Large Scale Tensor Factorization via Parallel Sketches

Bo Yang, Ahmed S. Zamzam, Nicholas D. Sidiropoulos

Abstract—Tensor factorization methods have recently gained increased popularity. A key feature that renders tensors attractive is the ability to directly model multi-relational data. In this work, we propose ParaSketch, a parallel tensor factorization algorithm that enables massive parallelism, to deal with large tensors. The idea is to compress the large tensor into multiple small tensors, decompose each small tensor in parallel, 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. Adopting sketching matrices for compression, the proposed method enjoys a dramatic reduction in compression complexity, and features a much lighter combining step. Moreover, theoretical analysis shows that the compressed tensors inherit latent identifiability under mild conditions, hence establishing correctness of the overall approach. Numerical experiments corroborate the theory and demonstrate the effectiveness of the proposed algorithm.

1 Introduction

Tensors are a natural generalization of matrices: whereas matrices model binary relations (e.g. user-item), tensors can model multi-relations (e.g. user-item-time). Tensor factorizations have long been invaluable tools for Chemometrics and Psycometrics, where fluorescence of chemical analytes and hidden traits of personalities are examined under the lens of low-rank tensor factorization. Tensors have also attracted considerable (and rapidly growing) attention from the computer science community, most notably in Data Mining (DM) and Machine Learning (ML), due to the emergence of large, multi-relational data sets. Pioneering works in this area include [2], where the authors applied tensor factorization methods to the problem of webpage link analysis, and [3] where tensor factorization methods were used to analyze social network graphs. Subsequently, tensor methods have been applied to recommender systems [4], [5], [6], [7], [8], topic modeling [9], [10], clustering [11], [12], and bioinfomatics [13], to name a few. We refer the interested readers to the overview papers [14], [15] for an in-depth review.

B. Yang is with Amazon.com. This work was done when he was a PhD student at University of Minnesota. Email: yang4173@umn.edu
A. Zamzam is with National Renewable Energy Laboratory. This work was done when he was he was a PhD student at University of Minnesota. Email: zamza002@umn.edu
N.D. Sidiropoulos is with the Electrical and Computer Engineering Department, University of Virginia. Email: nikos@virginia.edu
Preliminary version [1] of part of this work was presented at the 2018 SIAM International Conference on Data Mining.

For DM applications, the large data volume poses a major challenge to tensor factorization methods. For *canonical polyadic decomposition* (CPD), a very popular tensor factorization model, classical applications involve mainly small datasets, where "heavy" algorithms like alternating least squares (ALS) or Gauss-Newton usually suffice. On the other hand, modern datasets in DM are often huge, thus performing CPD is challenging. To tackle this problem, several algorithms have been proposed recently, e.g. [16], [17], [18], [19], [20], [21]

GigaTensor [17] proposes a way to avoid "intermediate data explosion" in using ALS to perform CPD. DFacTo proposes 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. As such, these works rely on distributed implementation, where data and (or) variables are stored in different machines – but the networking links can be a bottleneck. PARCUBE [21] puts forth a "divide-and-conquer" approach which offers scalable approximation of sparse data tensors using sparse factors. However, it cannot guarantee the recovery of the true underlying factors. That is, while the true latent factors of the original tensor may be identifiable, PARCUBE may fail to identify them, owing to the divide-and-conquer strategy it employs. This gives up one important advantage of tensors over matrices: the ability to *recover* true factors from the tensor – formally known as the *essential uniqueness* property [22]. Essential uniqueness means that, 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 matrix factorization models, where linear transformation ambiguity is usually present.

From the application perspective, model uniqueness / identifiability is necessary for interpretability, as different model parametrizations are associated with different explanations of the data. On the theoretical front, uniqueness of CPD has been tapped to establish *model identifiability* – a key property in statistical learning – for some well-known models, e.g., the Gaussian mixture model (GMM), hidden Markov model (HMM), and the latent Dirichilet allocation (LDA), see e.g., [9], [23], [24]. In such cases, tensor factorization is employed as an estimation tool for model parameters (e.g. conditional probability mass functions), hence it is paramount that these parameters can be *uniquely* pinned

down from data – which is what the essential uniqueness property of low-rank tensor decomposition offers.

With identifiability in mind, PARACOMP [20] proposes a similar technique as PARCUBE, where the sub-sampling step of PARCUBE is replaced by random compression. With this modification, strong identifiability guarantees for the rank-one tensor were established [20]. Albeit PARACOMP enables parallel computation, there are two issues that limit its application. First, the compression step involves multiplying dense matrices with tensors. Second, the combining step involves solving a large dense linear system of equations. These two issues can become a bottleneck when dealing with large tensors.

This is an undesirable compromise: the strong model identifiability guarantee of PARACOMP comes from using dense random compression matrices, which can become a computation bottleneck; whereas the scalability of PARCUBE comes from random sampling, which lacks theoretical identifiability guarantees. In this paper, we aim at bridging this apparent dichotomy. Specifically, we propose ParaSketch, a parallel algorithm for tensor factorization that enjoys the following properties:

- *Scalability*: The proposed method employs sketching matrices to perform compression, which are much more computationally efficient compared to the Gaussian random matrices 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.

We emphasize that identifiability of the factorization of the compressed / sampled small tensors is crucial in this "divide-and-conquer" approach, because without it we cannot guarantee correctness of the final result.

By design, the ParaSketch algorithm is naturally parallelizable as the sketched tensors are independently factored. However, the huge computational gains of parallelization come with some degradation in the accuracy of the recovered factors of the original tensor. In order to elucidate this trade-off, we also introduce a new joint factorization approach as a baseline. This method processes all the sketched tensors in a centralized fashion. Numerical experiments confirm that the joint factorization approach provides a more accurate estimation of the latent factors of the original tensor, at the expense of higher computational complexity (and of course loss of parallelization).

A conference version of part of this work was presented at the 2018 SIAM International Conference on Data Mining [1]. Relative to [1], this journal version includes several additional contributions, namely i) extension of the ParaSketch framework from the CPD to the so-called block term decomposition (BTD, [25], [26]), which is well motivated in applications where one of the factor matrices is known to have colinear columns, see [25]; ii) more comprehensive experimental results and insights; and iii) the new joint factorization approach, which serves as a baseline to allow us to gauge the trade-off between accuracy and parallel computation.

We conclude this introduction by noting that sketching [27] has also been utilized to accelerate tensor decompositions in a different way. Specifically, in [28], sketching methods were used to solve the overdetermined linear least squares sub-problems in the ALS algorithm for CPD. Different from our method, the approach in [28] requires centralized processing of the big tensor, thus hindering its application for large-scale datasets. Additionally, while we focus on large scale *unconstrained* tensor factorization, it is worth mentioning that several recent works [16], [29], [30], [31] have proposed scalable algorithms for constrained tensor factorization.

Notation. The symbols $\otimes$, $\odot$, $\circ$ denote the Kronecker product, Khatri-Rao product, and outer product, respectively. We use bold lowercase letters to denote vectors (e.g. $\boldsymbol{x}$), and bold uppercase for matrices (e.g. $\boldsymbol{A}$). The vectors are assumed to be column vectors, and $(\cdot)^{\mathsf{T}}$ denotes transposition. Bold symbols with underscore denote tensors, e.g. $\underline{\boldsymbol{X}}$. The symbols $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the ℓ_2-norm and the Frobenius norm, respectively. We use $f(t) = \mathcal{O}(t)$ to mean that there exits constants C_1 and C_2, such that $C_1 t \leq f(t) \leq C_2 t$, and $f(t) = \Omega(t)$ to mean $f(t) \geq Ct$ for some constant C. More specialized notation will be introduced where needed in the main text.

2 Background

2.1 CPD and its uniqueness property

The CPD model is defined as

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

where F is the smallest integer such that the factorization (1) is possible. This smallest integer is defined to be the rank of $\underline{\boldsymbol{X}}$. The symbol $\circ$ denotes outer product. We use $\boldsymbol{A}(:,f)$ to denote the f-th column of $\boldsymbol{A}$, and $(:)$ means all the elements in the corresponding argument. Henceforth, we use the shorthand notation $\underline{\boldsymbol{X}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$ to denote (1). To facilitate discussion, we will also make use of the matricized and vectorized forms of tensors. For matricization, we stack 1-D vectors of a tensor into a matrix. For instance, consider a 3-D tensor $\underline{\boldsymbol{X}} \in \mathbb{R}^{I \times J \times K}$, we denote

$$\boldsymbol{X}_1 = \left[\begin{array}{ccc} \text{vec}(\underline{\boldsymbol{X}}(1,:,:)) & \cdots & \text{vec}(\underline{\boldsymbol{X}}(I,:,:)) \end{array} \right] \tag{2a}$$

$$\boldsymbol{X}_2 = \left[\begin{array}{ccc} \text{vec}(\underline{\boldsymbol{X}}(:,1,:)) & \cdots & \text{vec}(\underline{\boldsymbol{X}}(:,J,:)) \end{array} \right] \tag{2b}$$

$$\boldsymbol{X}_3 = \left[\begin{array}{ccc} \text{vec}(\underline{\boldsymbol{X}}(:,:,1)) & \cdots & \text{vec}(\underline{\boldsymbol{X}}(:,:,K)) \end{array} \right] \tag{2c}$$

where $\text{vec}(\cdot)$ is the usual vectorization operator, which stacks *columns* of a matrix into a long column vector. We call these mode-1, mode-2, and mode-3 matricization. Correspondingly, three different vectorized versions of the same tensor can be obtained as

$$\boldsymbol{x}_1 = \text{vec}(\boldsymbol{X}_1), \quad \boldsymbol{x}_2 = \text{vec}(\boldsymbol{X}_2), \quad \boldsymbol{x}_3 = \text{vec}(\boldsymbol{X}_3), \tag{3}$$

which are permuted versions of one another. The matricization and vectorization operations are illustrated in Figure 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2982144, IEEE Transactions on Knowledge and Data Engineering

It can be readily checked that if $\underline{\boldsymbol{X}}$ admits a CPD model $\underline{\boldsymbol{X}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$ of rank F, we have

$$\boldsymbol{X}_1 = (\boldsymbol{C} \odot \boldsymbol{B})\boldsymbol{A}^\mathsf{T}$$
$$\boldsymbol{X}_2 = (\boldsymbol{C} \odot \boldsymbol{A})\boldsymbol{B}^\mathsf{T}$$
$$\boldsymbol{X}_3 = (\boldsymbol{B} \odot \boldsymbol{A})\boldsymbol{C}^\mathsf{T}.$$

The symbol $\odot$ denotes the Khatri-Rao product, which is defined as

$$\boldsymbol{A} \odot \boldsymbol{B} := [\boldsymbol{A}(:,1) \otimes \boldsymbol{B}(:,1), \cdots, \boldsymbol{A}(:,F) \otimes \boldsymbol{B}(:,F)],$$

where $\otimes$ denotes the Kronecker product. Consequently, for the vectorized form, we have $\boldsymbol{x}_1 = (\boldsymbol{A} \odot \boldsymbol{C} \odot \boldsymbol{B})\mathbf{1}$, $\boldsymbol{x}_2 = (\boldsymbol{B} \odot \boldsymbol{C} \odot \boldsymbol{A})\mathbf{1}$, and $\boldsymbol{x}_3 = (\boldsymbol{C} \odot \boldsymbol{B} \odot \boldsymbol{A})\mathbf{1}$, where $\mathbf{1}$ is an all-one vector of compatible size.

