A TT-Based Hierarchical Framework for Decomposing High-Order Tensors

Yassine Zniyed, Remy Boyer, André L. F. de Almeida, Gérard Favier

A TT-BASED HIERARCHICAL FRAMEWORK FOR DECOMPOSING HIGH-ORDER TENSORS

YASSINE ZNIYED*, RÉMY BOYER†, ANDRÉ L. F. DE ALMEIDA‡, AND GÉRARD FAVIER§

Abstract. In the context of big data, high-order tensor decompositions have to face a new challenge in terms of storage and computational costs. The tensor train (TT) decomposition provides a very useful graph-based model reduction, whose storage cost grows linearly with the tensor order D. The computation of the TT-core tensors and TT-ranks can be done in a stable sequential (*i.e.*, non-iterative) way thanks to the popular TT-SVD algorithm. In this paper, we exploit the ideas developed for the hierarchical/tree Tucker decomposition in the context of the TT decomposition. Specifically, a new efficient estimation scheme, called TT-HSVD for Tensor-Train Hierarchical SVD, is proposed as a solution to compute the TT decomposition of a high-order tensor. The new algorithm simultaneously delivers the TT-core tensors and their TT-ranks in a hierarchical way. It is a stable (*i.e.*, non-iterative) and computationally more efficient algorithm than the TT-SVD one, which is very important when dealing with large-scale data. The TT-HSVD algorithm uses a new reshaping strategy and a tailored partial SVD, which allows to deal with smaller matrices compared to those of the TT-SVD. In addition, TT-HSVD suits well for a parallel processing architecture. An algebraic analysis of the two algorithms is carried out, showing that TT-SVD and TT-HSVD compute the same TT-ranks and TT-core tensors up to specific bases. Simulation results for different tensor orders and dimensions corroborate the effectiveness of the proposed algorithm.

Key word. Tensor Train, hierarchical SVD, dimensionality reduction, tensor graph

1. Introduction.

Massive and heterogeneous data processing and analysis have been clearly identified by the scientific community as key problems in several application areas [11, 12, 13]. It was popularized under the generic terms of "data science" or "big data". Processing large volumes of data, extracting their hidden patterns, while performing prediction and inference tasks has become crucial in economy, industry and science. Modern sensing systems exploit simultaneously different physical technologies. Assume that D specific sensing devices are available, to measure D different parameters, or "modalities", in a heterogeneous dataset. Treating independently each set of measured data, or each modality, is clearly a reductive approach. By doing that, "hidden relationships" or inter-correlations between the modes of the dataset may be totally missed. Tensor decompositions have received a particular attention recently due to their capability to handle a variety of mining tasks applied to massive datasets, being a pertinent framework taking into account the heterogeneity and multi-modality of the data. In this case, data can be arranged as a D-dimensional array, also referred to as a D-order tensor, and denoted by $\boldsymbol{\mathcal{X}}$. In the context of big data processing and analysis, the following properties are desirable: (*i*) a stable (*i.e.*, non-iterative) recovery algorithm, (*ii*) a low storage cost (*i.e.*, the number of free parameters must scale linearly with D), and (*iii*) an adequate graphical formalism allowing a simple but rigorous visualization of the decomposition of tensors with $D > 3$.

In the literature [26, 36, 34, 58], two decompositions are popular, sharing a common goal, *i.e.*, to decompose a tensor $\boldsymbol{\mathcal{X}}$ into a D-order core tensor $\boldsymbol{\mathcal{G}}$ and D factor matrices. Due to the diagonality of the core tensor, the CPD [30, 28, 9] extends the

*Laboratoire des Signaux et Systèmes (L2S), CNRS, Université Paris-Sud (UPS), CentraleSupelec, Gif-sur-Yvette, France. (yassine.zniyed@l2s.centralesupelec.fr).

†CRIStAL, Université de Lille, Villeneuve d'Ascq, France. (remy.boyer@univ-lille.fr).

‡Department of Teleinformatics Engineering, Federal University of Fortaleza, Brazil. (andre@gtel.ufc.br).

§Laboratoire I3S, Université Côte d'Azur, CNRS, Sophia Antiplois, France. (favier@i3s.unice.fr).

definition of rank used in linear algebra to multilinear algebra. Indeed, the canonical rank of a tensor $\boldsymbol{\mathcal{X}}$ is defined as the smallest number R of rank-1 tensors necessary to yield $\boldsymbol{\mathcal{X}}$ in a linear combination. As a consequence, the number of free parameters (as well as the storage cost) of the CPD grows linearly with the tensor order D. Unfortunately, without any additional structural assumptions [23, 67, 54, 52, 7], the recovery of (i) the canonical rank is NP-hard [29] and (ii) the factor matrices is ill-posed [17, 35]. This means that (i) there is no guarantee on the existence of a low-rank approximation of a tensor by means of its rank-R CPD and (ii) a stable and efficient algorithm to compute the factor matrices may not exist, or it performs poorly when dealing with big data tensors. The HOSVD [41, 1] is the second approach for decomposing a high-order tensor. In this case, the D factor matrices are obtained from a low-rank approximation of the unfolding matrices of the tensor, which is possible by means of the SVD, under orthonormality constraint on the factor matrices. Unfortunately, this orthonormality constraint implies that the core tensor $\boldsymbol{\mathcal{G}}$ is generally not diagonal. Two remarks can be made at this point. First, unlike the SVD [21], the HOSVD is a multilinear rank-revealing decomposition [4] but does not reveal the *canonical* rank. Second, the storage cost associated with the computation of the core tensor grows exponentially with the order D of the data tensor. From this brief panorama, we can conclude that the CPD and the HOSVD are not the appropriate solutions to deal with high-order big data tensors, and more efficient decompositions should be considered.

Recently, a new paradigm for model reduction has been proposed in [48], therein referred to as the tensor train (TT) decomposition and before in the physic of particules community [46]. The main idea of TT, also known as "tensor networks" (TNs) [3, 12] is to split a high-order ($D > 3$) tensor into a product set of lower-order tensors represented as a factor graph. Factor graphs allow visualizing the factorization of multi-variate or multi-linear functions (the nodes) and their dependencies (the edges) [42]. The graph formalism is useful in the big data context [56]. Particularly, the TT decomposition [48, 47, 26] represents a D-order tensor by a product set of D 3-order tensors. Each 3-order tensor is associated with a node of the graph and is connected to its left and right "neighbors" encoded in a one-to-one directional edge [46]. The storage cost with respect to the order D has the same linear behavior [48] for the CPD and the TT decomposition. Moreover, the TT decomposition has a stable (non-iterative) SVD-based algorithm [48], even for large tensors. Therefore, the TT decomposition helps to break the *curse of dimensionality* [49], as it can be seen as a special case of the hierarchical/tree Tucker (HT) decomposition [25, 26, 65, 60, 49]. Note that the HT and the TT decompositions allow to represent a D-order tensor of size $I \times \cdots \times I$ with a storage cost of $O(DIR + DR^3)$ and $O(DIR^2)$, respectively [24, 11], where R is the rank of the considered decomposition. The use of the TT decomposition is motivated by its applicability in a number of interesting problems, such as super-compression [33], tensor completion [38], blind source separation [5], fast SVD of large scale matrices [43], for linear diffusion operators [32], and machine learning [58], to mention a few. Another class of methods consists of data sub-division of the data tensor into smaller "blocks", followed by efficient computations over smaller sub-tensors [51, 15, 16, 44]. These methods focus only on the CPD of a high-order tensor.

In this paper, our interest is on the TT decomposition and the associated TT-SVD algorithm [48]. It is worth noting that TT-SVD is a sequential algorithm, *i.e.*, the TT-cores are computed one after the other and not at the same time. Moreover, it is a very costly algorithm in terms of complexity, since it involves the application of SVD

to matrices of very large sizes. To tackle the computational complexity problem associated with the decomposition of large-scale tensors, a number of methods have been proposed [13], which either replace non-iterative methods by closed-form ones when the tensor is structured [1], or exploit sparsity of the data [37, 50] or, yet, reduce the size of the data using compression with parallel processing [45, 59]. As an alternative to the TT-SVD algorithm, we propose a new algorithm called TT-HSVD algorithm, for Tensor-Train Hierarchical SVD. This algorithm allows to simultaneously recover the TT-core tensors and their TT-ranks in a hierarchical way and is computationally more efficient than TT-SVD. The proposed TT-HSVD algorithm adopts a new unfolding and reshaping strategy that, on one hand, enables to parallelize the decomposition across several processors and, on the other hand, results in a less expensive computational cost compared to competing solutions based on the TT-SVD. Our algebraic analysis also shows that TT-SVD and TT-HSVD compute the same TT-ranks and TT-core tensors up to specific bases.

This paper is organized as follows. In the next section, we introduce the notations and some important algebraic definitions. In Section 3, we give an overview of standard tensor decompositions. In Section 4, after briefly recalling the TT-SVD algorithm, we formulate the proposed TT-HSVD one. In Section 5, an algebraic analysis of both methods is made. In Section 6, the computational complexity of the two algorithms is compared, and illustrated by means of some simulation results. Finally, in Section 7, the paper is concluded and some perspectives for future work are drawn.

2. Notations and algebraic background. The notations used throughout this paper are the following ones: the superscripts $(\cdot)^T$ and $(\cdot)^\dagger$, and rank$(\cdot)$ denote, respectively, the transpose, the pseudo-inverse, and the rank. The Kronecker, outer, and n-mode products are denoted as $\otimes$, $\circ$, and $\times_n$, respectively. The Frobenius norm is defined by $||\cdot||_F$. $\kappa(\cdot)$ refers to the dominant term of a complexity. Scalars, vectors, matrices and tensors are represented by x, $\mathbf{x}$, $\mathbf{X}$ and $\boldsymbol{\mathcal{X}}$, respectively. The $(i_1, i_2, \cdots, i_D)$-th entry of the D-order tensor $\boldsymbol{\mathcal{X}}$ is denoted as $\boldsymbol{\mathcal{X}}(i_1, i_2, \cdots, i_D)$, while $\mathbf{X}(i,:)$ and $\mathbf{X}(:,j)$ are the i-th row and the j-th column of $\mathbf{X} \in \mathbb{R}^{I \times J}$, respectively.

DEFINITION 2.1. *The $\times_n^m$ tensor-tensor product of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$, where $I_n = J_m$, gives the $(N+M-2)$-order tensor defined as [11]:*

$$\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} \times_n^m \boldsymbol{\mathcal{Y}} \quad \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$$

where the entries of $\boldsymbol{\mathcal{Z}}$ are expressed as:
$\boldsymbol{\mathcal{Z}}(i_1, \cdots, i_{n-1}, i_{n+1}, \cdots, i_N, j_1, \cdots, j_{m-1}, j_{m+1}, \cdots, j_M) =$
$\sum_{i=1}^{I_n} \boldsymbol{\mathcal{X}}(i_1, \cdots, i_{n-1}, i, i_{n+1}, \cdots, i_N) \boldsymbol{\mathcal{Y}}(j_1, \cdots, j_{m-1}, i, j_{m+1}, \cdots, j_M)$.

The $\times_n^m$ product is a generalization of the usual matrix product.

Indeed, $\mathbf{A}$ and $\mathbf{B}$ being two matrices of respective dimensions $I_1 \times I_2$ and $J_1 \times J_2$, with $I_2 = J_1$, we have:

$$\mathbf{A} \times_2^1 \mathbf{B} = \mathbf{AB}.$$

In this paper, factor graph representations will be intensively used. In a factor graph, a node can be a vector, a matrix or a tensor as illustrated in Fig. 1-$(a), (b), (c)$. The edge encodes the dimension of the node. The order of the node is given by the number of edges. In Fig. 1-(d), the $\times_3^1$ product of two 3-order tensors with a common dimension is illustrated.

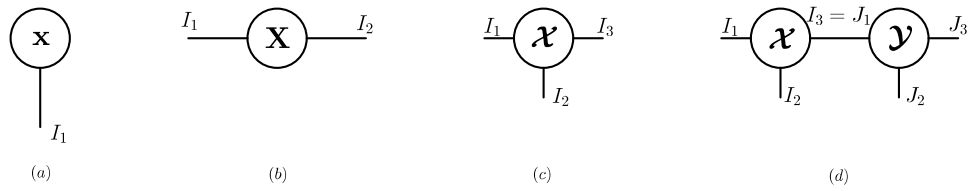

FIG. 1. *(a) Vector* $\mathbf{x}$ *of size* $I_1 \times 1$*, (b) matrix* $\mathbf{X}$ *of size* $I_1 \times I_2$*, (c) 3-order tensor* $\boldsymbol{\mathcal{X}}$ *of size* $I_1 \times I_2 \times I_3$*, (d)* $\times_3^1$ *product of two 3-order tensors.*

The truncated SVD [21], also known as low rank approximation, will be also intensively used. Note that the famous Eckart-Young theorem [18] provides a proof of existence of the best approximation (in the sense of the Frobenius norm) of a given matrix by another matrix of smaller rank. In addition, this theorem ensures that the best low rank approximation can be computed in a stable, *i.e.*, non-iterative, way by means of the truncated SVD.

