# POSTER: Shared-Memory Parallelization of MTTKRP for Dense Tensors*

Extended Abstract

Koby Hayashi, Grey Ballard, Yujie Jiang, Michael J. Tobia
Wake Forest University
{hayakb13,ballard,jiany14,tobiamj}@wfu.edu

## Abstract

The matricized-tensor times Khatri-Rao product (MTTKRP) is the computational bottleneck for algorithms computing CP decompositions of tensors. In this work, we develop shared-memory parallel algorithms for MTTKRP involving dense tensors. The algorithms cast nearly all of the computation as matrix operations in order to use optimized BLAS subroutines, and they avoid reordering tensor entries in memory. We use our parallel implementation to compute a CP decomposition of a neuroimaging data set and achieve a speedup of up to 7.4× over existing parallel software.

**ACM Reference Format:**
Koby Hayashi, Grey Ballard, Yujie Jiang, Michael J. Tobia. 2018. POSTER: Shared-Memory Parallelization of MTTKRP for Dense Tensors: Extended Abstract. In *PPoPP '18: Principles and Practice of Parallel Programming, February 24–28, 2018, Vienna, Austria.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3178487.3178522

## 1 Introduction

The CP decomposition is a generalization of the matrix singular value decomposition, providing a low-rank approximation of multidimensional data. We use the CP decomposition in a neuroimaging data analysis problem involving functional MRI (fMRI) data. We wish to extract functional brain networks to study how they behave over time relative to a cognitive task and how they relate to and differentiate among subjects [6]. The existing approach uses the Matlab Tensor Toolbox [2], but the computational time is a bottleneck in the analysis process. In order to decrease the time and allow for analysis of larger data sets, our goal is to develop shared-memory parallelizations of the MTTKRP computation.

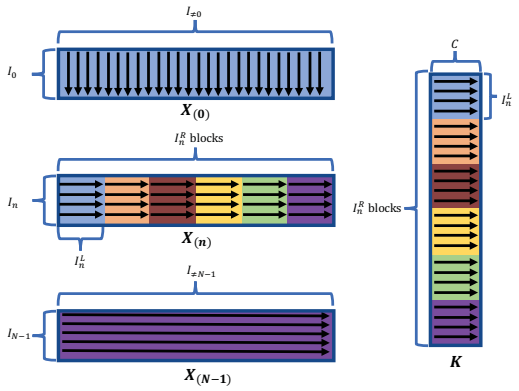



**Figure 1.** Data layout of the matricizations of an $N$-way tensor $\mathcal{X}$. A conformal partitioning of the $n$th-mode Khatri-Rao product $K$ is depicted on the right.

The bulk of MTTKRP corresponds to a single matrix-matrix multiplication. Unfortunately, using the BLAS interface requires that matrices be stored in regular layouts in memory, and it is impossible to choose a dense tensor data layout in memory that is conducive to direct BLAS calls in all cases (see Figure 1). The main task in optimizing MTTKRP is to employ BLAS in a way that avoids tensor reordering.

The primary contributions of this work are as follows: we develop a parallel row-wise algorithm for computing a Khatri-Rao product of multiple matrices; we implement a new 1-step and an existing 2-step MTTKRP algorithm and parallelize the algorithms using a combination of OpenMP and multithreaded BLAS; we demonstrate performance improvement over a baseline approach and achieve parallel speedups of up to 12× and 8× over 12 threads; and we obtain up to a 7.4× speedup over existing parallel software for computing the CP decomposition of fMRI tensors. For more details, see the associated technical report [3].

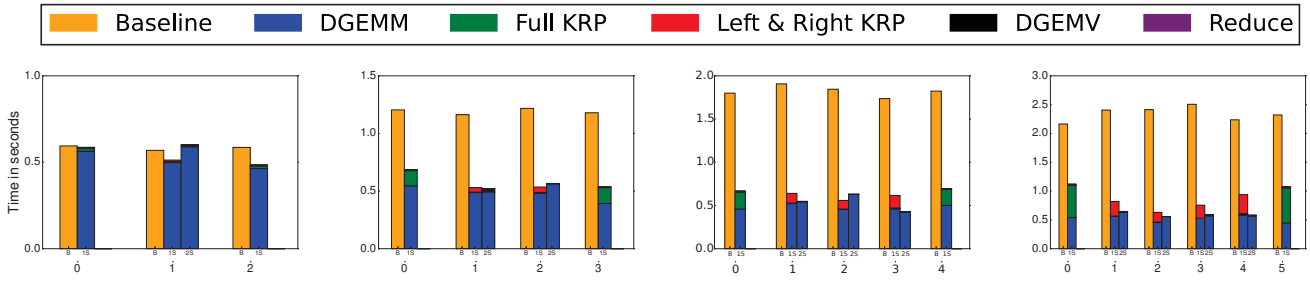
## 2 Algorithms

***Khatri-Rao Product*** Our algorithm for computing a Khatri-Rao product (KRP) works row-wise. The advantages of this

*This material is based upon work supported by the NSF Grant No. ACI-1642385 and a WFU ACC-IAC Creativity and Innovation Fellowship.