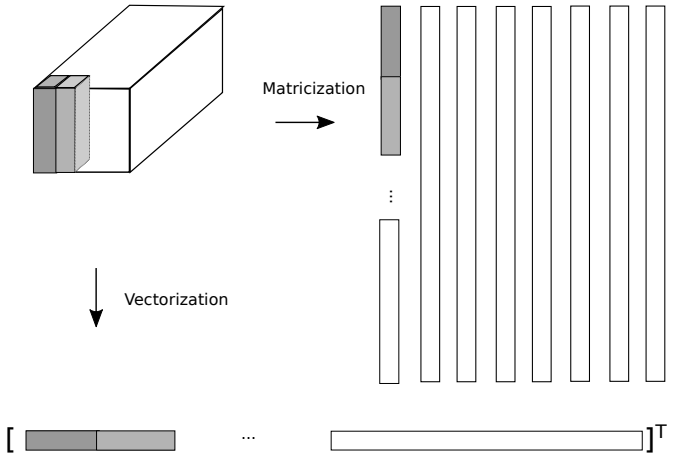

Fig. 1: Tensor matricization and vectorization. The matrix and vector correspond to $\boldsymbol{X}_3$ and $\boldsymbol{x}_3$.

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

Definition 1. (Essential uniqueness) Given a tensor $\underline{\boldsymbol{X}}$ of rank F, we say that its CPD 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 there exists inherently unresolvable permutation ambiguity since permutation of rank-1 tensors does not change their sum. 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 CPD, 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 the largest 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 CPD.

Theorem 1. (Identifiability of CPD, [32]) 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 $\text{rank}(\underline{\boldsymbol{X}}) = F$ and the decomposition of $\underline{\boldsymbol{X}}$ is essentially unique.

Theorem 1 asserts that under some mild conditions, the factors of a tensor are identifiable.

2.2 Block term decomposition

A closely related model of CPD is the so-called PARALIND [25], where factors have colinear columns. This model is well motivated for analyzing real-world data, when it is known *a priori* that linear dependency exists due to, e.g., experimental design, or the underlying physics / chemistry – see [25] for some examples on analyzing fluorescence data of chemical reactions. It is worth pointing out that the PARALIND model is the same as a particular case of the so-called block term decomposition (BTD, [26]) – and we adopt the name BTD in this work for conciseness. The rank-$(R, R, 1)$ BTD model is defined as

$$\underline{\boldsymbol{X}} = \sum_{f=1}^{F} \left(\boldsymbol{A}_f \boldsymbol{B}_f^\mathsf{T}\right) \circ \boldsymbol{c}_f, \tag{4}$$

where the matrices $\boldsymbol{A}_f$ and $\boldsymbol{B}_f$ have rank R – hence the name "rank-$(R, R, 1)$ BTD". In fact, the BTD can also be written as a special polyadic decomposition (PD) (but not necessarily minimal – hence not canonical), as follows

$$\underline{\boldsymbol{X}} = \sum_{f=1}^{F} \left(\boldsymbol{A}_f \boldsymbol{B}_f^\mathsf{T}\right) \circ \boldsymbol{c}_f \tag{5}$$
$$= \sum_{f=1}^{F} \left(\sum_{r=1}^{R} \boldsymbol{A}_f(:,r)\boldsymbol{B}_f(:,r)^\mathsf{T}\right) \circ \boldsymbol{c}_f \tag{6}$$
$$= \sum_{f=1}^{F} \sum_{r=1}^{R} \boldsymbol{A}_f(:,r) \circ \boldsymbol{B}_f(:,r) \circ \boldsymbol{c}_f \tag{7}$$
$$= \sum_{i=1}^{FR} \widehat{\boldsymbol{A}}_i \circ \widehat{\boldsymbol{B}}_i \circ \widehat{\boldsymbol{C}}_i, \tag{8}$$

where $\boldsymbol{A}_f(:,r)$ is the r-th column of matrix $\boldsymbol{A}_f$, and we have defined $\widehat{\boldsymbol{A}} = [\boldsymbol{A}_1, \cdots, \boldsymbol{A}_F]$, $\widehat{\boldsymbol{B}} = [\boldsymbol{B}_1, \cdots, \boldsymbol{B}_F]$ and $\widehat{\boldsymbol{C}} = [\boldsymbol{c}_1\mathbf{1}_R^\mathsf{T}, \cdots, \boldsymbol{c}_F\mathbf{1}_R^\mathsf{T}]$. One can see that indeed BTD can be thought as a special PD, where the third mode has a special structure: each original column in the third mode of BTD is repeated R times in the corresponding PD representation.

Uniqueness of BTD cannot be established via uniqueness of CPD – note in particular that the condition in Theorem 1 always fails to work for BTD, due to the colinearity in the third mode. Instead, uniqueness of BTD was established in [26] using a generalized notion of Kruskal rank, named k'-rank. To introduce this definition, we consider a partitioned matrix $\boldsymbol{A} = [\boldsymbol{A}_1, \cdots, \boldsymbol{A}_F]$, where each $\boldsymbol{A}_i$ is a matrix.

Definition 3. (k'-rank, [26]) For a partitioned matrix $\boldsymbol{A}$, the k'-rank, denoted as $k'_{\boldsymbol{A}}$ (or $k'(\boldsymbol{A})$) is the maximal integer r such that any r submatrices of $\boldsymbol{A}$ form a set of linearly independent columns.

Theorem 2. (Theorem 4.4 of [26]) Let $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ represent a decomposition of a tensor $\underline{\boldsymbol{X}}$ into F rank-$(R, R, 1)$ terms. If

$$k_{\boldsymbol{C}} = F \quad \text{and} \quad k'_{\boldsymbol{A}} + k'_{\boldsymbol{B}} \geq F + 2, \tag{9}$$

then $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ is essentially unique.

Here essential uniqueness of $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ means that the F rank-$(R, R, 1)$ terms are unique; it is clear that within each of these terms, there is freedom for arbitrary nonsingular linear transformation. For instance, $(\boldsymbol{A}_f \boldsymbol{B}_f^{\mathsf{T}}) \circ \boldsymbol{c}_f = (\boldsymbol{A}_f \boldsymbol{M} \boldsymbol{M}^{-1} \boldsymbol{B}_f^{\mathsf{T}}) \circ \boldsymbol{c}_f$ for any nonsingular matrix $\boldsymbol{M}$. Note that this is analogous to essential uniqueness of CPD, where the F rank-1 terms are unique; but due to the inner dimension being 1, there is only scaling / counter-scaling freedom in that case.

2.3 Compressed tensor factorization

We aim at developing "divide-and-conquer" strategies for large scale tensor factorization. PARACOMP [20] serves as the starting point of our work. PARACOMP uses parallel compression stages to create multiple small tensors, performing independent CPD on the small tensors, and combining the results to get CPD factors 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 [20] 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 $N \leq K$. The compression scheme is shown in Figure 2, where $\boldsymbol{U}$ multiplies the tensor on the first mode, yielding a tensor of size $L \times J \times K$. Similar compression is performed on the second and third mode with $\boldsymbol{V}$ and $\boldsymbol{W}$. Suppose the original tensor $\underline{\boldsymbol{X}}$ admits a CPD model, with latent factors $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$, i.e. $\boldsymbol{x} = (\boldsymbol{C} \odot \boldsymbol{B} \odot \boldsymbol{A})\mathbf{1}$, where the subscript 3 is omitted for conciseness, the compressed tensor $\boldsymbol{y}$ can be written as [33]

$$\begin{aligned} \boldsymbol{y} &= (\boldsymbol{W} \otimes \boldsymbol{V} \otimes \boldsymbol{U})(\boldsymbol{C} \odot \boldsymbol{B} \odot \boldsymbol{A})\mathbf{1} \\ &= ((\boldsymbol{W}\boldsymbol{C}) \odot (\boldsymbol{V}\boldsymbol{B}) \odot (\boldsymbol{U}\boldsymbol{A}))\,\mathbf{1}. \end{aligned} \tag{10}$$

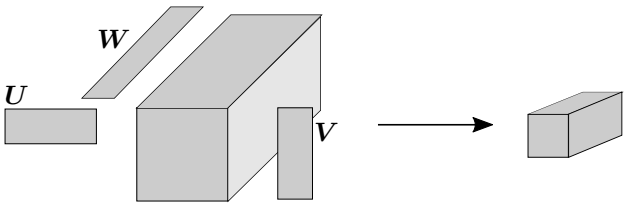

Fig. 2: Tensor compression with random projection.

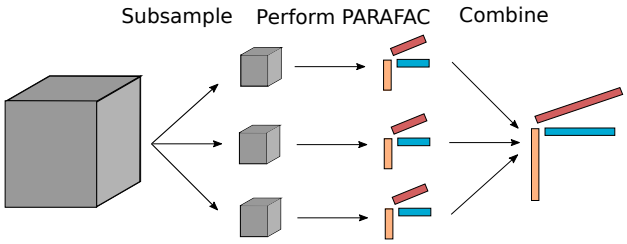

Fig. 3: Schematic of PARCUBE [21]

To materialize the compression and factorization scheme, a key issue is under what conditions are the latent factors of the compressed tensor $\boldsymbol{y}$ identifiable. For CPD, the following theorem answers this question.

