Institute of Numerical Mathematics
Russian Academy of Sciences
119333, Moscow, Gubkina 8
www.inm.ras.ru

$$\begin{bmatrix} \mathrm{P} & \mathrm{U} & \mathrm{B} \\ \mathrm{I} & \mathrm{N} & \mathrm{M} \\ \mathrm{R} & \mathrm{A} & \mathrm{S} \end{bmatrix}$$
pub.inm.ras.ru

I.V. Oseledets, E.E. Tyrtyshnikov

Tensor tree decomposition does not need a tree

Preprint 2009-08 September 14, 2009

Moscow 2009

Tensor tree decomposition does not need a tree

I.V. Oseledets, E.E. Tyrtyshnikov

Institute of Numerical Mathematics, Russian Academy of Sciences, Russia, 119333 Moscow, Gubkina 8,
ivan.oseledets@gmail.com, tee@inm.ras.ru

August 14, 2009

[$]This work was supported by RFBR grants 08-01-00115, 09-01-00565, 09-01-12058, RFBR/DFG grant 09-01-91332 and Priority Research Programme of Department of Mathematical Sciences of Russian Academy of Sciences.

Abstract

We expose connections between *tensor networks* and recent recursive representations of high-dimensional tensors described by *binary tensor trees* and based on a sequence of skeleton (dyadic) decompositions for special unfolding matrices for a given tensor.

As the main result, we prove that a tensor decomposition by *any* binary tensor tree with certain restrictions on the distribution of spatial and auxiliary indices reduces to one and same for a particular case of tree. Since the latter tree is of simple predetermined shape, it becomes not needed at all in the construction of numerical algorithms. The tree input is replaced with a permutation of spatial indices (modes). The corresponding decomposition is given by the so-called *tensor trains* and known as *TT decomposition.*

It is observed that the transition from tensor trains with one ordering of modes to another tensor-train decomposition with some other ordering can be made by the TT-cross approximation algorithm. Also, extended tensor trees are introduced to yield the extended tensor-train decompositions with further cut in the number of representation parameters.

AMS classification: 15A12; 65F10; 65F15

1. Introduction

The history of tensor computations in linear and multilinear algebra is rather short compared to matrix computations (cf. [1]) and obviously impeded by the used low-parametric formats for representation and approximation of tensors. Certainly, save for particular cases of sparsity or shift-invariance, we never expect that an array (tensor) $A(i_1, \dots, i_d)$ with really many indices is stored as a collection of all its elements. There should be some other ways to introduce it.

The most popular ones are the *canonical* [2, 3, 4] and the *Tucker* [5] decompositions. The former reads

$$A(i_1, i_2, \dots, i_d) = \sum_{\alpha} U_1(i_1, \alpha)\, U_2(i_2, \alpha)\, \dots\, U_d(i_d, \alpha), \tag{1}$$

and the latter is of the form

$$A(i_1, \dots, i_d) = \sum_{\alpha_1, \dots, \alpha_d} G(\alpha_1, \dots, \alpha_d)\, U_1(i_1, \alpha_1)\, \dots\, U_d(i_d, \alpha_d). \tag{2}$$

Orthogonality restrictions on the factors in (2) are not essential for us (although important otherwise) and not assumed. For useful applications of both formats see [1, 6, 7, 8, 9].

Despite many crucial differences between the canonical and Tucker formats, from a certain viewpoint they are both representatives of one general approach: both can be described in the terms of *tensor networks* used as mathematical models in quantum many-body systems (cf. [10]) and in other sciences (e.g. biological neural networks). The topic of tensor networks, however, seems to be just becoming to penetrate into the communities of matrix computations and numerical analysis (cf. [11]).

What is a tensor network? Informally, we can explain this by the two examples for one and the same tensor $A(i_1, i_2, i_3, i_4, i_5)$ on Fig.1 and Fig.2.

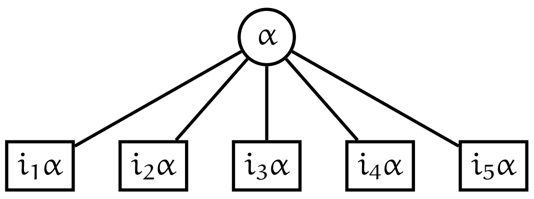

Figure 1. Tensor network for the canonical decomposition.

The indices i_1, i_2, i_3, i_4, i_5 are called *spatial indices*, other indices are called *auxiliary*. Every box represents a tensor with indices from this box. Fig.1 is associated with the canonical decomposition (1) of a tensor in $d = 5$ dimensions. Thus, the boxes are allied with the factors $U_k(i_k, \alpha)$.

Every circle is in one-to-one correspondence with some auxiliary index and indicates summation over this index of the product of tensor elements in the boxes connected to this circle. Such a summation for tensors is referred to as *contraction*. In the case

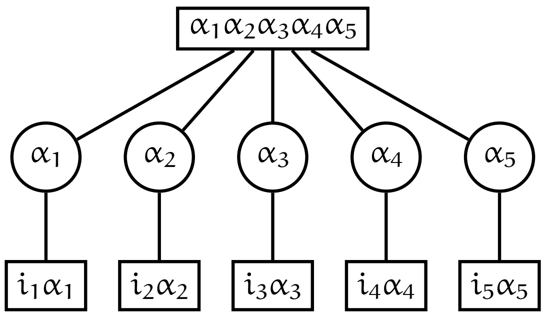

Figure 2. Tensor network for the Tucker decomposition.

of Fig.1 we have only one auxiliary index α in the circle connected to five boxes. The result of contraction over α is a tensor with all its indices being spatial. According to (1), this tensor must coincide with $A(i_1, i_2, i_3, i_4, i_5)$.

Fig.2 pertains to the Tucker decomposition (2). Now we have five auxiliary indices, so we need to perform five contractions over each of them. It is not difficult to verify that the order in which we do this plays no role. As soon as one contraction is completed, the network transforms so that the corresponding circle with its edges disappears and all connected tensors shrink down to one including all the involved indices except for the one in the removed circle. A sequence of these transformations starting from the network of Fig.2 is shown on Fig.3. Notice that the last contraction over α_5 produces the single box with the spatial indices i_1, i_2, i_3, i_4, i_5.