*PPoPP '18, February 24–28, 2018, Vienna, Austria*

ACM ISBN 978-1-4503-4982-6/18/02…$15.00
https://doi.org/10.1145/3178487.3178522




**(a)** Par. $N = 3$ **(b)** Par. $N = 4$ **(c)** Par. $N = 5$ **(d)** Par. $N = 6$

**Figure 2.** Time breakdown of 1-step and 2-step MTTKRP (and baseline DGEMM) across modes for varying numbers of modes.

approach are easy vectorization within rows and parallelization across rows. Our main optimization is to store and re-use partially computed Hadamard products across related rows to avoid redundant computation. With this optimization, the performance of our KRP algorithm is comparable to the peak memory bandwidth achieved by the STREAM benchmark.

***1-Step MTTKRP*** The main idea of our 1-Step MTTKRP is to perform the matrix multiplication without reordering tensor entries, using multiple BLAS calls. Our algorithm is based on the observation that given the natural linearization of tensor entries, the $n$th mode matricization can be seen as a contiguous set of submatrices, each of which is stored row-major in memory [1, 4]. Figure 1 shows how $\mathbf{X}_{(n)}$ is ordered in memory, and it also shows how the KRP matrix $\mathbf{K}$ can be conformally partitioned to perform the matrix multiplication as a block inner product. For external modes (0 and $N-1$), only one BLAS call is required. We parallelize over the inner dimension of the multiply for external modes and parallelize over BLAS calls for internal modes.

***2-step MTTKRP*** The 2-step algorithm, developed by Phan *et al.* [5, Section III.B], first performs a *partial MTTKRP* and then finishes the computation with multiple tensor-times-vector operations (*multi-TTV*). The principal observation is that we can first compute the matrix product of the matricization $\mathbf{X}_{(0:n)}$, which assigns modes 0 through $n$ to the rows of the matrix, and a Khatri-Rao product of a subset of the input matrices with a single BLAS (GEMM) call.

The output of a partial MTTKRP is an intermediate quantity which must be combined with the remaining input matrices to obtain the final output. Each column of the output matrix can be computed by a tensor-times-vector (TTV) operation involving the corresponding columns of the input matrices and one sub-tensor of the intermediate quantity, which can be cast as a single BLAS (DGEMV) call.

## 3 Experimental Results

***Time Breakdown*** We compare the performance of 1-step and 2-step algorithms using 12 threads in Fig. 2, noting that the two algorithms are equivalent for external modes. For a baseline, we also compare against the performance of a single BLAS DGEMM call with the same dimensions. We consider $N = \{3, 4, 5, 6\}$, and for each tensor, each dimension is the same and chosen so that the total number of tensor entries is approximately 750 million; we use rank $C = 25$.

Overall, we see that both 1-step and 2-step algorithms outperform the baseline, which we attribute to more efficient parallelization. We also note that the 2-step algorithm is usually faster than the 1-step algorithm, as it spends less time in KRP computations.

***CP-ALS on Neuroscience Data*** We use our MTTKRP implementations for computing CP decompositions on 3D and 4D fMRI data tensors with various ranks, using both sequential and parallel, MATLAB Tensor Toolbox [2] and C implementations. We observe up to a 2× speedup of our sequential implementation over 1-core MATLAB and up to a 7.4× speedup in parallel over 12-core MATLAB.

## References

[1] W. Austin, G. Ballard, and T. G. Kolda. 2016. Parallel Tensor Compression for Large-Scale Scientific Data. In *IPDPS*. 912–922. https://doi.org/10.1109/IPDPS.2016.67

[2] B. W. Bader, T. G. Kolda, et al. 2015. MATLAB Tensor Toolbox Version 2.6. Available online. (February 2015). http://www.sandia.gov/~tgkolda/TensorToolbox/

[3] K. Hayashi, G. Ballard, Y. Jiang, and M. J. Tobia. 2017. *Shared Memory Parallelization of MTTKRP for Dense Tensors.* Technical Report 1708.08976. arXiv. https://arxiv.org/abs/1708.08976

[4] J. Li, C. Battaglino, I. Perros, J. Sun, and R. Vuduc. 2015. An Input-Adaptive and In-Place Approach to Dense Tensor-Times-Matrix Multiply. In *SC (SC '15)*. ACM, New York, NY, USA, Article 76, 12 pages. https://doi.org/10.1145/2807591.2807671

[5] A.-H. Phan, T. Tichavsky, and A. Cichocki. 2013. Fast Alternating LS Algorithms for High Order CANDECOMP/PARAFAC Tensor Factorizations. *IEEE Transactions on Signal Processing* 61, 19 (Oct 2013), 4834–4846. https://doi.org/10.1109/TSP.2013.2269903

[6] M. J. Tobia, K. Hayashi, G. Ballard, I. H. Gotlib, and C. E. Waugh. 2017. Dynamic functional connectivity and individual differences in emotions during social stress. *Human Brain Mapping* 38, 12 (2017), 6185–6205. https://doi.org/10.1002/hbm.23821