Theorem 3. (Identifiability of CPD under compression, [20]) Let $\boldsymbol{x} = (\boldsymbol{C} \odot \boldsymbol{B} \odot \boldsymbol{A})\mathbf{1} \in \mathbb{R}^{IJK}$, where $\boldsymbol{A}$ is $I \times F$, $\boldsymbol{B}$ is $J \times F$, $\boldsymbol{C}$ is $K \times F$, and consider compressing it to $\boldsymbol{y} = (\boldsymbol{W} \otimes \boldsymbol{V} \otimes \boldsymbol{U})\boldsymbol{x} = ((\boldsymbol{W}\boldsymbol{C}) \odot (\boldsymbol{V}\boldsymbol{B}) \odot (\boldsymbol{U}\boldsymbol{A}))\,\mathbf{1} = (\tilde{\boldsymbol{C}} \odot \tilde{\boldsymbol{B}} \odot \tilde{\boldsymbol{A}})\mathbf{1} \in \mathbb{R}^{LMN}$, 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

$$\min(L, k_{\boldsymbol{A}}) + \min(M, k_{\boldsymbol{B}}) + \min(N, k_{\boldsymbol{C}}) \geq 2F + 2, \tag{11}$$

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 / counter-scaling.

Remark 1. One may be wondering, why does uniqueness of factorization of the compressed tensors matter at all? In this "divide-and-conquer" algorithm design context, uniqueness of factorization of the compressed tensors is important mainly due to *algorithmic correctness* considerations. In short, if the compressed tensors do not admit unique factorization, then there is no principled way of combining results from these compressed tensors, see Section 3.2.

Despite the desirable identifiability guarantees offered by PARACOMP, its compression stage can be expensive since all the compression matrices are dense and unstructured. Overall, for a dense tensor, the computational 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 perform compression on the tensor faster than the dense matrix-tensor multiplication as in PARACOMP. Now the question is, how to construct such compression matrices, so that one can still guarantee identifiability of the compressed tensors, with reasonable assumptions?

2.4 Fast Johnson Lindenstrauss Transform

To answer the above question, we introduce the following subsampled randomized Hadamard transform (SRHT), a fast version of the celebrated Johnson-Lindenstrauss transform (JLT, [34]). For an overview on JLT and its variants, please see [27]. The SRHT of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is $\boldsymbol{S}\boldsymbol{x}$, where $\boldsymbol{S}$ is a specially structured matrix, $\boldsymbol{S} = \boldsymbol{P}\boldsymbol{H}\boldsymbol{D}$. The $\boldsymbol{P}$ is a random subsampling matrix, i.e., each row of $\boldsymbol{P}$ contains only one nonzero element (equal to 1) whose position is uniformly distributed. The $\boldsymbol{H}$ is a Hadamard matrix, which is recursively defined as

$$\boldsymbol{H}_2 = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad \boldsymbol{H}_{2^k} = \begin{bmatrix} \boldsymbol{H}_{2^{k-1}} & -\boldsymbol{H}_{2^{k-1}} \\ \boldsymbol{H}_{2^{k-1}} & \boldsymbol{H}_{2^{k-1}} \end{bmatrix}. \tag{12}$$

Note that $\boldsymbol{H}$ is defined only for dimensions that are a certain power of 2. The $\boldsymbol{D}$ is a diagonal matrix with i.i.d. diagonal entries, each being a Rademacher random variable that takes values in $\{1, -1\}$ with equal probability.

Proposition 1. (SRHT, [27], [35], [36]) Let $\boldsymbol{S} = \frac{1}{\sqrt{mn}}\boldsymbol{P}\boldsymbol{H}_n\boldsymbol{D}$, where $\boldsymbol{D}$ is an $n \times n$ diagonal matrix with i.i.d. diagonal entries $\boldsymbol{D}_{ii} = 1$ with probability $1/2$, and $\boldsymbol{D}_{ii} = -1$ with probability $1/2$. Matrix $\boldsymbol{H}_n$ is a Hadamard matrix of size $n \times n$, where n is assumed to be a power of 2. The $m \times n$ matrix $\boldsymbol{P}$ samples m coordinates of an n-dimensional vector uniformly at random, where

$$m = \Omega\left(\epsilon^{-2}(\ln d)\big(\sqrt{d} + \sqrt{\ln n}\,\big)^2\right).$$

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

$$\|\boldsymbol{I} - \boldsymbol{U}^\mathsf{T}\boldsymbol{S}^\mathsf{T}\boldsymbol{S}\boldsymbol{U}\|_2 \leq \epsilon. \tag{13}$$

Moreover, for any vector $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{S}\boldsymbol{x}$ can be computed in $\mathcal{O}(n \ln m)$ time.

A proof of this result can be found in [35]. This proposition asserts that, with enough rows in the SRHT matrix $\boldsymbol{S}$, the matrix $\boldsymbol{S}\boldsymbol{U}$ preserves the dimension of the subspace spanned by the columns of $\boldsymbol{U}$, as shown in (13).

3 Proposed approach: ParaSketch for CPD

Our goal is to design an efficient parallel tensor factorization algorithm, by replacing the Gaussian random matrices in PARACOMP with SRHT matrices. Towards that end, we first address the identifiability issue of the CPD of the compressed tensors in Section 3.1; then, in Section 3.2, we discuss how to exploit the structure of SRHT matrices to lower the computation burden of the combining step. We analyze the complexity of the proposed method in Section 3.3, and compare it with that of PARACOMP. Finally, in Section 3.4, we propose a new baseline method.

3.1 Identifiability of compressed tensor CPD with SRHT matrices

Theorem 4. (main result) For $\boldsymbol{x} = (\boldsymbol{C} \odot \boldsymbol{B} \odot \boldsymbol{A})\boldsymbol{1}$, the CPD model of the sketched tensor $\boldsymbol{y} = (\boldsymbol{S}^c \otimes \boldsymbol{S}^b \otimes \boldsymbol{S}^a)(\boldsymbol{C} \odot \boldsymbol{B} \odot \boldsymbol{A})\boldsymbol{1}$ is identifiable with high probability if

$$k_{\boldsymbol{A}} + k_{\boldsymbol{B}} + k_{\boldsymbol{C}} \geq 2F + 2, \tag{14}$$

and $\boldsymbol{S}^a \in \mathbb{R}^{L\times I}$, $\boldsymbol{S}^b \in \mathbb{R}^{M\times J}$, $\boldsymbol{S}^c \in \mathbb{R}^{N\times K}$ are all SRHT matrices, with number of rows

$$L = \Omega\left(\epsilon^{-2}(\ln k_{\boldsymbol{A}})\big(\sqrt{k_{\boldsymbol{A}}} + \sqrt{\ln I}\,\big)^2\right),$$
$$M = \Omega\left(\epsilon^{-2}(\ln k_{\boldsymbol{B}})\big(\sqrt{k_{\boldsymbol{B}}} + \sqrt{\ln J}\,\big)^2\right),$$
$$N = \Omega\left(\epsilon^{-2}(\ln k_{\boldsymbol{C}})\big(\sqrt{k_{\boldsymbol{C}}} + \sqrt{\ln K}\,\big)^2\right),$$

respectively, where $\epsilon \in (0, 1)$ is the accuracy parameter in (13).

This result is similar to Theorem 3. The key difference is that we replace the dense compression matrices with sketching matrices. This has a tremendous impact on the computation complexity, see Table 1 and Figure 5. Moreover, the argument used in [20] to establish unique factorization of compressed tensors does not apply here, as it hinges on compression matrices drawn from absolutely continuous distributions (c.f. Theorem 3), which is not the case for SRHT matrices. Before proving Theorem 4, several remarks are in order.

Remark 2. In Theorem 4, we assume dimensions $I,\ J,\ K$ are powers of 2. In practice, one can simply pad the tensor with slabs of all zeros if they are not. The redundant dimensions can be removed in the final factors after combining.

Remark 3. In many cases of practical interest, the tensor data is "long", in the sense that one of the dimension is much larger than the others. For instance, the spectrogram tensor in analyzing EEG or MEG data (see e.g. [37]) has three modes: time, frequency, and channels. The number of frequency bands and number of channels are often on the order of hundreds, whereas the time index can be in the order of hundreds of thousands, due to the high temporal resolution and long duration of EEG and MEG recordings. In such a case, one might be interested in analyzing the frequency mode and channel mode factors, but not the full temporal factor – which is very long, and potentially overly detailed. Our method can be easily adapted to this case. Assuming we are interested in compressing the third mode, we perform sketching on the third mode. For this setting, we have the following corollary.

Corollary 1. For tensor $\boldsymbol{x} = (\boldsymbol{C} \odot \boldsymbol{B} \odot \boldsymbol{A})\boldsymbol{1}$, the CPD of the partially sketched tensor $\boldsymbol{y} = (\boldsymbol{S}^c \otimes \boldsymbol{I} \otimes \boldsymbol{I})(\boldsymbol{C} \odot \boldsymbol{B} \odot \boldsymbol{A})\boldsymbol{1}$ is identifiable with high probability, provided $k_{\boldsymbol{A}} + k_{\boldsymbol{B}} + k_{\boldsymbol{C}} \geq 2F + 2$ and $\boldsymbol{S}^c \in \mathbb{R}^{N\times K}$ is SRHT with

$$N = \Omega\left(\epsilon^{-2}(\ln(k_{\boldsymbol{C}}))\big(\sqrt{k_{\boldsymbol{C}}} + \sqrt{\ln(K)}\,\big)^2\right),$$

rows, where $\epsilon \in (0, 1)$ is the accuracy parameter in (13).

To prove Theorem 4, 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 1. Let $\boldsymbol{S} \in \mathbb{R}^{L\times I}$ be an SRHT matrix with

$$L = \Omega\left(\epsilon^{-2}(\ln k_{\boldsymbol{A}})\big(\sqrt{k_{\boldsymbol{A}}} + \sqrt{\ln I}\,\big)^2\right), \tag{15}$$

rows. Then, with high probability, $k(\boldsymbol{S}\boldsymbol{A}) = k_{\boldsymbol{A}}$.

The proofs of Lemma 1 and Theorem 4 can be found in the appendix.

