Communication-Optimal Algorithms for CP Decompositions of Dense Tensors

Grey Ballard, Koby Hayashi, Ramakrishnan Kannan, Nicholas Knight, Kathryn Rouse

May 8, 2018

SIAM Conference on Applied Linear Algebra
MS04: Constrained Low-Rank Matrix and Tensor Approximations
We establish communication lower bounds for matricized-tensor times Khatri-Rao product (MTTKRP)
- key kernel for computing CP decomposition

We present optimal parallel dense MTTKRP algorithm
- attains the lower bound to within constant factors

We implement and benchmark optimal CP-ALS algorithm
- remains computation bound and scales well
- dimension tree optimization avoids redundant computation
CP Decomposition: sum of outer products

Matrix: \( \mathbf{M} \approx \sum_{r=1}^{R} \mathbf{u}_r (\sigma_r \mathbf{v}_r^T) \)

Tensor: \( \mathbf{X} \approx \sum_{r=1}^{R} \mathbf{u}_r \odot \mathbf{v}_r \odot \mathbf{w}_r \)

This is known as the CANDECOMP or PARAFAC or canonical polyadic or CP decomposition
For fixed rank $R$, we want to solve

$$\min_{u,v,w} \left\| X - \sum_{r=1}^{R} u_r \circ v_r \circ w_r \right\|$$

which is a nonlinear, nonconvex optimization problem.

- in the matrix case, the SVD gives us the optimal solution
- in the tensor case, need iterative optimization scheme
Alternating Least Squares (ALS)

Fixing all but one factor matrix, we have a linear LS problem:

\[
\min_V \left\| X - \sum_{r=1}^{R} \hat{u}_r \odot \hat{v}_r \odot \hat{w}_r \right\|
\]

or equivalently

\[
\min_V \left\| X_{(2)} - V (\hat{W} \odot \hat{U})^T \right\|_F
\]

\(\odot\) is the Khatri-Rao product, a column-wise Kronecker product or row-wise Hadamard (element-wise) product

ALS works by alternating over factor matrices, updating one at a time by solving the corresponding linear LS problem.
Repeat

1. Solve \( U(V^TV \ast W^TW) = X(1)(W \odot V) \) for \( U \)
2. Solve \( V(U^TU \ast W^TW) = X(2)(W \odot U) \) for \( V \)
3. Solve \( W(U^TU \ast V^TV) = X(3)(V \odot U) \) for \( W \)

Linear least squares problems solved via normal equations using identity \( (A \odot B)^T(A \odot B) = A^T A \ast B^T B \), where \( \ast \) is Hadamard product.

All optimization schemes that compute the gradient must also compute \textit{MTTKRP} in all modes: e.g.,

\[
\frac{\partial f}{\partial V} = V(U^TU \ast W^TW) - X(2)(W \odot U)
\]
MTTKRP bottleneck

- How do we compute MTTKRP efficiently?

- How do we parallelize MTTKRP efficiently?
  - how do we load balance computation?
  - how do we minimize communication?
MTTKRP via Matrix Multiplication

MTTKRP: \[ M = X_{(2)}(W \odot U) \]

Standard approach to MTTKRP for dense tensors

1. “form” matricized tensor (a matrix)
2. compute Khatri-Rao product (a matrix)
3. call matrix-matrix multiplication subroutine

Can we communicate less by exploiting tensor structure? (avoiding forming explicit Khatri-Rao product)
MTTKRP for 3-way Tensors

Matrix equation:

\[ M = X(2)(W \odot U) \]

Element equation:

\[ m_{jr} = \sum_{i=1}^{I} \sum_{k=1}^{K} x_{ijk} u_{ir} w_{kr} \]

Example pseudocode:

```plaintext
for i = 1 to I do
    for j = 1 to J do
        for k = 1 to K do
            for r = 1 to R do
                M(j, r) += X(i, j, k) \cdot U(i, r) \cdot W(k, r)
```
MTTKRP for \( N \)-way Tensors

