CCF Transactions on High Performance Computing (2019) 1:111–130
https://doi.org/10.1007/s42514-019-00012-w

REGULAR PAPER

PASTA: a parallel sparse tensor algorithm benchmark suite

Jiajia Li[1] · Yuchen Ma[2] · Xiaolong Wu[3] · Ang Li[1] · Kevin Barker[1]

Received: 22 January 2019 / Accepted: 18 July 2019 / Published online: 5 August 2019

Abstract
Tensor methods have gained increasingly attention from various applications, including machine learning, quantum chemistry, healthcare analytics, social network analysis, data mining, and signal processing, to name a few. Sparse tensors and their algorithms become critical to further improve the performance of these methods and enhance the interpretability of their output. This work presents a sparse tensor algorithm benchmark suite (PASTA) for single- and multi-core CPUs. To the best of our knowledge, this is the first benchmark suite for sparse tensor world. PASTA targets on: (1) helping application users to evaluate different computer systems using its representative computational workloads; (2) providing insights to better utilize existed computer architecture and systems and inspiration for the future design. This benchmark suite is publicly released at https://gitlab.com/tensorworld/pasta, under version 0.1.0.

Keywords Sparse tensor · Tensor methods · Benchmarking · High performance computing · Sparsity

1 Introduction

Tensors draw increasing attention from various domains, such as machine learning, quantum chemistry, healthcare analytics, social network analysis, data mining, and signal processing, to name a few. Tensor methods have been noted for their ability to discover multi-dimensional inherent relationships from underlying application logic. A tensor is a multi-dimensional array, generalized matrices and vectors to more dimensions. In data-oriented tensor applications (Chi and Kolda 2012; Henderson et al. 2017; Ho et al. 2014b; Papalexakis and Sidiropoulos 2011; Sidiropoulos et al. 2017), sparse tensors are often found, where most of its entries are zeros.

High-performance computing (HPC) now enters the era of extreme heterogeneity. As many general purpose accelerators, such as Graphics Processing Unit (GPUs), Intel Xeon Phi, and Field-Programmable Gate Array (FPGAs), and domain-specific architectures, such as near-memory, thread migratory architecture Emu Hein et al. (2018) and Google Tensor processing unit (TPU) Jouppi et al. (2017), emerge, it is natural to ask whether the critical sparse-tensor based algorithms can be efficiently executed on these platforms, with their non-regular parallelism to be effectively exploited. However, the lack of a concrete, comprehensive, and easy to use sparse tensor algorithm benchmark suite prevents us from answering this question easily.

In this paper, we fill this gap by proposing a PArallel Sparse Tensor Algorithm benchmark suite called PASTA. PASTA incorporates various sparse tensor algorithms and operations, serving as a handy tool for application developers to assess different platforms, in terms of their tensor processing capability. Consisting state-of-the-art sequential and parallel versions, while adopting the most popular sparse tensor format COO, PASTA can also supply a fair baseline for evaluating performance improvement brought by new sparse tensor methods. Application developers seeking to exploit tensor sparsity for further performance speedup may also find it useful as a good reference.

This paper makes the following contributions:

✉ Jiajia Li
jiajia.li@pnnl.gov

Yuchen Ma
m13253@hotmail.com

Xiaolong Wu
wu1565@purdue.edu

Ang Li
ang.li@pnnl.gov

[1] Pacific Northwest National Laboratory, 902 Battelle Blvd, Richland, WA 99354, USA

[2] Hangzhou Dianzi University, Hangzhou 310005, China

[3] Purdue University, West Lafayette, IN 47907, USA

- We show the importance of sparse tensor operations and tensor methods in diverse tensor applications (Sect. 3)
- We extract 12 computational sparse tensor operations as PASTA workloads: Tensor Element-Wise operations–Tew-eq (addition/subtraction/multiplication/division) and Tew (addition/subtraction/multiplication), Tensor-Scalar operations–Ts addition/multiplication, Tensor-Times-Vector operation (Ttv), Tensor-Times-Matrix operation (Ttm), and Matricized Tensor Times Khatri-Rao Product (Mttkrp) (Sect. 4)
- We implement sequential and multicore parallel algorithms for all workloads, based on the most popular coordinate (COO) sparse tensor format. Our experiments and analysis show the usefulness of PASTA on single- and multi-core CPUs (Sects. 5, 6, 7)

2 Motivation

This work is motivated by first demonstrating the challenges of sparse tensor algorithms and then illustrating that existed libraries or toolsets cannot meet the requirements of a benchmark suite from diversity, timeliness, research support, and dataset four aspects.

2.1 Challenges of sparse tensor algorithms

We summarize the challenges of sparse tensor algorithms into five points:

The curse of dimensionality refers to the issue that the number of entries of an intermediate or output tensor can grow exponentially with the tensor order, resulting in significant computational and storage overheads. Even when the tensor is structurally sparse, meaning it consists mostly of zero entries, the execution time of one important tensor method, CANDECOMP/PARAFAC decomposition introduced in Sect. 3.1, generally grows quadratically with the number of non-zeros (Bader and Kolda 2007; Bader et al. 2017). And there is an increasing interest in applications involving a large number of dimensions De Lathauwer et al. (2017), Lebedev et al. (2014), Novikov et al. (2015), which makes this problem more difficult.

Mode orientation refers to the issue of a particular storage format favoring the iteration of tensor modes in a certain sequence, which is of particular concern in the sparse case. Since most methods of interest require more than one sequence, being efficient for every sequence generally requires storing the tensor in multiple formats, thereby trading extra memory for speed. A question arises, that is whether one can achieve both a neutral mode orientation and compact storage which also helps reduce memory footprint.

Tensor transformation(s) refers to a common pattern for attaining speed in some implementations of tensor algorithms, which starts by reorganizing the tensor into a matrix and then perform equivalent matrix operations using highly tuned linear algebra libraries. Done naïvely, this approach appears to require an extra memory copy, which can even come to dominate the overall running time. We observe instances in which such a copy consumes 70% or more of the total running time (in the case of a Ttm operation).

Irregularity refers to two issues. The first is that a tensor may have dimension sizes that vary widely; the second is that a sparse tensor may have an irregular non-zero pattern, resulting in irregular memory references.

Arbitrary tensor orders generate various implementations of a tensor operation. For the sake of performance, programmers usually implement and optimize third-order tensor algorithms apart from higher-order ones. These implementations makes no one optimization method can fit all variations, e.g., different number of loops and diverse memory access behavior.

These challenges bring non-trivial computational and storage overheads, and some of them are even harder to overcome than their counterparts in classical linear algebra. To overcome these challenges, it is necessary to build a sparse tensor benchmark suite to evaluate diverse algorithms and computer systems.

2.2 Requirements for a benchmark suite

By surveying some benchmark suites (Bienia et al. 2008; Che et al. 2009; Dixit 1991; KleinOsowski and Lilja 2002; Lee et al. 1997; Poovey et al. 2009; Wang et al. 2014a) we present the following four requirements for a benchmark suite.

Diversity. We analyze diversity from two aspects: *application diversity* and *platform diversity*. *Application diversity* means a benchmark suite should represent a broad and representative applications. For example, EEMBC benchmark suite Poovey et al. (2009) is developed for autonomous driving, mobile imaging, the Internet of Things, mobile devices, and many other applications; PARSEC benchmark suite Bienia et al. (2008) covers computer vision, video encoding, financial analytics, animation physics and image processing, etc.. Sparse tensor methods have a broad application domains (refer to Sect. 3.2), the workloads in our benchmark suite also need to represent the diversity of these domains. *Platform diversity* is that a benchmark suite should support different computer architectures and platforms, especially the emerging ones. For example, SPEC benchmarks Dixit (1991) supports scientific applications on diverse platforms: CPUs, distributed platforms, accelerators, web servers, cloud platforms, etc. A recent Tartan benchmark Li et al. (2018b) collected kernels from machine learning, data analysis, high performance simulation, molecular dynamics and so on and optimized them on multi-GPU platforms.

Timeliness. A benchmark suite should be kept updated by including the state-of-the-art *data structures*, *algorithms*, and *optimization techniques*. Especially for sparse data, the data structure is closely relevant to the performance of its algorithm. This phenomenon has been observed from sparse matrices, where different sparse formats behave quite differently on diverse input matrices (Li et al. 2013; Sedaghati et al. 2015; Su and Keutzer 2012; Zhao et al. 2018). As mentioned in the work (Bienia et al. 2008), an outdated algorithm cannot well reflect the current status of an application. This can easily mislead the researchers using this benchmark suite to test a machine's behavior. As the computer architectures keep evolving, an under-optimized code, e.g., sequential benchmark programs for a multicore machine, cannot be a fair measurement. Optimized implementations for architectures have to be taken account.

Research support. Research support also includes two aspects: support of domain research and benchmarked workload research. The former requires a benchmark suite to be *compatible*, while the latter requires it to be *extensible*. Since some workloads are still open research problems in an application domain, a compatible workload should be able to do easy comparison with other research work by supporting unified input/output format and interface to high-level applications. The workload research mainly develops its high performance, power or other efficiency. An extensible workload is easy to be assembled with new data structures, algorithms, and optimization techniques.

Dataset. Data becomes essential to data-intensive applications and their workloads which widely exist in real world. Traditionally, two types of dataset are considered: *synthetic* and *real* data. Real data comes directly from real-world applications, which can best reflects the application features. However, due to some factors such as information protection, sensitive data, etc., researchers are usually short of data. Thus, synthetic data are generated according to some regulations and scenarios from applications.

2.3 PASTA in need

Some tensor libraries or toolsets have existed for sparse tensor algorithms. The most popular libraries are Tensor Toolbox (Bader et al. 2017) and TensorLab (Vervliet et al. 2016). They are both implemented using MATLAB. The main shortcoming is that these two libraries are hard to be implemented on various platforms, such as multicore CPUs and GPUs, which violates the platform diversity requirement. Besides, their performance efficiency is low because of MATLAB environment. Recently, many other highly performance efficient libraries emerge, such as SPLATT (Smith et al. 2015), Cyclops Tensor Framework (CTF) (Solomonik and Hoefler 2015), DFacTo (Choi and Vishwanathan 2014), GigaTensor (Kang et al. 2012), HyperTensor (Kaya and Uçar 2015), GenTen (Phipps and Kolda 2018), ParTI (Li et al. 2018a), to name a few. However, these libraries are specific to one or two particular sparse tensor operations, this violates the application diversity requirement. Beyond these, the requirements of timeliness, research support, and dataset are barely met by these libraries. Our PASTA is proposed to meet all the requirements from our continuous effort.

Table 1 The relationship between tensor domains, tensor methods, and workloads

Domains	Tensor methods	Workloads
Machine learning	Cpd, Tpm, tucker, TT, hTucker	Ts, Mttkrp, Ttv, Ttm, Ttt
Healthcare analytics	Cpd	Mttkrp
Social network analysis	Cpd, tucker	Ttm
Quantum chemistry	Cpd, tucker	Ts, Tew, Ttm, Mttkrp, Ttt
Brain signal analysis	Cpd	Mttkrp
Personalized web search	Cpd, tucker	Mttkrp, Ttm
Recommendation systems	Cpd, tucker	Mttkrp, Ttm
Signal processing	Cpd	Mttkrp
Direct numerical simulation	Tucker	Ttm
Power grid	Cpd, tucker	Mttkrp, Ttm

3 Tensor methods and applications

This section describes the broad applications of tensors methods in diverse domains, along with the tensor methods and their computational operations. The summarized form is presented in Table 1.

3.1 Tensor methods

In this section, we summarize tensor methods in three categories: tensor decompositions, tensor network models, and tensor regression. Though tensor network models also belong to tensor decomposition methods, because of their network format and more emphasizing on high-order tensors, we discuss them separately.

3.1.1 Tensor decompositions

We introduce three low-rank tensor decompositions which have applications for sparse data.

Cpd **.** The CANDECOMP/PARAFAC decomposition (Cpd) was first introduced in 1927 by Hitchcock (Hitchcock 1927), and independently introduced by others (Carroll and Chang 1970; Harshman 1970). Cpd decomposes an Nth-order tensor into a sum of component rank-one tensors with different weights (Kolda and Bader 2009). In a low-rank approximation, a tensor rank R is chosen to be a small number less than 100. From a data science standpoint, the results can be interpreted by viewing the tensor as being composed of R latent rank-1 factors. Cpd has proven both scalable and effective in many applications in Sect. 3.2.

