HAL open science

Parallel Tensor Train through Hierarchical Decomposition

Laura Grigori, Suraj Kumar

To cite this version:

Laura Grigori, Suraj Kumar. Parallel Tensor Train through Hierarchical Decomposition. 2021. hal-03081555v3

HAL Id: hal-03081555

https://inria.hal.science/hal-03081555v3

Preprint submitted on 19 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Parallel Tensor Train through Hierarchical Decomposition

Laura Grigori[1] and Suraj Kumar[1]

Alpines, Inria Paris,
Sorbonne Université, Université de Paris, CNRS,
Laboratoire Jacques-Louis Lions, Paris, France
Email:{laura.grigori,suraj.kumar}@inria.fr

Abstract. We consider the problem of developing parallel decomposition and approximation algorithms for high dimensional tensors. We focus on a tensor representation named Tensor Train (TT). It stores a d-dimensional tensor in $\mathcal{O}(nr^2d)$, much less than the $\mathcal{O}(n^d)$ entries in the original tensor, where r is usually a very small number and depends on the application. Sequential algorithms to compute TT decomposition and TT approximation of a tensor have been proposed in the literature. Here we propose a parallel algorithm to compute TT decomposition of a tensor. We prove that the ranks of TT-representation produced by our algorithm are bounded by the ranks of unfolding matrices of the tensor. Additionally, we propose a parallel algorithm to compute approximation of a tensor in TT-representation. Our algorithm relies on a hierarchical partitioning of the dimensions of the tensor in a balanced binary tree shape and transmission of leading singular values of associated unfolding matrix from the parent to its children. We consider several approaches on the basis of how leading singular values are transmitted in the tree. We present an in-depth experimental analysis of our approaches for different low rank tensors and also assess them for tensors obtained from quantum chemistry simulations. Our results show that the approach which transmits leading singular values to both of its children performs better in practice. Compression ratios and accuracies of the approximations obtained by our approaches are comparable with the sequential algorithm and, in some cases, even better than that. We also show that our algorithms transmit only $\mathcal{O}(\log^2 P \log d)$ number of messages along the critical path for a d-dimensional tensor on P processors. The lower bound on the number of messages for any algorithm which exchanges data on P processors is $\Omega(\log P)$, and our algorithms achieve this bound, modulo polylog factor.

Keywords: Tensor Decompositions · Tensor Train Representation · Parallel Algorithms · Low Rank Approximations · Compression Ratios.

1 Introduction

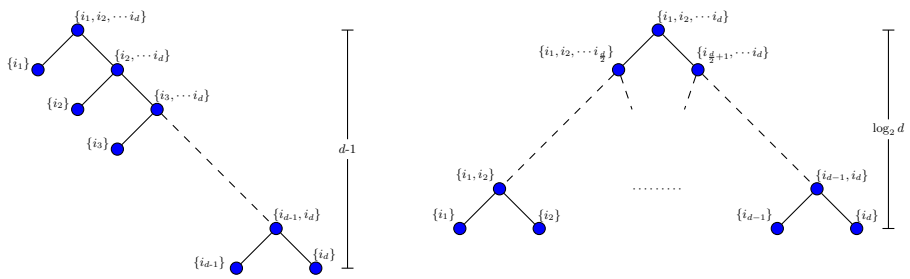
Multidimensional data is ubiquitous in scientific computing and data analysis. Tensors are becoming a popular choice in recent years to represent and manipu-

late such data. Tensor decomposition and approximation play important roles in several domains, for instance, quantum molecular dynamics, signal processing, data mining, neurosciences, computer vision, psychometrics, chemometrics and more. Tensor decomposition based methods are also being used in drug discovery for the current pandemic [29]. We point the reader to [20] for a nice survey on tensor decompositions and applications.

Historically there has been a great emphasis on minimizing the complexity of computations. With the advent of multicores, the focus has shifted towards developing parallel algorithms. In recent years, the number of computational cores has increased dramatically and communication becomes bottleneck for an application. Thus the focus is now moving towards developing parallel and communication optimal algorithms [10,4].

Historically there has been a great emphasis on minimizing the complexity of computations. With the advent of multicores, the focus has shifted towards developing parallel algorithms. In recent years, the number of computational cores has increased dramatically and communication becomes bottleneck for an application. Thus the focus is now moving towards developing parallel and communication optimal algorithms [10,4].

Recent advances in high performance architectures make us enable to find efficient solutions for some of the most challenging scientific problems. Solving problems with large tensors on such architectures is still tough due to their large computational effort and memory requirements (amount of memory and computations grow exponentially in number of dimensions). For example, a molecular simulation involving just 100 spatial orbitals requires one to manipulate a 100-dimensional tensor with 4^{100} elements. These problems can not be tackled directly. It is therefore necessary to exploit patterns of the data. Finding low dimensional structure of high dimensional data is a powerful approach in this context. Several tensor representations such as CP, Tucker, Tensor-Train are based on low dimensional structure of the tensors. In this article, we concentrate on one of the tensor representations, namely, Tensor Train (TT). This representation and an algorithm to compute it were proposed in [26]. It represents a d-dimensional tensor with 2 matrices and d-2 3-dimensional tensors. We call the algorithm TT-decomposition. The algorithm works in d-1 steps. In each step, one dimension is separated from the remaining tensor. Figure 1a shows the separation of dimensions of a d-dimensional tensor by this algorithm. Even if each step of the TT-decomposition can be parallelized, the separation of each dimension from the remaining ones is an intrinsically sequential process. This is reflected by the depth of the tree which is equal to one less than the number of dimensions of the tensor. From this figure, we also observe that we could achieve maximum parallelism if the tree is structured such that the width is full at each level. Based on this observation, we propose a parallel algorithm to compute TT-representation of a tensor and we call it Parallel Tensor Train (PTT) decomposition throughout the text. Designing communication optimal algorithms to compute TT-representation of a tensor is a part of our future work. Our PTT decomposition algorithm divides the tensor into two subtensors



(a) Splitting of dimensions in TT-decomposition.

(b) Splitting of dimensions in PTT-decomposition.

Fig. 1: Splitting of dimensions in TT and PTT decompositions for a d-dimensional tensor. Each node shows a set of dimensions associated with it.

at each level. It exposes parallelism in a balanced binary tree shape and has maximum parallelism at the last level of the tree. Figure 1b shows the splitting of dimensions by this algorithm. We prove that the ranks of TT-representation computed by this algorithm are bounded by the ranks of unfolding matrices of the tensor. We also propose a parallel algorithm to compute approximation of a tensor in TT-representation. Our algorithm relies on a hierarchical partitioning of the dimensions of the tensor in a balanced binary tree shape and transmission of leading singular values of associated unfolding matrix from the parent to its children. We consider several approaches based on how leading singular values are transmitted in the tree and evaluate them for different tensor sizes. Our results show that the approach which transmits leading singular values to both of its children performs better in practice. We also show that our algorithms transmit only $\mathcal{O}(\log^2 P \log d)$ number of messages along the critical path for a d-dimensional tensor on P processors.

The rest of the paper is organized as follows. Section 2 describes some popular tensor decompositions and their representations. In Section 3, we first present different notations required to express our algorithms, and then explain TT-representation and TT-decomposition algorithm. Parallel algorithms to compute decomposition and approximation of a tensor in TT-representation are presented in Sections 4 and 5 respectively. In Section 6, we perform an assessment of our approximation algorithm for several low rank tensors. We finally propose conclusions and perspectives in Section 7.

2 Related Work

Tensor decomposition was first introduced by *Hitchcock* in 1927 [16]. This is now known as canonical polyadic (CP) decomposition (or CANDECOMP/PARAFAC). It was rediscovered several times, mainly in psychometrics literature [7,15]. It

can be viewed as high order generalization of singular value decomposition (SVD). This represents a d-dimensional tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ with elements $\mathbf{A}(i_1, \cdots, i_d)$ as, $\mathbf{A}(i_1, \cdots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha) \cdots U_d(i_d, \alpha)$. The minimum number of r required to express $\mathbf{A}$ is called the canonical rank. The matrices $[U_k(i_k, \alpha)] \in \mathbb{R}^{n_i \times r}$, for $1 \leq k \leq d$, are called canonical factors. The number of entries in the decoupled representation is $\mathcal{O}(nrd)$, much less than the $\mathcal{O}(n^d)$ entries in the original tensor. The most fascinating example of this decomposition in algorithmic complexity is Strassen matrix multiplication, which represents a 2×2 matrix multiplication as a decomposition of a $4 \times 4 \times 4$ tensor [21]. This decomposition suffers from several drawbacks. Determining the canonical rank of a tensor is an NP-complete problem [17] and 3 or higher dimensional tensors can fail to have best approximations for a fixed canonical rank [27]. There are several algorithms to compute CP decomposition of a tensor for a fixed canonical rank [7,24,28]. The most popular one is alternating least squares. CP decomposition has also been considered for sparse tensors [19]. It is difficult to apply conventional algorithms for large sparse tensors. Hence randomized algorithms are being employed to compute CP decomposition of these tensors [22].

Matricized tensor times Khatri-Rao product (MTTKRP) is a key and time consuming operation in algorithms for computing CP decomposition of a tensor. There has been a lot of work in this context. Communication lower bounds for this operation have been established recently. Parallel and sequential algorithms that attain the lower bounds also have been presented [2]. Hypergraph based methods were proposed in [18] to balance the load and reduce the communication requirements of MTTKRP for distributed memory systems. A load balanced implementation of MTTKRP on GPUs is presented in [25]. An implementation of MTTKRP with hierarchical storage for sparse tensors has been presented recently in [23].

Another popular tensor decomposition is Tucker decomposition [30]. This is also viewed as high order generalization of singular value decomposition. It decomposes a tensor into a set of matrices and a small core tensor. It is represented as, $\mathbf{A}(i_1, \cdots, i_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_d=1}^{r_d} g_{\alpha_1 \cdots \alpha_d} U_1(i_1, \alpha_1) \cdots U_d(i_d, \alpha_d)$, where $\{r_i\}_{1 \leq i \leq d}$ is a set of ranks. Tucker approximations can be computed by several SVDs for auxiliary matrices. The approximation computed by HOSVD algorithm increases the error at most by $\sqrt{d}$ with respect to the optimal error [8]. For $r_1 = r_2 = \cdots = r_d = r$, Tucker approximations store $\mathcal{O}(ndr + r^d)$ entries. As the number of entries in this format is exponential in the number of dimensions, it is suitable for small dimensions, but not for large dimensions.

Tucker representation has further been improved in [14,12], and called Hierarchical Tucker (or H-Tucker). This is represented by a tree. Matrices of Tucker decomposition represent leaf nodes of this tree. Internal nodes correspond to the core tensor of the Tucker decomposition. Each internal node except root contains a 3-dimensional tensor. Root node contains a matrix. The number of entries in this representation is $\mathcal{O}(ndr + dr^3)$.