3.2 Combining the results

The combining process generally follows [20], [21], but we develop special treatments to fully exploit the structure of SRHT matrices to get away with 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 CPD on each of these small tensors in parallel, and obtain $[\boldsymbol{A}_t, \boldsymbol{B}_t, \boldsymbol{C}_t],\ t \in \{1, \cdots T\}$. Then we can write

$$\boldsymbol{A}_t = \boldsymbol{S}_t\boldsymbol{A}\boldsymbol{\Pi}_t\boldsymbol{\Lambda}_t, \tag{16}$$

which means that we still have *individual* permutation ($\boldsymbol{\Pi}_t$) and scaling ($\boldsymbol{\Lambda}_t$) ambiguities for each t. This is because factoring each small tensor *independently* leads to possibly different scaling and permutation of the (compressed) true factors, as indicated by the subscript t in (16). Only after reconciling these different permutations and scalings can we combine the results to 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

$$\overline{\boldsymbol{A}}_t = \overline{\boldsymbol{S}}\boldsymbol{A}\boldsymbol{\Pi}_t\boldsymbol{\Lambda}_t. \tag{17}$$

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

$$\widehat{\boldsymbol{A}}_t = \overline{\boldsymbol{S}}\boldsymbol{A}\boldsymbol{\Pi}_t\boldsymbol{\Lambda}. \tag{18}$$

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

$$\min_{\boldsymbol{\Pi}\in\mathcal{P}_F} \ \|\widehat{\boldsymbol{A}}_1 - \widehat{\boldsymbol{A}}_t\boldsymbol{\Pi}\|_{\mathrm{F}}^2, \tag{19}$$

where we use $\mathcal{P}_F$ to denote the set of permutation matrices of size $F \times F$. With a little manipulation, problem (19) is equivalent to

$$\max_{\boldsymbol{\Pi}\in\mathcal{P}_F} \ \mathrm{tr}\left(\widehat{\boldsymbol{A}}_1^{\mathsf{T}}\widehat{\boldsymbol{A}}_t\boldsymbol{\Pi}\right). \tag{20}$$

Problem (20) is known as the Linear Assignment Problem (LAP), and can be solved efficiently by the Hungarian algorithm [38].

With a little abuse of notation, we denote the resulting permutation matrix from (20) as $\boldsymbol{\Pi}_t, \ t \in \{2, \cdots, T\}$. We use these $\boldsymbol{\Pi}_t$ to permute the results $\boldsymbol{A}_t$ in (16), and get the following

$$\check{\boldsymbol{A}}_t = \boldsymbol{S}_t\boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}_t. \tag{21}$$

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

$$\check{\boldsymbol{A}}_t = \boldsymbol{S}_t\boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}. \tag{22}$$

After fixing scaling and permutation, we can combine the results as follows

$$\begin{bmatrix} \check{\boldsymbol{A}}_1 \\ \check{\boldsymbol{A}}_2 \\ \vdots \\ \check{\boldsymbol{A}}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_1 \\ \boldsymbol{S}_2 \\ \vdots \\ \boldsymbol{S}_T \end{bmatrix} \boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}. \tag{23}$$

By SRHT definition, we have $\boldsymbol{S}_i = \boldsymbol{P}_i\boldsymbol{H}\boldsymbol{D}$, and we can write

$$\begin{aligned} \begin{bmatrix} \check{\boldsymbol{A}}_1 \\ \check{\boldsymbol{A}}_2 \\ \vdots \\ \check{\boldsymbol{A}}_T \end{bmatrix} &= \begin{bmatrix} \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \\ \vdots \\ \boldsymbol{P}_T \end{bmatrix} \boldsymbol{H}\boldsymbol{D}\boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda} \\ &:= \widehat{\boldsymbol{P}}\boldsymbol{H}\boldsymbol{D}\boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}, \end{aligned} \tag{24}$$

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 costs $\mathcal{O}(I\ln(I)F)$.

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

Algorithm 1 ParaSketch

Input: Data tensor $\underline{\boldsymbol{X}}$, number of replicas T, compression dimension $L, \ M, \ N$, CPD 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 CPD 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 tensor by solving (19), and permute the factors in (16) to obtain (21);
2: Resolve scaling ambiguity by dividing entries in each column with values in a common row;
3: Invert $\widehat{\boldsymbol{P}}$, $\boldsymbol{H}$, $\boldsymbol{D}$ in (24) in turn.

Remark 4. *(Number of replicas needed)* By our construction, each compression shares $\boldsymbol{H}$ and $\boldsymbol{D}$. To ensure inversion in (24) can be carried out, we need each row index in matrix $\boldsymbol{H}\boldsymbol{D}\boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}$ 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 (see e.g. [39]) that one needs to try $\mathcal{O}(n\ln(n))$ times in order to collect n coupons. In our problem, this means that the number of replicas T is in the order $\mathcal{O}(I\ln(I)/L)$.

Remark 5. *(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 $T = \eta I\ln(I)/L$, where η is some constant. In this case, the compression ratio can be expressed as $I^2/(\eta L^2\ln I)$. In our simulations, we observed that once η exceeds 1.5, we get exact recovery (up to numerical precision) of the factors in noiseless cases, suggesting that the compression ratio can be very high when $L \ll I$.

3.3 Complexity analysis

To simplify analysis, we focus on the first mode. In PARACOMP, to ensure the first mode factor of the original tensor

TABLE 1: Time complexity comparison

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

can be recovered from that of the compressed tensors, one needs the number of replicas to be $T \geq I/L$. Note that this means PARACOMP needs a factor of $\ln(I)$ 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 $\mathcal{O}(IJK \ln L)$. 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 $\mathcal{O}(I \ln I)$ for a length I vector, and $\mathcal{O}(IJK \ln I)$ for the whole tensor.

In the final combining procedure, our method costs $\mathcal{O}(IF \ln I)$, while PARACOMP costs $\mathcal{O}(I^3)$ in general. Note that in principle one can invoke the property that a random Gaussian matrix is approximately orthogonal for large I, and replace the inversion with transpose, resulting in lower cost of $\mathcal{O}(I^2F)$ – but solution accuracy deteriorates if such an approximation is adopted. Time complexity comparison of PARACOMP and ParaSketch is summarized in Table 1, see Figure 5 for numerical experiment results.

In terms of space (memory) complexity, both algorithms require storage of the compressed tensors, as well as the compression matrices for use in the combining step. To simplify analysis, we focus on a tensor of size $I \times I \times I$. For PARACOMP, the storage requirement for the compressed tensors is $\mathcal{O}(T_{\text{PARACOMP}}L^3)$, and $\mathcal{O}(T_{\text{PARACOMP}}LI)$ for the compression matrices. For ParaSketch, the storage of compressed tensors is $\mathcal{O}(T_{\text{ParaSketch}}L^3)$. The Hadamard matrices in compression do not need to be stored, and the total storage in compression is due to $\{\boldsymbol{P}_t\}$ and $\boldsymbol{D}$, which incur $\mathcal{O}(T_{\text{ParaSketch}}L)$ storage requirement.

The difference in T_{PARACOMP} and $T_{\text{ParaSketch}}$ makes comprehensive comparison of the respective memory footprints challenging. Here we let $T_{\text{PARACOMP}} = \lceil I/L \rceil$, since this is the minimal number of replicas needed to recover the factors of the original tensor in PARACOMP, and $T_{\text{ParaSketch}} = \mathcal{O}(\lceil (I \ln I)/L \rceil)$, since this ensures successful recovery w.h.p, as discussed in Remark 4. With these choices, we get the storage complexity $\mathcal{O}(IL^2 + I^2)$ for PARACOMP, and $\mathcal{O}(IL^2 \ln I + I \ln I)$ for ParaSketch. Further, if we assume the compression dimension to be $L = \sqrt{I}$, we get space complexity $\mathcal{O}(I^2)$ for PARACOMP, and $\mathcal{O}(I^2 \ln I)$ for ParaSketch.

Remark 6. If one aims at achieving *full* parallelism when factoring all the compressed tensors, the proposed ParaSketch method takes more computational resources than PARACOMP: a factor of $(\ln I)$ more storage is needed, as well as parallel processors. However, when resources are limited, one can instead instantiate these small tensors in batch, and discard them after factorization. In Section 5, we show that factoring small tensors takes a small fraction of time compared to compression and combining in PARACOMP. Hence we can afford to factor these small tensors in *sequential* batches (and perform *parallel* factorization within each batch), so long as the compression and combining time are reduced dramatically – which is exactly the case of ParaSketch (c.f. Figure 5).

3.4 Joint factorization of sketched tensors

We next explore a new baseline method, to better understand the trade-off between parallelism speed and accuracy of the final result. After the compression process, the randomly sketched tensors can instead be factored jointly in order to obtain the latent factors of the original tensor. The joint factorization is solved as a whole, hence the benefits of parallel computing as in ParaSketch are lost – but the resulting factors will be more accurate than those of ParaSketch, as we shall see in Section 5. The reason for this is that one exploits all sketched data simultaneously, and avoids the noisy division and permutation-matching which are fallible in low SNR and/or high-rank scenarios. The joint factorization formulation is as follows.

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \quad \sum_{t=1}^{T} \|\boldsymbol{Y}_t - \big(\,(\boldsymbol{S}_t^a\,\boldsymbol{A}) \odot (\boldsymbol{S}_t^b\,\boldsymbol{B})\big)\,(\boldsymbol{S}_t^c\,\boldsymbol{C})^{\mathsf{T}}\,\|_F^2 \quad (25)$$

where $\boldsymbol{Y}_t$ is the mode-3 matricization of the t-th sketched tensor, and $\boldsymbol{S}_t^a$, $\boldsymbol{S}_t^b$ and $\boldsymbol{S}_t^c$ are the sketching matrices of the t-th sketched tensor in the first, second and third mode, respectively. This optimization problem is clearly nonconvex since the unknowns $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ are multiplied together. However, we can adopt an alternating minimization (AltMin) approach, which updates each factor while fixing the other two in a cyclic fashion, enjoys a closed form optimal solution for each individual factor.

In order to update the matrix $\boldsymbol{C}$, we exploit one of the Kronecker product properties to re-write (25) as

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \quad \sum_{t=1}^{T} \|\boldsymbol{y}_t - \Big(\boldsymbol{S}_t^c \otimes \big((\boldsymbol{S}_t^a\boldsymbol{A}) \odot (\boldsymbol{S}_t^b\boldsymbol{B})\big)\Big)\mathrm{vec}(\boldsymbol{C}^{\mathsf{T}})\|_F^2 \quad (26)$$

where $\boldsymbol{y}_t$ is the vectorization of $\boldsymbol{Y}_t$. In the AltMin approach, instantiating the Kronecker product in (26) represents a computational challenge. The size of the Kronecker product is $(LMN \times FK)$. In addition, similar problems need to be solved for $\boldsymbol{A}$ and $\boldsymbol{B}$ at each iteration.

