Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions

Andrzej CICHOCKI
RIKEN Brain Science Institute, Japan
and Systems Research Institute of the Polish Academy of Science, Poland
a.cichocki@riken.jp

Part of this work was presented on the International Workshop on Smart Info-Media Systems in Asia, (invited talk - SISA-2013) Sept.30–Oct.2, 2013, Nagoya, Japan

Abstract—Many problems in computational neuroscience, neuroinformatics, pattern/image recognition, signal processing and machine learning generate massive amounts of multidimensional data with multiple aspects and high dimensionality. Tensors (i.e., multi-way arrays) provide often a natural and compact representation for such massive multidimensional data via suitable low-rank approximations. Big data analytics require novel technologies to efficiently process huge datasets within tolerable elapsed times. Such a new emerging technology for multidimensional big data is a multiway analysis via tensor networks (TNs) and tensor decompositions (TDs) which represent tensors by sets of factor (component) matrices and lower-order (core) tensors. Dynamic tensor analysis allows us to discover meaningful hidden structures of complex data and to perform generalizations by capturing multi-linear and multi-aspect relationships. We will discuss some fundamental TN models, their mathematical and graphical descriptions and associated learning algorithms for large-scale TDs and TNs, with many potential applications including: Anomaly detection, feature extraction, classification, cluster analysis, data fusion and integration, pattern recognition, predictive modeling, regression, time series analysis and multiway component analysis.

Keywords: Large-scale HOSVD, Tensor decompositions, CPD, Tucker models, Hierarchical Tucker (HT) decomposition, low-rank tensor approximations (LRA), Tensorization/Quantization, tensor train (TT/QTT) - Matrix Product States (MPS), Matrix Product Operator (MPO), DMRG, Strong Kronecker Product (SKP).

I. Introduction and Motivations

Big Data consists of multidimensional, multi-modal data-sets that are so huge and complex that they cannot be easily stored or processed by using standard computers. Big data are characterized not only by big Volume but also another specific “V” features (see Fig. 1). High Volume implies the need for algorithms that are scalable; High Velocity address the challenges related to process data in near real-time, or virtually real-time; High Veracity demands robust and predictive algorithms for noisy, incomplete or inconsistent data, and finally, high Variety may require integration across different kind of data, e.g., neuroimages, time series, spiking trains, genetic and behavior data. Many challenging problems for big data are related to capture, manage, search, visualize, cluster, classify, assimilate, merge, and process the data within a tolerable elapsed time, hence demanding new innovative solutions and technologies. Such emerging technology is Tensor Decompositions (TDs) and Tensor Networks (TNs) via low-rank matrix/tensor approximations. The challenge is how to analyze large-scale, multiway data

Figure 1: Big Data analysis for neuroscience recordings. Brain data can be recorded by electroencephalography (EEG), electrocorticography (ECoG), magnetoencephalography (MEG), fMRI, DTI, PET, Multi Unit Recording (MUR). This involves analysis of multiple modalities/multiple subjects neuroimages, spectrograms, time series, genetic and behavior data. One of the challenges in computational and system neuroscience is to make fusion (assimilation) of such data and to understand the multiple relationships among them in such tasks as perception, cognition and social interactions. The our “V”s of big data: Volume - scale of data, Variety - different forms of data, Veracity - uncertainty of data, and Velocity - analysis of streaming data, comprise the challenges ahead of us.
Figure 2: A 3rd-order tensor $X \in \mathbb{R}^{I \times J \times K}$, with entries $x_{i,j,k} = X(i,j,k)$ and its sub-tensors: Slices and fibers. All fibers are treated as column vectors.

Figure 3: Block matrices and their representations by (a) a 3rd-order tensor and (b) a 4th-order tensor.
II. Basic Tensor Operations

A higher-order tensor can be interpreted as a multiway array, as illustrated graphically in Figs. 2, 3 and 4. Our adopted convenience is that tensors are denoted by bold underlined capital letters, e.g., $\mathbf{X} \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_N}$, and that all data are real-valued. The order of a tensor is the number of its “modes”, “ways” or “dimensions”, which can include space, time, frequency, trials, classes, and dictionaries. Matrices (2nd-order tensors) are denoted by boldface capital letters, e.g., $\mathbf{X}$, and vectors (1st-order tensors) by boldface lowercase letters; for instance the columns of the matrix $\mathbf{A} = [a_1, a_2, \ldots, a_R] \in \mathbb{R}^{I \times R}$ are denoted by $a_r$ and elements of a matrix (scalars) are denoted by lowercase letters, e.g., $a_{ir}$ (see Table I).

The most common types of tensor multiplications are denoted by: $\otimes$ for the Kronecker, $\circ$ for the Hadamard (componentwise), $\circ$ for the outer and $\times_n$ for the mode-$n$ products (see Table I).

TNs and TDs can be represented by tensor network diagrams, in which tensors are represented graphically by nodes or any shapes (e.g., circles, spheres, triangular, squares, ellipses) and each outgoing edge (line) emerging from a shape represents a mode (a way, dimension, indices) (see Fig. 4). Tensor network diagrams are very useful not only in visualizing tensor decompositions, but also in their different transformations/reshapings and graphical illustrations of mathematical (multilinear) operations.

It should also be noted that block matrices and hierarchical block matrices can be represented by tensors.
at least two different ways.

**Remark:** The multi–index can be defined using two different conventions:

1. **The little-endian convention**

   \[
   i_1, i_2, \ldots, i_N = i_1 + (i_2 - 1)I_1 + (i_3 - 1)I_1I_2 + \cdots + (i_N - 1)I_1\cdots I_{N-1}. \tag{1}
   \]

2. **The big-endian**

   \[
   i_1, i_2, \ldots, i_N = i_N + (i_{N-1} - 1)I_N + (i_{N-2} - 1)I_NI_{N-1} + \cdots + (i_1 - 1)I_2 \cdots I_N. \tag{2}
   \]

The little–endian notation is consistent with the Fortran style of indexing, while the big–endian notation is similar to numbers written in the positional system and corresponds to reverse lexicographic order. The definition unfolding of tensors and the Kronecker (tensor) product \( \otimes \) should be also consistent with the chosen convention\(^1\). In this paper we will use the big-endian notation, however it is enough to remember that \( c = a \otimes b \) means that \( c_{ij} = aibj \).

The Kronecker product of two tensors: \( A \in \mathbb{R}^{l_1 \times \cdots \times l_N} \) and \( B \in \mathbb{R}^{j_1 \times \cdots \times j_N} \) yields \( C = A \otimes B \in \mathbb{R}^{l_1j_1 \times \cdots \times l_Nj_N} \), with entries \( c_{i_1i_2\ldots i_Nj_1j_2\ldots j_N} = a_{i_1\ldots i_N}b_{j_1\ldots j_N} \), where \( i_n,j_n = j_n + (i_n - 1)j_n \).

The mode-\( n \) product of a tensor \( A \in \mathbb{R}^{l_1 \times \cdots \times l_N} \) by a vector \( b \in \mathbb{R}^{l_n} \) is defined as a tensor \( C = A \times_n b \in \mathbb{R}^{l_1 \times \cdots \times l_{n-1} \times l_{n+1} \times \cdots \times l_N} \), with entries \( c_{i_1i_2\ldots i_{n-1}j_ni_{n+1}\ldots i_N} = \sum_{j_n=1}^{l_n} a_{i_1i_2\ldots i_{n-1}j_ni_{n+1}\ldots i_N}b_{j_n} \). This can also be expressed in a matrix form as \( C_{(n)} = BA_{(n)} \) (see Fig. 8), which allows us to employ fast matrix by vector and matrix by matrix multiplications for very large scale problems.

If we take all the modes, then we have a full multilinear product of a tensor and a set of matrices, which is compactly written as \([4]\) (see Fig. 9 (a)):

\[
C = \mathcal{G} \times_1 B^{(1)} \times_2 B^{(2)} \cdots \times_N B^{(N)} = [G_1 B^{(1)}, B^{(2)}, \ldots, B^{(N)}]. \tag{3}
\]

In a similar way to mode-\( m \) multilinear product, we can define the mode-\( \binom{m}{n} \) product of two tensors (tensor contraction) \( A \in \mathbb{R}^{l_1 \times \cdots \times l_N} \) and

\(^1\)The standard and more popular definition in multilinear algebra assumes the big–endian convention, while for the development of the efficient program code for big data usually the little–endian convention seems to be more convenient (see for more detail papers of Dolgov and Savostyanov [18], [19]).
with entries $c_{i_1,i_2,...,i_{n+1}} = a_{i_1,i_2,...,i_n} b_{j_{n+1}}$, or equivalently $C_{(n)} = A B_{(n)}$.

$C = A \otimes B$ tensor or outer product of $A \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and $B \in R^{J_1 \times J_2 \times \cdots \times J_M}$ yields $(N + M)$th-order tensor $C$, with entries $c_{i_{n+1},j_{n+1},...,j_M}$ = $a_{i_1,i_2,...,i_n} b_{j_{n+1},...,j_M}$.

$X = a \circ b \circ c \in R^{I \times J \times K}$ tensor or outer product of vectors forms a rank-1 tensor, with entries $x_{ijk} = a_i b_j c_k$.

$A^T$, $A^{-1}$, $A^\dagger$ transpose, inverse and Moore-Penrose pseudo-inverse of $A$.

$C = A \circ \otimes B$ Kronecker product of $A \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and $B \in R^{J_1 \times J_2 \times \cdots \times J_M}$ yields $C \in R^{I_1J_1 \times I_2J_2 \times \cdots \times I_NJ_M}$, with entries $c_{i_1,j_1,i_2,j_2,...,i_N,j_M} = a_{i_1,i_2,...,i_N} b_{j_1,j_2,...,j_M}$, where $i_1,j_1,j_2,...,i_N,j_M = (i_1 + (i_2 - 1))I_1$.

$C = A \otimes B$ Kronecker product of $A \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and $B \in R^{J_1 \times J_2 \times \cdots \times J_M}$ with common modes $I_N = J_1$ can be written as

$$C = A \times_{I_1} B = A \times_{J_1} B = A \bullet B \in R^{I_1 \times I_2 \times \cdots \times I_N \times J_2 \times \cdots \times J_M},$$

with entries $c_{i_1,i_2,...,i_N,j_{n+1},...,j_M} = \sum_{i_1} a_{i_1,i_2,...,i_N} b_{j_{n+1},...,j_M}$ (see Fig. 10 (a)). This operation can be considered as a contraction of two tensors in single common mode. Tensors can be contracted in several modes or even in all modes as illustrated in Fig. 10.

If not confusing a super- or sub-index $m,n$ can be neglected. For example, the multilinear product of the tensors $A \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and $B \in R^{J_1 \times J_2 \times \cdots \times J_M}$, with a common modes $I_N = J_1$ can be written as

$$C = A \times_{I_1} B \Longrightarrow C_{(0)} = B A_{(n)}$$

with $B \in R^{I_1 \times I_2 \times \cdots \times I_M}$ with common modes $I_{n+1}$ that yields an $(N + M - 2)$-order tensor $C \in R^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times I_{n+1} \times \cdots \times I_M}$; $C = A \times_{I_1} B$ (4)

$C = A \times_{I_1} B \circ \otimes C \Longrightarrow C = A \times_{I_1} (B \circ \otimes C)$ for $b < c$.

Remark: If we use contraction for more than two tensors the order has to be specified (defined) as follows: $A \times_{b} B \times_{c} C = A \times_{b} (B \times_{d} C)$ for $b < c$.

The outer or tensor product $C = A \circ \otimes B$ of the tensors $A \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and $B \in R^{J_1 \times J_2 \times \cdots \times J_M}$ is the tensor $C \in R^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$ with entries $c_{i_1,...,i_N,j_{n+1},...,j_M} = a_{i_1,...,i_N} b_{j_{n+1},...,j_M}$. Specifically, the outer product of two
nonzero vectors $a \in \mathbb{R}^l$, $b \in \mathbb{R}^l$, and $c \in \mathbb{R}^k$ produces a 3rd-order rank-1 tensor: $X = a \circ b \circ c$, where $a$, $b$, and $c$ are $l$, $l$, and $k$-dimensional vectors, respectively. A tensor $X \in \mathbb{R}^{l \times l \times \cdots \times l}$ is said to be rank-1 if it can be expressed exactly as $X = b_1 \circ b_2 \circ \cdots \circ b_N$, where $b_N \in \mathbb{R}^{l}$ are nonzero vectors.