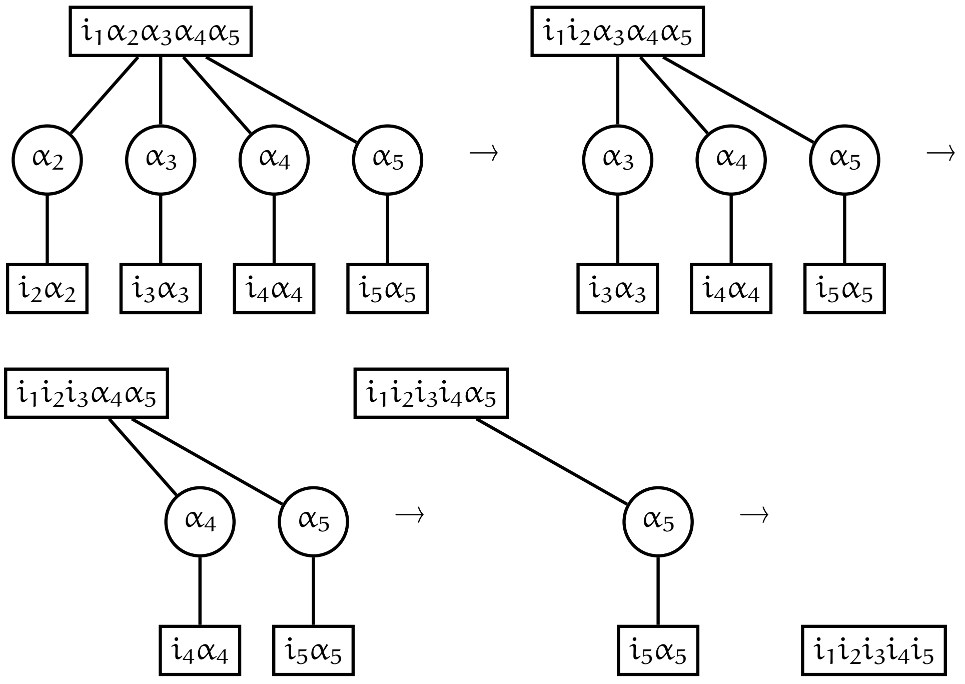

Figure 3. After successive contractions over auxiliary indices.

In general, a tensor-network for a tensor $A(i_1, \ldots, i_d)$ means that it can be defined through some other tensors (in the boxes) with spatial and auxiliary indices by summation over all auxiliary indices (in the circles)[1] of products of elements of these other tensors. In other words, the tensor is obtained via contractions of the other tensors. It is expected, of course, that one deals with those networks that provide a compressed representation for high-dimensional tensors.

Since tensor networks are conceived as tools to *introduce* tensors, it is natural to ask about what we can do with them further. In principle, we may be interested in the complexity of computation of one, several or maybe all elements of the tensor. It could heavily depend on the order of contractions (although the order does not affect the result). None the less, in view of the curse of dimensionality one should never request for *all* elements. A kind of main question seemingly raised in the literature on tensor networks is about efficient implementation of the *mode contraction*, which means computation of the scalar quantity

$$\gamma = \sum_{i_1,\ldots,i_d} A(i_1,\ldots,i_d)\; x(i_1)\; \ldots\; x_d(i_d)$$

for any given vectors $x_1,\ldots,x_d$. The chief vehicle is again the choice of order of the contractions.

However, we suggest that the truly main question ought to have been one that was not addressed properly. It should be the problem of *model reduction* by which we mean *approximation* of a given tensor network by another tensor network with *fewer* representation parameters and *faster* mode contraction. Since a given network is likely to emerge as already an approximation to some tensor, this additional transformation can be referred to as *reapproximation* or *recompression*. In this regard we must have been able to evaluate how close are two tensors and may require that our tensor networks allow us to do this efficiently and reliably (which is not at all trivial).

Moreover, it is very important to consider at least a few more operations other than mode contractions. From the viewpoint of tensor structure in matrices and vectors, three operations of primary interest should be as follows:

- Elementwise summation

$$C(i_1,\ldots,i_d) = A(i_1,\ldots,i_d) + B(i_1,\ldots,i_d).$$

- Dot product

$$\mu = \sum_{i_1,\ldots,i_d} A(i_1,\ldots,i_d)B(i_1,\ldots,i_d).$$

- Matrix-like multiplication

$$C(i_1,\ldots,i_d;\; j_1,\ldots,j_h) =$$

$$\sum_{s_1,\ldots,s_p} A(i_1,\ldots,i_d;\; s_1,\ldots,s_p)B(s_1,\ldots,s_p;\; j_1,\ldots,j_h).$$

[1]There could be several summation indices in one circle; if not originally than after transformations caused by contractions.

Here and throughout, we use semicolon to separate the indicators of row and column when a tensor is viewed as a matrix; otherwise and anyway, semicolon can be replaced with comma. As a rule, each operation must be accompanied by reapproximation for the resulting network to be described by an affordable number of parameters. If we need to figure out how close is one tensor to another one, we can perform subtraction and then estimate the Frobenius norm (square root of the sum of squared moduli of elements) or some related scalar function of the resulting tensor.

The above discussion makes it clear that *numerically viable* tensor networks cannot be arbitrary. We suggest that they should admit acceptable approximations by *sufficiently simple* networks for which all the considered requirements are met. Neither canonical, nor Tucker network tallies with those requirements: canonical decompositions are eventually lacking a robust reapproximation procedure (despite some advances [12]), and in higher dimensions the Tucker decompositions evidently suffer from the curse of dimensionality.

Suitable tensor networks with efficient and effective algorithms for the above listed problems are recently proposed in [13, 14] on the base of recursive reduction of dimensionality using some binary tree and then in [15, 16] without using any tree. In [17] the latter constructions are justly given the name of *tensor trains*.

In those works, nevertheless, the framework of tensor networks is not used, nor even mentioned. One purpose here is to expose connections between those works and tensor networks. Another intrinsically related goal is to establish that a general recursive procedure of [13, 14] via a suitable permutation of indices *always* reduces to the special case considered in [15, 16]. Thus, the latter is in effect equivalent to the general case and should be no longer considered as a particular case. Consequently, all algorithms of [15, 16] can be applied in the general case.

As a complementary goal, we consider extended tensor trees that yield the extended tensor-train decompositions with further cut in the number of representation parameters.

We also observe that the transition from tensor trains with one ordering of modes to another tensor-train decomposition with some other ordering can be made by the TT-cross approximation algorithm [17]. This algorithm can be also employed for the approximation of possibly complicated tensor networks provided that a reasonably fast algorithm exists for computation of any individual element of the tensor.

2. Dimensionality reduction and binary tensor trees

Despite the name was not mentioned therein, the subject of [13, 14] is by all means about certain tensor networks. However, those networks were not assumed to be merely given but have arisen as the result of a specific procedure of reduction of dimensionality.