In the joint factorization approach, the original factors are estimated directly. Therefore, there is no need for reconciling the permutation and scaling as in the original ParaSketch procedure. While the joint factorization approach constitutes the direct solution to obtain the original factors given the sketched tensors, it represents a heavy computational burden. On the other hand, it is often more accurate in terms of estimating the factors.

4 PARASKETCH FOR BTD

We now show how to extend the ParaSketch framework to BTD. Let the rank of $\boldsymbol{A}_f$ and $\boldsymbol{B}_f$ to be R for all $f = 1 \cdots F$. To apply the compression strategy presented in Section 2 to a tensor that conforms to a BTD model, we first note that

$$\mathrm{vec}(\underline{\boldsymbol{X}}) = \sum_{f=1}^{F} \mathrm{vec}\left(\left(\boldsymbol{A}_f\boldsymbol{B}_f^{\mathsf{T}}\right) \circ \boldsymbol{c}_f\right) \quad (27)$$

$$= \sum_{f=1}^{F} \boldsymbol{c}_f \odot ((\boldsymbol{B}_f \odot \boldsymbol{A}_f)\boldsymbol{1}_R)\,. \quad (28)$$

Then the compression can be written as

$$\boldsymbol{y} = (\boldsymbol{W} \otimes \boldsymbol{V} \otimes \boldsymbol{U})\,\mathrm{vec}(\underline{\boldsymbol{X}}) \tag{29a}$$

$$= \sum_{f=1}^{F} (\boldsymbol{W} \otimes \boldsymbol{V} \otimes \boldsymbol{U})\,(\boldsymbol{c}_f \odot ((\boldsymbol{B}_f \odot \boldsymbol{A}_f)\mathbf{1}_R)) \tag{29b}$$

$$= \sum_{f=1}^{F} (\boldsymbol{W}\boldsymbol{c}_f) \odot ((\boldsymbol{V} \otimes \boldsymbol{U})(\boldsymbol{B}_f \odot \boldsymbol{A}_f)\mathbf{1}_R) \tag{29c}$$

$$= \sum_{f=1}^{F} (\boldsymbol{W}\boldsymbol{c}_f) \odot [((\boldsymbol{V}\boldsymbol{B}_f) \odot (\boldsymbol{U}\boldsymbol{A}_f))\,\mathbf{1}_R], \tag{29d}$$

where we applied the mixed product rule: $(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \odot \boldsymbol{D}) = (\boldsymbol{A}\boldsymbol{C}) \odot (\boldsymbol{B}\boldsymbol{D})$ in obtaining (29c) and (29d). We can see that the compressed tensor admits a BTD model with *factors* being compressed, as follows

$$\underline{\boldsymbol{Y}} = \sum_{f=1}^{F} \left(\left((\boldsymbol{U}\boldsymbol{A}_f)(\boldsymbol{V}\boldsymbol{B}_f)^{\mathsf{T}} \right) \circ (\boldsymbol{W}\boldsymbol{c}_f) \right). \tag{30}$$

To characterize the identifiability condition of the compressed BTD model, we first introduce the following lemma.

Lemma 2. Let $\boldsymbol{S}$ be an SRHT matrix, and $\boldsymbol{A} = [\boldsymbol{A}_1, \cdots, \boldsymbol{A}_F]$ be a partitioned matrix, where each block has dimension $I \times R$. Suppose $\boldsymbol{S}$ has dimension $L \times I$, where L satisfies

$$L = \Omega\left(\epsilon^{-2} \ln[k'_{\boldsymbol{A}} R] \left(\sqrt{k'_{\boldsymbol{A}} R} + \sqrt{\ln(I)}\,\right)^2 \right), \tag{31}$$

then with high probability, we have

$$k'(\boldsymbol{S}\boldsymbol{A}) = k'_{\boldsymbol{A}}. \tag{32}$$

The proof is relegated to the appendix. With this lemma in place, we are ready to state the following theorem.

Theorem 5. Let $\underline{\boldsymbol{X}} \in \mathbb{R}^{I \times J \times K}$ admit a rank-$(R, R, 1)$ BTD. If we compress this tensor by SRHT $\boldsymbol{S}_a \in \mathbb{R}^{L \times I}$, $\boldsymbol{S}_b \in \mathbb{R}^{M \times J}$, and $\boldsymbol{S}_c \in \mathbb{R}^{N \times K}$, where the compression dimensions satisfy

$$L = \Omega\left(\epsilon^{-2} \ln[k'_{\boldsymbol{A}} R] \left(\sqrt{k'_{\boldsymbol{A}} R} + \sqrt{\ln(I)}\,\right)^2 \right)$$

$$M = \Omega\left(\epsilon^{-2} \ln[k'_{\boldsymbol{B}} R] \left(\sqrt{k'_{\boldsymbol{B}} R} + \sqrt{\ln(J)}\,\right)^2 \right)$$

$$N = \Omega\left(\epsilon^{-2} \ln[k_{\boldsymbol{C}}] \left(\sqrt{k_{\boldsymbol{C}}} + \sqrt{\ln(K)}\,\right)^2 \right),$$

and in addition

$$\min(N, k_{\boldsymbol{C}}) = F \tag{33}$$

$$k'_{\boldsymbol{A}} + k'_{\boldsymbol{B}} \geq F + 2, \tag{34}$$

then with high probability $(\boldsymbol{S}^a\boldsymbol{A}, \boldsymbol{S}^b\boldsymbol{B}, \boldsymbol{S}^c\boldsymbol{C})$ is essentially unique for the rank-$(R, R, 1)$ BTD of $\underline{\boldsymbol{Y}}$.

The proof of this theorem can be found in the appendix.

Unlike CPD, for BTD, the roles played by the three different modes are different. The first and second modes are "coupled" since they form low-rank matrices within each term of the BTD. For ease of implementation, we focus on a special scenario, where one of the "coupled" modes is small, and we compress the larger mode – assuming that to be the first mode. Note that compression on both the first and second modes together will result in a low-rank matrix recovery problem in the combining step, which is computationally challenging and devising a computationally efficient algorithm for this step is of interest in its own right (and beyond the scope of this work). We therefore consider compressing the first and third modes. Recall that in Section 3.2, we align factors estimated from different replicas by introducing a small number of anchor rows in factor $\boldsymbol{A}$. Due to the special structure of BTD, we introduce anchor rows in the third mode, since it is uniquely identified up to column permutation – whereas the first and second modes are not.

Suppose we create T replicas with sketching matrices $\{\boldsymbol{S}^a_t, \boldsymbol{S}^c_t\}_{t=1}^T$, i.e.

$$\boldsymbol{y}_t = (\boldsymbol{S}^c_t \otimes \boldsymbol{I} \otimes \boldsymbol{S}^a_t)\,\mathrm{vec}(\boldsymbol{X}_t).$$

By applying the nonlinear least square (NLS) algorithm proposed in [40] on $\boldsymbol{y}_t$, we obtain $\{\widetilde{\boldsymbol{M}}_{f,t}\}_{f=1}^F$ and $\widetilde{\boldsymbol{C}}_t$, where $\widetilde{\boldsymbol{M}}_{f,t}$ is the low-rank matrix product of the first and second mode factors in the f-th term of replica t. We let each $\boldsymbol{S}^c_t$ have the first two rows in common, so that we can reconcile the permutations as in Section 3.2. For the $\boldsymbol{C}$ factor, we subsequently resolve the scaling ambiguity as in Section 3.2. The procedure is exactly the same as in Algorithm 2. Once we align the permutation of $\boldsymbol{C}$, we can permute the corresponding low-rank matrices $\{\widetilde{\boldsymbol{M}}_{f,t}\}_{f=1}^F$ since the permutation is common within each replica, by virtue of the essential uniqueness property of BTD.

Recovery of the low-rank matrix requires more care. Since there is inherent scaling ambiguity between columns of $\boldsymbol{C}$ and the corresponding low-rank matrix, we need a method to align the scaling of the low-rank matrices from different replicas. For this purpose, we let $\boldsymbol{S}^a_t$'s have the first row in common. By this design, the (1,1) elements of the low-rank matrices from different replicas are common, and dividing by this element unifies the scaling of different replicas, and we denote the result as $\{\widehat{\boldsymbol{M}}_{f,t}\}_{f=1}^F$

After aligning permutation and scaling, for each low-rank matrix $\boldsymbol{A}_f\boldsymbol{B}_f^{\mathsf{T}}$, we have the following problem

$$\min_{\boldsymbol{A}_f,\,\boldsymbol{B}_f} \sum_{t=1}^{T} \left\| \widehat{\boldsymbol{M}}_{f,t} - \boldsymbol{S}^a_t \boldsymbol{A}_f \boldsymbol{B}_f^{\mathsf{T}} \right\|_F^2. \tag{35}$$

Let $\boldsymbol{S}^a = [\boldsymbol{S}^a_1; \cdots; \boldsymbol{S}^a_T]$ be a vertical concatenation of $\boldsymbol{S}^a_t$'s, and $\widehat{\boldsymbol{M}}_f = [\widehat{\boldsymbol{M}}_{f,1}; \cdots; \widehat{\boldsymbol{M}}_{f,T}]$ be a vertical concatenation of $\widehat{\boldsymbol{M}}_{f,t}$'s. Problem (35) has a closed-form solution

$$\boldsymbol{A}_f\boldsymbol{B}_f^{\mathsf{T}} = \left((\boldsymbol{S}^a)^{\mathsf{T}}\boldsymbol{S}^a \right)^{-1/2} \boldsymbol{U}_R \boldsymbol{\Sigma}_R \boldsymbol{V}_R^{\mathsf{T}}, \tag{36a}$$

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}} = \left((\boldsymbol{S}^a)^{\mathsf{T}}\boldsymbol{S}^a \right)^{-1/2} (\boldsymbol{S}^a)^{\mathsf{T}} \widehat{\boldsymbol{M}}_f \tag{36b}$$

where

$$\left((\boldsymbol{S}^a)^{\mathsf{T}}\boldsymbol{S}^a \right)^{-1/2} = \boldsymbol{D}^{-1}\boldsymbol{H}^{-1}((\boldsymbol{P}^a)^{\mathsf{T}}\boldsymbol{P}^a)^{-1/2}\boldsymbol{H}^{-1}\boldsymbol{D}^{-1}. \tag{37}$$

Equation (36b) is the Singular Value Decomposition (SVD), and $\boldsymbol{U}_R$ ($\boldsymbol{V}_R$) collects the R columns of $\boldsymbol{U}$ ($\boldsymbol{V}$) that correspond to the largest R singular values, while $\boldsymbol{\Sigma}_R$ is $\boldsymbol{\Sigma}$ with the remaining singular values (other than the largest R singular values) replaced by zeros. Symbol $\boldsymbol{P}^a$ denotes the vertical concatenation of $\boldsymbol{P}^a_t$ since $\boldsymbol{S}^a_t = \boldsymbol{P}^a_t\boldsymbol{H}\boldsymbol{D}$.