3. Standard tensor decompositions.

3.1. Canonical Polyadic and Tucker decompositions.

DEFINITION 3.1. *A D-order tensor of size* $I_1 \times \ldots \times I_D$ *that follows a Canonical Polyadic Decomposition (CPD) [28] of rank-R admits the following definition:*

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{M}_1(:,r) \circ \mathbf{M}_2(:,r) \circ \ldots \circ \mathbf{M}_D(:,r)$$

where the d-th factor $\mathbf{M}_d$ *is of size* $I_d \times R$ *with* $1 \leq d \leq D$*. The canonical rank* R *is defined as the smallest number of rank-one tensors,* $\mathbf{M}_1(:,r) \circ \mathbf{M}_2(:,r) \circ \ldots \circ \mathbf{M}_D(:,r)$*, that exactly generate* $\boldsymbol{\mathcal{X}}$ *as their sum over index* r*. The storage cost of a CPD is* $O(DIR)$ *where* $I = \max\{I_1, \cdots, I_D\}$ *and thus is linear in the order* D*.*

DEFINITION 3.2. *A D-order tensor of size* $I_1 \times \ldots \times I_D$ *that follows a Tucker decomposition [63] of multilinear rank-*$(R_1, \cdots, R_D)$ *admits the following definition:*

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{F}_1 \times_2 \mathbf{F}_2 \ldots \times_D \mathbf{F}_D$$

where the d-th factor $\mathbf{F}_d$ *is of size* $I_d \times R_d$ *and* $\boldsymbol{\mathcal{G}}$ *is called the core tensor of size* $R_1 \times \ldots \times R_D$*. The storage cost of a Tucker decomposition is* $O(DIR + R^D)$ *where* $R = \max\{R_1, \cdots, R_D\}$*. Note that this cost grows exponentially with the order* D*. This is usually called the "curse of dimensionality".*

3.2. Tensor Network and Tensor Train for model reduction. The main idea in model reduction is to split a high-order tensor into a collection of lower order ones. Usually, such a lower order is at most equal to 3.

3.2.1. Graph-based illustrations of TN decomposition. Fig. 2-(a) gives the graph-based representation of a Tucker decomposition of a 6-order tensor. Its graph-based decompositions are given in Fig(s). 2-(b). The decomposition of the initial tensor is not unique since several graph-based configurations are possible depending on the tensor order and the number of connected nodes. In the literature,

the graph on the right of Fig. 2-(b) is viewed as a Tensor Network (TN), also called Hierarchical Tucker (HT) decomposition [27]. This is due to the property that a TN can be alternatively represented as a tree (see Fig. 2-(c)) where the nodes are called leafs. The graph-based representation on the left of Fig. 2-(b) is viewed as a train of tensors with non-identity leafs. We call a *Tensor Train* (TT) decomposition [49, 48] if all the leafs are associated with an identity matrix except the first and last ones. The TT decomposition is one of the simplest and compact TN, and it is a special case of the HT decomposition [65, 25, 26]. Hereafter, we give the definition of the TT decomposition, and its graph-based representation is illustrated in Fig. 3.

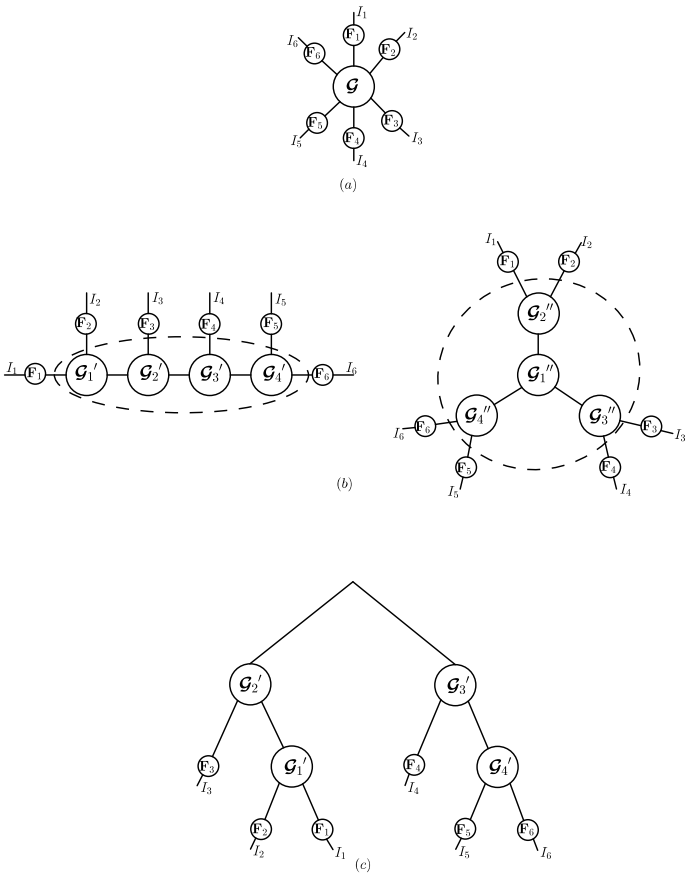

FIG. 2. *(a) Graph-based Tucker decomposition of a* 6*-order tensor, (b) TT-based (*left*) and TN-based (*right*) decompositions, (c) HT decomposition.*

3.2.2. TT decomposition and tensor reshaping.

DEFINITION 3.3. *The TT decomposition [48] with TT-ranks* $(R_1, \ldots, R_{D-1})$ *of a tensor* $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ *into* D 3*-order TT-core tensors denoted by* $\{\boldsymbol{\mathcal{G}}_1, \ldots, \boldsymbol{\mathcal{G}}_D\}$

is given by

$$\begin{aligned}\boldsymbol{\mathcal{X}}(i_1, i_2, \cdots, i_D) &= \boldsymbol{\mathcal{G}}_1(:, i_1, :)\boldsymbol{\mathcal{G}}_2(:, i_2, :)\cdots\boldsymbol{\mathcal{G}}_D(:, i_D, :) \\ &= \sum_{r_1,\cdots,r_{D-1}=1}^{R_1,\cdots,R_{D-1}} \mathbf{G}_1(i_1, r_1)\boldsymbol{\mathcal{G}}_2(r_1, i_2, r_2)\cdots \\ &\qquad \cdots\boldsymbol{\mathcal{G}}_{D-1}(r_{D-2}, i_{D-1}, r_{D-1})\mathbf{G}_D(r_{D-1}, i_D).\end{aligned}$$

where $\boldsymbol{\mathcal{G}}_d(:, i_d, :)$ is a $R_{d-1} \times R_d$ matrix with the "boundary conditions" $R_0 = R_D = 1$. Finally, collecting all the entries[1]*, we have*

$$\boldsymbol{\mathcal{X}} = \mathbf{G}_1 \times_2^1 \boldsymbol{\mathcal{G}}_2 \times_3^1 \boldsymbol{\mathcal{G}}_3 \times_4^1 \cdots \times_{D-1}^1 \boldsymbol{\mathcal{G}}_{D-1} \times_D^1 \mathbf{G}_D. \tag{1}$$

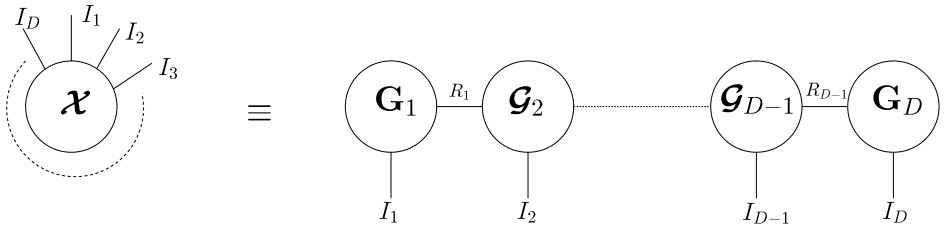

FIG. 3. *TT decomposition of a D-order tensor.*

In tensor-based data processing, it is standard to unfold a tensor into matrices. We refer to Eq. (5) in [20], for a general matrix unfolding formula, also called tensor reshaping. The d-th reshaping $\mathbf{X}_{(d)}$, of size $(\prod_{s=1}^{d} I_s) \times (\prod_{s=d+1}^{D} I_s)$, of the tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ using the native reshape function of the software `MATLAB` [39], is defined by:

$$\mathbf{X}_{(d)} = \text{reshape}\left(\boldsymbol{\mathcal{X}}, \prod_{l=1}^{d} I_l, \prod_{l=d+1}^{D} I_l\right). \tag{2}$$

DEFINITION 3.4. *The d-th reshaping $\mathbf{X}_{(d)}$ of the tensor $\boldsymbol{\mathcal{X}}$ which follows the TT decomposition (1) admits the following expression:*

$$\mathbf{X}_{(d)} = \sum_{r_1,\cdots,r_{D-1}=1}^{R_1,\cdots,R_{D-1}} \Big(\mathbf{g}_d(r_{d-1}, r_d) \otimes \ldots \otimes \mathbf{g}_1(r_1)\Big)\Big(\mathbf{g}_D^T(r_{D-1}) \otimes \cdots \otimes \mathbf{g}_{d+1}^T(r_d, r_{d+1})\Big) \tag{3}$$

where $\mathbf{g}_1(r_1) = \mathbf{G}_1(:, r_1)$, $\mathbf{g}_D(r_{D-1}) = \mathbf{G}_D(r_{D-1}, :)^T$ and $\mathbf{g}_d(r_{d-1}, r_d) = \boldsymbol{\mathcal{G}}_d(r_{d-1}, :, r_d)$ are column-vectors of length I_1, I_D and I_d, respectively. An alternative expression of (1) in terms of sum of outer products of vectors is given by:

$$\boldsymbol{\mathcal{X}} = \sum_{r_1,\cdots,r_{D-1}=1}^{R_1,\cdots,R_{D-1}} \mathbf{g}_1(r_1) \circ \mathbf{g}_2(r_1, r_2) \circ \cdots \circ \mathbf{g}_{D-1}(r_{D-2}, r_{D-1}) \circ \mathbf{g}_D(r_{D-1}).$$

[1] Note that the product $\times_n^m$ is used here in a different way as in [11].

3.2.3. Why to use the TT decomposition. There are four main motivations for using a TT decomposition:

- The TT decomposition has a unique graph-based representation for a tensor of known order, *i.e.*, any D-order tensor can decomposed into the TT format as a product set of D TT-cores of order at most 3, with respect to the one configuration where all nodes of the underlying tensor network are aligned. This is not the case for the HT decomposition as illustrated by Fig. 2-(c). The number of possible configurations rapidly grows with the order D. For instance, a 10-order tensor admits 11 different HT decompositions [39].
- The ranks for HT are upper bounded by R^2 if the TT-ranks are upper bounded by R [25]. If equal ranks is assumed and I is not too large, the number of free parameters in the TT format is smaller than that in the HT format due to the presence of leafs different from the identity matrix in the HT format.
- The TT decomposition storage grows linearly with D, whereas the HOSVD storage grows exponentially with D, with an important computational cost.
- The TT decomposition has a compact form, unlike the tree-like decompositions, such as Tree Tucker [49], that requires recursive algorithms based on the competition of Gram matrices, which is complicated to implement [48].

4. Hierarchical methodology to compute the TT decomposition. In this section, before presenting our main contribution, we first recall the TT-SVD algorithm [49].

4.1. Description of the TT-SVD algorithm. In the practice, the TT decomposition is performed thanks to the TT-SVD algorithm [49]. The complete algorithm is described in Fig. 4 for a 4-order tensor. Note that when we apply the truncated SVD on matrix $\mathbf{X}_{(d)}$, the diagonal matrix constituted by the singular values is absorbed in $\mathbf{V}_d$.

From this simple example, one can conclude that the TT-SVD algorithm consists in sequentially truncating the SVD of a reshaped version of the matrices $\mathbf{V}_d$, for $d = 1, \cdots, D-1$. It is important to note that each step of the algorithm yields a TT-core, leading to a sequential algorithm which cannot be parallelized.

4.2. Description of the TT-HSVD algorithm. In this section, the TT-HSVD algorithm is presented, with the aim to derive the TT-cores in a parallel hierarchical way. The main difference between the TT-SVD and TT-HSVD algorithms lies in the initial matrix unfolding to be processed by the SVD, and the reshaping strategy, i.e. the way to reshape the SVD factors $(\mathbf{U}_d, \mathbf{V}_d)$, at each step. Fig. 5 illustrates the proposed strategy by means of the graph-based representation of the TT decomposition of a 4-order tensor. This figure is to be compared with Fig. 4. With the TT-HSVD algorithm, for an *a priori* chosen index $\bar{D} \in \{1, \ldots, D\}$, the first matrix unfolding $\mathbf{X}_{(\bar{D})}$ is of size $(I_1 \cdots I_{\bar{D}}) \times (I_{\bar{D}+1} \cdots I_D)$ instead of $I_1 \times (I_2 \cdots I_D)$ as for the TT-SVD algorithm, which leads to a more rectangular matrix. Its $R_{\bar{D}}$-truncated SVD provides two factors $\mathbf{U}_{\bar{D}}$ and $\mathbf{V}_{\bar{D}}$ of size $(I_1 \cdots I_{\bar{D}}) \times R_{\bar{D}}$ and $R_{\bar{D}} \times (I_{\bar{D}+1} \cdots I_D)$, respectively. These two factors are now reshaped in parallel, which constitutes the main difference with the TT-SVD algorithm for which only a single reshaping operation is applied to $\mathbf{V}_1$. This processing is repeated after each SVD computation, as illustrated in Fig. 5 for a 4-order tensor.