The starting point in [13, 14] is the following. Given a tensor $A(i_1, \dots, i_d)$, consider it as a matrix A_k with the elements $A(i_1, \dots, i_k;\ i_{k+1}, \dots, i_d)$ and represent A_k by a skeleton (dyadic) decomposition that reads (we adopt the notation from MATLAB)

$$A_k = \sum_{\alpha=1}^{r} U(:, \alpha)\ V(\alpha, :),$$

or, in the element-wise form,

$$A(i_1, \dots, i_k;\ i_{k+1}, \dots, i_d) = \sum_{\alpha=1}^{r} U(i_1, \dots, i_k;\ \alpha)\ V(\alpha;\ i_{k+1}, \dots, i_d), \tag{3}$$

where, obviously,

$$r \geqslant \operatorname{rank} A_k. \tag{4}$$

It emanates from (3) that the tensor $A(i_1, \dots, i_d)$ is entirely defined by the tensors $U(i_1, \dots, u_k, \alpha)$ and $V(\alpha, i_{k+1}, \dots, i_d)$ of reduced dimensionality: instead of *one tensor* of dimensionality d we can deal with *two tensors* of dimensionality $k+1$ and $d-k+1$, respectively. Then, we can proceed in the same way with the tensors $U(i_1, \dots, i_k, \alpha)$ and $V(\alpha, i_{k+1}, \dots, i_d)$ and further by recursion.

In a more general setting, in order to reduce one tensor to two other tensors we split (at this stage arbitrarily) its indices into two disjoint sets and accordingly to this splitting distribute them between the two new tensors which, in addition, acquire one common *auxiliary* index. Of course, this kind of recursion is described by some binary (maybe unbalanced) tree and sets of indices assigned to each of its nodes.

In the end, tensors associated with the leafs of this tree will completely define the original tensor be the following

Basic rule: *every element* $A(i_1, \dots, i_d)$ *is the sum over all auxiliary indices of products of elements of the leaf tensors.*

The tree itself will be said about as a *tensor tree*, and the type of recursion in chime with (3) implies that this tree is binary. Beware of the difference in the meaning between such tensor trees and tensor network diagrams in spite of similarity of symbolism: now we use circles to emphasize that the node is a leaf, other nodes are marked by boxes.

To illustrate the idea on some examples, let us consider a 5-dimensional tensor $A(i_1, i_2, i_3, i_4, i_5)$. One possible way of recursion is depicted on Fig.4.

Each time we split exactly one spatial index from the others. However, in the end one of the leaf tensors is still of dimensionality 5. Note that this tensor includes only auxiliary indices. Denote it by G, and let U_k denote the leaf tensor with i_k.

In spite of our intention to reduce dimensionality we did not succeed in this with the employed way of splitting. It is interesting enough, however, that the sum of products of the leaf tensor reads

$$A(i_1, i_2, i_3, i_4, i_5) = \sum_{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5} G(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)\ U_1(i_1, \alpha_1)\ \dots\ U_5(i_5, \alpha_5)$$

and has the same form as the Tucker decomposition (2).

One might think that the failure to decrease dimensionality of all leaf tensors is explained by our decision to split exactly one spatial index from all the others. However,

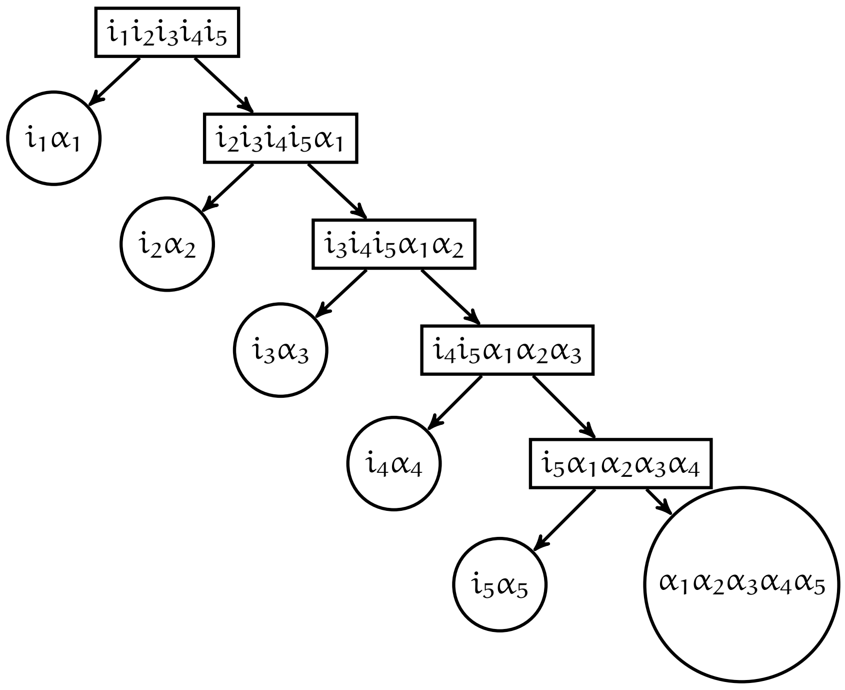
$i_1i_2i_3i_4i_5$
$i_1\alpha_1$
$i_2i_3i_4i_5\alpha_1$
$i_2\alpha_2$
$i_3i_4i_5\alpha_1\alpha_2$
$i_3\alpha_3$
$i_4i_5\alpha_1\alpha_2\alpha_3$
$i_4\alpha_4$
$i_5\alpha_1\alpha_2\alpha_3\alpha_4$
$i_5\alpha_5$
$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$

Figure 4. Tensor tree for the Tucker decomposition.

the true reason is that we splitted exactly one index at all from the others, the quality about it to be spatial was not significant.

To really decrease the number of indices in all leaf tensors and circumvent the curse of dimensionality, we have to impose some *restrictions* on the course of splitting. Let us require the following.

Restriction 1. If a tensor under reduction has exactly two auxiliary indices, then they must be relegated to different child tensors.

Restriction 2. A tensor is declared to be a leaf tensor if and only if it contains exactly one spatial index.

Lemma 1. *Let a binary tensor tree satisfy the restrictions 1 and 2. Then it possesses the following properties:*

(a) *any leaf tensor has at most two auxiliary indices;*

(b) *exactly two leaf tensors have a single auxiliary index;*

(c) *any auxiliary index belongs to one and only one pair of the leaf tensors;*

(d) *any nonempty proper subset of leafs must have at least one auxiliary index in common with a leaf outside this subset.*

Proof. We can regard the tree as the final in the sequence of intermediate trees produced at each step of reduction. Then, if a current tensor under reduction has exactly one auxiliary index then we export this index to one of the child tensors. Hence, one of the child tensors inherits this auxiliary index of the preceding tensor and obtains one more auxiliary index that appeared during the reduction. So it has exactly two auxiliary indices. The other child tensor has exactly one auxiliary index (shared by the two childes).