The right hand side (RHS) of (36a) may seem intimidating on first sight, but it is in fact easy to compute. By the design of $\boldsymbol{P}_t^a$, the product $((\boldsymbol{P}^a)^\mathsf{T}\boldsymbol{P}^a)$ is a diagonal matrix. Then one can see that after computing the $\boldsymbol{U}_R\boldsymbol{\Sigma}_R\boldsymbol{V}_R^\mathsf{T}$ term, we can perform the multiplication from right to left, and the multiplications are either an inverse Hadamard transform, or multiplying with a diagonal matrix, which are all light-weight operations, thanks to the sketching scheme we proposed. Similarly, we can form the RHS of (36b) from right to left, enjoying the light computation of the inverse Hadamard transform and multiplication with diagonal matrices.

In summary, we have shown that for the BTD model, similar sketching and combining techniques can be applied, with identifiability guarantees. The combining procedure for the coupled modes is different, and we summarize it in Algorithm 3. We remark again that we aim at recovering only the product terms $\boldsymbol{A}_f\boldsymbol{B}_f^\mathsf{T}$ in Algorithm 3, since there is inherent linear transformation ambiguity between $\boldsymbol{A}_f$ and $\boldsymbol{B}_f^\mathsf{T}$, per the definition of BTD.

Algorithm 3 Combining procedure for coupled-modes of BTD

Input: $\{\widetilde{\boldsymbol{M}}_{f,t}\},\ f=1,\cdots,F,\ t=1,\cdots,T$
Output: $\{\boldsymbol{A}_f\boldsymbol{B}_f^\mathsf{T}\},\ f=1,\cdots,F$

1: Permute factors using permutation matrices determined from $\boldsymbol{C}$ factor;
2: Resolve scaling ambiguity by dividing the (1,1) entries, yielding $\{\widehat{\boldsymbol{M}}_{f,t}\},\ f=1,\cdots,F,\ t=1,\cdots,T$;
3: For each $f=1,\cdots,F$, solve for $\boldsymbol{A}_f\boldsymbol{B}_f^\mathsf{T}$ according to (36a).

5 Numerical Results

In this section, the performance of our presented approach for different problem settings is measured. For CPD, 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 CPD (without any compression) as a baseline, for comparison purposes. For the case of BTD, we will compare with the NLS [40] method, which is the state-of-the-art algorithm for BTD. We also note that ParaSketch is a meta-algorithm that can use any tensor decomposition algorithm as a building block for decomposing the individual tensor sketches. We conduct our experiments on a Linux workstation with 16 parallel processors, and 128 GB RAM.

We first focus on CPD. Synthetic data is generated to test factor estimation accuracy and run time performance. Specifically, we generate factors $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}\}$, and use them to synthesize (see (1)) 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

$$\text{NMSE} = \min_{\substack{\pi\in\Pi,\\ c_f\in\{1,-1\}}} \frac{1}{F}\sum_{f=1}^{F}\left\|\frac{\boldsymbol{A}(:,f)}{\|\boldsymbol{A}(:,f)\|_2} - c_f\frac{\widehat{\boldsymbol{A}}(:,\pi_f)}{\|\widehat{\boldsymbol{A}}(:,\pi_f)\|_2}\right\|_2^2 \tag{38}$$

where $\widehat{\boldsymbol{A}}$ is the estimated factor, $\boldsymbol{A}$ is the ground truth factor, Π is the set of all permutations of the set $\{1,2,\cdots,F\}$, and π_f is the corresponding f after applying the permutation π. The permutation π is needed due to the inherent permutation ambiguity of CPD. Finding the best π is the celebrated linear assignment problem, which can be efficiently solved by the Hungarian algorithm. Symbol c_f is to resolve sign ambiguity.

5.1 Comparing ParaSketch to baselines

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 dimensions of each sketched replica 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 direct CPD, which doesn't scale well for dense tensors. The number of common rows in each replica is set to 3. The factor $\boldsymbol{A}$ 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 $\eta=\frac{TL}{I\ln(I)}$ takes values between 1 and 3.5. To factor the compressed small tensors, we use the same CPD solver provided in the N-way toolbox[1] [41] for both ParaSketch and PARACOMP. 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 4. As can be seen, the proposed ParaSketch performs as well as PARACOMP once enough replicas are employed. As we point out in Remark 5, the data compression ratio is $\frac{I^2}{\eta L^2\ln I}$. In this experiment, we have $I=512, L=64$, the compression ratio is $\frac{I^2}{\eta L^2\ln I}=3.55$ 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 1 and Figure 5).

In the second experiment, we compare the run time performance of the proposed algorithm against baseline methods. 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 $T_{\text{ParaSketch}}=\eta(I\ln I/L)$. For PARACOMP, we set $T_{\text{PARACOMP}}=3I$, since we observed that this gives good factor estimation accuracy. We repeat each experiment 20 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 ParaSketch and PARACOMP. We include two other strong baselines for comparison: the CP-OPT method proposed in [42] and the NLS method from [40]. The CP-OPT method is a first-order method, aiming at tackling large scale tensor factorization. The NLS method [40] exploits structure of tensor factorization to derive an

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2982144, IEEE Transactions on Knowledge and Data Engineering

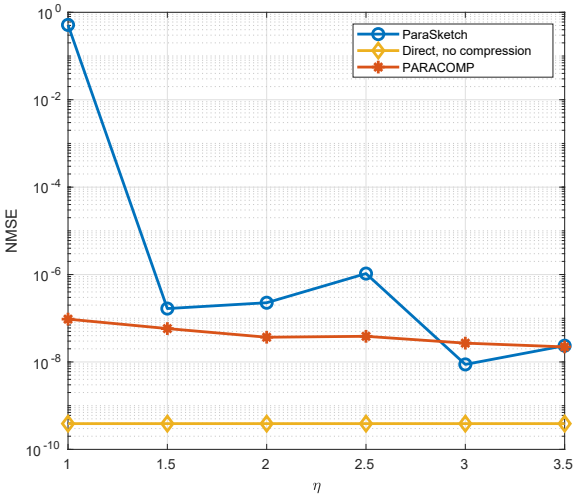

Fig. 4: The NMSE of estimating $\boldsymbol{A}$

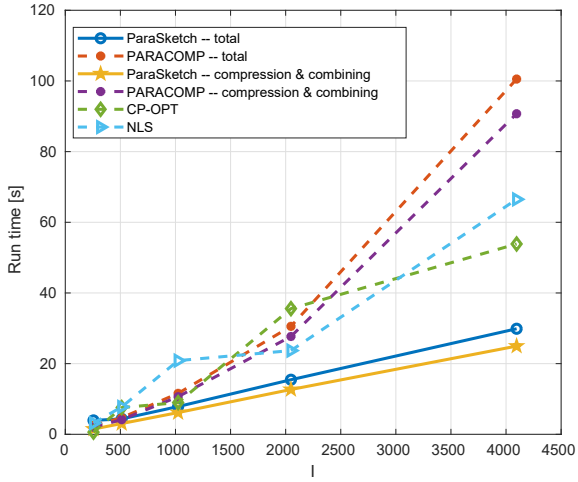

Fig. 5: Run time comparison for CPD

efficient implementation of Levenberg-Marquardt algorithm [43]. Both these two methods have well-documented success in the literature. For this experiment, we use a stopping criterion of 10^{-6} tolerance for all the algorithms: if the cost function changes less than this threshold, the algorithms are terminated, otherwise the algorithm is run till the maximum number of steps set by the authors of each algorithm.

The run time results are shown in Figure 5. 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 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 5 verify the complexity analysis presented in Table 1. The methods CP-OPT and NLS require longer running time compared with the proposed ParaSketch especially when the tensor gets large. This comparison highlights the necessity of exploiting parallel processing for large scale tensor factorization.

In Figure 6, we show factor estimation performance under different noise power levels. The noise level is quantified by signal to noise ratio (SNR), which is defined as

$$\text{SNR} = 10\log_{10}\left(\frac{\|\underline{\boldsymbol{X}}\|_F^2}{(\#\text{ of elements in }\underline{\boldsymbol{X}})\times\sigma^2}\right), \qquad (39)$$

where σ^2 is the variance of Gaussian noise added to each entry of the tensor. The dimensions of the tensor are shown in the figure. As expected, the estimation performance improves as the SNR get larger (i.e. relatively weaker noise). Again, one can observe that the proposed method achieves similar estimation performance as PARACOMP.

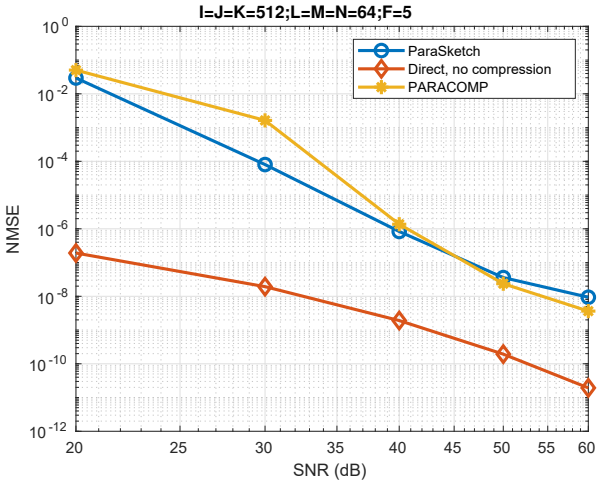

Fig. 6: Factor estimation performance under different SNR.

