Parallel Tensor Train through Hierarchical Decomposition

Suraj Kumar

Inria Paris

TOPAL Working Group
May 20, 2021

Collaborators

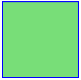
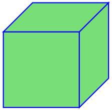
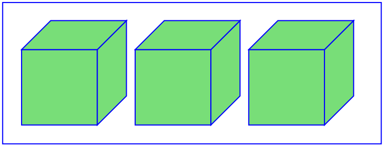
This is joint work with ...

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

Overview

Table of Contents

Tensors: Multidimensional Arrays

Dimension	Name	
1	Vector	
2	Matrix	
3	3-dimensional tensor	
4	4-dimensional tensor	

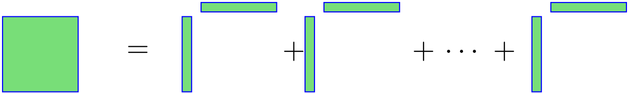
Tensors

- 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

Table of Contents

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
  - Σ 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

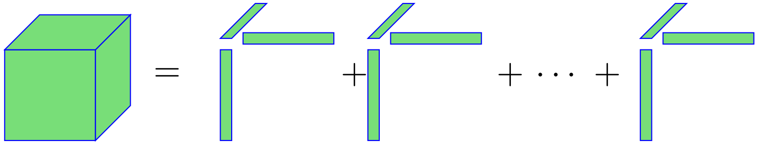


- Canonical decomposition (equivalently known as Canonical Polyadic or CANDECOMP or PARAFAC)
- Tucker decomposition
- Tensor Train decomposition (equivalently known as Matrix Product States)

Tensor Notations

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

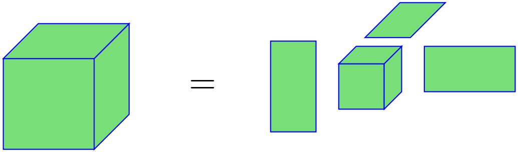
Canonical Representation



- $\mathbf{A}(i_1, \cdots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha) \cdots U_d(i_d, \alpha)$

- (+) For $n_1 = n_2 = \cdots n_d = n$, the number of entries $= \mathcal{O}(nrd)$
- (-) Determining the minimum value of r is an NP-complete problem
- (-) No robust algorithms to compute this representation

Tucker Representation

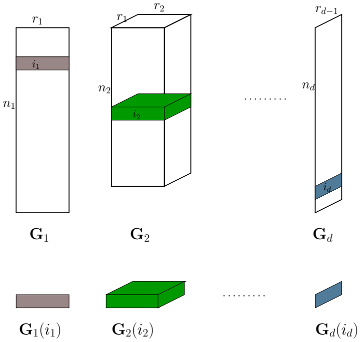


- Represents a tensor with d matrices and a small core tensor
- $\mathbf{A}(i_1, \cdots, i_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_d=1}^{r_d} \mathbf{g}_{\alpha_1 \cdots \alpha_d} U_1(i_1, \alpha_1) \cdots U_d(i_d, \alpha_d)$

- (+) 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 $= \mathcal{O}(ndr + r^d)$

Tensor Train Representation: Product of Matrices View

- A d-dimensional tensor is represented with 2 matrices and d-2 3-dimensional tensors.



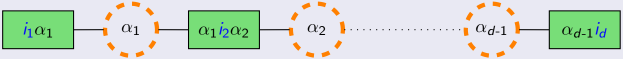
$$\mathbf{A}(i_1, i_2, \cdots, i_d) = \mathbf{G}_1(i_1)\mathbf{G}_2(i_2)\cdots\mathbf{G}_d(i_d)$$

An entry of $\mathbf{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is computed by multiplying corresponding matrix (or row/column) of each core.

Tensor Train Representation

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

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



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

Table of Contents

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 $\mathbf{A}$ is defined as, $||\mathbf{A}||_F = \sqrt{\sum_{i_1,i_2,\cdots,i_d} \mathbf{A}(i_1, i_2, \cdots, i_d)^2}$

k-th unfolding matrix

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

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

- 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.