As mentioned in the Introduction, TT-decomposition is a recently proposed tensor decomposition algorithm. We explain this algorithm in details in the next

section. The TT-representation is known in the quantum chemistry community from a long time by the name of matrix product states [11]. A communication efficient parallel algorithm to perform rounding in TT-format has recently been proposed [3].

3 Notations & TT-Representation

Let $\mathbf{A}(i_1, i_2, \cdots, i_d)$ denote the elements of a d-dimensional tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. We use bold letters to denote tensors. The k-th unfolding matrix of tensor $\mathbf{A}$ is represented by A_k.

$$A_k = [A_k(i_1, i_2, \cdots, i_k; i_{k+1}, \cdots, i_d)]$$

The first k indices represent the rows of A_k and the last d-k the columns of A_k. The size of this matrix is $(\prod_{l=1}^{k} n_l) \times (\prod_{l=k+1}^{d} n_l)$. The rank of this matrix is denoted by r_k. Let $(r_1, r_2, \cdots, r_{d-1})$ denote the ranks of unfolding matrices of tensor $\mathbf{A}$.

Let $A(i; j)$ denote the element of ith row and jth column of a matrix A. i and j can be set of indices. For example, row and column indices of element $A_k(i_1, i_2, \cdots, i_k; i_{k+1}, \cdots, i_d)$ are the entries corresponding to $(i_1, i_2, \cdots, i_k)$ and $(i_{k+1}, \cdots, i_d)$ in rows and columns of A_k respectively.

Let $||\mathbf{A}||_F$ denote the frobenius norm of a d-dimensional tensor $\mathbf{A}$ and it is defined as, $||\mathbf{A}||_F = \sqrt{\sum_{i_1, i_2, \cdots, i_d} A(i_1, i_2, \cdots, i_d)^2}$. $||A||_2$ and $||A||_F$ denote the spectral norm and the frobenius norm of a matrix A and they are defined as, $||A||_2 =$ *maximum singular value of* A and $||A||_F = \sqrt{\sum_{i,j} A(i; j)^2}$. Let $tr(A)$ denote the trace of a square matrix A and it is computed as, $tr(A) = \sum_i A(i; i)$.

3.1 TT-Representation

An efficient representation of a tensor with small number of variables provides opportunities to work with high dimensional tensors. TT-representation is a popular way to represent such tensors with few number of variables, as discussed in [26]. It represents a d-dimensional tensor with 2 matrices and d-2 3-dimensional tensors. These are called cores of the TT-representation.

In TT-representation, a d-dimensional tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is represented with cores $\mathbf{G}_k$ of size $r_{k-1} \times n_k \times r_k$, $k = 1, 2, \cdots d$, $r_0 = r_d = 1$ and its elements satisfy the following expression:

$$\begin{aligned}\mathbf{A}(i_1, \cdots, i_d) &= \sum_{\alpha_0=1}^{r_0} \cdots \sum_{\alpha_d=1}^{r_d} \mathbf{G}_1(\alpha_0, i_1, \alpha_1) \cdots \mathbf{G}_d(\alpha_{d-1}, i_d, \alpha_d) \\ &= \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \mathbf{G}_1(1, i_1, \alpha_1) \cdots \mathbf{G}_d(\alpha_{d-1}, i_d, 1)\end{aligned}$$

Here $\mathbf{G}_k(\alpha_{k-1}, i_k, \alpha_k)$ are the elements of $\mathbf{G}_k$. Since $r_0 = r_d = 1$, $\mathbf{G}_1$ and $\mathbf{G}_d$ cores are matrices. The other cores are 3-dimensional tensors. First and last

cores are matrices, we still use bold letters to represent all cores. For $n_1 = n_2 = \cdots = n_d = n$ and $r_1 = r_2 = \cdots = r_{d-1} = r$, the number of entries in this representation is $\mathcal{O}(ndr^2)$.

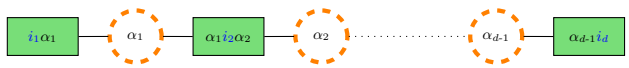
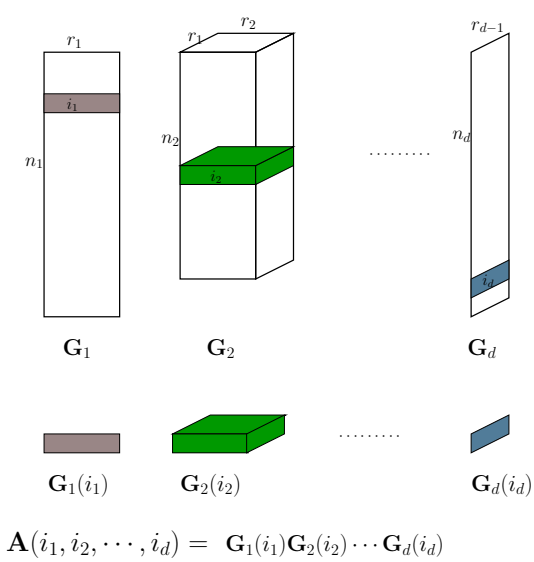


Fig. 2: Chain diagram of TT-representation.

Figure 2 exhibits that the above expression can also be represented graphically by a linear chain where sum over circle nodes (indices α_k) are assumed to compute an entry of the tensor. This figure looks like a train, hence the tensor train name is used for the representation.

Figure 3 also shows how an entry of a tensor can be computed with the cores of TT-representation. With a slight abuse of notation, $\mathbf{G}_k(i_k)$ denotes the i_kth slice of kth core of TT-representation.



$$\mathbf{A}(i_1, i_2, \cdots, i_d) = \mathbf{G}_1(i_1)\mathbf{G}_2(i_2)\cdots\mathbf{G}_d(i_d)$$

Fig. 3: TT-representation of a d-dimensional tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. An entry of the tensor is computed by multiplying corresponding matrix (or row/column) of each core.

3.2 TT-Decomposition

As mentioned earlier, the TT-Decomposition algorithm has been proposed in [26] to compute TT-representation of a tensor. Here we present the main idea of the algorithm. Let $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ be a d-dimensional tensor. The algorithm operates in d-1 steps. In each step, one dimension of the tensor is decoupled from the remaining dimensions using SVD. In the first step, $\mathbf{A}$ is represented as a matrix of size $n_1 \times n_2 n_3 \cdots n_d$ and SVD is computed for this matrix. The left singular vectors corresponding to nonzero singular values are the first core of TT-representation. In the second step, interaction with the first core and the second dimension are represented as rows of the matrix while other dimensions $n_3 \cdots n_d$ represent the columns. Again SVD is computed for this matrix and left singular vectors are rearranged to obtain the second core of TT-representation. This process is repeated for d-1 steps. Hence d-1 cores are obtained by this process. Remaining matrix, which represents the interaction with the (d-1)th core and n_d, constitutes the last core of the representation. This algorithm produces cores $\mathbf{G}_k(\alpha_{k-1}, n_k, \alpha_k)_{1 \leq k \leq d}$ and also ensures that $\alpha_k \leq r_k$. We refer the reader to the original paper for more details about this algorithm.

The separation of each dimension from the remaining ones is a sequential process in TT-decomposition. Most modern computing platforms are composed of several number of nodes and cores. Hence running a sequential process on these platforms may result in poor utilization of resources. In the next section, we propose a parallel algorithm to obtain TT-representation of a tensor.

4 Parallel Tensor Train Decomposition

The original indices of a tensor are called external indices, while the indices obtained due to SVD are called internal indices. $nEI(\mathbf{A})$ denotes the number of external indices of a tensor $\mathbf{A}$. A tensor with elements $\mathbf{A}(\alpha, i_1, i_2, i_3, \beta)$ has 3 external and 2 internal indices. We also extend the definition of unfolding matrix to take internal indices into account. The k-th unfolding of tensor $\mathbf{A}$, whose elements are $\mathbf{A}(\alpha, i_1, i_2, \cdots, i_k, i_{k+1}, \cdots, \beta)$, is represented as, $A_k = [A_k(\alpha, i_1, i_2, \cdots, i_k; i_{k+1} \cdots, \beta)]$. Here we consider all indices from the beginning to i_k as the rows of A_k and the remaining indices as the columns of A_k.

Let $\mathbf{Tensor}(A_l)$ convert an unfolding matrix A_l to its tensor form. For example, if $A_l(\alpha, i_1, \cdots, i_l; i_{l+1} \cdots, i_m, \beta)$ represent the elements of an unfolding matrix A_l then $\mathbf{Tensor}(A_l)$ produces a tensor $\mathbf{A}$ with elements $\mathbf{A}(\alpha, i_1, \cdots, i_l, i_{l+1}, \cdots, i_m, \beta)$.

We present a parallel algorithm to compute TT-representation of a tensor in Algorithm 1. It is possible to work directly with unfolding matrices, however for the ease of presentation, intermediate unfolding matrices are converted to tensors. We can also note that selection of X_k, S_k and Y_k is not specified in line number 6. Our proof applies no matter how these are chosen. However, the practical performance of the approximation algorithm, which is based on this

Algorithm 1 PTT-decomposition (parallel Tensor Train Decomposition)

Require: d-dimensional tensor $\mathbf{A}$ and ranks $(r_1, r_2, \cdots r_{d-1})$
Ensure: Cores $\mathbf{G}_k(\alpha_{k-1}, n_k, \alpha_k)_{1\leq k\leq d}$ of the TT-representation with $\alpha_k \leq r_k$ and $\alpha_0 = \alpha_d = 1$

if $nEI(\mathbf{A})$ >1 **then**
  Find the middle external index k
  Compute unfolding matrix A_k
  Compute SVD: $A_k = U\Sigma V^T$
  Compute rank of Σ, $\alpha_k = \text{rank}(\Sigma)$
  Select diagonal matrices X_k, S_k and Y_k such that $X_k S_k Y_k = \Sigma(1:\alpha_k; 1:\alpha_k)$
  $\mathbf{A}_{left} = \mathbf{Tensor}(U(;1:\alpha_k)X_k)$
  list1 = PTT-decomposition($\mathbf{A}_{left}$, $(r_1, \cdots r_{k-1}, \alpha_k)$)
  $\mathbf{A}_{right} = \mathbf{Tensor}(Y_k V^T(1:\alpha_k;))$
  list2= PTT-decomposition($\mathbf{A}_{right}$,$(\alpha_k, r_{k+1}, \cdots r_{d-1})$)
  return {list1, list2}
else
  Find the external index k
  if k is the last index of $\mathbf{A}$ **then**
    $\alpha_k = 1$
  else if k is the first index of $\mathbf{A}$ **then**
    $\alpha_{k\text{-}1} = 1$
    $\mathbf{A}(i_k, \beta) = \sum_{\beta=1}^{\alpha_k} \mathbf{A}(i_k, \beta) S_k(\beta; \beta)$
  else
    $\mathbf{A}(\gamma, i_k, \beta) = \sum_{\beta=1}^{\alpha_k} \mathbf{A}(\gamma, i_k, \beta) S_k(\beta; \beta)$
  end if
  $\mathbf{G}_k = \mathbf{A}$
  return $\mathbf{G}_k$
end if

algorithm, depends on the selection of these matrices. In Section 6, we compare three options: i) $X_k = S_k = I$, $Y_k = \Sigma(1:\alpha_k; 1:\alpha_k)$ ii) $X_k = Y_k = \Sigma(1:\alpha_k; 1:\alpha_k)^{1/2}$, $S_k = I$ iii) $X_k = Y_k = \Sigma(1:\alpha_k; 1:\alpha_k)$, $S_k = \Sigma(1:\alpha_k; 1:\alpha_k)^{-1}$. In most cases, third option is often the better choice.