If a tensor under reduction has exactly two auxiliary indices, then by the restriction 1 these indices are distributed between the two childes. In addition, both child tensors obtain one more auxiliary index that accounts for the reduction. Therefore, both child tensors possess exactly two auxiliary indices.

After the initial reduction step we obtain the childes satisfying the first of the above options. Then, by induction, the two options are maintained for all subsequent child tensors, and hence, for the leaf tensors as well. This proves the claim (a). Moreover, the first option repeats itself in exactly one of the childes, and this eventually leads to the property (b). Since every auxiliary index is duplicated only once when it first appears, after each step of reduction it is shared by exactly two tensors being the leafs when this step is accomplished. This proves the property (c).

And the last, assume the contrary, e.g. that a nonempty proper subset has no index in common with any of the leafs outside this subset. Then consider all the predecessor nodes to the subset's leafs and take up one that has no predecessors. This node is obliged to be a root of the tree. Hence, the number of all spatial indices of the tensor represented by the tree must be equal to the number of the subset's leafs. Since it is at any rate the number of all leafs, we conclude that the subset is not proper, and this proves the property (d). $\square$

A possible way in which the Tucker tree of Fig.4 modifies to meet the imposed above restrictions 1 and 2 is shown on Fig.5. It is easy to check that now, in line with Lemma 1, the leaf tensors are endowed with the properties (a), (b), (c) and (d).

Let us have a look at the representation formula arising with the tree on Fig.5. Denote the leaf tensors by G_1, G_2, G_3, G_4, G_5 but think of how these letters would match the leafs. As Lemma 1 states, by the property (b) there are two tensors with a single auxiliary index, and by the property (d) they do not have this index common. According to Fig.5, these are the leafs with indices i_1, α_1 and i_2, α_2.

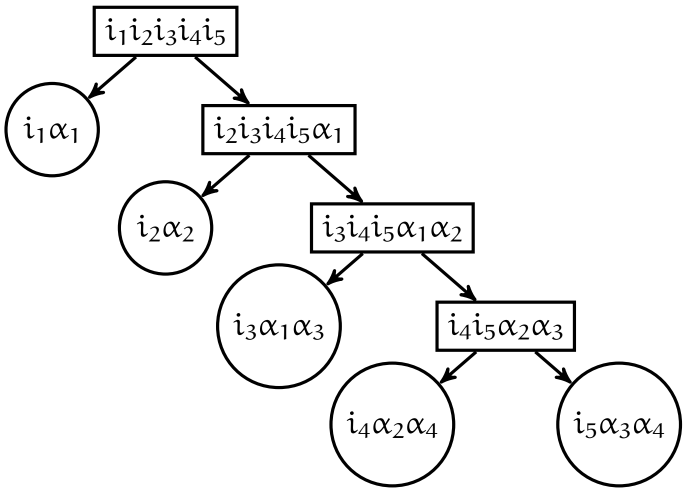

Figure 5. Tensor tree with restrictions.

Choose the leaf with indices i_1, α_1 and ally it with G_1. So the first leaf tensor is $G_1(i_1, \alpha_1)$. By the property (c), there should be exactly one more leaf containing α_1

and it will match G_2. Define the second tensor as $G_2(\alpha_1, i_3, \alpha_3)$ (we find it convenient to order the indices as we did). Again by the property (c), there is exactly one more leaf with the index α_3, in our case it will correspond to G_3. In this tensor we take the order of indices similarly as above and define the third tensor as $G_3(\alpha_3, i_5, \alpha_4)$. Then we find another leaf with α_4 and associate it with $G_4(\alpha_4, i_4, \alpha_2)$.

Next and the last, α_2 belongs to exactly one more leaf, the one with indices i_2, α_2. It will give us the last tensor $G_5(\alpha_2, i_2)$. The formula we are after now reads

$$\begin{aligned} A(i_1, i_2, i_3, i_4, i_5) = \sum_{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5} & G_1(i_1, \alpha_1)\ G_2(\alpha_1, i_3, \alpha_3) \cdot \\ & \cdot G_3(\alpha_3, i_5, \alpha_4)\ G_4(\alpha_4, i_4, \alpha_2)\ G_5(\alpha_2, i_2). \end{aligned} \tag{5}$$

By the terminology of [17], the right-hand side is a *tensor train* with *tensor carriages* G_k. Notice that every carriage is connected with its neighbor carriage by one common index, the first and last carriages have one index less as they have a neighbor only from one side (however, we could add formal additional auxiliary indices equal to 1 and make them look as all other carriages).

One might think that the restrictions are fulfilled only when we split by one spatial index during the reduction. To refute this we suggest one more example shown on Fig.6. It demonstrates that we can easily meet all the restrictions so that at each reduction step the spatial indices are split almost by half.

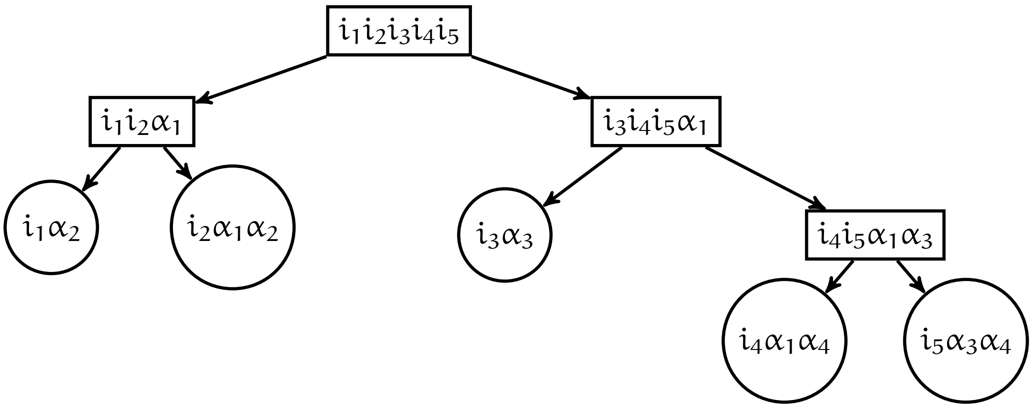

Figure 6. Tensor tree with quasi-by-half splittings.