Generally speaking, the choice of the best reshaping strategy, *i.e.*, the choice of the index $\bar{D}$, is depending on an *a priori* physical knowledge related to each application

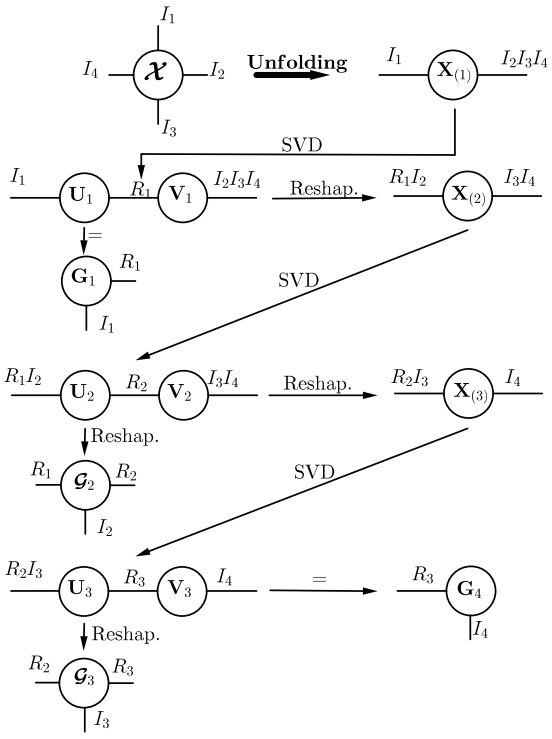

FIG. 4. *TT-SVD applied to a 4-order tensor.*

[36, 6], as for instance, in bio-medical signal analysis, or in wireless communications [14, 66, 20]. In Section 7, this best choice is discussed in terms of algorithmic complexity. To illustrate this choice, two reshaping strategies are considered in Fig. 6 (left) and Fig. 6 (right) for computing the TT decomposition of a 8-order tensor with the TT-HSVD algorithm. More precisely, Fig. 6 (left) corresponds to a balanced unfolding with $I_{\bar{D}} = I_4$, while Fig. 6 (right) corresponds to an unbalanced unfolding with $I_{\bar{D}} = I_3$. From this simple example, one can conclude that this graph-based representation is not illustrative, which motivated the new graph-based representation using patterns, introduced in the next section.

4.3. Graph-based representation with patterns. A group of algorithmic instructions can be identified as recurrent in the TT-SVD and TT-HSVD algorithms. Such a group will be called a **pattern**. The concept of pattern is introduced in this work as a useful graphical tool to model the processing steps of TT decomposition algorithms in an illustrative way. It also facilitates understanding of the main steps of the TT-HSVD algorithm, although it can also be used to describe any algorithm. Three types of pattern are now described.

4.3.1. Splitting/Splitting pattern.
The Splitting/Splitting pattern takes as input any matrix unfolding $\mathbf{X}_{(\bar{D})}$ of size $R_{\bar{D}_f}(I_{\bar{D}_f+1}I_{\bar{D}_f+2}\cdots I_{\bar{D}}) \times (I_{\bar{D}+1}I_{\bar{D}+2}\cdots I_{\bar{D}_l})R_{\bar{D}_l}$, where $\bar{D}_f$ stands for the first index and $\bar{D}_l$ for the last index. This pattern applies the SVD to the input matrix and generates a matrix $\mathbf{U}_{\bar{D}}$, of size $R_{\bar{D}_f}(I_{\bar{D}_f+1}I_{\bar{D}_f+2}\cdots I_{\bar{D}}) \times R_{\bar{D}}$, which contains the left singular vectors, and a matrix $\mathbf{V}_{\bar{D}}$, of size $R_{\bar{D}} \times (I_{\bar{D}+1}I_{\bar{D}+2}\cdots I_{\bar{D}_l})R_{\bar{D}_l}$, equal to

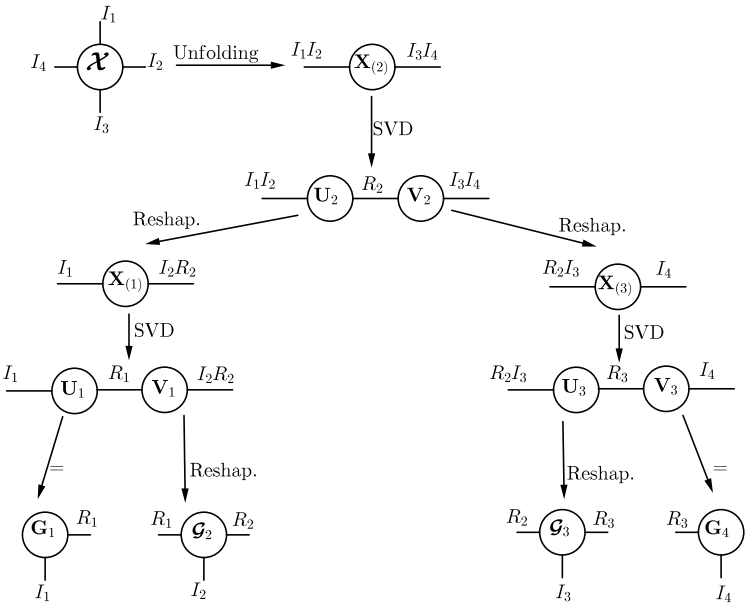

FIG. 5. *TT-HSVD applied to a* 4*-order tensor.*

the product of the diagonal singular values matrix with the matrix composed of the right singular vectors. These two factors $(\mathbf{U}_{\bar{D}}, \mathbf{V}_{\bar{D}})$ are then reshaped using two new indices $\bar{D}'$ and $\bar{D}''$. This pattern is represented in Fig. 7. The corresponding graph is characterized by one input $\mathbf{X}_{(\bar{D})}$, two indices $\bar{D}'$ and $\bar{D}''$, and two outputs $\mathbf{X}_{(\bar{D}_f+\bar{D}')}$ and $\mathbf{X}_{(\bar{D}+\bar{D}'')}$. This pattern plays the role of data preparation, before generating the desired TT-cores using other patterns that will be called core generation patterns. One can notice that this pattern is always used at the top of the TT-HSVD algorithm, with $\bar{D}_f = 0$, and $\bar{D}_l = D$, the tensor order. (See Figs. 5, 6 (left) and 6 (right)).

4.3.2. Mixed patterns: data processing and TT-core generation. The first mixed pattern, that will be called Splitting/Generation pattern, takes as input any matrix $\mathbf{X}_{(\bar{D})}$ of size $R_{\bar{D}_f}(I_{\bar{D}_f+1}I_{\bar{D}_f+2}\cdots I_{\bar{D}}) \times (I_{\bar{D}+1}R_{\bar{D}+1})$, and returns a reshaped matrix and a tensor. It has the same structure as the splitting/splitting pattern, except that it generates a matrix and a tensor instead of two matrices. This pattern is represented in Fig. 8 (left). For example, this pattern can be seen in Fig. 6 (right) as the function that takes as input $\mathbf{X}_{(2)}$ and generates $\mathbf{X}_{(1)}$ and $\boldsymbol{\mathcal{G}}_3$. The second mixed pattern, that will be called Generation/Splitting pattern, takes as input any matrix $\mathbf{X}_{(\bar{D})}$ of size $(R_{\bar{D}-1}I_{\bar{D}}) \times (I_{\bar{D}+1}I_{\bar{D}+2}\cdots I_{\bar{D}_l})R_{\bar{D}_l}$, and returns a tensor and a reshaped matrix. This pattern is represented in Fig. 8 (right). For example, this pattern can be seen in Fig. 6 (right) as the function that takes as input $\mathbf{X}_{(6)}$ and generates $\boldsymbol{\mathcal{G}}_6$ and $\mathbf{X}_{(7)}$.

4.3.3. TT-core generation pattern. The TT-core generation pattern generates two core tensors in parallel. It will be called Generation/Generation pattern. It

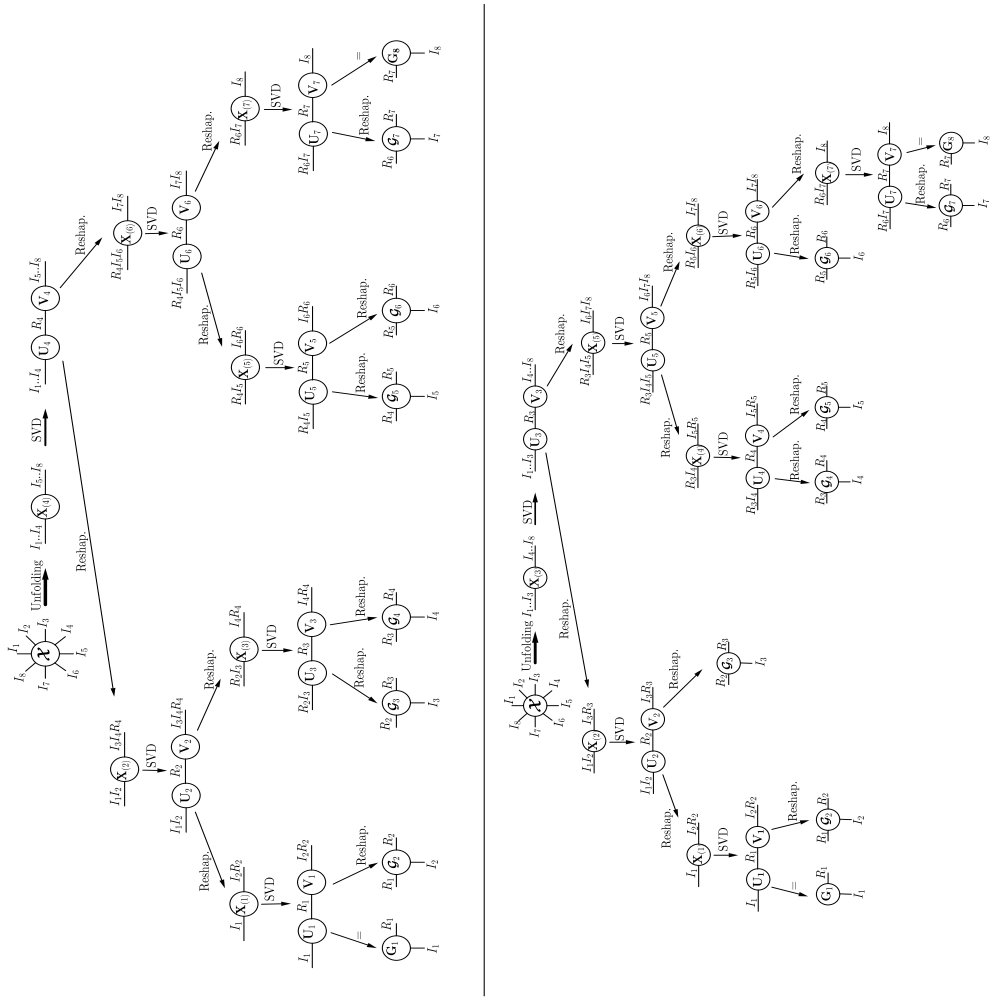

FIG. 6. *Balanced (left) and Unbalanced (right) TT-HSVD applied to a* 8*-order tensor.*

is represented in Fig. 9. It takes as input $\mathbf{X}_{(\bar{D})}$ of size $(R_{\bar{D}-1}I_{\bar{D}}) \times (I_{\bar{D}+1}R_{\bar{D}+1})$ and returns two core tensors as outputs. For example, this pattern can be recognized in Fig. 6 (right) as the function that takes as input $\mathbf{X}_{(4)}$ and generates $\boldsymbol{\mathcal{G}}_4$ and $\boldsymbol{\mathcal{G}}_5$.

4.4. Application of the pattern formalism. Note that the TT-SVD algorithm uses one Splitting/Splitting pattern at the beginning and then a sequence of Generation/Splitting patterns, while the three different patterns can be used for the TT-HSVD algorithm. Figs.10 and 11 illustrate the pattern-based representation of the TT-HSVD algorithm applied to an 8-order tensor, in the balanced and unbalanced cases, respectively. These figures are to be compared with Figs. 6 (left) and 6 (right).

A pseudo-code of the TT-HSVD based on the patterns formalism is presented in Algorithm 1.