- $(r_1, r_2, \cdots, r_{d-1})$ denotes the ranks of unfolding matrices of the tensor.

Tensor Train Decomposition (Oseledets's Algorithm)

Algorithm 1 Tensor Train Decomposition

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

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

Temporary tensor: $\mathbf{C} = \mathbf{A}$, $\alpha_0 = 1$

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

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

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

    Compute rank of Σ, $\alpha_k = \text{rank}(\Sigma)$

    New core: $\mathbf{G}_k := reshape(U(; 1 : \alpha_k), \alpha_{k-1}, n_k, \alpha_k)$

    $\mathbf{C} = \Sigma(1 : \alpha_k; 1 : \alpha_k) V^T (1 : \alpha_k;)$

end for

$\mathbf{G}_d = \mathbf{C}$, $\alpha_d = 1$

return $\mathbf{G}_1, \cdots, \mathbf{G}_d$

Tensor Train Approximation (Oseledets's Algorithm)

Algorithm 2 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 Tensor Train representation such that $||\mathbf{A}-\mathbf{B}||_F$ is not more than ϵ

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

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

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

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

  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 $||E_K||_F \leq \delta$

  New core: $\mathbf{G}_k := reshape(U(;1:\alpha_k), r_{k-1}, n_k, r_k)$

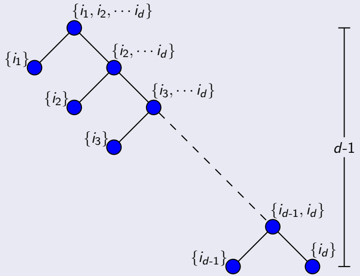
  $\mathbf{C} = \Sigma(1:\alpha_k;1:\alpha_k)V^T(1:\alpha_k;)$

end for

$\mathbf{G}_d=\mathbf{C}$, $\alpha_d=1$

return $\mathbf{B}$ in Tensor Train representation with cores $\mathbf{G}_1,\cdots,\mathbf{G}_d$

Separatation of Dimensions in Sequential Algorithms



Separatation of Dimensions for Maximum Parallelization

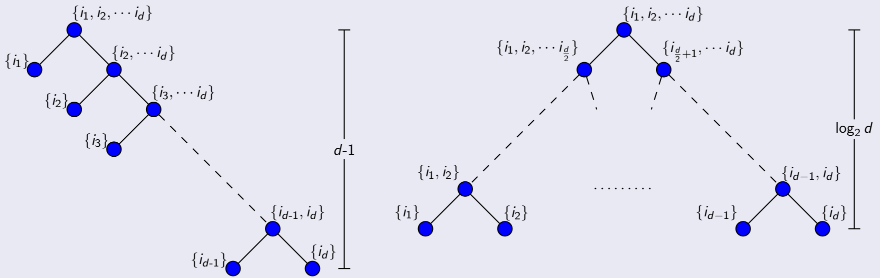


Table of Contents

Extra Definitions for Parallel Algorithms

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

k-th Unfolding Matrix

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

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 $\mathbf{A}$ and ranks $(r_1, r_2, \cdots r_{d-1})$

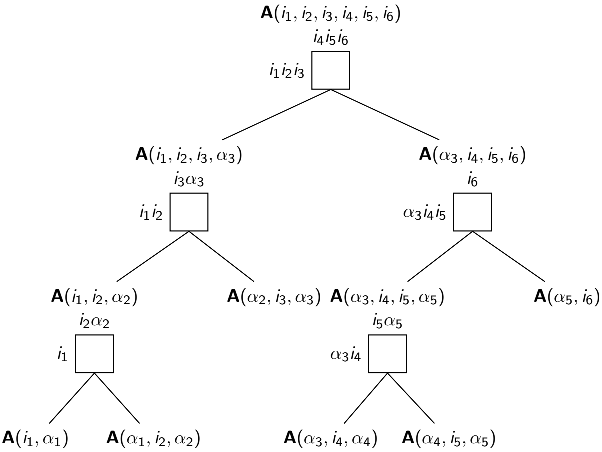
Ensure: Cores $\mathbf{G}_k(\alpha_{k-1}, n_k, \alpha_k)_{1\leq k\leq d}$ of the Tensor Train representation with $\alpha_k \leq r_k$ and $\alpha_0 = \alpha_d = 1$