Other variants of Cpd exist by restructuring of the factors or their constraints to accommodate diverse situations, such as INDSCAL (Carroll and Chang 1970), CANDELINC (Carroll et al. 1980), PARAFAC2 (Harshman 1972; Perros et al. 2017), and DEDICOM (Harshman 1970). Many Cpd methods have been proposed in a broad area of research, such as Alternating Least Squares (ALS) based methods (Harshman 1970; Karlsson et al. 2016; Kaya and Uçar 2015; Kolda and Bader 2009), block coordinate descent (BCD) based methods (Li et al. 2015; Mohlenkamp 2010), Gradient Descent based methods (Beutel et al. 2013; Ravindran et al. 2014; Smith et al. 2016; Sorber et al. 2013), quasi-Newton and Nonlinear Least Squares (NLS) based methods (Chi and Kolda 2012; Hansen et al. 2015; Ishteva et al. 2011; Savas and Lim 2010; Sorber et al. 2013; Tomasi and Bro 2006; Wright and Nocedal 1999), alternating optimization (AO) with the alternating direction method of multipliers (ADMM) based methods (Boyd et al. 2011; Smith et al. 2017a), exact line search based methods (Rajih and Comon 2005; Sorber et al. 2016), and randomized/sketching methods (Battaglino et al. 2018; Cheng et al. 2016; Papalexakis et al. 2012; Reynolds et al. 2016; Song et al. 2016; Vervliet and Lathauwer 2016). Sparse Cpd comes from two aspects: the sparse tensor from applications (Bader and Kolda 2007; Chi and Kolda 2012; Choi et al. 2018; Choi and Vishwanathan 2014; Kang et al. 2012; Kaya and Uçar 2018; Kolda and Bader 2009; Li 2018; Li et al. 2017, 2018c; Liu et al. 2017; Phipps and Kolda 2018; Ravindran et al. 2014; Sidiropoulos et al. 2017; Smith and Karypis 2016; Smith et al. 2017c, 2015) and the constrained sparse factors from some Cpd models (Henderson et al. 2017; Ho et al. 2014b; Papalexakis and Sidiropoulos 2011).

The computational bottleneck of Cpd is the matriced tensor-times-Khatri-Rao product (Mttkrp) (will be described in Sect. 4.6).

Tucker. Tucker decomposition, first introduced by Ledyard R. Tucker Tucker (1966), provides a more general decomposition. It decomposes an Nth-order tensor into a small-sized Nth-order core tensor along with N factor matrices that are all orthogonal. The core tensor models a potentially complex pattern of mutual interaction between tensor modes. Its size is determined by N ranks which can be chosen according to the work (Kiers and der Kinderen 2003). In a low-rank approximation, the rank sizes are usually less than 100.

Some variants of Tucker decomposition are PARATUC K2 (Harshman and Lundy 1996), lossy Tucker decomposition (Zhou et al. 2014), and so on. Methods for Tucker decomposition include higher-order SVD (HOSVD) (De Lathauwer et al. 2000a), truncated HOSVD (De Lathauwer et al. 2000a), Alternating Least Squares (ALS) based methods (Kapteyn et al. 1986), the popular higher-order orthogonal iteration (HOOI) (De Lathauwer et al. 2000b, Newton–Grassmann optimization (Eldén and Savas 2009. Sparse Tucker also comes from two aspects: the sparse tensor from applications (Li 2018; Liu et al. 2017; Ma et al. 2018; Smith and Karypis 2017), and the constrained sparse factors.

The computational tensor kernel of Tucker decomposition is the Tensor-Times-Matrix operation (Ttm) (will be described in Sect. 4.4).

Tpm **.** Tensor power method (Anandkumar et al. 2014; De Lathauwer et al. 2000b), is an approach for orthogonal tensor decomposition, which decomposes a symmetric tensor into a collection of orthogonal vectors with corresponding positive scalars as weights. Some variations have been proposed (Anandkumar et al. 2014; Yu et al. 2017). When the tensor is sparse, we need to use sparse method correspondingly.

The computational tensor kernel of tensor power method is the Tensor-Times-Vector operation (Ttv) (will be described in Sect. 4.3).

3.1.2 Tensor network models

Cpd and Tucker decompositions assume a model in which all modes interact with all the other modes, which ignores the situations where modes could interact in subgroups or hierarchies. Tensor network models decompose a tensor in tensor networks which expose more localized relationships between modes. Tensor networks have flexibility in modeling and compute/storage efficiency especially for high-order tensors.

TT **.** Tensor Train (TT) decomposition, also called Matrix Product State (MPS) in quantum physics community (Cichocki et al. 2016; Grasedyck et al. 2013), was first proposed by Ivan Oseledets in the work (Oseledets 2011). TT decomposes a high-order tensor into a linear sequence of tensor-times-tensor/matrix products. The contraction modes are in small rank sizes in low-rank approximation.

The variants of TT include tensor chain (TC), tensor networks with cycles: Projected Entangled Pair States (PEPS) (Orús 2014), Projected Entangled Pair Operators (PEPO) (Evenbly and Vidal 2009), Honey–Comb Lattice (HCL) (Giovannetti et al. 2008), Multi-scale Entanglement Renormalization Ansatz (MERA) (Orús 2014).

The computational tensor kernels of TT are the Tensor-Scalar (Ts), Tensor-Times-Matrix (Ttm) and Tensor-Times-Tensor (Ttt) operations. Ts and Ttm will be described in Sects. 4.2 and 4.4 respectively, and Ttt will be one of our future work.

hTucker. Hierarchical Tucker (hTucker) decomposition, also called hierarchical tensor representation, was introduced in Cichocki et al. (2016), Grasedyck (2010), Grasedyck et al.

(2013), Hackbusch and Kühn (2009). hTucker recursively splits the set of tensor modes, resulting a binary tree containing a subset of modes at each node. This binary tree is called dimension tree, and the modes from different nodes do not overlap. TT decomposition is a special case of hTucker while the dimension tree is linear and extremely unbalanced.

Variants of hTucker include the Tree Tensor Network States (TTNS) model (Nakatani and Chan 2013), multi-layer multi-configuration time-dependent Hartree method (ML-MCTDH) (Wang and Thoss 2003). Sparsity has been considered by Perros et al. (2015) in the work.

The computational tensor kernels of hTucker are the Tensor-Scalar (Ts), Tensor-Times-Matrix (Ttm) and Tensor-Times-Tensor (Ttt) operations. Ts and Ttm will be described in Sects. 4.2 and 4.4 respectively, and Ttt will be one of our future work.

3.1.3 Tensor regression

Tensor regression is an extension of classical regression model, but using tensors to represent input and covariates data. Tensor regression approximates coefficient tensor with a low-rank decomposition, thus tensor decomposition methods introduced above can be easily adopted here. Some tensor regression methods have been proposed (Romera-Paredes et al. 2013; Signoretto et al. 2014; Wimalawarne et al. 2014; Yu and Liu 2016; Yu et al. 2017; Zhao et al. 2011; Zhou et al. 2013).

3.2 Tensor applications

Tensor methods can be used in applications to expose the inherent relationship in the observed data and to represent the data in a more compressed way. This section does not keen to give a thorough survey of tensor applications but emphasizes on showing the broad application scenarios tensor methods can be applied and useful in. Please refer to these surveys for more complete tensor applications (Anandkumar et al. 2014; Cichocki 2014; Cichocki et al. 2016, 2015; De Lathauwer 2008; Kolda and Bader 2009; Sidiropoulos et al. 2017).

3.2.1 Machine learning

The diversity needs of machine learning algorithms have promoted the exploitation of various tensor-based decompositions, regressions, and techniques from this community. Cpd, Tucker and TT decompositions have been leveraged in the context of neural networks (Hutchinson et al. 2013; Janzamin et al. 2015; Lebedev et al. 2014; Novikov et al. 2018, 2015; Setiawan et al. 2015; Socher et al. 2013; Yu et al. 2012, 2018), with the weight matrix of a fully-connected layer or a convolutional layer stored compressedly in a low-rank tensor, thus reducing redundancies in the network parameterization. As concerns improving theoretical aspects and understanding of deep neural networks through tensors, Cohen et al. (2015) analyzed the expressive power of deep architectures by drawing analogies between shallow networks and the rank-1 Cpd, as well as between deep networks and the hTucker decomposition. Novikov et al. applied TT in Google's TensorFlow (Abadi et al. 2015; Novikov et al. 2018), which expresses a wide variety of algorithms as operators (graph nodes) that communicate tensor objects through the graph's edges. Other Machine Learning applications include using TT to improve Markov Random Field (MRF) inference problem (Novikov et al. 2014) and extending standard Machine Learning algorithms such as Support Vector Machines and Fisher discriminant analysis to handle tensor-based input (Tao et al. 2007).

3.2.2 Healthcare analytics

The work on tensor-based healthcare data analysis has been driven by the need of improving the interpretability and the robustness of underlying methods, with the goal that healthcare professionals may eventually use consulting tools based on these methods. As a result, recent work has focused on modifying traditional tensor methods like Cpd by adding constraints that better describe the underlying data and exploit domain knowledge. One particular focus is handling *sparsity*, which is particularly important when handling event-recording tensors describing healthcare data (Ho et al. 2014c, a, b; Matsubara et al. 2014; Perros et al. 2015; Wang et al. 2015; Zhou et al. 2013).

3.2.3 Social network analysis

Some studies have been done on DBLP authorship data Papalexakis et al. (2013) by using dynamic/static tensor analysis (include Cpd, Tucker decompositions and their variants) to demonstrate clustering (Kolda and Sun 2008; Sun et al. 2006), find interesting events (or anomalies) in the users' social activities (Papalexakis et al. 2012, 2015). Jiang et al. identified patterns in human behavior through a dynamic tensor decomposition of user interactions within a microblogging service (Jiang et al. 2014). Sun et al. demonstrated a sampling-based Tucker decomposition (Sun et al. 2009), to jointly model the sender-recipient interaction and share content within business networks. The work in Benson et al. (2015) utilizes tensors to model higher-order structures, such as cycles or feed-forward loops in a graph clustering framework.

3.2.4 Quantum chemistry

Tensors have a long history in quantum chemistry because of the nature of high-dimensional data there (Khoromskaia and Khoromskij 2018). Hartree–Fock (HF) is a method of approximation for the energy of a quantum many-body system and large-scale electronic structure calculations. Koppl et al. proposed sparsity using local density fitting in Hartree–Fock calculations, which heavily involves Ttt and Ttm operations (Köppl and Werner 2016). Lewis et al. introduced a clustered low-rank tensor format to exploit element and rank sparsities (Lewis et al. 2016). Block sparsity has been utilized in coupled-cluster singles and doubles (CCSD) in the work (Calvin and Valeev 2016; Epifanovsky et al. 2013; Kaliman and Krylov 2017; Manzer et al. 2017; Peng et al. 2016). Scaled opposite spin second order Møller-Plesset perturbation theory (SOS-MP2) method uses tensor hypercontraction (Thc), approximating a electron Coulomb repulsion integrals (ERI) tensor by decomposing into lower order tensors, with sparsity (Song and Martínez 2016).

3.2.5 Data mining

Tensor decompositions have become a standard approach in brain signal analysis due to multiple heterogeneous data sources. Some recent methods have been surveyed in Cao et al. (2015), Cichocki (2013). Electroencephalogram (EEG) and fMRI data are treated as tensors and analyzed by different tensor decompositions (e.g., Cpd) to study the structure of epileptic seizures (Acar et al. 2007, 2011a), better understand the active brain regions and their behavior (Davidson et al. 2013), (Latchoumane et al. 2012), do feature selection (Cao et al. 2014), and model neuroimaging data (Mørup et al. 2008). BrainQ is a widely available tensor dataset consisting of a sparse tensor with (subject, brain-voxel, noun) as dimensions and a matrix (noun, properties), which are measured from brain activity where individual subjects are shown nouns. Factorizing this is known as a coupled factorization (Acar et al. 2011b), and Papalexakis et al. demonstrated a scalable method using random sampling (Papalexakis et al. 2014). On the supervised learning setting, Wang et al. used fMRI data and adapted the Sparse Logistic Regression to accept tensor input that consequently avoided the loss of correlation information among different orders (Wang et al. 2014b).

Personalized web search tailors the results of a search query for a particular user by utilizing the click history of this user's previous search results. Researchers constructed tensors from (user, query, webpage) information and used Cpd (Kolda and Bader 2006) and Tucker decompositions (Sun et al. 2005) to tackle this problem.

Recommendation systems have also found tensor methods effective to resolve overloaded tags. Some approaches have been explored using Cpd and Tucker decompositions and their variants on collaborative filtering (Xu et al. 2006), a tag-recommendation engine (Karatzoglou et al. 2010), (Rendle et al. 2009), (Symeonidis et al. 2008), personalized tags (Fang and Pan 2014), and sparse international relationships (Schein et al. 2015).

3.2.6 Signal processing

There has been an extensive research from the Signal Processing community, which examines theoretical aspects of tensor methods (Jiang and Sidiropoulos 2004) such as identifiability, or improves existing decompositions (Bro et al. 1999; Sidiropoulos et al. 2000). A tutorial addressing signal processing applications can be found in Cichocki et al. (2015). Please refer to the survey Sidiropoulos et al. (2017) for more complete applications in signal processing.

