{N}$*. Then the* hierarchical tensor format (balanced) *is defined as*

$$\left(\bigtimes_{\mu=1}^{d} \{f : \mathbb{N} \to \mathcal{V}_\mu\}\right) \times \{f : \mathbb{N}^2 \to \mathbb{K}\} \times \left(\bigtimes_{\mu=2}^{d-1} \{f : \mathbb{N}^3 \to \mathbb{K}\}\right) \to \mathcal{V},$$

$$(v_1, \dots, v_d, w_1, \dots, w_{d-1}) \mapsto \sum_{j_1,\dots,j_{2d-2}} w_1(j_1, j_2) \prod_{\nu=2}^{k} \left(\prod_{\mu=2^{\nu-1}}^{2^\nu - 1} w_\mu(j_{\mu-1}, j_{2\mu-1}, j_{2\mu}) \right) \bigotimes_{\mu=1}^{d} v_\mu(j_{\mu+2^k-2}).$$

We say, a tensor v *is a* hierarchical tensor (balanced) *(or a tensor in* hierarchical (balanced) representation*) with representation rank* $(r_1, \dots, r_{2d-2}) \in \mathbb{N}^{2d-2}$ *if we represent* v *by*

$$\sum_{j_1,\dots,j_{2d-2}=1}^{r_1,\dots,r_{2d-2}} w_1(j_1, j_2) \prod_{\nu=2}^{k} \left(\prod_{\mu=2^{\nu-1}}^{2^\nu - 1} w_\mu(j_{\mu-1}, j_{2\mu-1}, j_{2\mu}) \right) \bigotimes_{\mu=1}^{d} v_\mu(j_{\mu+2^k-2})$$

with $w_1, \dots, w_{d-1}, v_1, \dots, v_d$ *as defined above. Compare [1, Example 2.7].*

Remark 2.1.5. *In comparison to the hierarchical format, a similar structure was introduced as tree tensor network in [16].*

The graph structure is illustrated in Figure 2.5. The general hierarchical format as of [15] allows general trees. For applications of this format, we refer to [6].

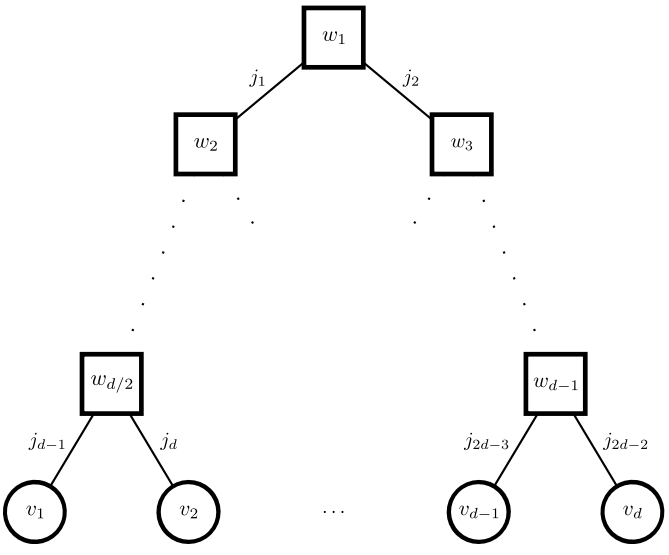

Figure 2.5: Hierarchical tensor (balanced) of order d

If we force the tree to have maximal depth, we still remain in the setting of the hierarchical format, but the visualization and numerical properties change and we end up with a structure as follows:

Definition 2.1.6 (Hierarchical format (linear))**.** *Let $d \in \mathbb{N}\backslash\{1\}$, then we define the* hierarchical tensor format (linear) *as*

$$\left(\bigtimes_{\mu=1}^{d} \{f : \mathbb{N} \to \mathcal{V}_\mu\}\right) \times \left(\bigtimes_{\mu=1}^{d-1} \{f : \mathbb{N}^2 \to \mathbb{K}\}\right) \to \mathcal{V},$$

$$(v_1, \dots, v_d, w_1, \dots, w_{d-1}) \mapsto \sum_{j_1,\dots,j_{2d-2}} w_1(j_1, j_2) \prod_{\mu=2}^{d-1} w_\mu(j_{2\mu-2}, j_{2\mu-1}, j_{2\mu}) \bigotimes_{\mu=1}^{d-1} v_\mu(j_{2\mu-1}) \otimes v_d(j_{2d-2}).$$

We say, a tensor v is a hierarchical tensor (linear) *(or a tensor in* hierarchical (linear) representation*) with representation rank $(r_1, \dots, r_{2d-2}) \in \mathbb{N}^{2d-2}$ if we represent v by*

$$\sum_{j_1,\dots,j_{2d-2}=1}^{r_1,\dots,r_{2d-2}} w_1(j_1, j_2) \prod_{\mu=2}^{d-1} w_\mu(j_{2\mu-2}, j_{2\mu-1}, j_{2\mu}) \bigotimes_{\mu=1}^{d-1} v_\mu(j_{2\mu-1}) \otimes v_d(j_{2d-2})$$

with $w_1, \dots, w_{d-1}$ and $v_1, \dots, v_d$ as defined above.

A graphical representation of this format can be found in Figure 2.6.

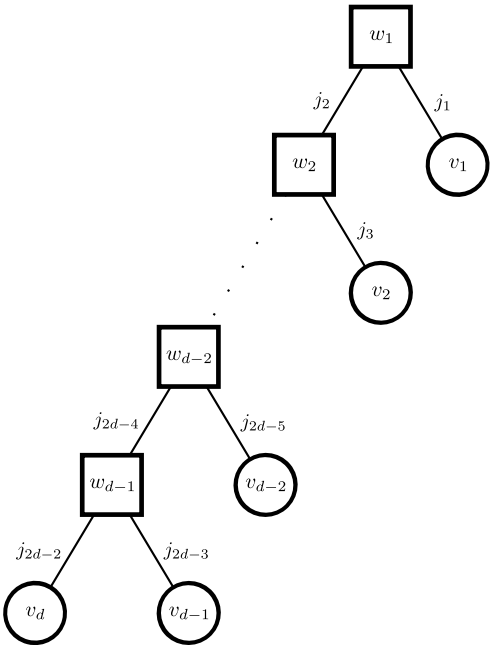

Figure 2.6: Hierarchical tensor (linear) of order d

Remark 2.1.7. *The definition of the general hierarchical format can be found in [5, Chapter 11] and in Example 3.3.7 of Chapter 3. In this section, we will neglect the general hierarchical format for the sake of simplicity. For our purposes, the balanced version of this format is sufficient.*

Having defined the linear hierarchical format, we also have to and want to give the definition of the equivalent so called TT (or Tensor Train) format that is introduced in [17], [18] and [19]. The difference to the hierarchical linear format is that the TT format integrates the scalar factors into the spatial vectors. That is, for $\mu \in \{2, \dots, d-1\}$ we redefine v_μ

$$v_\mu^{TT}(j_{2\mu-2}, j_{2\mu}) \mapsto \sum_{j_{2\mu-1}=1}^{r_{2\mu-1}} w_\mu(j_{2\mu-2}, j_{2\mu-1}, j_{2\mu}) v_\mu(j_{2\mu-1})$$

and v_1

$$v_1^{TT}(j_2) \mapsto \sum_{j_1=1}^{r_1} w_1(j_1, j_2) v_1(j_1)$$

such that we obtain new mappings v_μ^{TT} and the mappings $w_1, \dots, w_{d-1}$ vanish as well as the summations over $j_1, j_3, \dots, j_{2d-3}$. Therefore, we obtain

Definition 2.1.8 (TT format)**.** *Let $d \in \mathbb{N}$, then the* TT format *(or* Tensor Train format*) is defined as*

$$\{f : \mathbb{N} \to \mathcal{V}_1\} \times \left(\bigtimes_{\mu=2}^{d-1} \{f : \mathbb{N}^2 \to \mathcal{V}_\mu\} \right) \times \{f : \mathbb{N} \to \mathcal{V}_d\} \to \mathcal{V}$$

$$(v_1, \dots, v_d) \mapsto \sum_{j_1, \dots, j_{d-1}} v_1(j_1) \otimes \left(\bigotimes_{\mu=2}^{d-1} v_\mu(j_{\mu-1}, j_\mu) \right) \otimes v_d(j_{d-1}).$$

We say, a tensor v is a TT tensor *(or a* tensor in TT representation*) with representation rank $(r_1, \dots, r_{d-1}) \in \mathbb{N}^{d-1}$, if we represent v by*

$$\sum_{j_1, \dots, j_{d-1}=1}^{r_1, \dots, r_{d-1}} v_1(j_1) \otimes \left(\bigotimes_{\mu=2}^{d-1} v_\mu(j_{\mu-1}, j_\mu) \right) \otimes v_d(j_{d-1})$$

with $v_1, \dots, v_d$ as defined above. Compare [1, Example 2.7].

The name *train* is motivated by the visual representation that is shown in Figure 2.7.

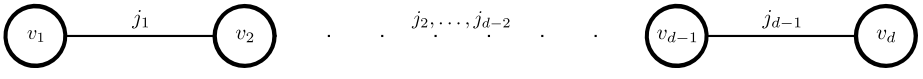

Figure 2.7: Tensor Train of order d

Remark 2.1.9. *This TT format is known to chemists as MPS (**m**atrix **p**roduct **s**tates), see [20].*

In [18, Section 3] the Tucker core is represented in the TT format. If we additionally interpret the Tucker legs as tensors where each vector space dimension is 2, we obtain the following format which has been introduced in [14]:

Definition 2.1.10 (QTT-Tucker format)**.** *Let $d, m_1, \dots, m_d \in \mathbb{N}$ where $\dim \mathcal{V}_\mu = 2^{m_\mu}$ for $\mu \in \{1, \dots, d\}$ and define $s_j = \sum_{i=1}^{j} m_i$ and $s_0 := 0$, then the QTT-Tucker format is defined as*

$$\begin{aligned}\bigtimes_{\mu=1}^{d} \left(\bigtimes_{\nu=1}^{m_\mu - 1} \{f : \mathbb{N}^2 \to \mathbb{K}^2\} \times \{f : \mathbb{N} \to \mathbb{K}^2\} \right) \times \{f : \mathbb{N}^2 \to \mathbb{K}\} \times \left(\bigtimes_{\mu=2}^{d-1} \{f : \mathbb{N}^3 \to \mathbb{K}\} \right) \\ \times \{f : \mathbb{N}^2 \to \mathbb{K}\} \to \mathcal{V},\end{aligned}$$

$$\begin{aligned}(v_1, \dots, v_{s_d}, w_1, \dots, w_d) \mapsto \sum_{j_1, \dots, j_{s_d + d - 1}} w_1(j_1, j_{m_1+1}) \left(\prod_{\mu=2}^{d-1} w_\mu(j_{s_{\mu-1}+\mu-1}, j_{s_{\mu-1}+\mu}, j_{s_\mu+\mu}) \right) \\ w_d(j_{s_{d-1}+d-1}, j_{s_{d-1}+d}) \\ \bigotimes_{\mu=1}^{d} \left(\left(\bigotimes_{\nu=s_{\mu-1}+1}^{s_\mu - 1} v_\nu(j_{\mu+\nu-1}, j_{\mu+\nu}) \right) \otimes v_{s_\mu}(j_{s_\mu+\mu-1}) \right),\end{aligned}$$

where the Q *stands for* quantized *(see [21, 22] for an introduction into the quantized TT format, [23] for an example of its advantages and [24] for an application).*

We say, a tensor v is a QTT-Tucker tensor *(or a tensor in* QTT-Tucker representation*) with representation rank $(r_1, \dots, r_{s^d+d-1}) \in \mathbb{N}^{s^d+d-1}$ if we represent v by*

$$\begin{aligned}\sum_{j_1, \dots, j_{s_d+d-1}=1}^{r_1, \dots, r_{s_d+d-1}} w_1(j_1, j_{m_1+1}) \left(\prod_{\mu=2}^{d-1} w_\mu(j_{s_{\mu-1}+\mu-1}, j_{s_{\mu-1}+\mu}, j_{s_\mu+\mu}) \right) w_d(j_{s_{d-1}+d-1}, j_{s_{d-1}+d}) \\ \bigotimes_{\mu=1}^{d} \left(\left(\bigotimes_{\nu=s_{\mu-1}+1}^{s_\mu - 1} v_\nu(j_{\mu+\nu-1}, j_{\mu+\nu}) \right) \otimes v_{s_\mu}(j_{s_\mu+\mu-1}) \right)\end{aligned}$$

with $w_1, \dots, w_d$ and $v_1, \dots, v_{s_d}$ as defined above.

The main difference to the so called *extended TT format* (see [18, Section 3]) is the artificial interpretation of the Tucker legs as $2 \times 2 \times \dots \times 2$ tensors (so called *quantics*).

Since the vector spaces $\mathcal{V}_\mu$ may have different dimensions for different μ, the Tucker legs (see Definition 2.1.2) can have different *length*. The general QTT-Tucker format is visualized in Figure 2.8. This figure is also valid for the extended TT format.

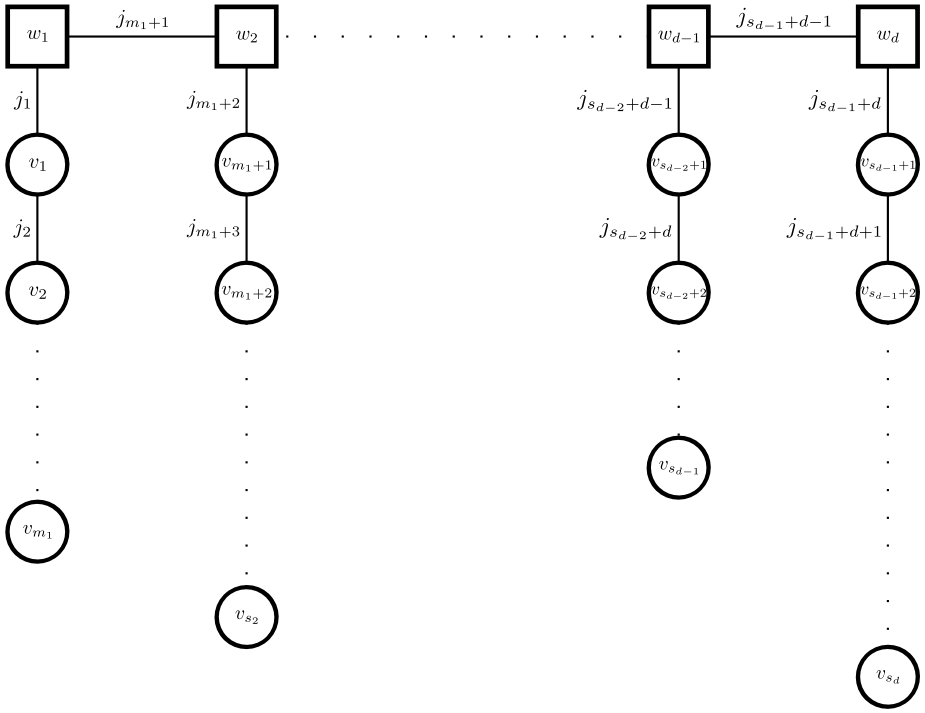

Figure 2.8: QTT-Tucker tensor of order d

2.1.4 Tensor Chain format

In the physicists community, there is the need to treat problems that have an underlying structure of a ring to respect periodic boundary conditions. In [21, Definition 2.1], the following format has been introduced:

Definition 2.1.11 (Tensor Chain format)**.** *Let $d \in \mathbb{N}$, then the* Tensor Chain format *(or* TC format*) is defined as*

$$\bigtimes_{\mu=1}^{d} \{f : \mathbb{N}^2 \to \mathcal{V}_\mu\} \to \mathcal{V},$$

$$(v_1, \dots, v_d) \mapsto \sum_{j_1,\dots,j_d} v_1(j_1, j_d) \otimes \bigotimes_{\mu=2}^{d} v_\mu(j_{\mu-1}, j_\mu).$$

We say, a tensor v is a TC tensor (or a tensor in TC representation) with representation rank $(r_1, \dots, r_d) \in \mathbb{N}^d$ if we represent v by

$$\sum_{j_1,\dots,j_d=1}^{r_1,\dots,r_d} v_1(j_1, j_d) \otimes \bigotimes_{\mu=2}^{d} v_\mu(j_{\mu-1}, j_\mu)$$

with $v_1, \dots, v_d$ as defined above.

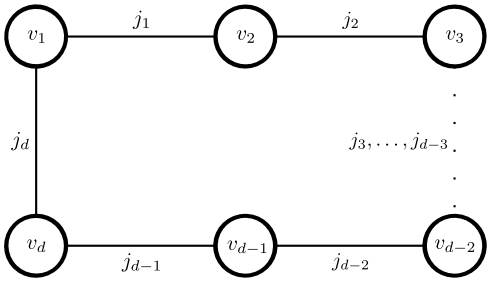

Figure 2.9: Tensor Chain of order d

Remark 2.1.12. *The Tensor chain format is not closed as shown in [25] and [26, Theorem 14.1.2.2] which leads to numerical instability.*

Remark 2.1.13. *The TT format as of Definition 2.1.8 is a special case of the TC format where the first summation index is fixed, see [21]. The reversal is not true.*

2.1.5 PEPS format

The PEPS format is a tensor format that is used in quantum chemistry together with various modifications, see for instance [27] for an introduction. PEPS stands for ***p**rojected **e**ntagled **p**air **s**tates*. An entangled state is understood to be a tensor that cannot be represented as an elementary tensor. The state is called *projected* because not the real physical state is used, but a projection to some subspace. The term *pair* refers to the entanglement being only considered in terms of maximally entangled state pairs (compare [28, Chapter 7]). Each dimension in this format also has a physical meaning. See [29] for applications.

Extending the TT format from a *one dimensional* to a *two dimensional* tensor structure in the shape of a $2D$ grid.

Definition 2.1.14 (PEPS format)**.** *Let* $d_1, d_2 \in \mathbb{N}$. *Then* PEPS format *of order* (d_1, d_2)

is defined as

$$\begin{aligned}
&\{f : \mathbb{N}^2 \to \mathcal{V}_1\} \times \bigtimes_{\mu=2}^{d_1-1} \{f : \mathbb{N}^3 \to \mathcal{V}_\mu\} \times \{f : \mathbb{N}^2 \to \mathcal{V}_{d_1}\} \\
&\qquad \times \bigtimes_{z=2}^{d_2-1} \left(\{f : \mathbb{N}^3 \to \mathcal{V}_{(z-1)d_1+1}\} \times \bigtimes_{\mu=2}^{d_1-1} \{f : \mathbb{N}^4 \to \mathcal{V}_{(z-1)d_1+\mu}\} \times \{f : \mathbb{N}^3 \to \mathcal{V}_{zd_1}\} \right) \\
&\qquad\qquad \times \{f : \mathbb{N}^2 \to \mathcal{V}_{(d_2-1)d_1+1}\} \times \bigtimes_{\mu=2}^{d_1-1} \{f : \mathbb{N}^3 \to \mathcal{V}_{(d_2-1)d_1+\mu}\} \times \{f : \mathbb{N}^2 \to \mathcal{V}_{d_1 d_2}\} \\
&\qquad\qquad\qquad \to \mathcal{V},
\end{aligned}$$

$$\begin{aligned}
(v_1, \dots, v_{d_1 d_2}) \mapsto \sum_{j_1,\dots,j_{2d_1d_2-d_1-d_2}} & v_1(j_{\downarrow\rightarrow}) \otimes \bigotimes_{\mu=2}^{d_1-1} v_\mu(j_{\leftarrow\rightarrow\downarrow}(\mu)) \otimes v_{d_1}(j_{\leftarrow\downarrow}) \\
& \otimes \bigotimes_{z=2}^{d_2-1} \left(v_{(z-1)d_1+1}(j_{\uparrow\rightarrow\downarrow}(z)) \otimes \bigotimes_{\mu=2}^{d_1-1} v_{(z-1)d_1+\mu}(j_{\uparrow\leftarrow\rightarrow\downarrow}(z,\mu)) \otimes v_{zd_1}(j_{\uparrow\leftarrow\downarrow}(z)) \right) \\
& \otimes v_{(d_2-1)d_1+1}(j_{\uparrow\rightarrow}) \otimes \bigotimes_{\mu=2}^{d_1-1} v_{(d_2-1)d_1+\mu}(j_{\uparrow\leftarrow\rightarrow}(\mu)) \otimes v_{d_1 d_2}(j_{\uparrow\leftarrow})
\end{aligned}$$

where

$$\begin{aligned}
j_{\rightarrow\downarrow} &:= (j_1, j_{d_1}) \\
j_{\leftarrow\rightarrow\downarrow}(\mu) &:= (j_{\mu-1}, j_\mu, j_{d_1-1+\mu}) \\
j_{\leftarrow\downarrow} &:= (j_{d_1-1}, j_{2d_1-1}) \\
j_{\uparrow\rightarrow\downarrow}(z) &:= (j_{(z-1)(d_1-1)+(z-2)d_1+1}, j_{(z-1)(2d_1-1)+1}, j_{z(d_1-1)+(z-1)d_1+1}) \\
j_{\uparrow\leftarrow\rightarrow\downarrow}(z,\mu) &:= (j_{(z-1)(d_1-1)+(z-2)d_1+\mu}, j_{(z-1)(2d_1-1)+\mu-1}, j_{(z-1)(2d_1-1)+\mu}, j_{z(d_1-1)+(z-1)d_1+\mu}) \\
j_{\uparrow\leftarrow\downarrow}(z) &:= (j_{(z-1)(d_1-1)+(z-2)d_1+d_1}, j_{(z-1)(2d_1-1)+d_1-1}, j_{z(d_1-1)+(z-1)d_1+d_1}) \\
j_{\uparrow\rightarrow} &:= (j_{(d_2-1)(d_1-1)+(d_2-2)d_1+1}, j_{(d_2-1)(2d_1-1)+1}) \\
j_{\uparrow\leftarrow\rightarrow}(\mu) &:= (j_{(d_2-1)(d_1-1)+(d_2-2)d_1+\mu}, j_{(d_2-1)(2d_1-1)+\mu-1}, j_{(d_2-1)(2d_1-1)+\mu}) \\
j_{\uparrow\leftarrow} &:= (j_{(d_2-1)(d_1-1)+(d_2-2)d_1+d_1}, j_{2d_1d_2-d_1-d_2})
\end{aligned}$$

such that we end up with the structure that is visualized in Figure 2.10. So the arrows $\uparrow, \leftarrow, \rightarrow$ *and* $\downarrow$ *of the mappings* $j_\cdot$ *indicate what kind of connection (in terms of the grid) the vertex at position* (z, μ) *has. The parameter* z *stands for the row and* μ *for the column where we omit the parameters if they are out of question.*

We say, a tensor v *is a* PEPS tensor *(or a tensor in* PEPS representation*) with*

representation rank $(r_1, \dots, r_{2d_1d_2-d_1-d_2}) \in \mathbb{N}^{2d_1d_2-d_1-d_2}$ *if we represent* v *by*

$$\sum_{j_1,\dots,j_{2d_1d_2-d_1-d_2}=1}^{r_1,\dots,r_{2d_1d_2-d_1-d_2}} v_1(j_{\downarrow\rightarrow}) \otimes \bigotimes_{\mu=2}^{d_1-1} v_\mu(j_{\leftarrow\rightarrow\downarrow}(\mu)) \otimes v_{d_1}(j_{\leftarrow\downarrow})$$
$$\otimes \bigotimes_{z=2}^{d_2-1} \left(v_{(z-1)d_1+1}(j_{\uparrow\rightarrow\downarrow}(z)) \otimes \bigotimes_{\mu=2}^{d_1-1} v_{(z-1)d_1+\mu}(j_{\uparrow\leftarrow\rightarrow\downarrow}(z,\mu)) \otimes v_{zd_1}(j_{\uparrow\leftarrow\downarrow}(z)) \right)$$
$$\otimes v_{(d_2-1)d_1+1}(j_{\uparrow\rightarrow}) \otimes \bigotimes_{\mu=2}^{d_1-1} v_{(d_2-1)d_1+\mu}(j_{\uparrow\leftarrow\rightarrow}(\mu)) \otimes v_{d_1d_2}(j_{\uparrow\leftarrow})$$

with $v_1, \dots, v_{d_1d_2}$ *as defined above. Compare [1, Example 2.7].*

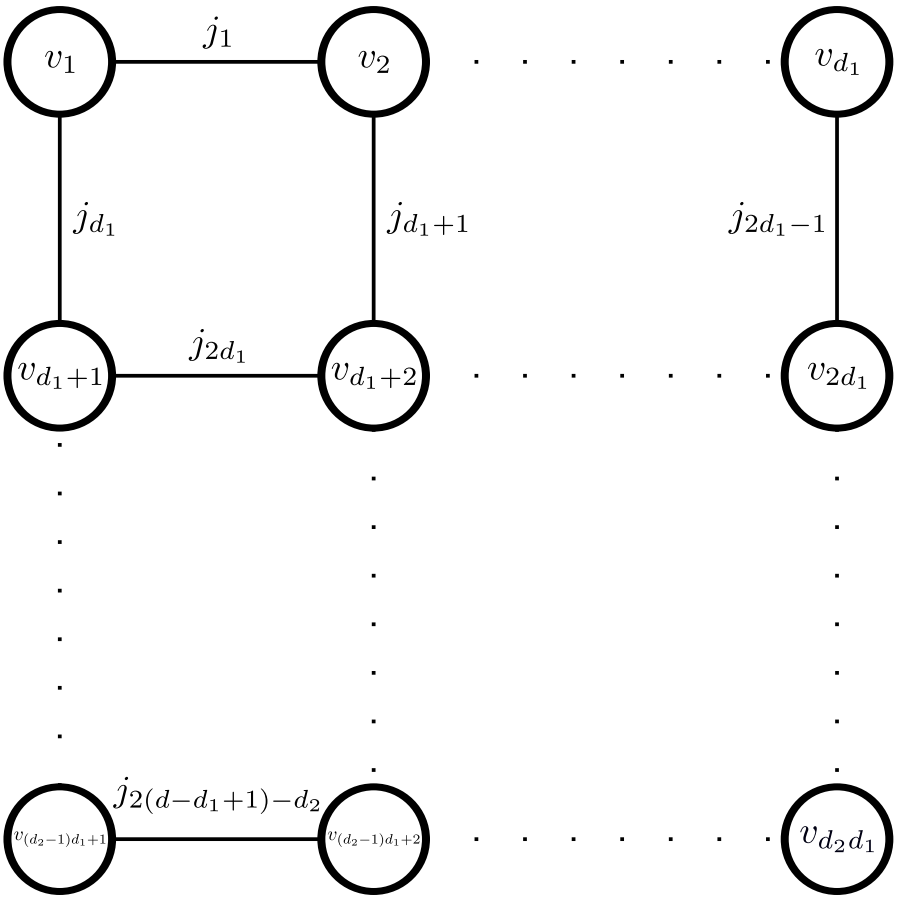

Figure 2.10: PEPS tensor of order (d_1, d_2)

A detailed overview over the PEPS format can be found in [28].

2.2 Conversion into other formats

This section will contain algorithms that describe the exact (i.e. no approximation) conversion between the described representations up to a certain extend. Note that we will only describe the conversions for represented tensors (i.e. tensor formats with a representation rank, see Definition 3.2.5). Therefore, we are not exactly converting formats, but representations of a certain format into representations of other formats.

We neglect the algorithm that will be proposed in Chapter 5, which can in general be used to perform the conversion.

In order to shorten the notation that we need, we first state the tensors that are going to be converted. The following representations will be used in the subsequent subsections:

- r-term representation with representation rank (r^{CP})

$$v^{CP} = \sum_{j=1}^{r^{CP}} \bigotimes_{\mu=1}^{d} v_\mu^{CP}(j),$$

  with $v_1^{CP}, \dots, v_d^{CP}$ as of Definition 2.1.1

- Tucker representation with representation rank $(r_1^T, \dots, r_d^T)$

$$v^T = \sum_{j_1,\dots,j_d=1}^{r_1^T,\dots,r_d^T} w^T(j_1, \dots, j_d) \bigotimes_{\mu=1}^{d} v_\mu^T(j_\mu)$$

  with $w^T, v_1^T, \dots, v_d^T$ as of Definition 2.1.2

- Hierarchical (balanced) representation with representation rank $(r_1^H, \dots, r_{2d-2}^H)$ and $2^k = d$

$$v^H = \sum_{j_1,\dots,j_{2d-2}=1}^{r_1^H,\dots,r_{2d-2}^H} w_1^H(j_1, j_2) \prod_{\nu=2}^{k} \left(\prod_{\mu=2^{\nu-1}}^{2^\nu-1} w_\mu^H(j_{\mu-1}, j_{2\mu-1}, j_{2\mu}) \right) \bigotimes_{\mu=1}^{d} v_\mu^H(j_{\mu+2^k-2})$$

  with $w_1^H, \dots, w_{2^k-1}^H, v_1^H, \dots, v_d^H$ as of Definition 2.1.4

- TT representation with representation rank $(r_1^{TT}, \dots, r_{d-1}^{TT})$

$$v^{TT} = \sum_{j_1,\dots,j_{d-1}=1}^{r_1^{TT},\dots,r_{d-1}^{TT}} v_1^{TT}(j_1) \otimes \left(\bigotimes_{\mu=2}^{d-1} v_\mu^{TT}(j_{\mu-1}, j_\mu) \right) \otimes v_d^{TT}(j_{d-1})$$

  with $v_1^{TT}, \dots, v_d^{TT}$ as of Definition 2.1.8

- TC representation with representation rank $(r_1^{TC}, \dots, r_d^{TC})$

$$v^{TC} = \sum_{j_1,\dots,j_d}^{r_1^{TC},\dots,r_d^{TC}} v_1^{TC}(j_1, j_d) \otimes \bigotimes_{\mu=2}^{d} v_\mu^{TC}(j_{\mu-1}, j_\mu)$$

  with $v_1^{TC}, \dots, v_d^{TC}$ as of Definition 2.1.11

So any time we reference $v_3^{CP}(4)$ for instance, we mean the vector of direction 3 of the 4th addend of the r-term tensor.

2.2.1 r-term to Tucker

For this conversion, we set

$$\begin{aligned}(r_1^T, \dots, r_d^T) &:= (r^{CP}, \dots, r^{CP}),\\ v_\mu^T &:= v_\mu^{CP}\end{aligned}$$

and

$$w^T(j_1, \dots, j_d) := \begin{cases} 1 & \text{if } j_1 = \dots = j_d \\ 0 & \text{else.} \end{cases}$$

2.2.2 r-term to hierarchical (balanced)

Here we set

$$r_\mu^H := r^{CP} \ \forall \mu \in \{1, \dots, j_{2d-2}\}$$

and for all fixed $(i, j, k) \in \{1, \dots, r^{CP}\}^3$

$$\begin{aligned}v_\mu^H(i) &:= v_\mu^{CP}(i),\\ w_1^H(i, j) &:= \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases}\end{aligned}$$

and

$$w_\mu^H(i, j, k) := \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{else} \end{cases} \quad \forall \mu \in \{2, \dots, d-1\}.$$

2.2.3 r-term to TT

We set

$$\begin{aligned}(r_1^{TT},\dots,r_{d-1}^{TT}) &:= (r^{CP},\dots,r^{CP}),\\ v_1^{TT}(j_1) &:= v_1^{CP}(j_1),\\ v_\mu^{TT}(j_{\mu-1},j_\mu) &:= \begin{cases} v_\mu^{CP}(j_\mu) & \text{if } j_{\mu-1}=j_\mu\\ 0 & \text{else}\end{cases}\end{aligned}$$

and

$$v_d^{TT}(j_{d-1}) := v_d^{CP}(j_{d-1}).$$

2.2.4 Tucker to r-term

Define

$$i = \operatorname*{argmax}_{\mu=1,\dots,d} r_\mu^T,$$

then we set

$$r^{CP} := \prod_{\substack{\mu=1\\ \mu\neq i}}^{d} r_\mu^T.$$

The first step is to combine the Tucker core with the Tucker leg vertex that has the maximal Tucker rank (see Figure 2.11 and the resulting Figure 2.12), so for all fixed $(j_1,\dots,j_{i-1},j_{i+1},\dots,j_d) \in \bigtimes_{\substack{\mu=1\\ \mu\neq i}}^{d} \{1,\dots,r_\mu^T\}$ we define

$$f_i : \bigtimes_{\substack{\mu=1\\ \mu\neq i}}^{d} \{1,\dots,r_\mu^T\} \to \{1,\dots,r^{CP}\}$$

to be an arbitrary bijective mapping and then set

$$v_i^{CP}(f_i(j_1,\dots,j_{i-1},j_{i+1},\dots,j_d)) := \sum_{j_i=1}^{r_i^T} w^T(j_1,\dots,j_d) v_i^T(j_i).$$

Finally, we interpret all remaining $d-1$ summations as one large summation, that is

$$v_\mu^{CP}(f_i(j_1,\dots,j_{i-1},j_{i+1},\dots,j_d)) := v_\mu^T(j_\mu)\ \forall \mu \in \{1,\dots,d\}\backslash\{i\}.$$

So we obtain

$$\sum_{j_1,\dots,j_d=1}^{r_1^T,\dots,r_d^T} w^T(j_1,\dots,j_d) \bigotimes_{\mu=1}^{d} v_\mu(j_\mu) = \sum_{j_1,\dots,j_{i-1},j_{i+1},\dots,j_d=1}^{r_1^T,\dots,r_{i-1}^T,r_{i+1}^T,\dots,r_d^T} \bigotimes_{\mu=1}^{i-1} v_\mu(j_\mu) \otimes \left(\sum_{j_i=1}^{r_i^T} w^T(j_1,\dots,j_d) v_i(j_i) \right) \otimes \bigotimes_{\mu=i+1}^{d} v_\mu(j_\mu)$$

$$= \sum_{j_1,\dots,j_{i-1},j_{i+1},\dots,j_d=1}^{r_1^T,\dots,r_{i-1}^T,r_{i+1}^T,\dots,r_d^T} \bigotimes_{\mu=1}^{d} v_\mu^{CP}(f_i(j_1,\dots,j_{i-1},j_{i+1},\dots,j_d))$$

which is an r-term tensor if we interpret the summation indices as one multi-index

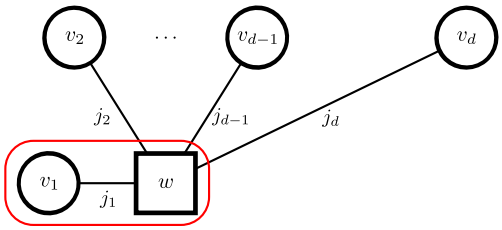

Figure 2.11: Merging one spatial vertex and the Tucker core

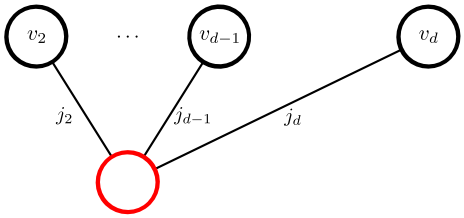

Figure 2.12: New structure after vertex merge (not yet an r-term tensor)

2.2.5 Tucker to hierarchical (balanced)

In this case, we consider the *Tucker core* w^T as a full tensor of order d where w.l.o.g. $d = 2^k$ for some $k \in \mathbb{N}$. The dimension of the Tucker core vector spaces is equal to the corresponding Tucker rank. We can convert this full tensor into a (balanced) hierarchical tensor with representation rank $(r_1^H, \dots, r_{2d-2}^H) \in \mathbb{N}^{2d-2}$ (see [5, Chapter 11 and especially Section 11.3] for a detailed overview) such that we obtain for each scalar entry of w^T

through

$$w^T(j_1,\dots,j_d) = \sum_{i_1,\dots,i_{2d-2}=1}^{r_1^H,\dots,r_{2d-2}^H} w_1^H(i_1,i_2) \prod_{\nu=2}^{k} \left(\prod_{\mu=2^{\nu-1}}^{2^\nu-1} w_\mu^H(i_{\mu-1},i_{2\mu-1},i_{2\mu}) \right) \prod_{\mu=1}^{d} \tilde{v}_\mu^H(i_{d-2+\mu})_{j_\mu}.$$

Now we define for all $\mu \in \{1,\dots,d\}, i_{d-2+\mu} \in \{1,\dots,r_{d-2+\mu}^H\}$

$$v_\mu^H(i_{d-2+\mu}) := \sum_{j_\mu=1}^{r_\mu^T} \tilde{v}_\mu^H(i_{d-2+\mu})_{j_\mu} v_\mu^T(i).$$

and obtain a balanced hierarchical tensor representation.