5.2 Comparison with joint factorization approach

In order to identify the advantages and disadvantages of the ParaSketch approach compared to the joint factorization approach, we conduct several simulations assessing the estimation accuracy of the original factors as well as the computational performance.

First, we test the ability of the ParaSketch algorithm and the joint factorization approach to identify the low-rank factors of the original tensor. We measure and report the averaged normalized mean squared error (NMSE) in estimating the low-rank factors over 100 simulations. In each simulation, the size of the original tensor is $512 \times 128 \times 32$ where the rank of the underlying factors is 5. A total of 128 sketched tensors of size $64 \times 16 \times 4$ are created. Then, both approaches are used to identify the factors of the original tensor. From Figure 7, it is clear that the joint approach is more accurate in identifying the factors when the noise level is high. The ParaSketch algorithm requires higher SNR in order to identify the low-rank factors with an acceptable accuracy.

However, run time for the algorithms shows large difference. For this experiment, the ParaSketch method factors the sketched tensors on all the 16 parallel processors. The average time needed by the joint factorization approach is 150.0 seconds, and the average time needed by ParaSketch

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2982144, IEEE Transactions on Knowledge and Data Engineering

to factor the sketched tensors and combine the resulting factors is 1.0 seconds – a 150 times difference. Note also that both algorithms offer higher accuracy in estimating factor matrices of the smaller modes. An explanation for this is that the conditional least squares updates for the smaller modes are more over-determined, so assuming reasonable estimates for the other modes gives a higher 'coherent combining' advantage for these modes.

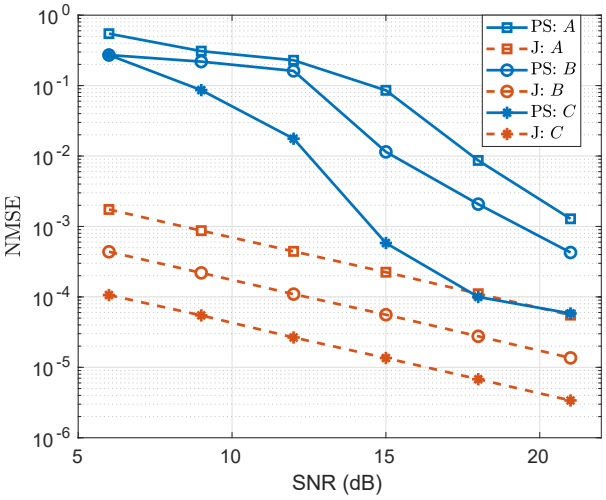

Fig. 7: Factor estimation performance under different SNR. PS – ParaSketch; J – Joint factorization.

In the next experiment, we compare the accuracy in estimating the original tensor factors using both approaches for different ranks. Again, we average the NMSE of the estimated low-rank factors over 100 simulations. The size of the original tensor is $64 \times 64 \times 32$ where the noise power is set such that the signal to noise ratio is 70 dB. A total of 64 sketched tensors of sizes $16 \times 16 \times 8$ are created, and then used to estimate the factors of the original tensor. In Figure 8, the estimation performance is depicted for both methods. It can be seen that the considered approaches can accurately identify the low-rank factors when the rank of the factors is relatively small. However, when the rank of the original tensor grows, the joint factorization approach is more robust in identifying the low-rank factors than the ParaSketch approach. This confirms our intuition that using parallel processing and combining results from multiple processors can introduce some accuracy degradation, as discussed in Section 1. On the other hand, Figure 9 depicts the average time required by both methods to estimate the low-rank factors from the sketched tensors. The ParaSketch approach is a least an order (and sometimes close to two orders) of magnitude faster than the joint factorization approach for all values of rank.

5.3 Simulation for BTD

We next perform simulations to evaluate the proposed method for BTD. As mentioned earlier, the state of the art algorithm for BTD is NLS. We compare the proposed method with directly applying NLS to the large tensor. The subproblems in the proposed method are also solved with the same NLS method. In this experiment, the initialization

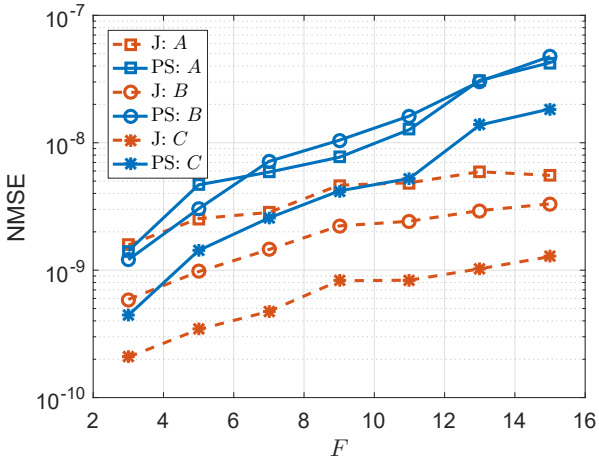

Fig. 8: Factor estimation performance for different ranks. PS – ParaSketch; J – Joint factorization.

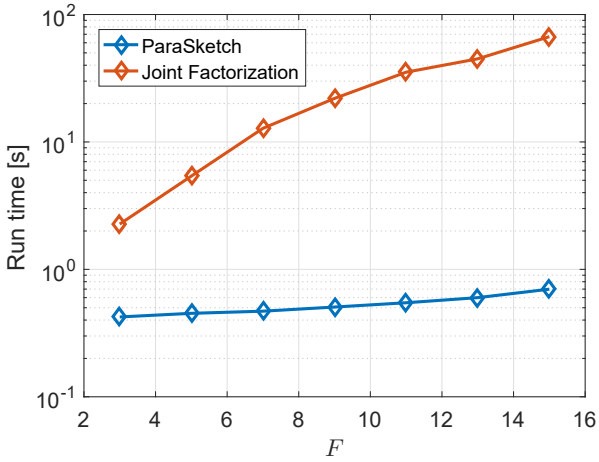

Fig. 9: Computational time of ParaSketch and joint factorization for different ranks.

method for NLS is set to generalized eigenvalue decomposition, as implemented in [44]. We set the dimensions to be $J = 256,\ K = 1024$, and vary I. The compression dimensions are $L = I/8$ and $N = 128$. The rank parameters are set to $F = R = 4$, i.e. there are four "block terms", within which the first and second modes form rank-4 matrices. Zero-mean Gaussian noise with standard deviation $\sigma = 10^{-3}$ is added to the elements of the generated tensor.

The result for run time comparison is shown in Figure 10. As can be seen, the run time of the proposed method scales more favorably than directly applying NLS on the large tensor.

BTD brings some complications for testing factor estimation performance: the first and second modes have linear-transformation ambiguity within each block term, which means e.g. $\widehat{\boldsymbol{A}}_f$ and $\widehat{\boldsymbol{B}}_f$ are *not* essentially unique as opposed to CPD. In order to calculate NMSE as defined in (38) , we adopt the following strategy: we multiply each pair of ($\widehat{\boldsymbol{A}}_f$, $\widehat{\boldsymbol{B}}_f$) and vectorize it into a vector of size $IJ \times 1$, which are then collected into a matrix of size $IJ \times F$. This matrix

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2982144, IEEE Transactions on Knowledge and Data Engineering

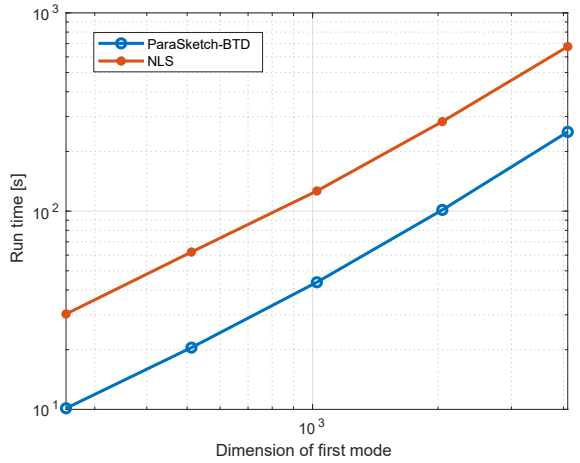

Fig. 10: Run time comparison for BTD

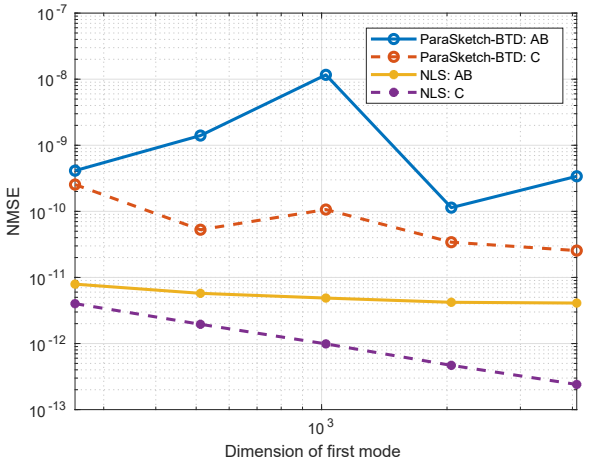

Fig. 11: NMSE comparison for BTD

is compared, using metric (38), with the matrix produced from the ground truth factors with the same "multiplying-vectorization" process. The results are shown in Figure 11. As expected, notwithstanding some degradation in accuracy compared to applying NLS directly, the proposed method estimates factors with high accuracy.

5.4 Real world data mining

In the last experiment, we test our algorithm on a dataset of taxi trajectories in Beijing during the Chinese New Year (a major Chinese festival) week of 2008 [45]. 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 in time. Then, the ParaSketch algorithm for CPD 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. We set the rank parameter in ParaSketch to 5 in this experiment.