5. Algebraic analysis of the TT-SVD and TT-HSVD algorithms. From an algorithmic point of view, the TT-HSVD algorithm is based on a different reshaping-

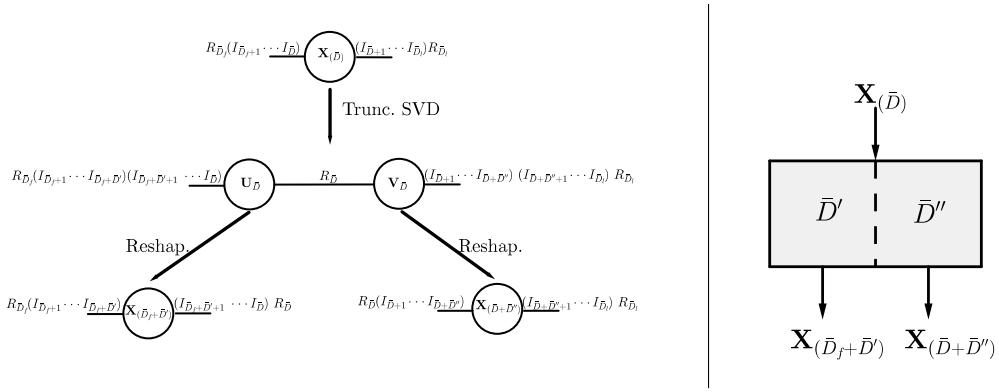

FIG. 7. *Splitting/Splitting pattern.*

Algorithm 1 TT-HSVD algorithm

Input: D-order tensor $\boldsymbol{\mathcal{X}}$, set of indices $\bar{D}, \bar{D}^{'}$ and $\bar{D}^{''}$.

Output: TT-cores: $\hat{\mathbf{G}}_1^{\mathtt{hrl}}, \hat{\boldsymbol{\mathcal{G}}}_2^{\mathtt{hrl}}, \cdots \hat{\boldsymbol{\mathcal{G}}}_{D-1}^{\mathtt{hrl}}, \hat{\mathbf{G}}_D^{\mathtt{hrl}}$.

1: Apply a splitting/splitting pattern to the matrix $\mathbf{X}_{(\bar{D})}$ of size $(I_1 I_2 \cdots I_{\bar{D}}) \times (I_{\bar{D}+1} I_{\bar{D}+2} \cdots I_D)$ with respect to indices $\bar{D}^{'}$ and $\bar{D}^{''}$.

2: **while** $\mathbf{X}_{(\bar{D})}$ is of size $R_{\bar{D}_f}(I_{\bar{D}_f+1} I_{\bar{D}_f+2} \cdots I_{\bar{D}}) \times (I_{\bar{D}+1} I_{\bar{D}+2} \cdots I_{\bar{D}_l}) R_{\bar{D}_l}$, with $\bar{D}_f + 1 \neq \bar{D}$ or $\bar{D} + 1 \neq \bar{D}_l$[2]:
Choose to apply either
- a splitting/splitting pattern (*cf.* Section 4.3.1),
- a generation/splitting pattern (*cf.* Section 4.3.2),
- or a splitting/generation pattern (*cf.* Section 4.3.2)

to this matrix with respect to indices $\bar{D}^{'}$ and $\bar{D}^{''}$[3].
end

3: **if** $\mathbf{X}_{(\bar{D})}$ is of size $(R_{\bar{D}-1} I_{\bar{D}}) \times (I_{\bar{D}+1} R_{\bar{D}+1})$:
Apply
- a splitting/generation pattern if $R_{\bar{D}-1} = 1$,
- a generation/splitting pattern if $R_{\bar{D}+1} = 1$,
- else, apply a generation/generation pattern (*cf.* Section 4.3.3).

end

strategy compared to the TT-SVD algorithm leading to a more flexible way to compute the TT-cores. From the algebraic point of view, it is crucial to study the relationship between the estimated TT-cores and the true ones. In Lemma 5.1 and Lemma 5.2, we expose this property in the context of the TT-SVD and the TT-HSVD algorithms. Specifically, we show that these matrices play the role of change-of-basis matrices when a dominant subspace is extracted by the SVD. It is important to note

[2]This condition means that we can still get down into the tree and continue to generate the TT-cores. Furthermore, one may note that indices $\bar{D}_f, \bar{D}$, and $\bar{D}_l$, defined in 4.3.1, are different for each matrix.

[3]Indices $\bar{D}^{'}$ and $\bar{D}^{''}$ are chosen by the user to fit a given topology of the tree.

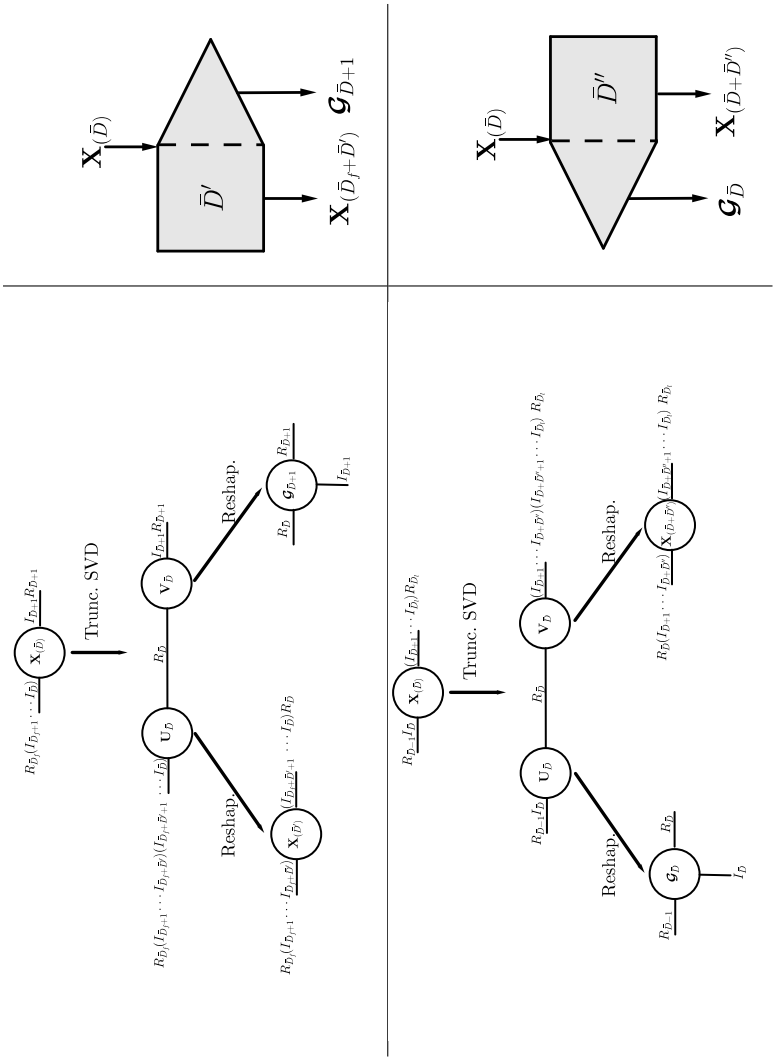

FIG. 8. *Splitting/Generation pattern (left), Generation/Splitting pattern (right).*

that although the TT-SVD has been proposed in [48], its algebraic analysis is to the best of our knowledge original and is carried out for the first time in this work. In [65, 55], the non-uniqueness of the TT decomposition is discussed. Each TT-core can be post and pre-multiplied by any sequence of invertible matrices. First, we recall that for a given rank-deficient matrix, the dominant left singular vectors of the SVD span the column space up to an invertible *change-of-basis* matrix. In the sequel, these matrices are denoted by $\mathbf{P}$ or $\mathbf{Q}$.

5.1. Structure of the estimated TT-cores for the TT-SVD algorithm.

LEMMA 5.1. *Let* $\hat{\boldsymbol{\mathcal{G}}}_d^{seq}$ *be the sequentially estimated TT-core using the TT-SVD algorithm and define a set of* change-of-basis *matrices* $\{\mathbf{P}_1, \ldots, \mathbf{P}_{D-1}\}$ *with* $\mathbf{P}_d$ *of dimensions* $R_d \times R_d$. *The TT-cores associated with the TT-SVD algorithm verify the*

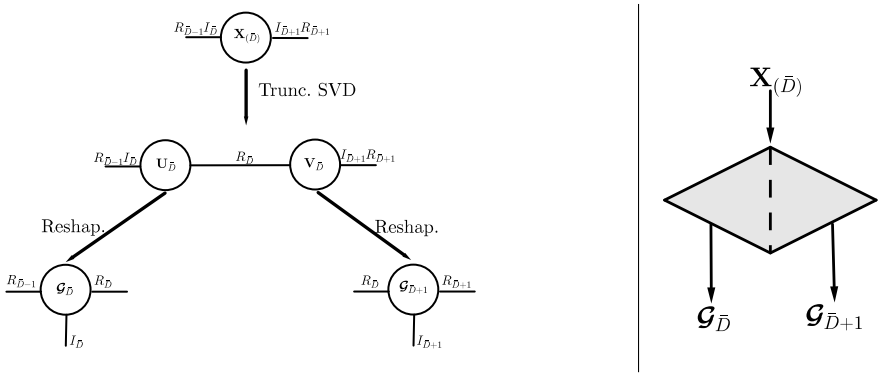

FIG. 9. *Generation/Generation pattern.*

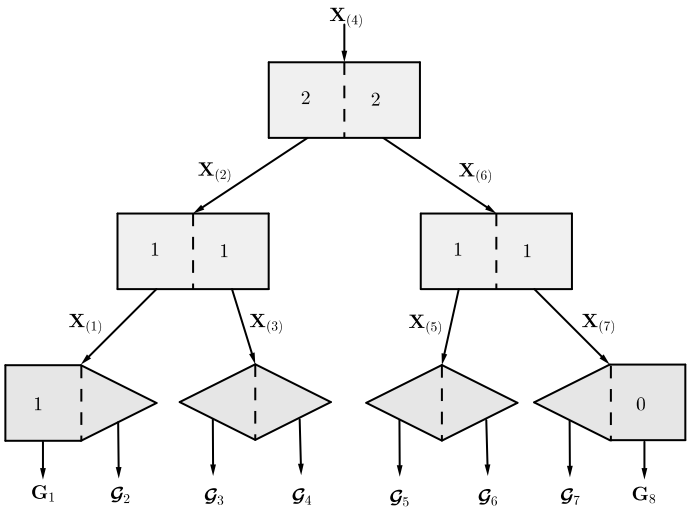

FIG. 10. *Graph-based representation using patterns for a balanced TT-HSVD applied to an 8-order tensor.*

following relations:

$$
\begin{aligned}
\hat{\mathbf{G}}_1^{seq} &= \mathbf{G}_1 \mathbf{P}_1, \\
\hat{\boldsymbol{\mathcal{G}}}_d^{seq} &= \mathbf{P}_{d-1}^{-1} \times_2^1 \boldsymbol{\mathcal{G}}_d \times_3^1 \mathbf{P}_d, \qquad \textit{for } 2 \leq d \leq D-1, \\
\hat{\mathbf{G}}_D^{seq} &= \mathbf{P}_{D-1}^{-1} \mathbf{G}_D.
\end{aligned} \tag{4}
$$

Proof. Based on the algebraic formalism of the patterns given in Appendix 8.1, and giving the facts that

- the Splitting/Splitting patterns are always applied first, before applying any other type of patterns in the TT-SVD algorithm, *i.e.*, we always generate matrices $\mathbf{G}_1$ and $\mathbf{X}_{(2)}$ from matrix $\mathbf{X}_{(1)}$ at the first step.
- Generation/Splitting pattern is always applied after Generation/Splitting and Splitting/Splitting patterns in the TT-SVD. A combination of this type is allowed, since the format of the outputs of these latters corresponds to the

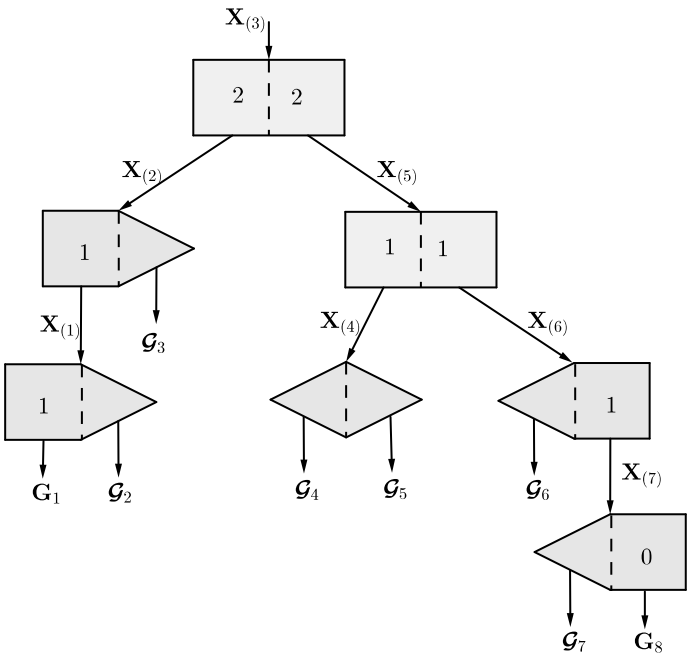

FIG. 11. *Graph-based representation using patterns for an unbalanced TT-HSVD applied to an* 8*-order tensor.*

format of the input of the Generation/Splitting pattern. This means that the expressions of the outputs in (15) and (17) have the same structure of the input in (16).

