Parallel Tensor Train through Hierarchical Decomposition

Suraj Kumar

Inria Paris

TOPAL Working Group

May 20, 2021
This is joint work with ...

- Laura Grigori – Inria Paris, France
- Olivier Beaumont – Inria Bordeaux, France
- Alena Shilova – Inria Bordeaux, France
Overview

1. Introduction

2. Low Rank Tensor Representations

3. Algorithms to Compute Tensor Train Representation
   - Sequential Algorithms
   - Parallel Algorithms

4. Experimental Evaluation

5. Conclusion
Table of Contents

1 Introduction

2 Low Rank Tensor Representations

3 Algorithms to Compute Tensor Train Representation
   - Sequential Algorithms
   - Parallel Algorithms

4 Experimental Evaluation

5 Conclusion
<table>
<thead>
<tr>
<th>Dimension</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vector</td>
</tr>
<tr>
<td>2</td>
<td>Matrix</td>
</tr>
<tr>
<td>3</td>
<td>3-dimensional tensor</td>
</tr>
<tr>
<td>4</td>
<td>4-dimensional tensor</td>
</tr>
</tbody>
</table>
Tensors

- Tensors are used in several domains
  - Quantum molecular dynamics, signal processing, data mining, neurosciences, computer vision, psychometrics, chemometrics, ...

- Memory and computation requirements are exponential in the number of dimensions
  - A molecular simulation involving just 100 spatial orbitals manipulate a huge tensor with $4^{100}$ elements
  - People work with low dimensional structure of the tensors

- Limited work on parallelization of tensor algorithms
1. Introduction

2. Low Rank Tensor Representations

3. Algorithms to Compute Tensor Train Representation
   - Sequential Algorithms
   - Parallel Algorithms

4. Experimental Evaluation

5. Conclusion
Singular Value Decomposition (SVD) of Matrices

- It decomposes a matrix $A \in \mathbb{R}^{m \times n}$ to the form $U \Sigma V^T$
  - $U$ is an $m \times m$ orthogonal matrix
  - $V$ is an $n \times n$ orthogonal matrix
  - $\Sigma$ is an $m \times n$ rectangular diagonal matrix
- It represents a matrix as the sum of rank one matrices
  - $A = \sum_i \Sigma(i; i) U_i V_i^T$
  - Minimum number of rank one matrices required in the sum is called the rank of the original matrix
Popular Tensor Decompositions
Higher Order Generalization of SVD

- Canonical decomposition (equivalently known as Canonical Polyadic or CANDECOMP or PARAFAC)
- Tucker decomposition
- Tensor Train decomposition (equivalently known as Matrix Product States)

Tensor Notations

- $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is a $d$-dimensional tensor
- $A(i_1, \cdots, i_d)$ represent elements of $A$
- Use bold letters to denote tensors
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)

(+) For n_1 = n_2 = \cdots n_d = n, the number of entries = O(nrd)

(-) Determining the minimum value of r is an NP-complete problem

(-) No robust algorithms to compute this representation
Tucker Representation

- Represents a tensor with $d$ matrices and a small core tensor
- $A(i_1, \cdots, i_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_d=1}^{r_d} g_{\alpha_1 \cdots \alpha_d} U_1(i_1, \alpha_1) \cdots U_d(i_d, \alpha_d)$

(+) SVD based stable algorithms to compute this representation

(-) For $n_1 = n_2 = \cdots n_d = n$ and $r_1 = r_2 = \cdots = r_d = r$, the number of entries $= O(ndr + r^d)$
A $d$-dimensional tensor is represented with 2 matrices and $d-2$ 3-dimensional tensors.