III. Tensor Networks

A tensor network aims to represent or decompose a higher-order tensor into a set of lower-order tensors (typically, 2nd (matrices) and 3rd-order tensors called cores or components) which are sparsely interconnected. In other words, in contrast to TDs, TNs represent decompositions of the data tensors into a set of sparsely (weakly) interconnected lower-order tensors. Recently, the curse of dimensionality for higher-order tensors has been considerably alleviated or even completely avoided through the concept of tensor networks (TN) [20], [21]. A TN can be represented by a set of nodes interconnected by lines. The lines (leads, branches, edges) connecting tensors between each other correspond to contracted modes, whereas lines that do not go from one tensor to another correspond to open (physical) modes in the TN (see Fig. 11).

An edge connecting two nodes indicates a contraction of the respective tensors in the associated pair of modes as illustrated in Fig. 10. Each free (dangling) edge corresponds to a mode, that is not contracted and, hence, the order of the entire tensor network is given by the number of free edges (called often physical indices). A tensor network may not contain any loops, i.e., any edges connecting a node with itself. Some examples of tensor network diagrams are given in Fig. 11.

If a tensor network is a tree, i.e., it does not contain any cycle, each of its edges splits the modes of the data tensor into two groups, which is related to the suitable matricization of the tensor. If, in such a tree tensor network, all nodes have degree 3 or less, it corresponds to an Hierarchical Tucker (HT) decomposition shown in Fig. 12 (a). The HT decomposition has been first introduced...
in scientific computing by Hackbusch and Kühn and further developed by Grasedyck, Kressner, Tobler and others [7], [22]–[26]. Note that for 6th-order tensor, there are two such tensor networks (see Fig. 12 (b)), and for 10th-order there are 11 possible HT decompositions [24], [25].

A simple approach to reduce the size of core tensors is to apply distributed tensor networks (DTNs), which consists in two kinds of cores (nodes): Internal cores (nodes) which have no free edges and external cores which have free edges representing physical indices of a data tensor as illustrated in Figs. 12 and 13.

The idea in the case of the Tucker model, is that a core tensor is replaced by distributed sparsely interconnected cores of lower-order, resulting in a Hierarchical Tucker (HT) network in which only some cores are connected (associated) directly with factor matrices [7], [22], [23], [26].

For some very high-order data tensors it has been observed that the ranks $R_n$ (internal dimensions of cores) increase rapidly with the order of the tensor and/or with an increasing accuracy of approximation for any choice of tensor network, that is, a tree (including TT and HT decompositions) [25]. For such cases, the Projected Entangled-Pair State (PEPS) or the Multi-scale Entanglement Renormalization Ansatz (MERA) tensor networks can be used. These contain cycles, but have hierarchical structures (see Fig. 13). For the PEPS and MERA TNs the ranks can be kept considerably smaller, at the cost of employing 5th and 4th-order cores and consequently a higher computational complexity w.r.t. tensor contractions. The main advantage of PEPS and MERA is that the size of each core tensor in the internal tensor network structure is usually much smaller than the cores in TT/HT decompositions, so consequently the total number of parameters can be reduced. However, it should be noted that the contraction of the resulting tensor network becomes more difficult when compared to the basic tree structures represented by TT and HT models. This is due to the fact that the PEPS and MERA tensor networks contain loops.

IV. Basic Tensor Decompositions and their Representation via Tensor Networks Diagrams

The main objective of a standard tensor decomposition is to factorize a data tensor into physically interpretable or meaningful factor matrices and a single core tensor which indicates the links between components (vectors of factor matrices) in different modes.

A. Constrained Matrix Factorizations and Decompositions – Two-Way Component Analysis

Two-way Component Analysis (2-way CA) exploits a priori knowledge about different characteristics, features or morphology of components (or source signals) [1], [27] to find the hidden components thorough constrained

---

Figure 12: (a) The standard Tucker decomposition and its transformation into Hierarchical Tucker (HT) model for an 8th-order tensor using interconnected 3rd-order core tensors. (b) Various exemplary structure HT/TT models for different order of data tensors. Green circles indicate factor matrices while red circles indicate cores.
matrix factorizations of the form
\[ X = AB^T + E = \sum_{r=1}^{R} a_r \circ b_r + E = \sum_{r=1}^{R} a_r b_r^T + E, \]  
where the constraints imposed on factor matrices A and/or B include orthogonality, sparsity, statistical independence, nonnegativity or smoothness. The CA can be considered as a bilinear (2-way) factorization, where \( X \in \mathbb{R}^{I \times J} \) is a known matrix of observed data, \( E \in \mathbb{R}^{I \times J} \) represents residuals or noise, \( A = [a_1, a_2, \ldots, a_R] \in \mathbb{R}^{I \times R} \) is the unknown (usually, full column rank \( R \)) mixing matrix with \( R \) basis vectors \( a_r \in \mathbb{R}^I \), and \( B = [b_1, b_2, \ldots, b_R] \in \mathbb{R}^{J \times R} \) is the matrix of unknown components (factors, latent variables, sources).

Two-way component analysis (CA) refers to a class of signal processing techniques that decompose or encode superimposed or mixed signals into components with certain constraints or properties. The CA methods exploit \textit{a priori} knowledge about the true nature or diversities of latent variables. By diversity, we refer to different characteristics, features or morphology of sources or hidden latent variables [27].

For example, the columns of the matrix B that represent different data sources should be: as statistically independent as possible for ICA; as sparse as possible for SCA; take only nonnegative values for (NMF) [1], [16], [29]. The rapidly emerging field of tensor decompositions is the next important step that naturally generalizes 2-way CA/BSS algorithms and paradigms. We proceed to show how constrained matrix factorizations and component analysis (CA) models can be naturally generalized to multilinear models using constrained tensor decompositions, such as the Canonical Polyadic Decomposition (CPD) and Tucker models, as illustrated in Figs. 14 and 15.

B. The Canonical Polyadic Decomposition (CPD)

The CPD (called also PARAFAC or CANDECOMP) factorizes an \( N \)-th-order tensor \( X \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \) into a linear combination of terms \( b_r^{(1)} \circ b_r^{(2)} \circ \cdots \circ b_r^{(N)} \), which are rank-1 tensors, and is given by [30]–[32]
\[ X \cong \sum_{r=1}^{R} \lambda_r b_r^{(1)} \circ b_r^{(2)} \circ \cdots \circ b_r^{(N)} = \Lambda \times_1 B^{(1)} \times_2 B^{(2)} \cdots \times_N B^{(N)} \]
\[ = \{\Lambda B^{(1)}_B, B^{(2)}_B, \ldots, B^{(N)}_B\}, \]
where the only non-zero entries \( \lambda_r \) of the diagonal core tensor \( \mathbf{G} = \Lambda \in \mathbb{R}^{R \times R \times \cdots \times R} \) are located on the main diagonal (see Fig. 14 for a 3rd-order and 4th-order tensors).

Via the Khatri-Rao products the CPD can also be expressed in a matrix/vector form as:
\[ \chi_{(n)} \cong B^{(n)}(1) \circ \cdots \circ B^{(n-1)}(1) \circ B^{(n+1)}(1) \circ \cdots \circ B^{(N)}(1)^T \]
\[ \vec{\chi} = [B^{(1)}(1) \circ \cdots \circ B^{(N)}(1)]^T \Lambda, \]
where \( B^{(n)} = [b_r^{(1)}(n), b_r^{(2)}(n), \ldots, b_r^{(N)}(n)] \in \mathbb{R}^{I_n \times R} \), \( \Lambda = [\lambda_1, \lambda_2, \ldots, \lambda_R]^T \) is a diagonal matrix.

The rank of tensor \( X \) is defined as the smallest \( R \) for which CPD (9) holds exactly.

\textbf{Algorithms to compute CPD.} In the presence of noise in real world applications the CPD is rarely exact and has to be estimated by minimizing a suitable cost function, typically of the Least-Squares (LS) type in the form of the Frobenius norm \( |||X - [\Lambda B^{(1)}, B^{(2)}, \ldots, B^{(N)}]||_F \), or using Least Absolute Error (LAE) criteria [33]. The Alternating Least Squares (ALS) algorithms [1], [13],
collinear components) and high SNR, the performance of
ALS algorithms is often satisfactory. For ill-conditioned
problems, more advanced algorithms exist, which typi-
ically exploit the rank-1 structure of the terms within
CPD to perform efficient computation and storage of
the Jacobian and Hessian of the cost function [35], [36].

Constraints. The CPD is usually unique by itself,
and does not require constraints to impose uniqueness
[37]. However, if components in one or more modes
are known to be e.g., nonnegative, orthogonal, statisti-
cally independent or sparse, these constraints should
be incorporated to relax uniqueness conditions. More
importantly, constraints may increase the accuracy and
stability of the CPD algorithms and facilitate better
physical interpretability of components [38], [39].

C. The Tucker Decomposition

The Tucker decomposition can be expressed as follows
[40]:

\[
X \cong \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1\cdots r_N} (b_1^{(1)} \odot b_2^{(2)} \odot \cdots \odot b_N^{(N)})
\]

\[
= \mathbf{G} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} \times_4 \mathbf{B}^{(4)}
\]

where \( X \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \) is the given data tensor,
\( \mathbf{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N} \) is the core tensor and \( b^{(n)} =
[ b_1^{(n)}, b_2^{(n)}, \ldots, b_{R_n}^{(n)} ] \) is the mode-\( n \) component
matrix. \( n = 1, 2, \ldots, N \) (see Fig. 15).

Using Kronecker products the decomposition in (12)
can be expressed in a matrix and vector form as follows:

\[
X_{(n)} \cong \mathbf{B}^{(n)} \mathbf{G}_{(n)} [(\mathbf{B}^{(1)} \cdots \otimes \mathbf{B}^{(n-1)} \otimes \mathbf{B}^{(n+1)} \cdots \otimes \mathbf{B}^{(N)})^T
\vec{X} \cong [\mathbf{B}^{(1)} \otimes \mathbf{B}^{(2)} \cdots \otimes \mathbf{B}^{(N)}] \vec{G}.
\]

The core tensor (typically, \( R_n < I_n \)) models a poten-
tially complex pattern of mutual interaction between
the vectors (components) in different modes.

Multilinear rank. The \( N \)-tuple \((R_1, R_2, \ldots, R_N)\) is
called the multilinear-rank of \( X \), if the Tucker decom-
position holds exactly.

Note that the CPD can be considered as a special
case of the Tucker decomposition, in which the core
tensor has nonzero elements only on main diagonal.
In contrast to the CPD the Tucker decomposition, in
general, is non unique. However, constraints imposed
on all factor matrices and/or core tensor can reduce the
indeterminacies to only column-wise permutation and
scaling [41].

In Tucker model some selected factor matrices can
be identity matrices, this leads to Tucker-\((K,N)\) model,
which is graphically illustrated in Fig. 16 (a). In a such
model \((N-K)\) factor matrices are equal to identity
matrices. In the simplest scenario for 3rd-order tensor
\( X \in \mathbb{R}^{I_1 \times I_2 \times I_3} \) the Tucker-(2,3) model, called simply
Tucker-2, can be described as

\[
X \cong \mathbf{G} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)}.
\]
I. Tucker Decompositions

(a) Standard block diagram of TD

![Diagram](image)

(b) TD in tensor network notations

![Diagram](image)

**Figure 15:** Representation of the Tucker Decomposition (TD).

(a) TD of a 3rd-order tensor $X \cong G \times_1 B^{(1)} \times_2 B^{(2)} \times_3 B^{(3)}$. The objective is to compute factor matrices $B^{(n)}$ and core tensor $G$.