When we split indices almost by half, we decrease the maximal dimensionality of the child tensors most rapidly. At the first glance it looks as worthy to do. However, let us consider the final tensor decomposition behind the scheme of Fig.6. As we see, we have five leaf tensors, and let them match the leafs by the method used in the previous example:

$$\begin{aligned} A(i_1, i_2, i_3, i_4, i_5) = \sum_{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5} & G_1(i_1, \alpha_2)\ G_2(\alpha_2, i_2, \alpha_1) \cdot \\ & \cdot G_3(\alpha_1, i_4, \alpha_4)\ G_4(\alpha_4, i_5, \alpha_3)\ G_5(\alpha_3, i_3). \end{aligned} \tag{6}$$

Perhaps surprisingly, with a quasi-by-half splitting we arrive at some tensor train as well. Tensor carriages in (5) and (6) may (and do) differ, of course. Nevertheless, we can find out that *precisely the same* decomposition (6) is brought about by an entirely different course of reduction shown on Fig.7, where the spatial indices are split by one.

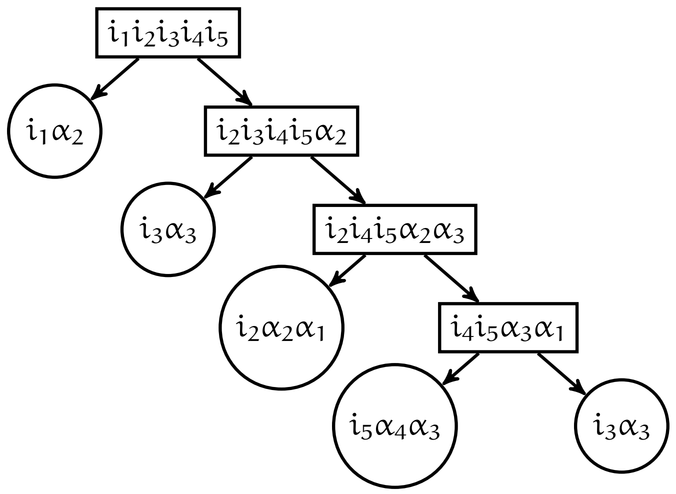

Figure 7. By-one splitting with the same result as quasi-by-half.

We are in a position now to encapsulate our observations in a general theorem. Before we do this and whatever clear might it already be from the examples, recall the definition of *tensor-train decomposition* or *TT decomposition* [15, 16, 17]:

$$A(i_1,\ldots,i_d) = \sum_{\alpha,\ldots,\alpha_{d-1}} G_1(i_1,\alpha_1)\ G_2(\alpha_1,i_2,\alpha_2)\ \ldots \\ \ldots\ G_{d-1}(\alpha_{d-2},i_{d-1},\alpha_{d-1})\ G_d(\alpha_{d-1},i_d), \tag{7}$$

or, with all carriages of identical form,

$$A(i_1,\ldots,i_d) = \sum_{\alpha,\ldots,\alpha_{d-1}} G_1(\alpha_0,i_1,\alpha_1)\ G_2(\alpha_1,i_2,\alpha_2)\ \ldots \\ \ldots\ G_{d-1}(\alpha_{d-2},i_{d-1},\alpha_{d-1})\ G_d(\alpha_{d-1},i_d,\alpha_d), \tag{8}$$

where $\alpha_0 = \alpha_d = 1$. If the initial tensor is of size $n_1 \times \ldots \times n_d$, then the tensor carriages $G_k(\alpha_{k-1},i_k,\alpha_k)$ are of sizes $r_{k-1} \times n_k \times r_k$, where r_k are called *compression ranks* and bounded from below by the ranks of unfolding matrices according to (4). By definition, $r_0 = r_d = 1$, and in case of nonzero tensors notice that $\operatorname{rank} A_0 = \operatorname{rank} A_d = 1$, for the former unfolding matrix is a single row and the latter is a single column.

Now, let σ be a permutation of integers from 1 to d. Then we can consider formally more general tensor-train decompositions of the following form:

$$A(i_1,\ldots,i_d) = \sum_{\beta_1,\ldots,\beta_{d-1}} G_1(\beta_0,i_{\sigma(1)},\beta_1)\ G_2(\beta_1,i_{\sigma(2)},\beta_2)\ \ldots \\ \ldots\ G_{d-1}(\beta_{d-2},i_{\sigma(d-1)},\beta_{d-1})\ G_d(\beta_{d-1},i_{\sigma(d)},\beta_d). \tag{9}$$

Now the size of spatial mode in G_k is obviously $n_{\sigma(k)}$. Denote the size of the tensor carriage G_k by $r^{\sigma}_{k-1} \times n_{\sigma(k)} \times r^{\sigma}_k$. As before, $r^{\sigma}_0 = r^{\sigma}_d = 1$, but other values of r^{σ}_k are not necessarily a permutation of r_k.

Let A^{σ} denote the tensor of size $n_{\sigma(1)} \times \ldots \times n_{\sigma(d)}$ with the elements

$$A^{\sigma}(i_{\sigma(1)},\ldots,i_{\sigma(d)}) = A(i_1,\ldots,i_d).$$

Lemma 2. *For any tensor-train decomposition of the form (9) the compression ranks r_k^σ satisfy the inequality*

$$r_k^\sigma \geqslant \operatorname{rank} A_k^\sigma, \quad A_k^\sigma \equiv \left[A^\sigma(i_{\sigma(1)}, \dots, i_{\sigma(k)};\ i_{\sigma(k+1)}, \dots, i_{\sigma(d)})\right], \tag{10}$$

and such a decomposition exists with

$$r_k^\sigma = \operatorname{rank} A_k^\sigma, \quad 1 \leqslant k \leqslant d.$$

Proof. Straightforwardly from (9),

$$A^\sigma(i_{\sigma(1)}, \dots, i_{\sigma(k)};\ i_{\sigma(k+1)}, \dots, i_{\sigma(d)}) =$$

$$\sum_{\beta_k=1}^{r_k^\sigma} U(i_{\sigma(1)}, \dots, i_{\sigma(k)};\ \beta_k)\ V(\beta_k;\ i_{\sigma(k+1)}, \dots, i_{\sigma(d)}),$$

where

$$\begin{aligned} U(i_{\sigma(1)}, \dots, i_{\sigma(k)};\ \beta_{k-1}) &= \sum_{\beta_1, \dots, \beta_{d-1}} G_1(\beta_0, i_{\sigma(1)}, \beta_1) \dots\ G_k(\beta_{k-1}, i_{\sigma(k)}, \beta_k), \\ V(\beta_k;\ i_{\sigma(k+1)}, \dots, i_{\sigma(d)}) &= \sum_{\beta_{k+1}, \dots, \beta_d} G_{k+1}(\beta_k, i_{\sigma(k)}, \beta_{k+1}) \dots G_d(\beta_{d-1}, i_{\sigma(d)}, \beta_d). \end{aligned}$$

This is the element-wise form of a skeleton (dyadic) decomposition for the unfolding matrix A_k^σ, and thence the minimal possible value for r_k^σ is $\operatorname{rank} A_k^\sigma$. The existence of tensor-train decomposition with minimal compression ranks can be proved by induction as in the similar statements in [15, 17]. □