$$A(i_1, i_2, \cdots, i_d) = G_1(i_1)G_2(i_2)G_d(i_d)$$

An entry of $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is computed by multiplying corresponding matrix (or row/column) of each core.
A $\in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is represented with cores $G_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$, $k=1, 2, \cdots d$, $r_0=r_d=1$ and its elements satisfy the following expression:

$$
A(i_1, \cdots, i_d) = \sum_{\alpha_0=1}^{r_0} \cdots \sum_{\alpha_d=1}^{r_d} G_1(\alpha_0, i_1, \alpha_1) \cdots G_d(\alpha_{d-1}, i_d, \alpha_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} G_1(1, i_1, \alpha_1) \cdots G_d(\alpha_{d-1}, i_d, 1)
$$

For $n_1 = n_2 = \cdots = n_d = n$ and $r_1 = r_2 = \cdots = r_{d-1} = r$, the number of entries $= O(n dr^2)$
# Table of Contents

1. Introduction

2. Low Rank Tensor Representations

3. Algorithms to Compute Tensor Train Representation
   - Sequential Algorithms
   - Parallel Algorithms

4. Experimental Evaluation

5. Conclusion
Unfolding Matrices of a Tensor & Notations

- Frobenius norm of a matrix $A$ is defined as, $||A||_F = \sqrt{\sum_{i,j} A(i;j)^2}$
- Frobenius norm of a $d$-dimensional tensor $A$ is defined as, $||A||_F = \sqrt{\sum_{i_1,i_2,\ldots,i_d} A(i_1,i_2,\ldots,i_d)^2}$

$k$-th unfolding matrix

$A_k$ denotes $k$-th unfolding matrix of tensor $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.

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

- Size of $A_k$ is $(\prod_{l=1}^k n_l) \times (\prod_{l=k+1}^d n_l)$
- $r_k$ denotes the rank of $A_k$.

- $(r_1, r_2, \ldots, r_{d-1})$ denotes the ranks of unfolding matrices of the tensor.
Algorithm 1 Tensor Train Decomposition

**Require:** $d$-dimensional tensor $A$ and ranks $(r_1, r_2, \cdots, r_{d-1})$

**Ensure:** Cores $G_k(\alpha_{k-1}, n_k, \alpha_k)_{1 \leq k \leq d}$ of the Tensor Train representation with $\alpha_k \leq r_k$ and $\alpha_0 = \alpha_d = 1$

1. Temporary tensor: $C = A$, $\alpha_0 = 1$
2. for $k = 1 : d - 1$ do
3. $A_k = \text{reshape}(C, \alpha_{k-1} n_k, \frac{\text{numel}(C)}{\alpha_{k-1} n_k})$
4. Compute SVD: $A_k = U \Sigma V^T$
5. Compute rank of $\Sigma$, $\alpha_k = \text{rank}(\Sigma)$
6. New core: $G_k := \text{reshape}(U(:, 1 : \alpha_k), \alpha_{k-1}, n_k, \alpha_k)$
7. $C = \Sigma(1 : \alpha_k; 1 : \alpha_k) V^T(1 : \alpha_k; )$
8. end for
9. $G_d = C$, $\alpha_d = 1$
10. return $G_1, \cdots, G_d$
Algorithm 2 Tensor Train Approximation

Require: $d$-dimensional tensor $A$ and expected accuracy $\epsilon$

Ensure: Cores $G_k(\alpha_{k-1}, n_k, \alpha_k)_{1 \leq k \leq d}$ of the approximated tensor $B$ in Tensor Train representation such that $||A - B||_F$ is not more than $\epsilon$

1: Temporary tensor: $C = A$, $\alpha_0 = 1$, $\delta = \frac{\epsilon}{\sqrt{d-1}}$

2: for $k = 1 : d - 1$ do

3: $A_k = \text{reshape}(C, \alpha_{k-1} n_k, \frac{\text{numel}(C)}{\alpha_{k-1} n_k})$

4: Compute SVD: $A_k = U \Sigma V^T$

5: Compute $\alpha_k$ such that $A_k = U(:,1 : \alpha_k) \Sigma(1 : \alpha_k; 1 : \alpha_k) V^T(1 : \alpha_k; ) + E_k$ and $||E_k||_F \leq \delta$

