ParaSketch: Parallel Tensor Factorization via Sketching

Bo Yang * Ahmed Zamzam * Nicholas D. Sidiropoulos †

Abstract

Tensor factorization methods have gained increased popularity in the data mining community. A key feature that renders tensors attractive is the essential uniqueness (identifiability) of their decomposition into latent factors: this is crucial for *explanatory* data analysis – model uniqueness makes interpretations well grounded. In this work, we propose ParaSketch, a distributed tensor factorization algorithm that enables massive parallelism, to deal with large tensors. The idea is to compress/sketch the large tensor into multiple small tensors, decompose each small tensor, and combine the results to reconstruct the desired latent factors. Prior art in this direction entails potentially very high complexity in the (Gaussian) compression and final combining stages. Utilizing sketching matrices for compression, the proposed method greatly reduces compression complexity, and features much simpler combining. Moreover, theoretical analysis shows that the compressed tensors inherit latent identifiability under mild conditions, hence establishing correctness of the overall approach. Our approach to establish identifiability for the sketched tensor is original, and of interest in its own right.

1 Introduction

Tensors are natural generalization of matrices: whereas each matrix element is indexed by two indices, a tensor element is indexed by three or more indices. Tensor factorizations have long been an effective data analysis set of tools for Chemometrics and Psycometrics. They have also drawn much attention from the computer science community, and have been heavily used in Data Mining (DM) and Machine Learning (ML) since about a decade ago. Pioneering works in this application area include [9], where the authors applied tensor factorization methods to the problem of webpage link analysis, and [4] where tensor factorization methods were used to analyze social network graphs. Subsequently, tensor methods have been applied to recommender systems [8], topic modeling [2], and bioinfomatics [13], just to name a few. We refer the interested reader to the recent overview paper [14] for an in-depth review.

Perhaps the most important advantage of tensors over matrices lies in the *essential uniqueness* property of low-rank tensor factorizations: under mild conditions, the latent factor matrices of a low-rank tensor can be identified up to column scaling and permutation – which is drastically different from unconstrained matrix factorization models, where rotational ambiguity is usually present. Model uniqueness helps produce interpretable and insightful results in data mining.

One key challenge brought by applications in DM/ML is the sheer amount of data. For the Parallel Factor Analysis (PARAFAC) tensor factorization model, classical usages are mainly concerned with small datasets, where algorithms like alternating least squares (ALS) usually suffice. On the other hand, modern datasets, especially those of interest to ML/DM, are usually of large size. Performing PARAFAC at this scale is challenging. To tackle this problem, specialized toolbox [5] has been developed, and several algorithms have been recently proposed, including GigaTensor [7], DFacTo [6], PARACOMP [16], PARCUBE [11, 12].

GigaTensor proposed a way to avoid "intermediate data explosion" in using ALS to perform PARAFAC, and DFacTo proposed a way to exploit the inherent parallelism in ALS and gradient descent (GD) iterations. However, these methods operate on the *whole* tensor in each iterative step, which is still prohibitive when the tensor is very large. It is then of great interest to develop a "divide and conquer" strategy, where one first "breaks" the large tensor into small tensors, then finds factors of the small tensors, and then recovers factors of the large tensor from factors of the small tensors.

PARCUBE was the first such approach, which consists of three steps: (1) subsample the large tensor data into multiple small tensors, (2) perform PARAFAC on the small tensors, (3) combine the factors of the small tensors to recover the factors of the original large tensor, see Figure 1 for an illustration. PARCUBE offers great scalability, but its aim is to *approximate* the data tensor with sparse factors. For general tensors, it is unclear whether the sampled small tensors produced by PARCUBE are identifiable when the original tensor is identifiable. If the identifiability of the sampled small tensors cannot be established, the correctness in terms of recovering the true underlying factors in the final result will also be hard to assess.

PARACOMP is a similar technique, where the subsampling step in PARCUBE is replaced with random

*Electrical and Computer Engineering Department, University of Minnesota. Emails: {yang4173, ahmedz}@umn.edu, † Electrical and Computer Engineering Department, University of Virginia. Email: nikos@virginia.edu

projections. With this modification, strong identifiability guarantees for the compressed tensors and the original tensor were established [16]. Although PARACOMP enables massive parallelism that can facilitate large scale tensor factorization, there are two issues that limit its application. First, the compression step involves multiplication of dense matrices and tensors. Second, the combining step (post-processing after parallel PARAFAC) involves solving a large dense linear system, where no special structure can be exploited. These two issues can be a bottleneck when dealing with large tensors.

This seems to be a dilemma: Strong identifiability guarantee of PARACOMP comes from compression using dense random matrices, which however can become a performance bottleneck; whereas the scalability of PARCUBE comes from sampling, but theoretical guarantee of identifiability is hard to establish. We set out to address these issues. Specifically, we propose ParaSketch, a parallel PARAFAC algorithm that enjoys the following desirable properties:

- **Scalability**: The proposed method utilize sketching matrices to perform compression, which is much more computationally efficient compared to the compression with Gaussian random matrices adopted in PARACOMP. Moreover, with this new design, the final combining step only involves inverting *structured* matrices (Hadamard matrices), which is much simpler than solving a general linear system as in PARACOMP;

- **Identifiability**: With sketching matrices, we characterize conditions under which the compressed small tensors admit unique factorization, thus ensuring that one can correctly recover factors of the large tensor from those of the small tensors. The argument used to establish identifiability in PARACOMP cannot be applied here. To circumvent this, we provide a novel technique which judiciously exploits properties of sketching matrices.

We emphasize again that identifiability of the factorization of the compressed / sampled small tensors is crucial in this "divide and conquer" approach, because without it we cannot tell if the final combined result is correct.

To encourage *reproducibility*, the code and data for all our experiments will be made publicly available. The notation used in this work is summarized in Table 1.