Matrix equation:

\[ M^{(n)} = X^{(n)}(U^{(N)} \odot \ldots \odot U^{(n+1)} \odot U^{(n-1)} \odot \ldots \odot U^{(1)}) \]

Element equation:

\[ m_{inr}^{(n)} = \sum x_{i_1 \ldots i_N} \prod_{m \neq n} u_{imr}^{(m)} \]

Example pseudocode:

\[
\begin{align*}
&\text{for } i_1 = 1 \text{ to } l_1 \text{ do} \\
&\quad \ldots \\
&\quad \text{for } i_N = 1 \text{ to } l_N \text{ do} \\
&\quad \quad \text{for } r = 1 \text{ to } R \text{ do} \\
&\quad \quad \quad \mathbf{M}^{(n)}(i_n, r) += X(i_1, \ldots, i_N) \cdot U^{(1)}(i_1, r) \cdots U^{(N)}(i_N, r)
\end{align*}
\]
MTTKRP is a set of nested loops that accesses arrays
- Nick’s PhD thesis was “Communication-Optimal Loop Nests"
- References: thesis [Kni15] and paper [CDK+13]

From Nick’s thesis...
- tabulate how the arrays are accessed
- use Hölder-Brascamp-Lieb-type inequality in LB proof
- solve linear program to get tightest lower bound
Lower bound argument follows \cite{cdk+13} almost directly.
Lower bound argument follows [CDK+13] almost directly

Gotcha: the number of nested loops is not constant
  Fixed using a technique similar to one used for tightening the constant in matrix multiplication lower bound [SvdG17]
Lower bound argument follows [CDK$^+$13] almost directly

Gotcha: the number of nested loops is not constant
  - Fixed using a technique similar to one used for tightening the constant in matrix multiplication lower bound [SvdG17]

Gotcha: memory-independent bounds most relevant
  - inspiration from matrix multiplication [BDH$^+$12, DEF$^+$13]
Lower bound argument follows [CDK$^+$13] almost directly

Gotcha: the number of nested loops is not constant
- Fixed using a technique similar to one used for tightening the constant in matrix multiplication lower bound [SvdG17]

Gotcha: memory-independent bounds most relevant
- inspiration from matrix multiplication [BDH$^+$12, DEF$^+$13]

Key assumption: algorithm is not allowed to pre-compute and re-use temporary values
- e.g., forming explicit Khatri-Rao product
- e.g., computing and re-using “partial” MTTKRP
Theorem

Any parallel MTTKRP algorithm involving a tensor with \( I_k = I^{1/N} \) for all \( k \) and that evenly distributes one copy of the input and output performs at least

\[
\Omega \left( \left( \frac{NIR}{P} \right)^{\frac{N}{2^{N-1}}} + NR \left( \frac{I}{P} \right)^{1/N} \right)
\]

sends and receives. (Either term can dominate.)

- \( N \) is the number of modes
- \( I \) is the number of tensor entries
- \( I_k \) is the dimension of the \( k \)th mode
- \( R \) is the rank of the CP model
- \( P \) is the number of processors
Communication-Optimal Parallel Algorithm (3D)

Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix

2. All-Gathers all the rows needed from $\mathbf{U}^{(1)}$
Communication-Optimal Parallel Algorithm (3D)

Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$
Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$
4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)
Communication-Optimal Parallel Algorithm (3D)

Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$
4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)
Communication-Optimal Parallel Algorithm (3D)

Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U(1)$
3. All-Gathers all the rows needed from $U(3)$
4. Computes its contribution to rows of $M(2)$ (local MTTKRP)
5. Reduce-Scatters to compute and distribute $M(2)$ evenly
### Theoretical Comparisons