- the TT-SVD algorithm can be seen as a Splitting/Splitting pattern followed by a succession of Generation/Splitting patterns.

the expression of the Generation/Splitting pattern output, given in (18), shows that the TT-core associated to the TT-SVD admits a general and final formulation given by (4). ∎

5.2. Presentation and analysis of the TT-HSVD algorithm. In the following, we present the TT-HSVD algorithm. We also establish the link between the TT-HSVD and TT-SVD algorithms, in terms of the TT-cores and TT-ranks.

5.2.1. Structure of the estimated TT-cores. Hereafter, we formulate a similar result for the TT-HSVD.

LEMMA 5.2. *Let* $\hat{\boldsymbol{\mathcal{G}}}_d^{\hbar rl}$ *be the hierarchically estimated TT-core using the TT-HSVD algorithm and define a set of* change-of-basis *matrices* $\{\mathbf{Q}_1, \ldots, \mathbf{Q}_{D-1}\}$ *where* $\mathbf{Q}_d$ *is a* $R_d \times R_d$*. The TT-cores associated with the TT-HSVD algorithm verify the following relations:*

$$
\begin{aligned}
\hat{\mathbf{G}}_1^{\hbar rl} &= \mathbf{G}_1 \mathbf{Q}_1 \\
\hat{\boldsymbol{\mathcal{G}}}_d^{\hbar rl} &= \mathbf{Q}_{d-1}^{-1} \times_2^1 \boldsymbol{\mathcal{G}}_d \times_3^1 \mathbf{Q}_d, \qquad \text{for } 2 \leq d \leq D-1 \\
\hat{\mathbf{G}}_D^{\hbar rl} &= \mathbf{Q}_{D-1}^{-1} \mathbf{G}_D.
\end{aligned} \tag{5}
$$

Proof. The demonstration of this Lemma is based on the algebraic formalism of the patterns given in Appendix 8.1. Note that in the TT-HSVD algorithm:

- the Splitting/Splitting patterns are always applied first, before applying any other type of patterns.
- the input and the outputs of the Splitting/Splitting pattern have the same format. Any combination of Splitting/Splitting patterns is possible and allowed (See (9), (14) and (15)).
- the output of the Splitting/Splitting pattern has the same format as the input of all other generation patterns. Any combination of Splitting/Splitting patterns with the other patterns is possible and allowed (See (14), (15), (16) and (19)). ∎

Based on the expressions of the generation patterns outputs, given in (18), (20) and (21), it can be noticed that the TT-core associated to the TT-HSVD admits a general and final formulation given by (5).

5.3. Comparison of the two schemes. In Fig. 12 and Fig. 13, we can see that the TT-SVD and the TT-HSVD in the algebraic perspective estimate the same TT-core up to different change-of-basis matrices. Based on the previous relations on

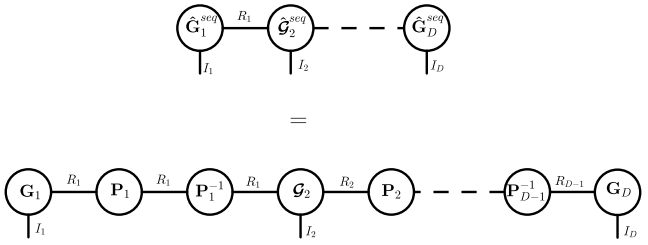

FIG. 12. *Graph-based illustration of the TT-SVD algorithm.*

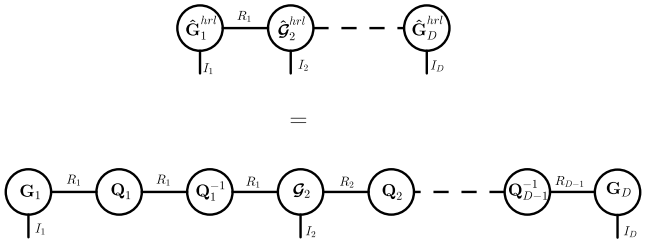

FIG. 13. *Graph-based illustration of the TT-HSVD algorithm.*

the structure of the TT-cores, the following result can be formulated.

LEMMA 5.3. *Define a set of matrices:* $\{\mathbf{H}_1, \ldots, \mathbf{H}_{D-1}\}$, *with* $\mathbf{H}_d = \mathbf{P}_d^{-1}\mathbf{Q}_d$. *The TT-cores obtained by applying the TT-SVD and TT-HSVD algorithms satisfy the following relations:*

$$\hat{\mathbf{G}}_1^{hrl} = \hat{\mathbf{G}}_1^{seq}\mathbf{H}_1, \tag{6}$$

$$\hat{\boldsymbol{\mathcal{G}}}_d^{hrl} = \mathbf{H}_{d-1}^{-1} \times_2^1 \hat{\boldsymbol{\mathcal{G}}}_d^{seq} \times_3^1 \mathbf{H}_d \qquad \textit{for } 2 \leq d \leq D-1\ , \tag{7}$$

$$\hat{\mathbf{G}}_D^{hrl} = \mathbf{H}_{D-1}^{-1}\hat{\mathbf{G}}_D^{seq}. \tag{8}$$

Proof. Note that to demonstrate Eq. (7), we use the identity $\mathbf{A} \times_2^1 \mathbf{B} = \mathbf{AB}$. From (4), we deduce:

$$\boldsymbol{\mathcal{G}}_d = \mathbf{P}_{d-1} \times_2^1 \hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{seq}} \times_3^1 \mathbf{P}_d^{-1}.$$

Substituting this last equation into (5), we have:

$$\hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{hrl}} = (\mathbf{P}_{d-1}^{-1}\mathbf{Q}_{d-1})^{-1} \times_2^1 \hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{seq}} \times_3^1 (\mathbf{P}_d^{-1}\mathbf{Q}_d) \qquad \text{for } 2 \leq d \leq D-1 \qquad \square$$

The above Lemma allows us to formulate the following theorem.

THEOREM 5.4. *The TT-ranks are equal for the TT-SVD and the TT-HSVD algorithms.*

Proof. From (6) and (8) in Lemma 5.3, the proof of the theorem is straightforward for $d = 1$ and $d = D$. For $2 \leq d \leq D-1$, two alternative Tucker-based formulations of (7) are

$$\begin{aligned}
\hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{hrl}} &= \hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{seq}} \times_1 \mathbf{H}_{d-1}^{-1} \times_2 \mathbf{I}_{I_d} \times_3 \mathbf{H}_d^T, \\
\hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{seq}} &= \hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{hrl}} \times_1 \mathbf{H}_{d-1} \times_2 \mathbf{I}_{I_d} \times_3 \mathbf{H}_d^{-T}.
\end{aligned}$$

Based on the two above relations, tensors $\hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{hrl}}$ and $\hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{seq}}$ have a multilinear (R_{d-1}, I_d, R_d)-rank. Since the TT-ranks correspond to the dimensions of the first and third modes of $\hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{hrl}}$ or $\hat{\boldsymbol{\mathcal{G}}}_d^{\mathtt{seq}}$, the proof is completed. $\square$

Intuitively, the above theorem seems natural since the TT-ranks are essentially related to the TT model, and in particular to the matrices given by (2), and not to the choice of the algorithm.

One may note that contrary to [48], in the TT-HSVD, we have to deal with non-orthonormal factor since we reshape and use both the left and right parts of the SVD at each step. In [61, 19], the authors have proposed an interesting strategy to deal with non-orthonormal factors that can be applied in the context of the TT-HSVD algorithm. However, we will show in the next section that even if the tensor is affected by noise, both algorithms, namely the TT-SVD and the TT-HSVD, can still have the same robustness when the TT-ranks are either assumed to be known or when they are estimated in the algorithms.

6. Computational complexity and simulation results.

6.1. Numerical computation of the SVD. In this section, we compare the computational complexity of both algorithms using the truncated SVD. Note that TT-HSVD an TT-SVD involve the same number of SVDs. However, the TT-HSVD has the advantage of applying SVDs to matrices of smaller dimensions compared to the TT-SVD algorithm, which results in a lower computational complexity. The numerical stability of the TT-SVD and the TT-HSVD is relied to the well-known numerical rank determination in the SVD. We have essentially two scenarios of interest.

1. The true rank is a priori known. This is often realistic due to *a priori* knowledge on the physical model as for instance in wireless communication applications where the rank can be the number of path/user, in array processing for sky imaging where the rank is the number of most brining stars (known due to decade and decade of sky observation), ... For these important scenarios, the TT-SVD and the TT-HSVD algorithms provide an exact factorization of

a high-order tensor into a collection of low-order tensors. This is mainly the framework of our contribution. The computation of the SVD is done based on two orthogonal iteration (OI) algorithms [21] executed in parallel. We recall that computing the matrix $\mathbf{U}$ of size $m \times r$ consists in [21] recursively computing the QR factorization of the $m \times r$ matrix $(\mathbf{A}\mathbf{A}^T)\mathbf{U}_{i-1} = \mathbf{U}_i\mathbf{R}$ with dominant complexity [21] $O(r^2m)$. In the same way, we can calculate matrix $\mathbf{V}$ of size $n \times r$, using the QR decomposition of $(\mathbf{A}^T\mathbf{A})\mathbf{V}_{i-1} = \mathbf{V}_i\mathbf{R}$ with dominant complexity [21] $O(r^2n)$. The initial matrices $\mathbf{U}_0$ and $\mathbf{V}_0$ can be randomly initialized, and the convergence is assumed according to a tolerence function or when a maximum number of iterations is reached. If the `svds` function of MatLab is used, the initial matrices $\mathbf{U}_0$ and $\mathbf{V}_0$ are then drawn from a Gaussian distribution, and the tolerence function is chosen by default to be smaller than 10^{-10}, while the maximum number of iterations is fixed by default at 100. The singular values are automatically obtained with the calculation of $\mathbf{U}$ and $\mathbf{V}$ from the matrix $\mathbf{R}$. Considering the matrix multiplication cost $O\big(rmn\big)$, the overall SVD cost is evaluated at $O\big(r^2(m+n) + rmn\big)$.

2. The true rank is unknown. In this case, the problem is to find a robust estimation of the true rank, called the numerical rank. Considering a numerical rank larger than the true one can be problematic since the last columns of the basis matrices are pondered by near-zero singular values. Typically, the true rank is numerically estimated by searching a numerical gap toward the dominant singular values and the others. Many dedicated methods exist to find this gap [53, 64, 10, 40, 8]. In our work, the numerical rank is computed with the native routine `rank.m` of MatLab as the number of singular values that are larger than a tolerance. The tolerance is a function of the spacing of floating point (`eps`), the size and the norm of the matrix as for instance $\max\{m,n\} \cdot eps(||\mathbf{A}||)$ where $eps(x)$ is the distance from $|x|$ to the next larger in magnitude floating point number of the same precision as x. It makes sense to use the bi-iteration algorithm based on the sequential computation of two OI(s). Unlike the OI algorithm, the singular-values are an output of the bi-iteration algorithm [21, 62]. The complexity cost per iteration of the bi-iteration algorithm is in the same order as a single OI.

Finally, the two schemes, namely the popular TT-SVD and our proposition called the TT-HSVD, inherit from the numerical robustness of the SVD. But, we think that the optimization of this important operation (the numerical rank estimation) is out of scope of our paper.

6.2. Complexity analysis and simulations. Considering a 4-order hypercubic (I) tensor of TT-ranks equal to R, the complexity of the TT-SVD algorithm is given by $O(RI^4) + O(R^2I^3) + O(R^2I^2)$ where,

1. the complexity of the first R-truncated SVD is $O(R^2(I^3+I)+RI^4) \approx O(RI^4)$ for large I.
2. The complexity of the second R-truncated SVD applied on $\mathbf{X}_{(2)}$ is $O(R^2(RI+I^2) + R^2I^3) \approx O(R^2I^3)$ if $I \gg R$.
3. The complexity of the last R-truncated SVD applied on $\mathbf{X}_{(3)}$ is $O(R^2(RI+I) + R^2I^2) \approx O(R^2I^2)$.

Following the same reasoning, the complexity of the TT-HSVD algorithm for a mono-core architecture is given by $O(RI^4) + O(R^2I^2)$ where,

1. The complexity of the first R-truncated SVD of $\mathbf{X}_{(2)}$ is $O(R^2(I^2+I^2)+RI^4) \approx O(RI^4)$.

2. The complexity of the R-truncated SVD of $\mathbf{X}_{(1)}$ related to the left part of the tree is $O(R^2(IR+I)+R^2I^2) \approx O(R^2I^2)$.
3. The complexity of the R-truncated SVD of $\mathbf{X}_{(3)}$ related to the right part of the tree is $O(R^2(IR+I)+R^2I^2) \approx O(R^2I^2)$.