3.2.7 Other areas

The usage of tensors and tensor decompositions as tools facilitating the extraction of useful information out of complex data is not limited to the categories mentioned above. For example, Benson, et al. used Tucker decomposition to compress scientific data obtained by Direct Numerical Simulation (DNS) (Austin et al. 2016). Song et al. applied Cpd to forecast of the power demand and detect anomalies in smart electrical grid (Song et al. 2017). A variant of Tucker decomposition was used in AC optimal power flow in the work (Oh 2016). TT was used in the hierarchical uncertainty quantification to reduce the computational cost of circuit simulation (Zhang et al. 2015). Electronic design automation (EDA) problems employed Cpd, Tucker, and TT decompositions to ease the suffer of the curse of dimensionality (Zhang et al. 2017). Motion control problems in the context of robotics took TT into consider for its compressed representations (Gorodetsky et al. 2008).

4 Benchmark workloads

This section we describe the workloads in PASTA, which includes element-wise addition/subtraction/multiplication/division, tensor-scalar, tensor-times-vector, tensor-times-matrix, and tensor-times-matrix sequence operations. We referred to the surveys (Anandkumar et al. 2014; Cichocki 2014; Cichocki et al. 2016, 2015; De Lathauwer 2008; Kolda and Bader 2009; Sidiropoulos et al. 2017) and papers Li (2018) for these definitions.

A tensor, abstractly defined, is a function of three or more indices. In computational data analytics, one may regard a tensor as a multidimensional array, where each of its dimensions is also called a *mode* and the number of dimensions or modes is its *order*. For example, a scalar is a tensor of order 0; a vector is a tensor of order 1; and a matrix, order 2, with two modes (its rows and its columns). Notationally, we represent tensors as calligraphic capital letters, e.g., $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$; matrices by boldface capital letters, e.g., $\mathbf{U} \in \mathbb{R}^{I \times J}$; vectors by boldface lowercase letters, e.g., $\mathbf{x} \in \mathbb{R}^{I}$; and scalars by lowercase letters, such as x_{ijk} for the

(i, j, k) element of a third-order tensor $\mathcal{X}$. A *slice* is a two-dimensional cross-section of a tensor, achieved by fixing all mode indices but two, e.g., $\mathbf{S}_{::k} = \mathcal{X}(:, :, k)$ in MATLAB notation. A *fiber* is a vector extracted from a tensor along some mode, selected by fixing all indices but one, e.g., $\mathbf{f}_{:jk} = \mathcal{X}(:, j, k)$.

A tensor can be reshaped to a matrix, which is called matricization. For a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$, its matricized tensor along with mode-n is $\mathbf{X}_{(n)} \in \mathbb{R}^{I_1 \cdots I_{n-1} I_{n+1} \cdots I_N \times I_n}$. A matrix can be also reshaped to a tensor by splitting one mode into two or more.

4.1 Tensor element-wise operations

Tensor element-wise (TEW) operations include addition, subtraction, multiplication, and division operations, which are applied to every corresponding pair of elements from two tensor objects if they have the same order and shape (dimension sizes). For example, element-wise tensor addition of $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is $\mathcal{Z} = \mathcal{X} .+ \mathcal{Y}$, where

$$z_{i_1 \ldots i_N} = x_{i_1 \ldots i_N} + y_{i_1 \ldots i_N}. \tag{1}$$

Similarly for element-wise tensor subtraction $\mathcal{Z} = \mathcal{X} .- \mathcal{Y}$, multiplication $\mathcal{Z} = \mathcal{X} .* \mathcal{Y}$, and division $\mathcal{Z} = \mathcal{X} ./ \mathcal{Y}$. When the two input tensors have exactly the same non-zero distribution, element-wise operations can be easily implemented by iterating all non-zeros of the two sparse tensors and doing the corresponding operation for each element. The tricky cases are when the non-zero patterns of tensors $\mathcal{X}$ and $\mathcal{Y}$ are different and even worse they could be in different shapes. For these two cases, we cannot easily predict the output tensor $\mathcal{Z}$'s storage space before computation. These two cases we use dynamic vectors and an optimization strategy for parallel algorithms.

4.2 Tensor-scalar operations

A Tensor-Scalar (TS) operation is the addition (TSA) /subtraction (TSS) /multiplication (TSM) /division (TSD) of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_N}$ with a scalar $s \in \mathbb{R}$ for every non-zero entry. For example, the TSM operation, denoted by $\mathcal{Y} = \mathcal{X} \times s$, is defined as

$$y_{i_1 \ldots i_{n-1} r i_{n+1} \ldots i_N} = s \times x_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N}. \tag{2}$$

Since $\mathcal{Y} = \mathcal{X} \times s$ is the same with $\mathcal{Y} = \mathcal{X} / s^{-1}$ and $\mathcal{Y} = \mathcal{X} + s$ is the same with $\mathcal{Y} = \mathcal{X} - (-s)$, so implementing TSA and TSM is enough.

4.3 Tensor-times-vector operation

The tensor-times-vector (TTV) in mode n is the multiplication of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$ with a vector $\mathbf{v} \in \mathbb{R}^{I_n}$, along mode n, and is denoted by $\mathcal{Y} = \mathcal{X} \times_n \mathbf{v}$. This results in a $I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N$ tensor which has one less dimension. Its operation is defined as

$$y_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} v_{i_n}. \tag{3}$$

4.4 Tensor-times-matrix operation

The tensor-times-matrix (TTM) in mode n, also known as the n-mode product, is the multiplication of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$ with a matrix $\mathbf{U} \in \mathbb{R}^{I_n \times R}$, along mode n, and is denoted by $\mathcal{Y} = \mathcal{X} \times_n \mathbf{U}$.[1] This results in a $I_1 \times \cdots \times I_{n-1} \times R \times I_{n+1} \times \cdots \times I_N$ tensor, and its operation is defined as

$$y_{i_1 \ldots i_{n-1} r i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} u_{i_n r}. \tag{4}$$

TTM is a special case of tensor contraction. We consider TTM specifically because of its more common usage in tensor decompositions for data analysis, such as the Tucker decomposition. Also, note that R is typically much smaller than I_n in such decompositions, and typically $R < 100$.

TTM is also equivalent to a matrix-matrix multiplication in the following form:

$$\mathcal{Y} = \mathcal{X} \times_n \mathbf{U} \quad \Leftrightarrow \quad \mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)}. \tag{5}$$

Therefore, one feasible way to implement an TTM is to first matricize the tensor, then use an optimized matrix-matrix multiplication to compute the matricized output $\mathcal{Y}$, and, finally, tensorize to obtain $\mathcal{Y}$. However it has the tensor-matrix transformation as the extra overhead and does not work well for sparse tensors.

4.5 Kronecker and Khatri-Rao products

Kronecker and Khatri-Rao products are both matrix products. The *Kronecker product* generalizes the outer product for matrices. Given $\mathbf{U} \in \mathbb{R}^{I \times J}$ and $\mathbf{V} \in \mathbb{R}^{K \times L}$, the Kronecker product $\mathbf{U} \otimes \mathbf{V} \in \mathbb{R}^{IK \times JL}$ is

$$\mathbf{U} \otimes \mathbf{V} = \begin{bmatrix} u_{11}\mathbf{V} & u_{12}\mathbf{V} & \cdots & u_{1J}\mathbf{V} \\ u_{21}\mathbf{V} & u_{22}\mathbf{V} & \cdots & u_{2J}\mathbf{V} \\ \vdots & \vdots & \ddots & \vdots \\ u_{I1}\mathbf{V} & u_{I2}\mathbf{V} & \cdots & u_{IJ}\mathbf{V} \end{bmatrix} \tag{6}$$

The *Khatri-Rao product* is a "matching column-wise" Kronecker product between two matrices with the same number

[1] Our convention for the dimensions of $\mathbf{U}$ differs from that of Kolda and Bader's definition (Kolda and Bader 2009). In particular, we transpose the matrix modes $\mathbf{U}$, which leads to a more efficient TTM under the row-major storage convention of the C language.

of columns. Given matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$ and $\mathbf{B} \in \mathbb{R}^{J \times R}$, their Khatri-Rao product is denoted by $\mathbf{A} \odot \mathbf{B} \in \mathbb{R}^{(IJ) \times R}$,

$$\mathbf{A} \odot \mathbf{B} = \left[\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \dots, \mathbf{a}_R \otimes \mathbf{b}_R\right], \tag{7}$$

where $\mathbf{a}_r$ and $\mathbf{b}_r$, $r = 1, \dots, R$, are columns of $\mathbf{A}$ and $\mathbf{B}$.

Kronecker and Khatri-Rao products appear frequently in tensor decompositions that are formulated as matrix operations. However, such formulations typically also require redundant computation or extra storage to hold matrix operands, so in practice these operations are tend to be not implemented directly but rather integrated into tensor operations.

4.6 Tensor-times-matrix sequence operation

There are two types of tensor-times-matrix sequence operations, Ttm chain and Mttkrp. Ttm chain is a sequence of Ttm operations with one's output as the next one's input. An alternative way to think Ttm chain is a matriced tensor times the Kronecker product of matrices. Mttkrp, matricized tensor times Khatri-Rao product, is a matricized tensor times the Khatri-Rao product of matrices. For an *Nth*-order tensor $\mathcal{X}$ and given matrices $\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(N)}$, the mode-$n$ Mttkrp is

$$\begin{aligned} \tilde{\mathbf{U}}^{(n)} &= \mathbf{X}_{(n)} \left(\odot_{i=1,\dots,N}^{i \neq n} \mathbf{U}_i \right) \\ &= \mathbf{X}_{(n)} \left(\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(n+1)} \odot \mathbf{U}^{(n-1)} \odot \cdots \odot \mathbf{U}^{(1)} \right), \end{aligned} \tag{8}$$

where $\mathbf{X}_{(n)}$ is the mode-n matricization of tensor $\mathcal{X}$, $\odot$ is the Khatri-Rao product.

4.7 Others

We also provide the transformation between tensors and matrices and some sorting algorithms for sparse tensors.

Fig. 1 COO format of an example $4 \times 4 \times 3$ tensor

i	j	k	val
0	0	0	1
0	1	0	2
1	0	0	3
1	0	2	4
2	1	0	5
2	2	2	6
3	0	1	7
3	3	2	8

5 Data structures, algorithms, and implementations

5.1 Data structures

Since COO Kolda and Bader (2009) is the simplest and arguably de facto standard way to store a sparse tensor, and it is mode generic, we only support COO format in this work. Other state-of-the-art formats (Li et al. 2018c; Nisa et al. 2019; Smith and Karypis 2015) will be included as our future work. We use inds and val to represent the indices and values of the non-zeros of a sparse tensor respectively. val is a size-M array of floating-point numbers, inds is a size-M array of integer tuples. Figure 1 shows a $4 \times 4 \times 3$ sparse tensor in COO format. The indices of each mode are represented as i, j, and k. Observe that some indices in inds repeat, for example, entries (1, 0, 0) and (1, 0, 2) have the same i and j indices. This redundancy suggests some compression of this indexing metadata should be possible, as proposed in some work (Liu et al. 2017; Smith et al. 2015).

5.2 Algorithms

This section describes the sequential algorithms for the workloads in Sect. 4. All algorithms directly operates on the input sparse tensor(s) without explicit tensor-matrix transformation.

Algorithm 1 Sequential COO-Tew-eq-Addition algorithm for tensors in the same order and shape.

```
Input: A third-order sparse tensor $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I \times J \times K}$ with $M$ non-zeros;
Output: Sparse tensor $\mathcal{Z} \in \mathbb{R}^{I \times J \times K}$;
                                                                      ▹ $\mathcal{Z} = \mathcal{X} .+ \mathcal{Y}$
1: Allocate $\mathcal{Z}$ space with $M$ non-zeros;                    ▹ Pre-allocation space.
2: for $m = 1, \dots, M$ do
3:     $\text{inds}_z^1(m) = \text{inds}_x^1(m)$, $\text{inds}_z^2(m) = \text{inds}_x^2(m)$, $\text{inds}_z^3(m) = \text{inds}_x^3(m)$;
4:     $\text{val}_z(m) = \text{val}_x(m) + \text{val}_y(m)$;
5: return $\mathcal{Z}$;
```

5.2.1 TEW

As mentioned in Sect. 4.1, TEW operation has two cases: one is between two tensors in exactly the same shape and non-zero distribution; the other only requires the two tensors are in the same tensor order.

For the first case, we show TEW addition as an example in Algorithm 1. The output tensor has the same shape and non-zero distribution with the two input tensors, thus it can be pre-allocated. Then the calculation simply does addition by looping all non-zeros.