Table 1: Summary of notations

Notations	Meaning
$\otimes$	Kronecker product
$\odot$	Khatri-Rao product
$\circ$	outer product
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$	vectors
$\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$	matrices
$\underline{\boldsymbol{X}}, \underline{\boldsymbol{Y}}$	tensors
$r(\boldsymbol{A})$, $r_{\boldsymbol{A}}$	rank of $\boldsymbol{A}$
$k(\boldsymbol{A})$, $k_{\boldsymbol{A}}$	Kruskal rank of $\boldsymbol{A}$

2 Background

2.1 PARAFAC and its uniqueness property

The PARAFAC model is defined as

$$\underline{\boldsymbol{X}} = \sum_{f=1}^{F} \boldsymbol{A}(:,f) \circ \boldsymbol{B}(:,f) \circ \boldsymbol{C}(:,f), \tag{2.1}$$

where F is the smallest integer such that the factorization (2.1) is possible, and is defined to be the rank of $\underline{\boldsymbol{X}}$. We use $\boldsymbol{A}(:,f)$ to denote the f-th column of $\boldsymbol{A}$. Henceforth, we use the shorthand notation $\underline{\boldsymbol{X}} = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$ to denote (2.1). For convenience, we will also make use of the vectorized form of tensors. Note that there are different ways to vectorize a tensor. We adopt the following definition of tensor vectorization

$$\boldsymbol{x}((i-1)JK + (j-1)K + k) = \underline{\boldsymbol{X}}(i,j,k),$$

where $\underline{\boldsymbol{X}} \in \mathbb{R}^{I \times J \times K}$, and $\boldsymbol{x} \in \mathbb{R}^{IJK \times 1}$. This vectorization corresponds to stacking vectors (a.k.a. fibers, in tensor literature) in the third mode. It can be easily checked that if $\underline{\boldsymbol{X}} = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$, we have $\boldsymbol{x} = (\boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{C})\mathbf{1}$, where $\mathbf{1}$ is an all-one vector of size $F \times 1$. For a more complete introduction on tensor vectorization, see e.g., [15].

One central concept in this work is uniqueness of PARAFAC, which is defined as follows.

Definition 1. *(Essential uniqueness) Given a tensor $\underline{\boldsymbol{X}}$ of rank F, we say that its PARAFAC decomposition is essentially unique if the rank-1 terms in the decomposition are unique, i.e., there is no other way to decompose $\underline{\boldsymbol{X}}$ for the given number of terms. Note that we can of course permute these terms without changing their sum, hence there exists an inherently unresolvable permutation ambiguity in the rank-1 tensors. If $\underline{\boldsymbol{X}} = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$, with $\boldsymbol{A} : I \times F$, $\boldsymbol{B} : J \times F$, and $\boldsymbol{C} : K \times F$, then essential uniqueness means that $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are unique up to a common permutation and*

scaling / counter-scaling of columns. In other words, if $\underline{\boldsymbol{X}} = [\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}}]$*, for some* $\tilde{\boldsymbol{A}} : I \times F$*,* $\tilde{\boldsymbol{B}} : J \times F$*, and* $\tilde{\boldsymbol{C}} : K \times F$*, then there exists a permutation matrix* $\boldsymbol{\Pi}$ *and diagonal scaling matrices* $\boldsymbol{\Lambda}_1$*,* $\boldsymbol{\Lambda}_2$*, and* $\boldsymbol{\Lambda}_3$ *such that*

$$\tilde{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}_1,\ \tilde{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{\Pi}\boldsymbol{\Lambda}_2,\ \tilde{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{\Pi}\boldsymbol{\Lambda}_3,\ \boldsymbol{\Lambda}_1\boldsymbol{\Lambda}_2\boldsymbol{\Lambda}_3 = \boldsymbol{I}.$$

To present identifiability results of PARAFAC, we need the definition of Kruskal rank of a matrix.

DEFINITION 2. *(Kruskal rank) The Kruskal rank* $k_{\boldsymbol{A}}$ *of an* $I \times F$ *matrix* $\boldsymbol{A}$ *is thelargest integer* k *such that any* k *columns of* $\boldsymbol{A}$ *are linearly independent.*

Clearly, $k_{\boldsymbol{A}} \leq r_{\boldsymbol{A}} := \text{rank}(\boldsymbol{A}) \leq \min(I, F)$. We next state the following classical result on identifiability of PARAFAC.

THEOREM 2.1. *(Identifiability of PARAFAC, [10]) Given* $\underline{\boldsymbol{X}} = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$*, with* $\boldsymbol{A} : I \times F$*,* $\boldsymbol{B} : J \times F$*, and* $\boldsymbol{C} : K \times F$*, if* $k_{\boldsymbol{A}} + k_{\boldsymbol{B}} + k_{\boldsymbol{C}} \geq 2F + 2$*, then rank*$(\underline{\boldsymbol{X}}) = F$ *and the decomposition of* $\underline{\boldsymbol{X}}$ *is essentially unique.*

Theorem 2.1 asserts that under some conditions, the factors of a tensor is identifiable. The identifiability property has far-reaching consequences, especially in explanatory data analysis applications, where one is interested in uncovering the true underlying factors in data.

2.2 Identifiability of PARAFAC with random compression To confront the challenges brought by large scale data analysis, it is desirable to develop "divide and conquer" strategies for large scale tensor factorization. PARACOMP [16] is such a solution: randomly compress the tensor multiple times in parallel, perform independent PARAFAC on the compressed small tensors, and combine the results to get PARAFAC results for the original large tensor. To illustrate the idea, we consider a 3-way tensor $\underline{\boldsymbol{X}} \in \mathbb{R}^{I \times J \times K}$. Reference [16] propose to compress the 3 modes of the tensor independently, with 3 matrices $\boldsymbol{U} \in \mathbb{R}^{L \times I}$, $\boldsymbol{V} \in \mathbb{R}^{M \times J}$ and $\boldsymbol{W} \in \mathbb{R}^{N \times K}$, where $L \leq I$, $M \leq J$, and $N \leq K$. Suppose the original tensor $\underline{\boldsymbol{X}}$ has latent factors $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$, i.e. $\boldsymbol{x} = (\boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{C})\mathbf{1}$, where $\boldsymbol{x} = \text{vec}(\underline{\boldsymbol{X}})$. The compressed tensor $\boldsymbol{y}$ can be written as [15]

$$\begin{aligned} \boldsymbol{y} &= (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})(\boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{C})\mathbf{1} \\ &= ((\boldsymbol{U}\boldsymbol{A}) \odot (\boldsymbol{V}\boldsymbol{B}) \odot (\boldsymbol{W}\boldsymbol{C}))\,\mathbf{1}. \end{aligned} \tag{2.2}$$

To materialize this compression and factorization scheme, a key issue that needs to be addressed is under what conditions are the latent factors of the compressed tensor identifiable. The following theorem answers this question.