In PTT-decomposition algorithm, if the input tensor $\mathbf{A}$ has more than one external index then first we compute the middle external index k. After that, we compute the rank α_k and the SVD of the unfolding matrix A_k. Three diagonal matrices X_k, S_k and Y_k are chosen such that their product corresponds to the matrix obtained by selecting leading α_k rows and columns of the singular value matrix. Selection of the diagonal matrices is related to how singular values will be transferred in recursive calls. Now, a matrix containing leading α_k left singular vectors is multiplied by X_k. The result is converted in tensor format and PTT-decomposition algorithm is called for this subtensor. Similarly, Y_k is multiplied by a matrix which contains the transpose of leading α_k right singular vectors, and again PTT-decomposition algorithm is called with the tensor format of the result. When the call for both functions returns, we receive cores of TT-

representations of both subtensors. We combine them and this constitutes a TT-representation of $\mathbf{A}$.

Now we consider the case when number of external indices in the tensor is 1. If the external index k is the last index then we want to compute the last core of the TT-representation. Hence we simply assign the current tensor to the last core and this is returned to the caller. When the external index corresponds to the first or last index of the tensor, we set α_0 or α_d appropriately for the core. For the correctness, now the input tensor is multiplied with the diagonal matrix S_k, and the result is assigned to the kth core of the TT-representation. In the end, this core is returned to the caller.

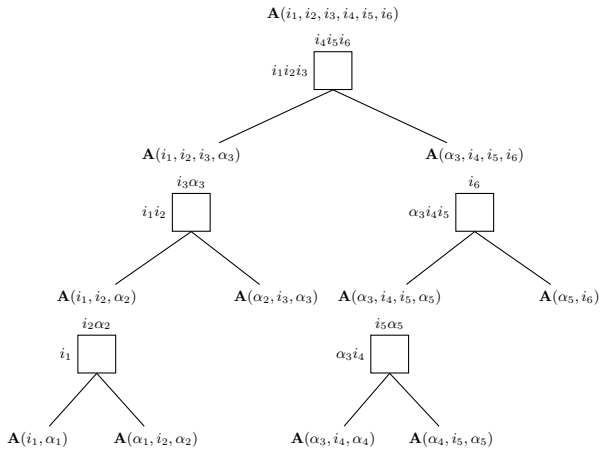


Fig. 4: Diagramatic representation of unfolding matrices at all non-leaf nodes for a 6-dimensional tensor by Algorithm 1.

PTT-decomposition algorithm returns the cores of a TT presentation for $\mathbf{A}$. This algorithm exposes parallelism in a binary tree shape. It achieves maximum $\mathcal{O}(d)$-level of parallelism at the last level of the tree.

Figure 4 illustrates the working tree of Algorithm 1 for a 6-dimensional tensor. It shows a tensor at each node in its index form. It also displays unfolding matrices for non-leaf nodes. Cores of the TT-representation are assigned at leaf nodes after multiplying the subtensors with corresponding S_k matrices (Lines 18 and 20 of Algorithm 1).

We can notice that working tree of our algorithm is similar to Hierarchical Tucker [12]. However, intrinsic details of both algorithms are quite different. In our algorithm, cores of the TT-representation are assigned at leaf nodes. While in Hierarchical Tucker algorithms, compressed representation is stored in a tree format. Leaf nodes correspond to factor matrices of Tucker decomposition and

each internal node stores a 3-dimensional tensor or a matrix. All internal nodes combined together correspond to the core tensor of Tucker decomposition.

The proof on the bound of TT ranks obtained by Algorithm 1 is described in the following theorem.

Theorem 1. *If for each unfolding A_k of a d-dimensional tensor $\boldsymbol{A}$, $rank(A_k) = r_k$, then Algorithm 1 produces a TT-representation with ranks not higher than r_k.*

Proof. Let us consider the unfolding matrix A_k of a d-dimensional tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$. The SVD of A_k is represented as, $A_k = U \Sigma V^T$. The rank of the unfolding matrix is r_k; hence it can be written as:

$$\begin{aligned}
&A_k(i_1, \cdots, i_k; i_{k+1}, \cdots, i_d) \\
&\quad = \sum_{\alpha=1}^{r_k} U(i_1, \cdots, i_k; \alpha) \Sigma(\alpha; \alpha) V^T(\alpha; i_{k+1}, \cdots, i_d) \\
&\quad = \sum_{\alpha=1}^{r_k} U(i_1, \cdots, i_k; \alpha) X(\alpha; \alpha) S(\alpha; \alpha) Y(\alpha; \alpha) V^T(\alpha; i_{k+1}, \cdots, i_d) \\
&\quad = \sum_{\alpha=1}^{r_k} B(i_1, \cdots, i_k; \alpha) S(\alpha; \alpha) C(\alpha; i_{k+1}, \cdots, i_d).
\end{aligned}$$

In matrix form we obtain,

$$\begin{aligned}
A_k &= BSC, \\
B &= A_k C^{-1} S^{-1} = A_k Z, \\
C &= S^{-1} B^{-1} A_k = W A_k,
\end{aligned}$$

or in the index form,

$$\begin{aligned}
B(i_1, \cdots, i_k; \alpha) &= \sum_{i_{k+1}=1}^{n_{k+1}} \cdots \sum_{i_d=1}^{n_d} \mathbf{A}(i_1, \cdots, i_d) Z(i_{k+1}, \cdots, i_d; \alpha), \\
C(\alpha; i_{k+1}, \cdots, i_d) &= \sum_{i_1=1}^{n_1} \cdots \sum_{i_k=k}^{n_k} \mathbf{A}(i_1, \cdots, i_d) W(\alpha; i_1, i_2, \cdots, i_k).
\end{aligned}$$

B and C can be treated as $k+1$ and d-$k+1$ dimensional tensors $\mathbf{B}$ and $\mathbf{C}$ respectively. Now we consider unfolding matrices $B_1, \cdots, B_{k-1}$ and $C_{k+1}, \cdots, C_{d-1}$ of $\mathbf{B}$ and $\mathbf{C}$. We will show that $\text{rank}(B_{k'}) \leq r_{k'}$ $_{1 \leq k' \leq k-1}$ and $\text{rank}(C_{k'}) \leq r_{k'}$ $_{k+1 \leq k' \leq d-1}$.
The rank of $A_{k'}$ is $r_{k'}$. Therefore, $\mathbf{A}$ can be represented as,

$$\mathbf{A}(i_1, \cdots, i_d) = \sum_{\beta=1}^{r_{k'}} F(i_1, \cdots, i_{k'}; \beta) G(\beta; i_{k'+1}, \cdots, i_d)$$

Now,

$$B_{k'} = [B_{k'}(i_1, \cdots, i_{k'}; i_{k'+1}, \cdots, i_k, \alpha)]$$

$$\begin{aligned}
&B_{k'}(i_1, \cdots, i_{k'}; i_{k'+1}, \cdots, i_k, \alpha) \\
&\quad = \sum_{i_{k+1}=1}^{n_{k+1}} \cdots \sum_{i_d=1}^{n_d} \mathbf{A}(i_1, \cdots, i_d) Z(i_{k+1}, \cdots, i_d; \alpha) \\
&\quad = \sum_{\beta=1}^{r_{k'}} \sum_{i_{k+1}=1}^{n_{k+1}} \cdots \sum_{i_d=1}^{n_d} F(i_1, \cdots, i_{k'}; \beta) G(\beta; i_{k'+1}, \cdots, i_d) Z(i_{k+1}, \cdots, i_d; \alpha) \\
&\quad = \sum_{\beta=1}^{r_{k'}} F(i_1, i_2, \cdots, i_{k'}; \beta) H(\beta; i_{k'+1}, \cdots, i_k, \alpha).
\end{aligned}$$

where,

$$\begin{aligned}
&H(\beta; i_{k'+1}, \cdots, i_k, \alpha) \\
&\quad = \sum_{i_{k+1}=1}^{n_{k+1}} \cdots \sum_{i_d=1}^{n_d} G(\beta; i_{k'+1}, \cdots, i_d) Z(i_{k+1}, \cdots, i_d; \alpha).
\end{aligned}$$

Row and column indices of $B_{k'}$ are now separated. Hence rank$(B_{k'}) \leq r_{k'}$.

Similarly for $C_{k'}$,
$C_{k'} = [C_{k'}(\alpha, i_{k+1}, \cdots, i_{k'}; i_{k'+1}, \cdots, i_d)]$

$$\begin{aligned}
&C_{k'}(\alpha, i_{k+1}, \cdots, i_{k'}; i_{k'+1}, \cdots, i_d) \\
&\quad = \sum_{i_1=1}^{n_1} \cdots \sum_{i_k=k}^{n_k} \mathbf{A}(i_1, i_2, \cdots, i_d) W(\alpha; i_1, i_2, \cdots, i_k) \\
&\quad = \sum_{\beta=1}^{r_{k'}} \sum_{i_1=1}^{n_1} \cdots \sum_{i_k=k}^{n_k} F(i_1, i_2, \cdots, i_{k'}; \beta) G(\beta; i_{k'+1}, \cdots, i_d) W(\alpha; i_1, i_2, \cdots, i_k) \\
&\quad = \sum_{\beta=1}^{r_{k'}} M(\alpha, i_{k+1}, \cdots, i_{k'}; \beta) G(\beta; i_{k'+1}, \cdots, i_d).
\end{aligned}$$

where,

$$\begin{aligned}
&M(\alpha, i_{k+1}, \cdots, i_{k'}; \beta) \\
&\quad = \sum_{i_1=1}^{n_1} \cdots \sum_{i_k=k}^{n_k} F(i_1, i_2, \cdots, i_{k'}; \beta) W(\alpha; i_1, i_2, \cdots, i_k).
\end{aligned}$$

Here also row and column indices of $C_{k'}$ are separated. Hence rank$(C_{k'}) \leq r_{k'}$. As the above proof holds for each recursive partition, hence ranks of the TT-representation produced by Algorithm 1 are bounded by r_k. This completes the proof.