For the second case, its algorithm is shown in Algorithm 2. The output tensor size is set by the maximum dimension size of the two input tensors. Since we do not know the number of the output non-zeros, we cannot pre-allocate the space of the output tensor $\mathcal{Z}$ but using dynamic allocation to append non-zeros. First, we need to sort tensors $\mathcal{X}$ and $\mathcal{Y}$ in the order of mode $1 \succ 2 \succ 3$, then compare the indices in lexicographical order for each non-zero pair-to-pair, e.g., indices $(2, 1, 1) > (1, 1, 2) > (1, 1, 1)$. If two indices are the equal, then we append the indices and the sum of the two non-zero values to the output $\mathcal{Z}$. Otherwise, we append the smaller indices and its corresponding value to $\mathcal{Z}$. Only if we run out of non-zeros in either $\mathcal{X}$ or $\mathcal{Y}$, we append the rest indices and values of the other one to $\mathcal{Z}$.

Algorithm 2 Sequential COO-TEW-Addition algorithm for general tensors.

```
Input: A third-order sparse tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times J_1 \times K_1}$ with $M_1$ non-zeros, $\mathcal{Y} \in \mathbb{R}^{I_2 \times J_2 \times K_2}$ with $M_2$ non-zeros;
Output: Sparse tensor $\mathcal{Z} \in \mathbb{R}^{I_3 \times J_3 \times K_3}$;
                                                                          ▷ $\mathcal{Z} = \mathcal{X} .+ \mathcal{Y}$
 1: $I_3 = max\{I_1, I_2\}, J_3 = max\{J_1, J_2\}, K_3 = max\{K_1, K_2\}$        ▷ Unify the tensor shape.
 2: Sort $\mathcal{X}$ and $\mathcal{Y}$ in the same dimension order.
 3: $m_1 = 1, m_2 = 1$
 4: while $m_1 < M_1$ and $m_2 < M_2$ do
 5:     if $\text{inds}_x == \text{inds}_y$ then
 6:         Append($\text{inds}_z^1$, $\text{inds}_x^1(m_1)$); Append($\text{inds}_z^2$, $\text{inds}_x^2(m_1)$); Append($\text{inds}_z^3$, $\text{inds}_x^3(m_1)$);
 7:         Append($\text{val}_z$, $\text{val}_x(m_1) + \text{val}_y(m_2)$);
 8:     if $\text{inds}_x > \text{inds}_y$ then
 9:         Append($\text{inds}_z^1$, $\text{inds}_y^1(m_1)$); Append($\text{inds}_z^2$, $\text{inds}_y^2(m_1)$); Append($\text{inds}_z^3$, $\text{inds}_y^3(m_1)$);
10:         Append($\text{val}_z$, $\text{val}_y(m_2)$);
11:     if $\text{inds}_x < \text{inds}_y$ then
12:         Append($\text{inds}_z^1$, $\text{inds}_x^1(m_1)$); Append($\text{inds}_z^2$, $\text{inds}_x^2(m_1)$); Append($\text{inds}_z^3$, $\text{inds}_x^3(m_1)$);
13:         Append($\text{val}_z$, $\text{val}_x(m_1)$);
14: if $m_1 < M_1$ then
15:     Append($\text{inds}_z^1$, $\text{inds}_x^1(m_1, :)$); Append($\text{inds}_z^2$, $\text{inds}_x^2(m_1, :)$); Append($\text{inds}_z^3$, $\text{inds}_x^3(m_1, :)$);
16:     Append($\text{val}_z$, $\text{val}_x(m_1, :)$)
17: if $m_2 < M_2$ then
18:     Append($\text{inds}_z^1$, $\text{inds}_x^1(m_2, :)$); Append($\text{inds}_z^2$, $\text{inds}_x^2(m_2, :)$); Append($\text{inds}_z^3$, $\text{inds}_x^3(m_2, :)$);
19:     Append($\text{val}_z$, $\text{val}_x(m_2, :)$)
20: return $\mathcal{Z}$;
```

Algorithm 3 Sequential COO-Tsm algorithm.

Input: A third-order sparse tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ with M non-zeros;
Output: Output sparse tensor $\mathcal{Y} \in \mathbb{R}^{I\times J\times K}$;

```
                                                        ▹ $\mathcal{Y} = \mathcal{X} \times s$
1: Allocate $\mathcal{Y}$ space with $M$ non-zeros;    ▹ Pre-allocation space.
2: for $m = 1, \ldots, M$ do
3:     $\text{inds}^1_y(m) = \text{inds}^1_x(m)$, $\text{inds}^2_y(m) = \text{inds}^2_x(m)$, $\text{inds}^3_y(m) = \text{inds}^3_x(m)$;
4:     $\text{val}_y(m) = s \times \text{val}_x(m)$;
5: return $\mathcal{Y}$;
```

5.2.2 Ts

Ts algorithm is simple. The output $\mathcal{Y}$ can be pre-allocated and computed by looping all non-zeros. Algorithm 3 shows the Tsm algorithm.

Algorithm 4 Sequential COO-Ttv algorithm.

Input: A third-order sparse tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, dense vector $\mathbf{V} \in \mathbb{R}^{K}$, mode $n = 3$;
Output: Sparse tensor $\mathcal{Y} \in \mathbb{R}^{I\times J}$;

```
                                                        ▹ $\mathcal{Y} = \mathcal{X} \times_n \mathbf{v}$
1: Pre-process to obtain $M_F$: the number of mode-n fibers of $\mathcal{X}$ and $f_{\text{ptr}}$: the beginnings of each $\mathcal{X}$ mode-n
   fiber, sized $M_F$.
2: Allocate $\mathcal{Y}$ space with $M_F$ non-zeros;    ▹ Pre-allocation space.
3: for $f = 1, \ldots, M_F$ do
4:     $\text{inds}^1_Y(f) = \text{inds}^1_X(f_{\text{ptr}}(f))$, $\text{inds}^2_Y(f) = \text{inds}^2_X(f_{\text{ptr}}(f))$
5:     for $m = f_{\text{ptr}}(f), \ldots, f_{\text{ptr}}(f+1) - 1$ do
6:         $k = \text{inds}^3_X(m)$
7:         $\text{val}_Y(f) += \text{val}_X(m) \times u(k)$
8: Return $\mathcal{Y}$;
```

5.2.3 Ttv

Ttv algorithm in mode-n is shown in Algorithm 4. It first pre-compute the number of fibers M_F of input tensor $\mathcal{X}$ and the beginning positions of each fiber. Then we can pre-allocate the output tensor $\mathcal{Y}$ with M_F, because this product does not influence the non-zero layout for I and J modes. The algorithm loops all the fibers of $\mathcal{X}$, and a reduction happens for all non-zeros in each fiber.

Algorithm 5 Sequential COO-TTM algorithm [84].

```
Input: A sparse tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, a dense matrix $\mathbf{U} \in \mathbb{R}^{K\times R}$, and an integer $n = 3$;
Output: Sparse tensor $\mathcal{Y} \in \mathbb{R}^{I\times J\times R}$;
                                                        ▷ $\mathcal{Y} = \mathcal{X} \times_n \mathbf{U}$
1:  Pre-process to obtain $M_F$: the number of mode-n fibers of $\mathcal{X}$ and $f_{ptr}$: the beginnings of each $\mathcal{X}$ mode-n fiber, size $M_F$.
2:  Allocate $\mathcal{Y}$ space with $M_F \times R$ non-zeros;          ▷ Pre-allocation space.
3:  for $f = 1, \dots, M_F$ do
4:      $i = \text{inds}_X^1(f_{ptr}(f)), j = \text{inds}_X^2(f_{ptr}(f))$
5:      for $r = 1, \dots, R$ do
6:          $\text{inds}_Y^1(f \times R + r) = i, \text{inds}_Y^2(f \times R + r) = j, \text{inds}_Y^3(f \times R + r) = r$
7:      for $m = f_{ptr}(f), \dots, f_{ptr}(f + 1) - 1$ do
8:          $k = \text{inds}_X^3(m)$
9:          $value = \text{val}_X(m)$
10:         for $r = 1, \dots, R$ do
11:             $\text{val}_Y(f \times R + r) += value \times u(k \times R + r)$
12: Return $\mathcal{Y}$;
```

5.2.4 TTM

TTM algorithm is illustrated in Algorithm 5. Similarly to TTV algorithm, we obtain the number of fibers M_F and the beginning positions of each fiber then $M_F \times R$ space are allocated for the output tensor $\mathcal{Y}$. The algorithm loops all the M_F fibers and does a reduction between sized-R vectors. This TTM algorithm directly operates on the input sparse tensor by avoiding tensor transformation. The explanation of Algorithm 5 can be found in the work Li et al. (2016), Ma et al. (2018).

5.2.5 MTTKRP

MTTKRP algorithm, well studied in recent work (Li et al. 2019; Nisa et al. 2019; Smith et al. 2015), is shown in Algorithm 6, the output matrix of which is initialized before and only needs to be updated. This algorithm loops all non-zeros of the tensor $\mathcal{X}$ and times the corresponding two matrix vectors, to update the designated output matrix vector. Readers

Algorithm 6 Sequential COO-MTTKRP algorithm ([7]).

```
Input: A third-order sparse tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, dense matrices $\mathbf{B} \in \mathbb{R}^{J\times R}, \mathbf{C} \in \mathbb{R}^{K\times R}$;
Output: Updated dense matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{I\times R}$;
                                                        ▷ $\tilde{\mathbf{A}} \leftarrow \mathcal{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$
1: for $m = 1, \dots, M$ do
2:     $i = \text{inds}^1(m), j = \text{inds}^2(m), k = \text{inds}^3(m)$;
3:     $value = \text{val}(m)$
4:     for $r = 1, \dots, R$ do
5:         $\tilde{A}(i \times R + r) += value \times C(k \times R + r) \times B(j \times R + r)$
6: return $\tilde{\mathbf{A}}$;
```

can refer more details of this algorithm in Bader and Kolda (2007).

According to the above algorithms, we compute the storage, the number of floating-point operations (Flops), the amount of memory access in bytes, and the arithmetic intensity (the ratio of #Flops/#Bytes) in Table 2. For simplicity, we use a cubical third-order sparse tensor $\mathcal{X} \in \mathbb{R}^{I \times I \times I}$ with M non-zeros and M_F fibers as an example. Because of the irregular access pattern of sparse tensors, the memory access does not consider the cache effect. All workloads have arithmetic intensity less than 1, thus it is hard to easily achieve good performance on common architectures. While Mttkrp has the most Flops and memory access, its arithmetic intensity is smaller than Ttm, which it $\sim 1/2$. Tew and Ts have the smallest arithmetic intensity and the largest storage due to the output tensor. Despite of different algorithm behavior, these algorithms are generally considered memory intensive, which demonstrates the emphasis of our PASTA.

5.3 Multicore implementations

Some workloads are easy to parallelize. We parallelize the loop of all non-zeros in Tew-eq (Algorithm 1) and Ts (Algorithm 3). For Ttv (Algorithm 4) and Ttm (Algorithm 5), the loop of fibers is parallelized because each fiber computation is independent.

Tew (Algorithm 2) is difficult to be parallelized because of its dynamic append operations and no pre-allocation available. We partition the two tensors in such a way that there is no overlap between their indices, then we run Tew algorithm locally for a sub-tensor in each thread and append the results to a local output buffer. The partitioning first split one of the two tensors (say $\mathcal{X}$) by slices and meanwhile tend to evenly distribute its non-zeros. This makes sure that all non-zeros of a slice cannot be split into two partitions. Then the partitioning of the other tensor (say $\mathcal{Y}$) is according to this slice partitioning strategy. In this case, we assure every partition does not overlap with each other, thus they can independently computed in parallel.