In some applications, in the second stage, the core tensor is approximately factorized using the CPD as $G \cong \sum_{r=1}^{R} a_r \circ b_r \circ c_r$.

(b) Graphical representation of the Tucker and CP decompositions in two-stage procedure for a 4th-order tensor as: $X \cong G \times_1 B^{(1)} \times_2 B^{(2)} \times_3 B^{(3)} \times_4 B^{(4)} = \left[\begin{array}{c} A_1 \times_1 A^{(1)}_2 \times_3 B^{(2)}_3 \times_4 B^{(4)}_4 \\
A_2 \times_1 B^{(1)}_2 \times_3 A^{(2)}_3 \times_4 B^{(4)}_4 \\
A_3 \times_1 A^{(1)}_2 \times_3 A^{(2)}_3 \times_4 B^{(4)}_4 \\
A_4 \times_1 B^{(1)}_2 \times_3 A^{(2)}_3 \times_4 B^{(4)}_4 \end{array}\right]$, where the core tensor, called constrained tensor or interaction tensor, is expressed as $G = I \times_1 \Phi^{(1)}_1 \times_2 \Phi^{(2)}_2 \times K \Phi^{(K)}_K$,

with $K \leq N$. The factor matrices $\Phi^{(n)} \in \mathbb{R}^{R_n \times R}$, with $R \geq \max(R_n)$ are constrained matrices, called often interaction matrices (see Fig. 16 (b)).

Another important, more complex constrained CPD model, which can be represented graphically as nested

$2$PARALIND is abbreviation of PARAllel with LINear Dependencies, while CONFAC means CONstrained FACtor model (for more detail see [42] and references therein).

Similarly, we can define PARALIND/CONFAC-($K, N$) models$^2$ described as [42]

\[ X \cong G \times_1 B^{(1)} \times_2 B^{(2)} \cdots \times_N B^{(N)} \]

\[ = \left[ I, B^{(1)}_1 \Phi^{(1)}, \ldots, B^{(K)}_K \Phi^{(K)}_K, \Phi^{(K+1)}_K, \ldots, \Phi^{(N)}_N \right], \]

where the core tensor, called constrained tensor or interaction tensor, is expressed as

\[ G = I \times_1 \Phi^{(1)}_1 \times_2 \Phi^{(2)}_2 \cdots \times_K \Phi^{(K)}_K, \]

with $K \leq N$. The factor matrices $\Phi^{(n)} \in \mathbb{R}^{R_n \times R}$, with $R \geq \max(R_n)$ are constrained matrices, called often interaction matrices (see Fig. 16 (b)).

Another important, more complex constrained CPD model, which can be represented graphically as nested

$2$PARALIND is abbreviation of PARAllel with LINear Dependencies, while CONFAC means CONstrained FACtor model (for more detail see [42] and references therein).

D. Multiway Component Analysis Using Constrained Tucker Decompositions

A great success of 2-way component analysis (PCA, ICA, NMF, SCA) is largely due to the various constraints we can impose. Without constraints matrix factorization loses its most sense as the components are rather arbitrary and they do not have any physical meaning. There are various constraints that lead to all kinds of component analysis methods which are able give unique components with some desired physical meaning and properties and hence serve for different application purposes. Just similar to matrix factorization, unconstrained Tucker decompositions generally can only be served as multiway data compression as their results lack physical meaning. In the most practical applications we need to consider constrained Tucker decompositions which can
provide multiple sets of unique components with desired physical interpretation and meaning. This is direct extension of 2-way component analysis and is referred to as multiway component analysis (MWCA) [2].

The MWCA based on Tucker-N model can be considered as a natural and simple extension of multilinear SVD and/or multilinear ICA, in which we apply any efficient CA/BSS algorithms to each mode, which provides essential uniqueness [41].

There are two different models to interpret and implement constrained Tucker decompositions for MWCA. (1) the columns of the component matrices $B^{(n)}$ represent the desired latent variables, the core tensor $\mathbf{G}$ has a role of “mixing process”, modeling the links among the components from different modes, while the data tensor $\mathbf{X}$ represents a collection of 1-D or 2-D mixing signals; (2) the core tensor represents the desired (but hidden) $N$-dimensional signal (e.g., 3D MRI image or 4D video), while the component matrices represent mixing or filtering processes through e.g., time-frequency transformations or wavelet dictionaries [3].

The MWCA based on the Tucker-$N$ model can be computed directly in two steps: (1) for $n = 1, 2, \ldots, N$ perform model reduction and unfolding of data tensors sequentially and apply a suitable set of CA/BSS algorithms to reduced unfolding matrices $\hat{X}_{(n)}$, - in each mode we can apply different constraints and algorithms; (2) compute the core tensor using e.g., the inversion formula: $\mathbf{\hat{G}} = \mathbf{X} \times_1 \mathbf{B}^{(1)\dagger} \times_2 \mathbf{B}^{(2)\dagger} \cdots \times_N \mathbf{B}^{(N)\dagger}$ [41]. This step is quite important because core tensors illuminate complex links among the multiple components in different modes [1].

V. Block-wise Tensor Decompositions for Very Large-Scale Data

Large-scale tensors cannot be processed by commonly used computers, since not only their size exceeds available working memory but also processing of huge data is very slow. The basic idea is to perform partition of a big data tensor into smaller blocks and then perform tensor related operations block-wise using a suitable tensor format (see Fig. 17). A data management system that divides the data tensor into blocks is important approach to both process and to save large datasets. The method is based on a decomposition of the original tensor dataset into small block tensors, which are approximated via TDs. Each block is approximated using low-rank reduced tensor decomposition, e.g., CPD or a Tucker decomposition.

There are three important steps for such approach before we would be able to generate an output: First, an effective tensor representation should be chosen for the input dataset; second, the resulting tensor needs to be partitioned into sufficiently small blocks stored on a distributed memory system, so that each block can fit into the main memory of a single machine; third, a suitable algorithm for TD needs to be adapted so that it can take the blocks of the original tensor, and still output the correct approximation as if the tensor for the original dataset had not been partitioned [8]–[10].

Converting the input data tensors from its original format into this block-structured tensor format is straightforward, and needs to be performed as a preprocessing step. The resulting blocks should be saved into separate files on hard disks to allow efficient random or sequential access to all of blocks, which is required by most TD and TN algorithms.

We have successfully applied such techniques to CPD [10]. Experimental results indicate that our algorithms cannot only process out-of-core data, but also achieve high computation speed and good performance.

VI. Multilinear SVD (MLSVD) for Large Scale Problems

MultiLinear Singular Value Decomposition (MLSVD), called also higher-order SVD (HOSVD) can be considered as a special form of the Tucker decomposition [43], [44], in which all factor matrices $B^{(n)} = U^{(n)} \in \mathbb{R}^{l_1 \times l_n}$ are orthogonal and the core tensor $\mathbf{G} = \mathbf{S} \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_N}$ is all-orthogonal (see Fig. 18).

We say that the core tensor is all-orthogonal if it satisfies the following conditions:
such that
\[ \mathbf{X} = \mathbf{S} \times_1 \mathbf{U}^{(1)} T \times_2 \mathbf{U}^{(2)} T \cdots \times_N \mathbf{U}^{(N)} T. \] (20)

Due to orthogonality of the core tensor \( \mathbf{S} \) its slices are mutually orthogonal, this reduces to the diagonality in the matrix case.

In some applications we may use a modified HOSVD in which the SVD can be performed not on the unfolding mode-\( n \) matrices \( \mathbf{X}_{(n)} \) but on their transposes, i.e., \( \mathbf{X}^T_{(n)} = \mathbf{V}^{(n)} \Sigma_n \mathbf{U}^{(n)T} \). This leads to the modified HOSVD corresponding to Grassmann manifolds [45], that requires the computation of very large (tall-and-skinny) factor orthogonal matrices \( \mathbf{V}^{(n)} \in \mathbb{R}^{I_n \times I_n} \), where \( I_n = \prod_{k \neq n} I_k = I_1 \cdots I_{n-1} I_{n+1} \cdots I_N \), and the core tensor \( \tilde{\mathbf{S}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \) based on the following model:
\[ \mathbf{X} = \tilde{\mathbf{S}} \times_1 \mathbf{V}^{(1)} T \times_2 \mathbf{V}^{(2)} T \cdots \times_N \mathbf{V}^{(N)} T. \] (21)

In practical applications the dimensions of unfolding matrices \( \mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_n} \) may be prohibitively large (with \( I_n \gg I_h \)), easily exceeding memory of standard computers. A truncated SVD of a large-scale unfolding matrix \( \mathbf{X}_{(n)} = \mathbf{U}^{(n)} \Sigma_n \mathbf{V}^{(n)T} \) is performed by partitioning it into \( Q \) slices, as \( \mathbf{X}_{(n)} = [\mathbf{X}_{1,n}, \mathbf{X}_{2,n}, \ldots, \mathbf{X}_{Q,n}] = \mathbf{U}^{(n)} \Sigma_n [\mathbf{V}^{(1),t}_{1,n}, \mathbf{V}^{(2),t}_{2,n}, \ldots, \mathbf{V}^{(Q),t}_{Q,n}] \). Next, the orthogonal matrices \( \mathbf{U}^{(n)} \) and the diagonal matrices \( \Sigma_n \) are obtained from eigenvalue decompositions \( \mathbf{X}_{(n)} \mathbf{X}_{(n)T} = \mathbf{U}^{(n)} \Sigma_n \mathbf{U}^{(n)T} = \sum_q \mathbf{U}_{q,n} \mathbf{U}_{q,n}^T \in \mathbb{R}^{I_n \times I_n} \), allowing for the terms \( \mathbf{V}_{q,n} = \mathbf{X}_{q,n}^{T} \mathbf{U}^{(n)} \Sigma_n^{-1} \) to be computed separately. This allows us to optimize the size of the \( q \)-th slice \( \mathbf{X}_{q,n} \in \mathbb{R}^{I_n \times (I_h/Q)} \) so as to match the available computer memory. Such a simple approach to compute matrices \( \mathbf{U}^{(n)} \) and/or \( \mathbf{V}^{(n)} \) does not require loading the entire unfolding matrices at once into computer memory, instead the access to the dataset is sequential. Depending on the size of computer memory, the dimension \( I_h \) is typically less than 10,000, while there is no limit on the dimension \( I_n = \prod_{k \neq n} I_k \). More sophisticated approaches which also exploit partition of matrices or tensors into blocks for QR/SVD, PCA, NMF/NTF and ICA can be found in [10], [29], [46]–[48].

When a data tensor \( \mathbf{X} \) is very large and cannot be stored in computer memory, then another challenge is to compute a core tensor \( \mathbf{G} = \mathbf{S} \) by directly using the formula:
\[ \mathbf{G} = \mathbf{X} \times_1 \mathbf{U}^{(1)T} \times_2 \mathbf{U}^{(2)T} \cdots \times_N \mathbf{U}^{(N)T}, \] (22)
which is generally performed sequentially as illustrated in Fig. 19 (a) and (b) [8], [9].

For very large tensors it is useful to divide the data tensor \( \mathbf{X} \) into small blocks \( \mathbf{X}_{k_1,k_2,k_3,...,k_n} \) and in order to store them on hard disks or distributed memory. In similar way, we can divide the orthogonal factor matrices \( \mathbf{U}^{(n)T} \) into corresponding blocks of matrices \( \mathbf{U}^{(n)T}_{k_1,k_2,...,k_n} \) as illustrated in Fig. 19 (c) for 3rd-order tensors [9]. In a general case, we can compute blocks within the resulting...
We need to perform sequential computation of subtensors $X$, and (b) alternative method for very large-scale problems by applying divide and conquer approach, in which a data tensor $X$ and factor matrices $U^{(i)\top}$ are partitioned into suitable small blocks: Subtensors $X_{[k_1,k_2,k_3]}$ and blocks matrices $U^{(1)\top}_{[k_1,p_1]}$, respectively. We compute the blocks of tensor $Z = G^{(1)} = X \times U^{(1)\top}$ as follows $Z_{[k_1,k_2,k_3]} = \sum_{k_1=1}^{K_1} X_{[k_1,k_2,k_3]} \times U^{(1)\top}_{[k_1,p_1]}$ (see Eq. (23) for a general case).

$$G^{(n)} = \sum_{k_1=1}^{K_1} X_{[k_1,k_2,\ldots,k_n]} \times U^{(n)\top}_{[k_1,p_1]} \quad (23)$$

If a data tensor has low-multilinear rank, so that its multilinear rank $\{R_1, R_2, \ldots, R_N\}$ with $R_n \ll I_n$, \forall$n$, we can further alleviate the problem of dimensionality by first identifying a subtensor $W \in \mathbb{R}^{P_1 \times P_2 \times \cdots \times P_N}$ for which $P_n \geq R_n$, using efficient CUR tensor decompositions [49]. Then the HOSVD can be computed from subtensors as illustrated in Fig. 20 for a 3rd-order tensor. This feature can be formulated in more general form as the following Proposition.