The above proof is independent of from which index we split the tensor into two subtensors. However in Algorithm 1, we split the tensor from the middle external index.

5 Approximations of the Parallel Tensor Train Decomposition

Tensor decompositions are often approximated to cope with large amounts of data while controlling the loss of accuracy. In practical computations, rank-r_k approximation of a tensor or an approximated tensor with certain accuracy is desired. Algorithm 1 can be used to compute rank-r_k approximation of a tensor instead of exact low-rank decomposition. However we are yet to claim any optimality on its approximation and it is a part of our future work.

We modify Algorithm 1 to compute an approximated tensor in TT-representation that is less than or close to the prescribed accuracy, and present it in Algorithm 2.

Algorithm 2 PTT-approx (parallel Tensor Train approximation)

Require: d-dimensional tensor $\mathbf{A}$ and expected accuracy ϵ
Ensure: Cores $\mathbf{G}_k(\alpha_{k-1}, n_k, \alpha_k)_{1\leq k\leq d}$ of the approximated tensor $\mathbf{B}$ in TT-representation such that $||\mathbf{A}-\mathbf{B}||_F$ is close to or less than ϵ

if $nEI(\mathbf{A})$ >1 **then**
  Find the middle external index k
  Compute unfolding matrix A_k
  Compute SVD: $A_k = U\Sigma V^T$
  Compute truncation accuracy Δ
  Compute α_k such that $A_k = U(;1:\alpha_k)\Sigma(1:\alpha_k;1:\alpha_k)V^T(1:\alpha_k;) + E_k$ and $||E_K||_F \leq \Delta$
  Select diagonal matrices X_k, S_k and Y_k such that $X_kS_kY_k = \Sigma(1:\alpha_k;1:\alpha_k)$
  $\mathbf{A}_{left} = \mathbf{Tensor}(U(;1:\alpha_k)X_k)$
  list1 = PTT-approx($\mathbf{A}_{left}$, ϵ_1)
  $\mathbf{A}_{right} = \mathbf{Tensor}(Y_kV^T(1:\alpha_k;))$
  list2 = PTT-approx($\mathbf{A}_{right}$, ϵ_2)
  return {list1, list2}
else
  Find the external index k
  if k is the last index of $\mathbf{A}$ **then**
    $\alpha_k = 1$
  else if k is the first index of $\mathbf{A}$ **then**
    $\alpha_{k\text{-}1} = 1$
    $\mathbf{A}(i_k,\beta) = \sum_{\beta=1}^{\alpha_k}\mathbf{A}(i_k,\beta)S_k(\beta;\beta)$
  else
    $\mathbf{A}(\gamma,i_k,\beta) = \sum_{\beta=1}^{\alpha_k}\mathbf{A}(\gamma,i_k,\beta)S_k(\beta;\beta)$
  end if
  $\mathbf{G}_k = \mathbf{A}$
  return $\mathbf{G}_k$
end if

Expected accuracy is also an input to Algorithm 2. α_k is selected based on truncation accuracy Δ of the unfolding matrix. Approximations of left and right subtensors are called with expected accuracies of ϵ_1 and ϵ_2. Values of δ, ϵ_1 and ϵ_2

in Algorithm 2 depend on the truncation accuracy of the unfolding matrix and how leading singular values are passed to both subtensors (depend on selection of diagonal matrices X_k, Y_k and S_K). We show in next subsection how the product of approximated matrices impacts the accuracy of the result when applied with SVD truncation. Based on the expression of the next subsection, we compute ϵ_1 and ϵ_2 for different choices of X_k, Y_k and S_K in Section 5.2.

5.1 Frobenius Error with Product of Approximated Matrices

The SVD of a real matrix A is written as, $A = U\Sigma V^T$. This decomposition can also be written as,

$$A = (U_1 U_2)\begin{pmatrix}\Sigma_1 & 0\\ 0 & \Sigma_2\end{pmatrix}(V_1 V_2)^T = U_1\Sigma_1 V_1^T + U_2\Sigma_2 V_2^T$$
$$= U_1\Sigma_1 V_1^T + E_A = BSC + E_A.$$

Here $B = U_1 X$, $C = Y V_1^T$ and $XSY = \Sigma_1$. Matrices B and C are approximated by $\hat{B}$ and $\hat{C}$, i.e., $B = \hat{B} + E_B$ and $C = \hat{C} + E_C$. X, Y and S are diagonal matrices. E_A, E_B and E_C represent error matrices corresponding to low-rank approximations of A, B and C. We are interested to find an expression for $||A - \hat{B}S\hat{C}||_F$ in terms of $||E_A||_F$, $||E_B||_F$ and $||E_C||_F$. We have,

$$\begin{aligned} A - \hat{B}S\hat{C} &= A - (B - E_B)S(C - E_C)\\ &= A - BSC + BSE_C + E_BSC - E_BSE_C\\ &= E_A + BSE_C + E_BSC - E_BSE_C \end{aligned}$$

Now we take square of Frobenius norm on both sides,

$$\begin{aligned} &||A - \hat{B}S\hat{C}||_F^2\\ &\quad= ||E_A + BSE_C + E_BSC - E_BSE_C||_F^2\\ &\quad= ||E_A||_F^2 + ||BSE_C||_F^2 + ||E_BSC||_F^2 + ||E_BSE_C||_F^2\\ &\qquad + 2 < E_A, BSE_C >_F + 2 < E_A, E_BSC >_F\\ &\qquad - 2 < E_A, E_BSE_C >_F + 2 < BSE_C, E_BSC >_F\\ &\qquad - 2 < BSE_C, E_BSE_C >_F - 2 < E_BSC, E_BSE_C >_F . \end{aligned}$$

Here $< P, Q >_F$ denotes frobenius inner product of matrices P and Q, and it is defined as: $< P, Q >_F = tr(P^T Q) = tr(PQ^T)$. As U and V are othogonal matrices obtained from the SVD of A, we obtain the following expressions:

$$U_1^T U_2 = 0 \implies B^T E_A = 0 \implies < E_A, BSE_C >_F = 0,$$
$$V_1^T V_2 = 0 \implies CE_A^T = 0 \implies < E_A, E_BSC >_F = 0.$$

We thus obtain,

$$
\begin{aligned}
&||A - \hat{B}S\hat{C}||_F^2 \\
&\quad = ||E_A||_F^2 + ||BSE_C||_F^2 + ||E_BSC||_F^2 + ||E_BSE_C||_F^2 \\
&\qquad - 2 < E_A, E_BSE_C >_F + 2 < BSE_C, E_BSC >_F \\
&\qquad - 2 < BSE_C, E_BSE_C >_F - 2 < E_BSC, E_BSE_C >_F .
\end{aligned}
$$

In general, target error of any approximation is very low, therefore we assume that any term involving more than one error matrix would be close to zero. With this assumption, the above equation can be written as,

$$||A - \hat{B}S\hat{C}||_F^2 \approx ||E_A||_F^2 + ||BSE_C||_F^2 + ||E_BSC||_F^2$$

5.2 Different Approaches

Here we propose different approaches based on how leading singular values of the unfolding matrix are passed to the left and right subtensors in Algorithm 2. It is immediate that the expression of the previous subsection can be applied to compute ϵ_1 and ϵ_2 for each unfolding matrix of Algorithm 2. A of the previous subsection corresponds to A_k of Algorithm 2. Similarly, B and C correspond to the unfolding matrices of left and right subtensors. If the number of external dimensions in left and right subtensors are d_1 and d_2, then $d_1 + d_2 = d$. For simplicity, we assume $\frac{||E_B||_F^2}{d_1-1} = \frac{||E_C||_F^2}{d_2-1} = \delta^2$. This implies $\frac{\epsilon_1^2}{d_1-1} = \frac{\epsilon_2^2}{d_2-1} = \delta^2$. Similar to the approach of [26], we take $||E_A||_F = \Delta \leq \frac{\epsilon}{\sqrt{d-1}}$. Let Σ_α denote $\Sigma(1:\alpha_k; 1:\alpha_k)$ of Algorithm 2. X, S, Y, U_1 and V_1^T of the previous subsection correspond to X_k, S_k, Y_k, $U(;1:\alpha_k)$ and $V^T(1:\alpha_k;)$ of Algorithm 2.
Now we propose three approaches such that the approximation error of Algorithm 2 is less than or close to ϵ.

1. *Leading Singular values to Right subtensor* (*LSR*): We transmit leading singular values of the unfolding matrix to the right subtensor in this approach. It is equivalent to selecting $X = I$, $Y = \Sigma_\alpha$ and $S = I$.

$$
\begin{aligned}
&||A - \hat{B}S\hat{C}||_F^2 \\
&\qquad \approx ||E_A||_F^2 + ||U_1E_C||_F^2 + ||E_B\Sigma_\alpha V_1^T||_F^2 \\
&\qquad \leq \frac{\epsilon^2}{d-1} + ||E_C||_F^2 + ||E_B\Sigma_\alpha||_F^2 \\
&\qquad \leq \frac{\epsilon^2}{d-1} + ||E_C||_F^2 + ||E_B||_F^2||\Sigma_\alpha||_2^2 \\
&\qquad = \frac{\epsilon^2}{d-1} + \delta^2(d_2-1) + \delta^2(d_1-1)||\Sigma_\alpha||_2^2 \leq \epsilon^2.
\end{aligned}
$$

After simplifying this expression we obtain, $\delta \leq \epsilon\sqrt{\frac{d-2}{(d-1)(d_2-1+(d_1-1)||\Sigma_\alpha||_2^2)}}$.

2. *Square root of Leading Singular values to Both subtensors* (*SLSB*): We transmit square root of leading singular values to the both subtensors in this approach. It is equivalent to choosing $X = Y = \Sigma_\alpha^{\frac{1}{2}}$ and $S = I$.

$$
\begin{aligned}
&||A - \hat{B}S\hat{C}||_F^2 \\
&\quad \approx ||E_A||_F^2 + ||U_1\Sigma_\alpha^{\frac{1}{2}}E_C||_F^2 + ||E_B\Sigma_\alpha^{\frac{1}{2}}V_1^T||_F^2 \\
&\quad \leq \frac{\epsilon^2}{d-1} + ||\Sigma_\alpha^{\frac{1}{2}}E_C||_F^2 + ||E_B\Sigma_\alpha^{\frac{1}{2}}||_F^2 \\
&\quad \leq \frac{\epsilon^2}{d-1} + ||E_C||_F^2||\Sigma_\alpha^{\frac{1}{2}}||_F^2 + ||E_B||_F^2||\Sigma_\alpha^{\frac{1}{2}}||_F^2 \\
&\quad = \frac{\epsilon^2}{d-1} + \delta^2(d_2 - 1 + d_1 - 1)tr(\Sigma_\alpha) \leq \epsilon^2.
\end{aligned}
$$

After simplifying this expression we obtain, $\delta \leq \frac{\epsilon}{\sqrt{(d-1)tr(\Sigma_\alpha)}}$.