2.2.6 Tucker to TT

In the first step, we compute (i.e. no approximation) the TT representation with representation rank $(r_1^{TT},\dots,r_{d-1}^{TT}) \in \mathbb{N}^{d-1}$ of the Tucker core w^T, such that

$$w^T(j_1,\dots,j_d) = \sum_{i_1,\dots,i_{d-1}=1}^{r_1^{TT},\dots,r_{d-1}^{TT}} c_1(i_1)_{j_1} \left(\prod_{\mu=2}^{d-1} c_\mu(i_{\mu-1},i_\mu)_{j_\mu} \right) c_d(i_{d-1})_{j_d}$$

with

$$\begin{aligned} c_1 &: \mathbb{N} \to \mathcal{V}_1, \\ c_\mu &: \mathbb{N}^2 \to \mathcal{V}_\mu \text{ for } \mu = 2,\dots,d-1 \end{aligned}$$

and

$$c_d : \mathbb{N} \to \mathcal{V}_d.$$

Then we define for all fixed $(i_1,\dots,i_{d-1}) \in \bigtimes_{\mu=1}^{d-1} \{1,\dots,r_\mu^{TT}\}$

$$v_1^{TT}(i_1) := \sum_{j_1=1}^{r_1^T} c_1(i_1)_{j_1} v_1^T(j_1),$$

$$v_\mu^{TT}(i_{\mu-1},i_\mu) := \sum_{j_\mu=1}^{r_\mu^T} c_\mu(i_{\mu-1},i_\mu)_{j_\mu} v_\mu^T(j_\mu) \ \forall \mu \in \{2,\dots,d-1\}$$

and

$$v_d^{TT}(i_{d-1}) := \sum_{j_d=1}^{r_d^T} c_d(i_{d-1})_{j_d} v_d^T(j_d).$$

2.2.7 Hierarchical (balanced) to r-term

Similar to Subsection 2.2.4, we first define

$$f : \bigtimes_{\mu=2}^{d} \{1, \dots, r^H_{d-2+\mu}\} \to \{1, \dots, r^{CP}\}$$

to be an arbitrary bijective mapping,

$$r^{CP} := \prod_{\mu=2}^{d} r^H_{d-2+\mu}$$

and for all fixed $(j_d, \dots, j_{2d-2}) \in \bigtimes_{\mu=2}^{d} \{1, \dots, r^H_{d-2+\mu}\}$

$$v_1^{CP}(f(j_d, \dots, j_{2d-2})) := \sum_{j_1, \dots, j_{d-1}=1}^{r_1^H, \dots, r_{d-1}^H} w_1^H(j_1, j_2) \prod_{\nu=2}^{k} \left(\prod_{\mu=2^\nu}^{2^\nu - 1} w_\mu^H(j_{\mu-1}, j_{2\mu-1}, j_{2\mu}) \right) \cdot v_1^H(j_{d-1})$$

and

$$v_\mu^{CP}(f(j_d, \dots, j_{2d-2})) := v_\mu^H(j_{d+\mu-2}) \ \forall \mu \in \{2, \dots, d\}.$$

is again an arbitrary bijective mapping. So we first merge the scalar vertices with a spatial vertex (as visualized in Figure 2.13) such that we end up with a representation shown in Figure 2.14. Instead of combining the scalar vertices with the first spatial vertex, one could also combine the scalar vertices with any other spatial vertex. The choice of the spatial vertex has an influence on r^{CP} (see Subsection 2.2.4). Finally, we interpret the $d-1$ summations over $j_d, \dots, j_{2d-2}$ as one summation over a multi-index.

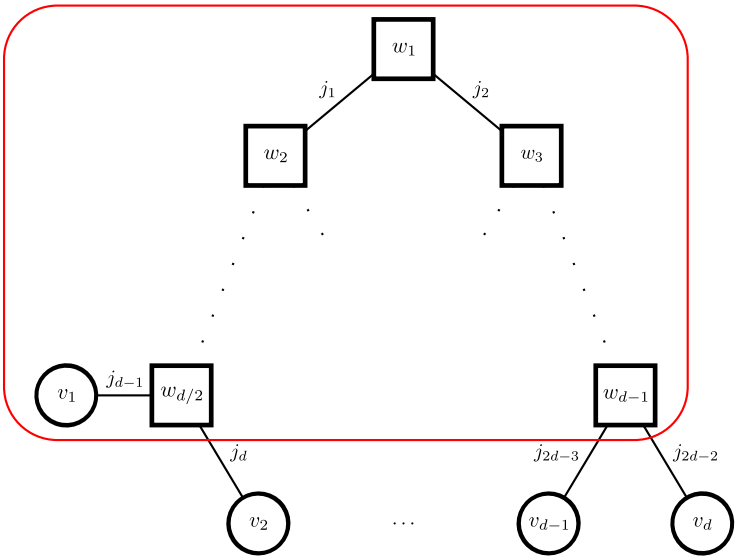

Figure 2.13: Merging the non-leaf vertices and one spatial vertex

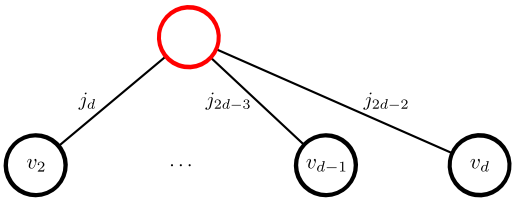

Figure 2.14: Newly summarized spatial vertex v_1 (not yet an r-term tensor)

2.2.8 Hierarchical (balanced) to Tucker

In the case of the reformulation of a hierarchically represented tensor to a Tucker represented one, we just have to define a new Tucker core. That is, we have $d = 2^k$ for some $k \in \mathbb{N}$ and

$$r_\mu^T := r_{d-2+\mu}^H \ \forall \mu \in \{1, \dots, d\}$$

and for all fixed $(j_1, \dots, j_d) \in \bigtimes_{\mu=1}^{d} \{1, \dots, r_\mu^T\}$

$$v_\mu^T(j_\mu) := v_\mu^H(j_\mu)$$

and define the new Tucker core as of Figure 2.15, i.e.

$$w^T(j_{d-1}, \dots, j_{2d-2}) := \sum_{j_1,\dots,j_{d-2}=1}^{r_1^H,\dots,r_{d-2}^H} w_1^H(i_1, i_2) \prod_{\nu=2}^{k} \left(\prod_{\mu=2^{\nu-1}}^{2^\nu-1} w_\mu^H(j_{\mu-1}, j_{2\mu-1}, j_{2\mu} \right)$$

for all $(j_{d-1}, \dots, j_{2d-2}) \in \bigtimes_{\mu=d-1}^{2d-2} \{1, \dots, r_\mu^H\}$.

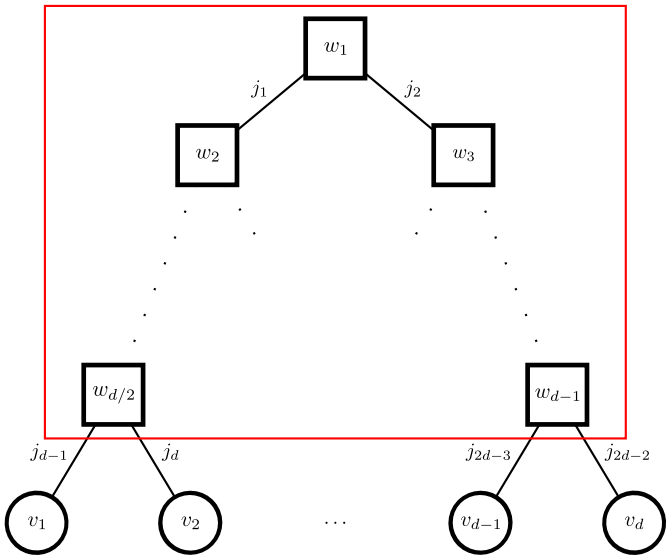

Figure 2.15: Summarizing the non-leaf vertices as the new Tucker core

2.2.9 Hierarchical (balanced) to TT

For this conversion of the tensor representation, we refer to [5, Section 12.3.3]. It involves the underlying topological structure of the hierarchical format, which we want to neglect in this work completely.

2.2.10 TT to r-term

We define

$$f : \bigtimes_{\mu=1}^{d-1} \{1, \dots, r_\mu^{TT}\} \to \{1, \dots, r^{CP}\}$$

to be an arbitrary bijective mapping, then set

$$r^{CP} := \prod_{\mu=1}^{d-1} r_\mu^{TT}$$

and for all fixed $(j_1, \dots, j_{d-1}) \in \bigtimes_{\mu=1}^{d-1} \{1, \dots, r_\mu^{TT}\}$

$$\begin{aligned} v_1^{CP}(f(j_1, \dots, j_{d-1})) &:= v_1^{TT}(j_1), \\ v_\mu^{CP}(f(j_1, \dots, j_{d-1})) &:= v_\mu^{TT}(j_{\mu-1}, j_\mu) \ \forall \mu \in \{2, \dots, d-1\} \end{aligned}$$

and

$$v_d^{CP}(f(j_1,\dots,j_{d-1})) := v_d^{TT}(j_{d-1}).$$

2.2.11 TT to Tucker

We set

$$\begin{aligned} r_1^T &:= r_1^{TT}, \\ r_\mu^T &:= r_{\mu-1}^{TT} r_\mu^{TT} \ \forall \mu \in \{2,\dots,d-1\}, \\ r_d^T &:= r_{d-1}^{TT} \end{aligned}$$

and define for $\mu \in \{1,\dots,d-2\}$

$$f_\mu : \{1,\dots,r_\mu^{TT}\} \times \{1,\dots,r_{\mu+1}^{TT}\} \to \{1,\dots,r_\mu^{TT} r_{\mu+1}^{TT}\}$$

as an arbitrary bijective mapping, as well as

$$\begin{aligned} c : \mathbb{N} \times \mathbb{N} &\to \{0,1\}, \\ (a,b) &\mapsto \begin{cases} 1 & \text{if } a = b \\ 0 & \text{else.} \end{cases} \end{aligned}$$

Then we have for all fixed $j_\mu, j'_\mu \in \{1,\dots,r_\mu^{TT}\}$ for $\mu = 1,\dots,d-1$

$$\begin{aligned} w^T(j_1, f_1(j'_1,j_2), f_2(j'_2,j_3),\dots,f_{d-2}(j'_{d-2},j_{d-1}),j'_{d-1}) &:= \prod_{\mu=1}^{d-1} c(j_\mu, j'_\mu), \\ v_1^T(j_1) &:= v_1^{TT}(j_1), \\ v_\mu^T(f_{\mu-1}(j_{\mu-1},j_\mu)) &:= v_\mu^{TT}(j_{\mu-1},j_\mu) \ \forall \mu \in \{2,\dots,d-1\} \end{aligned}$$

and

$$v_d^T(j_{d-1}) := v_d^{TT}(j_{d-1}),$$

such that

$$\begin{aligned} &\sum_{j_1,\dots,j_{d-1}=1}^{r_1^{TT},\dots,r_{d-1}^{TT}} v_1^{TT}(j_1) \otimes \left(\bigotimes_{\mu=2}^{d-1} v_\mu^{TT}(j_{\mu-1},j_\mu) \right) \otimes v_d^{TT}(j_{d-1}) = \\ &\sum_{j_1,j'_1,\dots,j_{d-1},j'_{d-1}=1}^{r_1^{TT},r_1^{TT},\dots,r_{d-1}^{TT},r_{d-1}^{TT}} \prod_{\mu=1}^{d-1} c(j_\mu,j'_\mu) v_1^{TT}(j_1) \otimes \left(\bigotimes_{\mu=2}^{d-1} v_\mu^{TT}(j'_{\mu-1},j_\mu) \right) \otimes v_d^{TT}(j'_{d-1}) = \\ &\sum_{j_1,j'_1,\dots,j_{d-1},j'_{d-1}=1}^{r_1^{TT},r_1^{TT},\dots,r_{d-1}^{TT},r_{d-1}^{TT}} w^T(j_1,f_1(j'_1,j_2),f_2(j'_2,j_3),\dots,f_{d-2}(j'_{d-2},j_{d-1}),j'_{d-1}) \\ &\qquad \cdot v_1^T(j_1) \otimes \left(\bigotimes_{\mu=2}^{d-1} v_\mu^T(f_{\mu-1}(j'_{\mu-1},j_\mu)) \right) \otimes v_d^T(j'_{d-1}). \end{aligned}$$

This procedure has two steps. In the first one, we introduce the artificial factors c such that we end up with a representation that can be visualized as of Figure 2.16. Subsequently, we merge the c-factors into the new Tucker core (see Figure 2.17). These two steps are needed since we have to obtain disjoint summation indices for the Tucker format.

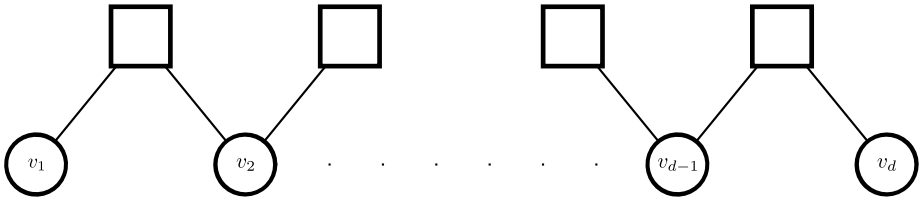

Figure 2.16: Introduction of artificial vertices for TT

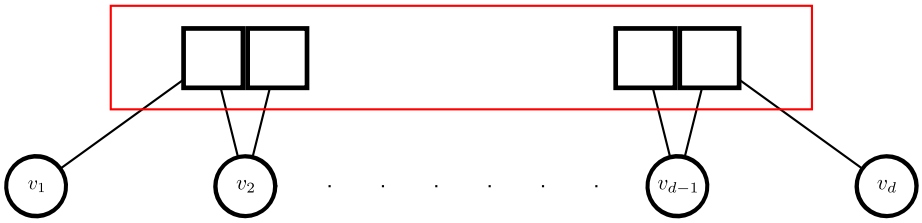

Figure 2.17: Summarizing artificial vertices to new Tucker core

2.2.12 TT to hierarchical (balanced)

Analogously to Subsection 2.2.9, we refer to [5, Section 12.3.2].

2.2.13 TC to TT

At first, we define for each $\mu \in \{1, \dots, d-1\}$

$$f_\mu : \{1, \dots, r_\mu^{TC}\} \times \{1, \dots, r_d^{TC}\} \to \{1, \dots, r_\mu^{TC} r_d^{TC}\}$$

as an arbitrary bijective mapping. We set, by using the mapping c from Subsection 2.2.11,

$$r_\mu^{TT} := r_\mu^{TC} \cdot r_d^{TC}, \mu \in \{1, \dots, d-1\}$$

and for all $(j_1, \dots, j_d, j'_d) \in \bigtimes_{\mu=1}^{d} \{1, \dots, r_\mu^{TC}\} \times \{1, \dots, r_d^{TC}\}$ we define

$$\begin{aligned} v_1^{TT}(f_1(j_1, j_d)) &:= v_1^{TC}(j_1, j_d), \\ v_\mu^{TT}(f_{\mu-1}(j_{\mu-1}, j_d), f_\mu(j_\mu, j'_d)) &:= \begin{cases} v_\mu^{TC}(j_{\mu-1}, j_\mu) & \text{if } j_d = j'_d \\ 0 & \text{else} \end{cases}, \mu \in \{2, \dots, d-1\} \\ &= c(j_d, j'_d) v_\mu^{TC}(j_{\mu-1}, j_\mu) \end{aligned}$$

and

$$v_d^{TT}(f_{d-1}(j_{d-1}, j_d)) := v_d^{TC}(j_{d-1}, j_d),$$

such that we get

$$\begin{aligned}
&\sum_{j_1,\dots,j_d=1}^{r_1^{TC},\dots,r_d^{TC}} v_1^{TC}(j_1, j_d) \otimes \bigotimes_{\mu=2}^{d} v_\mu^{TC}(j_{\mu-1}, j_\mu) = \\
&\sum_{j_1, j_d^{(1)}, \dots, j_{d-1}, j_d^{(d-1)}=1}^{r_1^{TC}, r_d^{TC}, \dots, r_{d-1}^{TC}, r_d^{TC}} v_1^{TC}(j_1, j_d^{(1)}) \otimes \left(\bigotimes_{\mu=2}^{d-1} \left(c(j_d^{(\mu-1)}, j_d^{(\mu)}) v_\mu^{TC}(j_{\mu-1}, j_\mu) \right) \right) \\
&\qquad \otimes v_d^{TC}(j_{d-1}, j_d^{(d-1)}) = \\
&\sum_{j_1, j_d^{(1)}, \dots, j_{d-1}, j_d^{(d-1)}=1}^{r_1^{TC}, r_d^{TC}, \dots, r_{d-1}^{TC}, r_d^{TC}} v_1^{TT}(f_1(j_1, j_d^{(1)})) \otimes \left(\bigotimes_{\mu=2}^{d-1} v_\mu^{TT}(f_{\mu-1}(j_{\mu-1}, j_d^{(\mu-1)}), f_\mu(j_\mu, j_d^{(\mu)})) \right) \\
&\qquad \otimes v_d^{TT}(f_{d-1}(j_{d-1}, j_d^{(d-1)})).
\end{aligned}$$

Remark 2.2.1. *The choice of the d-th summation as an indicator is artificial. Instead, one could also chose any other summation. Compare Subsections 2.2.4 and 2.2.7.*

2.2.14 Summary of conversions

We described *some* conversions of tensor representations that do not change the tensor itself, but its representation. If only an approximated conversion is needed, other algorithms should be used. Figure 2.18 shows the algorithms that we described for converting representations.

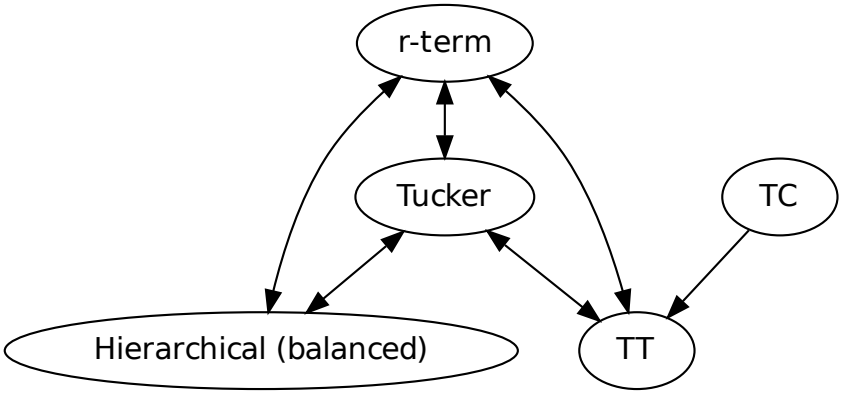

Figure 2.18: Described conversions of tensor representations

It is important to understand that performing conversions has a certain *price*. That is, a conversion from a tensor representation in format A into a tensor representation in format B and then converting back into a representation in format A is in general **not** the identity. In theses cases, the representation rank increases. If we know the origin of a converted tensor representation however, we might be able to use that knowledge to exploit the structure and perform a perfect (i.e. $\mathrm{A} \to \mathrm{B} \to \mathrm{A} \equiv Id$) reconversion (see [30, Special example]).

Chapter 3

Tensor network approach

In this chapter, we will define what we understand of tensor networks and give prominent examples of formats that fit into the same framework. We are going to explain an important connection to the r-term format which gives some benefits.
The content is heavily based on the joint paper [1] with Mike Espig, Wolfgang Hackbusch and Reinhold Schneider.

We start with a short Definition.

Definition. *Let A and B be sets. Then we define*

$$A \biguplus B := \{\{a, b\} : a \in A, b \in B, a \neq b\}$$

which is basically $A \times B$ with a neglected order. Analogously the n-fold $\biguplus$ is defined.

3.1 Graph related definitions

As we will utilize graphs to define tensor network structures, we start with general graph related definitions.

We modify and combine definitions from [31, p. 2] and [31, p. 32] and obtain

Definition 3.1.1 (Undirected graph, connected graph)**.** *Let V be the set of vertices and $E \subset V \biguplus V$ be the set of edges, which we define to be totally ordered sets with the relation $<$, such that*

$$\forall e \in E : \#e = 2$$

then $G = (V, E)$ is defined as an undirected graph. *For simplicity, we assume that $\forall \vartheta_1, \vartheta_2 \in V$ with $\vartheta_1 \neq \vartheta_2$ there exists a sequence $(e_1, \dots, e_k) \in E^k$ for some $k \in \mathbb{N}$ with $\vartheta_1 \in e_1, \vartheta_2 \in e_k$ and $e_i \cup e_{i+1} \neq \varnothing$ $\forall i \in \{1, \dots, k-1\}$. That defines (V, E) as a* connected graph.

Remark 3.1.2. *Without loss of generality, we can assume in our context that all graphs are connected. Therefore, even if we are only using the word graph, we actually mean connected and undirected graph. The set of edges will always be a totally ordered set with the relation symbol $<$.*

Generalizing the graph definition slightly, leaves us with

Definition 3.1.3 (Multi-graph)**.** *Let V be the set of vertices and $E \subset \mathcal{P}(V)$ be the set of edges with*

$$\forall e \in E : \#e \geqslant 2$$

then $G = (V, E)$ is defined as an (undirected) multi-graph*. Here we also assume for simplicity that $\forall \vartheta \in V \ \exists e \in E : \vartheta \in e$. Any graph as of Definition 3.1.1 is also a multi-graph. So a multi-graph may have edges that connect more that two vertices. $\mathcal{P}(V)$ denotes the power set of V.*

Note that this definition of a multi-graph differs from the multi-graph definition in [31, p. 3]. In [31] a multi-graph is defined to have different edges that connect the same two vertices.

Remark 3.1.4. *Since all $e \in E$ are sets that have a cardinality of at least two, we do not allow self loops. Therefore, we are dealing with simple undirected connected multi-graphs.*

By [1, Definition 2.4 (degree map)], we get

Definition 3.1.5 (Incidence map, degree map)**.** *Let $G = (V, E)$ be an undirected multi-graph as of Definition 3.1.3. Then we define*

$$\begin{aligned} \mathbb{I} : V &\to \mathcal{P}(E) \\ \vartheta &\mapsto \{e \in E : \vartheta \in e\} \end{aligned}$$

as the incidence map *of G. We further define*

$$\begin{aligned} g : V &\to \mathbb{N} \\ \vartheta &\mapsto \#\mathbb{I}(\vartheta) \end{aligned}$$

as the degree map *of G.*

Just like [1, directly after Definition 2.5], we simplify the notation with the help of the following Remark.

Remark 3.1.6. *We identify each element of V by a natural number such that we can identify each edge uniquely by a natural number from 1 to $\#V$. Therefore in the following, we do not distinguish between the vertex and its number. To be more precise,*

$$V \cong \{1, \ldots, \#V\}.$$

The same will be applied to E, i.e.

$$E \cong \{1, \ldots, \#E\}.$$

This remark has also an influence on both the definition of the incidence map and the degree map as well as on the upcoming definitions. Using the identification, we can formulate expressions like

$$e_i < e_j \text{ for some } e_i, e_j \in E$$

and

$$\vartheta_i < \vartheta_j \text{ for some } \vartheta_i, \vartheta_j \in V$$

in the sense of non-negative integers, which will correspond with the $<$ relation of the totally ordered set E and alter the definition of the degree map to be

$$g : \{1, \dots, \#V\} \to \mathbb{N}.$$

However, $e_i < e_j$ does not automatically imply $i < j$. For instance we can set

$$E := \{e_1, e_2, e_3, e_4\}$$

with $e_2 < e_3 < e_1 < e_4$ such that in the sense of the identification of E with $\{1, 2, 3, 4\}$, e_1 represents the 3rd edge of E. Consequently, E and V are treated as ordered sets in our context. We are going to use this identification implicitly through the whole work.

In [1, Definition 2.5], the following definition has been used to define the term *incidence map*. Since in this work, we use the standard definition of this term in Definition 3.1.5, we give

Definition 3.1.7 (Incidence* map)**.** *Let $G = (V, E)$ be an undirected connected multi-graph where $\mathbb{N} \ni \ell = \#E$ is the number of edges of G such that $E = \{e_1, \dots, e_\ell\}$. Then*

$$\begin{aligned} \mathcal{I} : V \times \mathbb{N}^\ell &\to \bigcup_{i=1}^{\ell} \mathbb{N}^i \\ (\vartheta, (j_1, \dots, j_\ell)) &\mapsto (j_{e_{h_1}}, \dots, j_{e_{h_{g(\vartheta)}}}) \end{aligned}$$

where $\{e_{h_1}, \dots, e_{h_{g(\vartheta)}}\} = \mathbb{I}(\vartheta)$ and $e_{h_f} < e_{h_{f+1}}$ for $f = 1, \dots, g(\vartheta) - 1$ is defined as the incidence* map *of G (read* incidence-star map*).*

3.2 Tensor networks

Our approach for tensor networks is based on multi-graphs and with the definitions of the previous section, we can construct a framework to treat structured sums of tensors to represent multi-dimensional data. When we neglect the word *network* from the terms *tensor format* and *tensor representation* as it is clear, that we are referring to tensor networks.

Remark 3.2.1. *In the following, we will define mappings by using graphs. In this context, we are going to interpret vertices of graphs as mappings. We will use the simple notation that vertex ϑ_i represents the mapping v_i if the vertex is of vector space meaning and w_{i-d} if the vertex is of scalar meaning. See also Remark 3.2.4.*

Generalizing [1, Definition 2.4 (Tensor network graph)], we get

Definition 3.2.2 (Tensor format graph set)**.** *Let $k \in \mathbb{N}, V$ be the set of vertices and $(V, E_1), \dots, (V, E_k)$ be multi-graphs as of Definition 3.1.3. Then $(V, \mathcal{E})$ with $\mathcal{E} := \{E_1, \dots, E_k\}$ is defined as a* tensor format graph set *of cardinality k.*

We now want to define another central object of this work, which is the *tensor format*. We will use the definition for the tensor format of [1, Definition 2.6] with the difference that we do not define and utilize a *parameter space* and with the extension to use multi-graphs:

Definition 3.2.3 (Tensor format)**.** *Let $(V, \mathcal{E})$ be a tensor format graph set of cardinality k with $\mathcal{E} = \{E_1, \dots, E_k\}$, $\ell_i := \#E_i$ for $i = 1, \dots, k$ and $d \in \mathbb{N}, L \in \mathbb{N}_0$ with $\#V = d + L$. Let further g_i be the degree map of (V, E_i) and $\mathcal{I}_i$ the incidence* map of (V, E_i) for $i = 1, \dots, k$. Then we define*

$$T_i : \bigtimes_{\mu=1}^{d} \left\{ f : \mathbb{N}^{g_i(\mu)} \to \mathcal{V}_\mu \right\} \times \bigtimes_{\mu=1}^{L} \left\{ f : \mathbb{N}^{g_i(d+\mu)} \to \mathbb{K} \right\} \to \mathcal{V}$$

$$(v_1, \dots, v_d, w_1, \dots, w_L) \mapsto \sum_{j \in \mathbb{N}^{\ell_i}} \prod_{\mu=1}^{L} w_\mu(\mathcal{I}_i(d+\mu, j)) \bigotimes_{\mu=1}^{d} v_\mu(\mathcal{I}_i(\mu, j)))$$

for all $i = 1, \dots, k$. Then

$$\{T_1, \dots, T_k\}$$

defines a tensor format of order (d, L)*. If $L = 0$ we simply say* tensor format of order d*. For a proper definition, we also require all mappings $v_1, \dots, v_d, w_1, \dots, w_L$ to have finite support (compare Section 2.0).*

Remark 3.2.4. *A tensor format is uniquely defined by its tensor format graph set. We need a set of graphs to define a format since one format can be based on multiple graphs (see Definition 2.1.4 and Definition 2.1.6 which both define a hierarchical format). A tensor format is a mapping which is defined by its underlying graph. Note that the graph vertices ϑ_i are not equal to the mappings v_i. However, in the graphical notation, we identify them with each other for the sake of the readability of the formulas. The vertices are needed to define the mappings. By using the underlying multi-graph, we can define the signatures of the mappings v_i and w_i. See also Section 2.0.*

A *tensor representation* as of [1, Definition 2.6] (while neglecting again a *parameter space*) is defined as follows

Definition 3.2.5 (Tensor representation). *Let (V, E) be a tensor format graph with $d \in \mathbb{N}, L \in \mathbb{N}_0$ and $d + L = \#V$ with corresponding degree map g, incidence map $\mathbb{I}$ and incidence* map $\mathcal{I}$. Let further $\ell := \#E, (r_1, \dots, r_\ell) \in \mathbb{N}^\ell$ and $\{e_{\mu,1}, \dots, e_{\mu,g(\mu)}\} = \mathbb{I}(\mu)$ with $e_{\mu,s} < e_{\mu,s+1}$ for $s = 1, \dots, g(\mu) - 1$ and $\mu = 1, \dots, d + L$. Then we define*

$$\tilde{T} : \bigtimes_{\mu=1}^{d} \left\{ f : \bigtimes_{j=1}^{g(\mu)} \{1, \dots, r_{e_{\mu,j}}\} \to \mathcal{V}_\mu \right\} \times \bigtimes_{\mu=1}^{L} \left\{ f : \bigtimes_{j=1}^{g(d+\mu)} \{1, \dots, r_{e_{d+\mu,j}}\} \to \mathbb{K} \right\} \to \mathcal{V}$$

$$(v_1, \dots, v_d, w_1, \dots, w_L) \mapsto \sum_{j_1,\dots,j_\ell=1}^{r_1,\dots,r_\ell} \prod_{\mu=1}^{L} w_\mu(\mathcal{I}(d+\mu, (j_1, \dots, j_\ell))) \bigotimes_{\mu=1}^{d} v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell)))$$

as tensor representation *of order (d, L) for tensor format graph (V, E) with representation rank $(r_1, \dots, r_\ell)$. Analogously to Definition 3.2.3, we neglect L if it is zero.*

We will also use XYZ tensor *as a synonym for* tensor in XYZ format *or* tensor in XYZ representation, *where* XZY *is a tensor format, compare the format definitions of the previous section.*

Frankly speaking, a tensor representation is an element of a tensor format where we limit the number of terms (note that the support of v_i and w_j is finite by definition). Also, the structure of the underlying tree is fixed for one specific tensor representation. This means that there is exactly one graph per tensor representation. Even if only the graph is changed without leaving a certain tensor format, it will lead to a different representation.

Remark 3.2.6. *As the representation rank is a tuple, one cannot without loss of generality say what rank the best representation rank in a certain format is (unless it is a 1-tuple, like in the r-term format). In practice, one measures the rank based on the application and makes comparisons based on certain operations (e.g. the inner product). See [5, Chapter 13] for a comparison.*

The L scalar vertices can and should be treated separately. This is emphasized in the following

Lemma 3.2.7 (Adding of scalar vertices). *Let (V, E) be a tensor format graph of order $(d, L) \in \mathbb{N} \times \mathbb{N}_0$ which defines a mapping v. Then for each $\{\vartheta_1, \vartheta_2\} \in E$, we can define a mapping v' that is equal to mapping v and defined by the tensor format graph (V', E') of order $(d, L + 1)$ with*

$$\begin{aligned} &\vartheta_{d+L+1} \notin V, \\ &\forall \vartheta \in V : \vartheta < \vartheta_{d+L+1}, \\ V' &:= V \cup \{\vartheta_{d+L+1}\} \end{aligned}$$

and

$$E' := (E \backslash \{\{\vartheta_1, \vartheta_2\}\}) \cup \{\{\vartheta_1, \vartheta_{d+L+1}\}, \{\vartheta_{d+L+1}, \vartheta_2\}\}.$$

Without loss of generality, $e < \{\vartheta_1, \vartheta_{d+L+1}\} < \{\vartheta_{d+L+1}, \vartheta_2\}$ for all $e \in E$ (remember that the edge set is an ordered set per definition).

Proof. We define

$$w_{L+1} : \mathbb{N} \times \mathbb{N} \to \mathbb{K}$$
$$(i,j) \mapsto \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else.} \end{cases}$$

and set $\ell := \#E$. W.l.o.g., we can assume $e < \{\vartheta_1, \vartheta_2\}$ for all $e \in E \backslash \{\{\vartheta_1, \vartheta_2\}\}$. Due to our assumption on the ordering of E, we have

$$\{\vartheta_1, \vartheta_2\} < \{\vartheta_1, \vartheta_{d+L+1}\} < \{\vartheta_{d+L+1}, \vartheta_2\}.$$

Additionally, we can assume $\vartheta_1 < \vartheta_2 < \vartheta_i \ \forall i \in \{3, \dots, d+L\}$. By adding an additional edge to the graph, the incidence map also changes. We define $\mathcal{I}$ to be the incidence* map with respect to graph (V, E) and $\mathcal{I}'$ to be the incidence* map w.r.t. (V', E'). Then we have

$$\sum_{j_1,\dots,j_\ell} \prod_{\mu=1}^{L} w_\mu(\mathcal{I}(\mu + d, (j_1, \dots, j_\ell))) \bigotimes_{\mu=1}^{d} v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell))) =$$
$$\sum_{j_1,\dots,j_{\ell+1}} \prod_{\mu=1}^{L} w_\mu(\mathcal{I}'(\mu + d, (j_1, \dots, j_{\ell+1}))) \cdot w_{L+1}(j_\ell, j_{\ell+1}) \bigotimes v_\mu(\mathcal{I}'(\mu, (j_1, \dots, j_{\ell+1})))$$

per definition of w_{L+1}. The following properties therefore hold

$$\begin{aligned} \mathcal{I}(\mu, (j_1, \dots, j_\ell)) &= \mathcal{I}'(\mu, (j_1, \dots, j_{\ell+1})) \ \forall \mu \in \{3, \dots, d+L\} \\ \mathcal{I}'(1, (j_1, \dots, j_{\ell+1})) &= \mathcal{I}'(1, (j_1, \dots, j_\ell, \cdot)) \\ \mathcal{I}'(2, (j_1, \dots, j_{\ell+1})) &= \mathcal{I}'(2, (j_1, \dots, j_{\ell-1}, \cdot, j_{\ell+1})) \end{aligned}$$

for all $(j_1, \dots, j_{\ell+1}) \in \mathbb{N}^{\ell+1}$ where $\cdot$ means that we can put in any value $x \in \mathbb{N}$. Consequently, if v is the mapping that is defined by (V, E) and v' is the mapping that is defined by (V', E') as stated above, we have

$$v(v_1, \dots, v_d, w_1, \dots, w_L) = v'(v_1, \dots, v_d, w_1, \dots, w_L, w_{L+1})$$

for all $v_1, \dots, v_d, w_1, \dots, w_L$ defined as before with w_{L+1} as defined above. □

Lemma 3.2.7 means in terms of our graph based notation that we can add arbitrary many scalar vertices between two vertices of our graph. Note that the type of the two vertices is not of importance. That is, we can add a scalar vertex between other scalar vertices as well as between vector space vertices and between a vector space vertex and a scalar vertex.