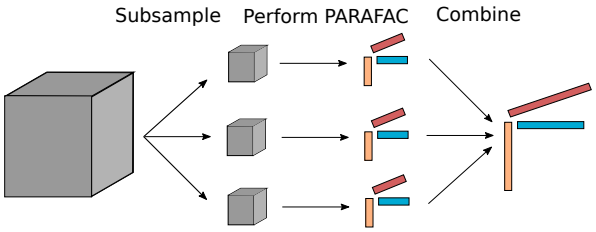


Figure 1: Schematic of PARCUBE [11]

THEOREM 2.2. *(Identifiability of compressed tensor, [16]) Let* $\boldsymbol{x} = (\boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{C})\mathbf{1} \in \mathbb{R}^{IJK}$*, where* $\boldsymbol{A}$ *is* $I \times F$*,* $\boldsymbol{B}$ *is* $J \times F$*,* $\boldsymbol{C}$ *is* $K \times F$*, and consider compressing it to* $\boldsymbol{y} = (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\boldsymbol{x} = ((\boldsymbol{U}\boldsymbol{A}) \odot (\boldsymbol{V}\boldsymbol{B}) \odot (\boldsymbol{W}\boldsymbol{C}))\,\mathbf{1} = (\tilde{\boldsymbol{A}} \odot \tilde{\boldsymbol{B}} \odot \tilde{\boldsymbol{C}})\mathbf{1} \in \mathbb{R}^{LMN}$*, where the mode-compression matrices* $\boldsymbol{U}(L \times I, L \leq I)$*,* $\boldsymbol{V}(M \times J, M \leq J)$*, and* $\boldsymbol{W}(N \times K, N \leq K)$ *are independently drawn from an absolutely continuous distribution. If*

$$\min(L, k_{\boldsymbol{A}}) + \min(M, k_{\boldsymbol{B}}) + \min(N, k_{\boldsymbol{C}}) \geq 2F + 2, \tag{2.3}$$

then $\tilde{\boldsymbol{A}}$*,* $\tilde{\boldsymbol{B}}$*,* $\tilde{\boldsymbol{C}}$ *are almost surely identifiable from the compressed data* $\boldsymbol{y}$ *up to a common column permutation and scaling.*

Comparing Theorem 2.2 with Theorem 2.1, one notes that if $L \geq k_{\boldsymbol{A}}$, $M \geq k_{\boldsymbol{B}}$, and $N \geq k_{\boldsymbol{C}}$, then the condition for uniqueness is the same in both theorems. This means that if the compression matrices $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{W}$ are drawn from absolutely continuous distribution, and have enough rows, then the compression process preserves identifiability. The requirement that compression matrices have to be drawn from absolutely continuous distribution leads to unstructured and dense matrices, like the Gaussian random matrices used in [16], which pose computational challenges.

PARACOMP is the first example of how one can carry out PARAFAC in a parallel fashion, while still being able to recover the correct result. However, as note by the authors of [16], the compression can be expensive since all the compression matrices are dense and unstructured. Even if the original tensor is sparse, after compression on the first mode, the result will be dense, and all the subsequent compression on other modes will need to be operated in the dense regime. Overall, the computation complexity is $\mathcal{O}(\min(L, M, N)IJK)$, which can be prohibitive even for moderate tensors, say $I = J = K = 1000$.

It would be interesting then, to seek matrices that can be multiplied to the tensor faster than the plain dense matrix-tensor multiplication. Now the question is, how to construct such compression matrices, so that one can still guarantee identifiability of the compressed

tensors, with reasonable assumptions?

2.3 Fast Johnson Lindenstrauss Transform To answer the above question, we introduce the following Subsampled Randomized Hadamard Transform (SRHT), a fast Johnson-Lindenstrauss Transform (JLT). For an overview on JLT and its variants, please see [18].

PROPOSITION 1. *(SRHT, [1,18]) Let* $\boldsymbol{S} = \frac{1}{\sqrt{IL}}\boldsymbol{P}\boldsymbol{H}_I\boldsymbol{D}$, *where* $\boldsymbol{D}$ *is an* $I \times I$ *diagonal matrix with i.i.d. diagonal entries* $\boldsymbol{D}_{ii} = 1$ *with probability 1/2, and* $\boldsymbol{D}_{ii} = -1$ *with probability 1/2.* $\boldsymbol{H}_I$ *is a Hadamard matrix of size* I*, which is assumed to be a power of 2. The* $L \times I$ *matrix* $\boldsymbol{P}$ *samples* L *coordinates of an* I*-dimensional vector uniformly at random, where*

$$L = \Omega(\epsilon^{-2}(\log F)(\sqrt{F} + \sqrt{\log I})^2).$$

Then with probability at least 0.99, for any fixed $\boldsymbol{U} \in \mathbb{R}^{I \times F}$ *with orthonormal columns,*

$$\|\boldsymbol{I}_L - \boldsymbol{U}^T\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{U}\|_2 \leq \epsilon. \tag{2.4}$$

Moreover, for any vector $\boldsymbol{x} \in \mathbb{R}^I$, $\boldsymbol{S}\boldsymbol{x}$ *can be computed in* $\mathcal{O}(I \log L)$ *time.*

A proof of this result can be found in [1].

3 Proposed approach

3.1 Identifiability of compressed tensors with SRHT matrices Consider a 3-way tensor $\underline{\boldsymbol{X}} := [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$ with unique low-rank factorization, the following statement gives conditions on identifiability of sketched tensors.