Proposition 1: If a tensor $X \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ has low multilinear rank $\{R_1, R_2, \ldots, R_N\}$, with $R_n \ll I_n$, \forall$n$, then it can be fully reconstructed via the HOSVD using only $N$ subtensors $X^{(n)} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, under the condition that subtensor $W \in \mathbb{R}^{P_1 \times P_2 \times \cdots \times P_N}$, with $P_n \geq R_n$, \forall$n$ has the multilinear rank $\{R_1, R_2, \ldots, R_N\}$.

In practice, we can compute the HOSVD for low-rank, large-scale data tensors in several steps. In the first step, we can apply the CUR FSTD decomposition [49] to identify close to optimal a subtensor $W \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ (see the next Section). In the next step, we can use the standard SVD for unfolding matrices $X^{(n)}$ of subtensors $X^{(n)}$ to compute the left orthogonal matrices $\bar{U}^{(n)} \in \mathbb{R}^{R_n \times R_n}$. Hence, we compute an auxiliary core tensor $\bar{G} = W \times \bar{B}^{(1)} \times \cdots \times \bar{B}^{(N)}$, where $\bar{B}^{(n)} \in \mathbb{R}^{R_n \times R_n}$ are inverses of the sub-matrices consisting the first $R_n$ rows of the matrices $U^{(n)}$. In the last step, we perform HOSVD decomposition of the relatively small core tensor as $\bar{G} = S \times Q^{(1)} \times \cdots \times Q^{(N)}$, with $Q^{(n)} \in \mathbb{R}^{R_n \times R_n}$ and then desired orthogonal matrices are computed as $U^{(n)} = \bar{U}^{(n)}Q^{(n)}$.

VII. CUR Tucker Decomposition for Dimensionality Reduction and Compression of Tensor Data

Note that instead of using the full tensor, we may compute an approximative tensor decomposition model from a limited number of entries (e.g., selected fibers, slices or subtensors). Such completion-type strategies have been developed for low-rank and low-multilinear rank [50], [51]. A simple approach would be to apply

Figure 19: Computation of a core tensor for a large-scale HOSVD: (a) using sequential computing of multilinear products $G = S = (\{X_1 \times U^{(1)\top}\} \times_2 U^{(2)\top}) \times_3 U^{(3)\top}$, and (b) by applying fast and distributed implementation of matrix by matrix multiplications; (c) alternative method for very large-scale problems by applying divide and conquer approach, in which a data tensor $X$ and factor matrices $U^{(i)\top}$ are partitioned into suitable small blocks: Subtensors $X_{[k_1,k_2,k_3]}$ and blocks matrices $U^{(1)\top}_{[k_1,p_1]}$, respectively. We compute the blocks of tensor $Z = G^{(1)} = X \times U^{(1)\top}$ as follows $Z_{[k_1,k_2,k_3]} = \sum_{k_1=1}^{K_1} X_{[k_1,k_2,k_3]} \times U^{(1)\top}_{[k_1,p_1]}$ (see Eq. (23) for a general case.)

Figure 20: Alternative approach to computation of the HOSVD for very large data tensor, by exploiting multilinear low-rank approximation. The objective is to select such fibers (up to permutation of fibers) that the subtensor $W \in \mathbb{R}^{P_1 \times P_2 \times P_3}$ with $P_n \geq R_n (n = 1, 2, 3)$ has the same multilinear rank $\{R_1, R_2, R_3\}$ as the whole huge data tensor $X$ with $R_n \ll I_n$. Instead of unfolding of the whole data tensor $X$ we need to perform unfolding (and applying the standard SVD) for typically much smaller subtensors $X^{(i)} = C \in \mathbb{R}^{P_1 \times P_2 \times P_3}$, $X^{(2)} = R \in \mathbb{R}^{P_1 \times P_2 \times I_3}$, each in a single mode-$n$, $(n = 1, 2, 3)$. This approach can be applied if data tensor admits low multilinear rank approximation. For simplicity of illustration, we assumed that fibers are permuted in a such way that the first $P_1 - 1$, $P_2$, $P_3$ fibers were selected.

Reduction and Compression of Tensor Data

CUR Tucker Decomposition for Dimensionality Reduction and Compression of Tensor Data

Note that instead of using the full tensor, we may compute an approximative tensor decomposition model from a limited number of entries (e.g., selected fibers, slices or subtensors). Such completion-type strategies have been developed for low-rank and low-multilinear rank [50], [51]. A simple approach would be to apply
CUR decomposition or Cross-Approximation by sampled fibers for the columns of factor matrices in a Tucker approximation [49], [52]. Another approach is to apply tensor networks to represent big data by high-order tensors not explicitly but in compressed tensor formats (see next sections). Dimensionality reduction methods are based on the fundamental assumption that large datasets are highly redundant and can be approximated by low-rank matrices and cores, allowing for a significant reduction in computational complexity and to discover meaningful components while exhibiting marginal loss of information.

For very large-scale matrices, the so called CUR matrix decompositions can be employed for dimensionality reduction [49], [52]–[55]. Assuming a sufficiently precise low-rank approximation, which implies that data has some internal structure or smoothness, the idea is to provide data representation through a linear combination of a few “meaningful” components, which are exact replicas of columns and rows of the original data matrix [56].

The CUR model, also called skeleton Cross-Approximation, decomposes a data matrix \( X \in \mathbb{R}^{I \times J} \) as [53], [54] (see Fig. 21):

\[
X = CUR + E,
\]

(24)

where \( C \in \mathbb{R}^{I \times C} \) is a matrix constructed from \( C \) suitably selected columns of the data matrix \( X \), \( R \in \mathbb{R}^{R \times J} \) consists of \( R \) rows of \( X \), and the matrix \( U \in \mathbb{R}^{C \times R} \) is chosen to minimize the norm of the error \( E \in \mathbb{R}^{I \times J} \). Since typically, \( C \ll J \) and \( R \ll I \), these columns and rows are chosen so as to exhibit high “statistical leverage” and provide the best low-rank fit to the data matrix, at the same time the error cost function \( \|E\|_F^2 \) is minimized. For a given set of columns (\( C \)) and rows (\( R \)), the optimal choice for the core matrix is \( U = C^\top X (R^\top)^\top \). This requires access to all the entries of \( X \) and is not practical or feasible for large-scale data. A pragmatic choice for the core matrix would be \( U = W^\top \), where the matrix \( W \in \mathbb{R}^{R \times C} \) is defined from the intersections of the selected rows and columns. It should be noted that, if \( \text{rank}(X) \leq C, R \), then the CUR approximation is exact. For the general case, it has been proven that, when the intersection sub-matrix \( W \) is of maximum volume (the volume of a sub-matrix \( W \) is defined as \( |\det(W)| \)), this approximation is close to the optimal SVD solution [54].

**Figure 21**: CUR decomposition for a huge matrix.

**Figure 22**: (a) CUR decomposition of a large 3rd-order tensor (for simplicity of illustration up to permutation of fibers) \( X \equiv U \times_1 C \times_2 R \times_3 T = [U; C, R, T] \), where \( U = W \times_1 W_1^{(1)} \times_2 W_2^{(2)} \times_3 W_3^{(3)} = [W; W_1^{(1)}, W_2^{(2)}, W_3^{(3)}] \). (b) Equivalent decomposition expressed via subtensor \( W \). (c) Tensor network diagram illustrating transformation from CUR Tucker format (a) to form (b) as: \( X \equiv W \times_1 B^{(1)} \times_2 B^{(2)} \times_3 B^{(3)} = [W; CW_1^{(1)}, RW_2^{(2)}, TW_3^{(3)}] \).
The concept of CUR decomposition has been successfully generalized to tensors. In [52] the matrix CUR decomposition was applied to one unfolded version of the tensor data, while in [49] a reconstruction formula of a tensor having a low rank Tucker approximation was proposed, termed the Fiber Sampling Tucker Decomposition (FSTD), which is a practical and fast technique. The FSTD takes into account the linear structure in all the modes of the tensor simultaneously. Since real-life data often have good low multilinear rank approximations, the FSTD provides such a low-rank Tucker decomposition that is directly expressed in terms of a relatively small number of fibers of the data tensor (see Fig. 22).

For a given 3rd-order tensor \( X \in \mathbb{R}^{I_1 \times I_2 \times I_3} \) for which an exact rank-(\(R_1, R_2, R_3\)) Tucker representation exists, FSTD selects \( P_n \geq R_n \) \((n = 1, 2, 3)\) indices in each mode, which determine an intersection sub-tensor \( W \in \mathbb{R}^{P_1 \times P_2 \times P_3} \) so that the following exact Tucker representation can be obtained:

\[
X = [U; C, R, T],
\]

in which the core tensor is computed as \( U = G = [W_1^T W_2^T W_3^T] \) and the factor matrices \( C \in \mathbb{R}^{I_1 \times P_1}, R \in \mathbb{R}^{I_2 \times P_2}, T \in \mathbb{R}^{I_3 \times P_3} \) contain the fibers (columns, rows and tubes, respectively). This can also be written as a Tucker representation:

\[
X = [W_1^T C W_2^T, R W_2^T, T W_3^T].
\]

Observe that for \( N = 2 \) this model simplifies into the CUR matrix case, \( X = CUR \), and the core matrix is \( U = [W_1^T W_2^T] = W^T \).

In a more general case for an \( N \)-th order tensor, we can formulate the following Proposition [49].

**Proposition 2:** If tensor \( X \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \) has low multilinear rank \([R_1, R_2, \ldots, R_N]\), with \( R_n \leq I_n, \forall n \), then it can be fully reconstructed via the CUR FSTD \( X = [U; C^{(1)}, C^{(2)}, \ldots, C^{(N)}] \) using only \( N \) factor matrices \( C^{(n)} \in \mathbb{R}^{I_n \times P_n} \) \((n = 1, 2, \ldots, N)\), built up from fibers of the data tensor, and a core tensor \( U = G = [W_1^T W_2^T, \ldots, W_N^T] \), under the condition that subtensor \( W \in \mathbb{R}^{P_1 \times P_2 \times \cdots \times P_N} \) with \( P_n \geq R_n, \forall n \) has multilinear rank \([R_1, R_2, \ldots, R_N]\).

An efficient strategy for the selection of suitable fibers, only requiring access to a partial (small) subset of entries of a data tensor through identifying the entries with maximum modulus within single fibers is given in [49]. The indices are selected sequentially using a deflation approach making the FSTD algorithm suitable for very large-scale but relatively low-order tensors (including tensors with missing fibers or entries).

**VIII. Analysis of Coupled Multi-Block Tensor Data – Linked Multiway Component Analysis (LMWCA)**

Group analysis or multi-block data analysis aims to identify links between hidden components in data making it possible to analyze the correlation, variability and consistency of the components across multi-block data sets. This equips us with enhanced flexibility: Some components do not necessarily need to be orthogonal or statistically independent, and can be instead sparse, smooth or non-negative (e.g., for spectral components). Additional constraints can be used to reflect the spatial distributions, spectral, or temporal patterns [3].