We used this property implicitly in Subsection 2.2.11 to convert the representation of a tensor from the TT format to the Tucker format. In the same way, we can use

Lemma 3.2.7 to convert the linear hierarchical representation to a TT representation. In [5, Section 12.2], the TT representation is interpreted as a hierarchical representation, which we also can formulate if we consider the representation of a tensor only up to the existence of scalar vertices. If the more general hierarchical format is used, there are some differences in terms of representation rank and the computational complexity as shown in [32]. We can always combine a scalar vertex ϑ_2 with a spatial vertex ϑ_1 if the underlying graph (V, E) has an edge $\{\vartheta_1, \vartheta_2\}$. Having this in mind, we could also interpret a Tucker representation as a hierarchical representation such that the Tucker format would become a special case of the hierarchical format.

For some theoretical aspects, it is not important to treat the scalar vertices in a special manner. We can treat them as spatial vertices with vector space dimension 1. This leads to

Definition 3.2.8 (Simplified notation)**.** *Let $(V, E) = (\{\vartheta_1, \dots, \vartheta_{d+L}\}, E)$ be a tensor format graph of order $(d, L) \in \mathbb{N} \times \mathbb{N}_0$ with corresponding incidence* map $\mathcal{I}$ with $\ell := \#E$. Let further*

$$\sum_{j_1,\dots,j_\ell=1}^{r_1,\dots,r_\ell} \prod_{\mu=1}^{L} w_\mu(\mathcal{I}(\mu + d, (j_1, \dots, j_\ell))) \bigotimes_{\mu=1}^{d} v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell))) \tag{3.1}$$

be a tensor representation of order (d, L) with representation rank $(r_1, \dots, r_\ell) \in \mathbb{N}^\ell$ for tensor format graph (V, E) as of Definition 3.2.5. Then we define

$$\sum_{j_1,\dots,j_\ell=1}^{r_1,\dots,r_\ell} \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell)))$$

to be the simplified notation *of the tensor representation* (3.1) *with*

$$v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell))) := w_{\mu-d}(\mathcal{I}(\mu, (j_1, \dots, j_\ell)))$$

for all $\mu \in \{d+1, \dots, d+L\}$. This representation is in a tensor network format of order $(d + L, 0)$ where the vector space dimension of $\mathcal{V}_{d+1}, \dots, \mathcal{V}_{d+L}$ are equal to 1.

We can - in some sense - formulate the reversal of Lemma 3.2.7:

Lemma 3.2.9 (Negligibility of scalar vertices)**.** *Let $G = (V, E) = (\{\vartheta_1, \dots, \vartheta_{d+L}\}, E)$ with $\vartheta_i < \vartheta_{i+1}$ for all $i \in \{1, \dots, d+L-1\}$ be a tensor network graph of order $(d, L) \in \mathbb{N}^2$ with degree map g, which defines a mapping v. Then we can find for all for all $e = \{\tilde{\vartheta}_1, \tilde{\vartheta}_2\} \in E$ where w.l.o.g*

$$\tilde{\vartheta}_2 > \vartheta_d$$

and

$$\tilde{\vartheta}_1 \leqslant \vartheta_d$$

with

$$\forall e' \in E \backslash \{e\} : e' \cap e = \varnothing,$$

a mapping v' that is equal to v and induced by the tensor format graph (V', E') of order $(d, L-1)$ with

$$V' := V \backslash \{\tilde{\vartheta}_2\}$$

and

$$E' := (E \backslash \{e' \in E \mid \tilde{\vartheta}_2 \in e'\}) \cup \{(e' \backslash \{\tilde{\vartheta}_2\}) \cup \{\tilde{v}_1\} \mid e' \in E \backslash \{e\}, \tilde{\vartheta}_2 \in e'\}.$$

Proof. Without loss of generality, we assume that $\tilde{\vartheta}_1 = \vartheta_d, \tilde{\vartheta}_2 = \vartheta_{d+1}$,

$$e < \{\tilde{\vartheta}_1, \tilde{\vartheta}_2\} \ \forall e \in E \backslash \{\{\tilde{\vartheta}_1, \tilde{\vartheta}_2\}\}$$

and

$$\forall e_1 \in \mathbb{I}(\tilde{\vartheta}_2) \backslash \{\{\tilde{\vartheta}_1, \tilde{\vartheta}_2\}\}, e_2 \in \mathbb{I}(\tilde{\vartheta}_1) \backslash \{\{\tilde{\vartheta}_1, \tilde{\vartheta}_2\}\} : e_1 < e_2,$$

where $\mathbb{I}$ is the incidence map (see Definition 3.1.5). As usual, $\ell := \#E$. We define

$$\begin{aligned} v'_d : \mathbb{N}^{g(d)+g(d+1)-1} &\to \mathcal{V}_d \\ (j_1, \dots, j_{g(d)+g(d+1)-1}) &\mapsto \sum_j w_1(j_1, \dots, j_{g(d+1)-1}, j) v_d(j_{g(d+1)}, \dots, j_{g(d)-1}, j), \end{aligned}$$

with g being the degree map as of Definition 3.1.5, such that we obtain the formula

$$v(v_1, \dots, v_d, w_1, \dots, w_L) = v'(v_1, \dots, v_{d-1}, v'_d, w_2, \dots, w_L),$$

which satisfies our conditions. □

Expressed in simple words, this lemma means that we can neglect vertices of the graph that have a scalar meaning in a certain sense. We have to *bind* the scalar vertex either to a vector space vertex or another scalar vertex which this scalar vertex is connected to by one edge. The other edges of the neglected vertex will be connected to the bound vertex. This can be successively applied.

In Subsection 2.2.8, we implicitly used Lemma 3.2.9 to convert a hierarchical tensor to a tensor that is represented in the Tucker format.

3.3 Examples

In Section 2.1, we described various examples of common tensor formats. As we now have defined a general framework, we are able to describe the definitions in this context.

The r-term format is the most simple and fundamental structure in the tensor network context (see Section 3.4), as there is only one edge.

Example 3.3.1 (r-term format). *Let $V = \{\vartheta_1, \dots, \vartheta_d\}$ with $d \in \mathbb{N}$ be the set of vertices and*

$$\mathcal{E} := \{E\}$$

with

$$E := \{\{\vartheta_1, \dots, \vartheta_d\}\}$$

the set of edge sets. Then the tensor format graph set $(V, \mathcal{E})$ defines the r-term format *of order d which has only one edge that connects all vertices.*
Note that (V, E) is a multi-graph as of Definition 3.1.3. Compare Definition 2.1.1.

Example 3.3.2 (Tucker format/ subspace format). *Let $V = \{\vartheta_1, \dots, \vartheta_{d+1}\}$ with $d \in \mathbb{N}$ be the set of vertices and*

$$\mathcal{E} = \{\{\{\vartheta_1, \vartheta_{d+1}\}, \dots, \{\vartheta_d, \vartheta_{d+1}\}\}\}$$

the set of edge sets. Then the tensor format graph set $(V, \mathcal{E})$ defines the Tucker format *(or* subspace format*) of order d which is a tensor format of order $(d, 1)$. Compare Definition 2.1.2.*

Example 3.3.3 (TT format/ MPS format). *Let $V = \{\vartheta_1, \dots, \vartheta_d\}$ with $d \in \mathbb{N}$ be the set of vertices and*

$$\mathcal{E} = \{\{\{\vartheta_1, \vartheta_2\}, \{\vartheta_2, \vartheta_3\}, \dots, \{\vartheta_{d-1}, \vartheta_d\}\}\}$$

the set of edge sets. Then the tensor format graph set $(V, \mathcal{E})$ defines the TT format *(or* MPS format*) of order d. Compare Definition 2.1.8.*

By adding an additional edge $\{\vartheta_d, \vartheta_1\}$ to the set of edges of the TT format, we get

Example 3.3.4 (TC format). *Let $V = \{\vartheta_1, \dots, \vartheta_d\}$ with $d \in \mathbb{N}$ be the set of vertices and*

$$\mathcal{E} = \{\{\{\vartheta_d, \vartheta_1\}, \{\vartheta_1, \vartheta_2\}, \{\vartheta_2, \vartheta_3\}, \dots, \{\vartheta_{d-1}, \vartheta_d\}\}\}$$

the set of edge sets. Then the tensor format graph set $(V, \mathcal{E})$ defines the TC format *of order d. Compare Definition 2.1.11.*

As of [31, definition of a cycle of a graph on p. 40], we get

Definition 3.3.5 (Cycle of a graph). *Let $G = (V, E)$ be an undirected and connected graph. We say, G has a* cycle*, if and only if there are at least one sequence $(a_1, \dots, a_n) \in V^n$ with $n \in \mathbb{N} \backslash \{1, 2\}$ such that*

$$\begin{aligned} \{a_i, a_{i+1}\} \in E \ \forall i \in \{1, \dots, n-1\}, \\ a_1 = a_n \end{aligned}$$

and

$$a_i \neq a_j \ \forall i \in \{1, \dots, j-1\}, j \in \{2, \dots, n-1\}.$$

The TC format graph as of Example 3.3.4 has a cycle.

Definition 3.3.6 (Tree ([31, Definition on p. 43]), leaf set)**.** *Let $G = (V, E)$ be a graph. G is called a* tree *if it does not contain cycles. The* leaf set *of a tree $G = (V, E)$ is defined as*

$$\{\vartheta \in V : g(\vartheta) = 1\}$$

where g is the degree map of G (see Definition 3.1.5).

Example 3.3.7 (Hierarchical format)**.** *Let $V = \{\vartheta_1, \dots, \vartheta_{2d-1}\}$ with $d \in \mathbb{N}$ be the set of vertices and*

$$\mathcal{E} = \left\{ E \subset V \bigsqcup V : (V, E) \text{ is a tree with leaf set } \{\vartheta_1, \dots, \vartheta_d\} \right\}$$

the set of edge sets. Then the tensor format graph set $(V, \mathcal{E})$ defines the hierarchical format *of order d which is a tensor format of order $(d, d-1)$.*

With Definition 2.1.4 and Definition 2.1.6, we introduced two special cases of the hierarchical format.

Definition 3.3.8 (Grid graph)**.** *Let $d_1, d_2 \in \mathbb{N}$, $V = \{\vartheta_{1,1}, \dots, \vartheta_{d_1,1}, \vartheta_{1,2}, \dots, \vartheta_{d_1,d_2}\}$ be a set of vertices such that $\#V = d_1 \cdot d_2$ and edge set*

$$E_{grid}(V, d_1, d_2) := \bigcup_{i=1}^{d_2} \bigcup_{j=1}^{d_1-1} \{\{\vartheta_{j,i}, \vartheta_{j+1,i}\}\} \cup \bigcup_{i=1}^{d_1} \bigcup_{j=1}^{d_2-1} \{\{\vartheta_{i,j}, \vartheta_{i,j+1}\}\}$$

then $(V, E_{grid}(V, d_1, d_2))$ is defined as the grid graph of order (d_1, d_2) *with vertex set V.*

Example 3.3.9 (PEPS format)**.** *Let $V = \{\vartheta_1, \dots, \vartheta_d\}$ with $d \in \mathbb{N}$ be the set of vertices and*

$$\mathcal{E} = \bigcup_{\substack{d_1, d_2 \in \mathbb{N} \\ d_1 \cdot d_2 = d}} \{E_{grid}(V, d_1, d_2)\}$$

the set of edge sets such that for all $E \in \mathcal{E}$ holds, that (V, E) is a grid graph. Then the tensor format graph set $(V, \mathcal{E})$ defines the PEPS format *of order d.*

Another example for a non-trivial multi-graph tensor network is the tensor hypercontraction (or THC) format that has been introduced in [33, (see Equation (20) for the formulation)].

Example 3.3.10 (THC format)**.** *Let $V = \{\vartheta_1, \dots, \vartheta_5\}$ be the set of vertices and*

$$\mathcal{E} = \{\{\{\vartheta_1, \vartheta_2, \vartheta_5\}, \{\vartheta_3, \vartheta_4, \vartheta_5\}\}\}$$

*be the set of edge sets. Then the tensor format graph set $(V, \mathcal{E})$ of order $(4, 1)$ defines the THC (or **t**ensor **h**yper**c**ontraction) format.*

This leads a THC tensor format to be defined by the summation structure

$$\bigtimes_{\mu=1}^{4} \{f : \mathbb{N} \to \mathcal{V}_\mu\} \times \{f : \mathbb{N}^2 \to \mathbb{K}\} \to \mathcal{V}$$

$$(v_1, \ldots, v_4, w) \mapsto \sum_{j_1, j_2} w(j_1, j_2) \cdot (v_1(j_1) \otimes v_2(j_1) \otimes v_3(j_2) \otimes v_4(j_2))$$

which is represented in Figure 3.1 (the reader is reminded of Section 2.0 which contains the explanation of the graphical notation).

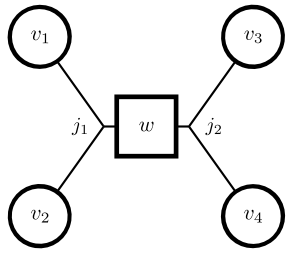

Figure 3.1: THC tensor[1]

3.4 Connection to the r-term format

As each tensor network representation can be interpreted as an r-term representation (with a possibly very large rank, see Section 2.2), one can also use a lot of algorithms that act on the spatial directions of a r-term tensor.

All algorithms that we use and describe for tensor network representations are especially valid for r-term tensors. A tensor representation in the r-term format has the *advantage* of being optimal in terms of complexity with respect to tensor network representations. That means that if the tensor is represented as an r-term tensor, we can treat it as a general tensor network representation without loosing properties of the r-term tensor as long as we do not act on single terms. A direct result of this is, that the implementation for general tensor network representations can be used also directly for r-term tensors without having a relevant computational disadvantage with respect to implementations that are meant and optimized for r-term tensors with the exception of working on single terms (see Subsection 4.1.2 and Section 4.2).

Remark 3.4.1. *For $d = 2$, all formats, that we described in this chapter, are equal (if $L > 0$, we can apply Lemma 3.2.9).*

[1] source: [34, Figure 1]

Chapter 4

Approximation algorithms

One of the main task of numerical analysis in the tensor representation context is to perform rank-compressions of the represented tensor in order to reduce the storage cost and to decrease the computational complexity. This can be done by computing low rank approximations of given tensors.

In this chapter, we will explain a selected set of algorithms that can be used in the tensor network framework to obtain approximating tensor representations within a given error bound.

4.0 Objective functions

In general, our goal is to minimize a certain function that has a tensor representation as its very own argument. This tensor is represented in *some* tensor format. As of [1, Section 1, (i)–(iv)] functions of interest may be

$$v \mapsto \|Av - a\|^2, \tag{4.1}$$

which can be used to approximate the solution of the equation $Av = a$, the function

$$v \mapsto \frac{1}{2}\langle v, v\rangle - \langle v, a\rangle \tag{4.2}$$

or the Rayleigh quotient, that can be used to determine the smallest Eigenvalue of A:

$$v \mapsto \frac{\langle Av, v\rangle}{\langle v, v\rangle} \tag{4.3}$$

where $v, a \in \mathcal{V}$ are tensor representations and $A : \mathcal{V} \to \mathcal{V}$. If A is the identity in $\mathcal{V}$ and if the inner product is symmetric with respect to $\mathcal{V}$, functions (4.1) and (4.2) have their minimum at the same point.

In the approximation algorithms that we will describe in this chapter, we use function (4.2) as the objective function such that we require the inner product to be symmetric. Therefore, we will refer to function (4.2) as *the objective function* or *our objective function.*

4.1 Non-linear block Gauss-Seidel

The Gauss-Seidel iteration is characterized in [35, Section 7.4] as follows:

Description. *We have a linear system*

$$Ax = b$$

that we want to solve iteratively where A is a $\mathbb{K}$-valued $n \times n$ matrix that is given as well as $b \in \mathbb{K}^n$. For $0 < i \leqslant n$, the ith component $x_i^{(k+1)}$ of the $(k+1)$th iterate $x^{(k+1)}$ is computed by the the components $x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$ of the $(k+1)$th iterate and the components $x_{i+1}^{(k)}, \ldots, x_n^{(k)}$ of the kth iterate $x^{(k)}$.

Compare also [1, Section 6]. A possibility for the initial guess $x^{(0)}$ is treated in Section 4.2. An alternative approach for the initial guess can be found in [1, Subsection 7.3].

This section also is based on [1] such that Subsections 4.1.2 and 4.1.3 are generalized versions of the corresponding sections and definitions of [1, Subsection 7.1 and Subsection 7.2].
The ALS and DMRG method is formulated in the setting of general tensor networks.

4.1.1 Partitioning of coordinates

To abstractly define the components of an iterate (e.g. ith component of the kth iterate above), we state a general partitioning scheme which has been introduced in [1, Definition 6.1]. Since we do not use a so called *parameter space*, we are going to adjust and modify the definition as follows:

Definition 4.1.1 (Partitioning of coordinates)**.** *Let $X_i \subset \{1, \ldots, d+L\} \times \left(\bigtimes_{k=1}^{\ell} \{1, \ldots, r_k\} \right)$ for $\ell \in \mathbb{N}$, $r_1, \ldots, r_\ell \in \mathbb{N}$ and $i \in I$ where I is an arbitrary index set. Then we define*

$$X := \{X_i \mid i \in I\}$$

to be a partitioning of coordinates *iff*

$$\bigcup_{i \in I} X_i = \{1, \ldots, d+L\} \times \left(\bigtimes_{k=1}^{\ell} \{1, \ldots, r_k\} \right).$$

We say the partitioning is disjoint, if for all $X_i, X_j \in X$ with $X_i \neq X_j$ we have $X_i \cap X_j = \varnothing$.

By that definition, the partitioning of coordinates is a decomposition of the directions and the representation rank tuples of a tensor representation.

4.1.2 ALS

The **A**lternating **L**east **S**quares (ALS) method is a non-linear block Gauss-Seidel method that depends on a partitioning that does not allow a structural change of the tensor representation and the representation rank. In general, for a tensor representation of order $(d, L) \in \mathbb{N} \times \mathbb{N}_0$ that is defined by the tensor format graph (V, E) with representation rank $(r_1, \dots, r_\ell) \in \mathbb{N}^\ell$ where $\ell := \#E$, we have two different types of partitionings:

$$X_i := \left\{ (i, j_1, \dots, j_\ell) \mid (j_1, \dots, j_\ell) \in \bigtimes_{\mu=1}^{\ell} \{1, \dots, r_\mu\} \right\} \tag{4.4}$$

on the one hand, where the index set I is $\{1, \dots, d + L\}$, and on the other hand

$$X_i := \{(\mu, i_1, \dots, i_\ell) \mid \mu \in \{1, \dots, d + L\}\} \tag{4.5}$$

where the index set I is $\bigtimes_{\mu=1}^{\ell} \{1, \dots, r_\mu\}$, compare [1, before Definition 6.1]. Partitioning (4.4) allows to optimize over each spatial dimension separately, whereas partitioning (4.5) allows to optimize over single terms of the representation sum. However, optimizing over single terms has huge hassles in complex tensor networks whereas it is easy in the r-term format (i.e. rank-one updates), which we describe later on.

Optimizing w.r.t. spatial directions

So we are now fixing all spatial direction but the one that we want to optimize, compare [1, Subsection 7.1]. We differentiate with respect to this fixed direction in terms of our objective function (4.2). Let (V, E) and $(\tilde{V}, \tilde{E})$ be two tensor network graphs of order $(d, L) \in \mathbb{N} \times \mathbb{N}_0$ with $\ell := \#E$ and $k := \#\tilde{E}$. Let furthermore v and a be tensor representations defined by (V, E) and $(\tilde{V}, \tilde{E})$, respectively with representation ranks $(r_1, \dots, r_\ell) \in \mathbb{N}^\ell$ and $(\tilde{r}_1, \dots, \tilde{r}_k) \in \mathbb{N}^k$, respectively. We differentiate with respect to direction $\nu \in \{1, \dots, d + L\}$ where we assume

$$\mathbb{I}_{(V,E)}(\nu) = (1, \dots, \ell_1)$$

and

$$\mathbb{I}_{(\tilde{V},\tilde{E})}(\nu) = (1, \dots, k_1)$$

are the corresponding incidence maps and consequently

$$\mathcal{I}_{(V,E)}(\nu, (j_1, \dots, j_\ell)) = (j_1, \dots, j_{\ell_1}) \tag{4.6}$$

and

$$\mathcal{I}_{(\tilde{V},\tilde{E})}(\nu, (j_1, \dots, j_k)) = (j_1, \dots, j_{k_1}) \tag{4.7}$$

are the corresponding incidence* maps, where $\ell_1 \leqslant \ell$ and $k_1 \leqslant k$. To keep the notation slightly shorter, we set $\mathbb{I} := \mathbb{I}_{(V,E)}, \mathbb{I}_a := \mathbb{I}_{(\tilde{V},\tilde{E})}, \mathcal{I} := \mathcal{I}_{(V,E)}$ and $\mathcal{I}_a := \mathcal{I}_{(\tilde{V},\tilde{E})}$ and obtain (in the simplified notation, see Definition 3.2.8)

$$\begin{aligned} v &= \sum_{j_1,\dots,j_\ell=1}^{r_1,\dots,r_\ell} \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \\ &\cong \sum_{j_1,\dots,j_{\ell_1}=1}^{r_1,\dots,j_{\ell_1}} v_\nu(j_1,\dots,j_{\ell_1}) \otimes \left(\sum_{j_{\ell_1+1},\dots,j_\ell=1}^{r_{\ell_1+1},\dots,r_\ell} \bigotimes_{\substack{\mu=1 \\ \mu\neq\nu}}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \right) \end{aligned}$$

as of (4.6) and analogously

$$\begin{aligned} a &= \sum_{i_1,\dots,i_k=1}^{\tilde{r}_1,\dots,\tilde{r}_k} \bigotimes_{\mu=1}^{d+L} \tilde{v}_\mu(\mathcal{I}_a(\mu,(i_1,\dots,i_k))) \\ &\cong \sum_{i_1,\dots,i_{k_1}=1}^{\tilde{r}_1,\dots,\tilde{r}_{k_1+1}} \tilde{v}_\nu(i_1,\dots,i_{k+1}) \otimes \left(\sum_{i_{k_1+1},\dots,i_k=1}^{\tilde{r}_{k_1+1},\dots,\tilde{r}_k} \bigotimes_{\substack{\mu=1 \\ \mu\neq\nu}}^{d+L} \tilde{v}_\mu(\mathcal{I}_a(\mu,(i_1,\dots,i_k))) \right) \end{aligned}$$

as of (4.7) such that we have

$$\begin{aligned} \langle v, v\rangle = \sum_{j_1,\dots,j_{\ell_1}=1}^{r_1,\dots,r_{\ell_1}} \sum_{i_1,\dots,i_{\ell_1}=1}^{r_1,\dots,r_{\ell_1}} &\langle v_\nu(j_1,\dots,j_{\ell_1}), v_\nu(i_1,\dots,i_{\ell_1})\rangle \\ &\underbrace{\left(\sum_{j_{\ell_1+1},\dots,j_\ell=1}^{r_{\ell_1+1},\dots,r_\ell} \sum_{i_{\ell_1+1},\dots,i_\ell=1}^{r_{\ell_1+1},\dots,r_\ell} \prod_{\substack{\mu=1 \\ \mu\neq\nu}}^{d+L} \langle v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))), v_\mu(\mathcal{I}(\mu,(i_1,\dots,i_\ell)))\rangle \right)}_{=:V_{[\nu]}(j_1,\dots,j_{\ell_1},i_1,\dots,i_{\ell_1})} \end{aligned}$$

and

$$\begin{aligned} \langle v, a\rangle = \sum_{j_1,\dots,j_{\ell_1}=1}^{r_1,\dots,r_{\ell_1}} \sum_{i_1,\dots,i_{k_1}=1}^{\tilde{r}_1,\dots,\tilde{r}_{k_1}} &\langle v_\nu(j_1,\dots,j_{\ell_1}), a_\nu(i_1,\dots,i_{k_1})\rangle \\ &\underbrace{\left(\sum_{j_{\ell_1+1},\dots,j_\ell=1}^{r_{\ell_1+1},\dots,r_\ell} \sum_{i_{k_1+1},\dots,i_k=1}^{\tilde{r}_{k_1+1},\dots,\tilde{r}_k} \prod_{\substack{\mu=1 \\ \mu\neq\nu}}^{d+L} \langle v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))), a_\mu(\mathcal{I}_a(\mu,(i_1,\dots,i_k)))\rangle \right)}_{=:A_{[\nu]}(j_1,\dots,j_{\ell_1},i_1,\dots,i_{k_1})}. \end{aligned}$$

Differentiating with respect to direction ν, leaves us for all fixed $(j_1,\dots,j_{\ell_1}) \in \bigtimes_{\mu=1}^{\ell_1}\{1,\dots,r_\mu\}$ with

$$\frac{\partial\left(\frac{1}{2}\langle v,v\rangle - \langle v,a\rangle\right)}{\partial v_\nu(j_1,\dots,j_{\ell_1})} = \sum_{i_1,\dots,i_{\ell+1}=1}^{r_1,\dots,r_{\ell_1}} v_\nu(i_1,\dots,i_{\ell_1})V_{[\nu]}(j_1,\dots,j_{\ell_1},i_1,\dots,i_{\ell_1}) - \sum_{i_1,\dots,i_{k_1}=1}^{\tilde{r}_1,\dots,\tilde{r}_{k_1}} a_\nu(i_1,\dots,i_{k_1})A_{[\nu]}(j_1,\dots,j_{\ell_1},i_1,\dots,i_{k_1}),$$

which leads in the derivative set equal to zero to

$$\begin{aligned}\sum_{i_1,\dots,i_{\ell_1}=1}^{r_1,\dots,r_{\ell_1}} v_\nu(i_1,\dots,i_{\ell_1})V_{[\nu]}(j_1,\dots,j_{\ell_1},i_1,\dots,i_{\ell_1}) \\ = \sum_{i_1,\dots,i_{k_1}=1}^{\tilde{r}_1,\dots,\tilde{r}_{k_1}} a_\nu(i_1,\dots,i_{k_1})A_{[\nu]}(j_1,\dots,j_{\ell_1},i_1,\dots,i_{k_1})\end{aligned} \tag{4.8}$$

for all fixed $j_1,\dots,j_{\ell_1}$. Analogously to [1, Subsection 7.1], in terms of a matrix notation with

$$\begin{aligned}\mathbf{A}_{[\nu]} &= \left(\left(A_{[\nu]}(j_1,\dots,j_{\ell_1},i_1,\dots,i_{k_1})\right)_{(j_1,\dots,j_{\ell_1}),(i_1,\dots,i_{k_1})}\right) \in \mathbb{K}^{\prod_{\mu=1}^{\ell_1} r_\mu \times \prod_{\mu=1}^{k_1}\tilde{r}_\mu},\\ \mathbf{a}_\nu &= \begin{pmatrix} a_\nu(1,\dots,1)\\ \vdots \\ a_\nu(\tilde{r}_1,\dots,\tilde{r}_{k_1})\end{pmatrix} \in \mathbb{K}^{n_\nu \prod_{\mu=1}^{k_1}\tilde{r}_\mu},\\ \mathbf{V}_{[\nu]} &= \left(\left(V_{[\nu]}(j_1,\dots,j_{\ell_1},i_1,\dots,i_{\ell_1})\right)_{(j_1,\dots,j_{\ell_1}),(i_1,\dots,i_{\ell_1})}\right) \in \mathbb{K}^{\prod_{\mu=1}^{\ell_1} r_\mu \times \prod_{\mu=1}^{\ell_1} r_\mu}\end{aligned}$$

and

$$\mathbf{v}_\nu = \begin{pmatrix} v_\nu(1,\dots,1)\\ \vdots \\ v_\nu(r_1,\dots,r_{\ell_1})\end{pmatrix} \in \mathbb{K}^{n_\nu \prod_{\mu=1}^{\ell_1} r_\mu}$$

we get an alternative notation of Equation (4.8)

$$(\mathbf{A}_{[\nu]} \otimes Id_{\mathcal{V}_\nu})\mathbf{a}_\nu = (\mathbf{V}_{[\nu]} \otimes Id_{\mathcal{V}_\nu})\mathbf{v}_\nu \tag{4.9}$$

that we have to solve in one, so called, *ALS step*. We can either do this by inverting $\mathbf{V}_{[\nu]}$ directly or, if the inverse does not exist, by computing its pseudo-inverse based on [1, Remark 7.1].

The computation of $\mathbf{A}_{[\nu]}$ and $\mathbf{V}_{[\nu]}$, respectively, depends on the tensor format. It has been described it in details in [1, Subsection 7.1] for the Tensor Chain format where, as

of [1, Corollary 7.3], the computational complexity is $\mathcal{O}(dr^3\tilde{r}^3)$ for $\mathbf{A}_{[\nu]}$ and $\mathcal{O}(dr^6)$ for $\mathbf{V}_{[\nu]}$ with

$$r := \max_{i \in \{1,\dots,\ell\}} r_i$$

and

$$\tilde{r} := \max_{i \in \{1,\dots,k\}} \tilde{r}_i.$$

Now we can describe the ALS Method for arbitrary tensor networks. Similar to [1, Algorithm 2], we state Algorithm 1.

Algorithm 1 ALS algorithm for general tensor networks w.r.t. spatial directions

1: **while** error condition not fulfilled **do**
2: **for** $\nu \in \{1, \dots, d + L\}$ **do**
3: find $\tilde{\mathbf{v}}_\nu$ such that
$$(\mathbf{A}_{[\nu]} \otimes Id_{\mathcal{V}_\nu})\mathbf{a}_\nu = (\mathbf{V}_{[\nu]} \otimes Id_{\mathcal{V}_\nu})\tilde{\mathbf{v}}_\nu$$
as of Equation (4.9)
4: $\mathbf{v}_\nu \mapsto \tilde{\mathbf{v}}_\nu$
5: **end for**
6: **end while**

The *error condition* is usually that $\|\mathbf{v}_\nu - \tilde{\mathbf{v}}_\nu\| <$ some threshold. The order in which we choose ν out of $\{1, \dots, d + L\}$ does a priori not matter. Solving Equation (4.9) is the crucial part of the algorithm (*ALS step*). Depending on the tensor format of v and a, we have different computational costs for solving one ALS step. As of [1, Lemma 7.4], if the tensor format of a and v is the TC format, the computational cost of solving Equation (4.9) is in

$$\mathcal{O}(dr^6) + \mathcal{O}(dr^3\tilde{r}^6) + \mathcal{O}(n(r^2\tilde{r}^2 + r^4))$$

(note that $L = 0$ in the TC format). Optimizing over all dimensions (i.e. performing the for loop of Algorithm 1) has in general $d + L$ times the complexity of a single ALS step. For certain formats however, this complexity can be reduced by introducing a prephase before each loop where intermediate matrices are computed (see [1, before Algorithm 2]). For our frequently used example of the TC format, as shown in [1, Lemma 7.5], this results in a complexity of

$$\mathcal{O}(dr^6) + \mathcal{O}(dr^3\tilde{r}^3) + \mathcal{O}(dn(r^2\tilde{r}^2 + r^4)),$$

which is still linear in d, for a whole ALS cycle. So we see that the complexity highly depends on the structure of the tensor representation. For the TC format, this however adds a constraint about the order in which to traverse $\{1, \dots, d\}$ (the set has to be traversed ascending or descending).

Optimizing w.r.t. single terms

Instead of fixing the spatial directions, we fix one single term of the sum of elementary tensors that define the tensor (see partitioning (4.5)). The tensor representations v and a are as defined before. Here, we choose a fixed

$$(b_1, \dots, b_\ell) \in \bigtimes_{\mu=1}^{\ell} \{1, \dots, r_\mu\}$$

we are optimizing the term such that we have

$$v = \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (b_1, \dots, b_\ell))) + \sum_{\substack{j_1,\dots,j_\ell=1 \\ (j_1,\dots,j_\ell)\neq(b_1,\dots,b_\ell)}}^{r_1,\dots,r_\ell} \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell)))$$

where we only optimize the first term. In the r-term format, we can perform an optimization with respect to the first term. In the case of general tensor networks however, this is not possible. We will state the reasons below. We have

$$\begin{aligned}
\langle v, v\rangle = &\left\langle \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (b_1, \dots, b_\ell))), \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (b_1, \dots, b_\ell))) \right\rangle \\
&+ \sum_{\substack{j_1,\dots,j_\ell=1 \\ (j_1,\dots,j_\ell)\neq(b_1,\dots,b_\ell)}}^{r_1,\dots,r_\ell} \left\langle \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (b_1, \dots, b_\ell))), \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell))) \right\rangle \\
&+ \sum_{\substack{j_1,\dots,j_\ell=1 \\ (j_1,\dots,j_\ell)\neq(b_1,\dots,b_\ell)}}^{r_1,\dots,r_\ell} \left\langle \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell))), \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (b_1, \dots, b_\ell))) \right\rangle \\
&+ \sum_{\substack{j_1,\dots,j_\ell=1 \\ (j_1,\dots,j_\ell)\neq(b_1,\dots,b_\ell)}}^{j_1,\dots,r_\ell} \sum_{\substack{i_1,\dots,i_\ell=1 \\ (i_1,\dots,i_\ell)\neq(b_1,\dots,b_\ell)}}^{r_1,\dots,r_\ell} \left\langle \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell))), \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (i_1, \dots, i_\ell))) \right\rangle
\end{aligned}$$

and

$$\begin{aligned}
\langle v, a\rangle = &\sum_{i_1,\dots,i_k=1}^{\tilde{r}_1,\dots,\tilde{r}_k} \left\langle \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (b_1, \dots, b_\ell))), \bigotimes_{\mu=1}^{d+L} a_\mu(\mathcal{I}_a(\mu, (i_1, \dots, i_k))) \right\rangle \\
&+ \sum_{\substack{j_1,\dots,j_\ell=1 \\ (j_1,\dots,j_\ell)\neq(b_1,\dots,b_\ell)}}^{r_1,\dots,r_\ell} \sum_{i_1,\dots,i_k=1}^{\tilde{r}_1,\dots,\tilde{r}_k} \left\langle \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu, (j_1, \dots, j_\ell))), \bigotimes_{\mu=1}^{d+L} a_\mu(\mathcal{I}_a(\mu, (i_1, \dots, i_k))) \right\rangle
\end{aligned}$$

such that by differentiating with respect to $\bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(b_1,\dots,b_\ell)))$, we obtain

$$\frac{\partial\left(\frac{1}{2}\langle v,v\rangle-\langle v,a\rangle\right)}{\partial \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(b_1,\dots,b_\ell)))} = \sum_{j_1,\dots,j_\ell=1}^{r_1,\dots,r_\ell} \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) - \sum_{i_1,\dots,i_k=1}^{\tilde{r}_1,\dots,\tilde{r}_k} \bigotimes_{\mu=1}^{d+L} a_\mu(\mathcal{I}_a(\mu,(i_1,\dots,i_k))).$$

Setting this derivative equal to zero allows us to exactly determine $\bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(b_1,\dots,b_\ell)))$:

$$\bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(b_1,\dots,b_\ell))) = -\sum_{\substack{j_1,\dots,j_\ell=1\\(j_1,\dots,j_\ell)\neq(b_1,\dots,b_\ell)}}^{r_1,\dots,r_\ell} \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) + \sum_{i_1,\dots,i_k=1}^{\tilde{r}_1,\dots,\tilde{r}_k} \bigotimes_{\mu=1}^{d+L} a_\mu(\mathcal{I}_a(\mu,(i_1,\dots,i_k))) \tag{4.10}$$

but this does not help at all, which is what we explain next. If the set

$$H_\mu(b) := \left\{ c \in \bigtimes_{i=1}^{\ell} \{1,\dots,r_i\} \mid c \neq b, \mathcal{I}(\mu,b) = \mathcal{I}(\mu,c) \right\} \tag{4.11}$$

is empty for all $\mu \in \{1,\dots,d+L\}$ and all $b \in \bigtimes_{i=1}^{\ell} \{1,\dots,r_i\}$, we get

$$\left(\bigcup_{\substack{c \in \bigtimes_{i=1}^{\ell} \{1,\dots,r_i\}\\ c\neq b}} \{\mathcal{I}(\mu,c) \mid \mu \in \{1,\dots,d+L\}\} \right) \cap \{\mathcal{I}(\mu,b) \mid \mu \in \{1,\dots,d+L\}\} = \varnothing, \tag{4.12}$$

so the LHS of Equation (4.10) is independent of the RHS of Equation (4.10).