6: New core: $G_k := \text{reshape}(U(:,1 : \alpha_k), r_{k-1}, n_k, r_k)$

7: $C = \Sigma(1 : \alpha_k; 1 : \alpha_k) V^T(1 : \alpha_k; )$

8: end for

9: $G_d = C$, $\alpha_d = 1$

10: return $B$ in Tensor Train representation with cores $G_1, \cdots, G_d$
Separation of Dimensions in Sequential Algorithms

\[ \{i_1, i_2, \ldots, i_d\} \]

\[ \{i_1\} \]

\[ \{i_2\} \]

\[ \{i_3\} \]

\[ \{i_4, \ldots, i_d\} \]

\[ \{i_d\} \]

\[ i_1, i_2, \ldots, i_d \]

\[ i_2, \ldots, i_d \]

\[ i_3, \ldots, i_d \]

\[ i_{d-1}, i_d \]

\[ i_d-1, i_d \]

\[ i_d \]
Separation of Dimensions for Maximum Parallelization

\{i_1, i_2, \cdots, i_d\}
\{i_1, i_2\}
\{i_1\}

\{i_2, \cdots, i_d\}
\{i_2\}
\{i_2\}

\{i_3, \cdots, i_d\}
\{i_3\}

\{i_{d-1}, i_d\}
\{i_{d-1}\}

\{i_d\}
\{i_d\}

\log_2 d

Suraj Kumar (Inria Paris)
Table of Contents

1 Introduction

2 Low Rank Tensor Representations

3 Algorithms to Compute Tensor Train Representation
   - Sequential Algorithms
   - Parallel Algorithms

4 Experimental Evaluation

5 Conclusion
Extra Definitions for Parallel Algorithms

- Original indices of a tensor are called external indices
- Indices obtained due to SVD are called internal indices
  - $A(\alpha, i_1, i_2, i_3, \beta)$ has 3 external and 2 internal indices
- $nEI(A)$ denotes the number of external indices of $A$

### $k$-th Unfolding Matrix

$k$-th unfolding of a tensor with elements $A(\alpha, i_1, i_2, \cdots, i_k, i_{k+1}, \cdots, \beta)$ is represented as, $A_k = [A_k(\alpha, i_1, i_2, \cdots, i_k; i_{k+1} \cdots, \beta)]$.

All indices from the beginning to $i_k$ denote the rows of $A_k$ and the remaining indices denote the columns of $A_k$.

- **Tensor**($A_l$) converts an unfolding matrix $A_l$ to its tensor form
Algorithm 3: PTT-decomposition (parallel Tensor Train Decomposition)

Require: $d$-dimensional tensor $A$ and ranks $(r_1, r_2, \cdots, r_{d-1})$

Ensure: Cores $G_k(\alpha_{k-1}, n_k, \alpha_k)_{1 \leq k \leq d}$ of the Tensor Train representation with $\alpha_k \leq r_k$ and $\alpha_0 = \alpha_d = 1$

1: if $nEI(A) > 1$ then
2: Find the middle external index $k$
3: Compute unfolding matrix $A_k$
4: Compute SVD: $A_k = U \Sigma V^T$
5: Compute rank of $\Sigma$, $\alpha_k = \text{rank}(\Sigma)$
6: Select diagonal matrices $X_k$, $S_k$ and $Y_k$ such that $X_k S_k Y_k = \Sigma(1 : \alpha_k; 1 : \alpha_k)$
7: $A_{left} = \text{Tensor}(U(; 1 : \alpha_k)X_k)$
8: list1 = PTT-decomposition($A_{left}, (r_1, \cdots, r_{k-1}, \alpha_k)$)
9: $A_{right} = \text{Tensor}(Y_k V^T(1 : \alpha_k;))$
list2 = PTT-decomposition($A_{right}, (\alpha_k, r_{k+1}, \ldots, r_{d-1})$)

return \{list1, list2\}

else

Find the external index \( k \)

if \( k \) is the last index of \( A \) then

\( \alpha_k = 1 \)

else if \( k \) is the first index of \( A \) then

\( \alpha_{k-1} = 1 \)