Consider the analysis of multi-modal high-dimensional data collected under the same or very similar conditions, for example, a set of EEG and MEG or fMRI signals recorded for different subjects over many trials and under the same experiment configuration and mental tasks. Such data share some common latent (hidden) components but can also have their own independent features. Therefore, it is quite important and necessary that they will be analyzed in a linked way instead of independently.

![Figure 23: Linked Multiway Component Analysis (LMWCA) for coupled multi-block 3rd-order tensors, with different dimensions in each mode except of the first mode. The objective is to find the common components \( B^{(1)}_{\text{C}} \in \mathbb{R}^{I_1 \times C_1} \), where \( C_1 \leq R_1 \) is the number of the common components in mode-1.](image)

The linked multiway component analysis (LMWCA) for multi-block tensors data is formulated as a set of approximate joint Tucker-(\(1, N\)) decompositions of a set of data tensors \( X^{(k)} \in \mathbb{R}^{I_1^{(k)} \times I_2^{(k)} \times \cdots \times I_N^{(k)}} \), with \( I_1^{(k)} = I_1, \forall k \) \((k = 1, 2, \ldots, K)\) (see Fig. 23):

\[
X^{(k)} = G^{(k)} \times_1 B^{(1,k)}, \quad (k = 1, 2, \ldots K)
\]

where each factor (component) matrix \( B^{(1,k)} = [B^{(1)}_{\text{C}} B^{(1,L)}] \in \mathbb{R}^{I_1 \times R_0} \) has two sets of components: (1) Components \( B^{(1)}_{\text{C}} \in \mathbb{R}^{I_1 \times C}\) \((0 \leq C \leq R)\), which are common for all available blocks and correspond to identical or maximally correlated components, and (2) components \( B^{(1,L)} \in \mathbb{R}^{I_1 \times (R_1 - C)} \), which are different.
independent processes, for example, latent variables independent of excitations or stimuli/tasks. The objective is to find core tensors which are maximally correlated for (a) Tensor Train and for (b) Tensor Tree States (Hierarchical Tucker).

If $\mathbf{B}^{(n,k)} = \mathbf{B}_C^{(n)} \in \mathbb{R}^{R_x \times R_y}$ for a specific mode $n$ (in our case $n = 1$), under additional assumption that tensors are of the same dimension. Then the problem simplifies into a generalization of Common Component Analysis or tensor Population Value Decomposition (PVD) [57] and can be solved by concatenating all data tensors along one mode, and perform constrained Tucker or CP tensor decompositions (see [57]).

In a more general case, when $C_n < R_n$, we can unfold each data tensor $\mathbf{X}^{(k)}$ in common mode, and perform a set of linked and constrained matrix factorizations: $\mathbf{X}^{(k)} \cong \mathbf{B}_C^{(1)} \mathbf{A}_C^{(1,k)} + \mathbf{B}_I^{(1,k)} \mathbf{A}_I^{(1,k)}$ through solving constrained optimization problems:

$$
\min_{\mathbf{B}_C^{(1)}, \mathbf{A}_C^{(1,k)}, \mathbf{B}_I^{(1,k)}, \mathbf{A}_I^{(1,k)}} \sum_{k=1}^{K} \| \mathbf{X}^{(k)} - \mathbf{B}_C^{(1)} \mathbf{A}_C^{(1,k)} - \mathbf{B}_I^{(1,k)} \mathbf{A}_I^{(1,k)} \|_F + f_1(\mathbf{B}_C^{(1)}) \quad \text{s.t.} \quad \mathbf{B}_C^{(1)^T} \mathbf{B}_I^{(1,k)} = 0 \quad \forall k,
$$

where $f_1$ are the penalty terms which impose additional constraints on common components $\mathbf{B}_C^{(1)}$, in order to extract as many as possible unique and desired components. In a special case, when we impose orthogonality constraints, the problem can be transformed to a generalized eigenvalue problem and solved by the power method [11]. The key point is to assume that common factor sub-matrices $\mathbf{B}_C^{(1)}$ are present in all multiple data blocks and hence reflect structurally complex (hidden) latent and intrinsic links between them. In practice, the number of common components $C_1$ in each mode is unknown and should be estimated (see [11] for detail).

The linked multiway component analysis model provides a quite flexible and general framework and thus supplements currently available techniques for group ICA and feature extraction for multi-block data. The LWCA models are designed for blocks of $K$ tensors, where dimensions naturally split into several different modalities (e.g., time, space and frequency). In this sense, a multi-block multiway CA attempts to estimate both common and independent or uncorrelated components, and is a natural extension of group ICA, PVD, and CCA/PLS methods (see [3], [11], [12], [58] and references therein). The concept of LMWCA can be generalized to tensor networks as illustrated in Fig. 24.

**IX. Mathematical and Graphical Description of Tensor Trains (TT) Decompositions**

In this section we discuss in more detail the Tensor Train (TT) decompositions which are the simplest tensor networks. Tensor train decomposition was introduced by Oseledets and Tyrtyshnikov [21], [59] and can take various forms depending on the order of input data as illustrated in Fig. 25.
Fig. 26: Illustration of tensor train decomposition (TT/MPS) of a 4th-order data tensor \( \mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4} \). (a) Tensor form via multilinear product of cores and/or the outer product of vectors (fibers) as sum of rank-1 tensors as:\n\[
\mathbf{X} \cong \mathbf{G}^{(1)} \circ \mathbf{G}^{(2)} \circ \mathbf{G}^{(3)} \circ \mathbf{G}^{(4)}
\]
for \( \mathbf{G}^{(r)} = \sum_{I_{r-1} \times I_{r} \times I_{r+1}} g_{I_{r-1},I_{r},I_{r+1}} \mathbf{g}_{r-1} \mathbf{g}_{r} \mathbf{g}_{r+1} \). Note that the TT model is equivalent to the MPS only if the MPS has the open boundary conditions (OBC) [65, 66].

The tensor train (TT/MPS) for an \( N \)th-order data tensor \( \mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N} \) can be described in the various equivalent mathematical forms as follows:

- In a compact tensor form using multilinear products:
  \[
  \mathbf{X} \cong \mathbf{A} \times_1 \mathbf{G}^{(2)} \times_2 \mathbf{G}^{(3)} \times_3 \mathbf{G}^{(N-1)} \times_4 \mathbf{B}
  = [\mathbf{A}, \mathbf{G}^{(2)}, \mathbf{G}^{(3)}, \ldots, \mathbf{G}^{(N-1)}, \mathbf{B}],
  \]
  (29)
  where 3rd-order cores are defined as \( \mathbf{G}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n} \) for \( n = 2,3,\ldots,N-1 \) (see Fig. 26 (a)).

- By unfolding of cores \( \mathbf{G}^{(n)} \) and suitable reshaping of matrices, we can obtain other very useful mathematical and graphical descriptions of the MPS, for example, as summation of rank-1 tensors using outer (tensor) product (similar to CPD, Tucker and PARATREE formats):

\[
\mathbf{X} \cong \sum_{r_1,r_2,\ldots,r_{N-1}} g^{(1)}_{r_1} \circ g^{(2)}_{r_1,r_2} \circ g^{(3)}_{r_2,r_3} \circ \cdots \circ g^{(N)}_{r_{N-1},1}
\]

(30)

where \( g^{(n)}_{r_1} \in \mathbb{R}^{I_n} \) are column vectors of matrices\n\[
\mathbf{G}^{(n)} = [g^{(n)}_{1}, g^{(n)}_{2}, \ldots, g^{(n)}_{R_{n-1},1}, g^{(n)}_{R_{n-1},2}, \ldots, g^{(n)}_{R_{n-1},R_n}] \in \mathbb{R}^{I_n \times R_{n-1} \times R_n}
\]

(\( n = 1,2,\ldots,N \), with \( R_0 = R_N = 1 \)). Note that \( g^{(1)}_{r_1} = a_{r_1} \) are columns of the matrix \( \mathbf{A} = [a_1, a_2, \ldots, a_{R_1}] \in \mathbb{R}^{I_1 \times R_1} \), while \( g^{(N-1)}_{r_{N-1},1} = b_{r_{N-1}} \) are vector of the transposed factor matrix \( \mathbf{B}^T = [b_1, b_2, \ldots, b_{R_{N-1}}] \in \mathbb{R}^{I_N \times R_{N-1}} \) (see Fig. 26 (a)).

The minimal \( (N-1) \) tuple \( \{R_1, R_2, \ldots, R_{N-1}\} \) is called TT-rank (strictly speaking for the exact TT decomposition).
Alternative representation of the tensor train decomposition (TT/MPS) expressed via strong Kronecker products of block matrices in the form of a vector as: 
\[ x_{i_1,i_2,\ldots,i_N} \cong  \sum_{r_1,r_2,\ldots,r_N=1} G^{(1)}_{i_1,r_1} G^{(2)}_{i_2,r_2} \cdots G^{(n)}_{i_N,r_N} \]

(31)

or equivalently using slice representations (see Fig. 26):
\[ x_{i_1,i_2,\ldots,i_N} = G^{(1)}_{i_1,1} G^{(2)}_{i_2,1} \cdots G^{(N)}_{i_N,1} \]

(32)

where slice matrices \( G^{(n)}_{i_n} \) denote vectors of the tensor in lexicographical order of indices and \( \otimes \) denotes strong Kronecker product.

The strong Kronecker product of two block matrices (e.g., unfolding cores):
\[ A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,R_2} \\ \vdots & \ddots & \vdots \\ A_{R_1,1} & \cdots & A_{R_1,R_2} \end{bmatrix} \in \mathbb{R}^{R_1 \times R_2 \times R_3 \times R_4} \]
A 4th-order tensor can be represented equivalently as a (block) matrix (lateral slice) of the tensor, while a 3rd-order tensor can be represented equivalently as a 4th-order cores, as illustrated in Fig. 29. Note that a MPO (Matrix Product Operator with Open Boundary Conditions) is defined as a block matrix

\[
 C = A \otimes B \in \mathbb{R}^{R_1 l_1 \times R_3 l_3 l_2},
\]

with blocks

\[
 C_{\tau_1 \tau_3} = \sum_{\tau_2=1}^{R_2} A_{\tau_1 \tau_2} \otimes B_{\tau_2 \tau_3} \in \mathbb{R}^{l_1 \times l_2 \times l_3},
\]

where \( A_{\tau_1 \tau_2} \in \mathbb{R}^{l_1 \times l_2} \) and \( B_{\tau_2 \tau_3} \in \mathbb{R}^{l_2 \times l_3} \) are block matrices of \( A \) and \( B \), respectively (see also Fig. 28 for graphical illustration).

The matrix strong Kronecker product can be generalized to block tensors as follows: Let

\[
 A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix},
\]

then the strong Kronecker product of \( A \) and \( B \) is defined by the \( R_1 \times R_3 \) block tensor

\[
 C = [C_{\tau_1 \tau_3}] = A \otimes B \in \mathbb{R}^{R_1 l_1 \times R_3 l_1 l_2},
\]

where

\[
 C_{\tau_1 \tau_3} = \sum_{\tau_2=1}^{R_2} A_{\tau_1 \tau_2} \otimes B_{\tau_2 \tau_3} \in \mathbb{R}^{l_1 l_2 \times l_3 l_1 l_2},
\]

for \( \tau_1 = 1, 2, \ldots, R_1 \) and \( \tau_3 = 1, 2, \ldots, R_3 \). Another important TT model, called matrix TT or MPO (Matrix Product Operator with Open Boundary Conditions), consists of a chain (train) of 3rd-order and 4th-order cores, as illustrated in Fig. 29. Note that a 3rd-order tensor can be represented equivalently as a (block) matrix (lateral slice) of the tensor, while a 4th-order tensor can represented equivalently as a block matrix. The TT/MPO model for \( N \)th-order tensor \( X \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_N} \) can be described mathematically in the following general forms (see also Table III).