<table>
<thead>
<tr>
<th>Words (“small” $P$)</th>
<th>Lower Bound</th>
<th>New Algorithm</th>
<th>Standard (MM*)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Omega \left( NR \left( \frac{I}{P} \right)^{1/N} \right)$</td>
<td>$O \left( NR \left( \frac{I}{P} \right)^{1/N} \right)$</td>
<td>$O \left( I^{1/N} R \right)$</td>
</tr>
</tbody>
</table>

- For relatively small $P$ (or small $R$) and even dimensions, parallel “stationary” algorithm attains lower bound
  - same algorithm for sparse [SK16] and dense 3D [LKL+17]
### Theoretical Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Lower Bound</th>
<th>New Algorithm</th>
<th>Standard (MM*)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Words</strong></td>
<td>$\Omega \left( NR \left( \frac{I}{P} \right)^{1/N} \right)$</td>
<td>$O \left( NR \left( \frac{I}{P} \right)^{1/N} \right)$</td>
<td>$O \left( I^{1/N} R \right)$</td>
</tr>
<tr>
<td>“small” $P$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Words</strong></td>
<td>$\Omega \left( \left( \frac{NIR}{P} \right)^\frac{N}{2^N-1} \right)$</td>
<td>$O \left( \left( \frac{NIR}{P} \right)^\frac{N}{2^N-1} \right)$</td>
<td>$O \left( \left( \frac{IR}{P} \right)^{2/3} \right)$</td>
</tr>
<tr>
<td>“large” $P$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- For relatively small $P$ (or small $R$) and even dimensions, parallel “stationary” algorithm attains lower bound
  - same algorithm for sparse [SK16] and dense 3D [LKL+17]
- For larger $P$ (or $R$), then we need more general algorithm to attain lower bound
  - involves communicating the tensor

*communication-optimal matrix multiplication from [DEF+13]
Modeled Communication Costs

Modeled Strong-Scaling Comparison

![Graph showing communication costs across different processors.](image)

- **Matrix Multiplication**
- **Stationary Tensor**
- **General Tensor**

Words Communicated vs. Processors
What about for a full CP-ALS iteration?

A full iteration of CP-ALS includes computing all $N$ MTTKRP.

**Lower Bound**
Lower bound for single MTTKRP applies to computing all $N$.

**Algorithm**
We can compute all $N$ with the same communication as just 1:
- lots of data overlap across MTTKRP.
- More computation required, but not that much more.
Avoiding re-communication across MTTKRP\textsc{s} \\

\textbf{while} not converged \textbf{do} \\
\textbf{for} \( n = 1 \) to \( N \) \textbf{do} \\
\%
\textit{Compute new factor matrix in nth mode} \\
\( \mathbf{M} = \text{Local-MTTKRP}(\mathbf{X}_{p_1 \ldots p_N}, \{ \mathbf{U}^{(i)}_{p_i} \}, n) \) \\
\( \mathbf{M}^{(n)}_{p} = \text{Reduce-Scatter}(\mathbf{M}, \text{PROC-SLICE}(n, p_n)) \) \\
\( \mathbf{S}^{(n)} = \mathbf{G}^{(1)} \ast \ldots \ast \mathbf{G}^{(n-1)} \ast \mathbf{G}^{(n+1)} \ast \ldots \ast \mathbf{G}^{(N)} \) \\
\( \mathbf{U}^{(n)}_{p} = \text{Local-Update}(\mathbf{S}^{(n)}, \mathbf{M}^{(n)}_{p}) \) \\
\%
\textit{Organize data for later modes} \\
\( \mathbf{H} = \mathbf{U}^{(n)\text{T}}_{p} \mathbf{U}^{(n)}_{p} \) \\
\( \mathbf{G}^{(n)} = \text{All-Reduce}(\mathbf{H}, \text{ALL-PROC}\text{S}) \) \\
\( \mathbf{U}^{(n)}_{p_n} = \text{All-Gather}(\mathbf{U}^{(n)}_{p}, \text{PROC-SLICE}(n, p_n)) \) \\

\text{Compute factor matrix, communicate it \textbf{once} for use in all other} \( N-1 \) \text{ modes}
Avoiding recomputation across MTTKRP