\( A(i_k, \beta) = \sum_{\beta=1}^{\alpha_k} A(i_k, \beta)S_k(\beta; \beta) \)

else

\( A(\gamma, i_k, \beta) = \sum_{\beta=1}^{\alpha_k} A(\gamma, i_k, \beta)S_k(\beta; \beta) \)

end if

\( G_k = A \)

return \( G_k \)

end if
Diagramatic Representation of the Algorithm

\[ A(i_1, i_2, i_3, \alpha_3) \]
\[ A(\alpha_3, i_4, i_5, i_6) \]
\[ A(i_1, i_2, i_3, i_4, i_5, i_6) \]

\[ i_1 i_2 i_3 \]
\[ i_4 i_5 i_6 \]

\[ A(i_1, i_2, \alpha_2) \]
\[ A(\alpha_2, i_3, \alpha_3) \]
\[ A(\alpha_3, i_4, \alpha_4) \]
\[ A(\alpha_4, i_5, \alpha_5) \]

\[ i_1 \]
\[ i_2 \alpha_2 \]
\[ i_3 \alpha_3 \]
\[ i_4 \alpha_4 \]
\[ i_5 \alpha_5 \]
\[ i_6 \]

\[ \alpha_3 i_4 i_5 \]
Ranks of Tensor Train Representation (Algorithm 3)

**Theorem**

If for each unfolding $A_k$ of a $d$-dimensional tensor $A$, $\text{rank}(A_k) = r_k$, then Algorithm 3 produces a Tensor Train representation with ranks not higher than $r_k$.

The rank of the $k$th-unfolding matrix is $r_k$; hence it can be written as:

$$A_k(i_1, \ldots, i_k; i_{k+1}, \ldots, i_d) = \sum_{\alpha=1}^{r_k} U(i_1, \ldots, i_k; \alpha) \Sigma(\alpha; \alpha) V^T(\alpha; i_{k+1}, \ldots, i_d)$$

$$= \sum_{\alpha=1}^{r_k} U(i_1, \ldots, i_k; \alpha) X(\alpha; \alpha) S(\alpha; \alpha) Y(\alpha; \alpha) V^T(\alpha; i_{k+1}, \ldots, i_d)$$

$$= \sum_{\alpha=1}^{r_k} B(i_1, \ldots, i_k; \alpha) S(\alpha; \alpha) C(\alpha; i_{k+1}, \ldots, i_d).$$

In matrix form we obtain, $A_k = BSC$, $B = A_k C^{-1} S^{-1} = A_k Z$, $C = S^{-1} B^{-1} A_k = W A_k$. or in the index form, $B(i_1, \ldots, i_k; \alpha) = \sum_{i_{k+1}=1}^{n_{k+1}} \cdots \sum_{i_d=1}^{n_d} A(i_1, \ldots, i_d) Z(i_{k+1}, \ldots, i_d; \alpha)$,

$$C(\alpha; i_{k+1}, \ldots, i_d) = \sum_{i_1=1}^{n_1} \cdots \sum_{i_k=1}^{n_k} A(i_1, \ldots, i_d) W(\alpha; i_1, i_2, \ldots, i_k).$$

$B$ and $C$ can be treated as $k + 1$ and $d-k+1$ dimensional tensors $B$ and $C$ respectively. We prove that $\text{rank}(B_k') \leq r_k'$, $1 \leq k' \leq k-1$ and $\text{rank}(C_k') \leq r_k'$, $k+1 \leq k' \leq d-1$. 