Theorem 1. *The tensor decomposition associated with an arbitrary binary tensor tree with the restrictions 1 and 2 can be written as a tensor train of the form (9).*

Proof. Restriction 2 implies that for a d-dimensional tensor we always have d leafs. Thus, denote the leaf tensors by $G_1, \dots, G_d$ and look for a method to match them to the leafs. By Lemma 1, there are two leafs with a single auxiliary index. Take one of them and suppose that it has indices i_{s_1}, α_{t_1}. Then let us set

$$\sigma(1) = s_1, \quad \beta_1 = \alpha_{t_1}$$

and ally this leaf with the tensor $G_1(\beta_0, i_{\sigma(1)}, \beta_1)$. Again by Lemma 1, there must be another and only one more leaf with the index $\beta_1 = \alpha_{t_1}$. Let it be a leaf with indices i_{s_2}, α_{t_2}. Then we set

$$\sigma(2) = s_2, \quad \beta_2 = \alpha_{t_2}$$

and associate this leaf with the tensor $G(\beta_1, i_{\sigma(2)}, \beta_2)$, and so on until we encounter another leaf with a single auxiliary index. By virtue of the claim (d) of Lemma 1, this could happen only when it is the last leaf to match. □

According to Theorem 1, for any binary tensor tree the resulting tensor decomposition is entirely defined by some *permutation* of the spatial indices.

For example, for the trees on Fig.5 and on Fig.6 the permutations are respectively of the form

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 5 & 4 & 2 \end{pmatrix} \quad \text{and} \quad \sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 4 & 5 & 3 \end{pmatrix}.$$

Being more interested in the decomposition rather than in a tree behind it, we can forget that it was by construction of some tree and consider just a permutation as the only control parameter.

A tensor network behind the tensor-train decomposition (6) and the corresponding tree on Fig.6 is shown on Fig.8.

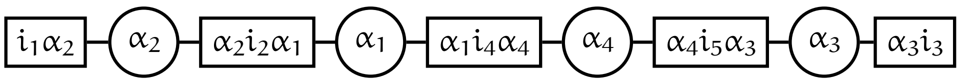

Figure 8. Tensor-train network.

A network of this kind of simple structure is sometimes referred to as a *linear tensor network* or *matrix-product network* (cf. [10, 11]). The latter name is well justified: the equation (9) is equivalent to

$$A(i_1, \dots, i_d) = G_1^{i_{\sigma(1)}} G_2^{i_{\sigma(2)}} \dots G_{d-1}^{i_{\sigma(d-1)}} G_d^{i_{\sigma(d)}}$$

with $G_k^{i_{\sigma(k)}}$ being a matrix with the elements $g_k(\beta_{k-1}, \beta_k) \equiv G_k(\beta_{k-1}, i_{\sigma(k)}, \beta_k)$.

In the language of tensor networks, the considered above process of dimensionality reduction can be viewed as a sequence of transformations that replace tensors of one network with some other networks. Consider, for example, the scheme of Fig.6 resulting in the network on Fig.8. The first reduction step yields a network with two tensors, then each of these tensors is substituted with some networks on the second reduction step, and finally one of the tensors is replaced with other network on the third step. These transformations are shown on Fig.9.

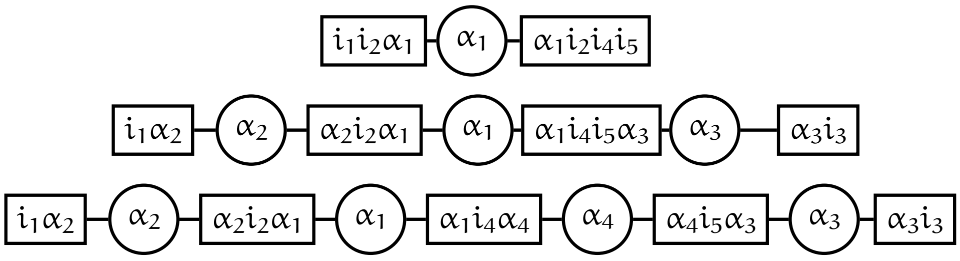

Figure 9. Reduction via transformations of tensor networks.

3. Extended tensor-tree decompositions

If each spatial index takes n values and each auxiliary index takes r values, then, by Lemma 1, the total number of entries in the leaf tensors and in the corresponding tensor-train decomposition is $2nr + (d-2)nr^2$.

If a tensor is initially given by its canonical decomposition with R terms, then it is already represented by dnR parameters. However, even in this case the number of representation parameters can be reduced if we get onto a tensor-train decomposition described by some tensor tree. It is guaranteed that we may always have $r \leqslant R$ [13], but more to that, in many cases of practical interest r turns out to be about d times less than R [13].

The problem of transition from the canonical to a general tensor-tree decomposition with reduced compression ranks was first considered and solved in [13]. The method of [13] starts with fast calculation of the *Tucker factors* $U_1, \dots, U_d$ of the Tucker decomposition (2), then the Tucker core tensor G in (2) with lesser mode sizes than the initial tensor is easily expressed in the canonical format with the same number of terms, and the recursive dimensionality reduction is then applied to it. The corresponding decomposition was called the *tree-Tucker decomposition* and contained $dnr+(d-2)r^2$ parameters[2]. The same number of parameters occurs in the hierarchical Φ-system representation[18, 19].

Nevertheless, we are going to show that such a reduced number of parameters can be achieved when a tree is still built up directly for the initial tensor. It is important for practice, because in many situations we do not have a canonical decomposition and the tensor might be given as a black box, i.e. by a procedure for computation of any required entry.

We define *extended tensor tree* as a binary tensor tree with the same restriction 1 and the restriction 2 modified as follows:

Extended restriction 2. A node is declared to be a leaf if and only if it contains one spatial and one auxiliary index or three only auxiliary indices.

An example of extended tensor tree is given on Fig.10. On the first place, here we take a tree from the previous example of reduction shown on Fig.5. Then we extend it so that any leaf with one spatial index accompanied by two auxiliary indices stops to be a leaf and gives birth to the two child tensors, one with one spatial and one auxiliary index and another one with three auxiliary indices.