We re-use communication and computation across MTTKRP

\[ M^{(1)} = X^{(1)} (U^{(3)} \odot U^{(2)}) \quad \text{and} \quad M^{(2)} = X^{(2)} (U^{(3)} \odot U^{(1)}) \]
Avoiding recomputation across MTTKRP

We re-use communication and computation across MTTKRP

\[
\mathbf{M}^{(1)} = \mathbf{X}_{(1)} \left( \mathbf{U}^{(3)} \odot \mathbf{U}^{(2)} \right) \quad \text{and} \quad \mathbf{M}^{(2)} = \mathbf{X}_{(2)} \left( \mathbf{U}^{(3)} \odot \mathbf{U}^{(1)} \right)
\]

We organize intermediate values in “dimension tree” [PTC13, LCP+17, KU18]

PM = Partial MTTKRP  \quad \text{mTTV} = \text{multi-Tensor-Times-Vector}
Avoiding recomputation across MTTKRPBs

We re-use communication and computation across MTTKRPBs

\[ M^{(1)} = X^{(1)} \left( U^{(3)} \circ U^{(2)} \right) \quad \text{and} \quad M^{(2)} = X^{(2)} \left( U^{(3)} \circ U^{(1)} \right) \]

We organize intermediate values in “dimension tree” [PTC13, LCP+17, KU18]

PM = Partial MTTKRP
mTTV = multi-Tensor-Times-Vector
Uses CP-ALS for non-negative CP problems
- minimize least squares loss function
- use block principal pivoting [KP11] to solve subproblems

Avoids redundant communication across MTTKRP}s

Avoids redundant computation across MTTKRP}s using dimension trees
Strong Scaling Results (3D)

Figure: $1024 \times 1024 \times 1024$ tensor on $2^k \times 2^k \times 2^k$ proc grids ($R = 32$)
Strong Scaling Results (5D)

![Graph showing strong scaling results for different node counts. The graph compares time (s) against the number of nodes (2^0 to 2^5). Two lines are shown: one for DimTree and another for NoDimTree. The graph indicates a decrease in time as the number of nodes increases, following a logarithmic scale.]

Figure: 64 × 64 × 64 × 64 × 64 tensor (R = 32)
Figure: 30,012 × 1200 × 500 tensor on 120 × 6 × 2 proc grid
Varying Rank Results (4D)

Figure: $1344 \times 1024 \times 33 \times 9$ tensor on $8 \times 8 \times 1 \times 1$ proc grid
We establish communication lower bounds for matricized-tensor times Khatri-Rao product (MTTKRP) key kernel for computing CP decomposition

We present optimal parallel dense MTTKRP algorithm
  - attains the lower bound to within constant factors

We implement and benchmark optimal CP-ALS algorithm
  - remains computation bound and scales well
  - dimension tree optimization avoids redundant computation
Brief announcement: strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds.

Communication lower bounds and optimal algorithms for programs that reference arrays - part 1.

Communication-optimal parallel recursive rectangular matrix multiplication.

Nicholas Knight.
Communication-Optimal Loop Nests.
PhD thesis, EECS Department, University of California, Berkeley, Aug 2015.

Jingu Kim and Haesun Park.

Oguz Kaya and Bora Uçar.
Parallel candecomp/parafac decomposition of sparse tensors using dimension trees.

Jiajia Li, Jee Choi, Ioakeim Perros, Jimeng Sun, and Richard Vuduc.
Model-driven sparse CP decomposition for higher-order tensors.
In IEEE International Parallel and Distributed Processing Symposium, IPDPS, pages 1048–1057, May 2017.
A. P. Liavas, G. Kostoulas, G. Lourakis, K. Huang, and N. D. Sidiropoulos.
Nesterov-based alternating optimization for nonnegative tensor factorization: Algorithm and parallel implementation.

Anh-Huy Phan, Petr Tichavsky, and Andrzej Cichocki.
Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations.