Suraj Kumar (Inria Paris)
Algorithm 4: PTT-approx (Parallel Tensor Train Approximation)

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

1: if \( nEI(A) > 1 \) then
2: Find the middle external index \( k \)
3: Compute unfolding matrix \( A_k \)
4: Compute SVD: \( A_k = U \Sigma V^T \)
5: Compute truncation accuracy \( \Delta \)
6: Compute \( \alpha_k \) such that \( A_k = U(1 : \alpha_k) \Sigma(1 : \alpha_k; 1 : \alpha_k)V^T(1 : \alpha_k; ) + E_k \) and \( \| E_k \|_F \leq \Delta \)
7: Select diagonal matrices \( X_k, S_k \) and \( Y_k \) such that \( X_k S_k Y_k = \Sigma(1 : \alpha_k; 1 : \alpha_k) \)
8: \( A_{\text{left}} = \text{Tensor}(U(1 : \alpha_k)X_k) \)
9: list1 = PTT-approx\( (A_{\text{left}}, \epsilon_1) \)
10: \( A_{\text{right}} = \text{Tensor}(Y_k V^T(1 : \alpha_k; )) \)
11: list2 = PTT-approx($A_{right}$, $\epsilon_2$)
12: return \{list1, list2\}
13: else
14: Find the external index $k$
15: if $k$ is the last index of $A$ then
16: \[\alpha_k = 1\]
17: else if $k$ is the first index of $A$ then
18: \[\alpha_{k-1} = 1\]
19: \[A(i_k, \beta) = \sum_{\beta=1}^{\alpha_k} A(i_k, \beta)S_k(\beta; \beta)\]
20: else
21: \[A(\gamma, i_k, \beta) = \sum_{\beta=1}^{\alpha_k} A(\gamma, i_k, \beta)S_k(\beta; \beta)\]
22: end if
23: \[G_k = A\]
24: return $G_k$
25: end if
Frobenius Error with Product of Approximated Matrices

The SVD of a real matrix $A$ can be written as,

$$A = (U_1U_2) \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} (V_1V_2)^T = U_1\Sigma_1 V_1^T + U_2\Sigma_2 V_2^T$$

$$= U_1\Sigma_1 V_1^T + E_A = BSC + E_A.$$ 

Here $B = U_1X$, $C = YV_1^T$ 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 $C$.

$$||A - \hat{B}SC\hat{C}||_F^2 \approx ||E_A||_F^2 + ||BSE_C||_F^2 + ||E_BS\hat{C}||_F^2$$
Our Approximation Approaches

We propose 3 approaches based on how leading singular values of the unfolding matrix are passed to the left and right subtensors in Algorithm 4.

- **Leading Singular values to Right subtensor (LSR)**
- **Square root of Leading Singular values to Both subtensors (SLSB)**
- **Leading Singular values to Both subtensors (LSB)**

<table>
<thead>
<tr>
<th>Approach</th>
<th>Description</th>
<th>$\Delta$</th>
<th>$\epsilon_1$</th>
<th>$\epsilon_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSR</td>
<td>$X = I$, $Y = \Sigma_\alpha$, $S = I$</td>
<td>$\frac{\epsilon}{\sqrt{d-1}}$</td>
<td>$\epsilon \sqrt{\frac{(d-2)(d_1-1)}{(d-1)(d_2-1+(d_1-1)tr(\Sigma_\alpha^2))}}$</td>
<td>$\epsilon \sqrt{\frac{(d-2)(d_2-1)}{(d-1)(d_2-1+(d_1-1)tr(\Sigma_\alpha^2))}}$</td>
</tr>
<tr>
<td>SLSB</td>
<td>$X = Y = \frac{1}{\alpha} \Sigma_\alpha$, $S = I$</td>
<td>$\frac{\epsilon}{\sqrt{d-1}}$</td>
<td>$\epsilon \sqrt{\frac{d_1-1}{(d-1)tr(\Sigma_\alpha)}}$</td>
<td>$\epsilon \sqrt{\frac{d_2-1}{(d-1)tr(\Sigma_\alpha)}}$</td>
</tr>
<tr>
<td>LSB</td>
<td>$X = Y = \Sigma_\alpha$, $S = \Sigma_\alpha^{-1}$</td>
<td>$\frac{\epsilon}{\sqrt{d-1}}$</td>
<td>$\epsilon \sqrt{\frac{d_1-1}{d-1}}$</td>
<td>$\epsilon \sqrt{\frac{d_2-1}{d-1}}$</td>
</tr>
<tr>
<td>STTA</td>
<td>$X = I$, $Y = \Sigma_\alpha$, $S = I$</td>
<td>$\frac{\epsilon}{\sqrt{d-1}}$</td>
<td>0</td>
<td>$\epsilon \sqrt{\frac{d_2-1}{d-1}}$</td>
</tr>
</tbody>
</table>