We define the *extended tensor-train decomposition* with a permutation σ of the spatial indices (modes) as representation of the form

$$\begin{aligned} A(i_1, \dots, i_d) = \sum_{\gamma_1, \dots, \gamma_d} & U_1(i_{\sigma(1)}, \gamma_1)\; U_2(i_{\sigma(2)}, \gamma_2)\; \dots\; U_d(i_{\sigma(d)}, \gamma_d) \cdot \\ \cdot \sum_{\beta_1, \dots, \beta_{d-1}} & F_1(\beta_0, \gamma_1, \beta_1)\; F_2(\beta_1, \gamma_2, \beta_2)\; \dots\; F_d(\beta_{d-1}, \gamma_d, \beta_d) \end{aligned} \tag{11}$$

with $\beta_0 = \beta_d = 1$. The relation with (9) is given by the equations

$$G_k(\beta_{k-1}, i_{\sigma(k)}, \beta_k) = \sum_{\gamma_k} U_k(i_{\sigma(k)}, \gamma_k) F_k(\beta_{k-1}, \gamma_k, \beta_k). \tag{12}$$

Assume that $1 \leqslant \beta_k \leqslant r_k^\sigma$ and $1 \leqslant \gamma_k \leqslant \rho_k^\sigma$.

From Lemma 2 we already know that $r_k^\sigma \geqslant \operatorname{rank} A_k^\sigma$ and in the same spirit one can derive that $\rho_k^\sigma \geqslant \operatorname{rank} A_{(k)}^\sigma$, where $A_{(k)}^\sigma$ is the *standard unfolding matrix* for the tensor

[2]Under the assumption that the Tucker ranks for all modes are equal.

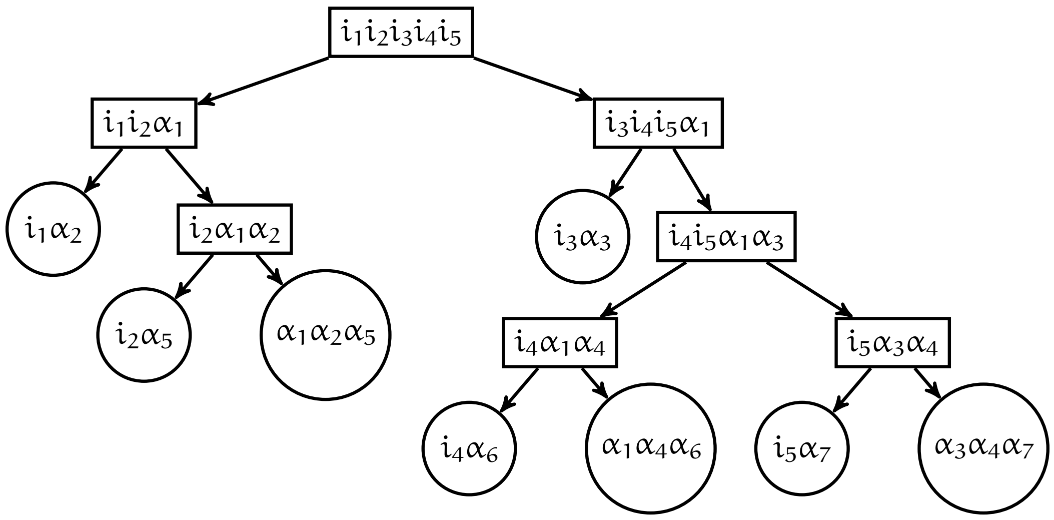

Figure 10. Extended tensor tree.

A^σ with respect to the mode k (it is a matrix with the column size equal to the size of the kth mode, the columns themselves are all fibres of the kth mode). We refer to ρ_k^σ as *mode compression ranks* and to r_k^σ as *tensor-train compression ranks*. A decomposition with all mode ranks and all tensor-train ranks attaining the above lower bounds is called *minimal*.

Lemma 3. *Any tensor can be represented by a minimal extended tensor-train decomposition.*

Proof. We know that the Tucker decomposition with minimal possible mode ranks always exists [20]. Let us write it in the form

$$A(i_1,\dots,i_d) = \sum_{\gamma_1,\dots,\gamma_d} G(\gamma_1,\dots,\gamma_d)\; U_1(i_{\sigma(1)},\gamma_1)\;\dots\;U_d(i_{\sigma(d)},\gamma_d).$$

Hence,

$$\operatorname{rank} A_k^\sigma \leqslant \operatorname{rank} G_k, \quad G_k = [G(i_1,\dots,i_k;\; i_{k+1},\dots,i_d)].$$

The minimality implies that the matrices $U_1,\dots,U_d$ are of full column rank and we do not lose generality if take them with orthonormal columns (with a correspondingly changed G). Consequently (overlined numbers mean complex conjugates),

$$G(\gamma_1,\dots,\gamma_d) = \sum_{i_1,\dots,i_d} A(i_1,\dots,i_d)\;\overline{U}_1(i_{\sigma(1)},\gamma_1)\;\dots\;\overline{U}_d(i_{\sigma(d)},\gamma_d),$$

and thence $\operatorname{rank} G_k \leqslant \operatorname{rank} A_k^\sigma$. Thus we infer that $\operatorname{rank} A_k^\sigma = \operatorname{rank} G_k$, and it remains to remark that the TT decomposition for G with minimal compression ranks always exists. $\square$

Theorem 2. *Let a tensor possess a canonical decomposition with R terms. Then there is a tensor tree with the restrictions 1 and 2 fulfilled and a tensor-train decomposition* *with all compression ranks not greater than* R*. Such a tree with minimal compression ranks can be extended in line with the extended restriction 2 so that all mode compression ranks do not exceed* R*.*

Proof. The first assertion stems from Lemma 3. The second claim is equivalent to the following property of tensor carriages: if a tensor-train decomposition (7) is with minimal possible compression ranks, then

$$\operatorname{rank}[G_k]_{(2)} = \operatorname{rank} A_{(k)},$$

where $[G_k]_{(2)}$ is the second standard unfolding matrix for the tensor G_k with respect to the second mode, i.e. it is a matrix with the elements

$$b(i_k;\ \alpha_{k-1}, \alpha_k) = G_k(\alpha_{k-1}, i_k, \alpha_k),$$

and similarly, $A_{(k)}$ is the kth standard unfolding matrix for the tensor $A(i_1, \ldots, i_d)$. Analogously to (10), we have $\operatorname{rank}[G_k]_{(2)}] \geqslant \operatorname{rank} A_{(k)}$. To prove the equality, from the minimality of tensor-train representation we deduce that

$$A(i_1 \ldots i_{k-1},\ i_k,\ i_{k+1} \ldots i_d) =$$

$$\Phi_k(i_1 \ldots i_{k-1}; \alpha_{k-1} G_k(\alpha_{k-1}, i_k, \alpha_k) \Psi_k(\alpha_k;\ i_{k+1} \ldots i_d)$$

for some matrices Φ_k and Ψ_k with linear independent columns and linear independent rows, respectively. Denote by $\Phi^\dagger$ and $\Psi^\dagger$ the corresponding pseudo-inverse matrices that satisfy the equations

$$\Phi^\dagger \Phi = I, \quad \Psi \Psi^\dagger = I$$

and have elements $\Phi^\dagger(\alpha_{k-1};\ i_1 \ldots i_{k-1})$ and $\Psi^\dagger(i_{k+1} \ldots i_d;\ \alpha_k)$, respectively. Then the above expression for $A(i_1 \ldots i_d)$ entails

$$G_k(\alpha_{k-1}, i_k, \alpha_k) =$$

$$\sum_{i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_d} \Phi^\dagger(\alpha_{k-1};\ i_1 \ldots i_{k-1})\ A(i_1 \ldots i_{k-1},\ i_k,\ i_{k+1} \ldots i_d) \Psi^\dagger(i_{k+1} \ldots i_d;\ \alpha_k).$$