We can state the following theorem, which tells us that we can perform ALS with respect to single terms if and only if the tensor is representation in the r-term format.

Theorem 4.1.2. *$H_\mu(b)$ as of (4.11) is empty for all $\mu \in \{1,\dots,d+L\}$ and all $b \in \bigtimes_{i=1}^{\ell} \{1,\dots,r_i\}$ if and only if v is a tensor represented in the r-term format.*

Proof. Let us assume that v is represented in the r-term format with representation rank $r \in \mathbb{N}$. Then we have $\mathcal{I}(\mu,b) = b$ for all $b \in \{1,\dots,r\}$ and all $\mu \in \{1,\dots,d+L\}$. Therefore, $H_\mu(b)$ is always an empty set since by definition, it cannot contain b.

On the other hand, Equation (4.12) implies the terms of the tensor representation to be independent of each other with respect to the summation index. This is only true for the r-term format (and for elementary tensors). □

We may conclude in saying that the ALS method applied to single terms of a tensor representation as we described it above, is not possible in general, but only for the r-term format. This renders the method to be less important in comparison to the other optimization methods that are mentioned in this work. Single parameters cannot identify single terms (i.e. multiple terms have a dependency to one single summation index value) and therefore, we cannot optimize these single terms with respect to the single parameters, which is the underlying problem.

4.1.3 DMRG

If we allow partitionings to overlap (i.e. for $X_i, X_i \in X$ we allow for $X_i \neq X_j, X_i \cap X_j \neq \varnothing$ for some $i, j \in I$ w.r.t. Definition 4.1.1 – see [1, before Definition 6.1]), we can get to a different kind of Gauss-Seidel algorithm as described for the TC format in [1, Subsection 7.2].

This leads us to the so called **D**ensity **M**atrix **R**enormalization **G**roup algorithm - or DMRG algorithm. This method allows us to adjust two directions in one step which enables us to adjust the rank of the edge between both vertices that represent the two directions, if there is a non-trivial (i.e. rank > 1) edge between them. We have the same objective function (4.2) as before, i.e. we minimize the distance between the two tensors v and a with respect to some norm by adjusting the representation of v (and v itself). The algorithm is well known in the chemists community since 1992 (see [36]) and has been extended and brought to the tensor representation context (in the sense of this thesis) in 2010[1] by Reinhold Schneider[2] (see also [37]).

We use the same notation as before in Subsection 4.1.2 for the tensor representations of v and a as well as for their corresponding incidence maps $\mathbb{I}, \mathbb{I}_a$ and incidence* maps $\mathcal{I}, \mathcal{I}_a$. In terms of our definition of the partitioning of the coordinates, we first define the index set

$$I := \{\{a,b\} \in \{1,\dots,d+L\} \biguplus \{1,\dots,d+L\} \mid \#(\mathbb{I}(a) \cap \mathbb{I}(b)) = 1, \\ \mathbb{I}(a) \cap \mathbb{I}(b) \cap \mathbb{I}(c) = \varnothing \ \forall c \notin \{a,b\}\}$$

such that we define $\forall\ i = (i_1, i_2) \in I$ the partitionings

$$X_i := \left\{ (i_1, j_1, \dots, j_\ell), (i_2, j_1, \dots, j_\ell) \mid \{j_1, \dots, j_\ell\} \in \bigtimes_{\mu=1}^{\ell} \{1, \dots, r_\mu\} \right\}.$$

Note that each element of I is a 2-tuple. So I contains the edges that connect exactly two vertices.
We simplify the notation slightly: Let $\nu_1, \nu_2 \in \{1,\dots,d+L\}$ with $\nu_1 \neq \nu_2$ and $\ell_1, \ell_2 \in \{1,\dots,\ell\}$ with $\ell_1 < \ell_2$. Furthermore, $\mathbb{I}(\nu_1) = \{1,\dots,\ell_1\}, \mathbb{I}(\nu_2) = \{\ell_1,\dots,\ell_2\}$ and $\ell_1 \notin \mathbb{I}(\mu)\ \forall \mu \in \{1,\dots,d+L\} \backslash \{\nu_1, \nu_2\}$ such that vertex ν_1 and ν_2 have exactly one edge in

[1] http://www.mis.mpg.de/calendar/conferences/2010/wskhor.html
[2] http://page.math.tu-berlin.de/~schneidr/

common (which is ℓ_1) that only connects the vertices ν_1 and ν_2. We additionally assume $\mathbb{I}_a(\nu_1) \cup \mathbb{I}_a(\nu_2) = \{1, \dots, k_1\}$ for simplicity. As a generalization of [1, Subsection 7.2], we therefore have

$$\begin{aligned} v \cong \sum_{j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},j_{\ell_2}=1}^{r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2}} \left(\sum_{j_{\ell_1}=1}^{r_{\ell_1}} v_{\nu_1}(j_1,\dots,j_{\ell_1}) \otimes v_{\nu_2}(j_{\ell_1},\dots,j_{\ell_2}) \right) \\ \otimes \left(\sum_{j_{\ell_2+1},\dots,j_\ell=1}^{r_{\ell_2+1},\dots,r_\ell} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{\nu_1,\nu_2\}}}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_{\ell_1-1},0,j_{\ell_1+1},\dots,j_\ell))) \right) \end{aligned}$$

where we keep in mind that $\ell_1 \notin \mathbb{I}(\mu)$ for $\mu \in \{1,\dots,d+L\}\backslash\{\nu_1,\nu_2\}$, and analogously

$$\begin{aligned} a \cong \sum_{j_1,\dots,j_{k_1}=1}^{\tilde{r}_1,\dots,\tilde{r}_{k_1}} a_{\nu_1}(\mathcal{I}_a(\nu_1,(j_1,\dots,j_{k_1},0,\dots))) \otimes a_{\nu_2}(\mathcal{I}_a(\nu_2,(j_1,\dots,j_{k_1},0,\dots))) \\ \otimes \left(\sum_{j_{k_1+1},\dots,j_k=1}^{\tilde{r}_{k_1+1},\dots,\tilde{r}_k} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{\nu_1,\nu_2\}}}^{d+L} a_\mu(\mathcal{I}_a(\mu,(j_1,\dots,j_k))) \right) \end{aligned}$$

such that by defining

$$v_{\nu_1,\nu_2}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}) := \sum_{j_{\ell_1}=1}^{r_{\ell_1}} v_{\nu_1}(j_1,\dots,j_{\ell_1}) \otimes v_{\nu_2}(j_{\ell_1},\dots,j_{\ell_2})$$

and

$$a_{\nu_1,\nu_2}(i_1,\dots,i_{k_1}) := a_{\nu_1}(\mathcal{I}_a(\nu_1,(i_1,\dots,i_{k_1},0,\dots))) \otimes a_{\nu_2}(\mathcal{I}_a(\nu_2,(i_1,\dots,i_{k_1},0,\dots)))$$

we conclude in

$$\begin{aligned} \langle v, v\rangle = \sum_{j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots j_{\ell_2}=1}^{r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2}} \sum_{i_1,\dots,i_{\ell_1-1},i_{\ell_1+1},\dots,i_{\ell_2}=1}^{r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2}} \langle v_{\nu_1,\nu_2}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}), \\ v_{\nu_1,\nu_2}(i_1,\dots,i_{\ell_1-1},i_{\ell_1+1},\dots,i_{\ell_2})\rangle \\ \cdot \underbrace{\left(\sum_{j_{\ell_2+1},\dots,j_\ell,j_{\ell_1}=1}^{r_{\ell_2+1},\dots,r_\ell,1} \sum_{i_{\ell_2+1},\dots,i_\ell,i_{\ell_1}=1}^{r_{\ell_2+1},\dots,r_\ell,1} \prod_{\substack{\mu=1 \\ \mu\notin\{\nu_1,\nu_2\}}}^{d+L} \langle v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))), v_\mu(\mathcal{I}(\mu,(i_1,\dots,i_\ell)))\rangle \right)}_{=:V_{[\nu_1,\nu_2]}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2},i_1,\dots,i_{\ell_1-1},i_{\ell_1+1},\dots,i_{\ell_2})} \end{aligned}$$

and

$$\langle v,a\rangle = \sum_{j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}=1}^{r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2}} \sum_{i_1,\dots,i_{k_1}=1}^{\tilde r_1,\dots,\tilde r_{k_1}} \langle v_{\nu_1,\nu_2}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}), a_{\nu_1,\nu_2}(i_1,\dots,i_{k_1})\rangle$$
$$\cdot \underbrace{\left(\sum_{j_{\ell_2+1},\dots,j_\ell,j_{\ell_1}=1}^{r_{\ell_2+1},\dots,r_\ell,1} \sum_{i_{k_1+1},\dots,i_k=1}^{\tilde r_{k_1+1},\dots,\tilde r_k} \prod_{\substack{\mu=1\\ \mu\notin\{\nu_1,\nu_2\}}}^{d+L} \langle v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))), a_\mu(\mathcal{I}_a(\mu,(i_1,\dots,i_k)))\rangle \right)}_{=:A_{[\nu_1,\nu_2]}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2},i_1,\dots,i_{k_1})}.$$

In contrary to the ALS algorithm of the previous subsection, we now differentiate with respect to $v_{\nu_1,\nu_2}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots j_{\ell_2})$ for fixed $(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}) \in \bigtimes_{\substack{\mu=1\\ \mu\neq\ell_1}}^{\ell_2} \{1,\dots,r_\mu\}$ which leads us to

$$\frac{\partial(\frac12\langle v,v\rangle - \langle v,a\rangle)}{\partial v_{\nu_1,\nu_2}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2})} = \sum_{i_1,\dots,i_{\ell_1-1},i_{\ell_1+1},\dots,i_{\ell_2}=1}^{r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2}} v_{\nu_1,\nu_2}(i_1,\dots,i_{\ell_1-1},i_{\ell_1+1},\dots,i_{\ell_2})$$
$$\cdot V_{[\nu_1,\nu_2]}(i_1,\dots,i_{\ell_1-1},i_{\ell_1+1},\dots,i_{\ell_2},j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2})$$
$$- \sum_{i_1,\dots,i_{k_1}=1}^{\tilde r_1,\dots,\tilde r_{k_1}} a_{\nu_1,\nu_2}(i_1,\dots,i_{k_1})$$
$$\cdot A_{[\nu_1,\nu_2]}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2},i_1,\dots,i_{k_1}).$$

So by defining

$$\mathbf{A}_{[\nu_1,\nu_2]} := \Big(\big(A_{[\nu_1,\nu_2]}(\dots)\big)_{(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}),(i_1,\dots,i_{k_1})} \Big)$$
$$\in \mathbb{K}^{\prod_{\mu=1,\mu\neq\ell_1}^{\ell_2} r_\mu \times \prod_{\mu=1}^{k_1}\tilde r_\mu}$$
$$\mathbf{a}_{\nu 1,\nu 2} := \begin{pmatrix} a_{\nu_1,\nu_2}(1,\dots,1)\\ \vdots \\ a_{\nu_1,\nu_2}(\tilde r_1,\dots,\tilde r_{k_1})\end{pmatrix} \in (\mathcal{V}_{\nu_1}\otimes\mathcal{V}_{\nu_2})^{\prod_{\mu=1}^{k_1}\tilde r_\mu}$$
$$\mathbf{V}_{[\nu_1,\nu_2]} := \Big(\big(V_{[\nu_1,\nu_2]}(\dots)\big)_{(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}),(i_1,\dots,i_{\ell_1-1},i_{\ell_1+1},\dots,i_{\ell_2})} \Big)$$
$$\in \mathbb{K}^{\prod_{\mu=1,\mu\neq\ell_1}^{\ell_2} r_\mu \times \prod_{\mu=1,\mu\neq\ell_1}^{\ell_2} r_\mu}$$
$$\mathbf{v}_{\nu_1,\nu_2} := \begin{pmatrix} v_{\nu_1,\nu_2}(1,\dots,1)\\ \vdots \\ v_{\nu_1,\nu_2}(r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2})\end{pmatrix} \in (\mathcal{V}_{\nu_1}\otimes\mathcal{V}_{\nu_2})^{\prod_{\mu=1,\mu\neq\ell_1}^{\ell_2} r_\mu}$$

we consequently have to solve

$$(\mathbf{A}_{[\nu_1,\nu_2]}\otimes Id_{\mathcal{V}_{\nu_1}\otimes\mathcal{V}_{\nu_2}})\mathbf{a}_{\nu_1,\nu_2} = (\mathbf{V}_{[\nu_1,\nu_2]}\otimes Id_{\mathcal{V}_{\nu_1}\otimes\mathcal{V}_{\nu_1}})\mathbf{v}_{\nu_1,\nu_2} \tag{4.13}$$

which is equivalent to Equation (4.9) in terms of the structure and in terms of the necessary computation of $\mathbf{V}^{-1}_{[\nu_1,\nu_2]}$ (see [1, Remark 7.1] for the treatment of non-regular matrices). The main difference to the ALS algorithm is that by solving Equation (4.13), we obtain $v_{\nu_1,\nu_2}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2})$ for all $(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}) \in \bigtimes_{\mu=1,\mu\neq\ell_1}^{\ell_2} \{1,\dots,r_\mu\}$. Each $v_{\nu_1,\nu_2}(\dots)$ however, is an element of the tensor space $\mathcal{V}_{\nu_1} \otimes \mathcal{V}_{\nu_2}$ such that we have to compute a decomposition into separate vector spaces. In general, $\mathbf{v}_{\nu_1,\nu_2}$ is a matrix, where the entries are in $\mathcal{V}_{\nu_1} \otimes \mathcal{V}_{\nu_2}$. So we decompose this matrix into a product of matrices X and Y^T such that the columns of X are elements of $\mathcal{V}_{\nu_1}$ and the columns of Y are elements of $\mathcal{V}_{\nu_2}$, where the number of columns of X and Y are equal to the matrix rank of $\mathbf{v}_{\nu_1,\nu_2}$. The result are the individual components v_{ν_1} and v_{ν_2}. In terms of matrix decompositions, we decompose

$$\Big(\big(v_{\nu_1,\nu_2}(j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2})_{i_{\nu_1},i_{\nu_2}}\big)_{(j_1,\dots,j_{\ell_1-1},i_{\nu_1}),(j_{\ell_1+1},\dots,j_{\ell_2},i_{\nu_2})}\Big) \tag{4.14}$$

into a minimal sum (w.r.t. the number of addends) of elementary order two tensors. If we allow the matrix (4.14) to be decomposed only approximatively (see Equation (4.15) and in the **Error estimate** part of this subsection), we may be able to reduce the decomposition rank for the price of accuracy. The main advantage in comparison to the ALS algorithm is the possibility of a rank adjustment.

The formal description of the DMRG algorithm, analogous to [1, Algorithm 3] is as in Algorithm 2. The error condition is usually chosen similarly to the one of the ALS algorithm (see Algorithm 1).
By finding $\bar{\mathbf{v}}_{\nu_1}$ and $\bar{\mathbf{v}}_{\nu_2}$ only approximatively, i.e. only requiring

$$\left\| reshape(\bar{\mathbf{v}}_{\nu_1,\nu_2}) - \sum_{j_{\ell_1}=1}^{\bar{r}} \bar{\mathbf{v}}_{\nu_1}(j_{\ell_1}) \otimes \bar{\mathbf{v}}_{\nu_2}(j_{\ell_1}) \right\| < \epsilon \tag{4.15}$$

for some $\epsilon > 0$ in the Frobenius norm, we introduce an error to the tensor representation. We will discuss the influence of this approximation subsequently.

Error estimate

At first, we want to specify the norm properties that we are using more precisely:

Definition 4.1.3 (Crossnorm, [5, Definition 4.31]). *Let $\|\cdot\|_V$ be a norm in V and $\|\cdot\|_W$ be a norm in W, then a norm $\|\cdot\|$ is called a* crossnorm *on $V \otimes W$ if*

$$\|v \otimes w\| = \|v\|_V \cdot \|w\|_W$$

for all $v \in V, w \in W$.

Taking Equation (4.15) as a basis, we assume we have

$$\hat{\mathbf{v}}_{\nu_1} : \mathbb{N} \to \mathcal{V}_{\nu_1}^{\prod_{i=1}^{\ell_1-1} r_i}$$

Algorithm 2 DMRG algorithm for general tensor networks

1: let a and v be tensor network representations with with representation rank $(r_1, \ldots, r_\ell) \in \mathbb{N}^\ell$ of same order in the same tensor space with equal underlying vector spaces just as defined before where v is the initial guess of a w.r.t. $\|a - v\|$ and (V, E) is the tensor network graph that defines v; let the definitions and premises of this subsection are also be valid
2: **while** error condition not fulfilled **do**
3: **for** $\{\nu_1, \nu_2\} \in E$ **do**
4: find $\bar{\mathbf{v}}_{\nu_1,\nu_2}$ such that

$$(\mathbf{A}_{[\nu_1,\nu_2]} \otimes Id_{\mathcal{V}_{\nu_1} \otimes \mathcal{V}_{\nu_2}}) \mathbf{a}_{\nu_1,\nu_2} = (\mathbf{V}_{[\nu_1,\nu_2]} \otimes Id_{\mathcal{V}_{\nu_1} \otimes \mathcal{V}_{\nu_2}}) \bar{\mathbf{v}}_{\nu_1,\nu_2}$$

as of Equation (4.13)
5: reshape $\bar{\mathbf{v}}_{\nu_1,\nu_2}$ as of (4.14) and decompose this reshaped matrix, i.e.

$$reshape(\bar{\mathbf{v}}_{\nu_1,\nu_2}) = \sum_{j_{\ell_1}=1}^{\bar{r}} \bar{\mathbf{v}}_{\nu_1}(j_{\ell_1}) \otimes \bar{\mathbf{v}}_{\nu_2}(j_{\ell_1})$$

where

$$\bar{v}_{\nu_1}(j_1, \ldots, j_{\ell_1})_{i_{\nu_1}} := \bar{\mathbf{v}}_{\nu_1}(j_{\ell_1})_{(j_1,\ldots,j_{\ell_1-1},i_{\nu_1})}$$
$$\bar{v}_{\nu_2}(j_{\ell_1}, \ldots, j_{\ell_2})_{i_{\nu_2}} := \bar{\mathbf{v}}_{\nu_2}(j_{\ell_1})_{(j_{\ell_1+1},\ldots,j_{\ell_2},i_{\nu_2})}$$

6: $v_{\nu_1} \mapsto \bar{v}_{\nu_1}$
7: $v_{\nu_2} \mapsto \bar{v}_{\nu_2}$
8: $r_{\ell_1} \mapsto \bar{r}$
9: **end for**
10: **end while**

and

$$\hat{\mathbf{v}}_{\nu_2} : \mathbb{N} \to \mathcal{V}_{\nu_2}^{\prod_{i=\ell_1+1}^{\ell_2} r_i}$$

and with $\|\cdot\|$ being crossnorms on the corresponding vector spaces, such that

$$\left\| \sum_{j_{\ell_1}=1}^{\hat{r}} \hat{\mathbf{v}}_{\nu_1}(j_{\ell_1}) \otimes \hat{\mathbf{v}}_{\nu_2}(j_{\ell_1}) - \sum_{j_{\ell_1}=1}^{\bar{r}} \bar{\mathbf{v}}_{\nu_1}(j_{\ell_1}) \otimes \bar{\mathbf{v}}_{\nu_2}(j_{\ell_1}) \right\| < \epsilon \tag{4.16}$$

for $\hat{r} < \bar{r}$. Together with

$$\begin{aligned}
\hat{v}_{\nu_1}(j_1, \ldots, j_{\ell_1})_{i_{\nu_1}} &:= \hat{\mathbf{v}}_{\nu_1}(j_{\ell_1})_{(j_1,\ldots,j_{\ell_1-1},i_{\nu_1})}, \\
\hat{v}_{\nu_2}(j_{\ell_1}, \ldots, j_{\ell_2})_{i_{\nu_2}} &:= \hat{\mathbf{v}}_{\nu_2}(j_{\ell_1})_{(j_{\ell_1+1},\ldots,j_{\ell_2},i_{\nu_2})}, \\
\bar{v}_{\cong} &:= \sum_{j_1,\ldots,j_{\ell_2}=1}^{r_1,\ldots,r_{\ell_1-1},\bar{r},r_{\ell_1+1},\ldots,r_{\ell_2}} \bar{v}_{\nu_1}(j_1, \ldots, j_{\ell_1}) \otimes \bar{v}_{\nu_2}(j_{\ell_1}, \ldots, j_{\ell_2}) \\
&\qquad \otimes \left(\sum_{j_{\ell_2+1},\ldots,j_\ell=1}^{r_{\ell_2+1},\ldots,r_\ell} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{\nu_1,\nu_2\}}}^{d+L} v_\mu(\mathcal{I}(\mu, (j_1, \ldots, j_\ell))) \right)
\end{aligned}$$

and

$$\begin{aligned}
\hat{v}_{\cong} &:= \sum_{j_1\ldots,j_{\ell_2}=1}^{r_1,\ldots,r_{\ell_1-1},\hat{r},r_{\ell_1+1},\ldots,r_{\ell_2}} \hat{v}_{\nu_1}(j_1, \ldots, j_{\ell_1}) \otimes \hat{v}_{\nu_2}(j_{\ell_1}, \ldots, j_{\ell_2}) \\
&\qquad \otimes \left(\sum_{j_{\ell_2+1},\ldots,j_\ell=1}^{r_{\ell_2+1},\ldots,r_\ell} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{\nu_1,\nu_2\}}}^{d+L} v_\mu(\mathcal{I}(\mu, (j_1, \ldots, j_\ell))) \right),
\end{aligned}$$

we end up with

$$\begin{aligned}
\|\bar{v}_{\cong} - \hat{v}_{\cong}\| = \Bigg\| \sum_{j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}=1}^{r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2}} \Bigg(& \sum_{j_{\ell_1}=1}^{\bar{r}} \bar{v}_{\nu_1}(j_1,\dots,j_{\ell_1}) \otimes \bar{v}_{\nu_2}(j_{\ell_1},\dots,j_{\ell_2}) \\
& - \sum_{j_{\ell_1}=1}^{\hat{r}} \hat{v}_{\nu_1}(j_1,\dots,j_{\ell_1}) \otimes \hat{v}_{\nu_2}(j_{\ell_1},\dots,j_{\ell_2}) \Bigg) \\
& \otimes \Bigg(\sum_{j_{\ell_2+1},\dots,j_{\ell}=1}^{r_{\ell_2+1},\dots,r_{\ell}} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{\nu_1,\nu_2\}}}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_{\ell_1-1},0,j_{\ell_1+1},\dots,j_\ell))) \Bigg) \Bigg\| \\
\leqslant \sum_{j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}=1}^{r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2}} & \Bigg\| \sum_{j_{\ell_1}=1}^{\bar{r}} \bar{v}_{\nu_1}(j_1,\dots,j_{\ell_1}) \otimes \bar{v}_{\nu_2}(j_{\ell_1},\dots,j_{\ell_2}) \\
& - \sum_{j_{\ell_1}=1}^{\hat{r}} \hat{v}_{\nu_1}(j_1,\dots,j_{\ell_1}) \otimes \hat{v}_{\nu_2}(j_{\ell_1},\dots,j_{\ell_2}) \Bigg\| \\
& \cdot \Bigg\| \sum_{j_{\ell_2+1},\dots,j_{\ell}=1}^{r_{\ell_2+1},\dots,r_{\ell}} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{\nu_1,\nu_2\}}}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_{\ell_1-1},0,j_{\ell_1+1},\dots,j_\ell))) \Bigg\| \\
\leqslant \Bigg(\sum_{j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}=1}^{r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2}} & \Bigg\| \sum_{j_{\ell_1}=1}^{\bar{r}} \bar{v}_{\nu_1}(j_1,\dots,j_{\ell_1}) \otimes \bar{v}_{\nu_2}(j_{\ell_1},\dots,j_{\ell_2}) \\
& \underbrace{- \sum_{j_{\ell_1}=1}^{\hat{r}} \hat{v}_{\nu_1}(j_1,\dots,j_{\ell_1}) \otimes \hat{v}_{\nu_2}(j_{\ell_1},\dots,j_{\ell_2}) \Bigg\|^2 \Bigg)^{\frac{1}{2}}}_{= \left\| \sum_{j_{\ell_1}=1}^{\bar{r}} \bar{\mathbf{v}}_{\nu_1}(j_{\ell_1}) \otimes \bar{\mathbf{v}}_{\nu_2}(j_{\ell_1}) - \sum_{j_{\ell_1}=1}^{\hat{r}} \hat{\mathbf{v}}_{\nu_1}(j_{\ell_1}) \otimes \hat{\mathbf{v}}_{\nu_2}(j_{\ell_1}) \right\| \overset{(4.16)}{<} \epsilon} \\
& \cdot \Bigg(\sum_{j_1,\dots,j_{\ell_1-1},j_{\ell_1+1},\dots,j_{\ell_2}=1}^{r_1,\dots,r_{\ell_1-1},r_{\ell_1+1},\dots,r_{\ell_2}} \Bigg\| \sum_{j_{\ell_2+1},\dots,j_{\ell}=1}^{r_{\ell_2+1},\dots,r_{\ell}} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{\nu_1,\nu_2\}}}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_{\ell_1-1},0,j_{\ell_1+1},\dots,j_\ell))) \Bigg\|^2 \Bigg)^{\frac{1}{2}}.
\end{aligned}$$

As a consequence, we may conclude with

Remark 4.1.4. *So we are able to control the error that we introduce within one DMRG-step w.r.t. the current approximation v. However, only the first factor of the product (which we can control by the error that we may introduce by the SVD) is bounded. The second factor can be arbitrarily large. Therefore it is only recommended to use this scheme for formats that are stable in their representation. For instable formats, one has to add additional constraints about the norm of the second factor (i.e. its boundedness by some constant).*

4.2 Initial guess

In all approximation cases involving iterative methods, it is crucial to find and use a proper initial guess, see for instance [5, Subsubsection 9.5.2.1]. For linear iteration schemes, a good initial guess helps to decrease the number of iteration steps that are needed to reach a certain accuracy, but it does not change the approximation result. For non-linear iteration schemes, the approximation result (i.e. the limit) may depend on the initial guess. We can try to use a gradient method together with the successive initial rank-one approximation to find a reasonable initial guess. This however does not overcome the problems for the general successive guess method for arbitrary tensor network formats.

Getting a good initial guess by successively increasing the tensor representation by a rank-one tensor is a good idea for tensor representations in a format with few edges, such as the r-term format. In even slightly more complex formats however, there are additional restrictions to the new rank-one terms such that it becomes almost unfeasible.

So an important question is how to construct an initial guess that is reasonable with respect to the approximation algorithm independently of the structure that the original tensor is given in. In [4, Section 5.4] is the description of an accelerated conjugated gradient method for the r-term format that acts roughly like the following to obtain a lowrank approximation $\hat{v}$, which is a tensor representation, of a given tensor a:

1. find the best rank 1 approximation $\hat{v}$ of a
2. find the best rank 1 approximation of $\hat{v} - a$ and add this result to $\hat{v}$ such that the rank of $\hat{v}$ increase by 1
3. minimize $\hat{v} \mapsto \|\hat{v} - a\|$ for fixed a
4. goto 2. until $\hat{v}$ has the maximum representation rank or $\|\hat{v} - a\|$ has the desired accuracy

The algorithm in its current form is formulated for the r-term format. We want to generalize the approach to be usable for arbitrary tensor network representations.
In fact basically the only modification that we have to make, is to take care of the rank increment procedure.

4.2.1 Algorithm

The general setting is, as before, that we have finite dimensional $\mathbb{K}$-vector spaces $\mathcal{V}_1, \dots, \mathcal{V}_d$ with $d \in \mathbb{N}$ being the order of the tensor space $\mathcal{V} := \bigotimes_{\mu=1}^{d} \mathcal{V}_\mu$.

In this section, our main goal will be to approximate a given tensor $a \in \mathcal{V}$ by a tensor representation $\hat{v}$ in a specific format that represents tensor $v \in \mathcal{V}$. So we are trying to minimize

$$\|v - a\|$$

in some norm in $\mathcal{V}$ (see Section 4.0). Subsequently, we will identify v with $\hat{v}$ such that we may write

$$\|v - a\| = \|\hat{v} - a\|$$

For the tensor representation $\hat{v}$ of tensor v, we are limited to one are limited to one tensor format graph. That is, the structure of the underlying graph may not be changed during the execution of the algorithm (for instance, the hierarchical format is based on multiple tensor format graphs). To compute an approximation, one could fix the rank of v's representation $\hat{v}$ and perform standard optimization algorithms such as non-linear block Gauss-Seidel (see Section 4.1 and [1, Section 6]) or CG methods, where the initial value may be chosen as of [1, Subsection 7.3]. In order to faster obtain a tensor representation that matches the desired error bound, we describe an algorithm that successively increases the representation rank of the tensor representation of v, starting from a rank-one tensor.

The algorithm that we will explain here, is a generalization of [4, Algorithm 5.4.1] for arbitrary tensor networks where the main difference is, that we have to handle a tuple of ranks instead of just one single rank. Increasing one of the components of the rank-tuple by one, results in adding in general more than one term to the tensor.

To clarify the difference to the existing approach, we state a small example for the r-term format. Let a be an order d tensor with unspecified tensor representation. One may think of a as a full tensor (in practice however, a should be represented more efficiently). Following the sketch from above, we perform

1. $\hat{v} := \underset{b:=\bigotimes_{\mu=1}^{d} b_\mu, b_\mu \in \mathcal{V}_\mu}{\operatorname{argmin}} \|b - a\|$

2. $\tilde{v} := \underset{c:=\bigotimes_{\mu=1}^{d} c_\mu, c_\mu \in \mathcal{V}_\mu}{\operatorname{argmin}} \|a - \hat{v} - c\|$ and afterwards $\hat{v} \mapsto \hat{v} + \tilde{v}$ such that the representation rank of $\hat{v}$ increases by one

3. minimize $\|\hat{v} - a\|$ with respect to $\hat{v}$ where representation rank of $\hat{v}$ is fixed

4. goto 2. until $\hat{v}$ has the maximum rank or $\|\hat{v} - a\|$ has the desired accuracy

This procedure works very well as the rank increment requires only adding an elementary tensor $b_1 \otimes \ldots \otimes b_d$. On the contrary, we have the procedure for a tensor representation in the TC format (see Definition 2.1.11):
The crucial part is the rank increment since the representation rank is a tuple of integers instead of just one integer. By looking at a tensor of order 3, represented in the TC format of order 3 with representation rank $(r_1, r_2, r_3) = (2, 2, 2)$ the important difference is immediately visible. Increasing r_1 by one such that the representation rank is going

to be $(3,2,2)$ leads us to

$$\begin{aligned}\sum_{j_1,j_2,j_3=1}^{3,2,2} v_1(j_1,j_3)\otimes v_2(j_1,j_2)\otimes v_3(j_2,j_3) = \\ \sum_{j_1,j_2,j_3=1}^{2,2,2} v_1(j_1,j_3)\otimes v_2(j_1,j_2)\otimes v_3(j_2,j_3) \\ + v_1(3,1)\otimes v_2(3,1)\otimes v_3(1,1) \\ + v_1(3,2)\otimes v_2(3,1)\otimes v_3(1,2) \\ + v_1(3,1)\otimes v_2(3,2)\otimes v_3(2,1) \\ + v_1(3,2)\otimes v_2(3,2)\otimes v_3(2,2)\end{aligned} \tag{4.17}$$

such that we have to add four terms if we want to increase r_1 to 3 instead of just one additional term for increasing the rank of a r-term tensor by one. Different representation ranks lead to a different number of terms that we have to add. For more complicated structures, like the PEPS format (see Definition 2.1.14 and [29] for applications), the number of terms also increases.
However, generalizing this scheme to arbitrary tensor networks is straightforward and omitted here (with the exception of Algorithm 3) in order to not disturb the reader from the idea behind this approach. A general rule for the number of terms that have to be added is a straightforward computation that result in

Remark 4.2.1. *Let v be a tensor representation of order $(d+L)$ that is based on the tensor format graph (V,E) with corresponding incidence* map $\mathcal{I}$ – with corresponding incidence map $\mathbb{I}$ – and has representation rank $(r_1,\dots,r_\ell)\in\mathbb{N}^\ell$ where $\ell := \#E$. Let further $i\in\{1,\dots,\ell\}$ where we can assume without loss of generality that $r_i > 1$. Then we have*

$$\begin{aligned} v &\cong \sum_{j_1,\dots,j_\ell=1}^{r_1,\dots,r_\ell} \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \\ &= \sum_{j_1,\dots,j_\ell=1}^{r_1\dots,r_{i-1},r_i-1,r_{i+1},\dots,r_\ell} \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \\ &\quad + \underbrace{\sum_{j_1,\dots,j_{i-1},j_{i+1},\dots,j_\ell=1}^{r_1,\dots,r_{i-1},r_{i+1},\dots,r_\ell} \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_{i-1},r_i,j_{i+1},\dots,j_\ell)))}_{=:S}. \end{aligned}$$

The number of addends in the second sum S that depend on the i-th summation is given by

$$\prod_{k\in I_i} r_k$$

where

$$I_i = \left\{\mathbb{I}(\mu) \mid \mu \in \mathbb{I}^{-1}(i)\right\} \setminus \{i\}.$$

Generalizing the algorithm for finding an initial guess can be summarized as Algorithm 3.