In Figure 12, we visualize the resulting most significant factor, which is defined as having the largest sum of squared

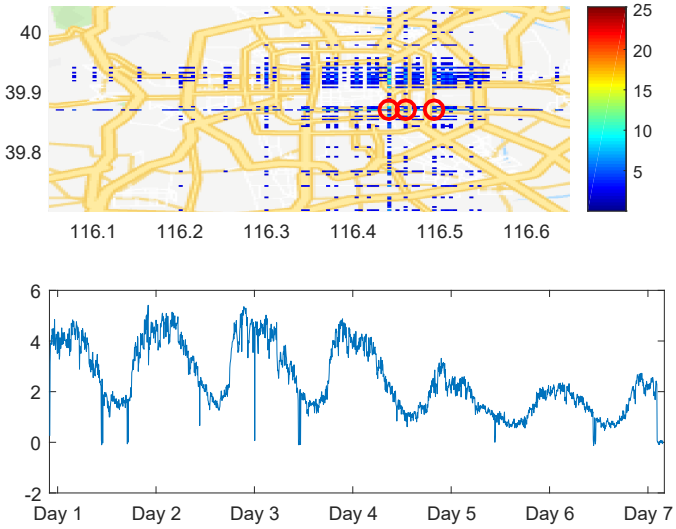

Fig. 12: Beijing taxi trajectory data analysis. Top: magnitude of the rank-1 matrix, formed by multiplying the first and second mode vectors of the most significant rank-1 factor of the tensor, visualized on the map. The red circles identify the three locations with the largest magnitude. Bottom: the corresponding third mode, showing temporal variation.

ℓ_2-norm of the three vectors corresponding to a rank-1 term. In the top sub-figure, the three markers correspond to the three largest absolute values of the rank-1 matrix formed by multiplying the first and second mode vectors. After cross-examining with the map of Beijing, we discover that these locations correspond to famous attractions: Temple of Heaven, Longtan Lake Park, and Happy Valley Amusement Park. The high intensity of taxi activities highlights their popularity during this festive period. In the bottom sub-figure, we show the magnitude of corresponding vector 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 see that there is a decline in taxi activity for the last 3 days. Inspection of the exact dates reveals that these days are the New Year's Eve (Day 5) and Spring Festival (Day 6, 7) – those are days for family reunions, hence fewer people on the road.

6 Conclusions

We propose an algorithm to facilitate *parallel* CPD and BTD on large tensor data. The approach provides great acceleration over existing prior art, rendering itself suitable for 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, for both CPD and BTD. We also characterized recovery conditions for the proposed approach, i.e., the number of rows in the sketching matrices and the 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.

ACKNOWLEDGMENT

The authors gratefully acknowledge support from NSF IIS-1447788, CIF-1525194, and IIS-1704074.

REFERENCES

[1] B. Yang, A. Zamzam, and N. D. Sidiropoulos, "ParaSketch: Parallel tensor factorization via sketching," in *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 2018, pp. 396–404.

[2] T. G. Kolda, B. W. Bader, and J. P. Kenny, "Higher-order web link analysis using multilinear algebra," in *Proceedings of the Fifth IEEE International Conference on Data Mining*. IEEE, 2005, pp. 8–pp.

[3] B. W. Bader, R. A. Harshman, and T. G. Kolda, "Temporal analysis of semantic graphs using ASALSAN," in *Proceedings of Seventh IEEE International Conference on Data Mining*. IEEE, 2007, pp. 33–42.

[4] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender Systems*. ACM, 2010, pp. 79–86.

[5] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme, "Learning optimal ranking with tensor factorization for tag recommendation," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 727–736.

[6] N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme, "Recommender system for predicting student performance," *Procedia Computer Science*, vol. 1, no. 2, pp. 2811–2819, 2010.

[7] B. Hidasi and D. Tikk, "Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 67–82.

[8] B. Yang, G. Wang, and N. D. Sidiropoulos, "Tensor completion via group-sparse regularization," in *2016 50th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2016, pp. 1750–1754.

[9] A. Anandkumar, R. Ge, D. J. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2773–2832, 2014.

[10] C. Hu, P. Rai, C. Chen, M. Harding, and L. Carin, "Scalable Bayesian non-negative tensor factorization for massive count data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 53–70.

[11] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro, "From K-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors," *IEEE transactions on Signal Processing*, vol. 61, no. 2, pp. 493–506, 2013.

[12] B. Yang, X. Fu, and N. D. Sidiropoulos, "Learning from hidden traits: Joint factor analysis and latent clustering," *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 256–269, 2017.

[13] I. Perros, E. E. Papalexakis, F. Wang, R. Vuduc, E. Searles, M. Thompson, and J. Sun, "SPARTan: Scalable PARAFAC2 for large & sparse data," *arXiv preprint arXiv:1703.04219*, 2017.

[14] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[15] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[16] A. H. Phan and A. Cichocki, "Parafac algorithms for large-scale problems," *Neurocomputing*, vol. 74, no. 11, pp. 1970–1984, 2011.

[17] U. Kang, E. Papalexakis, A. Harpale, and C. 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*. ACM, 2012, pp. 316–324.

[18] J. H. Choi and S. Vishwanathan, "DFacTo: Distributed factorization of tensors," in *Advances in Neural Information Processing Systems*, 2014, pp. 1296–1304.

[19] A. L. De Almeida and A. Y. Kibangou, "Distributed large-scale tensor decomposition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 26–30.

[20] N. D. Sidiropoulos, E. E. Papalexakis, and C. Faloutsos, "Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 57–70, 2014.

[21] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "ParCube: Sparse parallelizable CANDECOMP-PARAFAC tensor decomposition," *ACM Transactions on Knowledge Discovery from Data*, vol. 10, no. 1, p. 3, 2015.

[22] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[23] E. S. Allman, C. Matias, J. A. Rhodes *et al.*, "Identifiability of parameters in latent structure models with many observed variables," *The Annals of Statistics*, vol. 37, no. 6A, pp. 3099–3132, 2009.

[24] N. Kargas, N. D. Sidiropoulos, and X. Fu, "Tensors, learning, and Kolmogorov extension for finite-alphabet random vectors," *IEEE Transactions on Signal Processing*, vol. 66, no. 18, pp. 4854–4868, 2018.

[25] R. Bro, R. A. Harshman, N. D. Sidiropoulos, and M. E. Lundy, "Modeling multi-way data with linearly dependent loadings," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 23, no. 7-8, pp. 324–340, 2009.

[26] L. De Lathauwer, "Decompositions of a higher-order tensor in block termspart II: Definitions and uniqueness," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1033–1066, 2008.

[27] D. P. Woodruff *et al.*, "Sketching as a tool for numerical linear algebra," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.

[28] C. Battaglino, G. Ballard, and T. G. Kolda, "A practical randomized CP tensor decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 39, no. 2, pp. 876–901, 2018.

[29] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, "A flexible and efficient algorithmic framework for constrained matrix and tensor factorization," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5052–5065, 2016.

[30] S. Smith, A. Beri, and G. Karypis, "Constrained tensor factorization with accelerated ao-admm," in *2017 46th International Conference on Parallel Processing (ICPP)*. IEEE, 2017, pp. 111–120.

[31] S. Ono and T. Kasai, "Efficient constrained tensor factorization by alternating optimization with primal-dual splitting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 3379–3383.

[32] J. B. Kruskal, "Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear Algebra and its Applications*, vol. 18, no. 2, pp. 95–138, 1977.

[33] N. D. Sidiropoulos and A. Kyrillidis, "Multi-way compressed sensing for sparse low-rank tensors," *IEEE Signal Processing Letters*, vol. 19, no. 11, pp. 757–760, 2012.

[34] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary Mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[35] N. Ailon and B. Chazelle, "Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform," in *Proceedings of the thirty-eighth annual ACM Symposium on Theory of Computing*. ACM, 2006, pp. 557–563.

[36] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, "Faster least squares approximation," *Numerische mathematik*, vol. 117, no. 2, pp. 219–249, 2011.

[37] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener, "Multiway analysis of epilepsy tensors," *Bioinformatics*, vol. 23, no. 13, pp. i10–i18, 2007.

[38] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[39] P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229, 1992.

[40] L. Sorber, M. Van Barel, and L. De Lathauwer, "Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-(l_r,l_r,1) terms, and a new generalization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 695–720, 2013.

[41] C. A. Andersson and R. Bro, "The N-way toolbox for MATLAB," *Chemometrics and intelligent laboratory systems*, vol. 52, no. 1, pp. 1–4, 2000.

[42] E. Acar, D. M. Dunlavy, and T. G. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *Journal of Chemometrics*, vol. 25, no. 2, pp. 67–86, 2011.
[43] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.
[44] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. (2016, Mar.) Tensorlab 3.0. Available online. [Online]. Available: https://www.tensorlab.net
[45] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 316–324.

Bo Yang (S'15) received the B.Eng. and M.Eng. degrees in communication and information system from the Huazhong University of Science and Technology of China, Wuhan, China, in 2011 and 2014, respectively. Since 2014, he has been working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA. His research interests include data analytics and machine learning, with an emphasis on unsupervised learning, using tools such as matrix and tensor factorization, as well as (deep) neural networks. He received a Best Student Paper Award at IEEE CAMSAP 2015.

Ahmed S. Zamzam (S'14) is a PhD Candidate at the Department of Electrical and Computer Engineering at the University of Minnesota, where he is also affiliated with the Signal and Tensor Analytics Research (STAR) group under the supervision of Professor N. D. Sidiropoulos. Previously, he earned his BSc at Cairo University in 2013. In 2015, he received the MSc from Nile University. Ahmed received the Louis John Schnell Fellowship (2015), and the Doctoral Dissertation Fellowship (2018) from the University of Minnesota. He also received Student Travel Awards from the IEEE Signal Processing Society (2017), the IEEE Power and Energy Society (2018), and the Council of Graduate Students at the University of Minnesota (2016, 2018). His research interests include control and optimization of smart grids, large-scale complex energy systems, grid data analytics, and machine learning.

Nicholas D. Sidiropoulos (F'09) received the Diploma degree in electrical engineering from Aristotelian University of Thessaloniki, Thessaloniki, Greece, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland-College Park, College Park, MD, USA, in 1988, 1990, and 1992, respectively. He has served on the faculty of the University of Virginia (UVA), University of Minnesota, and the Technical University of Crete, Greece, prior to his current appointment as Chair of ECE at UVA.

His research interests are in signal processing, communications, optimization, tensor decomposition, and factor analysis, with applications in machine learning and communications. He received the NSF/CAREER award in 1998, the IEEE Signal Processing Society (SPS) Best Paper Award in 2001, 2007, and 2011, served as IEEE SPS Distinguished Lecturer (2008-2009), and currently serves as Vice President - Membership of IEEE SPS. He received the 2010 IEEE Signal Processing Society Meritorious Service Award, and the 2013 Distinguished Alumni Award from the University of Maryland, Dept. of ECE. He is a Fellow of IEEE (2009) and a Fellow of EURASIP (2014).