- A) In the tensor compact form using multilinear products

\[
 X \approx \mathbf{G}^{(1)} \times_4 \mathbf{G}^{(2)} \times_4 \cdots \times_4 \mathbf{G}^{(N)} = [\mathbf{G}^{(1)}, \mathbf{G}^{(2)}, \ldots , \mathbf{G}^{(N)}],
\]

where the cores are defined as \( \mathbf{G}^{(n)} \in \mathbb{R}^{R_{n-1} l_1 \times l_2 \times l_N} \), with \( R_0 = R_N = 1 \), \( (n = 1, 2, \ldots, N) \).

- B) Using the standard (rather long and tedious) scalar form:

\[
 x_{i_1,i_2,\ldots,i_{2N}} \approx \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N-1}=1}^{R_{N-1}} \delta_{r_1,i_1} \delta_{r_2,i_2} \cdots \delta_{r_{N-1},i_{N-1}} s_{r_1,r_2}^{(1)} s_{r_2,r_3}^{(2)} \cdots s_{r_{N-1},r_{N-2}}^{(N-1)} s_{r_{N-1},i_{N-1}}^{(N)},
\]

(37)

- C) By matrix representations of cores, the TT/MPO decomposition can be expressed by strong Kronecker products (see Fig. 29):

\[
 X_{(i_1,i_2,\ldots,i_{2N-1};j_1,j_2,\ldots,j_{2N-2})} \approx \tilde{G}^{(1)} \otimes \tilde{G}^{(2)} \otimes \cdots \otimes \tilde{G}^{(N)},
\]

(38)

where \( \tilde{X}_{(i_1,i_2,\ldots,i_{2N-1};j_1,j_2,\ldots,j_{2N-2})} \in \mathbb{R}^{l_1 l_2 \times \cdots \times l_N} \) is unfolding matrix of \( X \) in lexicographical order of indices and \( \tilde{G}^{(n)} \in \mathbb{R}^{R_{n-1} l_1 \times l_2 \times l_N} \) is block matrices with blocks \( \tilde{G}^{(n)}_{r_1,r_2} \in \mathbb{R}^{l_1 l_2 \times l_N} \) and the number of blocks \( R_{n-1} \times R_n \). In the special case when ranks of the TT/MPO \( R_n = 1 \), \( \forall n \) the strong Kronecker products simplify to the standard Kronecker products.

The Tensor Train (TT) format [59], can be interpreted as a special case of the HT [7], where all nodes of the underlying tensor network are aligned and where, moreover, the leaf matrices are assumed to be identities (and thus need not be stored). An advantage of the TT format is its simpler practical implementation using SVD or alternative low-rank matrix approximations, as no binary tree need be involved [21], [70] (see Figs. 30 and 31).

Two different types of approaches to perform tensor approximation via TT exist [26]. The first class of methods is based on combining standard iterative algorithms, with a low-rank decompositions, such as SVD/QR or CUR or Cross-Approximations. Similar to the Tucker decomposition, the TT and HT decompositions are usually based on low rank approximation of generalized unfolding matrices \( X_{(|n|)} \), and a good approximation in a decomposition for a given TT/HT-rank can be obtained using the truncated SVDs of the unfolding matrices [59], [70].

In [55] Oseledets and Tyrtyshnikov proposed for TT decomposition a new approximative formula in which a 1/Nth-order data tensor is interpolated using special form of Cross-Approximation, which is a modification of CUR algorithm. The total number of entries and the complexity of the interpolation algorithm depend linearly on the order of data tensor \( N \), so the developed algorithm does not suffer from the curse of dimensionality. The TT-Cross-Approximation is analog to the SVD/HOSVD like algorithms for TT/MPS, but uses adaptive cross-approximation instead of the computationally more expensive SVD.
The second class of algorithms based on optimization of suitable designed cost functions, often with additional penalty or regularization terms. Optimization techniques include gradient descent, conjugate gradient, and Newton-like methods (see [26], [71] and references therein). Gradient descent methods lead often to the Alternating Least Squares (ALS) type of algorithms, which can be improved in various ways [71], [73] (see Fig. 32).

A quite successful improvement in TNs (TT, HT) is called the DMRG method. It joins two neighboring factors (cores), optimizes the resulting “supernode”, and splits the result into separate factors by a low-rank matrix factorization [65], [66], [71], [72] (see Fig. 33).

Remark: In most optimization problems it is very convenient to present TT in a canonical form, in which all cores are left or right orthogonal [71], [74] (see also Fig. 32 and Fig. 33).

The N-order core tensor is called right-orthogonal if
\[ G^{(1)} G^T = I. \] (39)

Analogously the N-order core tensor is called left orthogonal if
\[ G^{(N)} G^T = I. \] (40)

In contrast, for the all-orthogonal core tensor we have \( G^{(n)} G^T = I, \) ~\( \forall n. \)
### Table III: Different forms of the Tensor Trains (TT): MPS and MPO (with OBC) representations of an $N$th-order tensor $X \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a 2Nth-order tensor $Y \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times I_N}$, respectively. It is assumed that the TT rank is $\{R_1,R_2,\ldots,R_{N-1}\}$, with $R_0 = R_N = 1$ ($r_0 = r_N = 1$).

<table>
<thead>
<tr>
<th>Type</th>
<th>TT/MPM</th>
<th>TT/MPO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scalar (standard) Representations</strong></td>
<td>$X_{i_1,i_2,\ldots,i_N} = \sum_{r_1,r_2,\ldots,r_{N-1}=1}^{R_1,R_2,\ldots,R_{N-1}} S^{(1)}<em>{r_1} \otimes S^{(2)}</em>{r_2} \otimes \cdots \otimes S^{(N)}<em>{r</em>{N-1},j_{N-1}}$</td>
<td>$Y_{i_1,i_2,\ldots,i_{2N}} = \sum_{r_1,r_2,\ldots,r_{N-1}=1}^{R_1,R_2,\ldots,R_{N-1}} S^{(1)}<em>{r_1,i_1,\ldots,i</em>{N-1},j_{N-1}} \otimes S^{(2)}<em>{r_2,i</em>{N-1},\ldots,i_{2N-1},j_{2N-1}} \otimes \cdots \otimes S^{(N)}<em>{r</em>{N-1},j_{N-1}}$</td>
</tr>
<tr>
<td>$g_{r_1,i_{N-1},r_{N-1}}$ entries of a 3rd-order core $G^{(n)} \in \mathbb{R}^{I_{N-1} \times I_N \times I_N}$</td>
<td>$g_{r_1,i_{N-1},r_{N-1}}$ entries of a 4th-order core $G^{(n)} \in \mathbb{R}^{I_{N-1} \times I_N \times I_N \times I_N}$</td>
<td></td>
</tr>
<tr>
<td><strong>Slice Representations</strong></td>
<td>$X = G^{(1)}(i_1) \otimes G^{(2)}(i_2) \otimes \cdots \otimes G^{(N-1)}(i_{N-1}) \otimes G^{(N)}(i_N)$</td>
<td>$Y = G^{(1)}(i_1, j_1) \otimes G^{(2)}(i_2, j_2) \otimes \cdots \otimes G^{(N)}(i_N, j_N)$</td>
</tr>
<tr>
<td>$G^{(n)}(i_n) \in \mathbb{R}^{I_{N-1} \times I_N}$</td>
<td>$G^{(n)}(i_n, j_n) \in \mathbb{R}^{I_{N-1} \times I_N \times I_N}$</td>
<td></td>
</tr>
<tr>
<td><strong>Tensor Representations: Multilinear Products (tensor contractions)</strong></td>
<td>$X = G^{(1)}(i_1) \times G^{(2)}(i_2) \times \cdots \times G^{(N)}(i_N)$</td>
<td>$Y = G^{(1)}(i_1) \times G^{(2)}(i_2) \times \cdots \times G^{(N)}(i_N)$</td>
</tr>
<tr>
<td>$G^{(n)} \in \mathbb{R}^{I_{N-1} \times I_N \times I_N}$</td>
<td>$G^{(n)} \in \mathbb{R}^{I_{N-1} \times I_N \times I_N \times I_N}$</td>
<td></td>
</tr>
<tr>
<td><strong>Tensor Representations: Outer Products</strong></td>
<td>$X = \sum_{r_1,r_2,\ldots,r_{N-1}=1}^{R_1,R_2,\ldots,R_{N-1}} G^{(1)}<em>{1,r_1} \otimes G^{(2)}</em>{r_1,2} \otimes \cdots \otimes G^{(N)}<em>{r</em>{N-1},N}$</td>
<td>$Y = \sum_{r_1,r_2,\ldots,r_{N-1}=1}^{R_1,R_2,\ldots,R_{N-1}} G^{(1)}<em>{r_1,1} \otimes G^{(2)}</em>{r_1,2} \otimes \cdots \otimes G^{(N)}<em>{r</em>{N-1},N}$</td>
</tr>
<tr>
<td>$g_{r_{N-1},r_{N-1}} \in \mathbb{R}^{I_N}$ blocks of a matrix $G^{(n)} = (G^{(n)}(j)) \in \mathbb{R}^{I_{N-1} \times I_N}$</td>
<td>$g_{r_{N-1},r_{N-1}} \in \mathbb{R}^{I_N \times I_N \times I_N}$ blocks of a matrix $G^{(n)} = (G^{(n)}(j)) \in \mathbb{R}^{I_{N-1} \times I_N \times I_N}$</td>
<td></td>
</tr>
<tr>
<td><strong>Vector/Matrix Representations: Kronecker and Strong Kronecker Products</strong></td>
<td>$x_{(i_1,i_2)} = \sum_{r_1,r_2,\ldots,r_{N-1}=1}^{R_1,R_2,\ldots,R_{N-1}} G^{(1)}<em>{1,r_1} \otimes G^{(2)}</em>{r_1,2} \otimes \cdots \otimes G^{(N)}<em>{r</em>{N-1},N}$</td>
<td>$y_{(i_1,i_2)} = \sum_{r_1,r_2,\ldots,r_{N-1}=1}^{R_1,R_2,\ldots,R_{N-1}} G^{(1)}<em>{r_1,1} \otimes G^{(2)}</em>{r_1,2} \otimes \cdots \otimes G^{(N)}<em>{r</em>{N-1},N}$</td>
</tr>
<tr>
<td>$G^{(n)}(i_1,i_2) \in \mathbb{R}^{I_N \times I_N}$ a block matrix with blocks $g^{(n)}<em>{r</em>{N-1},r_{N-1}} \in \mathbb{R}^{I_N \times I_N}$</td>
<td>$G^{(n)}(i_1,i_2) \in \mathbb{R}^{I_N \times I_N \times I_N}$ a block matrix with blocks $g^{(n)}<em>{r</em>{N-1},r_{N-1}} \in \mathbb{R}^{I_N \times I_N \times I_N}$</td>
<td></td>
</tr>
</tbody>
</table>

**TT-Rounding** TT-rounding (also called truncation or recompression) [55] is post-processing procedure to reduce the TT ranks which in the first stage after applying low-rank matrix factorizations are usually not optimal with respect of desired approximation errors. The optimal computation of TT-tensor is generally impossible without TT-rounding. The tensor expressed already in optimal computation of TT-tensor is generally impossible with respect of desired approximation errors. The optimal truncation algorithms for TT are described by Oseledets in [59]. In fact, the method exploits microiterations algorithm where the SVD is performed only on a relatively small core at each iteration. A similar approach has been developed by Grasedyck for the HT [76]. HT/TT algorithms that avoid the explicit computation of these SVDs when truncating a tensor that is already in tensor network format are discussed in [26], [72], [77].

TT Toolbox developed by Oseledets (http://spring.
inm.ras.ru/osel/?page_id=24) is focused on TT structures, and deals with the curse of dimensionality [75]. The Hierarchical Tucker (HT) toolbox by Kressner and Tobler (http://www.sam.math.ethz.ch/NLAgroup/hhtucker_toolbox.html) and Calculus library by Hackbusch, Waehnert and Espig, focus mostly on HT and TT tensor networks [25], [75], [77]. See also recently developed TDALAB (http://bsp.brain.riken.jp/TDALAB and TENSORBOX http://www.bsp.brain.riken.jp/~phan that provide user-friendly interface and advanced algorithms for selected TD (Tucker, CPD) models [78], [79]. The (http://www.esat.kuleuven.be/sista/tensorlab/) Tensorlab toolbox builds upon the complex optimization framework and offers efficient numerical algorithms for computing the TDs with various constraints (e.g. nonnegativity, orthogonality) and the possibility to combine and jointly factorize dense, sparse and incomplete tensors [80]. The problems related to optimization of existing TN/TD algorithms are active area of research [26], [71], [73], [81], [82].