Algorithm 3 Successive initial guess algorithm

1: Let $T = (V, E)$ be an undirected, connected multi-graph with $\ell := \#E$ and $\mathbf{r} := (r_1, \ldots, r_\ell) \in \mathbb{N}^\ell$ such that T and $\mathbf{r}$ define a tensor representation $\hat{v}$ with representation rank $\mathbf{r}$.
Set $\hat{v}$ to the best rank 1 approximation of order d tensor a (whose representation does not matter), choose $\epsilon > 0$, the maximal representation rank $(\bar{r}_1, \ldots, \bar{r}_\ell) \in \mathbb{N}^\ell$ and initial representation rank $(r_1, \ldots, r_\ell) := (1, \ldots, 1)$
2: **while** $\|\hat{v} - a\| > \epsilon$ and $(r_1, \ldots, r_\ell) \neq (\bar{r}_1, \ldots, \bar{r}_\ell)$ **do**
3: Select $k \in \{i \in \{1, \ldots, \ell\} \mid r_i < \bar{r}_i\}$
4: Find approximation $\tilde{v}$ of $\hat{v} - a$ such that T specifies $\hat{v} + \tilde{v}$ with representation rank $(r_1, \ldots, r_{k-1}, r_k + 1, r_{k+1}, r_\ell)$
5: $\hat{v} \mapsto \hat{v} + \tilde{v}, r_k \mapsto r_k + 1$ such that the representation rank of $\hat{v}$ changes
6: Minimize $\|\hat{v} - a\|$ w.r.t. $\hat{v}$ where the representation rank of $\hat{v}$ is fixed
7: **end while**

Except for the rank increment of this algorithm's line 4, the algorithm very similar to [4, Algorithm 5.4.1] such that we will focus on the description of the rank increasing procedure. Increasing one element of the representation rank by one is done by adding several terms to the tensor representation $\hat{v}$. It is immediately clear, that in general not all components of the new terms have to be unknown. In Equation (4.17) for instance, there is no new value needed for v_3 as all $v_3(i, j)$ with $i, j \in \{1, 2\}$ are already determined by the original tensor representation v with the representation rank $(2, 2, 2)$.
Consequently, we have to come up with a way to determine the unknown components. In contrary to [4, Algorithm 5.4.1, line 4], we cannot do this by rank 1 approximations.

We now state the rank increment algorithm for the TC format as an example. This algorithm explains how to increase the kth rank component with $k \in \{1, \ldots, d\}$ of a TC tensor representation $\hat{v}$ which represents the tensor v with representation rank $(r_1, \ldots, r_d)$ while approximating the tensor a. In general, we have to perform

Algorithm 4 Increase of the kth rank component for the TC format

Let $\hat{v}$ be a TC tensor representation of order d with representation rank $(r_1, \dots, r_d) \in \mathbb{N}^d$ as of Definition 2.1.11 which represents a tensor $v \in \mathcal{V}$ and $a \in \mathcal{V}$ be some order d tensor. Without loss of generality, we choose $k \in \{2, \dots, d-1\}$.

1: Set $j_k := r_k + 1$

2: Find approximation

$$\tilde{v} := \sum_{j_1,\dots,j_{k-1},j_{k+1},\dots,j_d=1}^{r_1,\dots,r_{k-1},r_{k+1},\dots,r_d} v_1(j_1, j_d) \otimes \bigotimes_{\mu=2}^{k-1} v_\mu(j_{\mu-1}, j_\mu) \otimes v_k(j_{k-1}, j_k) \otimes v_{k+1}(j_k, j_{k+1}) \otimes \bigotimes_{\mu=k+2}^{d} v_\mu(j_{\mu-1}, j_\mu)$$

of $v - a$ with fixed $v_1, \dots v_{k-1}, v_{k+2}, \dots, v_d$

3: $\hat{v} \mapsto \hat{v} + \tilde{v}$

Remark 4.2.2. *After obtaining the new representation $\hat{v}$, one could perform an additional optimization iteration for minimizing $\|v - a\|$ with ALS or CG as in line 6 of Algorithm 3. In practice, this is highly recommended and usually consumes only very few iterations (see Subsection 4.2.4). This step however, is not mandatory.*
Note that if $\hat{v}$ changes, the values of tensor v also change in general.

In the subsequent Section 4.2.2, we will illustrate the whole process in more details. Since in the case of a TC tensor, each summation is associated to exactly two directions of the tensor, the inner approximation in line 2 of Algorithm 4 is a problem of two dimensions (i.e. v_k and v_{k+1} have to be determined for some indices) such that we can compute it very fast.

4.2.2 Example

We want to give the reader a detailed example of the application of Algorithm 3 together with Algorithm 4 such that the main idea and the problems that arise from this approach become clearly visible.

The goal is to find an approximated representation $\hat{v}$ (which represents the tensor v) in the TC format with representation rank $(r_1, r_2, r_3, r_4) \in \mathbb{N}^4$ of tensor a (represented in an arbitrary format) of order $d = 4$. The maximal representation rank of v is $(\bar{r}_1, \bar{r}_2, \bar{r}_3, \bar{r}_4) \in \mathbb{N}^4$ and we want to find the minimal distance $\|\hat{v} - a\|$ for this representation rank. The initial representation rank of $\hat{v}$ is $(1, 1, 1, 1)$ such that we start with an elementary tensor.

As the interesting part is the rank increment, we will demonstrate this at first by

increasing r_2 by one. Similar to Equation (4.17), we define

$$\begin{aligned}\hat{v} &:= \sum_{j_1,j_2,j_3,j_4=1}^{r_1,r_2,r_3,r_4} v_1(j_1,j_4) \otimes v_2(j_1,j_2) \otimes v_3(j_2,j_3) \otimes v_4(j_3,j_4) \\ &= \sum_{j_1,j_2,j_3,j_4=1}^{r_1,r_2+1,r_3,r_4} v_1(j_1,j_4) \otimes v_2(j_1,j_2) \otimes v_3(j_1,j_3) \otimes v_4(j_4,j_4) \\ &\quad \underbrace{- \sum_{j_1,j_3,j_4=1}^{r_1,r_3,r_4} v_1(j_1,j_4) \otimes v_2(j_1,r_2+1) \otimes v_3(r_2+1,j_3) \otimes v_4(j_3,j_4)}_{\tilde{v}}\end{aligned}$$

such that we are interested in finding $\tilde{v}$ (which is a tenor train representation). The values of $v_1(j_1,j_4)$ and $v_4(j_3,j_4)$ are known for all $(j_1,\dots,j_4) \in \bigtimes_{\mu=1}^{4} \{1,\dots,r_\mu\}$ so they have to be left unchanged. On the other side, $v_2(j_1,r_2+1)$ and $v_3(r_2+1,j_3)$ are to unknown for all $(j_1,j_3) \in \{1,\dots,r_1\} \times \{1,\dots,r_3\}$ such that values are to be determined. Therefore, we need to find the tensor representation $\tilde{v}$ that minimizes

$$\|\hat{v} + \tilde{v} - a\|$$

under the constraints for the known values. This results in the complete algorithm to find an initial guess as follows:

Without loss of generality, $\bar{r}_4 \geqslant \bar{r}_i \ \forall i \in \{1,2,3\}$.

1st step

$$r := (1,1,1,1)$$

- Find best rank 1 approximation

  $$v_1(1,1) \otimes v_2(1,1) \otimes v_3(1,1) \otimes v_4(1,1)$$

  of a, i.e.

  $$\hat{v} := v_1(1,1) \otimes v_2(1,1) \otimes v_3(1,1) \otimes v_4(1,1)$$

2nd step

$$r := (2,1,1,1)$$

- Find best rank 1 approximation

  $$v_1(2,1) \otimes v_2(2,1) \otimes v_3(1,1) \otimes v_4(1,1)$$

  of $\hat{v} - a$ with $v_3(1,1)$ and $v_4(1,1)$ fixed since both $v_3(1,1)$ and $v_4(1,1)$ were already determined in the first step

$\Rightarrow$

$$\hat{v} := \sum_{j_1=1}^{2} v_1(j_1,1) \otimes v_2(j_1,1) \otimes v_3(1,1) \otimes v_4(1,1)$$

3rd step

$$r := (2,2,1,1)$$

- Find approximation

$$\sum_{j_1=1}^{2} v_1(j_1,1) \otimes v_2(j_1,2) \otimes v_3(2,1) \otimes v_4(1,1)$$

of $\hat{v} - a$ with fixed $v_1(j_1,1)$ for $j_1 \in \{1,2\}$ and $v_4(1,1)$ fixed

$\Rightarrow$

$$\hat{v} := \sum_{j_1,j_2=1}^{2,2} v_1(j_1,1) \otimes v_2(j_1,j_2) \otimes v_3(j_2,1) \otimes v_4(1,1)$$

4th step

$$r := (2,2,2,1)$$

- Find approximation

$$\sum_{j_1,j_2=1}^{2,2} v_1(j_1,1) \otimes v_2(j_1,j_2) \otimes v_3(j_2,2) \otimes v_4(2,1)$$

of $v - a$ with $v_1(j_1,1)$ and $v_2(j_1,j_2)$ for $j_1, j_2 \in \{1,2\}$ fixed

$\Rightarrow$

$$\hat{v} := \sum_{j_1,j_2,j_3=1}^{2,2,2} v_1(j_1,1) \otimes v_2(j_1,j_2) \otimes v_3(j_2,j_3) \otimes v_4(j_3,1)$$

5th step

$$r := (2,2,2,2)$$

- Find approximation

$$\sum_{j_1,j_2,j_3=1}^{2,2,2} v_1(j_1,2) \otimes v_2(j_1,j_2) \otimes v_3(j_2,j_3) \otimes v_4(j_3,2)$$

of $v - a$ with $v_2(j_1,j_2)$ and $v_3(j_2,j_3)$ for $j_1, j_2, j_3 \in \{1,2\}$ fixed

$\vdots$

$(\bar{r}_1 + \bar{r}_2 + \bar{r}_3 + \bar{r}_4 - 3)$**th step**

$$r := (\bar{r}_1, \bar{r}_2, \bar{r}_3, \bar{r}_4)$$

- Find approximation

$$\sum_{j_1, j_2, j_3 = 1}^{\bar{r}_1, \bar{r}_2, \bar{r}_3} v_1(j_1, \bar{r}_4) \otimes v_2(j_1, j_2) \otimes v_3(j_2, j_3) \otimes v_3(j_3, \bar{r}_4)$$

  of $v - a$ with $v_2(j_1, j_2)$ and $v_3(j_2, j_3)$ for $j_1, \in \{1, \dots, \bar{r}_1\}, j_2 \in \{1, \dots, \bar{r}_2\}$ and $j_3 \in \{1, \dots, \bar{r}_3\}$ fixed

In principle, we are not bound to any rank increasing order such we also could increase r_1 until $\bar{r}_1$ before increasing the other representation rank components. We will address this problem in the next subsection.

The algorithm can be also used to obtain not *only* an initial guess, but also an approximation result if we choose ϵ sufficiently small (see Remark 4.2.2).

4.2.3 Problems

We named one major problem (uncertainty in the increment order) in the proposed procedure and now, we will explain it together with the consequences. Additionally, we basically have the same problem as for the ALS algorithm for single terms (see conclusion about the ALS algorithm for single terms at page 69).

At first, we are going to show that changing the order of the rank increments leads to a different number of iterations that is needed to reach a certain error bound. The following setup is used for the computations

- a is tensor, represented by order d TC tensor representation with representation rank $(r_1, \dots, r_d) = (10, \dots, 10)$ filled with pseudo-random values $\in [0, 1]$
- $\hat{v}$ is a tensor representation that represents the tensor v
- all vector space dimensions $\dim \mathcal{V}_1, \dots, \dim \mathcal{V}_d$ are equal to 10
- the maximal representation rank of TC tensor representation $\hat{v}$ is $(10, \dots, 10)$
- the initial representation rank of $\hat{v}$ is $(1, \dots, 1)$ and its values is computed via adaptive cross approximation (ACA) as described in [1, Subsection 7.3])
- the initial values for the unknown components of each step are set to $1/\sqrt{n_\mu}$ where μ is the direction of the unknown component; afterwards ALS is used to determine the unknown values
- after the initial guess has been determined, standard ALS is applied to obtain the approximation result (with error condition threshold of $1 \cdot 10^{-8}$ as of Algorithm 1)
- implementation: [38, test/Representation/TensorChainTest.cpp]

- Intel i3-3220T CPU

For the first option (option 1), we increase each representation rank component one after another by one, just as described in the previous subsection. For the second option (option 2) however, we increase the first representation rank component successively by one until $(10, 1, \ldots, 1)$ is the representation rank of v. Afterwards, we increase component two until $(10, 10, 1, \ldots, 1)$, etc.

d	$\|a - v\|/\|a\|$		CPU time	
	option 1	option 2	option 1	option 2
4	$6.56 \cdot 10^{-4}$	$5.84 \cdot 10^{-4}$	$16.65s$	$22.54s$
6	$5.40 \cdot 10^{-5}$	$1.37 \cdot 10^{-7}$	$15.73s$	$24.77s$
8	$5.64 \cdot 10^{-5}$	$1.16 \cdot 10^{-4}$	$41.67s$	$67.67s$
10	$1.85 \cdot 10^{-4}$	$2.40 \cdot 10^{-4}$	$66.47s$	$67.90s$

Table 4.1: Different rank increment orders

The results in Table 4.1 emphasizes the problem, that has been stated above, very well. Although the ALS error condition for the final ALS approximation is equal, the results differ with respect to the relative error and the CPU time.

As a result of the unknown *optimal* increment order, we cannot tell when we should stop increasing a rank component if the maximal or reasonable value is unknown. If we want to have balanced rank components, this is an easy task, as we simply have to use option 1, but as soon as non-balanced rank components can be considered, we have to use other indications for a *good* rank distribution.
More complicated formats are of course more complicated to handle but if there is some indication about the rank distribution (from the physical background for instance), it should be applied to the algorithms in order to obtain reasonable results.

4.2.4 Numerical experiments

Performing actual computations with this approach in this example show that the proposed algorithm may result in a benefit in terms of the CPU time that is needed to reach a certain accuracy. Despite the problems, that we have described in Subsection 4.2.3, for certain structured data, we are able to improve the time that is needed to get a reasonable approximation in comparison to the ACA/ALS method in [1, Subsection 7.3] in some constellations. For random data, the successive approach performs significantly worse in most of the tested cases.

All our numerical experiments are performed with the help of the implementation in [38]. We have the following general setup for the subsequent experiments:

- random tensor a of order d that is represented in the TC format with representation rank $(r_1, \ldots, r_d) = (10, \ldots, 10)$ and vector space dimension $\dim \mathcal{V}_\mu = 10 \ \forall \mu \in \{1, \ldots, d\}$

- we try to approximate a by a TC tensor representation $\hat{v}$ where we start from representation rank $(1, \ldots, 1)$ and successively increase each rank component by one until $(9, \ldots, 9)$ (option 1 of the previous subsection)
- use ACA for the rank 1 initial guess
- the initial values for the unknown components of each step are set to $1/\sqrt{n_\mu}$ where μ is the direction of the unknown component ($n_\mu = \dim \mathcal{V}_\mu$); afterwards ALS is used to determine the unknown values
- the relative error is computed using the successive initial guess with subsequent ALS (with error condition threshold of $1 \cdot 10^{-8}$ as of Algorithm 1)
- for comparison, the ACA/ALS method (where the initial guess is computed via ACA as of [1, Subsection 7.3]) will be used until the relative error from above is reached
- implementation: [38, test/Representation/TensorChainTest.cpp]
- Intel i3-3220T CPU

The results are shown in Table 4.2.

d	Rel. error	CPU time	CPU time ACA/ALS
4	$7.77 \cdot 10^{-4}$	$13.86s$	$0.20s$
6	$4.78 \cdot 10^{-4}$	$3.10s$	$70.59s$
8	$7.90 \cdot 10^{-4}$	$8.41s$	$0.092s$
10	$8.65 \cdot 10^{-4}$	$25.79s$	$0.12s$

Table 4.2: Comparison of successive approach/ALS and ACA/ALS for pseudo-random data

These were only tests with random function values to show that the algorithm does not perform very well for this worst case scenario. However, if it comes to structured data, the successive approach is more effective.

Like in [30], we want to approximate the function

$$f(x_1, \ldots, x_d) = \frac{1}{\sqrt{1 + x_1^2 + \ldots + x_d^2}} \tag{4.18}$$

in $[0, 1]^d$. The representation rank of the approximation result TC tensors is $(3, \ldots, 3)$. As before, we determine the relative error using the successive initial guess approach and try to reach this error using ALS with an initial guess that has been generated using ACA. The *orig. rank* in Table 4.3 denotes the rank of the tensor representation that we want to approximate. We generate the to be approximated TC tensor by using the method, that is described in [30] (we use the SVD version with an SVD approximation accuracy of $1 \cdot 10^{-5}$ for $d \in \{3, \ldots, 7\}$) and $1 \cdot 10^{-4}$ for $d \in \{8, 9\}$).

d	n_μ	Orig. rank	Rel. error	CPU time	CPU time ACA/ALS
3	129	$(3,3,15)$	$2.2408 \cdot 10^{-6}$	$0.05s$	$1.09s^{(*)}(2.2411 \cdot 10^{-6})$
4	33	$(3,3,15,19)$	$1.46 \cdot 10^{-5}$	$0.16s$	$0.07s$
6	10	$(3,3,15,20,25,29)$	$3.75 \cdot 10^{-5}$	$1.25s$	$8.85s^{(*)}(3.77 \cdot 10^{-5})$
7	10	$(3,3,15,21,29,89,30)$	$4.54 \cdot 10^{-5}$	$20.34s$	$33.09s^{(*)}(5.77 \cdot 10^{-5})$
8	10	$(3,3,14,19,23,27,35,30)$	$4.93 \cdot 10^{-5}$	$4.55s$	$17.97s^{(*)}(5.60 \cdot 10^{-5})$
9	7	$(3,3,13,20,24,27,30,44,21)$	$7.20 \cdot 10^{-2}$	$42.73s$	$22.22s^{(*)}(1.39 \cdot 10^{-1})$

Table 4.3: Comparison of successive approach/ALS and ACA/ALS for structured data

The results are shown in Table 4.3 where the $(*)$ means that the given approximation error was not achieved using ACA/ALS (the final relative error is stated in brackets) after at most 5000 ALS steps over all d directions. This indicates that there may be an advantage using this approach for certain data. Due to the problems that we mentioned before, it is not surprising that the general approach does not result in a significant benefit. If it comes to real data that has to be approximated with high accuracy however, the successive initial guess algorithm with a subsequent ALS seems to be worth trying.

Remark 4.2.3. *The examples of this subsection have been chosen to show that the algorithm works for certain data. There have been no non-regular matrices during the ALS iteration. The initial values for the to be determined components have been set artificially to* $1/\sqrt{n_\mu}$*; choosing other initial values will lead to different results. The reader is also reminded of the instability of the TC format.*

Chapter 5

Constructive algorithms

Creating arbitrary tensor network representations out of a given tensor (the tensor may be defined by a tensor representation or pointwise) is a very important task as this step allows us to actually perform computations and optimizations in the tensor network framework. We want to describe how to change the topology of tensor network representations and give an outlook on future work in this area.

5.1 Changing the representation topology

If there is the need for a small structural (i.e. local) change of a tensor network representation, it should be done using the existing structure. An algorithm will be introduced and explained that can perform local changes of the tensor network structure. We do not change the tensor itself, we only change the way it is represented by sums of elementary tensors.

We can also utilize this algorithm for performing complete conversions of tensor formats such as a conversion from PEPS to the hierarchical format.

The content of this section is already published in the author's report [39].

5.1.1 Direct conversion from TC to TT without approximation

Our first example will be the topology change from a ring structure (Tensor Chain or TC, see Definition 2.1.11) to a string structure (Tensor Train or TT, see Definition 2.1.8). The two topologies differ only in one edge and therefore they have a lot in common which we can use for our advantage.

In Subsection 2.2.13 we already described the exact *conversion* without any kind of approximation, simply by copying values. We still want to describe a method that is more general with the help of this example. The initial results indicate that the exact conversion with the method of this section produces ridiculously high ranks in comparison to the previously described method. It turns out however that we can use approximations for the conversion (see Table 5.2) such that the results are from the same quality (in terms of representation ranks and the error) as the direct exact conversion

of Subsection 2.2.13. Considering Remark 4.1.4 and the fact that the TC format is not closed, we have to enforce a certain limit to the factors of the error estimate of Subsection 5.1.5 if they exceed some border (for our simple experiments, we do not need to do this).

Let $d \in \mathbb{N}\backslash\{1,2\}, n_1, \ldots, n_d, r_1, \ldots, r_d \in \mathbb{N}$ and set the vector spaces of Definition 2.1.11 to be

$$\mathcal{V}_\mu = \mathbb{K}^{n_\mu} \qquad \forall \mu \in \{1, \ldots, d\}.$$

We define

$$v = \sum_{j_1,\ldots,j_d=1}^{r_1,\ldots,r_d} v_1(j_1, j_d) \otimes \bigotimes_{\mu=2}^{d} v_\mu(j_{\mu-1}, j_\mu) \in \bigotimes_{\mu=1}^{d} \mathcal{V}_\mu$$

with

$$\begin{aligned} v_1 &: \{1, \ldots, r_1\} \times \{1, \ldots, r_d\} \to \mathcal{V}_1 \\ v_i &: \{1, \ldots, r_{i-1}\} \times \{1, \ldots, r_i\} \to \mathcal{V}_i \qquad \text{for } i = 2, \ldots, d \end{aligned}$$

such that in terms of Definition 2.1.11, v is a tensor represented in the Tensor Chain format with representation rank $(r_1, \ldots, r_d)$, see Figure 5.1.

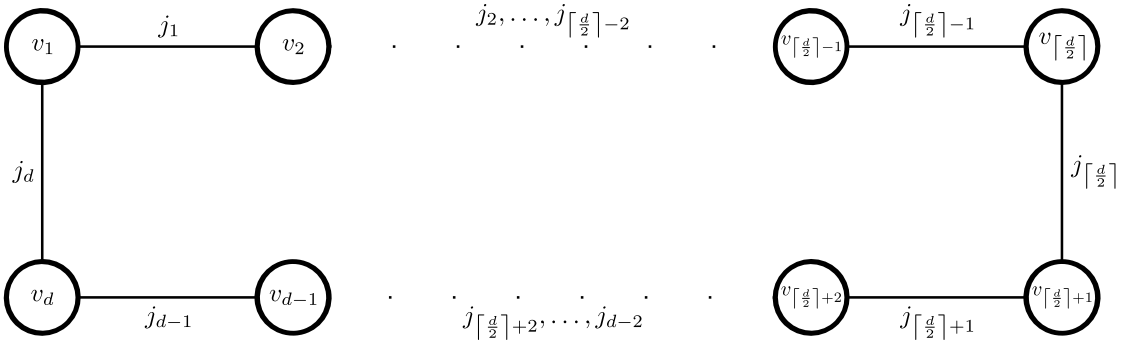

Figure 5.1: Tensor Chain of order d

This figure is in general the same as Figure 2.9 with the only difference of an emphasis on the *central* edge $j_{\lceil \frac{d}{2} \rceil}$. We want to change the tensors representation, i.e. our goal structure is defined by

$$\sum_{j_1,\ldots,j_{d-1}=1}^{\tilde{r}_1,\ldots,\tilde{r}_{d-1}} \tilde{v}_1(j_1) \otimes \bigotimes_{\mu=2}^{d-1} \tilde{v}_\mu(j_{\mu-1}, j_\mu) \otimes \tilde{v}_d(j_{d-1})$$

with

$$\begin{aligned} \tilde{v}_1 &: \{1, \ldots, \tilde{r}_1\} \to \mathcal{V}_1 \\ \tilde{v}_i &: \{1, \ldots, \tilde{r}_{i-1}\} \times \{1, \ldots, \tilde{r}_i\} \to \mathcal{V}_i \qquad \text{for } i = 2, \ldots, d-1 \\ \tilde{v}_d &: \{1, \ldots, \tilde{r}_{d-1}\} \to \mathcal{V}_d \end{aligned}$$

which is, in reference to Definition 2.1.8, a tensor represented in the *Tensor Train format* of order d with representation rank $(\tilde{r}_1, \ldots, \tilde{r}_{d-1}) \in \mathbb{N}^{d-1}$ and visualized in Figure 5.2 (where in comparison to Figure 2.7, the *central* edge is emphasized). As stated in Remark 2.1.13, a tensor in the TT format is also a tensor in the TC format (but not vice versa).

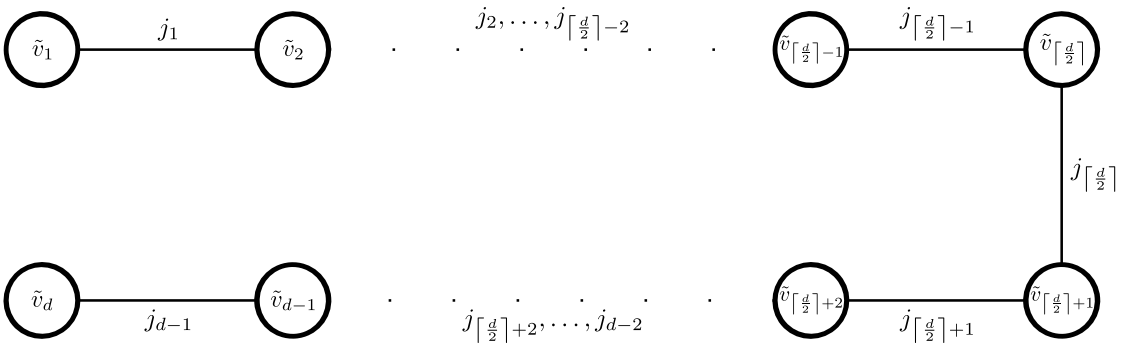

Figure 5.2: Tensor Train of order d

To be able to use the given ring structure of the Tensor Chain, we will convert it to the simplest possible order d tree, which is the Tensor Train. We successively *move* one particular edge of the ring further and further to the edge that is at the *center* of the ring without this specific *moved* edge. The following part visualizes this scheme.

In our description, we are using the singular value decomposition (SVD) to decompose a matrix. We could also utilize other decompositions, like the QR decomposition, but they have the same computational complexity as the SVD up to some constants. A main advantage of the SVD is, that it provides a best rank k approximation where $k \in \{1, \ldots, rank(\text{Matrix})\}$ for matrices which we want to use later in approximated results.

1st step

We define

$$v_{1,2}(j_d, j_2)_{i,j} := \sum_{j_1=1}^{r_1} \left(v_1(j_1, j_d) \otimes v_2(j_1, j_2)\right)_{i,j}$$

for all $j_d \in \{1, \ldots, r_d\}, j_1 \in \{1, \ldots, r_1\}$ and interpret $v_{1,2}$ as $n_1 \times n_2 \cdot r_d \cdot r_2$ matrix $\left((v_{1,2}(j_d, j_2)_{i,j})_{i,(j,j_d,j_2)}\right)$ of which we compute the SVD to obtain

$$\left((v_{1,2}(j_d, j_2)_{i,j})_{i,(j,j_d,j_2)}\right) = \left(\left(\sum_{j_1=1}^{\tilde{r}_1} \left(\tilde{v}_1(j_1) \otimes v_2'(j_1, j_2, j_d)\right)_{i,j}\right)_{i,(j,j_d,j_2)}\right), \tag{5.1}$$

with

$$\tilde{v}_1 : \{1, \ldots \tilde{r}_1\} \to \mathcal{V}_1$$

and

$$v_2' : \{1, \dots \tilde{r}_1\} \times \{1, \dots, r_2\} \times \{1, \dots, r_d\} \to \mathcal{V}_2$$

where $\tilde{r}_1 \leqslant \min(n_1, n_2 \cdot r_d \cdot r_2)$ is the full matrix rank. Consequently,

$$v = \sum_{j_1=1}^{\tilde{r}_1} \sum_{j_2,\dots,j_d=1}^{r_2,\dots,r_d} \tilde{v}_1(j_1) \otimes v_2'(j_1, j_2, j_d) \otimes v_3(j_2, j_3) \otimes \dots \otimes v_d(j_{d-1}, j_d),$$

whose schematic representation is Figure 5.3.

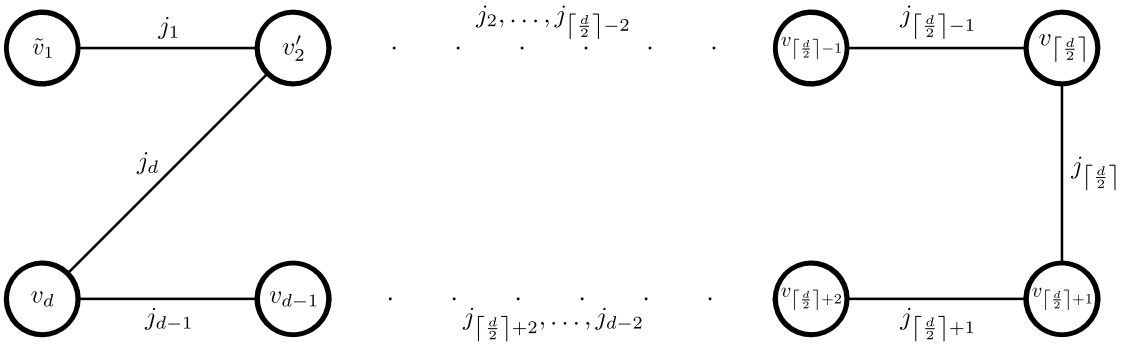

Figure 5.3: Structure after the 1st step

Remark 5.1.1. *Instead of using the full matrix notation as of Equation* (5.1)*, we will use a much shorter notation. Instead of Equation* (5.1) *for instance, we will simply write*

$$v_{1,2}(j_d, j_2) = \sum_{j_1=1}^{\tilde{r}_1} \tilde{v}_1(j_1) \otimes v_2'(j_1, j_2, j_d)$$

from now on.

2nd step

Analogous to step 1, we define

$$v_{d-1,d}(j_{d-2}, j_d)_{i,j} := \sum_{j_{d-1}=1}^{r_{d-1}} (v_{d-1}(j_{d-2}, j_{d-1}) \otimes v_d(j_{d-1}, j_d))_{i,j}$$

for all $j_{d-2} \in \{1, \dots, r_{d-2}\}, j_d \in \{1, \dots, r_d\}$ and interpret $v_{d-1,d}$ as $n_{d-1} \cdot r_{d-2} \cdot r_d \times n_d$ matrix $\left((v_{d-1,d}(j_{d-2}, j_d)_{i,j})_{(i,j_{d-2},j_d),j}\right)$ of which we compute the SVD, in order to get

$$v_{d-1,d}(j_{d-2}, j_d) = \sum_{j_{d-1}=1}^{\tilde{r}_{d-1}} v_{d-1}'(j_{d-2}, j_{d-1}, j_d) \otimes \tilde{v}_d(j_{d-1})$$

with

$$v'_{d-1} : \{1, \dots, r_{d-2}\} \times \{1, \dots, \tilde{r}_{d-1}\} \times \{1, \dots, r_d\} \to \mathcal{V}_{d-1}$$

and

$$\tilde{v}_d : \{1, \dots, \tilde{r}_{d-1}\} \to \mathcal{V}_d$$

where again $\tilde{r}_{d-1} \leqslant \min(n_{d-1} \cdot r_{d-2} \cdot r_d, n_d)$ is the full matrix rank. The result is

$$v = \sum_{j_1, j_{d-1}=1}^{\tilde{r}_1, \tilde{r}_{d-1}} \sum_{j_2, \dots, j_{d-2}, j_d=1}^{r_2, \dots, r_{d-2}, r_d} \tilde{v}_1(j_1) \otimes v'_2(j_1, j_2, j_d) \otimes v_3(j_2, j_3) \otimes \dots \otimes v_{d-2}(j_{d-3}, j_{d-2}) \otimes v'_{d-1}(j_{d-2}, j_{d-1}, j_d) \otimes \tilde{v}_d(j_{d-1})$$

as visualized in Figure 5.4.

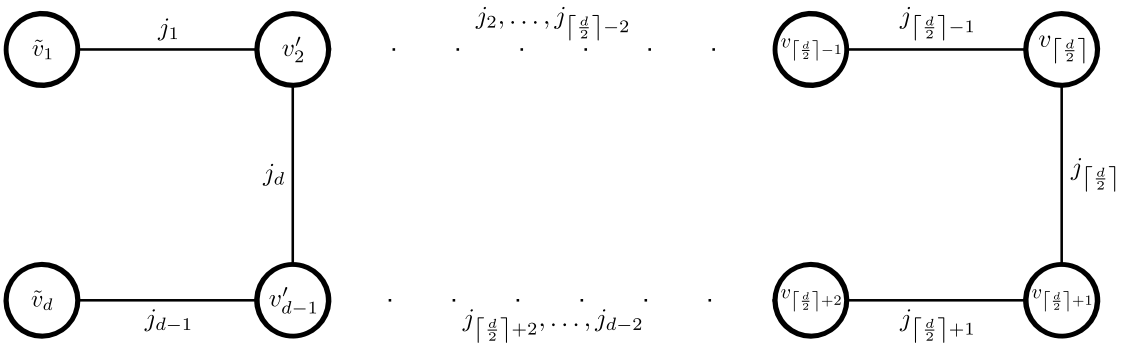

Figure 5.4: Structure after the 2nd step

Penultimate step

We apply the above written scheme successively, we end up in a situation that is equivalent to Figure 5.5.

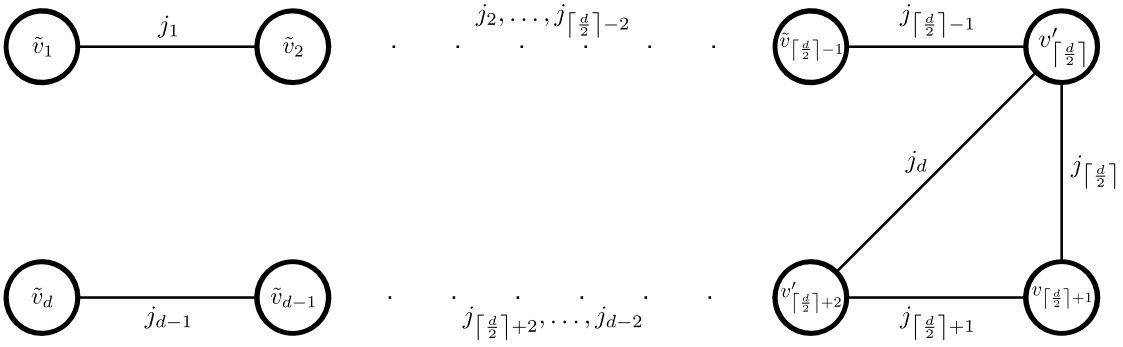

Figure 5.5: Structure before the penultimate step

The penultimate step is to compute the SVD analogously to the procedure that we described in the 1st and 2nd step to get

$$\sum_{j_{\lceil \frac{d}{2} \rceil + 1}=1}^{r_{\lceil \frac{d}{2} \rceil + 1}} v_{\lceil \frac{d}{2} \rceil + 1}\left(j_{\lceil \frac{d}{2} \rceil}, j_{\lceil \frac{d}{2} \rceil + 1}\right) \otimes v'_{\lceil \frac{d}{2} \rceil + 2}\left(j_{\lceil \frac{d}{2} \rceil + 1}, j_{\lceil \frac{d}{2} \rceil + 2}, j_d\right) \overset{SVD}{=}$$
$$\sum_{j_{\lceil \frac{d}{2} \rceil + 1}=1}^{\tilde{r}_{\lceil \frac{d}{2} \rceil + 1}} v'_{\lceil \frac{d}{2} \rceil + 1}\left(j_{\lceil \frac{d}{2} \rceil}, j_{\lceil \frac{d}{2} \rceil + 1}, j_d\right) \otimes \tilde{v}_{\lceil \frac{d}{2} \rceil + 2}\left(j_{\lceil \frac{d}{2} \rceil + 1}, j_{\lceil \frac{d}{2} \rceil + 2}\right)$$

for all $j_{\lceil \frac{d}{2} \rceil} \in \{1, \ldots, r_{\lceil \frac{d}{2} \rceil}\}, j_{\lceil \frac{d}{2} \rceil + 2} \in \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil + 2}\}$ with

$$v'_{\lceil \frac{d}{2} \rceil + 1} : \{1, \ldots, r_{\lceil \frac{d}{2} \rceil}\} \times \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil + 1}\} \times \{1, \ldots, r_d\} \to \mathcal{V}_{\lceil \frac{d}{2} \rceil + 1}$$

and

$$\tilde{v}_{\lceil \frac{d}{2} \rceil + 2} : \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil + 1}\} \times \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil + 2}\} \to \mathcal{V}_{\lceil \frac{d}{2} \rceil + 2}$$

and obtain

$$\begin{aligned} v = \sum_{j_1, \ldots, j_{\lceil \frac{d}{2} \rceil - 1}, j_{\lceil \frac{d}{2} \rceil + 1}, \ldots, j_{d-1}=1}^{\tilde{r}_1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil - 1}, \tilde{r}_{\lceil \frac{d}{2} \rceil + 1}, \ldots, \tilde{r}_{d-1}} \sum_{j_{\lceil \frac{d}{2} \rceil}, j_d=1}^{r_{\lceil \frac{d}{2} \rceil}, r_d} & \tilde{v}_1(j_1) \otimes \tilde{v}_2(j_1, j_2) \otimes \ldots \otimes \tilde{v}_{\lceil \frac{d}{2} \rceil - 1}\left(j_{\lceil \frac{d}{2} \rceil - 2}, j_{\lceil \frac{d}{2} \rceil - 1}\right) \\ & \otimes v'_{\lceil \frac{d}{2} \rceil}\left(j_{\lceil \frac{d}{2} \rceil - 1}, j_{\lceil \frac{d}{2} \rceil}, j_d\right) \otimes v'_{\lceil \frac{d}{2} \rceil + 1}\left(j_{\lceil \frac{d}{2} \rceil}, j_{\lceil \frac{d}{2} \rceil + 1}, j_d\right) \\ & \otimes \tilde{v}_{\lceil \frac{d}{2} \rceil + 2}\left(j_{\lceil \frac{d}{2} \rceil + 1}, j_{\lceil \frac{d}{2} \rceil + 2}\right) \otimes \ldots \otimes \tilde{v}_{d-1}(j_{d-2}, j_{d-1}) \\ & \otimes \tilde{v}_d(j_{d-1}) \end{aligned}$$

with the corresponding Figure 5.6.