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($\mathbf{A}_{left}$, $(r_1, \cdots r_{k-1}, \alpha_k)$)
    $\mathbf{A}_{right} = \mathbf{Tensor}(Y_k V^T(1:\alpha_k;))$

Parallel Tensor Train decomposition II

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

Diagramatic Representation of the Algorithm



Ranks of Tensor Train Representation (Algorithm 3)

Theorem

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

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

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

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

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

B and C can be treated as $k+1$ and d-$k+1$ dimensional tensors $\mathbf{B}$ and $\mathbf{C}$ respectively. We prove that rank$(B_{k'}) \leq r_{k'}$ $_{1 \leq k' \leq k-1}$ and rank$(C_{k'}) \leq r_{k'}$ $_{k+1 \leq k' \leq d-1}$.

Parallel Tensor Train Approximation I

Algorithm 4: PTT-approx (Parallel Tensor Train Approximation)

Require: d-dimensional tensor $\mathbf{A}$ and expected accuracy ϵ

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

if $nEl(\mathbf{A}) > 1$ **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 $||E_K||_F \le \Delta$
    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-approx($\mathbf{A}_{left}$, ϵ_1)
    $\mathbf{A}_{right} = \mathbf{Tensor}(Y_k V^T(1:\alpha_k;))$

Parallel Tensor Train Approximation II

list2 = PTT-approx($\mathbf{A}_{right}$, ϵ_2)
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**
$\alpha_{k\text{-}1} = 1$
$\mathbf{A}(i_k, \beta) = \sum_{\beta=1}^{\alpha_k} \mathbf{A}(i_k, \beta) S_k(\beta; \beta)$
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

Frobenius Error with Product of Approximated Matrices

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

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

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.

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

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*)

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	$\epsilon\sqrt{\frac{d_2-1}{d-1}}$

- *STTA* represents *Sequential Tensor Train Approximation*

Table of Contents

Low Rank Functions

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

We consider $d = 12$ and $i_j \in \{1, 2, 3, 4\}_{1 \le j \le d}$. 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 aprroximation, OA: approximation accuracy

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

Alternatives to SVD

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

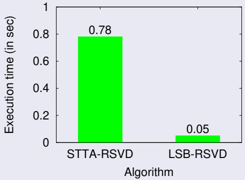
SVD vs QRCP+SVD vs RSVD for *Log* tensor

Approach	Rank	compr	ne	*LSB*-OA	*STTA*-OA
SVD	5	99.994	992	6.079e-06	6.079e-06
QRCP+SVD				1.016e-05	1.384e-05
RSVD				6.079e-06	6.079e-06
SVD	6	99.992	1376	1.323e-07	1.340e-07
QRCP+SVD				3.555e-07	5.737e-07
RSVD				1.322e-07	1.322e-07
SVD	7	99.989	1824	2.753e-09	2.279e-08
QRCP+SVD				6.620e-09	1.167e-08
RSVD				2.760e-09	2.774e-09

Performance Comparison

Sequential performance with *Log* tensor

Number of computations for both algorithms $= \mathcal{O}(n^d)$



Parallel performance counts along the critical path on P processors

Algorithm	# Computations	Communications	# Messages
STTA-RSVD	$\mathcal{O}(\frac{n^d}{P})$	$\mathcal{O}(n^d)$	$\mathcal{O}(d \log P)$
LSB-RSVD	$\mathcal{O}(\frac{n^d}{P})$	$\mathcal{O}(\frac{n^d}{\sqrt{P}} \log P)$	$\mathcal{O}(\log d \log P)$

Table of Contents

Conclusion & Ongoing Work

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!