- **STTA** represents *Sequential Tensor Train Approximation*
Table of Contents

1 Introduction

2 Low Rank Tensor Representations

3 Algorithms to Compute Tensor Train Representation
   • Sequential Algorithms
   • Parallel Algorithms

4 Experimental Evaluation

5 Conclusion
Low Rank Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log</td>
<td>( \log(\sum_{j=1}^{d} j i_j) )</td>
</tr>
<tr>
<td>Sin</td>
<td>( \sin(\sum_{j=1}^{d} i_j) )</td>
</tr>
<tr>
<td>Inverse-Square-Root (ISR)</td>
<td>( \frac{1}{\sqrt{\sum_{j=1}^{d} i_j^2}} )</td>
</tr>
<tr>
<td>Inverse-Cube-Root (ICR)</td>
<td>( \frac{1}{\sqrt[3]{\sum_{j=1}^{d} i_j^3}} )</td>
</tr>
<tr>
<td>Inverse-Penta-Root (IPR)</td>
<td>( \frac{1}{\sqrt[5]{\sum_{j=1}^{d} i_j^5}} )</td>
</tr>
</tbody>
</table>

We consider \( d = 12 \) and \( i_j \in \{1, 2, 3, 4\}_{1 \leq j \leq d} \). This setting produces a 12-dimensional tensor with \( 4^{12} \) elements for each low rank function.
Comparison of All Approaches for 12-dimensional Tensors

- Prescribed accuracy = $10^{-6}$
- compr: compression ratio, ne: number of elements in approximation, OA: approximation accuracy

<table>
<thead>
<tr>
<th>Appr.</th>
<th>Metric</th>
<th>Log</th>
<th>Sin</th>
<th>ISR</th>
<th>ICR</th>
<th>IPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTA</td>
<td>ne</td>
<td>1212</td>
<td>176</td>
<td>2240</td>
<td>3184</td>
<td>4864</td>
</tr>
<tr>
<td></td>
<td>OA</td>
<td>2.271e-07</td>
<td>2.615e-09</td>
<td>1.834e-07</td>
<td>4.884e-07</td>
<td>4.836e-07</td>
</tr>
<tr>
<td>LSR</td>
<td>ne</td>
<td>30632</td>
<td>344</td>
<td>14196</td>
<td>21176</td>
<td>29524</td>
</tr>
<tr>
<td></td>
<td>OA</td>
<td>3.629e-08</td>
<td>1.412e-11</td>
<td>1.118e-07</td>
<td>8.520e-08</td>
<td>5.811e-08</td>
</tr>
<tr>
<td>SLSB</td>
<td>ne</td>
<td>33772</td>
<td>176</td>
<td>8068</td>
<td>14824</td>
<td>21792</td>
</tr>
<tr>
<td></td>
<td>OA</td>
<td>2.820e-08</td>
<td>6.144e-12</td>
<td>1.118e-07</td>
<td>8.518e-08</td>
<td>5.664e-08</td>
</tr>
<tr>
<td>LSB</td>
<td>ne</td>
<td>1212</td>
<td>176</td>
<td>2240</td>
<td>3184</td>
<td>4964</td>
</tr>
<tr>
<td></td>
<td>OA</td>
<td>2.265e-07</td>
<td>1.252e-11</td>
<td>1.834e-07</td>
<td>4.884e-07</td>
<td>3.999e-07</td>
</tr>
</tbody>
</table>
## Alternatives to SVD