The complexity of the TT-HSVD is much lower than that of the TT-SVD due to the large term $O(R^2I^3)$.
Moreover, for both algorithms, the term dominating the complexity corresponds to the first SVD step, which is applied to the largest matrix at the beginning of the process. For a D-order tensor, by considering the dominant cost of the truncated SVD, and for a large $I_d = I$ for all d, the dominant complexities of TT-SVD and TT-HSVD (in the balanced case), are respectively given by

$$\kappa_{\mathtt{seq}} = O(RI^D) + O(R^2I^{(D-1)}) + O(R^2I^{(D-2)}) + \cdots + O(R^2I^2),$$

and

$$\kappa_{\mathtt{hrl}} = O(RI^D) + O(R^2I^{\frac{D}{2}}) + O(R^2I^{\frac{D}{4}}) + \cdots + O(R^2I^2).$$

This means that the complexity of TT-SVD grows faster than that of the TT-HSVD algorithm as a function of the tensor order. Thus, TT-HSVD offers a significant advantage over the TT-SVD algorithm especially for high-order data tensors. This gain in complexity for the TT-HSVD is due to the low storage cost of the considered matrices $\mathbf{X}_{(d)}$ in the TT-HSVD compared to those of the TT-SVD. One may note that the storage cost of these matrices in TT-SVD and TT-HSVD corresponds to the product of dimensions in the complexity of the truncated SVD. This means that the TT-HSVD has advantages in both the algorithmic complexity and the intermediate storage cost before recovering the TT-cores, which results in less memory resources (both in terms of memory space and memory access) for the TT-HSVD compared to the TT-SVD algorithm. For instance, for a 4-order tensor, the first reshaping matrices for the TT-SVD and the TT-HSVD have both I^4, the second reshapings of TT-SVD and TT-HSVD have respectively RI^3 and RI^2, and the last ones have RI^2 for both algorithms. This means that the TT-HSVD needs less memory in view of the terms RI^3 and RI^2 of the second reshapings. Now, if we consider a D-order hypercubic tensor $\boldsymbol{\mathcal{X}}$, where D is a power of 2, *i.e.*, $D = 2^n$, and n is an integer. This means that $\boldsymbol{\mathcal{X}}$ can have balanced unfoldings in all the TT-HSVD steps (as in Fig. 6 (left)). The storage cost needed for the singular vector matrices in the TT-HSVD at each step varies then as follows.

$$\underbrace{2I^{\frac{D}{2}}R}_{\text{1st step}} \rightarrow \underbrace{2I^{\frac{D}{4}}R^2 + 2I^{\frac{D}{4}}R}_{\text{2nd step}} \rightarrow \underbrace{6I^{\frac{D}{8}}R^2 + 2I^{\frac{D}{8}}R}_{\text{3rd step}} \rightarrow \cdots \rightarrow \underbrace{(D-2)IR^2 + 2IR}_{\text{last step}}.$$

On the other hand, the storage cost for the singular vector matrices in the TT-SVD varies as follows.

$$\underbrace{I^{(D-1)}R + IR}_{\text{1st step}} \rightarrow \underbrace{I^{D-2}R + IR^2}_{\text{2nd step}} \rightarrow \underbrace{I^{D-3}R + IR^2}_{\text{3rd step}} \rightarrow \cdots \rightarrow \underbrace{IR + IR^2}_{\text{last step}}.$$

We can note that the dominant storage cost of the TT-SVD is reduced by a factor of I between two consecutive steps, while for the TT-HSVD, this storage cost decreases at a faster pace.

In Table 1, we compare the computation time of both algorithms using the "Tic-Toc"functions of MATLAB to decompose a D-th order tensor $\boldsymbol{\mathcal{X}}$, for $9 \leq I \leq 13$.

To this end, we generate the tensor $\boldsymbol{\mathcal{X}}$ with $I = I_d$ and $R = R_i$, for $1 \leq d \leq D$, $1 \leq i \leq D-1$, where R_d are the associated TT-ranks. The execution time is expressed in seconds (s). The simulations were performed on a personal computer equipped with an *Intel(R) CORE(TM) i7-3687U CPU @ 2.10GHz* processor and *8Gb* RAM.

Remark 6.1. Note that the execution time of TT-HSVD is that of a sequential processing, where all the SVD steps are run in batch. This execution time could be significantly reduced [31] if a parallel processing architecture is used.

TABLE 1
Comparison of the computation time of both algorithms ($D = 8$, $R = 3$).

Tensor dimension	$I = 9$	$I = 10$	$I = 11$	$I = 12$	$I = 13$
TT-SVD algorithm	5.9	13.3	29	88.1	–
TT-HSVD algorithm	1.9	4	8.3	17.3	43.9
Gain	3.1	3.3	3.5	5.1	∞

In Table 1, we consider $(D, R) = (8, 3)$. Note that the results shown in this table are in agreement with the analytical complexity analysis, confirming that a significant complexity gain is achieved by TT-HSVD in comparison with TT-SVD, especially for very high-order tensors. In particular, for $I = 13$, the TT-SVD algorithm can not be executed by the computer, which returned an out of memory error. Note that the first unfolding matrix $\mathbf{X}_{(1)}$ of the TT-SVD algorithm is of size 13×13^7, which requires a right singular vectors matrix of size 3×13^7 for the truncated SVD. The size of this matrix is actually beyond the allowed storage capacity. Notice that the TT-HSVD algorithm has also the advantage of having a storage cost lower than the TT-SVD algorithm.

In Table 2, we evaluate the impact of the choice of the unfolding used as the starting point of the TT-HSVD, in terms of computational complexity. To this end, we choose different values of I_i, with $(D, R) = (8, 4)$. The two configurations used in the results of Figs. 6 (left) and 6 (right) are considered here. The results shown in this table correspond to the best computation time of the TT-HSVD algorithm.

TABLE 2
Comparison of the computation time of TT-HSVD and TT-SVD.

Scenarios $(I_1, \cdots, I_8)$	TT-HSVD (Fig. 6 (left))	TT-HSVD (Fig. 6 (right))	TT-SVD	Gain
$(18, 36, 32, 16, 6, 6, 6, 6)$.	22.6	16.3	152.2	9.3
$(22, 36, 32, 19, 6, 6, 6, 6)$.	45.1	23.7	993.7	42

It can be noted that choosing the most "square" unfolding matrix (i.e. the one with more balanced row and column dimensions) results in a lower overall computational complexity. The higher is the tensor order, more degrees of freedom are available for choosing the best unfolding to start the TT-HSVD algorithm.

In the following experiment, we generate an 8-order tensor $\mathcal{X}$ of size $I \times \cdots \times I$ in the TT-format, with TT-ranks $(R_1, \cdots, R_7)$ chosen randomly between 2 and 4. This tensor is decomposed by means of the TT-SVD and TT-HSVD algorithms. We fix $I = 4$ and observe the TT-ranks calculated by each algorithm. In Fig. 14, we plot

the observed ranks for both algorithms against the true TT-rank R_4. We can see that the TT-ranks calculated by both algorithms follow the true values. The same results are found for the other TT-ranks. We note here that for random realizations

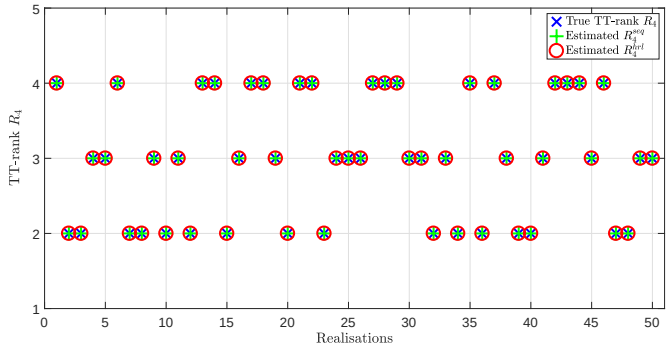

FIG. 14. *The estimated ranks for the true TT-rank R_4.*

of the TT-ranks, we observe exactly the same ranks for both algorithms TT-SVD and TT-HSVD. This illustrates well the results on the equality between the TT-SVD ranks and the TT-HSVD ranks obtained previously.
In Table 3, we present the normalized reconstruction error of the estimated tensor for both the TT-SVD and the TT-HSVD, considering a tensor affected by an additive Gaussian noise. The original tensor is an 8-order hypercubic tensor following a CPD of canonical rank $R = 2$ with dimension $I = 4$. This is an interesting and realistic case regarding the Joint dImensionality Reduction And Factor rEtrieval (JIRAFE) framework [69, 68], where the original tensor is a high-order CPD that is decomposed into a TT format to break its dimensionality before estimating its parameters. Two cases are considered in the following experiment, *i.e.*, when the original noisy tensor has known TT-ranks equal to R and when the TT-ranks are unkown and are estimated at each step of the algorithm. The given errors are obtained by averaging the results over 1000 Monte-Carlo runs. One may note that for a wide range of SNR, either when the TT-ranks are estimated or are assumed to be known, both algorithms have the same robustness. This means that the TT-HSVD can be a better alternative for the TT-SVD algorithm in the JIRAFE framework for parameters estimation of high-order tensors.

7. Conclusion and future work. In this work, we have proposed the TT-HSVD algorithm, a hierarchical algorithm for Tensor Train decomposition. This new algorithm allows to recover simultaneously/hierarchically the TT-core tensors and their TT-ranks. A new graph-based representation using patterns has also been proposed to simplify and make more illustrative the algorithmic representation of both TT-SVD and TT-HSVD algorithms. Our analyses have shown that the TT-SVD and TT-HSVD algorithms estimate the same TT-ranks and the same TT-core tensors up to specific bases, with significant time and memory savings for the TT-HSVD algorithm. Perspectives include the study of the combination of the tensor modes in a random way, and the consequences of this modification on the estimation and the complexity of the algorithm. Future research includes the use of cross interpolation instead of the SVD in the proposed hierarchical algorithm. Several works have addressed the cross interpolation solution, either for discrete tensors as in [57, 2] or its

TABLE 3
Comparison of the robustness of TT-HSVD and TT-SVD.

SNR (dB)	Unknown rank		Known rank	
	TT-HSVD	TT-SVD	TT-HSVD	TT-SVD
30	1.28×10^{-6}	1.27×10^{-6}	1.28×10^{-6}	1.27×10^{-6}
25	4.02×10^{-6}	4.01×10^{-6}	4.06×10^{-6}	4.05×10^{-6}
20	1.3×10^{-5}	1.3×10^{-5}	1.29×10^{-5}	1.28×10^{-5}
15	4.06×10^{-5}	4.05×10^{-5}	4.06×10^{-5}	4.06×10^{-5}
10	1.3×10^{-4}	1.3×10^{-4}	1.3×10^{-4}	1.3×10^{-4}
5	4.07×10^{-4}	4.1×10^{-4}	4.1×10^{-4}	4.2×10^{-4}
0	1.3×10^{-3}	1.3×10^{-3}	1.3×10^{-3}	1.4×10^{-3}
-5	4×10^{-3}	4.7×10^{-3}	4.4×10^{-3}	5.7×10^{-3}
-10	1.36×10^{-2}	2.11×10^{-2}	1.71×10^{-2}	2.95×10^{-2}

generalization to the continous case as in [22]. For example, in [57], the quasioptimal accuracy of the maximum-volume cross interpolation is proven, and an algorithm using the same logic as TT-SVD while substituting the SVD approximation by the interpolation is proposed. Besides the quasioptimality, results have also shown that this solution can provide low rank approximations faster than the SVD. We can therefore imagine that the use of TT-HSVD with patterns that use the interpolation instead of the SVD would be a logical and interesting continuation of this work. Future works also involve the application of the TT-HSVD algorithm on problems such as tensor completion [38], blind source separation [5] and fast SVD of large scale matrices [43].

8. Appendix.

8.1. Algebraic analysis of the patterns. In this section, we give an algebraic analysis of the patterns to prove Lemmas 5.1 and 5.2. In the sequel only the analysis of the Splitting/Splitting pattern is given in detail. The analysis of the Generation/Generation, Splitting/Generation and Generation/Splitting patterns is very similar and is omitted here to the lack of space, only the expressions of inputs and outputs of these patterns are given. Note that in an informatic point of view, all the patterns can be implemented using the same function, that we call "Pattern", since they can be considered similar up to index grouping. Before giving the algebraic analysis of the patterns, we present in Algorithm 2 a pseudocode of the function that can be used in the TT-HSVD algorithm. In this example, we suppose that the rank is estimated in the function "Pattern" using the `rank.m` of MatLab. Outputs $\boldsymbol{\mathcal{S}}_1$, $\boldsymbol{\mathcal{S}}_2$ can be either tensors or matrices depeding on their respective dimensions dim1 and dim2.

8.1.1. Splitting/Splitting Pattern. This first pattern (Fig. 7) takes as input a matrix and returns 2 matrices. It applies the SVD to the input matrix and reshape the 2 matrices generated according to the choice we want. The graphical representation of this pattern is given in Fig. 7.

Algorithm 2 Pattern

Input: $\mathbf{M}$, dim1, dim2
Output: $\boldsymbol{\mathcal{S}}_1$, $\boldsymbol{\mathcal{S}}_2$, R.