THEOREM 3.1. *(main result) For tensor* $\boldsymbol{x} = (\boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{C})\mathbf{1}$, *the sketched tensor* $\boldsymbol{y} = (\boldsymbol{S}^a \otimes \boldsymbol{S}^b \otimes \boldsymbol{S}^c)(\boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{C})\mathbf{1}$ *is identifiable with high probability if* $\boldsymbol{S}^a \in \mathbb{R}^{L \times I}$, $\boldsymbol{S}^b \in \mathbb{R}^{M \times J}$, $\boldsymbol{S}^c \in \mathbb{R}^{N \times K}$ *are all SRHT matrices, with number of rows* $L = \mathcal{O}(C_1F/\epsilon_1^2)$, $M = \mathcal{O}(C_2F/\epsilon_2^2)$, $N = \mathcal{O}(C_3F/\epsilon_3^2)$, *where* C_1, C_2, C_3 *are constants and* $\{\epsilon_1, \epsilon_2, \epsilon_3\} \in (0, 1)$ *are the accuracy parameters in* (2.4)*, and if the following condition is satisfied*

$$\min(L, k_{\boldsymbol{A}}) + \min(M, k_{\boldsymbol{B}}) + \min(N, k_{\boldsymbol{C}}) \geq 2F + 2. \tag{3.5}$$

This result is similar to Theorem 4 of [16]. The key difference, however, is that we replaced the dense compression matrices with sketching matrices. This seemingly small change has a tremendous impact on the computation complexity, see Table 2 and Figure 2b. Moreover, the arguments used in [16] to establish unique factorization of compressed tensors can not be applied here, as they rely on compression matrices drawn from absolutely continuous distributions, which is not the case for SRHT matrices.

To prove Theorem 3.1, we provide the following lemma. Consider a sketching matrix $\boldsymbol{S} \in \mathbb{R}^{L \times I}$ and a matrix $\boldsymbol{A} \in \mathbb{R}^{I \times F}$, we show the following result on the Kruskal rank of the sketched matrix.

LEMMA 3.1. *Suppose* S *is an SRHT matrix, then with high probability,* $k(\boldsymbol{S}\boldsymbol{A}) = \min(L, k_{\boldsymbol{A}})$.

Proof: First, let us consider the case $L \geq k_{\boldsymbol{A}}$, we need to show $k(\boldsymbol{S}\boldsymbol{A}) = k_{\boldsymbol{A}}$. Consider an arbitrary $k_{\boldsymbol{A}}$ columns of $\boldsymbol{S}\boldsymbol{A}$, and let $\boldsymbol{A}_s$ denotes the corresponding $k_{\boldsymbol{A}}$ columns of $\boldsymbol{A}$, we need to show that $r(\boldsymbol{S}\boldsymbol{A}_s) = k_{\boldsymbol{A}}$. Towards this end, consider the reduced SVD of $\boldsymbol{A}_s$, $\boldsymbol{A}_s = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}$, where $\boldsymbol{U} \in \mathbb{R}^{I \times k_{\boldsymbol{A}}}$, $\boldsymbol{V} \in \mathbb{R}^{k_{\boldsymbol{A}} \times k_{\boldsymbol{A}}}$, and $\boldsymbol{\Sigma} \in \mathbb{R}^{k_{\boldsymbol{A}} \times k_{\boldsymbol{A}}}$. We show that the matrix $\boldsymbol{S}\boldsymbol{U}$ is of full rank with high probability.

By definition of $\boldsymbol{S}$, with high probability, we have

$$\|\boldsymbol{I}_{k_{\boldsymbol{A}}} - \boldsymbol{U}^\mathsf{T}\boldsymbol{S}^\mathsf{T}\boldsymbol{S}\boldsymbol{U}\|_2 < \epsilon. \tag{3.6}$$

This suggests that $1 - \epsilon < \lambda_i < 1 + \epsilon$, $\forall i$, where λ_i's are the eigenvalues of $\boldsymbol{U}^\mathsf{T}\boldsymbol{S}^\mathsf{T}\boldsymbol{S}\boldsymbol{U}$. Now since $\epsilon \in (0, 1)$, all the eigenvalues of matrix $\boldsymbol{U}^\mathsf{T}\boldsymbol{S}^\mathsf{T}\boldsymbol{S}\boldsymbol{U}$ are positive. It follows that all the singular values of $\boldsymbol{S}\boldsymbol{U}$ are positive, and hence $\boldsymbol{S}\boldsymbol{U}$ is of full rank. With Sylvester's rank inequality and the fact that $\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}$ has full column rank, we can see that $r(\boldsymbol{S}\boldsymbol{A}_s) = r(\boldsymbol{S}\boldsymbol{U}) = k_{\boldsymbol{A}}$ by first noting that

$$\begin{aligned} r(\boldsymbol{S}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}) &\geq r(\boldsymbol{S}\boldsymbol{U}) + r(\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}) - k_{\boldsymbol{A}} \\ &= r(\boldsymbol{S}\boldsymbol{U}) + k_{\boldsymbol{A}} - k_{\boldsymbol{A}} \\ &= r(\boldsymbol{S}\boldsymbol{U}). \end{aligned}$$

But we also have $r(\boldsymbol{S}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}) \leq r(\boldsymbol{S}\boldsymbol{U})$, hence $r(\boldsymbol{S}\boldsymbol{A}_s) = r(\boldsymbol{S}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}) = r(\boldsymbol{S}\boldsymbol{U}) = k_{\boldsymbol{A}}$. Since any $k_{\boldsymbol{A}}$ columns of $\boldsymbol{S}\boldsymbol{A}$ are linearly independent, $k(\boldsymbol{S}\boldsymbol{A}) = k_{\boldsymbol{A}}$.

Now, consider the case $L < k_{\boldsymbol{A}}$, we then consider any arbitrary L columns of $\boldsymbol{S}\boldsymbol{A}$, and similarly denote the corresponding columns in $\boldsymbol{A}$ to be $\boldsymbol{A}_s$. By definition of Kruskal rank, $\boldsymbol{A}_s$ will be full rank, otherwise $k_{\boldsymbol{A}} < L$. Similar as above, we can show that $r(\boldsymbol{S}\boldsymbol{A}_s) = L$. We then conclude that $k(\boldsymbol{S}\boldsymbol{A}) = \min(L, k_{\boldsymbol{A}})$ with high probability.