Table 3 Description of sparse tensors

Tensors	Order	Dimensions	#Non-zeros	Density
vast	3	$165K \times 11K \times 2$	26M	6.9×10^{-3}
nell2	3	$12K \times 9K \times 29K$	77M	2.4×10^{-5}
choa	3	$712K \times 10K \times 767$	27M	5.0×10^{-6}
darpa	3	$22K \times 22K \times 24M$	28M	2.4×10^{-9}
fb-m	3	$23M \times 23M \times 166$	100M	1.1×10^{-9}
fb-s	3	$39M \times 39M \times 532$	140M	1.7×10^{-10}
deli	3	$533K \times 17M \times 2.5M$	140M	6.1×10^{-12}
nell1	3	$2.9M \times 2.1M \times 25M$	144M	9.1×10^{-13}
crime	4	$6K \times 24 \times 77 \times 32$	5M	1.5×10^{-2}
nips	4	$2K \times 3K \times 14K \times 17$	3M	1.8×10^{-6}
enron	4	$6K \times 6K \times 244K \times 1K$	54M	5.5×10^{-9}
flickr4d	4	$320K \times 28M \times 1.6M \times 731$	113M	1.1×10^{-14}
deli4d	4	$533K \times 17M \times 2.5M \times 1K$	140M	4.3×10^{-15}

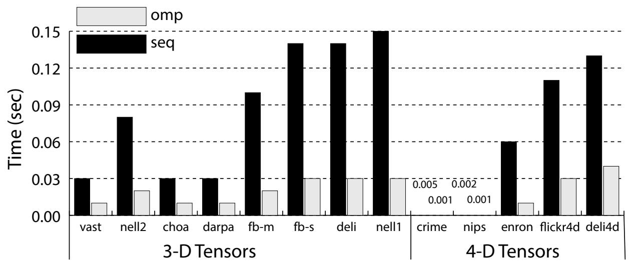

Fig. 2 Tew-eq-addition for sparse tensors in the same shape and non-zero pattern

We parallelize the loop of all non-zeros of Mttkrp (Algorithm 6) as well, but Line 4 may have data race by writing into the same location of $\tilde{\mathbf{A}}$. We implemented two solutions: (1) use atomics to protect the correctness, but the performance suffers much; (2) employ privatization approach to allocate a thread-local buffer. The data is first written to this buffer by each thread privately, then a global reduction for the buffers is used to get the final results. In this case, we can generally get better performance than using atomics.

Table 2 The analysis of data storage and their algorithms for third-order cubical tensors ($\mathcal{X} \in \mathbb{R}^{I \times I \times I}$)

Workloads	Storage (bytes)	Work (flops)	Memory access (bytes)	Arithmetic intensity (AI)
Tew	$48M$	M	$36M$	1 / 36
Ts	$32M$	M	$32M$	1 / 32
Ttv	$(16M + 12M_F)$	$2M$	$(12M + 20M_F)$	$\sim 1/6$
Ttm	$(16M + 16M_FR + 4IR)$	$2MR$	$4MR + 8M + 12M_FR + 8M_F$	$\sim 1/2$
Mttkrp	$(16M + 12IR)$	$3MR$	$12MR + 16M$	$\sim 1/4$

We consider all input tensors with M nonzero entries and M_F fibers, $I \ll M_F \ll M$. The indices use 32 bits, and values are single-precision floating-point numbers with 32 bits

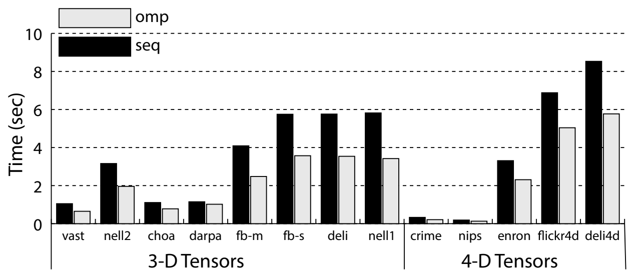

Fig. 3 Tew-addition for sparse tensors in the same order

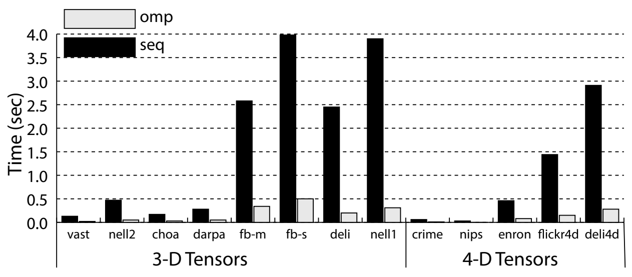

Fig. 5 Ttv: the sum of execution time of all the modes

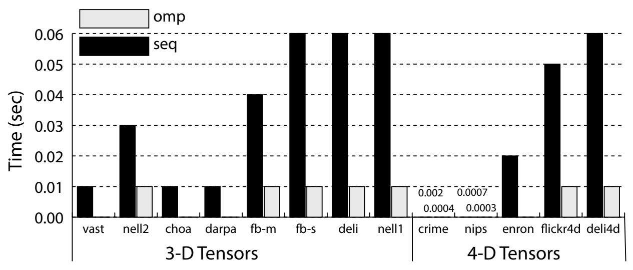

Fig. 4 Tsm execution time

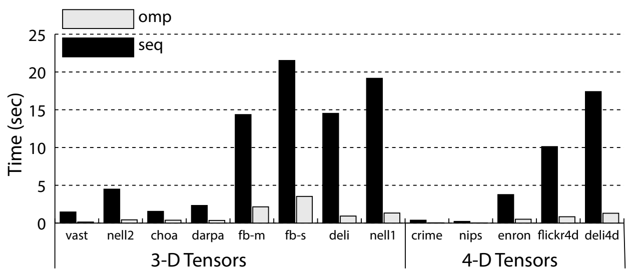

Fig. 6 Ttm: the sum of execution time of all the modes

For these parallel implementations, we have not considered the NUMA effect, which will be another piece of our future work.

6 Dataset

PASTA now only considers real-world data as input. The sparse tensors derived from real-world applications, that appear in Table 3, ordered by decreasing non-zero density separately for third- and fourth-order tensors. Most of these tensors are included in The Formidable Repository of Open Sparse Tensors and Tools (FROSTT) dataset (Refer to the details in Smith et al. 2017b). The darpa (source IP-destination IP-time triples), fb-m, and fb-s (short for "freebase-music" and "freebase-sampled", entity-entity-relation triples) are from the dataset of HaTen2 (Jeon et al. 2015), and choa is built from electronic health records (EHRs) of pediatric patients at Children's Healthcare of Atlanta (CHOA) (Perros et al. 2017). These tensors from different applications have diverse nonzero distribution features.

7 Experiments

We tested these schemes experimentally on a Linux-based Intel Xeon E5-2698 v3 multicore server platform with 32 physical cores distributed on two sockets, each with 2.3 GHz frequency. The processor microarchitecture is Haswell, having 32 KiB L1 data cache and 128 GiB memory. The code artifact is written in the C language using OpenMP parallelization, and was compiled using icc 18.0.1. All experiments use 32 threads for parallel code except being pointed out otherwise. The execution time are all averaged by five runs. For Ttm and Mttkrp, we set the rank $R = 16$.

We demonstrate the sequential and multicore parallel performance for every workload on the dataset (Table 3).

7.1 Tew

Figures 2 and 3 show the execution time of the two cases of Tew addition (Algorithm 1 and 2): in the same non-zero pattern and only in the same tensor order, on all third- and fourth-order tensors. We use the same tensor for the two input for Tew-eq and Tew to better show the algorithm effect. We observe for both cases, parallel Tew outperforms sequential Tew. However, the speedup of Tew-eq is $3.64 - 5.18\times$, while the speedup of Tew is much smaller, which is $1.13 - 1.70\times$. This is because: (1) the parallel strategy of Tew could have a lot more load imbalance than Tew-eq's even non-zero parallelization; (2) some tensors cannot fully use all 32 threads due to the slice partitioning (a heavy slice cannot be further partitioned in Algorithm 2). Besides, due to the dynamic append operation, the sequential Tew is tens of times slower than sequential Tew-eq. From our experiments, Tew subtraction, multiplication, and division behave very similar to Tew addition in execution time.

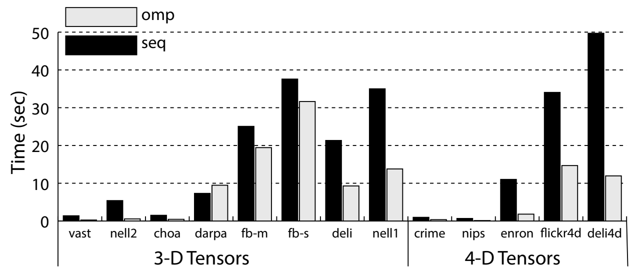

Fig. 7 Mttkrp: the sum of execution time of all the modes

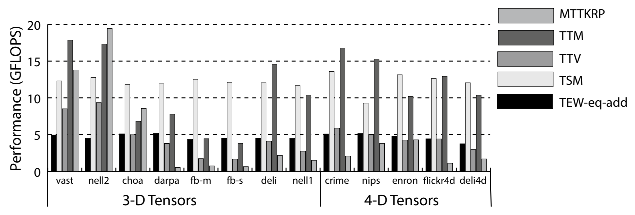

Fig. 8 Performance comparison between tensor workloads

7.2 Ts

Figure 4 plots the sequential and parallel execution time of Tsm. Parallel Tsm achieves 2.17 – 5.92× speedup over sequential Tsm, this is comparable to Tew-eq in Fig. 2. The sequential Tsm executes faster than the sequential Tew, which verifies the analysis in Table 2 and that these two algorithms are memory-bound. (Because they have the same #Flops, compute-bound algorithms should have similar execution time.) From the experiments, the execution times of sequential and parallel Tsa are very close to Tsm.

7.3 Ttv

We illustrate sequential and parallel Ttv time in Fig. 5. Parallel Ttv outperforms sequential case by 5.21 – 12.45×, this is much higher than the speedup of Tew-eq, Tew, and Tsm. This behavior again matches the analysis in Table 2 that Ttv has higher arithmetic intensity. Since higher arithmetic intensity potentially generates less memory contention, thus multicore parallelization could benefit more.

7.4 Ttm

Figure 6 shows the sequential and parallel execution time of Ttm. The speedup of parallel Ttm over sequential case is 4.09 – 15.67× which is comparable with Ttv 's. This also verifies the analysis that Ttm has the highest arithmetic intensity. Sequential Ttm is 4.91 – 11.11× slower than sequential Ttv, that shows the different behavior of timing a dense vector versus a dense matrix.

7.5 Mttkrp

We use privatization technique for parallel Mttkrp, because it performs better than atomics technique on most of tensors. The execution time of sequential and parallel Mttkrp is shown in Fig. 7, where the parallel case gains 0.77 – 9.49× speedup. For tensor darpa, the only case parallel Mttkrp is slower than sequential one because of its large thread-local buffer which consumes a large portion of time to do reduction. The atomics parallel approach could be better in this case, 7.93 versus 7.32 (sequential Mttkrp), but there is still not speedup for this tensor. Mttkrp obtains smaller speedup than Ttm and Ttv mainly because data race exists in the output. Even we use privatization technique to avoid the data race, the extra reduction still take nontrivial amount of time.

Figure 8 illustrates the performance of all workloads in GFLOPS (Giga floating-point operations per second). Generally, Tsm and Ttm obtain the highest performance numbers. Tew and Tsm have good spacial locality, while Tsm has a relatively higher arithmetic intensity. Irregular memory access exists in Ttv, Ttm, and Mttkrp, while Ttm gets the largest arithmetic intensity. From our experiments and analysis above, these relatively simple workloads can well reflect some architecture characteristics. This can help architecture designers and application users to evaluate computer systems.

8 Conclusion

This work presents a sparse tensor algorithm benchmark suite (PASTA) for single-core and multi-core CPUs, which is the first sparse tensor benchmark to the best of our knowledge. PASTA consists of Tew, Ts, Ttv, Ttm, Mttkrp workloads to represent sparse tensor algorithms from different tensor methods in a various application scenarios. Besides, these workloads can reflect computer architecture features differently from our analysis.

As a benchmark suite, PASTA already processes good properties such as application and machine diversity, state-of-the-art data structures, algorithms, and optimization techniques included, compatibility for research support, and real-world data set. Some future work should be done to make PASTA more complete and robust: (1) more computer systems support, such as GPUs, FPGAs, and distributed systems; (2) more workloads especially tensor-times-tensor product (Ttt); (3) more state-of-the-art sparse tensor formats, e.g., hierarchical COO (HiCOO) and compressed sparse fiber (CSF) format; (4) synthetic data generation for more precise machine performance measurement. PASTA is an open-source project and a continuously effort to keep its timeliness.

Acknowledgements This research was partially funded by the US Department of Energy, Office for Advanced Scientific Computing (ASCR) under Award No. 66150: "CENATE: The Center for Advanced Technology Evaluation". Pacific Northwest National Laboratory (PNNL) is a multiprogram national laboratory operated for DOE by Battelle Memorial Institute under Contract DE-AC05-76RL01830. This work was partially supported by the High Performance Data Analytics (HPDA) program at Pacific Northwest National Laboratory.