- SVD is expensive
- Good alternatives to SVD: QR factorization with column pivoting (QRCP), randomized SVD (RSVD)

## SVD vs QRCP+SVD vs RSVD for Log tensor

<table>
<thead>
<tr>
<th>Approach</th>
<th>Rank</th>
<th>compr</th>
<th>ne</th>
<th>LSB-OA</th>
<th>STTA-OA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>5</td>
<td>99.994</td>
<td>992</td>
<td>6.079e-06</td>
<td>6.079e-06</td>
</tr>
<tr>
<td>QRCP+SVD</td>
<td>5</td>
<td>99.994</td>
<td>992</td>
<td>1.016e-05</td>
<td>1.384e-05</td>
</tr>
<tr>
<td>RSVD</td>
<td>5</td>
<td>99.992</td>
<td>992</td>
<td>6.079e-06</td>
<td>6.079e-06</td>
</tr>
<tr>
<td>SVD</td>
<td>6</td>
<td>99.992</td>
<td>1376</td>
<td>1.323e-07</td>
<td>1.340e-07</td>
</tr>
<tr>
<td>QRCP+SVD</td>
<td>6</td>
<td>99.992</td>
<td>1376</td>
<td>3.555e-07</td>
<td>5.737e-07</td>
</tr>
<tr>
<td>RSVD</td>
<td>6</td>
<td>99.992</td>
<td>1376</td>
<td>1.322e-07</td>
<td>1.322e-07</td>
</tr>
<tr>
<td>SVD</td>
<td>7</td>
<td>99.989</td>
<td>1824</td>
<td>2.753e-09</td>
<td>2.279e-08</td>
</tr>
<tr>
<td>QRCP+SVD</td>
<td>7</td>
<td>99.989</td>
<td>1824</td>
<td>6.620e-09</td>
<td>1.167e-08</td>
</tr>
<tr>
<td>RSVD</td>
<td>7</td>
<td>99.989</td>
<td>1824</td>
<td>2.760e-09</td>
<td>2.774e-09</td>
</tr>
</tbody>
</table>
Performance Comparison

Sequential performance with Log tensor

Number of computations for both algorithms = $O(n^d)$

Parallel performance counts along the critical path on $P$ processors

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Computations</th>
<th>Communications</th>
<th># Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTA-RSVD</td>
<td>$O\left(\frac{n^d}{P}\right)$</td>
<td>$O(n^d)$</td>
<td>$O(d \log P)$</td>
</tr>
<tr>
<td>LSB-RSVD</td>
<td>$O\left(\frac{n^d}{P}\right)$</td>
<td>$O\left(\frac{n^d}{\sqrt{P}} \log P\right)$</td>
<td>$O(\log d \log P)$</td>
</tr>
</tbody>
</table>
Table of Contents

1 Introduction

2 Low Rank Tensor Representations

3 Algorithms to Compute Tensor Train Representation
   - Sequential Algorithms
   - Parallel Algorithms

4 Experimental Evaluation

5 Conclusion
Conclusion

- Proposed parallel algorithms to compute tensor train decomposition and approximation of a tensor
- LSB approach achieves similar compression to the sequential algorithm
- Accuracies of all approaches are within prescribed limit

Ongoing Work

- Proving quasi optimality for parallel approximation algorithms
- Implementation of parallel algorithms for distributed memory systems
Thank You!