REMARK 3.1. *In numerical linear algebra, sketching matrices were originally designed to perform subspace embedding. That is, one is interested in finding a sketching matrix* $\boldsymbol{S}$*, such that the norm of vectors that live in a certain subspace is preserved after embedding. More concretely*

$$(1 - \epsilon)\|\boldsymbol{y}\|_2^2 \leq \|\boldsymbol{S}\boldsymbol{U}\boldsymbol{y}\|_2^2 \leq (1 + \epsilon)\|\boldsymbol{y}\|_2^2, \ \forall \boldsymbol{y} \in \mathbb{R}^d \tag{3.7}$$

where $\boldsymbol{U} \in \mathbb{R}^{n\times d}$, a tall matrix with unit-length and orthogonal columns, represents a subspace. The approximation error ϵ affects the number of rows in $\boldsymbol{S}$, i.e. $\mathcal{O}(CF/\epsilon^2)$ if the SRHT sketching method is adopted. If a high approximation accuracy is needed, e.g. $\epsilon = 0.001$, then one will need undesirably many rows in $\boldsymbol{S}$. However, in establishing identifiability, (see the above proof of Lemma 3.1), we only need $1 - \epsilon > 0$. In the experiment section, we show empirically that we do not need many rows to correctly recover the factors.

Proof of Theorem 3.1: By Lemma 3.1, we have that $k(\boldsymbol{S}^a\boldsymbol{A}) = \min(L, k_{\boldsymbol{A}})$, $k(\boldsymbol{S}^b\boldsymbol{B}) = \min(M, k_{\boldsymbol{B}})$, and $k(\boldsymbol{S}^c\boldsymbol{C}) = \min(N, k_{\boldsymbol{C}})$. Then Theorem 3.1 follows directly from Kruskal's condition on identifiability of PARAFAC models [10].

3.2 Combining the results The combining process mainly follows that of [11,16]. However, we give special treatment of this process so that the structure of the proposed compression matrices can be fully exploited, resulting in much lighter computations. We illustrate the combining process using factor $\boldsymbol{A}$, and the other factors can be combined following the same procedure. To avoid clutter, we denote the sketching matrix for the first mode as $\boldsymbol{S}$, instead of $\boldsymbol{S}^a$ as above.

Suppose we spawn T processes to compress the tensor, with each compressed tensor denoted as $\boldsymbol{Y}_t$, $t \in \{1, \cdots, T\}$. We next perform PARAFAC on each of these small tensors, and obtain $[\boldsymbol{A}_t, \boldsymbol{B}_t, \boldsymbol{C}_t]$, $t \in \{1, \cdots T\}$. Then we can write

$$\tilde{\boldsymbol{A}}_t = \boldsymbol{S}_t\boldsymbol{A}\boldsymbol{\Pi}_t\boldsymbol{\Lambda}_t, \tag{3.8}$$

which suggests that even though we can guarantee each $\tilde{A}_t$ to be identifiable, we still have indeterminacy in permutation (e.g. $\boldsymbol{\Pi}_t$) and scaling (e.g. $\boldsymbol{\Lambda}_t$) to address. Only then can we combine the results and recover the true factors $[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$.

To resolve permutation ambiguity, we keep two common rows in each compression matrix $\boldsymbol{S}$, denote this part as $\overline{\boldsymbol{S}}$, then the corresponding part in factor $\boldsymbol{A}$ is

$$\overline{\boldsymbol{A}} = \overline{\boldsymbol{S}}\boldsymbol{A}\boldsymbol{\Pi}_t\boldsymbol{\Lambda}_t. \tag{3.9}$$

Dividing each column of $\overline{\boldsymbol{A}}$ by its largest magnitude element, we get

$$\widehat{\boldsymbol{A}}_t = \overline{\boldsymbol{S}}\boldsymbol{A}\boldsymbol{\Pi}_t\boldsymbol{\Lambda}. \tag{3.10}$$

One can see that scaling ambiguity is fixed: each replica now has a common scaling matrix $\boldsymbol{\Lambda}$. Next, we will match each replica to the first copy $\widehat{\boldsymbol{A}}_1$, i.e., we will solve the following problem

$$\min_{\boldsymbol{\Pi}\in\mathcal{P}_F} \|\widehat{\boldsymbol{A}}_1 - \widehat{\boldsymbol{A}}_t\boldsymbol{\Pi}\|_{\mathrm{F}}^2, \tag{3.11}$$

where we use $\mathcal{P}_F$ to denote the set of permutation matrices of size $F \times F$. With a little manipulation, problem (3.11) is equivalent to

$$\max_{\boldsymbol{\Pi}\in\mathcal{P}_F} \ \mathrm{tr}\left(\widehat{\boldsymbol{A}}_1^{\mathsf{T}}\widehat{\boldsymbol{A}}_t\boldsymbol{\Pi}\right). \tag{3.12}$$

Problem (3.12) is known as the Linear Assignment Problem (LAP), and can be solved efficiently by the Hungarian algorithm.

With a little abuse of notation, we denote the resulting permutation matrix from (3.12) as $\boldsymbol{\Pi}_t$, $t \in \{2, \cdots, T\}$. We use these $\boldsymbol{\Pi}_t$ to permute the results $\tilde{\boldsymbol{A}}_t$ in (3.8), and get the following

$$\check{\boldsymbol{A}}_t = \boldsymbol{S}_t\boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}_t. \tag{3.13}$$

Now the only remaining ambiguity is scaling $\boldsymbol{\Lambda}_t$, which can be removed by dividing $\check{\boldsymbol{A}}_t$ with one of the common rows. After this step, we get

$$\check{\boldsymbol{A}}_t = \boldsymbol{S}_t\boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}. \tag{3.14}$$

3.3 Solving for the factors After fixing scaling and permutation, we can combine the result as follow

$$\begin{bmatrix} \check{\boldsymbol{A}}_1 \\ \check{\boldsymbol{A}}_2 \\ \vdots \\ \check{\boldsymbol{A}}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_1 \\ \boldsymbol{S}_2 \\ \vdots \\ \boldsymbol{S}_T \end{bmatrix} \boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}. \tag{3.15}$$

The special construction of our sketching matrices allows us to write

$$\begin{aligned} \begin{bmatrix} \check{\boldsymbol{A}}_1 \\ \check{\boldsymbol{A}}_2 \\ \vdots \\ \check{\boldsymbol{A}}_T \end{bmatrix} &= \begin{bmatrix} \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \\ \vdots \\ \boldsymbol{P}_T \end{bmatrix} \boldsymbol{H}\boldsymbol{D}\boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda} \\ &:= \widehat{\boldsymbol{P}}\boldsymbol{H}\boldsymbol{D}\boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}, \end{aligned} \tag{3.16}$$

where we have defined $\widehat{\boldsymbol{P}}$ as the concatenation of $\boldsymbol{P}_t$'s. It can be easily seen that inverting $\widehat{\boldsymbol{P}}$ is simple, since each $\boldsymbol{P}_t$ is a sampling matrix, i.e. each row of $\boldsymbol{P}_t$ contains only one nonzero value. After inverting $\widehat{\boldsymbol{P}}$, we can invert $\boldsymbol{H}$ using the fast inverse Hadamard transform, and invert $\boldsymbol{D}$ with a simple scaling operation. The most expensive part is the inverse Hadamard transform operation, which cost $\mathcal{O}(I\log(I)F)$.