X. Hierarchical Outer Product Tensor Approximation (HOPTA) and Kronecker Tensor Decompositions

Recent advances in TDs/TNs include, TT/HT [26], [60], [71], PARATREE [83], Block Term Decomposition

Figure 30: Illustration of the SVD algorithm for TT/MPS for a 5th-order tensor. Instead of truncated SVD, we can employ any low-rank matrix factorizations, especially QR, CUR, SCA, ICA, NMF.

Figure 31: SVD algorithm for TT/MPO for a 6th-order tensor. Instead of the SVD we can use alternative Low-Rank Approximations (constrained matrix factorizations, e.g., CUR or NMF).
A half sweep from left to right

\[ \downarrow \]

1st opt. step

\[ \downarrow \]

QR + 2nd opt. step

\[ \downarrow \]

3rd opt. step

\[ \downarrow \]

A half sweep from right to left

\[ \downarrow \]

Flip sweeping direction

\[ \downarrow \]

Figure 32: Extension of the ALS algorithm for TT decomposition. The idea is to optimize only one core tensor at a time (by a minimization of suitable cost function), while keeping the others fixed. Optimization of each core tensor is followed by an orthogonalization step via the QR or more expensive SVD decomposition. Factor matrices \( R \) are absorbed (incorporated) into the following core.

(BTD) [84], Hierarchical Outer Product Tensor Approximation (HOPTA) and Kronecker Tensor Decomposition (KTD) [85]–[87].

HOPTA and KTD models can be expressed mathematically in simple nested (hierarchical) forms, respectively (see Fig. 34 and Fig. 35):

\[
X = \sum_{r=1}^{R} (A_r \circ B_r),
\]

(41)

where each factor tensor can be represented recursively as

\[
A_r \cong \sum_{i=1}^{R} \left( A_r^{(1)} \circ B_r^{(1)} \right), \quad B_r \cong \sum_{j=1}^{R} \left( A_r^{(2)} \circ B_r^{(2)} \right),
\]

etc.

The Kronecker product of two tensors: \( A \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N} \) and \( B \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N} \) yields \( C = A \otimes B \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N} \), with entries

\[
c_{i_1, \ldots, i_N} = a_{i_1, \ldots, i_N} b_{i_1, \ldots, i_N},
\]

where the operator \( \otimes \) for indices \( i_n = 1, 2, \ldots, I_n \) and \( j_n = 1, 2, \ldots, J_n \) is defined as follows:

\[
J_n(i_n) = j_n + (i_n - 1) J_n \quad \text{(see Fig. 34)}.
\]

Note that the 2\(n\)-th order sub-tensors \( A_r \circ B_r \) and \( A_r \otimes B_r \) actually have the same elements, arranged differently. For example, if \( X = A \circ B \) and \( X' = A \otimes B \), where

\[
A \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N} \quad \text{and} \quad B \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N},
\]

then

\[
X_{i_1,j_1,\ldots,j_N} = x_{i_1' \ldots, j_N}' = x_{i_1+K_1(j_1-1), \ldots, j_N+K_N(n-1)}^f.
\]

Figure 33: Modified ALS (MALS) algorithm related to the Density Matrix Renormalization Group (DMRG) for TT decomposition. In each optimization step, two neighbor cores are merged. An optimization performed over merged “supercore”. After optimization in the next step we apply truncated SVD or other low-rank matrix factorizations (LRA) to separate the optimized supercore. For example, for nonnegative TT SVD steps can replaced by a non-negative matrix factorization (NMF) algorithm. Note that each optimization sub-problem is more expensive than the standard ALS and complexity increases but convergence speed may increase dramatically (see also [71], [72]).

Figure 34: Kronecker product of two 4th-order tensors yields a tensor \( C = A \otimes B \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_4} \), with entries

\[
c_{k_1, \ldots, k_4} = a_{i_1, \ldots, i_4} b_{j_1, \ldots, j_4},
\]

where

\[
k_n = i_n J_n - i_n \otimes J_n = j_n + (i_n - 1) J_n \quad \text{(for} \ n = 1, 2, 3, 4).\]
The model can be considered as an extension or generalization of the Block Term Decomposition (BTD) model to higher order tensors.

\[
\sum_{r=1}^{R} a_r(A_r \cdot b_r) + \beta_p(A_p \cdot B_p)
\]

\[
\sum_{p=1}^{P} (A_p \cdot B_p)
\]

\[
\sum_{q=1}^{Q} \beta_q(B_q)
\]

\[
\sum_{r=1}^{R} a_r(A_r \cdot B_r)
\]

\[
\sum_{p=1}^{P} (A_p \cdot B_p)
\]

\[
\sum_{q=1}^{Q} \beta_q(B_q)
\]

\[
\sum_{r=1}^{R} a_r(A_r \cdot B_r)
\]

\[
\sum_{p=1}^{P} (A_p \cdot B_p)
\]

\[
\sum_{q=1}^{Q} \beta_q(B_q)
\]

\[
\sum_{r=1}^{R} a_r(A_r \cdot B_r \cdot C_r)
\]

\[
\sum_{p=1}^{P} (A_p \cdot B_p \cdot C_p)
\]

\[
\sum_{q=1}^{Q} \beta_q(B_q)
\]

\[
\sum_{r=1}^{R} a_r(A_r \cdot B_r \cdot C_r)
\]

\[
\sum_{p=1}^{P} (A_p \cdot B_p \cdot C_p)
\]

\[
\sum_{q=1}^{Q} \beta_q(B_q)
\]

Figure 35: Illustration of Hierarchical Outer Product Tensor Approximation (HOPTA) for higher-order data tensors of different orders. Each component tensor: \( A_r \), \( B_r \), and/or \( C_r \) can be further decomposed using a suitable tensor network model. The model can be considered as an extension or generalization of the Block Term Decomposition (BTD) model to higher order tensors.

Figure 36: Illustration of the decomposition of an 6th-order tensor using the BTD of rank-(\( L_r, L_r, 1 \)) as: \( X = \Sigma_{r=1}^{R} A_r \circ (b_r^{(1)} \circ b_r^{(2)} \circ b_r^{(3)}) \) [36], which can be considered as a special case of the HOPTA.

It is interesting to note that the KTD and HOPTA can be considered in special cases as a flexible form of Block Term Decomposition (BTD) introduced first by De Lathauwer [36], [84], [88], [89].

The definition of the tensor Kronecker product assumes that both core tensors \( A_r \) and \( B_r \) have the same order. It should be noted that vectors and matrices can be treated as tensors, e.g., matrix of dimension \( I \times J \) can be treated formally as 3rd-order tensor of dimension \( I \times J \times 1 \). In fact, from the KTD model, we can generate many existing and emerging TDs by changing structures and orders of factor tensors: \( A_r \) and \( B_r \), for example:

- If \( A_r \) are rank-1 tensors of size \( I_1 \times I_2 \times \cdots \times I_N \), and \( B_r \) are scalars, \( \forall r \), then (42) expresses the rank-

\[
\text{R CPD.}
\]

- If \( A_r \) are rank-\( L_r \) tensors in the Kruskal (CP) format, of size \( I_1 \times I_2 \times \cdots \times I_R \times 1 \times \cdots \times 1 \), and \( B_r \) are rank-1 CP tensor of size \( 1 \times \cdots \times 1 \times I_{R+1} \times \cdots \times I_N \), \( \forall r \), then (42) expresses the rank-(\( L_r \times 1 \)) BTD [84].

- If \( A_r \) and \( B_r \) are expressed by KTDs, we have Nested Kronecker Tensor Decomposition (NKTD), where Tensor Train (TT) decomposition is a particular case [59], [60], [90]. In fact, the model (42) can be used for the recursive TT-decompositions [59].

In this way, a large variety of tensor decomposition models can be generated. However, only some of them yield unique decompositions and to date only a few have found concrete applications is scientific computing.

The advantage of HOPTA models over BTD and KTD is that they are more flexible and can approximate very high order tensors with a relative small number of cores, and they allow us to model more complex data structures.

XI. Tensorization and Quantization – Blessing of Dimensionality

A. Curse of Dimensionality

The term curse of dimensionality, in the context of tensors, refers to the fact that the number of elements of an \( N \)th-order \((I \times I \times \cdots \times I)\) tensor, \( I^N \), grows exponentially with the tensor order \( N \). Tensors can easily become really big for very high order tensors since the size is exponentially growing with the number of dimensions (ways, or modes). For example, for the Tucker decomposition the number of entries of a original data tensor but also a core tensor scales exponentially in the tensor order, for instance, the number of entries of an \( N \)th-order \((R \times R \times \cdots \times R)\) core tensor is \( R^N \).

If all computations are performed on a CP tensor format and not on the raw data tensor itself, then instead of the original \( I^N \) raw data entries, the number of parameters in a CP representation reduces to \( NRI \), which scales linearly in \( N \) and \( I \) (see Table IV). This effectively bypasses the curse of dimensionality, however the CP approximation may involve numerical problems, since existing algorithms are not stable for high-order tensors. At the same time, existing algorithms for tensor networks, especially TT/HT ensure very good numerical properties (in contrast to CPD algorithms), making it possible to control an error of approximation i.e., to achieve a desired accuracy of approximation [60].

B. Quantized Tensor Networks

The curse of dimensionality can be overcome through quantized tensor networks, which represents a tensor of possibly very high-order as a set of sparsely interconnected low-order and very low dimensions cores [60], [91]. The concept of quantized tensor networks was first proposed by Khoromskij [92] and Oseledets [91]. The very low-dimensional cores are interconnected via tensor...
Figure 37: Tensorization. (a) Illustration of the concept of tensorization of a (large-scale) vector \( \ell = 2^K \) or matrix \( (2^L \times 2^L) \) into a higher-order tensor. In order to achieve super-compression through a suitable quantized tensor decomposition (e.g., decomposition into rank-1 tensors \( X \cong \sum_{r=1}^{R} b_r^{(1)} \odot b_r^{(2)} \odots b_r^{(6)} \) or rank-\( q \) terms using Hierarchical Outer Product Tensor Approximation (HOPTA) as: \( X \cong \sum_{n=1}^{R} A_n \odot B_n \odot C_n \) or Quantized Tensor Train (QTT). (b) Symbolic representation of tensorization of the vector \( x \in \mathbb{R}^l \) into \( K \)-th-order quantized tensor \( X \in \mathbb{R}^{l \times \ell \times \ell \times \cdots \times \ell} \). (c) Tensorization of a 3rd-order tensor \( X \in \mathbb{R}^{l_1 \times l_2 \times l_3} \) into \( (K_1 + K_2 + K_3) \)-th-order tensor \( Y \in \mathbb{R}^{l_1 \times K_1 \times l_2 \times K_2 \times l_3 \times K_3} \) with \( I_{k_r} = q \).

This is the underlying principle for big data analysis [91], [92]. For example, the quantization and tensorization of a huge vector \( x \in \mathbb{R}^l \), \( l = 2^K \) can be achieved through reshaping to give an \( (2 \times 2 \times \cdots \times 2) \) tensor \( X \) of order \( K \), as illustrated in Figure 37 (a). Such a quantized tensor \( X \) often admits low-rank approximations, so that a good compression of a huge vector \( x \) can be achieved by enforcing a maximum possible low-rank structure on the tensor network.

Even more generally, an \( N \)-th order tensor \( X \in \mathbb{R}^{I_1 \times \cdots \times I_N} \), with \( I_n = q^{k_n} \), can be quantized in all modes simultaneously to yield a \( (q \times q \times \cdots \times q) \) quantized tensor \( Y \) of higher-order, with small \( q \), (see Fig. 37 (c) and Fig. 38).