1: $R = \texttt{rank}(\mathbf{M})$
2: $[\mathbf{U}, \mathbf{D}, \mathbf{V}] = \texttt{svd}(\mathbf{M}, R)$
3: $\mathbf{V} = \mathbf{D}\mathbf{V}^T$
4: $\boldsymbol{\mathcal{S}}_1 = \texttt{reshape}(\mathbf{U}, \text{dim1})$
5: $\boldsymbol{\mathcal{S}}_2 = \texttt{reshape}(\mathbf{V}, \text{dim2})$

The input matrix $\mathbf{X}_{(\bar{D})}$ can be expressed as:

$$\begin{aligned}\mathbf{X}_{(\bar{D})} = &\sum_{r_{\bar{D}_f},\cdots,r_{\bar{D}_l}=1}^{R_{\bar{D}_f},\cdots,R_{\bar{D}_l}} \Big(\mathbf{g}_{\bar{D}}(r_{\bar{D}-1}, r_{\bar{D}}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}_f+1}(r_{\bar{D}_f}, r_{\bar{D}_f+1}) \otimes \mathbf{Q}_{\bar{D}_f}^{-1}(:, r_{\bar{D}_f})\Big) \\ &\cdot \Big(\mathbf{Q}_{\bar{D}_l}(r_{\bar{D}_l}, :)^T \otimes \mathbf{g}_{\bar{D}_l}(r_{\bar{D}_l-1}, r_{\bar{D}_l}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}+1}(r_{\bar{D}}, r_{\bar{D}+1})\Big)^T \qquad (9) \\ = &\sum_{r_{\bar{D}}=1}^{R_{\bar{D}}} \Big(\sum_{r_{\bar{D}_f},\cdots,r_{\bar{D}-1}=1}^{R_{\bar{D}_f},\cdots,R_{\bar{D}-1}} \mathbf{g}_{\bar{D}}(r_{\bar{D}-1}, r_{\bar{D}}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}_f+1}(r_{\bar{D}_f}, r_{\bar{D}_f+1}) \otimes \mathbf{Q}_{\bar{D}_f}^{-1}(:, r_{\bar{D}_f})\Big) \\ &\cdot \Big(\sum_{r_{\bar{D}+1},\cdots,r_{\bar{D}_l}=1}^{R_{\bar{D}+1},\cdots,R_{\bar{D}_l}} \mathbf{Q}_{\bar{D}_l}(r_{\bar{D}_l}, :)^T \otimes \mathbf{g}_{\bar{D}_l}(r_{\bar{D}_l-1}, r_{\bar{D}_l}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}+1}(r_{\bar{D}}, r_{\bar{D}+1})\Big)^T \\ = &\sum_{r_{\bar{D}}=1}^{R_{\bar{D}}} \mathbf{S}(:, r_{\bar{D}})\mathbf{T}(r_{\bar{D}}, :) = \mathbf{S}\mathbf{T}. \qquad (10)\end{aligned}$$

where $\mathbf{S}$ and $\mathbf{T}$ are respectively of size $R_{\bar{D}_f}(I_{\bar{D}_f+1} \cdots I_{\bar{D}}) \times R_{\bar{D}}$ and $R_{\bar{D}} \times (I_{\bar{D}+1} \cdots I_{\bar{D}_l}) R_{\bar{D}_l}$, and we have $\text{rank}(\mathbf{X}_{(\bar{D})}) = R_{\bar{D}}$.
Note that the expression (9) corresponds to the definition given in (3), where $R_{\bar{D}_f} = R_0 = 1$, $R_{\bar{D}_l} = R_D = 1$, and $\mathbf{Q}_0 = \mathbf{Q}_D = 1$.
Applying the SVD on $\mathbf{X}_{(\bar{D})}$ gives:

$$\mathbf{X}_{(\bar{D})} = \mathbf{U}_{\bar{D}} \mathbf{V}_{\bar{D}}. \qquad (11)$$

From (10) and (11), we can conclude that:

$$\begin{aligned}\mathbf{U}_{\bar{D}} &= \mathbf{S}\mathbf{Q}_{\bar{D}} \\ &= \sum_{r_{\bar{D}}=1}^{R_{\bar{D}}} \Big(\sum_{r_{\bar{D}_f},\cdots,r_{\bar{D}-1}=1}^{R_{\bar{D}_f},\cdots,R_{\bar{D}-1}} \mathbf{g}_{\bar{D}}(r_{\bar{D}-1}, r_{\bar{D}}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}_f+1}(r_{\bar{D}_f}, r_{\bar{D}_f+1}) \otimes \mathbf{Q}_{\bar{D}_f}^{-1}(:, r_{\bar{D}_f})\Big) \\ &\qquad \cdot \mathbf{Q}(r_{\bar{D}}, :) \\ &= \sum_{r_{\bar{D}_f},\cdots,r_{\bar{D}}=1}^{R_{\bar{D}_f},\cdots,R_{\bar{D}}} \Big(\mathbf{g}_{\bar{D}}(r_{\bar{D}-1}, r_{\bar{D}}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}_f+1}(r_{\bar{D}_f}, r_{\bar{D}_f+1}) \otimes \mathbf{Q}_{\bar{D}_f}^{-1}(:, r_{\bar{D}_f})\Big) \mathbf{Q}(r_{\bar{D}}, :) \qquad (12)\end{aligned}$$

and

$$\begin{aligned}\mathbf{V}_{\bar{D}} &= \mathbf{Q}_{\bar{D}}^{-1}\mathbf{T} \\ &= \sum_{r_{\bar{D}}=1}^{R_{\bar{D}}} \mathbf{Q}_{\bar{D}}^{-1}(:, r_{\bar{D}})\Big(\sum_{r_{\bar{D}+1},\cdots,r_{\bar{D}_l}=1}^{R_{\bar{D}+1},\cdots,R_{\bar{D}_l}} \mathbf{Q}_{\bar{D}_l}(r_{\bar{D}_l}, :)^T \otimes \mathbf{g}_{\bar{D}_l}(r_{\bar{D}_l-1}, r_{\bar{D}_l}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}+1}(r_{\bar{D}}, r_{\bar{D}+1})\Big)^T \\ &= \sum_{r_{\bar{D}},\cdots,r_{\bar{D}_l}=1}^{R_{\bar{D}},\cdots,R_{\bar{D}_l}} \mathbf{Q}_{\bar{D}}^{-1}(:, r_{\bar{D}})\Big(\mathbf{Q}_{\bar{D}_l}(r_{\bar{D}_l}, :)^T \otimes \mathbf{g}_{\bar{D}_l}(r_{\bar{D}_l-1}, r_{\bar{D}_l}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}+1}(r_{\bar{D}}, r_{\bar{D}+1})\Big)^T \end{aligned} \tag{13}$$

where $\mathbf{Q}_{\bar{D}}$ is a $R_{\bar{D}} \times R_{\bar{D}}$ *change-of-basis* matrix.
Let $\mathbf{X}_{(\bar{D}_f+\bar{D}')}$ and $\mathbf{X}_{(\bar{D}+\bar{D}'')}$ be the reshaping of the matrices $\mathbf{U}_{\bar{D}}$ and $\mathbf{V}_{\bar{D}}$ according to the chosen $\bar{D}'$ and $\bar{D}''$. From (12) and (13), we have:

$$\begin{aligned}\mathbf{X}_{(\bar{D}_f+\bar{D}')} &= \text{reshape}(\mathbf{U}_{\bar{D}}, R_{\bar{D}_f} I_{\bar{D}_f+1} \cdots I_{\bar{D}_f+\bar{D}'}, I_{\bar{D}_f+\bar{D}'+1} \cdots I_{\bar{D}} R_{\bar{D}}) \\ &= \sum_{r_{\bar{D}_f},\cdots,r_{\bar{D}}=1}^{R_{\bar{D}_f},\cdots,R_{\bar{D}}} \Big(\mathbf{g}_{\bar{D}_f+\bar{D}'}(r_{\bar{D}_f+\bar{D}'-1}, r_{\bar{D}_f+\bar{D}'}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}_f+1}(r_{\bar{D}_f}, r_{\bar{D}_f+1}) \otimes \mathbf{Q}_{\bar{D}_f}^{-1}(:, r_{\bar{D}_f})\Big) \\ &\quad \cdot \Big(\mathbf{Q}_{\bar{D}}(r_{\bar{D}}, :)^T \otimes \mathbf{g}_{\bar{D}}(r_{\bar{D}-1}, r_{\bar{D}}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}'+1}(r_{\bar{D}'}, r_{\bar{D}'+1})\Big)^T \end{aligned} \tag{14}$$

and

$$\begin{aligned}\mathbf{X}_{(\bar{D}+\bar{D}'')} &= \text{reshape}(\mathbf{V}_{\bar{D}}, R_{\bar{D}} I_{\bar{D}+1} \cdots I_{\bar{D}+\bar{D}''}, I_{\bar{D}+\bar{D}''+1} \cdots I_{\bar{D}_l} R_{\bar{D}_l}) \\ &= \sum_{r_{\bar{D}},\cdots,r_{\bar{D}_l}=1}^{R_{\bar{D}},\cdots,R_{\bar{D}_l}} \Big(\mathbf{g}_{\bar{D}+\bar{D}''}(r_{\bar{D}+\bar{D}''-1}, r_{\bar{D}+\bar{D}''}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}+1}(r_{\bar{D}}, r_{\bar{D}+1}) \otimes \mathbf{Q}_{\bar{D}}^{-1}(:, r_{\bar{D}})\Big) \\ &\quad \cdot \Big(\mathbf{Q}_{\bar{D}_l}(r_{\bar{D}_l}, :)^T \otimes \mathbf{g}_{\bar{D}_l}(r_{\bar{D}_l-1}, r_{\bar{D}_l}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}+\bar{D}''+1}(r_{\bar{D}+\bar{D}''}, r_{\bar{D}+\bar{D}''+1})\Big)^T \end{aligned} \tag{15}$$

8.1.2. Splitting/Generation Pattern:. This pattern (Fig. 8 (left)) also takes as input a matrix and gives as outputs a tensor and a reshaped matrix according to our choice. According to (14) and (15), the matrix $\mathbf{X}_{(\bar{D})}$ will be expressed as:

$$\begin{aligned}\mathbf{X}_{(\bar{D})} &= \sum_{r_{\bar{D}_f},\cdots,r_{\bar{D}+1}=1}^{R_{\bar{D}_f},\cdots,R_{\bar{D}+1}} \Big(\mathbf{g}_{\bar{D}}(r_{\bar{D}-1}, r_{\bar{D}}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}_f+1}(r_{\bar{D}_f}, r_{\bar{D}_f+1}) \otimes \mathbf{Q}_{\bar{D}_f}^{-1}(:, r_{\bar{D}_f})\Big) \\ &\quad \cdot \Big(\mathbf{Q}_{\bar{D}+1}(r_{\bar{D}+1}, :)^T \otimes \mathbf{g}_{\bar{D}+1}(r_{\bar{D}}, r_{\bar{D}+1})\Big)^T \end{aligned} \tag{16}$$

The outpouts $\mathbf{X}_{(\bar{D}')}$ and $\hat{\boldsymbol{\mathcal{G}}}_{\bar{D}+1}$ of this pattern will then be expressed as follows:

$$\begin{aligned}\mathbf{X}_{(\bar{D}_f+\bar{D}')} &= \text{reshape}(\mathbf{U}_{\bar{D}}, R_{\bar{D}_f} I_{\bar{D}_f+1} \cdots I_{\bar{D}_f+\bar{D}'}, I_{\bar{D}_f+\bar{D}'+1} \cdots I_{\bar{D}} R_{\bar{D}}) \\ &= \sum_{r_{\bar{D}_f},\cdots,r_{\bar{D}}=1}^{R_{\bar{D}_f},\cdots,R_{\bar{D}}} \Big(\mathbf{g}_{\bar{D}_f+\bar{D}'}(r_{\bar{D}_f+\bar{D}'-1}, r_{\bar{D}_f+\bar{D}'}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}_f+1}(r_{\bar{D}_f}, r_{\bar{D}_f+1}) \otimes \mathbf{Q}_{\bar{D}_f}^{-1}(:, r_{\bar{D}_f})\Big) \\ &\quad \cdot \Big(\mathbf{Q}_{\bar{D}}(r_{\bar{D}}, :)^T \otimes \mathbf{g}_{\bar{D}}(r_{\bar{D}-1}, r_{\bar{D}}) \otimes \cdots \otimes \mathbf{g}_{\bar{D}'+1}(r_{\bar{D}'}, r_{\bar{D}'+1})\Big)^T \end{aligned} \tag{17}$$

and

$$\begin{aligned}\hat{\boldsymbol{\mathcal{G}}}_{\bar{D}+1} &= \text{reshape}(\mathbf{V}_{\bar{D}}, R_{\bar{D}}, I_{\bar{D}+1}, R_{\bar{D}+1}) \\ &= \mathbf{Q}_{\bar{D}}^{-1} \times_2^1 \boldsymbol{\mathcal{G}}_{\bar{D}+1} \times_3^1 \mathbf{Q}_{\bar{D}+1}\end{aligned} \tag{18}$$

The Generation/Splitting pattern has same expressions as the Splitting/Generation pattern if the outputs are reversed.