The overall algorithm is summarized in Algorithm 1. The combining step is presented in Algorithm 2.

REMARK 3.2. (Number of replicas needed) *By our construction, each compression shares $\boldsymbol{H}$ and $\boldsymbol{D}$. To ensure inversion in* (3.16) *can be carried out, we need each*

Algorithm 1 ParaSketch

Input: Data tensor $\underline{\boldsymbol{X}}$, number of replicas T, compression dimension L, M, N, PARAFAC rank F.
Output: Factors $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$.

1: Perform T compressions as described in Sec. 3.1, yielding small tensors $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_T\}$;
2: Perform PARAFAC on the compressed tensors, yielding $\{\{\boldsymbol{A}_1, \boldsymbol{B}_1, \boldsymbol{C}_1\}, \cdots, \{\boldsymbol{A}_T, \boldsymbol{B}_T, \boldsymbol{C}_T\}\}$;
3: Perform combining procedure to get $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ as described in Algorithm. (2).

Algorithm 2 Combining procedure

Input: $\{\boldsymbol{A}_t\}$, $t \in 1, \cdots, T$
Output: $\boldsymbol{A}$

1: Determine permutation matrices for each compressed small tensors by solving (3.11), and permute the factors in (3.8) to obtain (3.13);
2: Resolve scaling ambiguity by dividing entries in each column with values in a common row;
3: Inverting $\widehat{\boldsymbol{P}}$, $\boldsymbol{H}$, $\boldsymbol{D}$ in (3.16) in turn.

row index in matrix $\boldsymbol{HDA\Pi\Lambda}$ to be sampled at least once. That is, $\widehat{\boldsymbol{P}}$ cannot have a column that is all zero. Since all the sampling is performed independently and uniformly, this sampling procedure corresponds to the famous Coupon Collector's Problem. In this problem, it is known that one needs to try $\mathcal{O}(n\log(n))$ times in order to collect n coupons. In our problem, this means that the number of replicas T is in the order $\mathcal{O}(I\log(I)/L)$.

REMARK 3.3. (Data compression ratio) *Suppose the original large data tensor has size $I \times I \times I$, and each mode is compressed to dimension L, then data compression ratio is $I^3/(TL^3)$. As argued above, we need T to be $T = \eta I\log(I)/L$, where η is some constant. In this case, the compression ratio can be expressed as $I^2/(\eta L^2 \log I)$. In our simulation, we observed that once η exceeds 1.5, we get exact recovery (up to numerical precision) of the factors in most cases, suggesting that compression ratio can be very high when $L \ll I$.*

3.4 Computational complexity analysis To make a clear comparison, we focus complexity analysis on the first mode. In PARACOMP, to ensure the first mode factor of the original tensor can be recovered from that of the compressed tensors, one needs number of replicas to be $T \geq I/L$. Note that this means PARACOMP needs a $\log(I)$ factor fewer replicas. However, the cost of multiplying a dense compression matrix of size $I \times L$ with a dense tensor of size $I \times J \times K$ is $\mathcal{O}(LIJK)$. In our construction, applying the compression costs $\mathcal{O}(I\log(L)JK)$. The total number of rows in PARACOMP is $\mathcal{O}(I)$, hence the total computations are $\mathcal{O}(I^2JK)$. The complexity of ParaSketch is dominated by the Hadamard transform, which takes time $\mathcal{O}(I\log I)$ for a length I vector, and $\mathcal{O}(IJK\log I)$ for the whole tensor.

Table 2: Complexity comparison

	PARACOMP	ParaSketch
Compression	$\mathcal{O}(I^2JK)$	$\mathcal{O}(IJK\log I)$
Combining	$\mathcal{O}(I^3)$	$\mathcal{O}(FI\log I)$

In the final combining procedure, our method costs $\mathcal{O}(I\log(I)F)$, while PARACOMP costs $\mathcal{O}(I^3)$ in general. Note that in theory one can invoke the property that a random Gaussian matrix is approximately orthogonal with large I, and replace the inversion with transpose, resulting a lower cost of $\mathcal{O}(I^2F)$ – but solution accuracy is likely to deteriorate if such an approximation is adopted. Complexity comparison of PARACOMP and ParaSketch is summarized in Table 2, see Figure 2b for numerical experiment results.

4 Related works

4.1 GigaTensor and DFacTo The alternating least square (ALS) algorithm is widely adopted in performing PARAFAC. One intermediate step of ALS involves performing a matricized-tensor-times-Khatri-Rao-product (abbreviated as MTTKRP in literature, see [5]). Directly performing MTTKRP will result in the so called "internal data explosion", where the intermediate data size is so large that hinders scalability. GigaTensor [7] proposed a way to mitigate the internal data explosion problem in tensor factorization. The main idea is to avoid forming the intermediate large matrix, and performing multiplication by first calculate the indices of elements.

DFacTo [6] builds upon GigaTensor, and proposed a way to exploit the inherent parallelism in MTTKRP.

4.2 Sketching for symmetric tensors Reference [17] proposed to utilize sketching technique to scale up factorization for symmetric tensors. Specifically, the problem considered in [17] is to factor tensors that are *presented* as sum of rank-1 symmetric tensors. As an example, empirical moment tensors $\underline{\boldsymbol{X}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{u}_i \circ \boldsymbol{u}_i \circ \boldsymbol{u}_i$, where $\boldsymbol{u}_i$ is a data sample and N is the total number of samples, are presented as sum of rank-1 tensors if we are presented with data samples $\{\boldsymbol{u}_i\}_{i=1}^N$.