In the example shown in Fig. 38 the Tensor Train of a huge 3rd-order tensor can be represented by the strong Kronecker products of block tensors with relatively small 3rd-order blocks.

Recall that the strong Kronecker product of two block cores: \( \mathbf{G}^{(n)} = \begin{bmatrix} \mathbf{G}_{1,1}^{(n)} & \cdots & \mathbf{G}_{1,k_n}^{(n)} \\ \vdots & \ddots & \vdots \\ \mathbf{G}_{k_n,1}^{(n)} & \cdots & \mathbf{G}_{k_n,k_n}^{(n)} \end{bmatrix} \in \mathbb{R}^{R_{n-1}I_{3n-2} \times R_n I_{3n-1} \times I_{3n}} \) and

\[
\mathbf{G}^{(n+1)} = \begin{bmatrix} \mathbf{G}_{1,1}^{(n+1)} & \cdots & \mathbf{G}_{1,k_n+1}^{(n+1)} \\ \vdots & \ddots & \vdots \\ \mathbf{G}_{k_n,1}^{(n+1)} & \cdots & \mathbf{G}_{k_n,k_n+1}^{(n+1)} \end{bmatrix} \in \mathbb{R}^{R_{n+1}I_{3n+1} \times R_n I_{3n+2} \times I_{3n+3}}
\]

is defined as a block tensor

\[
\mathbf{C} = \mathbf{G}^{(n)} \odot \mathbf{G}^{(n+1)} \in \mathbb{R}^{R_{n-1}I_{3n-2} \times R_n I_{3n-1} \times I_{3n+3}},
\]

with blocks \( \mathbf{G}_{r-1,r+1}^{(n)} = \sum_{r=1}^{R_{n-1}} \mathbf{G}_{r-1,r}^{(n)} \odot \mathbf{G}_{r,r+1}^{(n+1)} \) where \( \mathbf{G}_{r-1,r}^{(n)} \in \mathbb{R}^{I_{3n-2} \times I_{3n-1} \times I_{3n}} \) and \( \mathbf{G}_{r,r+1}^{(n+1)} \in \mathbb{R}^{I_{3n+2} \times I_{3n+3} \times I_{3n+3}} \) are block tensors of \( \mathbf{G}^{(n)} \) and \( \mathbf{G}^{(n+1)} \), respectively.

In practice, a fine \( (q = 2, 3, 4) \) quantization is desirable to create as many virtual modes as possible, thus allowing us to implement efficient low-rank tensor approximations. For example, the binary encoding \( (q = 2) \) reshapes an \( N \)-th order tensor with \( (2^{K_1} \times 2^{K_2} \times \cdots \times 2^{K_N}) \) elements into a tensor of order \( (K_1 + K_2 + \cdots + K_N) \), with the same number of elements. In other words, the idea of the quantized tensor is quantization of each \( n \)-th physical mode (dimension) by replacing it with \( K_n \) virtual modes, provided that the corresponding mode size \( I_n \) are factorized as \( I_n = I_{n,1}I_{n,2}I_{n,K_n} \). This corresponds to reshaping the \( n \)-th mode of size \( I_n \) into \( K_n \) modes of sizes \( I_{n,1}, I_{n,2}, \ldots, I_{n,K_n} \).

The TT decomposition applied to quantized tensors is referred to as the QTT; it was first introduced as a compression scheme for large-scale matrices [91], and also independently for more general settings [69], [74], [92]-[94]. The attractive property of QTT is that not only its rank is typically small (below 10) but it is...
assumling that

\[ \tilde{T}_1 = I_1 I_4 I_7 I_{10} \]

\[ \tilde{T}_2 = I_2 I_3 I_9 I_{11} \]

\[ \tilde{T}_3 = I_1 I_6 I_9 I_{12} \]

\[ \mathbf{X} \]

\[ \Rightarrow \]

Figure 38: Example of tensorization and decomposition of a large-scale 3rd-order tensor \( \mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3} \) into 12th-order tensor assuming that \( \tilde{T}_1 = I_1 I_4 I_7 I_{10} \), \( \tilde{T}_2 = I_2 I_3 I_9 I_{11} \) and \( \tilde{T}_3 = I_1 I_6 I_9 I_{12} \): (a) Tensorization and (b) representation of the higher-order tensor via generalized Tensor Train referred to as the Tensor Product States (TPS). The data tensor can be expressed by strong

Kronecker product of block tensors as

\[ \mathbf{X} \cong \tilde{G}^{(1)} \otimes \tilde{G}^{(2)} \otimes \tilde{G}^{(3)} \otimes \tilde{G}^{(4)} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5 \times I_6 \times I_7 \times I_8 \times I_9 \times I_{10} \times I_{11} \times I_{12} }, \]

where each block of the core

\[ \tilde{G}^{(n)} \in \mathbb{R}^{R_{n-1} \times R_n \times R_{n+1} \times I_n} \] is a 3rd-order tensor of dimensions \((I_{3n-2} \times I_{3n-1} \times I_{3n})\), with \( R_0 = R_4 = 1 \) for \( n = 1, 2, 3, 4 \).

almost independent or at least uniformly bounded by data size (even for \( I = 2^{10} \)), providing a logarithmic (sub-linear) reduction of storage requirements: \( O(I^N) \rightarrow O(N \log^4(I)) \) – so-called super-compression [92].

Compared to the TT decomposition (without quantization), the QTT format often represents more deep structure in the data by introducing some “virtual” dimensions. The high compressibility of the QTT-approximation is a consequence of the noticeable separability properties in the quantized tensor for suitably structured data.

The fact that the TT/QTT ranks are often moderate or even low, e.g., constant or growing linearly with respect to \( N \) and constant or growing logarithmically with respect to \( I \), is an important issue in the context of big data analytic and has been addressed so far mostly experimentally (see [26], [60], [92] and references therein). On the other hand, the high efficiency of multilinear algebra in the TT/QTT algorithms based on the well-posedness of the TT low-rank approximation problems and the fact that such problems are solved quite efficiently with the use of SVD/QR, CUR and other cross-approximation...
In general, tensor networks can be considered as distributed high-dimensional tensors built up from many core tensors of low dimension through specific tensor contractions. Indeed, tensor networks (TT, MPS, MPO, PEPS and HT) have already been successfully used to solve intractable problems in computational quantum chemistry and in scientific computing [63], [68], [69], [94]–[96].

However, in some cases, the ranks of the TT or QTT formats grow quite significantly with the linearly increasing of approximation accuracy. To overcome this problem, new tensor models of tensor approximation were developed, e.g., Dolgov and Khoromskij, proposed the QTT-Tucker format [74] (see Fig. 39), which exploits the TT approximation not only for the Tucker core tensor, but also QTT for the factor matrices. This model allows distributed computing, often with bounded ranks and to avoid the curse of dimensionality. For very large scale tensors we can apply a more advanced approach in which factor matrices are tensorized to higher-order tensors and then represented by TTs as illustrated in Fig. 39.

Modern methods of tensor approximations combine many TNs and TDs formats including the CPD, BTD, Tucker, TT, HT decompositions and HOPTA (see Fig. 35) low-parametric tensor format. The concept tensorization and by representation of a very high-order tensor in tensor network formats (TT/QTT, HT, QTT-Tucker) allows us to treat efficiently a very large-scale structured data that admit low rank tensor network approximations. TT/QTT/HT tensor networks have already found promising applications in very large-scale problems in scientific computing, such as eigenanalysis, super-fast Fourier transforms, and solving huge systems of large linear equations (see [26], [66], [74], [93], [97] and references therein).

In summary, the main concept or approach is to apply a suitable tensorization and quantization of tensor data and then perform approximative decomposition of this data into a tensor network and finally perform all computations (tensors/matrix/vectors operations) in tensor network formats.

### XII. Conclusions and Future Directions

Tensor networks can be considered as a generalization and extension of TDs and are promising tools for the analysis of big data due to their extremely good compression abilities and distributed and parallel processing. TDs have already found application in generalized multivariate regression, multi-way blind source separation, sparse representation and coding, feature extraction, classification, clustering and data assimilation. Unique

---

**TABLE IV:** Storage complexities of tensor decomposition models for an $N$-th-order tensor $\mathbf{X} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$, for which the original storage complexity is $O(I^N)$, where $I = \max\{I_1, I_2, \ldots, I_N\}$, while $R$ is an upper bound on the ranks of tensor decompositions considered, that is $R = \max\{R_1, R_2, \ldots, R_{N-1}\}$ or $R = \max\{R_1, R_2, \ldots, R_N\}$.

<table>
<thead>
<tr>
<th>Model</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. CPD</td>
<td>$O(NIR)$</td>
</tr>
<tr>
<td>2. Tucker</td>
<td>$O(NIR + R^N)$</td>
</tr>
<tr>
<td>3. TT/MPS</td>
<td>$O(NIR^2)$</td>
</tr>
<tr>
<td>4. TT/MPO</td>
<td>$O(NIR^2)$</td>
</tr>
<tr>
<td>5. Quantized TT/MPS (QTT)</td>
<td>$O(NR^2 \log_q(I))$</td>
</tr>
<tr>
<td>6. QTT+Tucker</td>
<td>$O(NR^2 \log_q(I) + NR^3)$</td>
</tr>
<tr>
<td>7. Hierarchical Tucker (HT)</td>
<td>$O(NIR + NR^3)$</td>
</tr>
</tbody>
</table>

Figure 39: QTT-Tucker format. (a) Distributed representation of a large matrix $\mathbf{A}_n \in \mathbb{R}^{R_n \times R_n}$ with large dimension of $I_n$ via QTT by tensorization of the matrix to high-order quantized tensor and next by performing QTT decomposition. (b) Distributed representation of a large-scale $N$-th-order Tucker model via a quantized TT model in which core tensor and all large-scale factor matrices $\mathbf{A}_n \ (n = 1, 2, \ldots, N)$ are represented by tensor trains [74].
advantages of tensor networks include potential ability of tera- or even peta-byte scaling and distributed fault-tolerant computations.

Overall, the benefits of multiway (tensor) analysis methods can be summarized as follows:

- “Super” compression of huge multidimensional, structured data which admits a low-rank approximation via TNs of high-order tensors by extracting factor matrices and/or core tensors of low-rank and low-order and perform all mathematical manipulations in tensor formats (especially, TT and HT formats).

- A compact and very flexible approximate representation of structurally rich data by accounting for their spatio-temporal and spectral dependencies.

- Opportunity to establish statistical links between cores, factors, components or hidden latent variables for blocks of data.

- Possibility to operate with noisy, incomplete, missing data by using powerful low-rank tensor/matrix approximation techniques.

- A framework to incorporate various diversities or constraints in different modes and thus naturally extend the standard (2-way) CA methods to large-scale multidimensional data.

Many challenging problems related to low-rank tensor approximations remain to be addressed.

- A whole new area emerges when several TNs which operate on different datasets are coupled or linked.

- As the complexity of big data increases, this requires more efficient iterative algorithms for their computation, extending beyond the ALS, MALS/DMRG, SVD/QR and CUR/Cross-Approximation class of algorithms.

- Methodological approaches are needed to determine the kind of constraints should that be imposed on cores to extract desired hidden (latent) variables with meaningful physical interpretation.

- We need methods to reliably estimate the ranks of TNs, especially for structured data corrupted by noise and outliers.

- The uniqueness of various TN models under different constraints needs to be investigated.

- Special techniques are needed for distributed computing and to save and process huge ultra-large-scale tensors.

- Better visualization tools need to be developed to address large-scale tensor network representations.

In summary, TNs and TDs is a fascinating and perspective area of research with many potential applications in multi-modal analysis of massive big data sets.

Acknowledgments: The author wish to express his appreciation and gratitude to his colleagues, especially: Drs. Namgill LEE, Danilo MANDIC, Qibin ZHAO, Cesar CAIAFA, Anh Huy PHAN, André ALMEIDA, Qiang WU, Guoxu ZHOU and Chao LI for reading the manuscript and giving him comments and suggestions.

REFERENCES