3. *Leading Singular values to Both subtensors* (*LSB*): We transmit leading singular values to the both subtensors in this approach. It is equivalent to selecting $X = Y = \Sigma_\alpha$ and $S = \Sigma_\alpha^{-1}$.

$$
\begin{aligned}
&||A - \hat{B}S\hat{C}||_F^2 \\
&\quad \approx ||E_A||_F^2 + ||U_1\Sigma_\alpha\Sigma_\alpha^{-1}E_C||_F^2 + ||E_B\Sigma_\alpha^{-1}\Sigma_\alpha V_1^T||_F^2 \\
&\quad \leq \frac{\epsilon^2}{d-1} + ||E_C||_F^2 + ||E_B||_F^2 \\
&\quad = \frac{\epsilon^2}{d-1} + \delta^2(d_2 - 1) + \delta^2(d_1 - 1) \leq \epsilon^2.
\end{aligned}
$$

After simplifying this expression we get, $\delta \leq \frac{\epsilon}{\sqrt{d-1}}$.
It is not hard to observe that this approach uses a uniform truncation accuracy of $\frac{\epsilon}{\sqrt{d-1}}$ for each unfolding matrix.

5.3 Additional Algorithm

We also consider the approximation algorithm proposed in [26] for evaluation. This can be considered as a special case of Algorithm 2, where $X = S = I$, $Y = \Sigma_\alpha$, and k is the first left external index. In each step, left subtensor only has one external index and is assigned to one of the cores of TT-representation. We recall that this is a sequential algorithm. A uniform truncation of $\frac{\epsilon}{\sqrt{d-1}}$ is applied for each unfolding, which is equivalent to selecting $\epsilon_1 = 0$ and $\epsilon_2 = \epsilon\sqrt{\frac{d_2-1}{d-1}}$. Approximation error of this algorithm is bounded by a function of truncation applied at each step. We call this approach *Sequential Tensor Train Approximation* (*STTA*).

Table 1 summarizes all parameters of different approaches for Algorithm 2. Let d_1 and d_2 denote the number of external dimensions for the left and right subtensors.

Approach	Description	Δ	ϵ_1	ϵ_2
LSR	$X = I,\ Y = \Sigma_\alpha,\ S = I$	$\frac{\epsilon}{\sqrt{d-1}}$	$\epsilon\sqrt{\frac{(d-2)(d_1-1)}{(d-1)(d_2-1+(d_1-1)tr(\Sigma_\alpha^2))}}$	$\epsilon\sqrt{\frac{(d-2)(d_2-1)}{(d-1)(d_2-1+(d_1-1)tr(\Sigma_\alpha^2))}}$
SLSB	$X = Y = \Sigma_\alpha^{\frac{1}{2}},\ S = I$	$\frac{\epsilon}{\sqrt{d-1}}$	$\epsilon\sqrt{\frac{d_1-1}{(d-1)tr(\Sigma_\alpha)}}$	$\epsilon\sqrt{\frac{d_2-1}{(d-1)tr(\Sigma_\alpha)}}$
LSB	$X = Y = \Sigma_\alpha,\ S = \Sigma_\alpha^{-1}$	$\frac{\epsilon}{\sqrt{d-1}}$	$\epsilon\sqrt{\frac{d_1-1}{d-1}}$	$\epsilon\sqrt{\frac{d_2-1}{d-1}}$
STTA	$X = I,\ Y = \Sigma_\alpha,\ S = I$	$\frac{\epsilon}{\sqrt{d-1}}$	0	$\epsilon\sqrt{\frac{d_2-1}{d-1}}$

Table 1: Summary of all considered approaches.

5.4 Costs of *STTA* and PTT-approx

Let us assume that the ranks of the TT-representations produced by both algorithms, *STTA* and PTT-approx, for a d dimensional tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ are same, and $r_1 = r_2 = \cdots = r_{d\text{-}1} = r$. We also assume that $n_1 = n_2 = \cdots = n_d = n$. For simplicity, we assume that d is perfect power of 2.

The most accurate low rank approximations rely on truncated SVD, but SVD is expensive. There are good alternatives like QR factorization with column pivoting (QRCP) or randomized SVD, whose computational complexity is much less compared to SVD. It is now established that those alternatives provide results very close to the ones of the truncated SVD with guaranteed bounds, see e.g. [9]. Some communication optimal implementations of QRCP exist in the community [5], hence we plan to use QRCP before SVD for our distributed memory implementations. The full QRCP decomposition of a matrix $A \in \mathbb{R}^{M \times N}$ can be represented as, $A\Pi = QR = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix}$, where Π is a column permutation matrix, $Q \in \mathbb{R}^{M \times M}$, $Q_1 \in \mathbb{R}^{M \times k}$, and $Q_2 \in \mathbb{R}^{M \times M-k}$ are orthogonal matrices, $R_{11} \in \mathbb{R}^{k \times k}$ is an upper triangular matrix, $R_{12} \in \mathbb{R}^{k \times N\text{-}k}$, and $R_{22} \in \mathbb{R}^{M\text{-}k \times N\text{-}k}$. k-rank approximation of A is obtained as, $\tilde{A} = Q_1 Q_1^T A = Q_1 \begin{pmatrix} R_{11} & R_{12} \end{pmatrix} \Pi^T$. Usually k is very small, hence $B = Q_1^T A$ is a short and fat matrix. To compute SVD of B, TSQR factorization of B^T, $B^T = Q_T R_T$, can be employed first and then SVD of a small $k \times k$ matrix R_T^T, $R_T^T = U_T \Sigma V_T^T$, is computed. Approximation $\tilde{A}$ can be expressed as, $\tilde{A} = U \Sigma V^T$, where $U = Q_1 U_T$ and $V = Q_T V_T$.

Performance counts of PTT-approx algorithm: We consider that the input d-dimensional tensor is distributed over a grid of $P = P_1 \times P_2 \times \cdots P_d$ processors. Each processor has a small subtensor of size $\frac{n}{P_1} \times \frac{n}{P_2} \times \cdots \frac{n}{P_d}$. We assume $P_1 = P_2 = \cdots P_d$. For computations, we assume processors are arranged in a grid of $P_r \times P_c$, where $P_r = P_c = \sqrt{P}$. Each row (or column) block of the processors has the same set of row (or column) indices of the unfolding matrix. Order of rows and columns of the unfolding matrix of the subtensor on each processor is different from the order of rows and columns in the original unfolding matrix of the tensor. In order to obtain correct results for each node in PTT tree, the final results must be multiplied with the appropriate row and column permutation matrices. We can notice that at the l-th level, total number of nodes in the tree is 2^{l-1}. Only $\frac{P}{2^{l-1}}$ number of processors participate to perform all computations of a node. Now we discuss the k-rank approximation cost of a l-th

level node with unfolding matrix $A \in \mathbb{R}^{M \times N}$ on $P' = \frac{P}{2^{l-1}}$ number of processors in PTT tree.

QRCP critical path cost The unfolding matrix is equally distributed on the given number of processors (based on our assumption in the beginning or distribution of data in the previous step). Total costs along the critical path to perform QRCP in parallel are the following [5]:

- Number of computations $= \mathcal{O}\big(\frac{MNk}{P'}\big)$
- Number of messages $= \mathcal{O}(\log_2^2 P')$
- Volume of communications $= \mathcal{O}(\frac{Mk}{\sqrt{P'}} \log_2 P')$

After this step, Q_1 matrix is distributed along the first column block of processors and $(R_{11}\ R_{12})$ matrix is distributed along the first row block of processors.

Multiplying permutation matrix, $R' = \big(R_{11}\ R_{12}\big)\Pi^T$

- Number of messages $= \mathcal{O}(\log_2 P')$
- Volume of communications $= \mathcal{O}(k^2 \log_2 P')$

Here we can note that, the same operation can be performed by $\mathcal{O}(k)$ number of messages and $\mathcal{O}(k^2)$ amount of communications. However, we are more interested to obtain an expression for the number of messages which is independent of k.

Redistribution of R' *to all available processors*

- Number of messages $= \mathcal{O}(\log_2 P')$ (scatter operation)
- Volume of communications $= \mathcal{O}(\frac{Nk}{\sqrt{P'}})$

TSQR critical path cost

- Number of computations $= \mathcal{O}(\frac{Nk^2}{P'})$
- Number of messages $= \mathcal{O}(\log_2 P')$
- Volume of communications $= \mathcal{O}(k^2 \log_2 P')$

After this step, a small $k \times k$ matrix is broadcasted to all processors.

Local SVD computation

- Number of computations $= \mathcal{O}(k^3)$

After this step, both orthogonal matrices are updated with the results of local SVD computations. Before advancing to the next level, the outputs need to be redistributed for two separate calls.

Redistribution of outputs for two new calls of the next level

- Number of messages $= \mathcal{O}(\log_2 P')$ (scatter data to $\frac{\sqrt{P'}}{2}$ number of processors)
- Volume of communications $= \mathcal{O}(\frac{\max(M,N)k}{\sqrt{P'}})$

After dropping the lower order terms, total computation costs along the critical path to compute k-rank approximation of a node with unfolding matrix $A \in \mathbb{R}^{M \times N}$ on P' number of processors is $\mathcal{O}(\frac{MNk}{P'})$. Similarly, total number of messages and volume of communications along the critical path are $\mathcal{O}(\log_2^2 P')$ and $\mathcal{O}(\frac{Mk}{\sqrt{P'}} \log_2 P')$ respectively.

As PTT tree has $\mathcal{O}(\log_2 d)$ nodes in the critical path, therefore total number of messages transmitted in this path is $\mathcal{O}(\log_2^2 P \log_2 d)$. The lower bound on the number of messages for any algorithm which exchanges data on P processors is $\Omega(\log_2 P)$, and PTT-approx achieves this bound, modulo polylog factor. For the first node of the PTT tree, $M = N = n^{\frac{d}{2}}$, thus total number of computations and amount of communications for this node are atleast $\mathcal{O}(n^{\frac{d}{4}})$ times more than the other nodes. After dropping the lower order terms, total amount of computations and communications along the critical path in PTT-approx are $\mathcal{O}(\frac{rn^d}{P})$ and $\mathcal{O}(\frac{rn^{\frac{d}{2}}}{P} \log_2 P)$ respectively. In our analysis, we did not consider the cost of multiplying with the global row and column permutation matrices because it is possible to directly work with the part of unfolding matrices instead of subtensors.