References

Abadi, M., et al.: Large-Scale Machine Learning on Heterogeneous Systems, 2015. TensorFlow, Google Brain Team, California (2015)

Acar, E., Aykut-Bingol, C., Bingol, H., Bro, R., Yener, B.: Multiway analysis of epilepsy tensors. Bioinformatics **23**(13), i10–i18 (2007). https://doi.org/10.1093/bioinformatics/btm210

Acar, E., Dunlavy, D.M., Kolda, T.G., Mørup, M.: Scalable tensor factorizations for incomplete data. Chemometr. Intell. Lab. Syst. **106**(1), 41–56 (2011)

Acar, E., Kolda, T.G., Dunlavy, D.M.: All-at-once optimization for coupled matrix and tensor factorizations. (2011)

Anandkumar, A., Ge, R., Hsu, D., Kakade, S.M., Telgarsky, M.: Tensor decompositions for learning latent variable models. J. Mach. Learn. Res. **15**(1), 2773–2832 (2014)

Austin, W., Ballard, G., Kolda, T.G.: Parallel tensor compression for large-scale scientific data. In: 2016 IEEE international parallel and distributed processing symposium (IPDPS), pp. 912–922. https://doi.org/10.1109/IPDPS.2016.67

Bader, B.W., Kolda, T.G.: Efficient MATLAB computations with sparse and factored tensors. SIAM J. Sci. Comput. **30**(1), 205–231 (2007). https://doi.org/10.1137/060676489

Bader, B.W., Kolda, T.G. et al. MATLAB Tensor Toolbox (Version 3.0-dev) (2017). https://www.tensortoolbox.org

Battaglino, C., Ballard, G., Kolda, T.G.: A practical randomized CP tensor decomposition. SIAM J. Matrix Anal. Appl. **39**(2), 876–901 (2018)

Benson, A.R., Gleich, D.F., Leskovec, J.: Tensor spectral clustering for partitioning higher-order network structures. arXiv:1502.05058 (2015)

Beutel, A., Kumar, A., Papalexakis, E., Talukdar, P.P., Faloutsos, C., Xing, E.P.: FLEXIFACT: scalable flexible factorization of coupled tensors on hadoop. In: NIPS 2013 big learning workshop (2013)

Bienia, C., Kumar, S., Singh, J.P., Li, K.: The PARSEC benchmark suite: characterization and architectural implications. In Proceedings of the 17th international conference on parallel architectures and compilation techniques, ACM, pp. 72–81 (2008)

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011). https://doi.org/10.1561/2200000016

Bro, R., Sidiropoulos, N.D., Giannakis, G.B.: A fast least squares algorithm for separating trilinear mixtures. In Independent Component Analysis (1999)

Calvin, J.A., Valeev, E.F.: TiledArray: a massively-parallel, block-sparse tensor framework (Version v0.6.0). Available from https://github.com/valeevgroup/tiledarray. (2016)

Cao, B., He, L., Kong, X., Philip, S.Y., Hao, Z., Ragin, A.B.: Tensor-based multi-view feature selection with applications to brain diseases. In: Data Mining (ICDM), 2014 IEEE international conference on, pp. 40–49 (2014). https://doi.org/10.1109/ICDM.2014.26

Cao, B., Kong, X., Yu, P.S.: A review of heterogeneous data mining for brain disorders. arXiv:abs/1508.01023 (2015)

Carroll, J.D., 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

Carroll, J.D., Pruzansky, S., Kruskal, J.B.: CANDELINC: a general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. Psychometrika **45**(1980), 3–24 (1980)

Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J.W., Lee, S., Skadron, K.: Rodinia: A benchmark suite for heterogeneous computing. In 2009 IEEE international symposium on workload characterization (IISWC), pp. 44–54 (2009). https://doi.org/10.1109/IISWC.2009.5306797

Cheng, D., Peng, R., Liu, Y., Perros, I.: SPALS: fast alternating least squares via implicit leverage scores sampling. In Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.), Advances in neural information processing systems 29, Curran Associates, Inc., pp. 721–729. http://papers.nips.cc/paper/6436-spals-fast-alternating-least-squares-via-implicit-leverage-scores-sampling.pdf (2016)

Chi, E.C., Kolda, T.G.: On tensors, sparsity, and nonnegative factorizations. SIAM J. Matrix Anal. Appl. **33**(4), 1272–1299 (2012)

Choi, J., Liu, X., Smith, S., Simon, T.: Blocking optimization techniques for sparse tensor computation. pp. 568–577 (2018). https://doi.org/10.1109/IPDPS.2018.00066

Choi, J.H., Vishwanathan, S.: DFacTo: Distributed Factorization of Tensors. In: Ghahramani, Z., Welling, M.C., Cortes, N.D., Lawrence, Weinberger, K.Q. (eds.), Advances in neural information processing systems 27, Curran Associates, Inc., pp. 1296–1304 (2014)

Cichocki, A.: Tensor decompositions: a new concept in brain data analysis? arXiv:1305.0395 (2013)

Cichocki, A.: Era of big data processing: a new approach via tensor networks and tensor decompositions. arXiv:abs/1403.2048 (2014)

Cichocki, A., Mandic, D., De Lathauwer, L., Zhou, G., Zhao, Q., Caiafa, C., Phan, H.A.: Tensor decompositions for signal processing applications: from two-way to multiway component analysis. Signal Process. Magz. IEEE **32**(2), 145–163 (2015). https://doi.org/10.1109/MSP.2013.2297439

Cichocki, A., Lee, N., Oseledets, I.V., Phan, A., Zhao, Q., Mandic, D.: Low-rank tensor networks for dimensionality reduction and large-scale optimization problems: perspectives and challenges PART 1. (2016). arXiv:cs.NA/1609.00893

Cohen, N., Sharir, O., Shashua, A.: On the expressive power of deep learning: a tensor analysis. arXiv:abs/1509.05009 (2015)

Davidson, I., Gilpin, S., Carmichael, O., Walker, P.: Network discovery via constrained tensor analysis of fMRI Data. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining(KDD '13). ACM, New York, pp. 194–202 (2013). https://doi.org/10.1145/2487575.2487619

De Lathauwer, L.: Decompositions of a higher-order tensor in block terms-part I: lemmas for partitioned matrices. SIAM J. Matrix Anal. Appl. **30**(3), 1022–1032 (2008). https://doi.org/10.1137/060661685

De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(2000), 1253–1278 (2000)

De Lathauwer, L., De Moor, B., Vandewalle, J.: On the best rank-1 and Rank-(R1,R2,.,RN) approximation of higher-order tensors. SIAM J. Matrix Anal. Appl. **21**(4), 1324–1342 (2000). https://doi.org/10.1137/S0895479898346995

De Lathauwer, L., Vervliet, N., Boussé, M., Debals, O.: Dealing with curse and blessing of dimensionality through tensor decompositions. (2017)

Dixit, K.M.: The SPEC benchmarks. Parallel Comput. **17**(10–11), 1195–1209 (1991)

Eldén, L., Savas, B.: A Newton–Grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor. SIAM J. Matrix Anal. Appl. **31**(2), 248–271 (2009). https://doi.org/10.1137/070688316

Epifanovsky, E., Wormit, M., Kuś, T., Landau, A., Zuev, D., Khistyaev, K., Manohar, P., Kaliman, I., Dreuw, A., Krylov, A.I.: New implementation of high-level correlated methods using a general block tensor library for high-performance electronic structure calculations. J. Comput. Chem. **34**(26), 2293–2309 (2013). https://doi.org/10.1002/jcc.23377

Evenbly, G., Vidal, G.: Algorithms for entanglement renormalization. Phys. Rev. B **79**(14), 144108 (2009)

Fang, X., Pan, R.: Fast DTT: a near linear algorithm for decomposing a tensor into factor tensors. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp. 967–976 (2014)

Giovannetti, V., Montangero, S., Fazio, R.: Quantum multiscale entanglement renormalization ansatz channels. Phys. Rev. Lett. **101**(18), 180503 (2008)

Gorodetsky, A.A., Sertac, K., Youssef, M.M.: Efficient high-dimensional stochastic optimal motion control using tensor-train decomposition (2008)

Grasedyck, L.: Hierarchical singular value decomposition of tensors. SIAM J. Matrix Anal. Appl. **31**(4), 2029–2054 (2010). https://doi.org/10.1137/090764189

Grasedyck, L., Kressner, D., Tobler, C.: A literature survey of low-rank tensor approximation techniques. GAMM-Mitteilungen **36**(1), 53–78 (2013)

Hackbusch, W., Kühn, S.: A new scheme for the tensor representation. J. Fourier Anal. Appl. **15**(5), 706–722 (2009). https://doi.org/10.1007/s00041-009-9094-9

Hansen, S., Plantenga, T., Kolda, T.G.: Newton-based optimization for Kullback–Leibler nonnegative tensor factorizations. Optim. Methods Softw. **30**(2015), 1002–1029 (2015)

Harshman, R., Lundy, M.: Uniqueness proof for a family of models sharing features of Tucker's three-mode factor analysis and PARAFAC/Candecomp. Psychometrika **61**(1), 133–154 (1996). http://EconPapers.repec.org/RePEc:spr:psycho:v:61:y:1996:i:1:p:133-154

Harshman, R.A.: Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multi-modal factor analysis. UCLA Work. Pap. Phonetic **16**(1), 84 (1970)

Harshman, R.A.: PARAFAC2: mathematical and technical notes. UCLA Work. Pap. Phonetic **22**, 30–44 (1972)

Hein, E., Conte, T., Young, J.S., Eswar, S., Li, J., Lavin, P., Vuduc, R., Riedy, J.: An initial characterization of the Emu Chick. 2018 IEEE international parallel and distributed processing symposium workshops, p. 10 (2018)

Henderson, J., Ho, J.C., Kho, A.N., Denny, J.C., Malin, B.A., Sun, J., Ghosh, J.: Granite: diversified, sparse tensor factorization for electronic health record-based phenotyping. In: 2017 IEEE international conference on healthcare informatics (ICHI), pp. 214–223 (2017). https://doi.org/10.1109/ICHI.2017.61

Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. J. Math. Phys **6**(1), 164–189 (1927)

Ho, J.C., Ghosh, J., Steinhubl, S.R., Stewart, W.F., Denny, J.C., Malin, B.A., Sun, J.: Limestone: high-throughput candidate phenotype generation via tensor factorization. J. Biomed. Inf. **52**(2014), 199–211 (2014c)

Ho, J.C., Ghosh, J., Sun, J.: Extracting phenotypes from patient claim records using nonnegative tensor factorization. In: Brain informatics and health, Springer, pp. 142–151 (2014a)

Ho, J.C., Ghosh, J., Sun, J.: Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining(KDD '14), ACM, New York, pp. 115–124 (2014b). https://doi.org/10.1145/2623330.2623658

Hutchinson, B., Deng, L., Dong, Y.: Tensor deep stacking networks. Pattern Anal. Mach. Intell. IEEE Trans. **35**(8), 1944–1957 (2013)

Ishteva, M., Absil, P., Van Huffel, S., De Lathauwer, L.: Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme. SIAM J. Matrix Anal. Appl. **32**(1), 115–135 (2011). https://doi.org/10.1137/090764827

Janzamin, M., Sedghi, H., Anandkumar, A.: Generalization bounds for neural networks through tensor factorization. arXiv:abs/1506.08473 (2015)

Jeon, I., Papalexakis, E.E., Kang, U., Faloutsos, C.: HaTen2: billion-scale tensor decompositions (Version 1.0). http://datalab.snu.ac.kr/haten2/. (2015)

Jiang, M., Cui, P., Wang, F., Xu, X., Zhu, W., Yang, S.: FEMA: flexible evolutionary multi-faceted analysis for dynamic behavioral pattern discovery. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 1186–1195 (2014)

Jiang, T., Sidiropoulos, N.D.: Kruskal's permutation lemma and the identification of CANDECOMP/PARAFAC and bilinear models with constant modulus constraints. Signal Process. IEEE Trans. **52**(9), 2625–2636 (2004)

Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-l., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T.V., Gottipati, R., Gulland, W., Hagmann, R., Ho, C.R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., Yoon, D.H.: In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th annual international symposium on computer architecture(ISCA '17). ACM, New York, NY, pp. 1–12.https://doi.org/10.1145/3079856.3080246

Kaliman, I.A., Krylov, A.I.: New algorithm for tensor contractions on multi-core CPUs, GPUs, and accelerators enables CCSD and EOM-CCSD calculations with over 1000 basis functions on a single compute node. J. Comput. Chem. **38**(11), 842–853 (2017). https://doi.org/10.1002/jcc.24713

Kang, U., Papalexakis, E., Harpale, A., Faloutsos, C.: GigaTensor: scaling tensor analysis up by 100 times - algorithms and discoveries. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining(KDD '12). ACM, New York, pp. 316–324 (2012). https://doi.org/10.1145/2339530.2339583

Kapteyn, A., Neudecker, H., Wansbeek, T.: An approach to n-mode components analysis. Psychometrika **51**(2), 269–275 (1986). https://doi.org/10.1007/BF02293984

Karatzoglou, A., Amatriain, X., Baltrunas, L., Oliver, N.: Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In: Proceedings of the 4th ACM conference on recommender systems(RecSys '10). ACM, New York, pp. 79–86 (2010). https://doi.org/10.1145/1864708.1864727