Hence,

$$\operatorname{rank}[G_k]_{(2)} \leqslant \operatorname{rank} A_k,$$

and since the opposite inequality is already ensured, both sides are equal. □

Tensor trains can be also used for *approximation* of a tensor given as a black box, i.e. by a procedure for computation if its entries, or by some representation scheme, e.g. by a canonical decomposition. The purpose of approximation is to reduce the number of representation parameters. It is worth noting that both tensor trains and extended tensor trains are free from the ill-posedness property of the canonical approximations of a fixed rank [21]. The latter means that a sequence of tensors of canonical rank bounded by R can converge to a tensor with the rank strictly larger than R. In [17] we proved that this cannon happen with the tensor trains with bounded compression ranks.

The reason is, in fact, that it cannot do with matrices and that compression ranks are defined as ranks of some unfolding matrices. The same reason applies to the extended tensor trains with bounded mode and tensor-train compression ranks. Thus, within those bounds on the mode and tensor-train ranks a minimizer of the Frobenius-norm distance between such approximants and the given tensor does always exist.

4. Discussion

The main message of this paper (Theorem 1) is that the construction of general tensor-tree decompositions does need a tree *at any stage*. All we need to know is just a permutation of modes (spatial indices). This simplifies the logic behind algorithms and results in easier programming and better performance.

All basic operations listed in the introduction have got efficient and reliable algorithms, the key to them being definitely the tensor-train recompression algorithm [15]. And all these algorithms can now be applied to any case of dimensionality reduction by binary tensor trees.

Note that we can pass from tensor trains with one ordering of modes to another tensor-train decomposition with some other ordering by using, for example, the TT-cross approximation algorithm [17].

References

[1] *Bader B. W., Kolda T. G.* Tensor decompositions and applications // *SIAM Review*. 2009. — Sep. V. 51, № 3. to appear.

[2] *Carroll J. D., Chang J. J.* Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart-Young decomposition // *Psychometrika*. 1970. V. 35. P. 283-319.

[3] *Harshman R. A.* Foundations of the Parafac procedure: models and conditions for an explanatory multimodal factor analysis // *UCLA Working Papers in Phonetics*. 1970. V. 16. P. 1-84.

[4] *Hitchcock F. L.* The expression of a tensor or a polyadic as a sum of products // *J. Math. Phys.* 1927. V. 6. P. 164-189.

[5] *Tucker L. R.* Some mathematical notes on three-mode factor analysis // *Psychometrika*. 1966. V. 31. P. 279-311.

[6] *Comon P.* Tensor decomposition: state of the art and applications // IMA Conf. Math. in Sig. Proc., Warwick, UK. — 2000.

[7] *Flad H.-J., Khoromskij B. N., Savostyanov D. V., Tyrtyshnikov E. E.* Verification of the cross 3D algorithm on quantum chemistry data // *Rus. J. Numer. Anal. Math. Model.* 2008. V. 23, № 4. P. 210-220.

[8] *Hackbusch W., Khoromskij B. N., Sauter S. A., Tyrtyshnikov E. E.* Use of Tensor Formats in Elliptic Eigenvalue Problems: Preprint 78. — Leipzig: MIS MPI, 2008. www.mis.mpg.de/preprints/2008/preprint2008_78.pdf.

[9] *Hackbush W., Khoromskij B. N., Tyrtyshnikov E. E.* Hierarchical Kronecker tensor-product approximations // *J. Numer. Math.* 2005. V. 13. P. 119-156.

[10] *Hübener R., Nebendahl V., Dür W.* Concatenated tensor network states. — arXiv:0904.1925v1 [quant-ph], 2009.

[11] *Van Loan C. F.* Tensor network computations in quantum chemistry. — www.cs.cornell.edu/cv/OtherPdf/ZeuthenCVL.pdf, 2008. www.cs.cornell.edu/cv/OtherPdf/ZeuthenCVL.pdf.

[12] *Espig M.* Effiziente Bestapproximation mittels Summen von Elementartensoren in hohen Dimensionen: Ph.D. thesis / Leipzig. — 2007.

[13] *Oseledets I. V., Tyrtyshnikov E. E.* Breaking the curse of dimensionality, or how to use SVD in many dimensions // *SIAM J. Sci. Comp.* 2009. http://pub.inm.ras.ru/preprint-2009-02.

[14] *Oseledets I. V., Tyrtyshnikov E. E.* Recursive decomposition of multidimensional tensors // *Doklady Math.* 2009.

[15] *Oseledets I. V.* Compact matrix form of the d-dimensional tensor decomposition: Preprint 2009-01: INM RAS, 2009. — Mar. http://pub.inm.ras.ru/preprint-2009-01.

[16] *Oseledets I. V.* On a new tensor decomposition // *Doklady Math.* 2009.

[17] *Oseledets I. V., Tyrtyshnikov E. E.* TT-Cross approximation for multidimensional arrays // *Linear ALgebar Appl.* 2009. www.math.hkbu.edu.hk/ICM/pdf/09-11.pdf.

[18] *Hackbusch W., Kūhn S.* A new scheme for the tensor representation: Preprint 2. — Leipzig: MPI MIS, 2009. www.mis.mpg.de/preprints/2009/preprint2009_3.pdf.

[19] *Grasedyck L.* Hierarchical singular value decomposition of tensors: Preprint 27. — Leipzig: MPI MIS, 2009. www.mis.mpg.de/preprints/2009/preprint2009_3.pdf.

[20] *de Lathauwer L., de Moor B., Vandewalle J.* A multilinear singular value decomposition // *SIAM J. Matrix Anal. Appl.* 2000. V. 21. P. 1253-1278.

[21] *de Silva V., Lim L.-H.* Tensor rank and the ill-posedness of the best low-rank approximation problem // *SIAM J. Matrix Anal. Appl.* 2008. V. 30, № 3. P. 1084-1127.