Performance counts for the sequential case: Here we discuss the k-rank approximation cost of an unfolding matrix $A \in \mathbb{R}^{M \times N}$ on one processor. The QRCP complexity to obtain Q_1 and $Q_1^T A$ is $\mathcal{O}(MNk)$. Complexity to obtain k singular values, left and right singular vectors of a matrix $A \in \mathbb{R}^{k \times N}$ by employing TSQR and SVD is $\mathcal{O}(Nk^2 + k^3)$. After dropping some of the lower order terms, the overall complexity of applying QRCP and SVD to obtain k-rank approximation in our approach is $\mathcal{O}(MNk + Nk^2)$.

STTA and PTT-approx both algorithms employ SVD d-1 times for a d dimensional tensor. With applying QRCP before SVD, computation costs of *STTA* and PTT-approx are $\mathcal{O}(rn^d + r^2(n^{d\text{-}1} + n^{d\text{-}2} + \cdots + n^2) + r^2(n^{d\text{-}1} + n^{d\text{-}2} + \cdots + n))$ and $\mathcal{O}(rn^d + 2r^2(n^{\frac{d}{2}} + \cdots + n^2) + r^3((2^{3\text{-}1}\text{-}2)n^{\frac{d}{4}} + \cdots + (2^{lg(d)\text{-}1}\text{-}2)n^2) + r^3((2^{2\text{-}1}\text{-}1)n^{\frac{d}{4}} + \cdots + (2^{lg(d)\text{-}1}\text{-}1)n) + r^2(n^{\frac{d}{2}} + \cdots + n))$ respectively. Computation cost of the first node for both *STTA* and PTT-approx algorithms are asymptotically same and equal to $\mathcal{O}(rn^d)$. After the first node, total number of computations in PTT-approx is $\mathcal{O}(n^{\frac{d}{2}\text{-}1})$ times less compared to *STTA*.

In this article, we show effectiveness and stability of our algorithms only with sequential SVD. Accuracy results with applying QRCP before SVD are presented in Appendix A. Distributed memory implementations of *STTA* and PTT-approx are currently underway and out of scope of this paper.

6 Experimental Setup & Results

6.1 Experimental Setup

We show effectiveness and stability of our methods using Matlab. We consider the tensors produced by the low rank functions of Table 2 for our evaluation. Such low rank functions arise in numerous applications and we refer to [1,6,13] for more details about these functions.

Log	$\log(\sum_{j=1}^{N} j i_j)$
Sin	$\sin(\sum_{j=1}^{N} i_j)$
Inverse-Square-Root (ISR)	$\frac{1}{\sqrt{\sum_{j=1}^{N} i_j^2}}$
Inverse-Cube-Root (ICR)	$\frac{1}{\sqrt[3]{\sum_{j=1}^{N} i_j^3}}$
Inverse-Penta-Root (IPR)	$\frac{1}{\sqrt[5]{\sum_{j=1}^{N} i_j^5}}$

Table 2: Low rank functions.

6.2 Performance Metrics

Let NE denote the number of entries in the original tensor. ne_{Ap} and ne_{min} represent the number of entries in the TT-representation produced by approach Ap and the minimum number of entries in TT-representation among all the considered approaches. We consider the following performance metrics to assess approach Ap for tensor $\mathbf{A}$.

1. compression ratio ($compr$): This is defined as, $compr_{Ap}(\mathbf{A}) = \frac{NE(\mathbf{A}) - ne_{Ap}(\mathbf{A})}{NE(\mathbf{A})} * 100$ (higher values are better). A high value for this ratio indicates that the TT-representation produced by the approach has a few number of entries.
2. ratio to minimal (r): This is defined as, $r_{Ap}(\mathbf{A}) = \frac{ne_{Ap}(\mathbf{A})}{ne_{min}(\mathbf{A})}$ (lower values are better). This ratio is at least 1 and a value close to 1 indicates a well-suited approach which achieves good compression for the given tensor.

We also observe accuracy (OA) of the approximation. If the TT-representation produced by Ap corresponds to tensor $\mathbf{B}$, then $OA_{Ap}(\mathbf{A}) = ||\mathbf{A} - \mathbf{B}||_F$. We present OA values in E notation (me-n indicates a value of $m \times 10^{-n}$).

6.3 Experimental Evaluation

First we consider $N = 12$ and $i_j \in \{1, 2, 3, 4\}_{1 \leq j \leq N}$ for the functions of Table 2. This configuration produces a 12-dimensional tensor with 4^{12} elements for each low rank function. We perform comparison of all the considered approaches for the target accuracies of 10^{-3} and 10^{-6} in Table 3.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
STTA	compr	99.996	99.999	99.994	99.991	99.986
	ne	596	176	992	1580	2336
	OA	3.692e-04	2.615e-09	3.224e-04	4.233e-04	4.774e-04
LSR	compr	99.974	99.999	99.979	99.964	99.950
	ne	4344	176	3444	6076	8384
	OA	7.497e-05	1.393e-11	2.037e-04	6.102e-05	2.985e-04
SLSB	compr	99.989	99.999	99.986	99.977	99.968
	ne	1832	176	2352	3840	5404
	OA	7.042e-05	6.144e-12	2.036e-04	6.097e-05	2.981e-04
LSB	compr	99.996	99.999	99.994	99.991	99.986
	ne	596	176	992	1528	2336
	OA	3.692e-04	1.252e-11	3.223e-04	5.042e-04	4.773e-04

(a) Prescribed accuracy = 10^{-3}.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
STTA	compr	99.993	99.999	99.987	99.981	99.971
	ne	1212	176	2240	3184	4864
	OA	2.271e-07	2.615e-09	1.834e-07	4.884e-07	4.836e-07
LSR	compr	99.817	99.998	99.915	99.874	99.824
	ne	30632	344	14196	21176	29524
	OA	3.629e-08	1.412e-11	1.118e-07	8.520e-08	5.811e-08
SLSB	compr	99.799	99.999	99.952	99.912	99.870
	ne	33772	176	8068	14824	21792
	OA	2.820e-08	6.144e-12	1.118e-07	8.518e-08	5.664e-08
LSB	compr	99.993	99.999	99.987	99.981	99.970
	ne	1212	176	2240	3184	4964
	OA	2.265e-07	1.252e-11	1.834e-07	4.884e-07	3.999e-07

(b) Prescribed accuracy = 10^{-6}.

Table 3: Comparison of different approaches for 12-dimensional tensors.

Table 3a shows comparable compression ratios for all the approaches. *SLSB* achieves slightly better compression ratios than *LSR*. *LSB* compression ratios are the best among all our approaches and are also almost equal or better than *STTA*. We can also observe that *ne* values of ICR function are more than ISR function and less than IPR function. This indicates that the low-rank approximation of function $(\sum_j i_j^p)^{\frac{1}{p}}$ requires more entries as p increases. OA values of all the approaches are also within the prescribed accuracy.

Table 3b also shows similar behavior. *LSB* compression ratios are the best among all our approaches and are similar to *STTA*. OA values of all the approaches are also within the prescribed limit of 10^{-6}.

Next we take $N = 6$ and $i_j \in \{1, 2 \cdots, 16\}_{1 \leq j \leq N}$ for the functions of Table 2. This configuration produces a 6-dimensional tensor with 16^6 elements for each function. We present our performance metrics of all the considered approaches for these tensors in Table 4.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
STTA	compr	99.987	99.998	99.976	99.962	99.928
	ne	2112	320	4064	6368	12032
	OA	2.831e-04	4.206e-10	4.587e-04	6.149e-04	4.429e-04
LSR	compr	99.846	99.998	99.933	99.902	99.840
	ne	25760	320	11264	16384	26880
	OA	7.304e-05	3.410e-12	2.033e-04	2.882e-04	2.221e-04
SLSB	compr	99.973	99.998	99.952	99.928	99.879
	ne	4512	320	8096	12016	20288
	OA	7.268e-05	3.486e-12	2.033e-04	2.882e-04	2.218e-04
LSB	compr	99.987	99.998	99.976	99.962	99.928
	ne	2112	320	4064	6368	12032
	OA	2.832e-04	4.409e-12	4.588e-04	6.150e-04	4.430e-04

(a) Prescribed accuracy = 10^{-3}.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
STTA	compr	99.974	99.998	99.945	99.915	99.845
	ne	4320	320	9184	14336	26048
	OA	4.156e-07	4.206e-10	3.990e-07	6.983e-07	4.536e-07
LSR	compr	99.644	99.998	99.703	99.570	99.442
	ne	59696	320	49808	72128	93536
	OA	3.584e-07	3.410e-12	2.822e-07	1.523e-07	1.804e-07
SLSB	compr	99.824	99.998	99.902	99.854	99.772
	ne	29456	320	16416	24576	38208
	OA	3.584e-07	3.486e-12	2.822e-07	1.522e-07	1.802e-07
LSB	compr	99.974	99.998	99.945	99.915	99.845
	ne	4320	320	9184	14336	26048
	OA	4.156e-07	4.409e-12	3.990e-07	6.985e-07	4.538e-07

(b) Prescribed accuracy = 10^{-6}.

Table 4: Comparison of different approaches for 6-dimensional tensors.

Table 4 displays comparable compression ratios for all the approaches. Again *LSB* compression ratios are the best among all our approaches and are same as *STTA*. *OA* values of all the approaches are also within the prescribed accuracy. Decay of singular values for *Sin* function is quite steep, hence it achieves the best compression ratios among all the low rank functions of Table 2.

Figure 5 depicts r values of all the considered approaches for 12-dimensional tensors. All the approaches achieve the similar compression ratio for *Sin* function, hence their r values are close to 1. r_{LSB} and r_{STTA} values are almost same and are equal to 1. For the prescribed accuracy of 10^{-3}, r_{LSR} values are between 3 to 7.5 for the other functions. *SLSB* compression ratios are slightly better than *LSR*, hence r_{SLSB} values are smaller than r_{LSR}, but still in the range of 2.25 to 3.25. For the prescribed accuracy of 10^{-6}, r_{LSR} and r_{SLSB} values for *Log* function are more than 25. This indicates that *LSR* and *SLSB* store 25× more

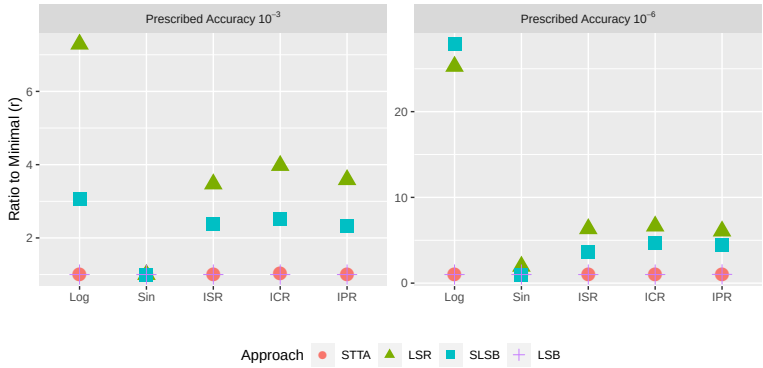


Fig. 5: Ratio to minimal comparison for 12-dimensional tensors.

entries than the best approach. r_{LSR} and r_{SLSB} are similar for ISR, ICR and IPR functions and are close to 5.