Karlsson, L., Kressner, D., Uschmajew, A.: Parallel algorithms for tensor completion in the CP format. Parallel Comput. **57**(2016), 222–234 (2016). https://doi.org/10.1016/j.parco.2015.10.002

Kaya, O., Uçar, B.: Scalable sparse tensor decompositions in distributed memory systems. In: Proceedings of the international conference for high performance computing, networking, storage and analysis(SC '15). ACM, New York, Article 77, p. 11 (2015). https://doi.org/10.1145/2807591.2807624

Kaya, O., Uçar, B.: Parallel candecomp/parafac decomposition of sparse tensors using dimension trees. SIAM J. Sci. Comput. **40**(1), C99–C130 (2018). https://doi.org/10.1137/16M1102744

Khoromskaia, V., Khoromskij, B.N.: Tensor numerical methods in quantum chemistry. Walter de Gruyter GmbH & Co KG (2018)

Kiers, H.A.L., der Kinderen, A.: A fast method for choosing the numbers of components in Tucker3 analysis. Br. J. Math. Stat. Psychol. **56**(1), 119–125 (2003)

KleinOsowski, A.J., Lilja, D.J.: MinneSPEC: a new SPEC benchmark workload for simulation-based computer architecture research. IEEE Comput. Archit. Lett. **1**(1), 7–7 (2002)

Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009). https://doi.org/10.1137/07070111X

Kolda, T.G., Bader, B.W.: The TOPHITS model for higher-order web link analysis. Workshop Link Anal. Counterterror. Secur. **7**, 26–29 (2006)

Kolda, T.G., Sun, J.: Scalable tensor decompositions for multi-aspect data mining. In: Proceedings of the 2008 eighth IEEE international conference on data mining(ICDM '08). IEEE Computer Society, Washington, DC, pp. 363–372 (2008). https://doi.org/10.1109/ICDM.2008.89

Köppl, C., Werner, H.-J.: Parallel and low-order scaling implementation of hartree-fock exchange using local density fitting. J. Chem. Theory Comput. **12**(7), 3122–3134 (2016). https://doi.org/10.1021/acs.jctc.6b00251

Latchoumane, C.-F.V. Vialatte, F.-B., Solé-Casals, J., Maurice, M., Wimalaratna, S.R. Hudson, N., Jeong, J., Cichocki, A.: Multiway array decomposition analysis of EEGs in Alzheimer's disease. J. Neurosci. Methods**207**(1), 41–50 (2012)

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., Lempitsky, V.: Speeding-up convolutional neural networks using fine-tuned CP-decomposition. arXiv:1412.6553 (2014)

Lee, C, Potkonjak, M, Mangione-Smith, W.H., MediaBench: a tool for evaluating and synthesizing multimedia and communications systems. In Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture, IEEE Computer Society, pp. 330–335 (1997)

Lewis, C.A., Calvin, J.A., Valeev, E.F.: Clustered low-rank tensor format: introduction and application to fast construction of Hartree-Fock exchange. J. Chem. Theory Comput. **12**(12), 5868–5880 (2016). https://doi.org/10.1021/acs.jctc.6b00884

Li, A., Song, S.L., Chen, J., Liu, X., Tallent, N., Barker, K.: Tartan: evaluating modern GPU interconnect via a multi-GPU benchmark suite. In: 2018 IEEE international symposium on workload characterization (IISWC), IEEE, pp. 191–202 (2018)

Li, J.: Scalable tensor decompositions in high performance computing environments. Ph.D. Dissertation. Georgia Institute of Technology, Atlanta, GA (2018)

Li, J., Choi, J., Perros, I., Sun, J., Vuduc, R.: Model-driven sparse CP decomposition for higher-order tensors. In: 2017 IEEE international parallel and distributed processing symposium (IPDPS), pp. 1048–1057 (2017). https://doi.org/10.1109/IPDPS.2017.80

Li, J., Ma, Y., Vuduc, R.: ParTI!: a parallel tensor infrastructure for multicore CPU and GPUs (Version 1.0.0). https://github.com/hpcgarage/ParTI. (2018)

Li, J., Ma, Y., Yan, C., Vuduc, R.: Optimizing sparse tensor times matrix on multi-core and many-core architectures. In: Proceedings of the 6th workshop on irregular applications: architectures and algorithms (IA3 '16). IEEE Press, Piscataway, pp. 26–33 (2016). https://doi.org/10.1109/IA3.2016.10

Li, J., Sun, J., Vuduc, R.: HiCOO: hierarchical storage of sparse tensors. In: Proceedings of the ACM/IEEE international conference on high performance computing, networking, storage and analysis (SC), Dallas, TX (2018)

Li, J., Tan, G., Chen, M., Sun, N.: SMAT: an input adaptive autotuner for sparse matrix-vector multiplication. In: Proceedings of the 34th ACM SIGPLAN conference on programming language design and implementation(PLDI '13), ACM, New York, pp. 117–126 (2013). https://doi.org/10.1145/2491956.2462181

Li, J., Uçar, B., Çatalyürek, Ü.V., Sun, J., Barker, K., Vuduc, R.: Efficient and effective sparse tensor reordering. In: Proceedings of the ACM international conference on supercomputing(ICS '19). ACM, New York, pp. 227–237 (2019). https://doi.org/10.1145/3330345.3330366

Li, Z., Uschmajew, A., Zhang, S.: On Convergence of the maximum block improvement method. SIAM J. Optim. **25**(1), 210–233 (2015). https://doi.org/10.1137/130939110

Liu, B., Wen, C., Sarwate, A.D., and Dehnavi, M.M.: A unified optimization approach for sparse tensor operations on GPUs. In: 2017 IEEE international conference on cluster computing (CLUSTER), pp. 47–57 (2017). https://doi.org/10.1109/CLUSTER.2017.75

Ma, Y., Li, J., Wu, X., Yan, C., Sun, J., Vuduc, R.: Optimizing sparse tensor times matrix on GPUs. J. Parallel Distrib. Comput. (2018). https://doi.org/10.1016/j.jpdc.2018.07.018

Manzer, S., Epifanovsky, E., Krylov, A.I., Head-Gordon, M.: A general sparse tensor framework for electronic structure theory. J. Chem. Theory Comput. **13**(3), 1108–1116 (2017). https://doi.org/10.1021/acs.jctc.6b00853

Matsubara, Y., Sakurai, Y., van Panhuis, W.G., Faloutsos, C.: FUNNEL: automatic mining of spatially coevolving epidemics. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp. 105–114 (2014)

Mohlenkamp, M.J.: Musings on multilinear fitting (2010)

Mørup, M., Hansen, L.K., Arnfred, S.M., Lim, L., Madsen, K.H.: Shift invariant multilinear decomposition of neuroimaging data. Accept Publ. NeuroImage **42**(4), 1439–50 (2008). http://www2.imm.dtu.dk/pubdb/p.php?5551

Nakatani, N., Chan, G.K.-L.: Efficient tree tensor network states (TTNS) for quantum chemistry: generalizations of the density matrix renormalization group algorithm. J. Chem. Phys. **138**(13), 134113 (2013). https://doi.org/10.1063/1.4798639

Nisa, I., Li, J., Sukumaran-Rajam, A., Vuduc, R.W., Sadayappan, P.: Load-balanced Sparse MTTKRP on GPUs. (2019). arXiv:1904.03329

Novikov, A., Izmailov, P., Khrulkov, V., Figurnov, M., Oseledets, I.V.: Tensor train decomposition on TensorFlow (T3F). arXiv:abs/1801.01928 (2018)

Novikov, A., Podoprikhin, D., Osokin, A., Vetrov, D.: Tensorizing neural networks. arXiv:abs/1509.06569 (2015)

Novikov, A., Rodomanov, A., Osokin, A., Vetrov, D.: Putting MRFs on a tensor train. In: Tony J., Eric P.X. (Eds.), Proceedings of the 31st international conference on machine learning (ICML-14), JMLR Workshop and Conference Proceedings, pp. 811–819 (2014). http://jmlr.org/proceedings/papers/v32/novikov14.pdf

Oh, H.: Tensors in power system computation I: distributed computation for optimal power flow, DC OPF. arXiv:abs/1605.06735 (2016)

Orús, R.: A practical introduction to tensor networks: matrix product states and projected entangled pair states. Ann. Phys. **349**(2014), 117–158 (2014)

Oseledets, I.V.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011). https://doi.org/10.1137/090752286

Papalexakis, E.E., Akoglu, L., Ienco, D.: Do more views of a graph help? Community detection and clustering in multi-graphs. In:

Proceedings of the 16th international conference on information fusion, FUSION 2013, Istanbul, July 9–12, 2013, pp. 899–905. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6641090

Papalexakis, E.E., Faloutsos, C., Mitchell, T.M., Talukdar, P.P., Sidiropoulos, N.D., Murphy, B.: Turbo-SMT: accelerating coupled sparse matrix-tensor factorizations by 200x. Chapter 14, pp. 118–126 (2014). https://doi.org/10.1137/1.9781611973440.14

Papalexakis, E.E., Faloutsos, C., Sidiropoulos, N.D.: ParCube: sparse parallelizable tensor decompositions. In: Proceedings of the 2012 European conference on machine learning and knowledge discovery in databases - Volume Part I (ECML PKDD'12). Springer, Berlin, pp. 521–536 (2012). https://doi.org/10.1007/978-3-642-33460-3_39

Papalexakis, E.E., Faloutsos, C., Sidiropoulos, N.D.: ParCube: sparse parallelizable CANDECOMP-PARAFAC tensor decomposition. ACM Trans. Knowl. Discov. Data 10, 1, Article 3, pp. 25 (2015). https://doi.org/10.1145/2729980

Papalexakis, E.E., Sidiropoulos, N.D.: Co-clustering as multilinear decomposition with sparse latent factors. In: Acoustics, speech and signal processing (ICASSP), 2011 IEEE international conference on. IEEE, 2064–2067 (2011)

Peng, C., Calvin, J.A., Pavošević, F., Zhang, J., Valeev, E.F.: Massively parallel implementation of explicitly correlated coupled-cluster singles and doubles using tiledarray framework. J. Phys. Chem. A **120**(51), 10231–10244 (2016). https://doi.org/10.1021/acs.jpca.6b10150

Perros, I, Chen, R, Vuduc, R, Sun, J.: Sparse hierarchical tucker factorization and its application to healthcare. In: Proceedings of the 2015 IEEE international conference on data mining (ICDM)*(ICDM '15)*. IEEE Computer Society, Washington, DC, pp. 943–948 (2015). https://doi.org/10.1109/ICDM.2015.29

Perros, I., Papalexakis, E.E., Wang, F., Vuduc, R., Searles, E., Thompson, M., Sun, J.: SPARTan: scalable PARAFAC2 for large & sparse data. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (KDD '17). ACM, New York, pp. 375–384 (2017). https://doi.org/10.1145/3097983.3098014

Phipps, E.T., Kolda, T.G.: Software for sparse tensor decomposition on emerging computing architectures. arXiv:abs/1809.09175 (2018)

Poovey, J.A., Conte, T.M.: Markus Levy, and Shay Gal-On. 2009. A benchmark characterization of the EEMBC benchmark suite. IEEE micro 29, 5 (2009)

Rajih, M., Comon, P.: Enhanced line search: a novel method to accelerate Parafac. In: 2005 13th European signal processing conference, pp. 1–4 (2005)

Ravindran, N., Sidiropoulos, N.D., Smith, S., Karypis, G.: Memory-efficient parallel computation of tensor and matrix products for big tensor decompositions. Proceedings of the Asilomar conference on signals, systems, and computers (2014)

Rendle, S., Balby Marinho, L., Nanopoulos, A., Schmidt-Thieme, L.: Learning optimal ranking with tensor factorization for tag recommendation. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining (KDD '09), ACM, New York, pp. 727–736 (2009). https://doi.org/10.1145/1557019.1557100

Reynolds, M., Doostan, A., Beylkin, G.: Randomized alternating least squares for canonical tensor decompositions: application to a PDE with random data. SIAM J. Sci. Comput. **38**(5), A2634–A2664 (2016). https://doi.org/10.1137/15M1042802

Romera-Paredes, B., Aung, M.H., Bianchi-Berthouze, N., Pontil, M.: Multilinear multitask learning. In Proceedings of the 30th international conference on international conference on machine learning, Volume 28 (ICML'13). JMLR.org, III–1444–III–1452 (2013). http://dl.acm.org/citation.cfm?id=3042817.3043098

Savas, B., Lim, L.: Quasi-Newton methods on Grassmannians and multilinear approximations of tensors. SIAM J. Sci. Comput. **32**(6), 3352–3393 (2010). https://doi.org/10.1137/090763172