We consider the general case of possibly fully asymmetric tensors. Unlike [17], we prove identifiability of the compressed and the original tensor factorizations. Our method can be applied to symmetric tensors, and

we do not assume that data tensor is symmetric or presented in sum of rank-1 tensors form.

4.3 PARCUBE The PARCUBE work [12] is related to this work. It also comprises 3 steps: (1) performing parallel sampling on the large tensor based on a notion of importance, (2) factoring the sampled small tensors, and (3) combining the resulting factors of the small tensors to form the desired factors of the large tensor.

The present paper differs from [12] in that we provide exact characterization of when will the sketched small tensors be identifiable, and this is where [12] falls short. Indeed, for PARCUBE, the focus is to approximate a data tensor, and the subsampled small tensors may lose identifiability in the general case. Note that the success of recovering factors of the large tensor hinges on the success of recovering factors of *all* the small tensors: if one small tensor factorization fails, the whole process fails. It is thus crucial that we have guarantee on whether the individual factorization will be successful, as discussed in this work.

5 Numerical results

In this section, the performance of our presented approach for different problem setups is measured. To test factor estimation accuracy, we compare the proposed ParaSketch with PARACOMP. PARCUBE is excluded from comparison since it is designed to approximate a tensor, not to recover the true underlying factors. We also include direct PARAFAC (without any compression) as a baseline, for comparison purposes.

We generate synthetic data to test factor estimation accuracy and run time performance. Specifically, we generate factors $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}$, and synthesis them (see (2.1)) into a tensor $\underline{\boldsymbol{X}}$, which is then fed into different algorithms to estimate $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}$. For factor estimation, the performance is measured by normalized mean-squared-error (NMSE), which is defined as

(5.17)
$$\text{NMSE} = \min_{\substack{\pi\in\Pi,\\ c_f\in\{1,-1\}}} \frac{1}{F}\sum_{f=1}^{F}\left\| \frac{\boldsymbol{A}_f}{\|\boldsymbol{A}_f\|_2} - c_f \frac{\widehat{\boldsymbol{A}}_{\pi_f}}{\|\widehat{\boldsymbol{A}}_{\pi_f}\|_2}\right\|_2^2,$$

where $\widehat{\boldsymbol{A}}$ is the estimated factor, $\boldsymbol{A}$ is the ground truth factor, and Π is the set of all permutations of the set $\{1, 2, \cdots, F\}$. $\boldsymbol{A}_f$ denotes f-th column of $\boldsymbol{A}$, and c_f is there to resolve the sign ambiguity.

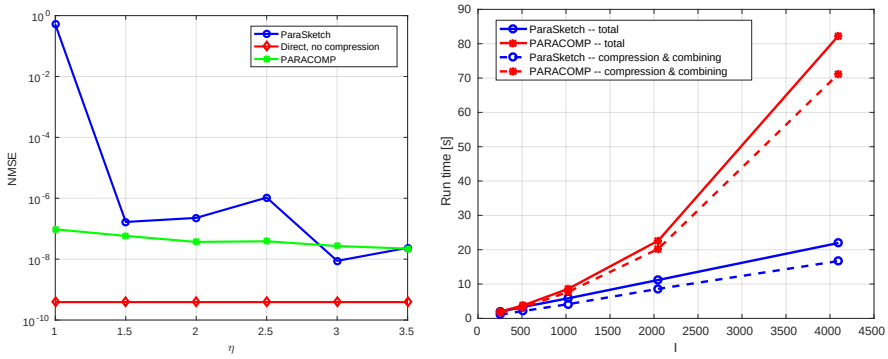
5.1 Factor estimation accuracy In the first experiment, we generate noisy data with additive Gaussian noise. The dimensions of the considered tensor are $I = 512$, $J = 512$, $K = 512$, and $F = 10$. The replicas dimensions are fixed to $L = 64$, $M = 64$, $N = 64$. The noise power is set to $\sigma = 0.01$. Note that this is a dense tensor, which has about 0.13 *billion* entries. Also note that we create moderate tensors to facilitate comparison with the un-scalable, direct PARAFAC. The number of common rows in each replica is set to 3. The factors $\boldsymbol{A}$ is generated in MATLAB as `randn`(I, F), and $\boldsymbol{B}$, $\boldsymbol{C}$ are generated in a similar fashion. We vary the number of replicas T such that $\eta = \frac{TL}{I\log(I)}$ takes values between 1 and 3.5. For each parameter configuration, 50 randomly generated problem instances are created, and the result is averaged across these instances.

The results are shown in Figure 2a. As can be seen, the proposed ParaSketch performs as well as PARACOMP once enough replicas are employed. As we point out in Remark 3.3, the data compression ratio is $\frac{I^2}{\eta L^2 \log I}$. In this experiment, we have $I = 512, L = 64$, the compression ratio is $\frac{I^2}{\eta L^2 \log I} = 3.55$ when $\eta = 2$. This means when only 28% of the original amount of data is used for factorization, an accurate estimation of the factors is achieved. Note that the proposed ParaSketch achieves similar factor estimation performance as PARACOMP, at much lighter computation cost (see Table 2 and Figure 2b).

5.2 Run time comparison In the second experiment, we compare the run time performance of the proposed ParaSketch and PARACOMP – the two methods that offer identifiability guarantees for the compressed tensors. Specifically, we vary dimension I and fix $J = 256$, $K = 256$ when generating data. Accordingly, the compression dimensions are set to $M = N = 64$, and $L = I/8$. The rank is set to $F = 10$, and the number of common rows is set to 3. For ParaSketch, the ratio is fixed to $\eta = 1.8$. Recall that the number of replicas for ParaSketch will be $T_{\text{ParaSketch}} = \eta(I \log I/L)$. For PARACOMP, we set $T_{\text{PARACOMP}} = 3 \times I$, since we observed that this gives good factor estimation accuracy. We conduct our experiments on a desktop with 16 parallel processors, and 128 GB RAM. To factor the compressed small tensors, we use the same PARAFAC solver provided in the N-way toolbox[1] [3] for both ParaSketch and PARACOMP. We repeat each experiment 10 times with different randomly generated data, and report the average run time. In addition to the total run time, we also record and report the part of time spent on compression and combining stages of both the two methods.