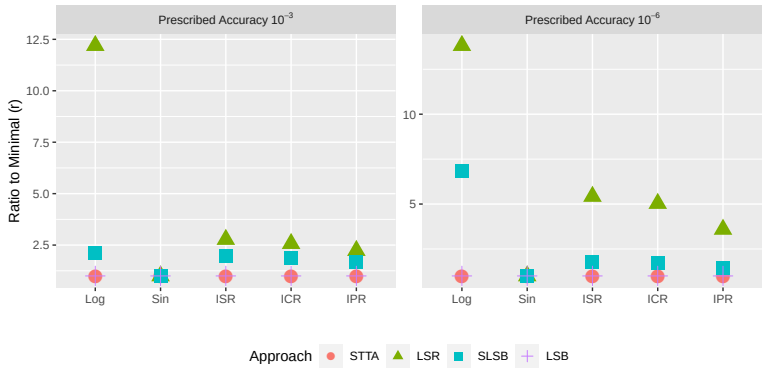


Fig. 6: Ratio to minimal comparison for 6-dimensional tensors.

Figure 6 shows r values of all the approaches for 6-dimensional tensors. r values of all the approaches for Sin function are 1. LSB and $STTA$ achieve the same compression ratios, hence their values are equal. For the prescribed accuracy of 10^{-3}, r_{LSR} value of Log function is more than 11. Therefore LSR requires 11× more storage than the best approach for this function. For the other functions, its values are between 1.75 to 3. r_{SLSB} values are slightly better than r_{LSR}, and are in the range of 1.75-2.5. For the prescribed accuracy of 10^{-6}, LSR and $SLSB$ require 12× and 6× more storage than the best approach for Log function. $SLSB$ achieves better compression ratios than LSR for ISR, ICR and IPR functions, and its r values are less than 2.5.

From the results of this section, we notice that OA values of $SLSB$ and LSB are always within the prescribed accuracy. We also observe that LSB achieves the best compression ratios among all our approaches.

6.4 Evaluation for Quantum Chemistry Data

As LSB is the best among all our proposed approaches, we will consider only $STTA$ and LSB for comparison in this section. We evaluate these approaches for tensors arising in quantum chemistry simulations. These tensors correspond to interactions between electrons of different orbitals (and virtual orbitals), and are obtained by choosing different basis sets of H_2O, H_2 and CH_4 molecules. The obtained tensors are 4-dimensional and have distinct sizes. Table 5 shows values of performance metrics for these tensors. Exp1 and Exp2 correspond to simulations with H_2O molecule, while Exp3 and Exp4 with H_2, and Exp5 with CH_4.

Table 5 shows that performance metrics of both approaches are almost same. This behavior can be understand from the fact that for the considered tensors first and last cores of the approximated tensors have maximum number of entries. For instance, if $\mathbf{G}_1$ and $\mathbf{G}_4$ are the first and last cores of the TT-representation of the approximated tensor for Exp2, then $\mathbf{G}_1$ and $\mathbf{G}_4$ both have $25 \times 25 = 625$ number of entries. The tensor obtained from Exp3 is very small and has only 16 entries. TT-representation of the approximation of this tensor requires more than 16 entries – 24 (resp. 32) for the prescribed accuracy of 10^{-3} (resp.

Appr.	Metric	Exp1	Exp2	Exp3	Exp4	Exp5
	NE	7^4	25^4	2^4	10^4	35^4
STTA	compr	46.939	88.800	-50.000	64.000	92.000
	ne	1274	43750	24	3600	120050
	OA	1.969e-04	4.819e-04	2.669e-04	2.536e-04	5.516e-04
LSB	compr	46.939	88.800	-50.000	64.000	92.000
	ne	1274	43750	24	3600	120050
	OA	1.969e-04	4.819e-04	2.669e-04	2.536e-04	5.516e-04

(a) Prescribed accuracy = 10^{-3}.

Appr.	Metric	Exp1	Exp2	Exp3	Exp4	Exp5
	NE	7^4	25^4	2^4	10^4	35^4
STTA	compr	6.122	73.760	-100.000	32.000	78.775
	ne	2254	102500	32	6800	318500
	OA	5.486e-07	5.689e-07	5.940e-16	4.525e-07	5.521e-07
LSB	compr	6.122	73.760	-100.000	32.000	78.775
	ne	2254	102500	32	6800	318500
	OA	5.486e-07	5.689e-07	5.057e-16	4.525e-07	5.521e-07

(b) Prescribed accuracy = 10^{-6}.

Table 5: Comparison of $STTA$ and LSB for 4-dimensional tensors obtained from quantum chemistry simulations.

10^{-6}). For intermediate and large size tensors, both approaches achieve very good compression ratios.

7 Conclusion

We considered the problem of designing parallel decomposition and approximation algorithms for tensors. Tensor train is a popular way to represent tensors in a concise way. It is based on low dimensional structure of the tensors. We proposed a parallel algorithm to compute tensor train representation of a tensor. We proved that the ranks of tensor train representation obtained by our algorithm are bounded by the ranks of unfolding matrices of the tensor. We also proposed several approaches to compute approximation of a tensor in tensor train representation. Our approaches are based on hierarchical partitioning of dimensions of a tensor in a balanced tree shape and how leading singular values are transmitted from the parent to its children. Our results show that the approach which transmits leading singular values to both of its children performs better in practice. A distributed memory implementation of our approaches is currently underway.

In future, we plan to extend our approach to parallelize algorithms like DMRG. We also plan to work on proving guarantees for rank-r_k approximation of our algorithms.

Acknowledgment

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No 810367).

References

1. Ahmadi-Asl, S., Cichocki, A., Phan, A.H., Oseledets, I., Abukhovich, S., Tanaka, T.: Randomized algorithms for computation of tucker decomposition and higher order svd (hosvd) (2020), https://arxiv.org/abs/2001.07124
2. Ballard, G., Knight, N., Rouse, K.: Communication lower bounds for matricized tensor times khatri-rao product. In: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 557–567 (2018), https://doi.org/10.1109/IPDPS.2018.00065
3. Ballard, G.: Parallel algorithms for cp, tucker,and tensor train decompositions (2019), http://users.wfu.edu/ballard/pdfs/PACO19.pdf
4. Ballard, G., Demmel, J., Holtz, O., Schwartz, O.: Minimizing communication in numerical linear algebra. SIAM Journal on Matrix Analysis and Applications **32**(3), 866–901 (2011). https://doi.org/10.1137/090769156, https://doi.org/10.1137/090769156
5. Beaupère, M., Grigori, L.: Communication avoiding low rank approximation based on QR with tournament pivoting (Sep 2020), https://hal.inria.fr/hal-02947991, working paper or preprint

6. Beylkin, G., Mohlenkamp, M.J.: Algorithms for numerical analysis in high dimensions. SIAM Journal on Scientific Computing **26**(6), 2133–2159 (2005). https://doi.org/10.1137/040604959, https://doi.org/10.1137/040604959
7. Carroll, J., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. Psychometrika **35**(3), 283–319 (1970), https://doi.org/10.1007/BF02310791
8. De Lathauwer, L., De Moor, B., Vandewalle, J.: On the best rank-1 and rank-(r1 ,r2 ,. . .,rn) approximation of higher-order tensors. SIAM Journal on Matrix Analysis and Applications **21**(4), 1324–1342 (2000). https://doi.org/10.1137/S0895479898346995, https://doi.org/10.1137/S0895479898346995
9. Demmel, J., Grigori, L., Rusciano, A.: An improved analysis and unified perspective on deterministic and randomized low rank matrix approximations. Tech. rep., Inria (2019), https://arxiv.org/abs/1910.00223
10. Demmel, J., Grigori, L., Hoemmen, M., Langou, J.: Communication-optimal parallel and sequential qr and lu factorizations. SIAM Journal on Scientific Computing **34**(1), A206–A239 (2012). https://doi.org/10.1137/080731992, https://doi.org/10.1137/080731992
11. Fannes, M., Nachtergaele, B., Werner, R.F.: Finitely correlated states on quantum spin chains. Comm. Math. Phys. **144**(3), 443–490 (1992), https://doi.org/10.1007/BF02099178
12. Grasedyck, L.: Hierarchical singular value decomposition of tensors. SIAM Journal on Matrix Analysis and Applications **31**(4), 2029–2054 (2010). https://doi.org/10.1137/090764189, https://doi.org/10.1137/090764189
13. Hackbusch, W., Khoromskij, B.N., Tyrtyshnikov, E.E.: Hierarchical kronecker tensor-product approximations. Journal of Numerical Mathematics **13**(2), 119 – 156 (01 Jun 2005). https://doi.org/https://doi.org/10.1515/1569395054012767, https://doi.org/10.1515/1569395054012767
14. Hackbusch, W., Kühn, S.: A new scheme for the tensor representation. Journal of Fourier Analysis and Applications **15**(5), 706–722 (oct 2009). https://doi.org/10.1007/s00041-009-9094-9, https://doi.org/10.1007/s00041-009-9094-9
15. Harshman, R.A.: Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. UCLA Working Papers in Phonetics **16**, 1–84 (1970)
16. Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. Journal of Mathematics and Physics **6**(1-4), 164–189 (1927). https://doi.org/10.1002/sapm192761164, https://doi.org/10.1002/sapm192761164
17. Håstad, J.: Tensor rank is np-complete. Journal of Algorithms **11**(4), 644 – 654 (1990). https://doi.org/https://doi.org/10.1016/0196-6774(90)90014-6, https://doi.org/10.1016/0196-6774(90)90014-6
18. Kaya, O., Uçar, B.: Scalable sparse tensor decompositions in distributed memory systems. In: SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 1–11 (2015), https://doi.org/10.1145/2807591.2807624
19. Kolda, T.G., Bader, B.W., Kenny, J.P.: Higher-order web link analysis using multilinear algebra. In: Fifth IEEE International Conference on Data Mining (ICDM'05). pp. 8 pp.– (2005), https://doi.org/10.1109/ICDM.2005.77

20. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review **51**(3), 455–500 (2009). https://doi.org/10.1137/07070111X, https://doi.org/10.1137/07070111X
21. Landsberg, J.: The border rank of the multiplication of 2x2 matrices is seven. Journal of the American Mathematical Society **19**(2), 447–459 (2005). https://doi.org/10.1090/s0894-0347-05-00506-0, https://doi.org/10.1090/s0894-0347-05-00506-0
22. Larsen, B.W., Kolda, T.G.: Practical leverage-based sampling for low-rank tensor decomposition (2020), https://arxiv.org/abs/2006.16438
23. Li, J., Sun, J., Vuduc, R.: Hicoo: Hierarchical storage of sparse tensors. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 238–252 (2018), https://doi.org/10.1109/SC.2018.00022
24. Ma, L., Solomonik, E.: Accelerating alternating least squares for tensor decomposition by pairwise perturbation (2020), https://arxiv.org/abs/1811.10573
25. Nisa, I., Li, J., Sukumaran-Rajam, A., Vuduc, R., Sadayappan, P.: Load-balanced sparse mttkrp on gpus. In: 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 123–133 (2019). https://doi.org/10.1109/IPDPS.2019.00023, https://doi.org/10.1109/IPDPS.2019.00023
26. Oseledets, I.V.: Tensor-train decomposition. SIAM Journal on Scientific Computing **33**(5), 2295–2317 (2011). https://doi.org/10.1137/090752286, https://doi.org/10.1137/090752286
27. de Silva, V., Lim, L.H.: Tensor rank and the ill-posedness of the best low-rank approximation problem. SIAM Journal on Matrix Analysis and Applications **30**(3), 1084–1127 (2008). https://doi.org/10.1137/06066518X, https://doi.org/10.1137/06066518X
28. Singh, N., Ma, L., Yang, H., Solomonik, E.: Comparison of accuracy and scalability of gauss-newton and alternating least squares for cp decomposition (2020), https://arxiv.org/abs/1910.12331
29. Taguchi, Y.h., Turki, T.: A new advanced in silico drug discovery method for novel coronavirus (sars-cov-2) with tensor decomposition-based unsupervised feature extraction. PLOS ONE **15**(9), 1–16 (09 2020). https://doi.org/10.1371/journal.pone.0238907, https://doi.org/10.1371/journal.pone.0238907
30. Tucker, L.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966), https://doi.org/10.1007/BF02289464

A Accuracy Results with QRCP and SVD

Here we perform analysis of *STTA* and *LSB* methods with the approach discussed in Section 5.4 (applying QRCP before SVD, we call this approach QRCP+SVD). We consider 12-dimensional tensors with 4^{12} elements for our experiments. Similar to Section 6, these tensors are produced by low rank functions of Table 2.

A.1 Approximation error less than or close to 10^{-3}

We aim for approximations with error not far away from 10^{-3}. Table 6 shows performance of both *STTA* and *LSB* algorithms for different low rank functions. Tensor train ranks of the approximations are depicted in Table 7.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
STTA	compr	99.996	99.999	99.994	99.991	99.986
	ne	596	176	992	1580	2336
	OA	3.692e-04	2.615e-09	3.224e-04	4.233e-04	4.774e-04
LSB	compr	99.996	99.999	99.994	99.991	99.986
	ne	596	176	992	1528	2336
	OA	3.692e-04	1.252e-11	3.223e-04	5.042e-04	4.774e-04

Table 6: Prescribed accuracy = 10^{-3}.

	STTA ranks $(r_0, r_1, \cdots, r_{12})$	*LSB* ranks $(r_0, r_1, \cdots, r_{12})$
Log	$(1,3,3,3,4,4,4,4,4,4,4,4,1)$	$(1,3,3,3,4,4,4,4,4,4,4,4,1)$
Sin	$(1,2,2,2,2,2,2,2,2,2,2,2,1)$	$(1,2,2,2,2,2,2,2,2,2,2,2,1)$
ISR	$(1,4,5,5,5,5,5,5,5,5,5,4,1)$	$(1,4,5,5,5,5,5,5,5,5,5,4,1)$
ICR	$(1,4,6,6,6,7,7,7,7,6,6,4,1)$	$(1,4,6,6,6,7,7,7,6,6,6,4,1)$
IPR	$(1,4,8,8,8,8,8,8,8,8,8,4,1)$	$(1,4,8,8,8,8,8,8,8,8,8,4,1)$

Table 7: Tensor Train ranks for prescribed accuracy of 10^{-3}.

Based on the ranks of Table 7, we select the maximum rank r for each experiment. Now we consider only *Max rank* (k) number of leading singular values for each SVD and QRCP+SVD decompositions in both algorithms. We choose k= r and $r+1$ for our experiments. Ranks of unfolding matrices in some cases are less than k, therefore we select fewer number of leading singular values for those cases.

Tables 8 and 9 show the comparison of SVD and QRCP+SVD approaches to perform fixed rank approximations of unfolding matrices in *STTA* and *LSB* methods respectively.

A.2 Approximation error less than or close to 10^{-6}

Similar to the previous subsection, here we aim for approximations with error not far away from 10^{-6} and show the results in Tables 10, 11, 12, and 13.

Tables 8, 9, 12, and 13 exhibit that acuuracies of the approximations obtained by QRCP+SVD approach are comparable with the SVD approach for *STTA* as well as *LSB* methods.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
SVD	Max rank	4	2	5	7	8
	compr	99.996	99.999	99.994	99.989	99.986
	ne	672	176	992	1824	2336
	OA	2.750e-04	2.615e-09	3.224e-04	9.849e-05	4.774e-04
QRCP +SVD	Max rank	4	2	5	7	8
	compr	99.996	99.999	99.994	99.989	99.986
	ne	672	176	992	1824	2336
	OA	0.0014448	2.277e-10	0.0031970	7.142e-04	0.0023176
SVD	Max rank	5	3	6	8	9
	compr	99.994	99.998	99.992	99.986	99.983
	ne	992	384	1376	2336	2912
	OA	6.079e-06	3.467e-09	2.913e-05	1.177e-05	1.227e-04
QRCP +SVD	Max rank	5	3	6	8	9
	compr	99.994	99.998	99.992	99.986	99.983
	ne	992	384	1376	2336	2912
	OA	1.384e-05	1.428e-10	1.515e-04	5.287e-05	8.425e-04

Table 8: Comparison of SVD and QRCP+SVD for fixed ranks with *STTA* tree.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
SVD	Max rank	4	2	5	7	8
	compr	99.996	99.999	99.994	99.989	99.986
	ne	672	176	992	1824	2336
	OA	2.750e-04	1.252e-11	3.223e-04	9.847e-05	4.774e-04
QRCP +SVD	Max rank	4	2	5	7	8
	compr	99.996	99.999	99.994	99.989	99.986
	ne	672	176	992	1824	2336
	OA	9.336e-04	7.338e-12	0.0017088	4.554e-04	0.0014991
SVD	Max rank	5	3	6	8	9
	compr	99.994	99.998	99.992	99.986	99.983
	ne	992	384	1376	2336	2912
	OA	6.079e-06	2.177e-11	2.913e-05	1.177e-05	1.226e-04
QRCP +SVD	Max rank	5	3	6	8	9
	compr	99.994	99.998	99.992	99.986	99.983
	ne	992	384	1376	2336	2912
	OA	1.016e-05	6.732e-12	7.719e-05	3.490e-05	6.393e-04

Table 9: Comparison of SVD and QRCP+SVD for fixed ranks with *LSB* tree.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
STTA	compr	99.993	99.999	99.987	99.981	99.971
	ne	1212	176	2240	3184	4864
	OA	2.271e-07	2.615e-09	1.834e-07	4.885e-07	4.836e-07
LSB	compr	99.993	99.999	99.987	99.981	99.970
	ne	1212	176	2240	3184	4964
	OA	2.265e-07	1.252e-11	1.834e-07	4.885e-07	3.999e-07

Table 10: Prescribed accuracy = 10^{-6}.

	STTA ranks $(r_0, r_1, \cdots, r_{12})$	*LSB* ranks $(r_0, r_1, \cdots, r_{12})$
Log	$(1, 4, 4, 5, 5, 6, 6, 6, 6, 6, 6, 4, 1)$	$(1, 4, 4, 5, 5, 6, 6, 6, 6, 6, 6, 4, 1)$
Sin	$(1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1)$	$(1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1)$
ISR	$(1, 4, 7, 8, 8, 8, 8, 8, 8, 8, 7, 4, 1)$	$(1, 4, 7, 8, 8, 8, 8, 8, 8, 8, 7, 4, 1)$
ICR	$(1, 4, 8, 9, 10, 10, 10, 10, 10, 9, 8, 4, 1)$	$(1, 4, 8, 9, 10, 10, 10, 10, 10, 9, 8, 4, 1)$
IPR	$(1, 4, 10, 12, 12, 12, 13, 12, 12, 12, 10, 4, 1)$	$(1, 4, 10, 12, 12, 13, 13, 12, 12, 12, 10, 4, 1)$

Table 11: Tensor Train ranks for prescribed accuracy of 10^{-6}.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
SVD	Max rank	6	2	8	10	13
	compr	99.992	99.999	99.986	99.979	99.965
	ne	1376	176	2336	3552	5856
	OA	1.340e-07	2.615e-09	1.744e-07	1.328e-07	8.592e-08
QRCP +SVD	Max rank	6	2	8	10	13
	compr	99.992	99.999	99.986	99.979	99.965
	ne	1376	176	2336	3552	5856
	OA	5.737e-07	2.277e-10	6.286e-07	4.117e-07	3.516e-07
SVD	Max rank	7	3	9	11	14
	compr	99.989	99.998	99.983	99.975	99.960
	ne	1824	384	2912	4256	6752
	OA	2.279e-08	3.467e-09	1.147e-08	1.219e-08	1.394e-08
QRCP +SVD	Max rank	7	3	9	11	14
	compr	99.989	99.998	99.983	99.975	99.960
	ne	1824	384	2912	4256	6752
	OA	1.167e-08	1.428e-10	4.331e-08	4.926e-08	8.778e-08

Table 12: Comparison of SVD and QRCP+SVD for fixed ranks with *STTA* tree.

Appr.	Metric	*Log*	*Sin*	*ISR*	*ICR*	*IPR*
SVD	Max rank	6	2	8	10	13
	compr	99.992	99.999	99.986	99.979	99.965
	ne	1376	176	2336	3552	5856
	OA	1.323e-07	1.252e-11	1.744e-07	1.328e-07	8.590e-08
QRCP +SVD	Max rank	6	2	8	10	13
	compr	99.992	99.999	99.986	99.979	99.965
	ne	1376	176	2336	3552	5856
	OA	3.555e-07	7.338e-12	3.399e-07	2.854e-07	1.397e-07
SVD	Max rank	7	3	9	11	14
	compr	99.989	99.998	99.983	99.975	99.960
	ne	1824	384	2912	4256	6752
	OA	2.753e-09	2.177e-11	1.147e-08	1.214e-08	1.387e-08
QRCP +SVD	Max rank	7	3	9	11	14
	compr	99.989	99.998	99.983	99.975	99.960
	ne	1824	384	2912	4256	6752
	OA	6.620e-09	6.732e-12	1.910e-08	2.732e-08	3.017e-08

Table 13: Comparison of SVD and QRCP+SVD for fixed ranks with *LSB* tree.