Schein, A., Paisley, J., Blei, D.M., Wallach, H.: Bayesian poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, (KDD '15). ACM, New York, pp. 1045–1054 (2015). https://doi.org/10.1145/2783258.2783414

Sedaghati, N., Mu, T., Pouchet, L.-N., Parthasarathy, S., Sadayappan, P.: Automatic selection of sparse matrix representation on GPUs. In: Proceedings of the 29th ACM on international conference on supercomputing (ICS '15). ACM, New York, pp. 99–108 (2015). https://doi.org/10.1145/2751205.2751244

Setiawan, H., Huang, Z., Devlin, J., Lamar, T., Zbib, R., Schwartz, R.M., Makhoul, J.: Statistical machine translation features with multitask tensor networks. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing of the Asian Federation of natural language processing, ACL 2015, July 26-31, 2015, Beijing, Volume 1: Long Papers, pp. 31–41. (2015). http://aclweb.org/anthology/P/P15/P15-1004.pdf

Sidiropoulos, N.D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E.E., Faloutsos, C.: Tensor decomposition for signal processing and machine learning. IEEE Trans. Signal Process. **65**(13), 3551–3582 (2017). https://doi.org/10.1109/TSP.2017.2690524

Sidiropoulos, N.D., Giannakis, G.B., Bro, R.: Blind PARAFAC receivers for DS-CDMA systems. Signal Process. IEEE Trans. **48**(3), 810–823 (2000)

Signoretto, M., Dinh, Q.T., De Lathauwer, L., Suykens, J.A.K.: Learning with tensors: a framework based on convex optimization and spectral regularization. Mach. Learn. **94**(3), 303–351 (2014). https://doi.org/10.1007/s10994-013-5366-3

Smith, S., Beri, A., Karypis, G.: Constrained tensor factorization with accelerated AO-ADMM. In: 46th international conference on parallel processing (ICPP '17). IEEE (2017)

Smith, S., Choi, J.W., Li, J., Vuduc, R., Park, J., Liu, X., Karypis, G.: FROSTT: the formidable repository of open sparse tensors and tools (2017). http://frostt.io/

Smith, S., Karypis, G.: Tensor-Matrix products with a compressed sparse tensor. In: Proceedings of the 5th workshop on irregular applications: architectures and algorithms. ACM, 7 (2015)

Smith, S, Karypis, G: A medium-grained algorithm for distributed sparse tensor factorization. In: Parallel and distributed processing symposium (IPDPS), 2016 IEEE international, IEEE (2016)

Smith, S., Karypis, G.: Accelerating the tucker decomposition with compressed sparse tensors. In: European conference on parallel processing, Springer, New York (2017)

Smith, S., Park, J., Karypis, G.: An exploration of optimization algorithms for high performance tensor completion. Proceedings of the 2016 ACM/IEEE conference on supercomputing (2016)

Smith, S., Park, J., Karypis, G.: Sparse tensor factorization on many-core processors with high-bandwidth memory. 31st IEEE international parallel & distributed processing symposium (IPDPS'17) (2017)

Smith, S., Ravindran, N., Sidiropoulos, N., Karypis, G.: SPLATT: efficient and parallel sparse tensor-matrix multiplication. In: Proceedings of the 29th IEEE international parallel & distributed processing symposium, (IPDPS) (2015)

Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in neural information processing systems, pp. 926–934 (2013)

Solomonik, E., Hoefler, T.: Sparse tensor algebra as a parallel programming model. (2015). arXiv:cs.MS/1512.00066

Song, C., Martínez, T.J.: Atomic orbital-based SOS-MP2 with tensor hypercontraction. I. GPU-based tensor construction and exploiting sparsity. J. Chem. Phys. **144**(17), 174111 (2016). https://doi.org/10.1063/1.4948438

Song, H.A., Hooi, B., Jereminov, M., Pandey, A., Pileggi, L.T., Faloutsos, C.: PowerCast: mining and forecasting power grid sequences. In: ECML/PKDD (2017)

Song, Z., Woodruff, D.P., Zhang, H.: Sublinear time orthogonal tensor decomposition. In: Proceedings of the 30th international conference on neural information processing systems (NIPS'16), Curran Associates Inc., USA, 793–801 (2016). http://dl.acm.org/citation.cfm?id=3157096.3157185

Sorber, L., Domanov, I., Barel, M., Lathauwer, L.: Exact line and plane search for tensor optimization. Comput. Optim. Appl. **63**(1), 121–142 (2016). https://doi.org/10.1007/s10589-015-9761-5

Sorber, L., Van Barel, M., De Lathauwer, L.: Optimization-based algorithms for tensor decompositions: canonical polyadic decomposition, decomposition in rank-(L_r, L_r, 1) terms, and a new generalization. SIAM J. Optim. **23**(2), 695–720 (2013). https://doi.org/10.1137/120868323

Su, B.-Y., Keutzer, K.: clSpMV: a cross-platform OpenCL SpMV Framework on GPUs. In: Proceedings of the 26th ACM international conference on supercomputing (ICS '12). ACM, New York, pp. 353–364 (2012). https://doi.org/10.1145/2304576.2304624

Sun, J., Papadimitriou, S., Lin, C.-Y., Cao, N., Liu, S., Qian, W.: MultiVis: content-based social network exploration through multi-way visual analysis. In: SDM, Vol. 9. SIAM, 1063–1074 (2009)

Sun, J., Tao, D., Faloutsos, C.: Beyond streams and graphs: dynamic tensor analysis. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining (KDD '06). ACM, New York, pp. 374–383 (2006). https://doi.org/10.1145/1150402.1150445

Sun, J.-T., Zeng, H.-J., Liu, H., Lu, Y., Chen, Z.: CubeSVD: a novel approach to personalized web search. In: Proceedings of the 14th international conference on world wide web (WWW '05). ACM, New York, 382–390 (2005). https://doi.org/10.1145/1060745.1060803

Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Tag recommendations based on tensor dimensionality reduction. In: Proceedings of the 2008 ACM conference on recommender systems, (RecSys '08). ACM, New York, NY, pp. 43–50 (2008). https://doi.org/10.1145/1454008.1454017

Tao, D., Li, X., Xindong, W., Weiming, H., Maybank, S.J.: Supervised tensor learning. Knowl. Inf. Syst. **13**(1), 1–42 (2007). https://doi.org/10.1007/s10115-006-0050-6

Tomasi, G., Bro, R.: A comparison of algorithms for fitting the PARAFAC model. Comput. Stat. Data Anal. **50**(7), 1700–1734 (2006). https://doi.org/10.1016/j.csda.2004.11.013

Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966). https://doi.org/10.1007/BF02289464

Vervliet, N., Debals, O., Sorber, L., Van Barel, M., De Lathauwer, L.: Tensorlab (Version 3.0). http://www.tensorlab.net. (2016)

Vervliet, N., De Lathauwer, L.: A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors. IEEE J. Select. Top. Signal Process. **10**(2), 284–295 (2016). https://doi.org/10.1109/JSTSP.2015.2503260

Wang, F., Zhang, P., Qian, B., Wang, X., Davidson, I.: Clinical risk prediction with multilinear sparse logistic regression. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp. 145–154 (2014)

Wang, H., Thoss, M.: Multilayer formulation of the multiconfiguration time-dependent Hartree theory. J. Chem. Phys. **119**(3), 1289–1299 (2003). https://doi.org/10.1063/1.1580111

Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., Gao, W., Jia, Z., Shi, Y., Zhang, S., Zheng, C., Lu, G., Zhan, K., Li, X., Qiu, B.: . BigDataBench: a big data benchmark suite from internet services. In: 2014 IEEE 20th international symposium on high performance computer architecture (HPCA), pp. 488–499 (2014). https://doi.org/10.1109/HPCA.2014.6835958

Wang, Y., Chen, R., Ghosh, J., Denny, J.C., Kho, A., Chen, Y., Malin, B.A., Sun, J.: Rubik: knowledge guided tensor factorization and completion for health data analytics. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining *(KDD '15)*, ACM, New York, pp. 1265–1274 (2015). https://doi.org/10.1145/2783258.2783395

Wimalawarne, K., Sugiyama, M., Tomioka, R.: Multitask learning meets tensor factorization: task imputation via convex optimization. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.), Advances in neural information processing systems 27, Curran Associates, Inc., 2825–2833 (2014). http://papers.nips.cc/paper/5628-multitask-learning-meets-tensor-factorization-task-imputation-via-convex-optimization.pdf

Wright, S., Nocedal, J.: Numerical optimization. Spring. Sci. **35**(67–68), 7 (1999)

Xu, Y., Zhang, L., Liu, W.: Cubic analysis of social bookmarking for personalized recommendation. In: Xiaofang, Z., Jianzhong, L., HengTao, Shen., Masaru, K., Yanchun, Z. (Eds.), Frontiers of WWW research and development-APWeb 2006, Lecture notes in computer science, Vol. 3841. Springer, Berlin, pp. 733–738 (2006). https://doi.org/10.1007/11610113_66

Yu, D., Deng, L., Seide, F.: Large vocabulary speech recognition using deep tensor neural networks. In: INTERSPEECH (2012)

Yu, Q.R., Liu, Y.: Learning from multiway data: simple and efficient tensor regression. arXiv:abs/1607.02535 (2016)

Yu, R., Li, G., Liu, Y.: Tensor regression meets gaussian processes. arXiv:1710.11345 (2017)

Yu, R., Zheng, S., Anandkumar, A., Yue, Y.: Long-term forecasting using tensor-train RNNs. (2018). https://openreview.net/forum?id=HJJ0w--0W

Zhang, Z., Batselier, K., Liu, H., Daniel, L., Wong, N.: Tensor computation: a new framework for high-dimensional problems in EDA. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **36**(4), 521–536 (2017)

Zhang, Z., Yang, X., Oseledets, I.V., Karniadakis, G.E., Daniel, L.: Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **34**(1), 63–76 (2015)

Zhao, Q., Caiafa, C.F., Mandic, D.P., Zhang, L., Ball, T., Schulze-bonhage, A., Cichocki, A.S.: Multilinear subspace regression: an orthogonal tensor decomposition approach. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q., (eds.), Advances in neural information processing systems 24, Curran Associates, Inc., 1269–1277 (2011). http://papers.nips.cc/paper/4328-multilinear-subspace-regression-an-orthogonal-tensor-decomposition-approach.pdf

Zhao, Y., Li, J., Liao, C., Shen, X.: Bridging the gap between deep learning and sparse matrix format selection. In: Proceedings of the 23rd ACM SIGPLAN symposium on principles and practice of parallel programming *(PPoPP '18)*, ACM, New York, pp. 94–108 (2018). https://doi.org/10.1145/3178487.3178495

Zhou, G., Cichocki, A., Xie, S.: Decomposition of big tensors with low multilinear rank. arXiv:abs/1412.1885 (2014)

Zhou, H., Li, L., Zhu, H.: Tensor regression with applications in neuroimaging data analysis. J. Am. Stat. Assoc. **108**(502), 540–552 (2013). https://doi.org/10.1080/01621459.2013.776499

Jiajia Li is a Research Scientist at High Performance Computing group of Pacific Northwest National Laboratory (PNNL), Richland, WA. Her current research emphasizes on optimizing tensor algorithms such as tensor decompositions and fundamental tensor operations especially for sparse data by utilizing various parallel architectures. She has received her Ph.D. degree in Computational Science and Engineering at Georgia Institute of Technology. She also has received a Ph.D. degree from Institute of Computing Technology at Chinese Academy of Sciences and her B.S. in Computational Mathematics from Dalian University of Technology.

Yuchen Ma is an under-graduate student at Zhuoyue Honors College, Hangzhou Dianzi University, China.

Xiaolong Wu is a Ph.D. student in Electrical and Computer Engineering at Purdue University. He has received his Master degree from the school of Computer science in Virginia Polytechnic Institute and State University.

Ang Li received his Bachelor degree from the CS department of Zhejiang University, Hangzhou, China, in 2010. From 2010 to 2012, he worked in industry as a software developer. In 2016, he received his joint two PhD degrees from the ECE department of National University of Singapore, Singapore and the EE department of Eindhoven University of Technology, Eindhoven, Netherlands. He is now working as a research scientist in the HPC group of Pacific Northwest National Laboratory, USA.

Kevin Barker Kevin is the Group Lead for the High-Performance Computing group in PNNL's Advanced Computing, Mathematics, and Data division. He received his Ph.D. in computer science from the College of William and Mary in 2004 in the area of adaptive and dynamic middle-ware for large-scale parallel applications. Since that time, he has been with the DOE National Laboratory complex working in the area of performance modeling, large-scale system design, and advanced architectures. Kevin's current research involves understanding the implications of emerging and novel computing architectures on workload performance, energy efficiency, and cybersecurity.