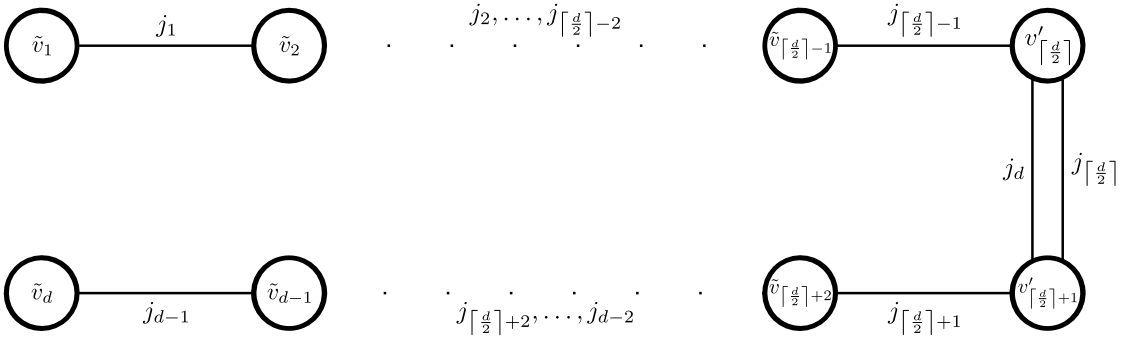

Figure 5.6: Structure after the penultimate step

Remark 5.1.2. *Formally speaking, this structure is already the Tensor Train format since we can interpret edge j_d and $j_{\lceil \frac{d}{2} \rceil}$ as together as one edge with multiplied ranks. This situation however, can be improved (in terms of a possible rank reduction) by computing one additional SVD to combine the two edges. This may also result in a rank reduction, i.e. the rank of the combined edge is from above by the product of the rank of the two edges.*

Final step

In the last step, we compute the singular value decomposition of the matrix

$$M := \left(\left(\sum_{j_{\lceil \frac{d}{2} \rceil}, j_d = 1}^{r_{\lceil \frac{d}{2} \rceil}, r_d} \left(v'_{\lceil \frac{d}{2} \rceil} \left(j_{\lceil \frac{d}{2} \rceil - 1}, j_{\lceil \frac{d}{2} \rceil}, j_d \right) \otimes v'_{\lceil \frac{d}{2} \rceil + 1} \left(j_{\lceil \frac{d}{2} \rceil}, j_{\lceil \frac{d}{2} \rceil + 1}, j_d \right) \right) \right)_{i,j} \right)_{\left(i, j_{\lceil \frac{d}{2} \rceil - 1}\right), \left(j, j_{\lceil \frac{d}{2} \rceil + 1}\right)} \tag{5.2}$$

and obtain analogously to the previous step the structure

$$M \overset{SVD}{=} \left(\left(\sum_{j_{\lceil \frac{d}{2} \rceil} = 1}^{\tilde{r}_{\lceil \frac{d}{2} \rceil}} \left(\tilde{v}_{\lceil \frac{d}{2} \rceil} \left(j_{\lceil \frac{d}{2} \rceil - 1}, j_{\lceil \frac{d}{2} \rceil} \right) \otimes \tilde{v}_{\lceil \frac{d}{2} \rceil + 1} \left(j_{\lceil \frac{d}{2} \rceil}, j_{\lceil \frac{d}{2} \rceil + 1} \right) \right) \right)_{i,j} \right)_{\left(i, j_{\lceil \frac{d}{2} \rceil - 1}\right), \left(j, j_{\lceil \frac{d}{2} \rceil + 1}\right)}$$

with

$$\tilde{v}_{\lceil \frac{d}{2} \rceil} : \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil - 1}\} \times \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil}\} \to \mathcal{V}_{\lceil \frac{d}{2} \rceil}$$

and

$$\tilde{v}_{\lceil \frac{d}{2} \rceil + 1} : \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil}\} \times \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil + 1}\} \to \mathcal{V}_{\lceil \frac{d}{2} \rceil + 1}$$

which is visualized in Figure 5.7.

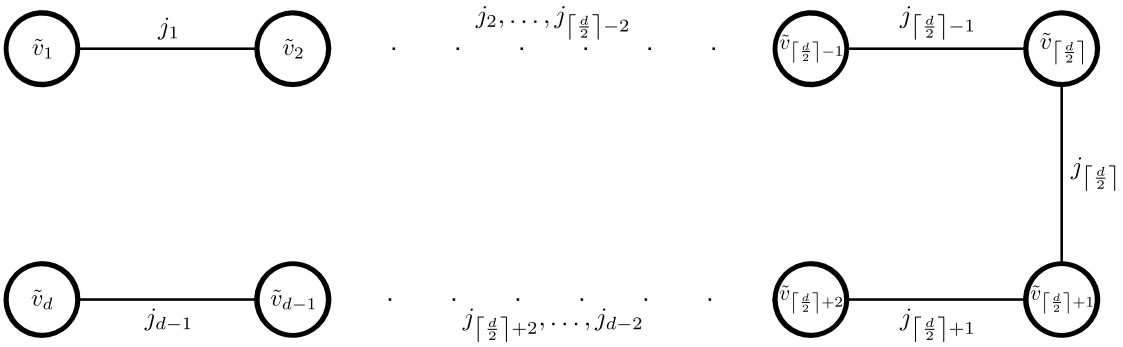

Figure 5.7: Completely converted structure

So by computing the SVD of the above stated matrix M of in (5.2), we combine the summations over j_d and $j_{\lceil \frac{d}{2} \rceil}$.

Ranks

If we consider the new ranks $\tilde{r}_1, \ldots, \tilde{r}_{d-1}$, we have to look at the dimension of the matrices of which we computed the SVD. With $n := \max(n_1, \ldots, n_d)$ and $r := \max(r_1, \ldots, r_d)$ we have

$$\tilde{r}_1 \leqslant \min(n_1, n_2 \cdot r_d \cdot r_d \cdot r_2) \leqslant n_1 \tag{5.3}$$

$$\tilde{r}_i \leqslant \min(n_i \cdot \tilde{r}_{i-1}, n_{i+1} \cdot r_d \cdot r_{i+1}) \leqslant \min(n^i, n_{i+1} \cdot r_d \cdot r_{i+1}) \qquad \text{for } i = 2, \ldots, \left\lceil \frac{d}{2} \right\rceil - 1 \tag{5.4}$$

$$\tilde{r}_{d-1} \leqslant \min(n_d, n_{d-1} \cdot r_{d-2} \cdot r_d) \leqslant n_d \tag{5.5}$$

$$\tilde{r}_i \leqslant \min(n_{i+1} \cdot \tilde{r}_{i+1}, n_i \cdot r_d \cdot r_{i-1}) \leqslant \min(n^{d-i}, n_i \cdot r_d \cdot r_{i-1}) \quad \text{for } i = d-2, \ldots, \left\lceil \frac{d}{2} \right\rceil + 1 \tag{5.6}$$

$$\tilde{r}_{\lceil \frac{d}{2} \rceil} \leqslant \min(n_{\lceil \frac{d}{2} \rceil} \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil - 1}, n_{\lceil \frac{d}{2} \rceil + 1} \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil + 1}) \leqslant \min(n^{\lceil \frac{d}{2} \rceil}, n^2 \cdot r^2), \tag{5.7}$$

which is summarized in Figure 5.8.

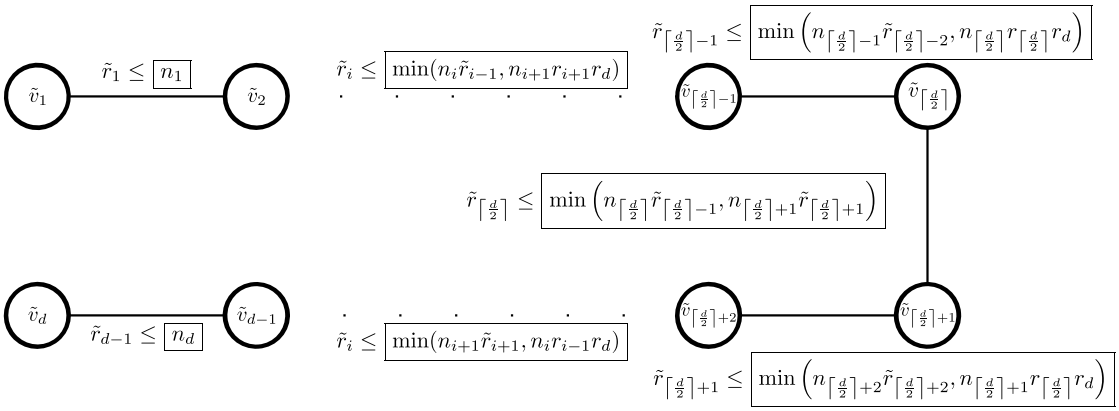

Figure 5.8: Final rank overview

Remark 5.1.3. *With this approach the upper bounds for the ranks are the full TT ranks (see [40] and compare with* (5.3) – (5.7)*), but also the original representation rank of the TC representation influence the resulting representation rank.*

Theorem 5.1.4. *The overall computational cost for the conversion is in*

$$\mathcal{O}\left((d-2) \cdot n^4 r^6 + n^6 r^6\right),$$

so it is linear in d where r is defined as before.

Proof. Due to (5.3) – (5.6), we have

$$\tilde{r}_i \leqslant n \cdot r_d \cdot r_i \leqslant n \cdot r^2 \qquad \forall i \in \{1, \dots, d-1\} \backslash \{\left\lceil \frac{d}{2} \right\rceil\}.$$

Consequently, the matrices, that we have to decompose by computing the SVD, have at most the size

$$n \cdot \tilde{r}_i \times n \cdot r_d \cdot r \qquad \forall i \in \{1, \dots, d-1\} \backslash \{\left\lceil \frac{d}{2} \right\rceil\}$$

except for the final step, the complexity for the SVD is in $\mathcal{O}(n^4 \cdot r^6)$ (see [41, Subsection 5.4.5] for the SVD's computational complexity). There, the matrix has at most the size

$$n \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil + 1} \times n \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil - 1}$$

due to (5.7), such that the complexity for the SVD is in $\mathcal{O}(n^6 \cdot r^6)$ which finishes the proof. □

Remark 5.1.5. *Steps* $1, 3, \dots, d_1$ *and* $2, 4, \dots, d_2$ *are independent of each other and therefore parallelizable where*

$$d_1 := \begin{cases} d-1 & \text{if } d \equiv 0 \mod 2, \\ d-2 & \text{otherwise} \end{cases}$$

and

$$d_2 := \begin{cases} d-2 & \text{if } d \equiv 0 \mod 2, \\ d-1 & \text{otherwise.} \end{cases}$$

Remark 5.1.6. *Instead of moving the edge* j_d *to the center of the chain, we can also for instance fix the* $v_d - j_d$ *connection and move the edge through the whole chain. This however would have the drawback of being not parallelizable.*

Approaches between both possibilities are also possible (e.g. perform step 3 *directly after step* 1 *such that the resulting double edge will not be in the center of the chain).*

We can easily extend this scheme to more complex structures which we will do in Subsection 5.1.2 by converting a rectangular grid structured tensor into a string structured one.

Numerical example

All numerical experiments in this section have been done with [38, test/Representation/TreeConversionTest.cpp] using an Intel i3-3220T CPU and the following setup:

- the function values have are generated with a pseudo-random number generator (i.e. the initial TT tensor is filled with pseudo-random values $\in [0, 1]$)

- each direction has 7 entries, i.e. $n_1 = \ldots = n_d = 7$
- the representation rank of the Tensor Chain tensor is $(r_1, \ldots, r_d) = (6, \ldots, 6)$

The first example will be the full conversion, where we use the maximal matrix rank after each edge *move*. The results are shown in Table 5.1.

d	CPU Time	Avg. rank	Max. rank
4	$0.004s$	16.67	36
5	$0.008s$	24.75	49
6	$0.19s$	29.60	49
7	$0.57s$	66.67	252
8	$25.2s$	93.14	252

Table 5.1: Exact TC to TT conversion

In practice, it is often sufficient to convert a tensor representation only approximately instead of an exact conversion. Our approach can be easily changed to an approximate conversion by using the SVD only up to a certain accuracy (i.e. we cut off small singular values σ_i for which $\sigma_i < \sigma_1 \cdot 10^{-10}$ where σ_1 is the largest singular value). We will demonstrate this by simple computations where the results are shown in Table 5.2.

d	CPU Time	Avg. rank	Max. rank	rel. error
4	$0.0017s$	16.67	36	$2.58 \cdot 10^{-8}$
10	$0.35s$	29.56	36	$2.98 \cdot 10^{-8}$
100	$6.3s$	35.41	36	$4.47 \cdot 10^{-8}$
1000	$65.83s$	35.94	36	$1.35 \cdot 10^{-6}$
10000	$640.7s$	35.99	36	$1.18 \cdot 10^{-5}$

Table 5.2: Approximated TC to TT conversion

So we see that in case of using an approximated SVD, we obtain a conversion that is equivalent to the specialized conversion algorithm of Subsection 2.2.13.

Remark 5.1.7. *We do not need to hold the whole tensor representation in the RAM since the conversion acts only locally on the two involved edges. This reduces the practical memory consumption to a very small fraction of the theoretical consumption (when storing the whole tensor representation in the RAM). Especially if we increase the accuracy of the singular value decomposition by increasing the rank, this locality-advantage is important.*

5.1.2 Converting PEPS to TT without approximation

In the previous Subsection 5.1.1, the topology changed only slightly as we removed just one edge from the graph to obtain a tree. The method that has been used there can

be also used for more complicated structures, such as grids or latices, which we want to explain in this subsection.

We are going to convert a PEPS (see Definition 2.1.14) structured tensor into a tree structured tensor in the TT format. In our framework of arbitrary tensor networks, a PEPS tensor representation v of order $(4,4)$ with representation rank $(r_1,\dots,r_{24})$ is represented as follows:

$$\begin{aligned} v = \sum_{j_1,\dots,j_{24}=1}^{r_1,\dots,r_{24}} & v_1(j_1,j_4)\otimes v_2(j_1,j_2,j_5)\otimes v_3(j_2,j_3,j_6)\otimes v_4(j_3,j_7) \\ & \otimes v_5(j_4,j_8,j_{11})\otimes v_6(j_5,j_8,j_9,j_{12})\otimes v_7(j_6,j_9,j_{10},j_{13})\otimes v_8(j_7,j_{10},j_{14}) \\ & \otimes v_9(j_{11},j_{15},j_{18})\otimes v_{10}(j_{12},j_{15},j_{16},j_{19})\otimes v_{11}(j_{13},j_{16},j_{17},j_{20})\otimes v_{12}(j_{14},j_{17},j_{21}) \\ & \otimes v_{13}(j_{18},j_{22})\otimes v_{14}(j_{19},j_{22},j_{23})\otimes v_{15}(j_{20},j_{23},j_{24})\otimes v_{16}(j_{21},j_{24}), \end{aligned}$$

see Figure 5.9 for the visualization. The Motivation for this conversion is due to the fact that the complexity of contracting (i.e. performing the summations) a PEPS tensor representation is very high and the optimization procedure is not stable (see [42] for an approximated contraction scheme). Tree structured tensor representations on the other hand are stable (as mentioned in Subsection 2.1.3) and easy to contract.

Each tree with p vertices has $p-1$ edges such that it is reasonable to choose the simplest tree structure, which is a string, as the destination structure. This is no restriction of the method, we just chose the string structure only for the sake of clearness ans simplicity.

In order to keep the notations simple, we set $n_1 = \ldots = n_d =: n \in \mathbb{N}$, so all our vector spaces $\mathcal{V}_\mu$ have the same dimension n.

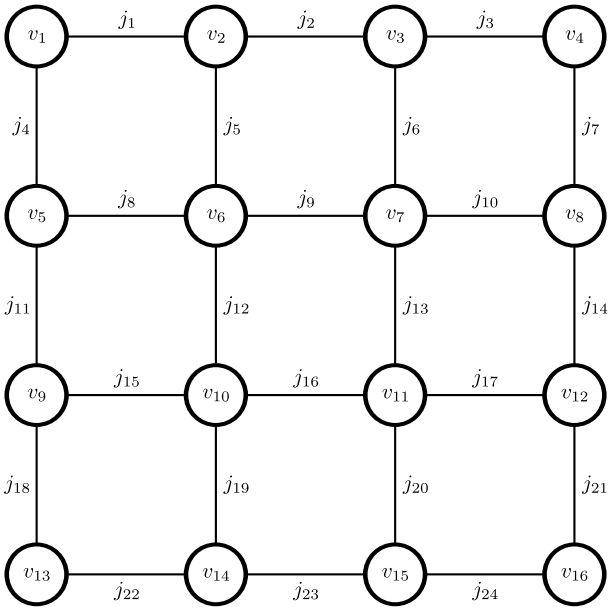

Figure 5.9: PEPS tensor of order $(4, 4)$

We want to visualize the scheme that we introduced in Subsection 5.1.1 by looking at the *upper left corner* of the PEPS tensor representation. Figure 5.10a displays the initial situation.
The first step, that we want to perform, is moving the edge j_4 to the left. Analogously to before, we are doing this by computing the singular value decomposition, such that we obtain (in compliance with Remark 5.1.1)

$$\sum_{j_1=1}^{r_1} v_1(j_1, j_4) \otimes v_2(j_1, j_2, j_5) \overset{SVD}{=} \sum_{j_1=1}^{\tilde{r}_1} \tilde{v}_1(j_1) \otimes v_2'(j_1, j_2, j_4, j_5)$$

for all $j_4 \in \{1, \dots, r_4\}, j_2 \in \{1, \dots, r_2\}$ and $j_5 \in \{1, \dots, r_5\}$ with

$$\tilde{v}_1 : \{1, \dots, \tilde{r}_1\} \to \mathcal{V}_1$$

and

$$v_2' : \{1, \dots, \tilde{r}_1\} \times \{1, \dots, r_2\} \times \{1, \dots, r_4\} \times \{1, \dots, r_5\} \to \mathcal{V}_2.$$

We get the structure as of Figure 5.10b. Hereafter, we apply the same procedure to get

$$\sum_{j_8=1}^{r_8} v_5(j_4, j_8, j_{11}) \otimes v_6(j_5, j_8, j_9, j_{12}) \overset{SVD}{=} \sum_{j_8=1}^{\tilde{r}_8} v_5'(j_8, j_{11}) \otimes v_6'(j_4, j_5, j_8, j_9, j_{12})$$

for all $j_i \in \{1, \dots, r_i\}, i \in \{4, 5, 9, 11, 12\}$ with

$$v_5' : \{1, \dots, \tilde{r}_8\} \times \{1, \dots, r_{11}\} \to \mathcal{V}_5$$

and

$$v_6' : \{1, \dots, r_4\} \times \{1, \dots, r_5\} \times \{1, \dots, \tilde{r}_8\} \times \{1, \dots, r_9\} \times \{1, \dots, r_{12}\} \to \mathcal{V}_6,$$

which is shown in Figure 5.10c. The next step could be to move those two edges j_4 and j_5 both further to the left, but this would increase the complexity of the formulas as well as of the schematic drawings. Additionally, it might be the case that the product of the moved edges ranks is unnecessarily high (see Remark 5.1.2).
So, we want to combine j_4 and j_5 into a new j_5 and we can do this by one SVD analogously to the **Final Step** of Subsection 5.1.3. We obtain

$$\sum_{j_4, j_5 = 1}^{r_4, r_5} v_2'(j_1, j_2, j_4, j_5) \otimes v_6'(j_4, j_5, j_8, j_9, j_{12}) \overset{SVD}{=} \sum_{j_5 = 1}^{\tilde{r}_5} v_2''(j_1, j_2, j_5) \otimes v_6''(j_5, j_8, j_9, j_{12})$$

for all $j_i \in \{1, \dots, r_i\}, i \in \{2, 9, 12\}$ and $j_k \in \{1, \dots, \tilde{r}_k\}, k \in \{1, 8\}$ with

$$v_2'' : \{1, \dots, \tilde{r}_1\} \times \{1, \dots, r_2\} \times \{1, \dots, \tilde{r}_5\} \to \mathcal{V}_2$$

and

$$v_6'' : \{1, \dots, \tilde{r}_5\} \times \{1, \dots, \tilde{r}_8\} \times \{1, \dots, r_9\} \times \{1, \dots, r_{12}\} \to \mathcal{V}_6$$

such that we get a structure as of Figure 5.10d.

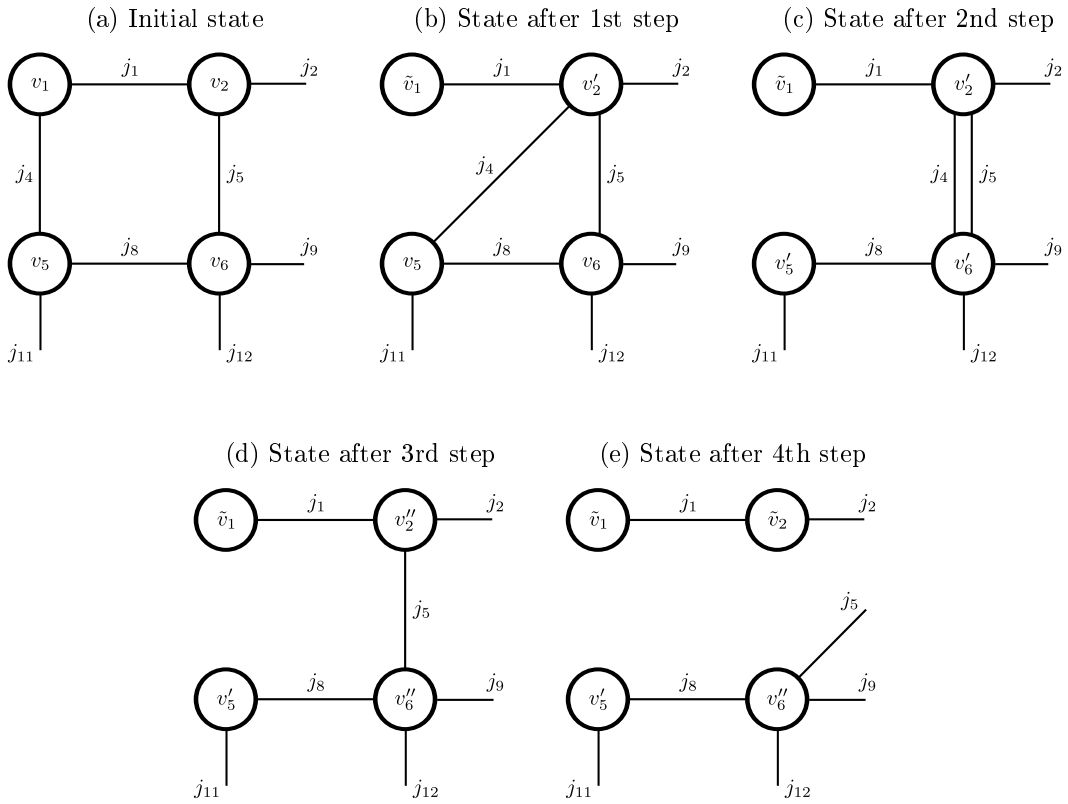

Figure 5.10: Iteration series 1 in details

Afterwards, we can proceed as before (see Figure 5.10e). If we apply this procedure until we have eliminated also edges j_5 and j_6 with edge j_7 left, we get a structure as in Figure 5.11.

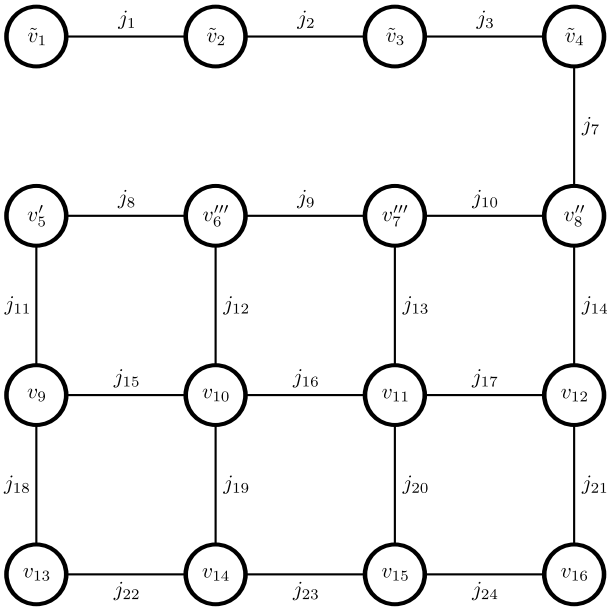

Figure 5.11: PEPS series 1

Applying this scheme also on edges (j_{18}, j_{19}, j_{20}) we first get the structure of Figure 5.12 and afterwards the structure of Figure 5.13 if we eliminate edges (j_{14}, j_{13}, j_{12}).

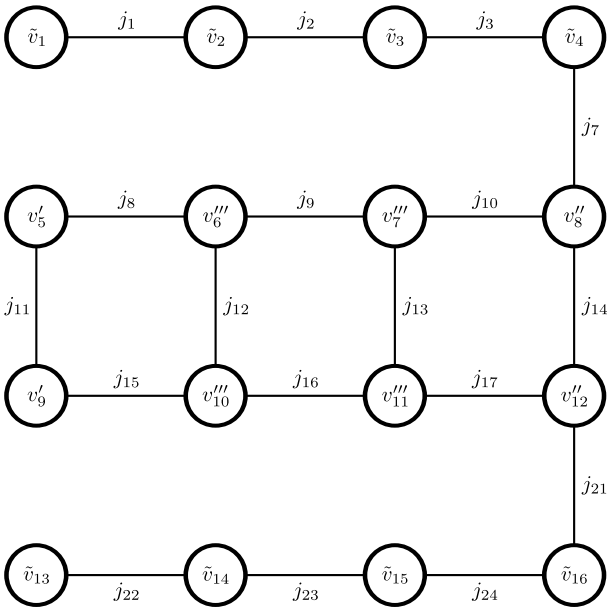

Figure 5.12: PEPS series 2

The edge elimination processes of (j_4, j_5, j_6) and of (j_{18}, j_{19}, j_{20}) do not affect each other such that we can state the rank distribution independently, where $\tilde{r}_i$ is the rank of the edge labeled with j_i after the elimination process:

$$
\begin{aligned}
\tilde{r}_1 &\leqslant n \cdot \min(1, r_2 \cdot r_4 \cdot r_5) = n \\
\tilde{r}_8 &\leqslant n \cdot \min(r_{11}, r_4 \cdot r_5 \cdot r_9 \cdot r_{12}) \\
\tilde{r}_5 &\leqslant n \cdot \min(\tilde{r}_1 \cdot r_2, \tilde{r}_8 \cdot r_9 \cdot r_{12}) = n \cdot \min(n \cdot r_2, \tilde{r}_8 \cdot r_9 \cdot r_{12})
\end{aligned}
$$

$$
\begin{aligned}
\tilde{r}_2 &\leqslant n \cdot \min(\tilde{r}_1, r_3 \cdot \tilde{r}_5 \cdot r_6) = n \cdot \min(n, r_3 \cdot \tilde{r}_5 \cdot r_6) \\
\tilde{r}_9 &\leqslant n \cdot \min(\tilde{r}_8 \cdot r_{12}, \tilde{r}_5 \cdot r_6 \cdot r_{10} \cdot r_{13}) \\
\tilde{r}_6 &\leqslant n \cdot \min(\tilde{r}_2 \cdot r_3, \tilde{r}_9 \cdot r_{10} \cdot r_{13})
\end{aligned}
$$

$$
\begin{aligned}
\tilde{r}_3 &\leqslant n \cdot \min(\tilde{r}_2, \tilde{r}_6 \cdot r_7) \\
\tilde{r}_{10} &\leqslant n \cdot \min(\tilde{r}_9 \cdot r_{13}, \tilde{r}_6 \cdot r_7 \cdot r_{14}) \\
\tilde{r}_7 &\leqslant n \cdot \min(\tilde{r}_3, \tilde{r}_{10} \cdot r_{14})
\end{aligned}
$$

$$
\begin{aligned}
\tilde{r}_{22} &\leqslant n \cdot \min(1, r_{18} \cdot r_{19} \cdot r_{23}) = n \\
\tilde{r}_{15} &\leqslant n \cdot \min(r_{11}, r_{12} \cdot r_{16} \cdot r_{18} \cdot r_{19}) \\
\tilde{r}_{19} &\leqslant n \cdot \min(\tilde{r}_{22} \cdot r_{23}, r_{12} \cdot \tilde{r}_{15} \cdot r_{16}) = n \cdot \min(n \cdot r_{23}, r_{12} \cdot \tilde{r}_{15} \cdot r_{16})
\end{aligned}
$$

$$
\begin{aligned}
\tilde{r}_{23} &\leqslant n \cdot \min(\tilde{r}_{22}, \tilde{r}_{19} \cdot r_{20} \cdot r_{24}) = n \cdot \min(n, \tilde{r}_{19} \cdot r_{20} \cdot r_{24}) \\
\tilde{r}_{16} &\leqslant n \cdot \min(r_{12} \cdot \tilde{r}_{15}, r_{13} \cdot r_{17} \cdot \tilde{r}_{19} \cdot r_{20}) \\
\tilde{r}_{20} &\leqslant n \cdot \min(\tilde{r}_{23} \cdot r_{24}, r_{13} \cdot \tilde{r}_{16} \cdot r_{17})
\end{aligned}
$$

$$
\begin{aligned}
\tilde{r}_{24} &\leqslant n \cdot \min(\tilde{r}_{23}, \tilde{r}_{20} \cdot r_{21}) \\
\tilde{r}_{17} &\leqslant n \cdot \min(r_{13} \cdot \tilde{r}_{16}, r_{14} \cdot \tilde{r}_{20} \cdot r_{21}) \\
\tilde{r}_{21} &\leqslant n \cdot \min(\tilde{r}_{24}, r_{14} \cdot \tilde{r}_{17})
\end{aligned}
$$

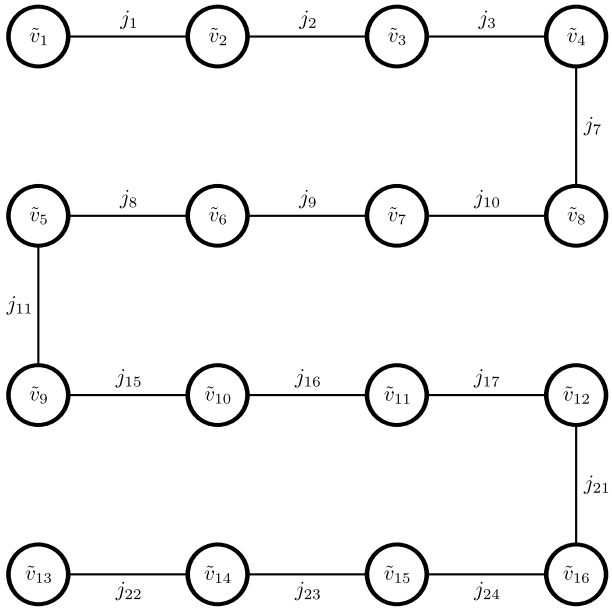

Figure 5.13: PEPS series 3

The edge elimination of the j_{12}, j_{13} and j_{14} results in an additional adjustment of the ranks, which are denoted with $\tilde{\tilde{r}}_i$ for edge j_i:

$$\begin{aligned}
\tilde{\tilde{r}}_{10} &\leqslant n \cdot \min(\tilde{r}_7, \tilde{r}_9 \cdot r_{13} \cdot r_{14}) \\
\tilde{\tilde{r}}_{17} &\leqslant n \cdot \min(\tilde{r}_{21}, r_{13} \cdot r_{14} \cdot \tilde{r}_{16}) \\
\tilde{\tilde{r}}_{13} &\leqslant n \cdot \min(\tilde{r}_9 \cdot \tilde{\tilde{r}}_{10}, \tilde{r}_{16} \cdot \tilde{\tilde{r}}_{17}) \\
\\
\tilde{\tilde{r}}_{9} &\leqslant n \cdot \min(\tilde{\tilde{r}}_{10}, \tilde{r}_8 \cdot r_{12} \cdot \tilde{r}_{13}) \\
\tilde{\tilde{r}}_{16} &\leqslant n \cdot \min(\tilde{\tilde{r}}_{17}, r_{12} \cdot \tilde{r}_{13} \cdot \tilde{r}_{15}) \\
\tilde{\tilde{r}}_{12} &\leqslant n \cdot \min(\tilde{r}_8 \cdot \tilde{\tilde{r}}_{9}, \tilde{r}_{15} \cdot \tilde{\tilde{r}}_{16}) \\
\\
\tilde{\tilde{r}}_{8} &\leqslant n \cdot \min(\tilde{\tilde{r}}_{9}, r_{11} \cdot \tilde{r}_{12}) \\
\tilde{\tilde{r}}_{15} &\leqslant n \cdot \min(\tilde{\tilde{r}}_{16}, r_{11} \cdot \tilde{r}_{12}) \\
\tilde{\tilde{r}}_{11} &\leqslant n \cdot \min(\tilde{\tilde{r}}_{15}, \tilde{\tilde{r}}_{8}),
\end{aligned}$$

resulting in the tree structure which had to be established.

Remark 5.1.8. *Series* 1 *and series* 2 *are parallelizable without any restriction since they do not affect a common vertex. Series* 3 *can be performed at the same time as series* 1 *and* 2 *but one has to be careful with overlapping cycle elimination series since it might*

be possible that v_6 is changed by two processes at the same time, for instance. This can be worked around by adding simple synchronization barriers. Note that there is at most one edge of the edges in common of two processes that may be changed simultaneously.