Shaden Smith and George Karypis.
A medium-grained algorithm for distributed sparse tensor factorization.
In IEEE 30th International Parallel and Distributed Processing Symposium, pages 902–911, May 2016.

Tyler Michael Smith and Robert A. van de Geijn.
Pushing the bounds for matrix-matrix multiplication.
A tensor can be decomposed into the fibers of each mode (fibers are vectors – fix all indices but one)
A tensor can be reshaped into a matrix, called a matricized tensor or unfolding, for a given mode, where each column is a fiber.
Theorem

For sufficiently large $I$, any sequential MTTKRP algorithm performs at least

$$\Omega \left( \frac{NIR}{M^{1-1/N}} \right)$$

loads and stores to/from slow memory.

- $N$ is the number of modes
- $I$ is the number of tensor entries
- $R$ is the rank of the CP model
- $M$ is the size of the fast memory
Loop over $b \times \cdots \times b$ blocks of the tensor.
Communication-Optimal Sequential Algorithm (3D)

1. Loop over \( b \times \cdots \times b \) blocks of the tensor
2. With block in memory, loop over subcolumns of input factor matrices, updating corresponding subcolumn of output matrix

choose \( b \approx M^{1/N} \)
### Theoretical Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Lower Bound</th>
<th>New Algorithm</th>
<th>Standard (MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flops</td>
<td>-</td>
<td>$NIR$</td>
<td>$2IR$</td>
</tr>
<tr>
<td>Words</td>
<td>$\Omega \left( \frac{NIR}{M^{1-1/N}} \right)$</td>
<td>$O \left( I + \frac{NIR}{M^{1-1/N}} \right)$</td>
<td>$O \left( I + \frac{IR}{M^{1/2}} \right)$</td>
</tr>
<tr>
<td>Temp Mem</td>
<td>-</td>
<td>-</td>
<td>$\frac{IR}{T_n}$</td>
</tr>
</tbody>
</table>
## Theoretical Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Lower Bound</th>
<th>New Algorithm</th>
<th>Standard (MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flops</strong></td>
<td>-</td>
<td>NIR</td>
<td>2IR</td>
</tr>
<tr>
<td><strong>Words</strong></td>
<td>$\Omega \left( \frac{NIR}{M^{1-1/N}} \right)$</td>
<td>$O \left( I + \frac{NIR}{M^{1-1/N}} \right)$</td>
<td>$O \left( I + \frac{IR}{M^{1/2}} \right)$</td>
</tr>
<tr>
<td><strong>Temp Mem</strong></td>
<td>-</td>
<td>-</td>
<td>$\frac{IR}{T_n}$</td>
</tr>
</tbody>
</table>

- New algorithm performs $N/2$ more flops than standard.
- For relatively small $R$, $I$ term dominates communication.
  - We expect this to be the typical case in practice.
- For relatively large $R$, new algorithm communicates less.
  - Better exponent on $M$. 
MTTKRP Loop Nest

\begin{align*}
\text{for } i_1 &= 1 \text{ to } I_1 \text{ do} \\
\quad \ldots \\
\text{for } i_N &= 1 \text{ to } I_N \text{ do} \\
\quad \text{for } r &= 1 \text{ to } R \text{ do} \\
M^{(n)}(i_n, r) &= X(i_1, \ldots, i_N) \times U^{(1)}(i_1, r) \times \cdots \times U^{(N)}(i_N, r)
\end{align*}

\[ \Delta = \begin{bmatrix}
U^{(1)} & 1 & \cdots & i_n & \cdots & i_N & r \\
\vdots & \ddots & \cdots & \vdots \\
M^{(n)} & 1 & \cdots & i_n & \cdots & i_N & r \\
\vdots & \ddots & \cdots & \vdots \\
U^{(N)} & \cdots & \cdots & 1 & \cdots & 1 \\
X & 1 & \cdots & 1 & \cdots & 1
\end{bmatrix} \]