Generation/Generation Pattern:. It is the third and last type of core generation patterns (Fig. 9). This pattern has as input matrices of dimension $(R_{\bar{D}-1} I_{\bar{D}}) \times (I_{\bar{D}+1} R_{\bar{D}+1})$. It outputs 2 tensors at time. From what we have seen before, the matrix $\mathbf{X}_{\bar{D}}$ will be expressed as:

$$\begin{aligned}\mathbf{X}_{(\bar{D})} = &\sum_{r_{\bar{D}-1},r_{\bar{D}},r_{\bar{D}+1}=1}^{R_{\bar{D}-1},R_{\bar{D}},R_{\bar{D}+1}} \Big(\mathbf{g}_{\bar{D}}(r_{\bar{D}-1}, r_{\bar{D}}) \otimes \mathbf{Q}_{\bar{D}-1}^{-1}(:, r_{\bar{D}-1})\Big) \\ &\cdot \Big(\mathbf{Q}_{\bar{D}+1}(r_{\bar{D}+1}, :)^T \otimes \mathbf{g}_{\bar{D}+1}(r_{\bar{D}}, r_{\bar{D}+1})\Big)^T\end{aligned} \tag{19}$$

The outpouts $\hat{\boldsymbol{\mathcal{G}}}_{\bar{D}}$ and $\hat{\boldsymbol{\mathcal{G}}}_{\bar{D}+1}$ of this pattern will then be expressed as follows:

$$\begin{aligned}\hat{\boldsymbol{\mathcal{G}}}_{\bar{D}} &= \text{reshape}(\mathbf{U}_{\bar{D}}, R_{\bar{D}-1}, I_{\bar{D}}, R_{\bar{D}}) \\ &= \mathbf{Q}_{\bar{D}-1}^{-1} \times_2^1 \boldsymbol{\mathcal{G}}_{\bar{D}} \times_3^1 \mathbf{Q}_{\bar{D}}\end{aligned} \tag{20}$$

and

$$\begin{aligned}\hat{\boldsymbol{\mathcal{G}}}_{\bar{D}+1} &= \text{reshape}(\mathbf{V}_{\bar{D}}, R_{\bar{D}}, I_{\bar{D}+1}, R_{\bar{D}+1}) \\ &= \mathbf{Q}_{\bar{D}}^{-1} \times_2^1 \boldsymbol{\mathcal{G}}_{\bar{D}+1} \times_3^1 \mathbf{Q}_{\bar{D}+1}\end{aligned} \tag{21}$$

REFERENCES

[1] R. Badeau and R. Boyer, *Fast multilinear singular value decomposition for structured tensors*, SIAM J. Matrix Anal. Appl., 30 (2008).

[2] J. Ballani and L. Grasedyck, *Hierarchical tensor approximation of output quantities of parameter-dependent pdes*, SIAM Journal on uncertainty quantification, 3 (2015), pp. 852–872.

[3] J. Ballani, L. Grasedyck, and M. Kluge, *Black box approximation of tensors in hierarchical Tucker format*, Linear algebra and its Applications, 438 (2013), pp. 639–657.

[4] G. Bergqvist and E. G. Larsson, *The higher-order singular value decomposition: Theory and an application [lecture notes]*, IEEE Signal Processing Magazine, 27 (2010), pp. 151–154.

[5] M. Boussé, O. Debals, and L. D. Lathauwer, *A tensor-based method for large-scale blind source separation using segmentation*, IEEE Transactions on Signal Processing, 65 (2016), pp. 346–358.
[6] R. Boyer, R. Badeau, and G. Favier, *Fast orthogonal decomposition of Volterra cubic kernels using oblique unfolding*, in 36th IEEE International Conference on Acoustics, Speech and Signal Processing, 2011.
[7] J. Brachat, P. Comon, B. Mourrain, and E. Tsigaridas, *Symmetric tensor decomposition*, Linear Algebra and its Applications, 433 (2010), pp. 1851 – 1872.
[8] G. Camba-Mendez and G. Kapetanios, *Statistical tests and estimators of the rank of a matrix and their applications in econometric modelling*, Econometric Reviews, 28 (2009), pp. 581–611.
[9] J. Carroll and J.-J. Chang, *Analysis of individual differences in multidimensional scaling via an n-way generalization of 'Eckart-Young' decomposition*, Psychometrika, 35 (1970), pp. 283–319.
[10] T. F. Chan, *Rank revealing QR factorizations*, Linear algebra and its applications, 88 (1987), pp. 67–82.
[11] A. Cichocki, *Era of big data processing: A new approach via tensor networks and tensor decompositions*, CoRR http://arxiv.org/pdf/1403.2048.pdf, (2014).
[12] A. Cichocki, *Tensor networks for big data analytics and large-scale optimization problems*, arXiv:1407.3124, (2014).
[13] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. Mandic, *Low-rank tensor networks for dimensionality reduction and large-scale optimization problems: Perspectives and challenges part 1*, CoRR, arXiv:1609.00893, (2016).
[14] A. L. F. de Almeida, G. Favier, and J. Mota, *A constrained factor decomposition with application to MIMO antenna systems*, IEEE Transactions on Signal Processing, 56 (2008), pp. 2429–2442.
[15] A. L. F. de Almeida and A. Y. Kibangou, *Distributed computation of tensor decompositions in collaborative networks*, in 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, Dec 2013, pp. 232–235.
[16] A. L. F. de Almeida and A. Y. Kibangou, *Distributed large-scale tensor decomposition*, in IEEE International Conference on Acoustics, Speech and Signal Processing, May 2014, pp. 26–30.
[17] V. De Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127.
[18] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, (1936), pp. 211–218.
[19] S. Etter, *Parallel als algorithm for solving linear systems in the hierarchical tucker representation*, SIAM J. Scientific Computing, 38 (2016), pp. 2585–2609.
[20] G. Favier and A. L. F. de Almeida, *Tensor space-time-frequency coding with semi-blind receivers for MIMO wireless communication systems*, IEEE Transactions on Signal Processing, 62 (2014), pp. 5987–6002.
[21] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 4th ed., 2013.
[22] A. Gorodetsky, S. Karaman, and Y. Marzouk, *A continuous analogue of the tensor-train decomposition*, Computer Methods in Applied Mechanics and Engineering, 347 (2019), pp. 59–84.
[23] J. H. d. M. Goulart, M. Boizard, R. Boyer, G. Favier, and P. Comon, *Tensor CP Decomposition with structured factor matrices: Algorithms and performance*, IEEE Journal of Selected Topics in Signal Processing, 10 (2016), pp. 757–769.
[24] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2029–2054.
[25] L. Grasedyck and W. Hackbusch, *An introduction to hierarchical (h-) rank and TT-rank of tensors with examples*, Comput. Meth. in Appl. Math., 11 (2011), pp. 291–304.
[26] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, CGAMM-Mitteilungen, 36 (2013), pp. 53–78.
[27] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, Journal of Fourier Analysis and Applications, 15 (2009), pp. 706–722.
[28] R. A. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.
[29] C. J. Hillar and L.-H. Lim, *Most tensor problems are NP-hard*, Journal of the ACM, 60 (2013), pp. 1–39.

[30] F. L. Hitchcock, *Multiple invariants and generalized rank of a p-way matrix or tensor*, J. of Mathematics and Physics, 7 (1927), pp. 39–79.
[31] M. Jacquelin, L. Marchal, and Y. Robert, *Complexity analysis and performance evaluation of matrix product on multicore architectures*, in International Conference on Parallel Processing, Vienna, Austria, 2009.
[32] V. Kazeev, O. Reichmann, and C. Schwab, *Low-rank tensor structure of linear diffusion operators in the TT and QTT formats*, Linear Algebra and its Applications, 438 (2013), pp. 4204 – 4221.
[33] B. Khoromskij, *O(d logN)-quantics approximation of n-d tensors in high-dimensional numerical modeling*, Constructive Approximation, 34 (2011), pp. 257–280.
[34] B. Khoromskij, *Tensors-structured numerical methods in scientific computing: Survey on recent advances*, Chemometrics and Intelligent Laboratory Systems, 110 (2011), pp. 1–19.
[35] T. G. Kolda, *A counterexample to the possibility of an extension of the eckart–young low-rank approximation theorem for the orthogonal rank tensor decomposition*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 762–767.
[36] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[37] T. G. Kolda and J. Sun, *Scalable tensor decompositions for multi-aspect data mining*, in Eighth IEEE International Conference on Data Mining, 2008.
[38] D. Kressner, M. Steinlechner, and B. Vandereycken, *Low-rank tensor completion by Riemannian optimization*, BIT Numerical Mathematics, 54 (2014), pp. 447–468.
[39] D. Kressner and C. Tobler, *Algorithm 941: h-Tucker - a matlab toolbox for tensors in hierarchical Tucker format*, Math. Softw., 40 (2014), p. 22.
[40] S. Kritchman and B. Nadler, *Non-parametric detection of the number of signals: Hypothesis testing and random matrix theory*, IEEE Transactions on Signal Processing, 57 (2009), pp. 3930–3941.
[41] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.
[42] S. L. Lauritzen, *Graphical models*, vol. 17, Clarendon Press, 1996.
[43] N. Lee and A. Cichocki, *Very large-scale Singular Value Decomposition using Tensor Train networks*, arXiv:1410.6895v2, (2014).
[44] A. P. Liavas and N. D. Sidiropoulos, *Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers*, IEEE Transactions on Signal Processing, 63 (2015), pp. 5450–5463.
[45] V.-D. Nguyen, K. Abed-Meraim, and N. Linh-Trung:, *Fast tensor decompositions for big data processing*, in International Conference on Advanced Technologies for Communications (ATC), Hanoi, Vietnam., 2016.
[46] R. Orus, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Annals of Physics, 349 (2014), pp. 117–158.
[47] I. Oseledets and E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Linear Algebra and its Applications, 432 (2010), pp. 70 – 88.
[48] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Scientific Computing, 33 (2011), pp. 2295–2317.
[49] I. V. Oseledets and E. E. Tyrtyshnikov, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Scientific Computing, 31 (2009), pp. 3744–3759.
[50] E. E. Papalexakis, C. Faloutsos, and N. Sidiropoulos, *ParCube: Sparse parallelizable tensor decompositions*, Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases, 7523 (2012), pp. 521–536.
[51] A. H. Phan and A. Cichocki, *Parafac algorithms for large-scale problems*, Neurocomputing, 74 (2011), pp. 1970 – 1984.
[52] L. Qi, Q. Wang, and Y. Chen, *Three dimensional strongly symmetric circulant tensors*, Linear Algebra and its Applications, 482 (2015), pp. 207 – 220.
[53] G. Quintana-Orti and E. S. Quintana-Orti, *Parallel codes for computing the numerical rank*, Linear algebra and its applications, 275 (1998), pp. 451–470.
[54] S. Ragnarsson and C. F. V. Loan, *Block tensors and symmetric embeddings*, Linear Algebra and its Applications, 438 (2013), pp. 853 – 874.
[55] T. Rohwedder and A. Uschmajew, *On local convergence of alternating schemes for optimization of convex problems in the tensor train format*, SIAM Journal on Numerical Analysis, 51 (2013), pp. 1134–1162.
[56] A. Sandryhaila and J. Moura, *Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure*, IEEE Signal Processing Magazine, 31 (2014), pp. 80–90.

[57] D. V. Savostyanov, *Quasioptimality of maximum-volume cross interpolation of tensors*, Linear Algebra and its Applications, 458 (2014), pp. 217–244.
[58] N. Sidiropoulos, L. D. Lathauwer, X. Fu, K. Huang, E. Papalexakis, and C. Faloutsos, *Tensor decomposition for signal processing and machine learning*, IEEE Transactions on Signal Processing, 65 (2017), pp. 3551 – 3582.
[59] N. Sidiropoulos, E. Papalexakis, and C. Faloutsos, *A parallel algorithm for big tensor decomposition using randomly compressed cubes (PARACOMP)*, in IEEE International Conference on Acoustics, Speech and Signal Processing, Florence, Italy, 2014.
[60] C. D. Silva and F. J. Herrmann, *Optimization on the hierarchical tucker manifold applications to tensor completion*, Linear Algebra and its Applications, 481 (2015), pp. 131 – 173.
[61] E. M. Stoudenmire and S. R. White, *Real-space parallel density matrix renormalization group*, Physical review B, 87 (2013).
[62] P. Strobach, *Bi-iteration svd subspace tracking algorithms*, IEEE Transactions on signal processing, 45 (1997), pp. 1222–1240.
[63] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
[64] S. Ubaru and Y. Saad, *Fast methods for estimating the numerical rank of large matrices*, in International Conference on Machine Learning, 2016, pp. 468–477.
[65] A. Uschmajew and B. Vandereycken, *The geometry of algorithms using hierarchical tensors*, Linear Algebra and its Applications, 439 (2013), pp. 133–166.
[66] L. Ximenes, G. Favier, A. L. F. de Almeida, and Y. Silva, *PARAFAC-PARATUCK semi-blind receivers for two-hop cooperative MIMO relay systems*, IEEE Transactions on Signal Processing, 62 (2014), pp. 3604–3615.
[67] C. Xu, *Hankel tensors, Vandermonde tensors and their positivities*, Linear Algebra and its Applications, 491 (2016), pp. 56 – 72.
[68] Y. Zniyed, R. Boyer, A. L. de Almeida, and G. Favier, *High-order cpd estimation with dimensionality reduction using a tensor train model*, in 26th European Signal Processing Conference (EUSIPCO), 2018.
[69] Y. Zniyed, R. Boyer, A. L. de Almeida, and G. Favier, *Multidimensional harmonic retrieval based on vandermonde tensor train*, Elsevier Signal Processing, 163 (2019), pp. 75–86.