Performing the conversion in parallel may lead to different ranks in the ranks that are adjusted more than once, since several ranks get adjusted twice and the order of these adjustments influences the final rank.

Remark 5.1.9. *The order of the series is not unique. One can choose any other series that produces a string-like tree. For instance, one could also choose to eliminate edges $j_1, j_3, j_8, j_9, j_{10}, j_{15}, j_{16}, j_{17}$ and j_{23}.*

5.1.3 Direct conversion from TT to TC without approximation

If the physical underlying structure of a problem suggests that a ring structure is more suitable than a string structure, one is able to change the topology of the tensor network that represents the tensor of interest. Due to the instability of the Tensor Chain format (see Remark 2.1.12), the conversion is also unstable.

We want to convert the Tensor Train representation into a cyclic structured tensor representation (Tensor Chain). In general, the Tensor Train format is a special Tensor Chain format since there is a rank-one edge between v_1 and v_d on every Tensor Train representation. Our objective here is to get a *balanced* distribution of the ranks for the Tensor Chain representation. Therefore, we have to perform a procedure that successively moves an artificially inserted edge to the start v_1 and the end v_d of the *train*. In practice however, this leads to several problems that are inspected in Subsection 5.1.3.

This procedure also depends on the singular value decomposition (SVD). An intermediate vertex (i.e. changed by a SVD but not yet the final vertex) will be denoted with $'$ whereas the final converted vertex has a $\tilde{}$, just as in Subsection 5.1.1.

1st step

Our first step will be to introduce an artificial edge between vertex $v_{\lceil \frac{d}{2} \rceil}$ and $v_{\lceil \frac{d}{2} \rceil + 1}$ which we want to name j_d (see Figure 5.14 for the visualization).

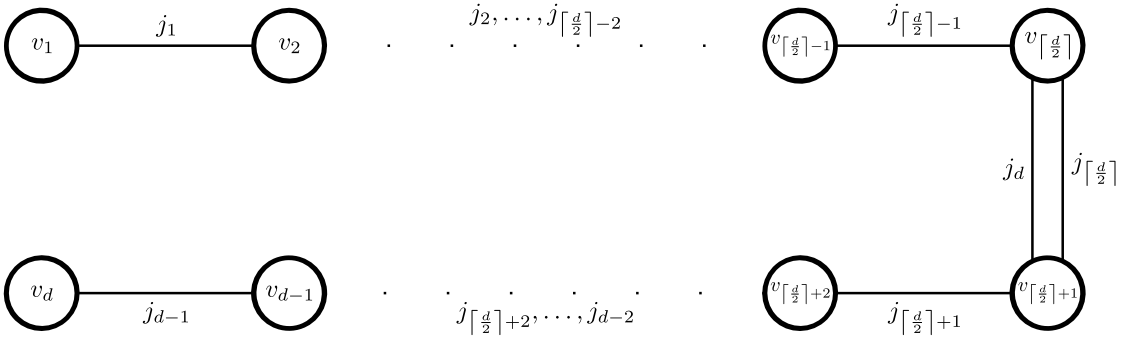

Figure 5.14: Artificially added edge j_d

So we get a new tensor representation with representation rank $(r_1, \dots, r_d)$. We choose r_d and $\tilde{r}_{\lceil \frac{d}{2} \rceil}$ such that $r_d \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil} \geqslant r_{\lceil \frac{d}{2} \rceil}$ and define the mapping

$$\begin{aligned} [\cdot,\cdot] : \{1, \dots, \tilde{r}_{\lceil \frac{d}{2} \rceil}\} \times \{1, \dots, r_d\} &\to \{1, \dots, r_d \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil}\} \\ a, b &\mapsto a + \tilde{r}_{\lceil \frac{d}{2} \rceil} \cdot b, \end{aligned} \tag{5.8}$$

so $[\cdot,\cdot]$ is a bijective map to assign a 2-tuple to a natural number. Consequently, we have

$$\begin{aligned} &\sum_{j_{\lceil \frac{d}{2} \rceil}=1}^{r_{\lceil \frac{d}{2} \rceil}} v_{\lceil \frac{d}{2} \rceil}\left(j_{\lceil \frac{d}{2} \rceil - 1}, j_{\lceil \frac{d}{2} \rceil}\right) \otimes v_{\lceil \frac{d}{2} \rceil + 1}\left(j_{\lceil \frac{d}{2} \rceil}, j_{\lceil \frac{d}{2} \rceil + 1}\right) = \\ &\sum_{j_{\lceil \frac{d}{2} \rceil}, j_d = 1}^{\tilde{r}_{\lceil \frac{d}{2} \rceil}, r_d} v_{\lceil \frac{d}{2} \rceil}\left(j_{\lceil \frac{d}{2} \rceil - 1}, \left[j_{\lceil \frac{d}{2} \rceil}, j_d\right]\right) \otimes v_{\lceil \frac{d}{2} \rceil + 1}\left(\left[j_{\lceil \frac{d}{2} \rceil}, j_d\right], j_{\lceil \frac{d}{2} \rceil + 1}\right). \end{aligned}$$

for all $j_{\lceil \frac{d}{2} \rceil - 1} \in \{1, \dots, r_{\lceil \frac{d}{2} \rceil - 1}\}$ and $j_{\lceil \frac{d}{2} \rceil + 1} \in \{1, \dots, r_{\lceil \frac{d}{2} \rceil + 1}\}$. If $r_d \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil} > r_{\lceil \frac{d}{2} \rceil}$, we have to redefine $v_{\lceil \frac{d}{2} \rceil}$ and $v_{\lceil \frac{d}{2} \rceil + 1}$:

$$\begin{aligned} v^{\text{redefine}}_{\lceil \frac{d}{2} \rceil} : \{1, \dots, r_{\lceil \frac{d}{2} \rceil - 1}\} \times \{1, \dots, r_d \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil}\} &\to \mathcal{V}_{\lceil \frac{d}{2} \rceil} \\ \left(j_{\lceil \frac{d}{2} \rceil - 1}, j_{\lceil \frac{d}{2} \rceil}\right) &\mapsto \begin{cases} v_{\lceil \frac{d}{2} \rceil}\left(j_{\lceil \frac{d}{2} \rceil - 1}, j_{\lceil \frac{d}{2} \rceil}\right) & \text{if } j_{\lceil \frac{d}{2} \rceil} \leqslant r_{\lceil \frac{d}{2} \rceil} \\ 0 & \text{else} \end{cases} \end{aligned}$$

and

$$\begin{aligned} v^{\text{redefine}}_{\lceil \frac{d}{2} \rceil + 1} : \{1, \dots, r_d \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil}\} \times \{1, \dots, r_{\lceil \frac{d}{2} \rceil + 1}\} &\to \mathcal{V}_{\lceil \frac{d}{2} \rceil + 1} \\ \left(j_{\lceil \frac{d}{2} \rceil}, j_{\lceil \frac{d}{2} \rceil + 1}\right) &\mapsto \begin{cases} v_{\lceil \frac{d}{2} \rceil + 1}\left(j_{\lceil \frac{d}{2} \rceil}, j_{\lceil \frac{d}{2} \rceil + 1}\right) & \text{if } j_{\lceil \frac{d}{2} \rceil} \leqslant r_{\lceil \frac{d}{2} \rceil} \\ 0 & \text{else} \end{cases}. \end{aligned}$$

Note that even if we had to redefine the two mappings, we still use the original mapping names $v_{\lceil \frac{d}{2} \rceil + 1}$ and $v_{\lceil \frac{d}{2} \rceil + 1}$ instead of $v^{\text{redefine}}_{\lceil \frac{d}{2} \rceil}$ and $v^{\text{redefine}}_{\lceil \frac{d}{2} \rceil + 1}$ for the sake of a readable notation.
We may also choose any other bijective mapping that satisfies

$$\{1, \dots, \tilde{r}_{\lceil \frac{d}{2} \rceil}\} \times \{1, \dots, r_d\} \to \{1, \dots, r_d \cdot \tilde{r}_{\lceil \frac{d}{2} \rceil}\}$$

which will we address later in this subsection.

2nd step

In this step, we want to move the edge j_d from vertex $v_{\lceil \frac{d}{2} \rceil + 1}$ to $v_{\lceil \frac{d}{2} \rceil + 2}$ and as written before, we will do this with a single SVD and – again in compliance with Remark 5.1.1

– end up with

$$\sum_{j_{\lceil \frac{d}{2} \rceil + 1}=1}^{r_{\lceil \frac{d}{2} \rceil + 1}} v_{\lceil \frac{d}{2} \rceil + 1}\left(\left[j_{\lceil \frac{d}{2} \rceil}, j_d\right], j_{\lceil \frac{d}{2} \rceil + 1}\right) \otimes v_{\lceil \frac{d}{2} \rceil + 2}\left(j_{\lceil \frac{d}{2} \rceil + 1}, j_{\lceil \frac{d}{2} \rceil + 2}\right) \overset{SVD}{=}$$

$$\sum_{j_{\lceil \frac{d}{2} \rceil + 1}=1}^{\tilde{r}_{\lceil \frac{d}{2} \rceil + 1}} \tilde{v}_{\lceil \frac{d}{2} \rceil + 1}\left(j_{\lceil \frac{d}{2} \rceil}, j_{\lceil \frac{d}{2} \rceil + 1}\right) \otimes v'_{\lceil \frac{d}{2} \rceil + 2}\left(j_{\lceil \frac{d}{2} \rceil + 1}, j_{\lceil \frac{d}{2} \rceil + 2}, j_d\right)$$

for all $j_{\lceil \frac{d}{2} \rceil} \in \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil}\}, j_{\lceil \frac{d}{2} \rceil + 2} \in \{1, \ldots, r_{\lceil \frac{d}{2} \rceil + 2}\}, j_d \in \{1, \ldots, r_d\}$ with

$$\tilde{v}_{\lceil \frac{d}{2} \rceil + 1} : \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil}\} \times \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil + 1}\} \to \mathcal{V}_{\lceil \frac{d}{2} \rceil + 1}$$

and

$$v_{\lceil \frac{d}{2} \rceil + 2} : \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil + 1}\} \times \{1, \ldots, r_{\lceil \frac{d}{2} \rceil + 2}\} \times \{1, \ldots, r_d\} \to \mathcal{V}_{\lceil \frac{d}{2} \rceil + 2}$$

analogously to before. The visual representation of the situation after the 2nd step is shown in Figure 5.15.

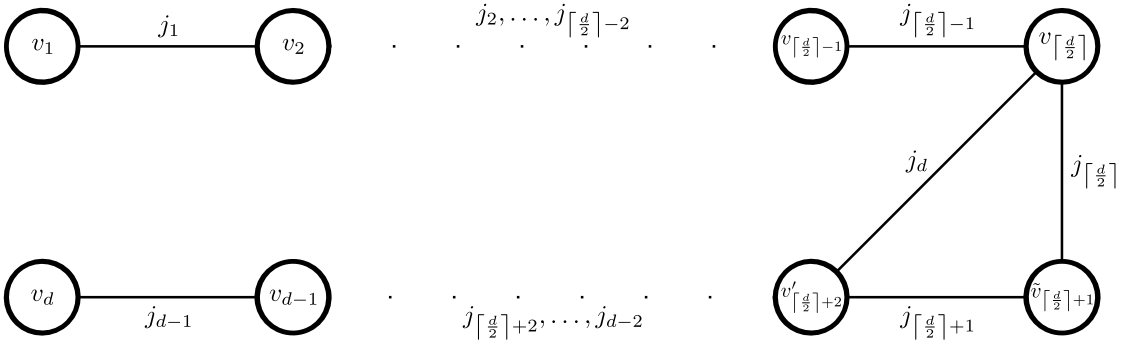

Figure 5.15: Situation after the 2nd step

3rd step

Edge j_d has to be changed such that it connects vertex $v_{\lceil \frac{d}{2} \rceil - 1}$ and vertex $v'_{\lceil \frac{d}{2} \rceil + 2}$. This will be done analogously to the second step, such that we get

$$\sum_{j_{\lceil \frac{d}{2} \rceil - 1}=1}^{r_{\lceil \frac{d}{2} \rceil - 1}} v_{\lceil \frac{d}{2} \rceil - 1}\left(j_{\lceil \frac{d}{2} \rceil - 2}, j_{\lceil \frac{d}{2} \rceil - 1}\right) \otimes v_{\lceil \frac{d}{2} \rceil}\left(j_{\lceil \frac{d}{2} \rceil - 1}, \left[j_{\lceil \frac{d}{2} \rceil}, j_d\right]\right) \overset{SVD}{=}$$

$$\sum_{j_{\lceil \frac{d}{2} \rceil - 1}=1}^{\tilde{r}_{\lceil \frac{d}{2} \rceil - 1}} v'_{\lceil \frac{d}{2} \rceil - 1}\left(j_{\lceil \frac{d}{2} \rceil - 2}, j_{\lceil \frac{d}{2} \rceil - 1}, j_d\right) \otimes \tilde{v}_{\lceil \frac{d}{2} \rceil}\left(j_{\lceil \frac{d}{2} \rceil - 1}, j_{\lceil \frac{d}{2} \rceil}\right),$$

for all $j_{\lceil \frac{d}{2} \rceil - 1} \in \{1, \ldots, r_{\lceil \frac{d}{2} \rceil - 2}\}, j_{\lceil \frac{d}{2} \rceil} \in \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil}\}, j_d \in \{1, \ldots, r_d\}$ with

$$v'_{\lceil \frac{d}{2} \rceil - 1} : \{1, \ldots, r_{\lceil \frac{d}{2} \rceil - 2}\} \times \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil - 1}\} \times \{1, \ldots, r_d\} \to \mathcal{V}_{\lceil \frac{d}{2} \rceil - 1}$$

and

$$\tilde{v}_{\lceil \frac{d}{2} \rceil} : \times \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil - 1}\} \times \{1, \ldots, \tilde{r}_{\lceil \frac{d}{2} \rceil}\} \to \mathcal{V}_{\lceil \frac{d}{2} \rceil},$$

which is visualized in Figure 5.16.

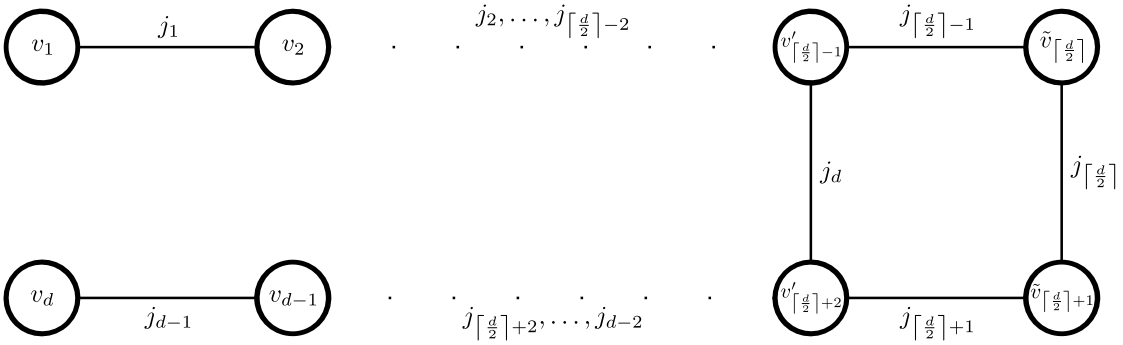

Figure 5.16: Situation after the 3rd step

Remark 5.1.10. *Step* 2 *and* 3 *are independent of each other and can be performed in parallel.*

Final step

After *moving* the edge successively further towards v_d and v_1, we get the situation that is visualized in Figure 5.17. The last step in the conversion is to *move* edge j_d such that it connects vertices $\tilde{v}_d$ and v_1 by using the described procedure.

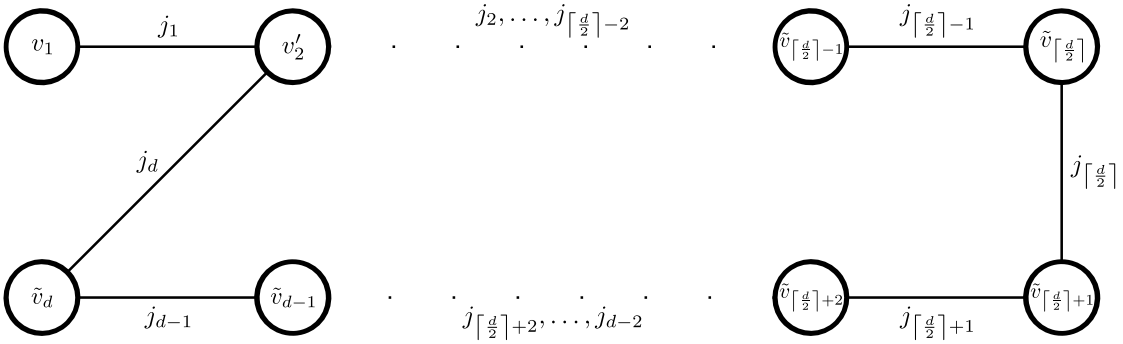

Figure 5.17: Situation before the final step

So in formulas, one SVD is computed to make the edge shift:

$$\sum_{j_1=1}^{r_1} v_1(j_1) \otimes v_2'(j_1, j_2, j_d) \overset{SVD}{=} \sum_{j_1=1}^{\tilde{r}_1} \tilde{v}_1(j_1, j_d) \otimes \tilde{v}_2(j_1, j_2),$$

for all $j_2 \in \{1, \ldots, \tilde{r}_2\}, j_d \in \{1, \ldots, r_d\}$ where

$$\tilde{v}_1 : \{1, \ldots, \tilde{r}_1\} \times \{1, \ldots, r_d\} \to \mathcal{V}_1$$

and

$$\tilde{v}_2 : \{1, \ldots, \tilde{r}_1\} \times \{1, \ldots, \tilde{r}_2\} \to \mathcal{V}_2,$$

which is resulting in the structure that we wanted to obtain (see Figure 5.18).

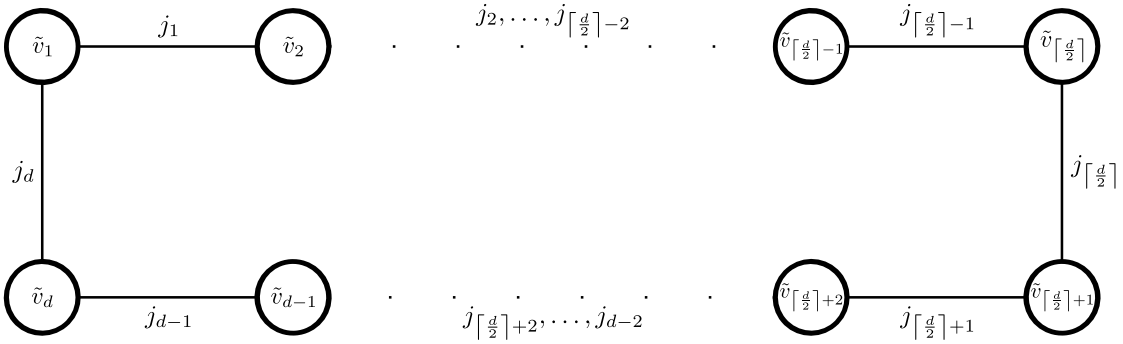

Figure 5.18: Situation after the final step

Ranks

After we have chosen the ranks r_d and $\tilde{r}_{\lceil \frac{d}{2} \rceil}$, we update all remaining $d-2$ ranks and get the following upper bounds (with n and r as defined before)

$$\tilde{r}_1 = \min(n_1 \cdot r_d, n_2 \cdot \tilde{r}_2) \leqslant n \cdot r, \tag{5.9}$$

$$\tilde{r}_i = \min(n_i \cdot r_{i-1} \cdot r_d, n_{i+1} \cdot \tilde{r}_{i+1}) \leqslant n \cdot r \cdot r_d \qquad \text{for } i = 2, \ldots, \left\lceil \frac{d}{2} \right\rceil - 1 \tag{5.10}$$

and

$$\tilde{r}_i = \min(n_{i+1} \cdot r_{i+1} \cdot r_d, n_i \cdot \tilde{r}_{i-1}) \leqslant n \cdot r \cdot r_d \qquad \text{for } i = \left\lceil \frac{d}{2} \right\rceil + 1, \ldots, d-1. \tag{5.11}$$

Theorem 5.1.11. *The computational cost of the described scheme is in*

$$\mathcal{O}\left((d-2) \cdot n^3 \cdot r^3 \cdot r_d^3 + n^3 \cdot r^3\right)$$

so it linear in d.

Proof. Follows directly from Equations (5.9) – (5.11) since these equations determine the upper bound for the matrix sizes. □

Problems

The main problem has its roots in the first step where an artificial edge is introduced into the graph structure. There we do not a priori know what the *best* rank splitting is and we also do not know which is the *best* assignment for the $[\cdot,\cdot]$ function (5.8). If we can solve these problems, we are - for example - able to convert a Tensor Chain formatted tensor into a Tensor Train formatted tensor and back without different ranks for the Tensor Chain tensor in before the conversion and after the back-conversion.

In [30, Section 4] for instance, there is an example where the represented tensor, that was originally represented in the Tensor Chain format, is perfectly recovered by choosing a suitable $[\cdot,\cdot]$ function. This example is artificial but it proves the point.

Numerical example

We have the same setup as in 5.1.1 (except that we are converting a Tensor Train representation with representation rank $(r_1,\dots,r_{d-1}) = (6,\dots,6)$ into a Tensor Chain representation) and obtain the results in Table 5.3 with approximated SVD (i.e. we neglect small singular values σ_i for which $\sigma_i < \sigma_1 \cdot 10^{-10}$ where σ_1 is the largest singular value) where the relative error is w.r.t. the initial tensor representation. The implementation [38, test/Representation/TreeConversionTest.cpp] was used with an Intel i3-3220T CPU.

d	CPU Time	Avg. rank	Max. Rank	Rel. error
4	$0.0003s$	7.25	12	$6.14 \cdot 10^{-8}$
10	$0.02s$	10.1	12	$8.16 \cdot 10^{-8}$
100	$0.38s$	11.81	12	$4.52 \cdot 10^{-7}$
1000	$4.09s$	11.98	12	$3.19 \cdot 10^{-6}$
10000	$41.25s$	12	12	$1.78 \cdot 10^{-5}$

Table 5.3: Approximated TT to TC conversion

To illustrate the problem that has been described in Subsection 5.1.3, we will run a second experiment: first, we will transform a Tensor Chain tensor representation into a Tensor Train tensor representation and then, we will re-transform it back to the original chain format. In this experiment (of which the results are shown in Table 5.4), we also have the same setup as in Subsection 5.1.1 (initial TC representation rank is $(r_1,\dots,r_d) = (6,\dots,6)$). No SVD approximation is considered.

d	Avg. converted TT rank	Avg. re-converted TC rank
4	16.67	24
6	29.6	30
8	93.14	96

Table 5.4: Exact TC to TT to TC conversion

In the previous computation, we used the full matrix ranks such that we did not

benefit from the possibility of a matrix approximation. So we are going to change the algorithm to not use the full rank, but a rank, generated by the SVD decomposition (where we cut off small singular values as before) for both conversions which results in Table 5.5. The error in this table is again the relative error with respect to the initial tensor representation. The initial TC representation rank is also $(r_1, \dots, r_d) = (6, \dots, 6)$.

d	Avg. converted TT rank	Rel. error TT	Avg. re-converted TC rank	Rel. error TC
4	16.67	$6.66 \cdot 10^{-8}$	24	$4.71 \cdot 10^{-8}$
6	24.4	$2.58 \cdot 10^{-8}$	26	$7.15 \cdot 10^{-8}$
8	27.71	$9.42 \cdot 10^{-8}$	28.5	$6.32 \cdot 10^{-8}$
10	29.56	$3.65 \cdot 10^{-8}$	30	$1.03 \cdot 10^{-7}$
12	30.73	$6.99 \cdot 10^{-8}$	31	$1.07 \cdot 10^{-7}$
20	32.95	$7.30 \cdot 10^{-8}$	33	$1.81 \cdot 10^{-7}$
30	34	$2.64 \cdot 10^{-7}$	34	$1.24 \cdot 10^{-7}$

Table 5.5: Approximated TC to TT to TC conversion

5.1.4 General method

For the description of the general method, we mainly focus on graph theoretical aspects. From the tensor network point of view, we only have to describe local changes, i.e. one edge shift only involves the vertices that are connected by one edge before and after the edge shift. The conversion of one graph into another is then performed by multiple local topology changes. So the general task is to convert one connected simple graph with d edges to another one.

At first, we want to give the definition of a *walk* in graph theoretical terms. We simplify [31, p. 29, Definition for *Walk*] and combine it with [31, p. 29, Definition for *Length of a walk*] to obtain

Definition 5.1.12 (Walk). *Let $G = (V, E)$ be a connected simple undirected graph as of Definition 3.1.1. Then a* walk *of length $\ell \in \mathbb{N}, \ell > 1$ from $v \in V$ to $w \in V$ is a sequence*

$$(v, u_1, \dots, u_{\ell-1}, w) \in V^{\ell+1}$$

of vertices with

$$\{v, u_1\} \in E, \{u_i, u_{i+1}\} \in E \ \forall i \in \{1, \dots, \ell-2\}, \{u_{\ell-1}, w\} \in E$$

and $u_i \notin \{v, w\}$ for all $i \in \{1, \dots, \ell-1\}$. For simplicity, we assume $v \neq w$.

So now let us assume we have a Graph $G = (V, E)$ where we want to eliminate edge $e = \{v_1, v_{\ell+1}\} \in V \biguplus V$ for some $\ell \in \mathbb{N}\backslash\{1\}$. The first requirement for our algorithm to work, is that there is a walk $(v_1, \dots, v_{\ell+1}) \in V^{\ell+1}$ of length ℓ from v_1 to $v_{\ell+1}$. If such a walk does not exist, we have no *connection* besides e from v_1 to $v_{\ell+1}$. Due to Remark 3.1.2 (connectedness of the graphs) we have at least one of the walk or the edge e.

Once we have found the walk, we can *move* the edge e successively along this walk to eliminate it (see Subsection 5.1.1 for the example). Therefore, we conclude with the following

Algorithm 5 Edge elimination algorithm (graph theory)

1: Let $G = (V, E)$ be an undirected simple connected graph, $(v_1, \dots, v_{\ell+1}) \in V^{\ell+1}$ be a walk of length $\ell \in \mathbb{N}\backslash\{1\}$ from v_1 to $v_{\ell+1}$ and $e := \{v_1, v_{\ell+1}\} \in E$
2: Define $start := 1$ and $end := \ell + 1$
3: **while** $end - start > 1$ **do**
4: Choose either $start \mapsto start + 1$ or $end \mapsto end - 1$
5: $E \mapsto (E\backslash\{e\}) \cup \{\{v_{start}, v_{end}\}\}$
6: $e \mapsto \{v_{start}, v_{end}\}$
7: **end while**

The last step of this algorithm adds an edge e to the edge set that is already contained in the edge set E. In line 5 of Algorithm 5, the topology of the graph is changed. In the graph notation this is easily written down, but in the language of tensor networks, more work has to be done. In Algorithm 5 we also did not consider double edges since we are utilizing only simple graphs.

So for adjusting this algorithm to the tensor network context we have to treat the single edge movement and the double edges in a special manner.

Example 5.1.13. *Consider the graph G as of Figure 5.19 where we are going to eliminate edge $e = \{v_6, v_4\}$. W.r.t. Algorithm 5, we first choose a walk from v_4 to v_6 which does not contain edge $\{v_6, v_4\}$. Two possible walks are*

$$(v_4, v_2, v_1, v_5, v_6)$$

and

$$(v_4, v_3, v_8, v_7, v_6).$$

There are of course also other walks that are possible. Let us choose the green walk to continue with. At first, we define $w_1 := v_4, w_2 := v_2, w_3 := v_1, w_4 := v_5$ and $w_5 := v_6$ such that we look at the walk

$$(w_1, w_2, w_3, w_4, w_5)$$

to be able to use the notation with start*-index and* end*-index just as in line 2 of Algorithm 5. We consequently have* start $= 1$ *and* end $= 5$.

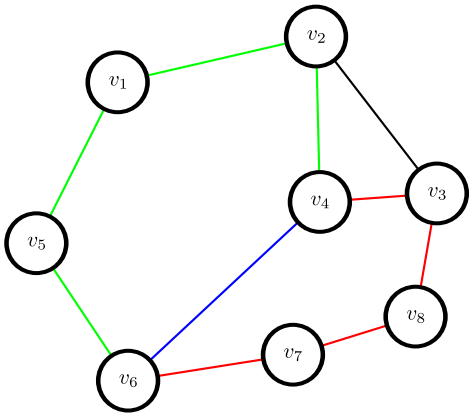

Figure 5.19: Graph with highlighted walks form v_4 to v_6

According to our algorithm, we now choose start $\mapsto$ start $+1$ *such that we get a new edge*

$$e' := \{w_2, w_5\} = \{v_2, v_6\}$$

that we add to the edge set of the graph. The next step is already where the graph topology is changed, i.e.

$$\begin{aligned} E \mapsto \{&\{v_1, v_2\}, \{v_1, v_5\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_2, v_6\}, \\ &\{v_3, v_4\}, \{v_3, v_8\}, \{v_5, v_6\}, \{v_6, v_7\}, \{v_7, v_8\}\} \end{aligned}$$

as of Figure 5.20. The next task is to choose between incrementing start *and decreasing* end *and to continue as before until* start $+1 =$ end.

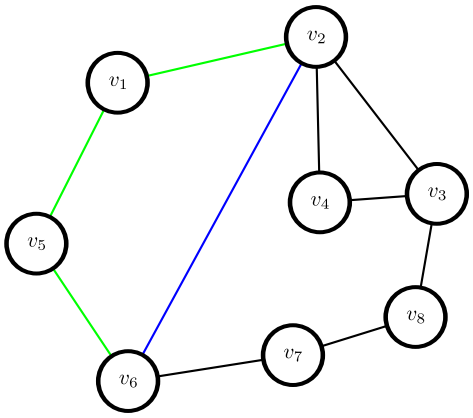

Figure 5.20: Changed topology graph

In our definition of a graph, we did not allow double edges by defining the edge set E as a set of sets. If in Algorithm 5, $start + 1 = end$, the algorithm finishes by adding $\{w_{start}, w_{end}\}$ to the edge set E, which it already contained. In the numerical treatment however we have to do additional work to perform this last step.

At first, we define the set of vertices that are directly connected (i.e. with one edge). From the definition of the adjacency matrix (see [43, Definition 1.1.17]), we deduce

Definition 5.1.14 (Adjacency set of a vertex)**.** *Let $G = (V, E)$ be a simple, undirected and connected multigraph as of Definition 3.1.3, $v \in V$ and $\mathbb{I}$ be the corresponding incidence map of G (see Definition 3.1.5). Then we define*

$$\mathbb{A}_v := \{v' \in V \mid \exists i \in \mathbb{I}(v) : v' \in i\}$$

as the adjacency set of vertex v.

So $\mathbb{A}_v$ is the set of vertices that are connected to vertex v via one edge. The general edge movement algorithm is stated in Algorithm 6.

Remark 5.1.15 (Double edge elimination)**.** *In general, we can ignore double edges since we can interpret a sum over two indices as a sum over one index where this one index has the cardinality of the product of the two indices. In practice, as described in Remark 5.1.2, the edges can be combined by computing a singular value decomposition.*

5.1.5 Error estimate

While shifting an edge, we can introduce an error in line 3 of Algorithm 6 by omitting small singular values of the SVD's result. Doing that, we can represent matrix A by an approximated matrix $\tilde{A}$ where we can control the error $\|A - \tilde{A}\|$ (with $\|\cdot\|$ being a crossnorm as of Definition 4.1.3) with these singular values. The influence on the whole tensor network representation has to be investigated: We first consider the change that is made in the second step of the TC to TT conversion of Subsection 5.1.1 such that the definitions of $v_1, \ldots, v_d, v'_1, v'_2, v'_{d-1}, v'_d$ that we made there are also valid for this subsection. We define

$$v := \sum_{j_1=1}^{\tilde{r}_1} \sum_{j_2,\ldots,j_d=1}^{r_2,\ldots,r_d} v'_1(j_1) \otimes v'_2(j_1, j_2, j_d) \otimes v_3(j_2, j_3) \otimes \ldots \otimes v_d(j_{d-1}, j_d)$$

and

$$\begin{aligned}\tilde{v} := \sum_{j_1,j_{d-1}=1}^{\tilde{r}_1,\tilde{r}_{d-1}} \sum_{j_2,\ldots,j_{d-2},j_d=1}^{r_2,\ldots,r_{d-2},r_d} & v'_1(j_1) \otimes v'_2(j_1, j_2, j_d) \otimes v_3(j_2, j_3) \otimes \ldots \\ & \otimes v_{d-2}(j_{d-3}, j_{d-2}) \otimes v'_{d-1}(j_{d-2}, j_{d-1}, j_d) \otimes v'_d(j_{d-1}),\end{aligned}$$

such that we have to estimate $\|v - \tilde{v}\|$ where $\tilde{v}$ is the tensor representation after the (approximated) edge shift. To shorten the notation, we introduce

$$\hat{v}(j_d, j_{d-2}) := \sum_{j_1=1}^{\tilde{r}_1} \sum_{j_2,\ldots,j_{d-3}=1}^{r_2,\ldots,r_{d-3}} v'_1(j_1) \otimes v'_2(j_1, j_2, j_d) \otimes v_3(j_2, j_3) \otimes \ldots \otimes v_{d-2}(j_{d-3}, j_{d-2})$$

Algorithm 6 Single edge movement (tensor networks)

Let $d \in \mathbb{N}$ and $G = (V, E)$ be a tensor network graph of order $(d, L) \in \mathbb{N} \times \mathbb{N}_0$ with $\ell := \#E$ and representation rank $(r_1, \dots, r_\ell)$ which represents a tensor $v \in \mathcal{V} = \bigotimes_{\mu=1}^{d}$ with $V = \{v_1, \dots, v_{d+L}\}$. Let further $\mathbb{I}$ be the incidence map and g be the degree map of G (see Definition 3.1.5).

We choose $a, b, c \in \{1, \dots, d + L\}$ such that $a \neq b, b \neq c$ and $a \neq c$ and $\{v_a, v_b\}, \{v_b, v_c\} \in E, \{v_a, v_c\} \notin E$. We define $e_0 := \{v_b, v_c\}, e := \{v_a, v_b\}$ and $e' := \{v_a, v_c\}$ (remember that v_z represents a mapping into $\mathcal{V}_z$ for $z \in \{1, \dots, d\}$ and into $\mathbb{K}$ for $z \in \{d+1, \dots, L\}$). The goal is to *replace* e in G by e' and still represent tensor v (the representation rank may differ).