The results are shown in Figure 2b. As we can see, the proposed ParaSketch scales much better compared with PARACOMP. More importantly, we can see that for PARACOMP, most of the time is spent on the compression and combining part. This highlights the

[1] http://www.models.life.ku.dk/nwaytoolbox



(a) The NMSE of estimating $\boldsymbol{A}$ (b) Run time comparison

Figure 2: Experiment results with simulated data

merit of the propose method: For large tensors, compression and combining can be a performance bottleneck for PARACOMP, and using SRHT matrices to perform compression can greatly alleviate this issue. Also note that the runtime results in Figure 2b verifies the complexity analysis presented in Table 2.

5.3 Real world data mining We also test our algorithm on a dataset of taxi trajectories in Beijing during the Chinese New Year (a major Chinese festival) week of 2008 [19]. We construct the tensor by discretizing the latitude and longitude to a 128×128 grid, and considering the time as the third dimension of the tensor. Therefore, each element in the tensor is an integer that represents the number of taxis that were at the corresponding area at a specific moment of time. Then, the ParaSketch algorithm is used to find the low rank components of the $128 \times 128 \times 8980$ tensor. We use 200 replicas of dimensions $16 \times 16 \times 256$ in order to perform ParaSketch. The ParaSketch algorithm finds the 5 most significant factors in the dataset.

In Figure 3, we visualize the most significant factor of the result. In the top subfigure, the three markers corresponds to the three largest values of the factor. These locations corresponds to an area known for some famous attractions: Beijing Zoo, Yuyuan Pond Park, and Temple of Moon Park. The high intensity of taxi activities verifies their popularity. In the bottom subfigure, we show the corresponding column in the third mode, which contains temporal information. As expected, high activities are observed during the day, and relatively low activities in the night. More interestingly, we can see that there is a decline in taxi activity for the last 3 days, which correspond to the New Year's Eve (Day 5), Spring Festival (Day 6, 7) – those are days for family reunions when a lot of people stay home.

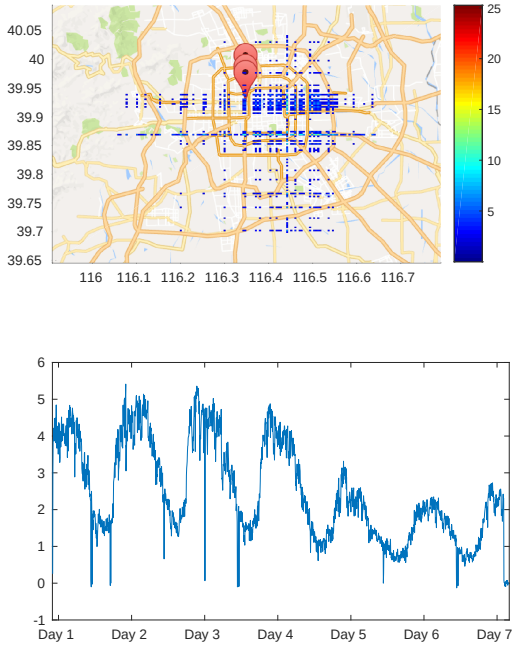


Figure 3: Beijing taxi trajectory analysis

6 Conclusions

We proposed an algorithm to enable *parallel* PARAFAC on large tensor data. Our approach provides great acceleration over existing prior art, which allows this method to be applied to much larger datasets. Our analysis establishes the correctness of the proposed algorithm, i.e., identifiability of the latent factors of the compressed tensors and the original uncompressed tensor. We also characterized recovery conditions for the proposed approach, i.e. number of rows in the sketching matrices and number of replicas needed, to ensure recovery of the factors of the large tensor data. Numerical results on synthetic as well as real world data confirm the efficacy of the proposed method.

References

[1] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM Symposium on Theory of Computing*, pages 557–563. ACM, 2006.

[2] Animashree Anandkumar, Rong Ge, Daniel J Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

[3] Claus A Andersson and Rasmus Bro. The N-way toolbox for MATLAB. *Chemometrics and intelligent laboratory systems*, 52(1):1–4, 2000.

[4] Brett W Bader, Richard A Harshman, and Tamara G Kolda. Temporal analysis of semantic graphs using ASALSAN. In *Proceedings of Seventh IEEE International Conference on Data Mining*, pages 33–42. IEEE, 2007.

[5] Brett W Bader and Tamara G Kolda. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, 2007.

[6] Joon Hee Choi and S Vishwanathan. DFacTo: Distributed factorization of tensors. In *Advances in Neural Information Processing Systems*, pages 1296–1304, 2014.

[7] U Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos. 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*, pages 316–324. ACM, 2012.

[8] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender Systems*, pages 79–86. ACM, 2010.

[9] Tamara G Kolda, Brett W Bader, and Joseph P Kenny. Higher-order web link analysis using multilinear algebra. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 8–pp. IEEE, 2005.

[10] Joseph B Kruskal. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*, 18(2):95–138, 1977.

[11] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. ParCube: Sparse parallelizable tensor decompositions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 521–536. Springer, 2012.

[12] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. ParCube: Sparse parallelizable CANDECOMP-PARAFAC tensor decomposition. *ACM Transactions on Knowledge Discovery from Data*, 10(1):3, 2015.

[13] Ioakeim Perros, Evangelos E Papalexakis, Fei Wang, Richard Vuduc, Elizabeth Searles, Michael Thompson, and Jimeng Sun. SPARTan: Scalable PARAFAC2 for large & sparse data. *arXiv preprint arXiv:1703.04219*, 2017.

[14] Nicholas Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 2017.

[15] Nicholas D Sidiropoulos and Anastasios Kyrillidis. Multi-way compressed sensing for sparse low-rank tensors. *IEEE Signal Processing Letters*, 19(11):757–760, 2012.

[16] Nicholas D Sidiropoulos, Evangelos E Papalexakis, and Christos Faloutsos. Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition. *IEEE Signal Processing Magazine*, 31(5):57–70, 2014.

[17] Yining Wang, Hsiao-Yu Tung, Alexander J Smola, and Anima Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems*, pages 991–999, 2015.

[18] David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[19] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 316–324. ACM, 2011.