1: w.l.o.g., we assume $\mathbb{I}(v_b) = \{1, \dots, \ell_0\}, \mathbb{I}(v_c) = \{\ell_0, \dots, \ell_1\}$ and $\ell_0 - 1 \in \mathbb{I}(v_a)$ with $2 < \ell_0 < \ell_1 < \ell$ such that $e_0 \hat{=} \ell_0 - 1$ and $e \hat{=} \ell_0$

2: define

$$A := \left(\left(\sum_{j_{\ell_0}=1}^{r_{\ell_0}} \left(v_b(j_1, \dots, j_{\ell_0}) \otimes v_c(j_{\ell_0}, \dots, j_{\ell_1})\right)_{x,y} \right)_{(x, j_1, \dots, j_{\ell_0-2}), (y, j_{\ell_0-1}, j_{\ell_0+1}, \dots, j_{\ell_1})} \right)$$

as a $\mathbb{K}$ valued $n_b \cdot \prod_{\mu \in \mathbb{I}(v_b) \setminus \{\ell_0 - 1, \ell_0\}} r_\mu \times n_c \cdot \prod_{\mu \in (\mathbb{I}(v_c) \setminus \{\ell_0\}) \cup \{\ell_0 - 1\}} r_\mu$ matrix where $n_b := \dim \mathcal{V}_b$ and $n_c := \dim \mathcal{V}_c$

3: compute a SVD of A such that

$$A = \left(\left(\sum_{j_{\ell_0}=1}^{\tilde{r}_{\ell_0}} \left(v'_b(j_1, \dots, j_{\ell_0-2}, j_{\ell_0}) \otimes v'_c(j_{\ell_0-1}, j_{\ell_0}, \dots, j_{\ell_1})\right)_{x,y} \right)_{(x, j_1, \dots, j_{\ell_0-2}), (y, j_{\ell_0-1}, j_{\ell_0+1}, \dots, j_{\ell_1})} \right)$$

where $\tilde{r}_{\ell_0}$ is the matrix rank of A

4: consequently we have obtained two mappings

$$v'_b : \mathbb{N}^{g(v_b)-1} \to \mathcal{V}_b \qquad \text{and} \qquad v'_c : \mathbb{N}^{g(v_c)+1} \to \mathcal{V}_c$$

5: replace v_b by v'_b and v_c by v'_c in V and all elements of E

6: define

$$G' := (V, E')$$

with

$$E' := (E \setminus \{e\}) \cup \{\{v_a, v'_c\}\}$$

Now, we have created a tensor representation with representation rank $(r_1, \dots, r_{\ell_0-1}, \tilde{r}_{\ell_0}, r_{\ell_0+1}, \dots, r_\ell)$ and underlying tensor network graph G', which also represents the tensor v.

and

$$B(j_d, j_{d-2}) := \sum_{j_{d-1}=1}^{r_{d-1}} v_{d-1}(j_{d-2}, j_{d-1}) \otimes v_d(j_{d-1}, j_d) - \sum_{j_{d-1}=1}^{\tilde{r}_{d-1}} v'_{d-1}(j_{d-2}, j_{d-1}, j_d) \otimes v'_d(j_{d-1}).$$

This leads into the following estimate

$$\begin{aligned}
\|v - \tilde{v}\| &= \left\| \sum_{j_d, j_{d-2}=1}^{r_d, r_{d-2}} \hat{v}(j_d, j_{d-2}) \otimes B(j_d, j_{d-2}) \right\| \\
&\leqslant \sum_{j_d, j_{d-2}=1}^{r_d, r_{d-2}} \|\hat{v}(j_d, j_{d-2})\| \cdot \|B(j_d, j_{d-2})\| \\
&\leqslant \left(\sum_{j_d, j_{d-2}=1}^{r_d, r_{d-2}} \|\hat{v}(j_d, j_{d-2})\|^2 \right)^{\frac{1}{2}} \left(\sum_{j_d, j_{d-2}=1}^{r_d, r_{d-2}} \|B(j_d, j_{d-2})\|^2 \right)^{\frac{1}{2}} \\
&= \left(\sum_{j_d, j_{d-2}=1}^{r_d, r_{d-2}} \|\hat{v}(j_d, j_{d-2})\|^2 \right)^{\frac{1}{2}} \cdot \|A - \tilde{A}\|
\end{aligned}$$

with the help of the triangle inequality and the Cauchy-Schwarz-inequality (in that order) and the crossnorm property. This gives us a precise estimate on when we are allowed to cut off singular values while still maintaining a certain error bound for $\|v - \tilde{v}\|$. There is a severe issue due to the non-closedness of the Tensor Chain format that we will explain after the general error estimate.

This error estimate can be easily generalized to other tensor representations. We want to state the proper error estimate for general tensor networks as well.

We will use the *simplified notation* as of Definition 3.2.8 for the represented tensor. We are going to write down everything explicitly without using abbreviations. As before, we are utilizing the Cauchy-Schwarz-inequality and the crossnorm property. Let the

notation of Algorithm 6 hold with the extension that w.l.o.g. $c = b+1$, such that we get

$$\begin{aligned}
\|v - \tilde{v}\| = \Bigg\| & \sum_{j_1,\dots,j_\ell=1}^{r_1,\dots,r_\ell} \bigotimes_{\mu=1}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \\
& - \sum_{j_1,\dots,j_\ell=1}^{r_1,\dots,r_{\ell_0-1},\tilde{r}_{\ell_0},r_{\ell_0+1},\dots,r_\ell} \bigotimes_{\mu=1}^{b-1} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \\
& \otimes v'_b(j_1,\dots,j_{\ell_0-2},j_{\ell_0}) \otimes v'_c(j_{\ell_0-1},j_{\ell_0},\dots,j_{\ell_1}) \otimes \\
& \bigotimes_{\mu=c+1}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \Bigg\| \\
= \Bigg\| & \sum_{j_1,\dots,j_{\ell_0-1},j_{\ell_0+1},\dots,j_{\ell_1}=1}^{r_1,\dots,r_{\ell_0-1},r_{\ell_0},\dots,r_{\ell_1}} \left(\sum_{j_{\ell_1+1},\dots,j_\ell,j_{\ell_0}=1}^{r_{\ell_1+1},\dots,r_\ell,1} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{b,c\}}}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \right) \\
& \otimes \left(\sum_{j_{\ell_0}=1}^{r_{\ell_0}} v_b(j_1,\dots,j_{\ell_0-1},j_{\ell_0}) \otimes v_c(j_{\ell_0},\dots,j_{\ell_1}) \right. \\
& \left. - \sum_{j_{\ell_0}=1}^{\tilde{r}_{\ell_0}} v'_b(j_1,\dots,j_{\ell_0-2},j_{\ell_0}) \otimes v'_c(j_{\ell_0-1},j_{\ell_0},\dots,j_{\ell_1}) \right) \Bigg\| \\
\leqslant & \sum_{j_1,\dots,j_{\ell_0-1},j_{\ell_0+1},\dots,j_{\ell_1}=1}^{r_1,\dots,r_{\ell_0-1},r_{\ell_0+1},\dots,r_{\ell_0}} \left\| \sum_{j_{\ell_1+1},\dots,j_\ell,j_{\ell_0}=1}^{r_{\ell_1+1},\dots,r_\ell,1} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{b,c\}}}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \right\| \\
& \cdot \left\| \sum_{j_{\ell_0}=1}^{r_{\ell_0}} v_b(j_1,\dots,j_{\ell_0-1},j_{\ell_0}) \otimes v_c(j_{\ell_0},\dots,j_{\ell_1}) \right. \\
& \left. - \sum_{j_{\ell_0}=1}^{\tilde{r}_{\ell_0}} v'_b(j_1,\dots,j_{\ell_0-2},j_{\ell_0}) \otimes v'_c(j_{\ell_0-1},j_{\ell_0},\dots,j_{\ell_1}) \right\| \\
\leqslant & \left(\sum_{j_1,\dots,j_{\ell_0-1},j_{\ell_0+1},\dots,j_{\ell_1}=1}^{r_1,\dots,r_{\ell_0-1},r_{\ell_0+1},\dots,r_{\ell_1}} \left\| \sum_{j_{\ell_1+1},\dots,j_\ell,j_{\ell_0}=1}^{r_{\ell_1+1},\dots,r_\ell,1} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{b,c\}}}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \right\|^2 \right)^{\frac{1}{2}} \\
& \cdot \left(\sum_{j_1,\dots,j_{\ell_0-1},j_{\ell_0+1},\dots,j_{\ell_1}=1}^{r_1,\dots,r_{\ell_0-1},r_{\ell_0+1},\dots,r_{\ell_1}} \left\| \sum_{j_{\ell_0}=1}^{r_{\ell_0}} v_b(j_1,\dots,j_{\ell_0-1},j_{\ell_0}) \otimes v_c(j_{\ell_0},\dots,j_{\ell_1}) \right. \right. \\
& \left. \left. - \sum_{j_{\ell_0}=1}^{\tilde{r}_{\ell_0}} v'_b(j_1,\dots,j_{\ell_0-2},j_{\ell_0}) \otimes v'_c(j_{\ell_0-1},j_{\ell_0},\dots,j_{\ell_1}) \right\|^2 \right)^{\frac{1}{2}} \\
= & \left(\sum_{j_1,\dots,j_{\ell_0-1},j_{\ell_0+1},\dots,j_{\ell_1}=1}^{r_1,\dots,r_{\ell_0-1},r_{\ell_0+1},\dots,r_{\ell_1}} \left\| \sum_{j_{\ell_1+1},\dots,j_\ell,j_{\ell_0}=1}^{r_{\ell_1+1},\dots,r_\ell,1} \bigotimes_{\substack{\mu=1 \\ \mu\notin\{b,c\}}}^{d+L} v_\mu(\mathcal{I}(\mu,(j_1,\dots,j_\ell))) \right\|^2 \right)^{\frac{1}{2}} \\
& \|A - \tilde{A}\|,
\end{aligned}$$

which also matches the error estimate for the TC to TT conversion from before. We want to emphasize again the importance of Remark 4.1.4. That is, we have to add constraints to the first factor of the above written product in order to preserve the error bound.

5.1.6 Alternative approaches

The proposed algorithm is of course not the only way to convert an arbitrary tensor network into a tensor tree network. For example, one could also evaluate the tensor network to obtain the full tensor and perform the Vidal decomposition (see [44, 45]) in order to obtain a tensor in the Tensor Train format. Another possibility is to decompose the full tensor with a high order SVD (HOSVD, see [46]) into a hierarchically formated tensor (see [15]). Evaluating all entries of a full tensor for large d however is in general not feasible due to the large number of elements of an order d tensor (which is exponential in d).

Another general approach is to fix the resulting format and use approximation algorithms such as ALS or DMRG (both are non-linear block Gauss-Seidel methods, see Section 4.1). This however is no direct conversion, but an approximation that has certain convergence rates.

5.2 Outlook on black-box construction

In [30, Section 5], a sketch of the description of an algorithm that may be used to create tensor representations in arbitrary formats out of black-box data. This approach is based on the algorithm that has been proposed in the same work ([30]) which describes how to create a tensor chain formatted tensor with the help of cross approximations out of a given black-box tensor.

Chapter 6

Open problems

In the treatment of tensor networks many different problems appear. We already named a few in the prior chapters. In this chapter, we want to describe the general problems and try to give approaches for overcoming the difficulties.

6.1 Efficient contraction

Contracting (i.e. performing the summations) a complex tensor network representation is also a hard task that requires huge computational effort. For some cases, we can state an optimal complexity about for the contraction (such as for the CP, TT, TC, Hierarchical and the Tucker format). Using more and more complex network structures (such as the PEPS representation, see Definition 2.1.14), the time that is needed to perform an **exact** contraction can increase exponentially with respect to the order of the tensor.

As mentioned before, [42] states a contraction algorithm for the PEPS format that results in an **approximated** contraction. This may be sufficient for certain applications but it may lead to an inappropriate error in some cases.

6.2 Finding the best network structure

In order to define what the *best* tensor network structure is, the purpose has to be stated in advance. For some applications it might be sufficient to reduce the storage cost for storing the tensor representation on a hard disk whereas for other applications, it may be important how fast certain operations can be performed (such as evaluating one single entry or computing the inner product).

In practice, something about the structure of the data is known and we can derive an *appropriate* underlying tensor network from it. However, there might be *better* (w.r.t. to the purpose) tensor network formats for a specific class of problems that do not originate from the data structure directly. The results may also differ from to be represented tensor to to be represented tensor.

6.3 Acting on single terms

As stated earlier, when performing a successive rank increment of a tensor network representation, we have - based on the structure of the network - a very restricted choice of arguments. This leads in general to adding terms that have a very small norm such that the approximation algorithm is unstable. This problem occurred Subsection 4.1.2 where we tried to optimize single terms and also in Section 4.2 where we created an initial guess by successively adding rank one tensors.

6.4 Unknown necessary information to recover a tensor representation

Another open question is what information is needed to perfectly recover a tensor network out of a full tensor that is defined by an unknown tensor representation.

In [30, Section 4] for instance, a given (artificial) full tensor that has been generated out of a Tensor Chain representation, could be perfectly recovered using the SVD. The important piece of information in that case was the ordering of the first edge split as it has been also done in in the first step of Subsection 5.1.3. This ordering problem has been also mentioned in [39].

Chapter 7

Tensor formats beyond tensor networks

By using the tensor network approach, we can cover a lot of currently used tensor formats. On the contrary, it is clear that there exist also other formats that do not match the described structure.

In this chapter we will describe the motivation of selected different approaches and describe some advantages and disadvantages of the with respect to the tensor network approach.

7.1 Toeplitz-like tensor representation

An idea by Aram Khachatryan, that has been communicated during discussions, is a method for handling certain breakage kernel functions, that have singularities on the diagonal, efficiently. We will describe the idea in this section.

Consider a matrix $(a_{ij}) = A \in \mathbb{R}^{n \times n}$ with $n \in \mathbb{N}$ where

$$a_{ij} := \begin{cases} \sqrt{\frac{i-j}{n}} + \sqrt{\frac{i}{n}} & \text{for } i > j \\ 0 & \text{else} \end{cases}$$

such that A has singularities on the diagonal. This is a discretization of the function

$$\begin{aligned} f : (0,1] \times (0,1] &\to [0,1], \\ (x,y) &\mapsto \begin{cases} \sqrt{x-y} + \sqrt{x} & \text{if } x > y \\ 0 & \text{else} \end{cases} \end{aligned}$$

on the interval $(0,1] \times (0,1] \subset \mathbb{R} \times \mathbb{R}$ with n grid points in each direction. Due to Remark 3.4.1, we only have one choice of representing this function in the tensor network framework by

$$f(x,y) \approx \sum_{i=1}^{r} f_1^{(i)}(x) f_2^{(i)}(y)$$

where $f_1^{(i)}, f_2^{(i)} : \mathbb{R} \to \mathbb{R}$ for $i \in \{1, \dots, n\}$, which usually yields in a large rank r . The idea by Aram Khachatryan is to use a slightly different approach that adds some focus on the diagonal (which is important for handling singularities there) by adding an additional factor to each term, such that

$$f(x,y) \approx \sum_{i=1}^{\tilde{r}} \tilde{f}_1^{(i)}(x) \tilde{f}_2^{(i)}(y) \tilde{f}_3^{(i)}(x-y),$$

where $\tilde{f}_j^{(i)} : \mathbb{R} \to \mathbb{R}$ for $i \in \{1, \dots, \tilde{r}\}$ and $j \in \{1,2,3\}$. Having such a structure, we can represent f by

$$f(x,y) = \underbrace{1}_{\tilde{f}_1^{(1)}(x)} \cdot \underbrace{1}_{\tilde{f}_2^{(1)}(y)} \cdot \underbrace{\sqrt{x-y}}_{\tilde{f}_3^{(1)}(x-y)} + \underbrace{\sqrt{x}}_{\tilde{f}_1^{(2)}(x)} \cdot \underbrace{1}_{\tilde{f}_2^{(2)}(y)} \cdot \underbrace{1}_{\tilde{f}_3^{(2)}(x-y)},$$

so $\tilde{r} = 2$ and we obtain an exact representation for $f(x,y)$.

Of course, this is a constructed example but it actually reflects a practical use. Breakage kernels for instance do have exactly such a structure and can be therefore well approximated using the modified approach. Also for computing the Hamiltonian of the Schrödinger Equation, this structure appears, which leads to an efficient numerical treatment.

7.2 Sparse tensors

Consider having an order d tensor $t \in \mathcal{V}_1 \otimes \dots \otimes \mathcal{V}_d = \mathcal{V}$ given as before such that we can address each entry

$$t_i$$

for each $i \in \bigtimes_{\mu=1}^{d} \{1, \dots, \dim \mathcal{V}_\mu\} =: \mathcal{I}_0$. The tensor is called a *sparse tensor* if

$$t_i = 0$$

for a vast majority of the indices $i \in \mathcal{I}_0$. The term *sparse* arises from Sparse Matrices where we have a grid that has only very few non-zero entries. See [47] for an overview. We further define

$$\mathcal{I} := \{i \in \mathcal{I}_0 : t_i \neq 0\}$$

such that t is uniquely determined by the index set $\mathcal{I}$ and the corresponding values of t, compare [5, Section 7.2]. In practice this means a reduction in terms of storage that is needed to exactly store t. Such a representation is not covered by the tensor network approach as the index set $\mathcal{I}$ is in general arbitrarily distributed amongst $\mathcal{I}_0$.

In [48, Section 5], the sparse tensor is converted into the r-term format. The naive approach would be to put each non-zero entry of t into a separate term such that

$$t = \sum_{i \in \mathcal{I}} t_i \bigotimes_{\mu=1}^{d} e_{\mu, i_\mu}$$

where e_{μ,i_μ} is the i_μth unit basis vector of $\mathcal{V}_\mu$. This would lead the rank of t in the r-term format to be equal to $\#\mathcal{I}$. Using the full tensor to r-term tensor conversion as described in [5, Subsection 7.6.1], the representation rank would be at most

$$\frac{\prod_{\mu=1}^{d} \dim \mathcal{V}_\mu}{\max\limits_{\mu=1}^{d} \dim \mathcal{V}_\mu}.$$

However, [48, Section 5] describes a possibility to obtain a lower representation rank for the r-term format (exact representation of the sparse tensor).

It highly depends on the application whether the representation in the r-term format is of any use. Representing a *sparse tensor* in a tensor format that fits in the tensor network framework however, frees the data from problems that arise from the sparsity (i.e. one can treat the tensor without paying attention to the original sparsity properties).

Summary and conclusions

In this work, we described a framework for handling some of the most commonly used tensor formats in dealing with high dimensional data. We have shown that by using the tensor network framework, we can describe efficient algorithms in an abstract way. We have described the approximation algorithms ALS and DMRG for arbitrary tensor networks and explained, where the instability of formats that contain a cycle may lead to problems.

The problems that have been pointed out throughout this thesis also show the limits of the above mentioned approach. It is usually recommended to use a tensor format that is well applicable to the problem that one wants to address. Unfortunately, we cannot provide a solution as to how one may exactly evaluate a tensor representation with a very complex structure (like the described PEPS format) efficiently. It is still a problem to treat tensor representations that have an instable format.

Another contribution of this thesis is the general conversion of different tensor formats in the context of tensor networks. We are able to efficiently change the underlying topology of a given tensor representation while using the similarities (if present) of both the original and the desired structure. This is an important feature if only minor structural changes are required.

With the help of different numerical experiments we were able to justify the new approach and show its flexibility.

We introduced a method to successively create an initial guess that improves some approximation results. This algorithm is based on successive rank 1 increments for the r-term format.

There are still open questions about how to find the optimal tensor format for a given general problem (e.g. storage, operations, etc.). For instance in the case where a physical background is given, we might have a clue as to how a *good* structure may look like. There is however, no guarantee that a *better* (with respect to the problem) representation structure does not exist.

Appendix A

Implementation

Describing and defining a framework for representing tensors in the form of tensor networks efficiently with respect to the mathematical notation is one important point of this thesis. However, it is also very important to design the program code efficiently with respect to the algorithms that are supposed to be implemented.

A.1 Data structures

The data structure to represent a tensor network formatted tensor has been developed together with Mike Espig and Philipp Wähnert at the Max-Planck-Institute for Mathematics in the Sciences in Leipzig, Germany (see [38] for the source code).

First of all, one has to think about how to store a d-dimensional object. One way to start this development is to store each dimension separately. Once we have made this substantial decision, we can think about how to store a single *dimension* of the tensor representation. When we look at the definition, of a mapping v_μ in Definition 3.2.3, we see that we can define this mapping by all possible argument-value-pairs. That is for each function argument, we store the function result. Since the definition requires the support of the mappings to be finite, this procedure leads to a finite number of to be stored pairs.

In [4, Notation 5.1.1] a block structure is defined that orders the values lexicographically with respect to the index and the dimension. Since we store each dimensions separately, we order the vectors lexicographically with respect to the index. For instance consider a TC tensor representation of order d

$$v = \sum_{j_1,\dots,j_d=1}^{r_1,\dots,r_d} v_1(j_1, j_d) \otimes \bigotimes_{\mu=2}^{d} v_\mu(j_{\mu-1}, j_\mu) \in \bigotimes_{\mu=1}^{d} \mathcal{V}_\mu$$

with representation rank $(r_1, \dots, r_d) = (2, \dots, 2) \in \mathbb{N}^d$ as of Definition 2.1.11 (i.e. $v_1, \dots, v_d$ is defined as before). In this case, we store v_2 similarly to [4, Equation (5.6)]

TensorRepresentation<T>
d: int # comonentDimensions: std::vector<int> # summations: std::vector<int> # v: std::vector<std::vector<T>> - incidenceMatrix: std::vector<std::vector<int>> - L: int - w: std::vector<std::vector<T>>

Figure A.1: Datastructure

as follows:

$$v_2 \cong \begin{pmatrix} v_2(1,1)_1 \\ v_2(1,1)_2 \\ \vdots \\ v_2(1,1)_n \\ v_2(2,1)_1 \\ \vdots \\ v_2(2,1)_n \\ \vdots \\ v_2(r_1,r_2)_1 \\ \vdots \\ v_2(r_1,r_2)_n \end{pmatrix} \tag{A.1}$$

where each $\dim \mathcal{V}_2 = n, v_2(\cdot,\cdot) \in \mathcal{V}_2$ and $v_2(x,y)_i$ is the i-th entry of the vector $v_2(x,y)$.

Since each dimension may be *connected* to other dimensions (in the graph-sense), we also have to store the information about the underlying graph. As we are dealing with multi-graphs (see Definition 3.1.3), we cannot use a adjacency matrix. Analogously to the definition of the tensor format (see Definition 3.2.3), we will use an incidence matrix stored sparsely (i.e. for each dimension, we only store what vertices it is connected to).

In our definition, we distinguish between vector space vertices and scalar vertices. The same distinction can be found in the implementation. That is, we store the data that represents $v_1, \dots, v_d$ in a different field than the data that represents $w_1, \dots, w_L$.

Altogether, we end up with data structure as of Figure A.1. Where the component dimensions are the dimensions of the vector spaces $\mathcal{V}_1, \dots, \mathcal{V}_d$.

Storing the data as of Equation (A.1) leads to some computational advantages. For instance if the Gramian matrix is has to be computed, this can be done by one simple matrix matrix multiplication since the store for the μth dimension can be interpreted as a matrix with columns in $\mathcal{V}_\mu$ with:
We interpret v_2 as $n \times r_1 \cdot r_2$ matrix

$$v_2 \cong \begin{pmatrix} v_2(1,1)_1 & v_2(2,1)_1 & \dots & v_2(r_1,r_2) \\ \vdots & \vdots & & \vdots \\ v_2(1,1)_n & v_2(2,1)_n & \dots & v_2(r_1,r_2)_n \end{pmatrix} := V_2, \tag{A.2}$$

such that we can compute all $\langle v_2(\cdot,\cdot), v_2(\cdot,\cdot)\rangle$ in one step by computing $V_2^T \cdot V_2 \in \mathbb{K}^{r_1 \cdot r_2 \times r_1 \cdot r_2}$.

A.2 Special format implementations

Analogously to the general tensor representation definition (Definition 3.2.5), we have a general implementation that covers all tensors that are representable by a tensor network. However, the generalization is done at the price of computational effort and therefore speed.

To avoid this in practice severe problem, there are special implementations for several tensor formats that override crucial methods such as ALS and the computation of a single entry. Consequently, we have specialized implementations for ALS (Subsection 4.1.2) and DMRG (Subsection 4.1.3) for the Tensor Chain format (and therefore also for the Tensor Train format). According to [1, Lemma 7.2] the matrices $\mathbf{A}_{[\mu]}$ and $\mathbf{V}_{[\mu]}$ of Equation (4.9) (that are needed to perform an ALS step) can be computed by reshaping a product of reshaped Gramian matrices. The ring structure of the Tensor Chain format also allows us to utilize the prephase as described in Subsection 4.1.2 in order to reduce the computational complexity.

A.3 Class hierarchy

Since we use C++ for implementing the algorithms, we utilize derived classes. We can stick to the theoretical inheritance, e.g. *every Tensor Train representation is a Tensor Chain representation.* This leads to a class structure a of Figure A.2.

Figure A.2: Class inheritance

A.4 Helper classes

Based on an idea by Mike Espig, we generalized a summation index to a multi-index. Since the number of summations is equal to the number of edges of the underlying graph, we need a flexible summation index.

We also added convenient methods to increment the index by one, convert the index to a single integer value and to fix certain components of the index while incrementing

the other components. The index class (see Figure A.3) also increments its components lexicographically.

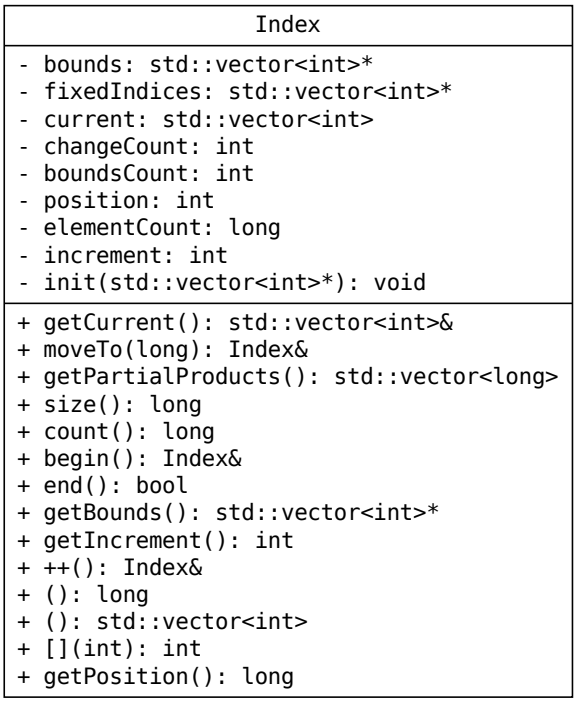

Figure A.3: Multi index UML diagram

A.5 Source code access

At the time of the submission of this thesis, the source code of the implementation that has been described in this chapter, is supposed to be available at [38]. The reader is warned that the availability and/or the location of the source code may change at any time.

A.6 Overview over implemented experiments

Summarizing all implemented experiments that we used in this thesis, we get:

- Performing ALS approximation in the Tensor Chain format
- Converting a Tensor Chain representation to a Tensor Train representation and vice versa
- Creating an initial guess in the Tensor Chain format by successively increasing the representation rank

- Creating a Tensor Chain representation out of given black-box data; this experiment is also used in [30]

The implementation is done to show how one can perform computations with arbitrary tensor networks. We had to limit the experiments to specific formats to be able to compute with larger d. Therefore, the code may be understood as an example on how to treat tensor networks in practice.

List of Algorithms

List of Figures

List of Tables

Bibliography

[1] M. Espig, W. Hackbusch, S. Handschuh, and R. Schneider. Optimization Problems in Contracted Tensor Networks. *Computing and Visualization in Science*, 14(6):271–285, 2011. doi:10.1007/s00791-012-0183-y.

[2] T. Yokonuma. *Tensor Spaces and Exterior Algebra*, volume 108 of *Translations of Mathematical Monographs*. American Mathematical Society, 1992. Originally published in Japanese by Iwanami Shoten, Publishers, Tokyo in 1977.

[3] W. H. Greub. *Multilinear Algebra*, volume 136 of *Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen*. Springer-Verlag Berlin Heidelberg New York, 1967.

[4] M. Espig. *Effiziente Bestapproximation mittels Summen von Elementartensoren in hohen Dimensionen*. PhD thesis, University of Leipzig, Faculty of Mathematics & Computer Science, 2008. In German.

[5] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer, 2012.

[6] S. Kühn. *Hierarchische Tensordarstellung*. PhD thesis, University of Leipzig, Faculty of Mathematics & Computer Science, 2012. In German.

[7] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *arXiv:1302.7121 [math.NA]*, 2013. arXiv:1302.7121.

[8] R. Penrose. Applications of Negative Dimensional Tensors. *Combinatorial mathematics and its applications*, 1971.

[9] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6:164–189, 1927.

[10] J. D. Carroll and J. J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[11] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "exploratory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[12] V. de Silva and L.-H. Lim. Tensor Rank and the Ill-Posedness of the Best Low-Rank Approximation Problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008. doi:10.1137/06066518X.

[13] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[14] S. Dolgov and B. N. Khoromskij. Two-Level QTT-Tucker Format for Optimized Tensor Calculus. *SIAM Journal on Matrix Analysis and Applications*, 34(2):593–623, 2013. doi:10.1137/120882597.

[15] W. Hackbusch and S. Kühn. A New Scheme for the Tensor Representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722, 2009. doi:10.1007/s00041-009-9094-9.

[16] Y. Y. Shi, L. M. Duam, and G. Vidal. Classical simulation of quantum many-body systems with a tree tensor network. *Physiscal Review A*, 74:022320, 2006. doi:10.1103/PhysRevA.74.022320.

[17] I. V. Oseledets. A new tensor decomposition. *Doklady Mathematics*, 80(1):495–496, 2009. doi:10.1134/S1064562409040115.

[18] I. V. Oseldets and E. E. Tyrtyshnikov. Tensor tree decomposition does not need a Tree. *INM RAS Preprint*, 08, 2009.

[19] I. V. Oseledets. Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011. doi:10.1137/090752286.

[20] A. Klümper, A. Schadschneider, and J. Zittartz. Equivalence and solution of anisotropic spin-1 models and generalized t-J fermion models in one dimension. *Journal of Physics A*, 24(16):L955, 1991. doi:10.1088/0305-4470/24/16/012.

[21] B. N. Khoromskij. $\mathcal{O}(d \log N)$-Quantics Approximation of $N - d$ Tensors in High-Dimensional Numerical Modeling. *Constructive Approximation*, 34:1–24, 2010. doi:10.1007/s00365-011-9131-1.

[22] B. N. Khoromskij and I. V. Oseledets. Quantics-TT collocation approximation of parameter-dependet and stochastic elliptic PDEs. *Computational Methods in Applied Mathematics*, 10(4):376–394, 2010. doi:10.2478/cmam-2010-0023.

[23] I. V. Oseledets. Approximation of $2^d \times 2^d$ Matrices Using Tensor Decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010. doi:10.1137/090757861.

[24] V. Khoromskaia, B. N. Khoromskij, and R. Schneider. QTT Representation of the Hartree and Exchange Operators in Electronic Structure Calculations. *Computational Methods in Applied Mathematics*, 11(3):327–341, 2011. doi:10.2478/cmam-2011-0018.

[25] J. M. Landsberg, Y. Qi, and K. Ye. On the geometry of tensor network states. *arXiv:1105.4449 [math.AG]*, 2011. arXiv:1105.4449.

[26] J. M. Landsberg. *Tensors: Geometry and Applications*, volume 128 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, Rhode Island, 2012.

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

[28] 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):143–224, 2008. doi:10.1080/14789940801912366.

[29] V. Murg, F. Verstraete, and J. I. Cirac. Exploring frustrated spin–systems using Projected Entangled Pair States (PEPS). *Physical Review B*, 79(19), 2009. doi:10.1103/PhysRevB.79.195119.

[30] M. Espig, K. K. Naraparaju, and J. Schneider. A note on approximation in Tensor Chain format. *MIS Preprint*, 16, 2012. revised version from September 2013. URL: http://www.mis.mpg.de/de/publications/preprints/2012/2012-16.html.

[31] J. L. Gross and J. Yellen. *Graph Theory and Its Applications*. Discrete Mathematics and its Applications. Chapman & Hall/CRC Press, second edition, 2006.

[32] L. Grasedyck and W. Hackbusch. An Introduction to Hierarchical ($\mathcal{H}$-) Rank and TT-Rank of Tensors with Example. *Computational Methods in Applied Mathematics*, 11(3):291–304, 2011.

[33] E. G. Hohenstein, R. M. Parrish, and T. J. Martínez. Tensor hypercontraction density fitting. I. Quadratic scaling second- and third-order Møller-Plesset pertubation theory. *The Journal of Chemical Physics*, 137(4), 044103 (2012). doi:10.1063/1.4732310.

[34] M. Espig, H. Auer, W. Hackbusch, U. Benedikt, and A. Auer. Tensor Representation Techniques in post-Hartree Fock Methods: Matrix Product State Tensor Format. *Molecular physics*, 111(16/17):2398–2413, 2013. doi:10.1080/00268976.2013.798433.

[35] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Computer Science and Applied Mathematics / Computer Science and Scientific Computing. Academic Press, Inc., 1970.

[36] S. R. White. Density Matrix Formulation for Quantum Renormalization Groups. *Physical Review Letters*, 69(19):2863–2866, 1992. doi:10.1103/PhysRevLett.69.2863.

[37] S. Holtz, T. Rohwedder, and R. Schneider. The Alternating Linear Scheme for Tensor Optimisation in the Tensor Train Format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012. doi:10.1137/100818893.

[38] M. Espig, M. Schuster, A. Killaitis, N. Waldren, P. Wähnert, S. Handschuh, and H. Auer. TensorCalculus library, 2008–2014. URL: https://gitorious.org/tensorcalculus.

[39] S. Handschuh. Changing the topology of Tensor Networks. *MIS Preprint*, 15, 2012. URL: http://www.mis.mpg.de/de/publications/preprints/2012/2012-15.html.

[40] I. V. Oseledets. Compact matrix form of the d-dimensional tensor decomposition. *INM RAS Preprint*, 09-01, 2009. URL: http://spring.inm.ras.ru/osel/?p=15.

[41] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, third edition, 1996.

[42] T. Huckle, K. Waldherr, and T. Schulte-Herbrüggen. Computations in Quantum Tensor Networks. *Linear Algebra and its Applications*, 438(2):750–781, 2013. doi:10.1016/j.laa.2011.12.019.

[43] D. B. West. *Introduction to Graph Theory*. Prentice-Hall, 2nd edition, 2001.

[44] I. V. Oseledets and E. E. Tyrtyshnikov. Breaking the Curse of Dimensionality, Or How to Use SVD in Many Dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, 2009. doi:10.1137/090748330.

[45] Guifré Vidal. Efficient Classical Simulation of Slightly Entangled Quantum Computations. *Physical Review Letters*, 91(14):147902, 2003. doi:10.1103/PhysRevLett.91.147902.

[46] L. De Lathauwer, B. De Moor, and J. Vandewalle. A Multilinear Singular Value Decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000. doi:10.1137/S0895479896305696.

[47] D. J. Rose and R. A. Willoughby. *Sparse matrices and their applications*. Proceedings of the Symposium on Sparse Matrices and Their Applications, held September 9–10, 1971 at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. Plenum Press, New York, 1972.

[48] W. Hackbusch. The Use of Sparse Grid Approximation for the r-Term Tensor Representation. In *Sparse grids and applications*, volume 88 of *Lecture notes in computational science and engineering*, pages 151–159. Springer, 2013. doi:10.1007/978-3-642-31703-3_7.

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbstständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, 6. Oktober 2014

Daten zum Autor

Persönliche Daten

Name	Stefan Handschuh
Geburtsdatum	13. August 1984

Wissenschaftlicher Werdegang

10/2003–10/2009	**Universität Leipzig** **Diplomstudium Mathematik** Spezialisierung: Wahrscheinlichkeitstheorie Diplomarbeit: *Abweichungen von Irrfahrten in zufälliger Szenerie*
seit 04/2010	**Doktorand am Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig**