A Stochastic Algorithm for the ParaTuck Decomposition

Yassine Zniyed, André L.F. de Almeida

A Stochastic Algorithm for the ParaTuck Decomposition

Yassine Zniyed[a], André L.F. de Almeida[b]

[a] *University of Toulon, Aix-Marseille University, CNRS, LIS UMR 7020, France*
[b] *Department of Teleinformatics Engineering, Federal University of Ceara, Brazil*

Abstract

This paper introduces a novel stochastic algorithm for the ParaTuck Decomposition (PTD), addressing the challenge of local minima encountered in the traditional alternating least squares (ALS) approach. The proposed method integrates stochastic steps into the ALS framework to avoid the common swamp problems, where numerical difficulties prevent accurate decompositions. Our simulations indicate good convergence properties for PTD, suggesting a potential increase in the efficiency and reliability of this tensor decomposition across various applications.

Keywords: Paratuck, tensor decompositions, alternating least squares

1. Introduction

Tensor decomposition and multilinear algebra have emerged as essential tools in various fields, particularly signal processing. They provide powerful means to represent complex data structures and have found wide applications in solving diverse problems. In recent years, with machine learning and the growing importance of data science, tensor decomposition techniques have gained even greater prominence. These techniques enable us to efficiently analyze and extract meaningful information from high-dimensional data, making them invaluable for feature extraction, dimensionality reduction, and pattern recognition tasks.

Several tensor models exist for various applications. Among the most classical and widely used in the literature are the Canonical Polyadic Decomposition (CPD) [1, 2], Tucker Decomposition (TD) [3], and Tensor Train Decomposition (TTD) [4]. The CPD is a compact representation that decomposes a tensor into a sum of rank-1 tensors and is highly valued for its

uniqueness properties. This decomposition is very useful for parameter estimation, with the Alternating Least Squares (ALS) [5, 6] algorithm being the workhorse for computing the CPD. A more general model than the CPD is the Tucker Decomposition. This decomposition represents a tensor with a core tensor of the same order as the original tensor and factor matrices. A particular case of this decomposition is the High-Order Singular Value Decomposition (HOSVD) [7], corresponding to a Tucker model with orthonormal factor matrices. The TD model is extensively used in subspace-based signal processing applications, such as data compression and data denoising [8]. Recently, the tensor train decomposition has gained attention. Its compact representation scales linearly with the tensor order, making it suitable for high-order tensor data [9]. Initially used in super-compression applications, TTD has recently become useful for parameter identification due to its equivalence with CPD [10]. Among the algorithms facilitating this decomposition are TT-SVD [4] and its hierarchical version, referred to as TT-HSVD [11].

This paper focuses on a specific tensor decomposition known as the ParaTuck decomposition (PTD) [1]. Several names have appeared in the literature for this decomposition, such as ParaTuck2 in [1, 5] and ParaTuck$-(N_1, N)$ in [12]. For simplicity, we use the name ParaTuck throughout this paper, as used in [1]. Initially proposed in the psychometric literature in 1994, PTD has garnered limited attention since its inception. One of the primary reasons for its under-utilization in practical applications has been the absence of a reliable algorithm for its computation. The name "ParaTuck" derives from its resemblance to two tensor decomposition techniques: PARAFAC, also referred to as the CPD and the Tucker decomposition. PTD can be seen as a unique blend of these methods, offering a two-level generalization of CPD, as elucidated in [13]. The PTD is a generalization of the CPD in the sense that PTD reduces to a CPD if one of the PTD factors is diagonal; at the same time, CPD can be seen as a special Tucker decomposition with a diagonal core tensor, the PTD can also be seen as a special Tucker-2 decomposition with a structured core tensor as will be shown later.

Even if PTD has not found multiple fields of application up to now, one notable area of exploitation is wireless communications [14]. In [15], the PTD structure has been exploited to design coding structures combining spreading and multiplexing across space and time. A generalization of this model to multi-input multi-output (MIMO) communication systems based on multi-

carrier transmission was later proposed in [16]. Furthermore, [17, 18] uses the PTD to model MIMO multi-hop relaying systems, proposing a receiver under a supervised scenario. Additionally, the authors of [19] have proposed semi-blind receivers for MIMO relaying multi-hop communication systems and demonstrated that the signals received at the destination follow a PTD model. In these applications, *a priori* knowledge of the coding structure provides valuable insights into certain factors of the PTD. When one or several factors of the PTD are known, the decomposition problem is easier than the general case, and generally, we have no difficulty finding the remaining factors of the decomposition. Furthermore, recent developments in [13] have shown that the PTD aligns well with two-layer neural networks with flexible activation functions. This discovery opens new horizons for leveraging this model in neural network learning. However, the authors could not propose a decomposition based on the PTD model directly; instead, they used second-order information from the neural networks, which aligns with a CPD model, to retrieve some factors. A part from that, the broader utility of PTD has remained limited due to the algorithmic challenges associated with its computation.

The uniqueness and identifiability of the Paratuck decomposition have not received extensive investigation, but experimental findings in [20] suggest its uniqueness in specific scenarios. Uniqueness means a tensor can be expressed in a single PTD-based format up to scaling and permutation indeterminacies. This uniqueness is important, particularly in source separation problems, where the rank-1 terms can be easily associated with interpretable data components. We recall in the sequel these PTD uniqueness conditions and how the scaling and permutation ambiguities are expressed.

Just like for the CPD, Rasmus Bro proposed an alternating least squares (ALS)-type algorithm for computing the PTD [21]. Bro's approach involves fixing all factors except one and optimizing a loss function for that particular factor. While this strategy has proven effective for CPD, it encounters significant challenges when applied to PTD. Unfortunately, despite extensive computations and iterations, this ALS-based approach for PTD often falls prey to numerical difficulties, primarily attributed to local minima. These numerical issues result in what is commonly referred to as "swamp" problems, making it exceedingly challenging to obtain accurate decompositions of tensors using the traditional ALS method. Since this initial attempt, no alternative algorithmic solution has been proposed for efficiently and reliably computing the PTD. Recently, another attempt was made in [22], where the

author proposed an algebraic solution. However, this solution is limited to where PTD ranks equal 2. The absence of an efficient algorithm for the PTD, other than the plain vanilla ALS, has limited its exploitation to cases where *a priori* information about the decomposition is known. However, decomposing a tensor in the general case remains a significant challenge, where no prior knowledge of the PTD is available. The probability of success for the ALS algorithm to converge is very low, except in rare instances where we have "lucky" initializations.

Finding the global optimum of the objective function can be a daunting challenge for tensor decompositions. Numerical results often reveal that traditional deterministic optimization algorithms, such as the ALS, may struggle to discover solutions in complex scenarios like the PTD. One plausible explanation for this difficulty is the presence of multiple local optima within the objective function. To address this issue, stochastic optimization algorithms offer a promising alternative approach. Stochastic optimization refers to a collection of methods for optimizing an objective function when randomness is present, and they replace the classical methods when optimization becomes too complex for different reasons. These methods embrace randomness as an integral part of the search procedure, allowing for less good local decisions during optimization. Several examples of stochastic optimization methods have emerged in the context of tensor decompositions, such as [23, 24, 25]. However, it is worth noting that, aside from the limited attempts we mentioned earlier, there remains a scarcity of such techniques tailored specifically for the PTD problem.

In light of the absence of a reliable algorithm for PTD, this paper presents a novel randomized/stochastic ALS-type algorithm as a solution to this long-standing challenge. Our proposed stochastic algorithm draws inspiration from Bro's original ALS method and incorporates ideas from stochastic optimization strategies. The fundamental concept behind our algorithm is to augment the conventional ALS with stochastic steps to mitigate the issues related to local minima. In essence, we minimize the same cost function as the original ALS but introduce a random solution test in each iteration. This stochastic element injects a degree of randomness into the optimization process, steering our algorithm away from the pitfalls of local minima inherent in the standard ALS approach. Simulation results demonstrate that our stochastic algorithm significantly enhances the convergence prospects for PTD. In simulations, the stochastic algorithm shows promise in improving the convergence likelihood of PTD compared to traditional ALS, which of-

ten faces convergence challenges despite the use of various initializations. This result underscores the potential of this algorithm to address new challenges associated with PTD and pave the way for more efficient and reliable applications of this tensor decomposition method in various domains. The contributions of this paper can be summarized as follows:

- We propose a stochastic algorithm for the PTD. This algorithm adds a controlled degree of randomness to the factor matrices' updating steps to avoid swamps.
- We discuss the complexity of the proposed stochastic PTD algorithm and its convergence, demonstrating that it has the same convergence guarantees as the classic ALS for the CPD model.
- We evaluate the proposed algorithm regarding convergence rate and robustness, considering noiseless and noisy contexts.

2. Background on the ParaTuck decomposition

2.1. Notations

The notations used throughout the rest of this paper are now defined. The symbol $(\cdot)^\dagger$ denotes the pseudo-inverse. The Hadamard, Kronecker, outer product, and Khatri-Rao products are denoted by $\boxdot$, $\otimes$, $\circ$, and $\odot$, respectively. We write the q-mode product as $\times_q$ [8]. Tensors are represented by bold calligraphic capital letters, e.g., $\boldsymbol{\mathcal{X}}$. The 3rd-order identity tensor is written as $\mathcal{I}_{3,R}$, where R is the size of its dimensions. $\text{unfold}_q\boldsymbol{\mathcal{X}}$ refers to the unfolding of tensor $\boldsymbol{\mathcal{X}}$ over its q-th mode [8]. The operator $\text{vec}(\cdot)$ forms a vector by stacking the columns of its matrix argument. The operator $\text{diag}(\cdot)$ forms a diagonal matrix from its vector argument. A Hadamard product between a matrix $\mathbf{A}$ of size $I \times J$ and a tensor $\boldsymbol{\mathcal{B}}$ of size $I \times J \times K$, along their common dimensions, is defined as $\boldsymbol{\mathcal{C}} = \mathbf{A} \boxdot_{\{I,J\}} \boldsymbol{\mathcal{B}}$, with $\boldsymbol{\mathcal{C}}$ of size $I \times J \times K$, and whose entries are expressed as $\boldsymbol{\mathcal{C}}_{i,j,k} = \mathbf{A}_{i,j} \cdot \boldsymbol{\mathcal{B}}_{i,j,k}$.

2.2. Paratuck decomposition

Definition 2.1. *A 3rd-order ParaTuck decomposition of a $I \times J \times K$ tensor $\boldsymbol{\mathcal{X}}$ is defined by its mode-3 slices as*

$$\boldsymbol{\mathcal{X}}_{:,:,k} = \mathbf{A} \cdot \mathbf{D}^k_{\mathbf{A}} \cdot \mathbf{H} \cdot \mathbf{D}^k_{\mathbf{B}} \cdot \mathbf{B}^T, \tag{1}$$

where $\boldsymbol{\mathcal{X}}_{:,:,k}$ *is an* $I \times J$ *matrix,* $\mathbf{A}$ *and* $\mathbf{B}$ *are loading matrices of size respectively* $I \times R$ *and* $J \times S$*. The diagonal matrices* $\mathbf{D}^k_{\mathbf{A}}$ *and* $\mathbf{D}^k_{\mathbf{B}}$ *are respectively of dimensions* R *and* S*. A matrix* $\mathbf{C}_A$ *of size* $K \times R$ *can be constructed, whose* k*-th row equals the diagonal of* $\mathbf{D}^k_{\mathbf{A}}$*. Similarly, we can define a* $K \times S$ *matrix* $\mathbf{C}_B$ *with respect to* $\mathbf{D}^k_{\mathbf{B}}$*. Finally, matrix* $\mathbf{H}$ *is of size* $R \times S$ *and contains the weights of the interactions. For a given set of indices* (i, j, k)*, the entry* $\boldsymbol{\mathcal{X}}_{i,j,k}$ *is defined as [12]:*

$$\boldsymbol{\mathcal{X}}_{i,j,k} = \sum_{r=1}^{R} \sum_{s=1}^{S} \mathbf{H}_{r,s} \cdot \mathbf{A}_{i,r} \cdot \mathbf{B}_{j,s} \cdot (\mathbf{C}_A)_{k,r} \cdot (\mathbf{C}_B)_{k,s}. \tag{2}$$

Another way to express the ParaTuck model can be found in [12], and it is expressed as $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A} \times_2 \mathbf{B}$, with the core tensor $\boldsymbol{\mathcal{G}}$ of size $R \times S \times K$, expressed using a Hadamard product as $\boldsymbol{\mathcal{G}} = \mathbf{H} \boxdot_{\{R,S\}} \boldsymbol{\mathcal{F}}$, where $\boldsymbol{\mathcal{F}}$, of size $R \times S \times K$, follows a CPD such that $\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{I}}_{3,K} \times_1 \mathbf{C}_A^T \times_2 \mathbf{C}_B^T$. It should also be noted that one can rewrite (1) using the Khatri-Rao and Kronecker products as (eq. (33) in [5]):

$$\text{unfold}_3 \boldsymbol{\mathcal{X}} = \left(\mathbf{C}_B^T \odot \mathbf{C}_A^T\right)^T \cdot \text{diag}(\text{vec}\mathbf{H}) \cdot \left(\mathbf{B} \otimes \mathbf{A}\right)^T. \tag{3}$$

Using (3), we can reformulate $\boldsymbol{\mathcal{X}}$ as a structured CPD, expressed as:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{I}}_{3,(R\cdot S)} \times_1 \underbrace{(\mathbf{A} \cdot \phi_1)}_{I \times (R\cdot S)} \times_2 \underbrace{(\mathbf{B} \cdot \phi_2)}_{J \times (R\cdot S)} \times_3 \underbrace{\left(\left(\mathbf{C}_B^T \odot \mathbf{C}_A^T\right)^T \cdot \text{diag}(\text{vec}\mathbf{H})\right)}_{K \times (R\cdot S)}, \tag{4}$$

with $\phi_1 = \mathbf{1}_S^T \otimes \mathbf{I}_R$ and $\phi_2 = \mathbf{I}_S \otimes \mathbf{1}_R^T$. An equivalent compact notation for the PTD that will be used in the sequel is $\boldsymbol{\mathcal{X}} = [\![\mathbf{A}, \mathbf{D}^k_{\mathbf{A}}, \mathbf{H}, \mathbf{D}^k_{\mathbf{B}}, \mathbf{B}]\!]$.

2.3. Uniqueness and ambiguities

The uniqueness of the model has not been much studied. The results of [20], in the case where $R = S$, stated that the ParaTuck decomposition can be unique in the same sense as the CPD (*i.e.*, up to trivial scaling and permutation ambiguities), under some mild conditions. In essence, the authors stated that the ParaTuck model will be mostly unique if the loading matrices are of full rank, matrix $\mathbf{H}$ has no zero elements, and the dimensions of the array are not too small [21]. For example, in the case of $R = S = 2$, we should have $K \geq 9$, and $K \geq 5$ if $\mathbf{C}_A = \mathbf{C}_B$. In [20], the authors also showed that ambiguities between two equivalent decompositions $[\![\mathbf{A}, \mathbf{D}^k_{\mathbf{A}}, \mathbf{H}, \mathbf{D}^k_{\mathbf{B}}, \mathbf{B}]\!]$

and $[\![\tilde{\mathbf{A}}, \tilde{\mathbf{D}}^k_{\mathbf{A}}, \tilde{\mathbf{H}}, \tilde{\mathbf{D}}^k_{\mathbf{B}}, \tilde{\mathbf{B}}]\!]$ are given by $\mathbf{A} = \tilde{\mathbf{A}} \cdot (\mathbf{\Pi_A} \cdot \mathbf{\Lambda_A})$, $\mathbf{B} = \tilde{\mathbf{B}} \cdot (\mathbf{\Pi_B} \cdot \mathbf{\Lambda_B})$, $\mathbf{H} = (\bar{\mathbf{\Lambda}}_{\mathbf{A}} \cdot \mathbf{\Lambda}_{\mathbf{A}}^{-1} \cdot \mathbf{\Pi}_{\mathbf{A}}^T) \cdot \tilde{\mathbf{H}} \cdot (\mathbf{\Pi_B} \cdot \mathbf{\Lambda}_{\mathbf{B}}^{-1} \cdot \bar{\mathbf{\Lambda}}_{\mathbf{B}})$, $\mathbf{D}^k_{\mathbf{A}} = (z_k \cdot \mathbf{\Pi}_{\mathbf{A}}^T) \cdot \tilde{\mathbf{D}}^k_{\mathbf{A}} \cdot (\mathbf{\Pi_A} \cdot \bar{\mathbf{\Lambda}}_{\mathbf{A}}^{-1})$ and $\mathbf{D}^k_{\mathbf{B}} = (z_k^{-1} \cdot \mathbf{\Pi}_{\mathbf{B}}^T) \cdot \tilde{\mathbf{D}}^k_{\mathbf{B}} \cdot (\mathbf{\Pi_B} \cdot \bar{\mathbf{\Lambda}}_{\mathbf{B}}^{-1})$, where $\mathbf{\Lambda_A}$, $\mathbf{\Lambda_B}$, $\bar{\mathbf{\Lambda}}_{\mathbf{A}}$ and $\bar{\mathbf{\Lambda}}_{\mathbf{B}}$ are diagonal matrices, $\mathbf{\Pi_A}$ and $\mathbf{\Pi_B}$ are permutation matrices, and z_k are nonzero scalars.

2.4. ALS algorithm

In the context of Paratuck decomposition, the optimization problem is formulated with a loss function defined for a 3rd-order tensor as follows:

$$\underset{\mathbf{A},\mathbf{D}^k_{\mathbf{A}},\mathbf{H},\mathbf{D}^k_{\mathbf{B}},\mathbf{B}}{\arg\min} \quad \sum_{k=1}^{K} ||\boldsymbol{\mathcal{X}}_{:,:,k} - \mathbf{A} \cdot \mathbf{D}^k_{\mathbf{A}} \cdot \mathbf{H} \cdot \mathbf{D}^k_{\mathbf{B}} \cdot \mathbf{B}^T||^2_F. \tag{5}$$

In [21], an ALS-type algorithm tailored for Paratuck decomposition was introduced. The ALS algorithm is a fundamental optimization technique used to solve tensor decompositions. It follows a step-by-step iterative approach. All factors except one are initially fixed, and the loss function (5) is minimized for the non-fixed factor. This process is repeated for all factors in a cyclic manner until convergence is achieved. The essence of ALS lies in its alternating optimization strategy, where each iteration aims to improve the approximation of the input tensor $\boldsymbol{\mathcal{X}}$ by gradually refining the factor matrices. While ALS has been effective in various tensor decomposition problems, it may suffer from issues such as local minima when applied to Paratuck decomposition, motivating the exploration of stochastic approaches to mitigate these challenges. The comprehensive description of the entire Paratuck-ALS algorithm [21] is given in Alg. 1. If interested in the details of the ALS iterations' derivation, we refer the reader to Section 4.5 of [21].

3. Proposed algorithm

3.1. Problem statement

One of the primary motivations for our proposed algorithm stems from empirical observations and simulations that highlight the challenges associated with the ALS method in Paratuck decomposition. ALS usually struggles to converge efficiently and tends to get trapped in what is commonly called "swamp" problems. These swamp problems can be attributed to local minima in the optimization landscape. As a result, the practical utility of PTD has been limited, particularly when none of the factors are known beforehand.

Algorithm 1 ALS for PTD [21]

Require: Input tensor $\boldsymbol{\mathcal{X}}$, ranks R and S
Ensure: $\mathbf{A}, \mathbf{D}_{\mathbf{A}}^k, \mathbf{H}, \mathbf{D}_{\mathbf{B}}^k$ and $\mathbf{B}$ (for $1 \leq k \leq K$)
Initialize $\mathbf{D}_{\mathbf{A}}^k, \mathbf{H}, \mathbf{D}_{\mathbf{B}}^k$ and $\mathbf{B}$ (for $1 \leq k \leq K$)
while a convergence criterion is not met do
  $\mathbf{A} = \text{unfold}_1 \boldsymbol{\mathcal{X}} \cdot \mathbf{F}_{\mathbf{A}}^{\dagger}$, with $\mathbf{F}_{\mathbf{A}} = \left[\mathbf{F}_{\mathbf{A}}^{(1)} \cdots \mathbf{F}_{\mathbf{A}}^{(K)}\right]$, and $\mathbf{F}_{\mathbf{A}}^{(k)} = \mathbf{D}_{\mathbf{A}}^k \cdot \mathbf{H} \cdot \mathbf{D}_{\mathbf{B}}^k \cdot \mathbf{B}^T$
  for $k = 1, \cdots, K$ do
    $\mathbf{D}_{\mathbf{A}}^k = \text{diag}\big((\mathbf{F}_{\tilde{\mathbf{A}}}^k \odot \mathbf{A})^{\dagger} \cdot \text{vec}(\boldsymbol{\mathcal{X}}_{:,:,k})\big)$, with $\mathbf{F}_{\tilde{\mathbf{A}}}^k = \mathbf{B} \cdot \mathbf{D}_{\mathbf{B}}^k \cdot \mathbf{H}^T$
  end for
  $\text{vec}\mathbf{H} = \mathbf{F}_{\mathbf{H}}^{\dagger} \cdot \text{vec}\boldsymbol{\mathcal{X}}$, with $\mathbf{F}_{\mathbf{H}} = \left[(\mathbf{B}\mathbf{D}_{\mathbf{B}}^1 \otimes \mathbf{A}\mathbf{D}_{\mathbf{A}}^1); \cdots; (\mathbf{B}\mathbf{D}_{\mathbf{B}}^K \otimes \mathbf{A}\mathbf{D}_{\mathbf{A}}^K)\right]$
  for $k = 1, \cdots, K$ do
    $\mathbf{D}_{\mathbf{B}}^k = \text{diag}\big((\mathbf{F}_{\tilde{\mathbf{B}}}^k \odot \mathbf{B})^{\dagger} \cdot \text{vec}(\boldsymbol{\mathcal{X}}_{:,:,k}^T)\big)$, with $\mathbf{F}_{\tilde{\mathbf{B}}}^k = \mathbf{A} \cdot \mathbf{D}_{\mathbf{A}}^k \cdot \mathbf{H}$
  end for
  $\mathbf{B} = \text{unfold}_2 \boldsymbol{\mathcal{X}} \cdot \mathbf{F}_{\mathbf{B}}^{\dagger}$, with $\mathbf{F}_{\mathbf{B}} = \left[\mathbf{F}_{\mathbf{B}}^{(1)} \cdots \mathbf{F}_{\mathbf{B}}^{(K)}\right]$, and $\mathbf{F}_{\mathbf{B}}^{(k)} = \mathbf{D}_{\mathbf{B}}^k \cdot \mathbf{H}^T \cdot \mathbf{D}_{\mathbf{A}}^k \cdot \mathbf{A}^T$
end while

To address these challenges, we propose the introduction of randomness into the optimization process. The underlying idea is to leverage stochastic optimization techniques to escape local minima and enhance the convergence prospects for PTD. Introducing randomness into optimization can take various forms, and our proposed approach aligns with the "Ruin & Recreate" strategy [26]. Stochastic optimization encompasses a wide range of techniques, including stochastic gradient descent [27], genetic algorithms [28], and evolution strategies [29], among others [30, 26]. We aim to apply a stochastic approach to the PTD problem, specifically tailored to mitigate the challenges posed by local minima, thus enhancing the algorithm's reliability and convergence.

3.2. Stochastic ALS

Stochastic algorithms have already been used in the context of tensor decompositions. We can cite, for example, the work in [31], where the authors have proposed a stochastic ALS algorithm for the CPD based on the collaborative evolution of a population. They put two candidates randomly picked from a population of solutions in competition. The best candidate is kept, a new candidate is created around this one, and the worst is deleted. In [32], the authors proposed another version of the stochastic ALS, also for the CPD case, where instead of updating the factors with the least squares solution, they update the factors as a weighted average of both the current solution and the least-square solution. To the best of our knowledge, this

is the first time that a stochastic algorithm is proposed for the ParaTuck decomposition, and the randomness is introduced in a "Ruin & Recreate" manner [26].

In the proposed stochastic solution, we aim to address the challenge of escaping bad local minima by introducing randomness through a strategic approach. While various strategies exist for introducing randomness, the conventional local search approach often results in small moves that do not effectively mitigate the problem in our context. Our novel approach involves making 'large moves" when our algorithm becomes trapped in a bad local minimum. In such instances, we initiate a non-local change on a macroscopic scale. The key strategy involves launching our algorithm, and as soon as it struggles to converge, we implement a process in which a portion of the achieved solution is deliberately destroyed. We then introduce a random factor to replace the destroyed part of the solution and pose two critical questions to the algorithm: firstly, which part of the solution is best to destroy, and secondly, whether the move with the new factor should be accepted. This approach introduces controlled randomness into the optimization process and forms a fundamental part of our proposed algorithm. Categorically, our solution can be classified as a "Ruin and Recreate" (R&R) algorithm [26], as it employs deliberate perturbations and the replacement of solution components with random counterparts to effectively navigate the optimization landscape and break free from local minima. The complete algorithm is summarized in Alg. 2.

In this paragraph, we explore the Stochastic alternating least squares (SALS) algorithm, delineating its principles and procedures step by step. SALS shares a common factor update strategy with the standard ALS algorithm, denoted as ALS(.) herein. When using ALS with two arguments, these represent the original tensor undergoing decomposition and the prescribed PTD ranks. In the case of ALS, called with three arguments, the third argument serves as the initialization from which ALS starts its iterations. An essential initial step in SALS involves starting an ALS procedure from a random initialization. Subsequently, we randomly select P factors from the five factors composing the PTD. For each chosen factor, we systematically ruin/remove the part of the solution corresponding to that factor, replacing it with a newly generated random factor akin to performing significant moves within the optimization landscape. This process runs parallel for P factors, with only one factor removed in each case. Following this removal, we subject the P resulting configurations to competition and retain

Algorithm 2 Stochastic ALS for PTD

Require: Input tensor $\boldsymbol{\mathcal{X}}$, ranks R and S, parameter P
Ensure: $\mathbf{A}, \mathbf{D}^k_{\mathbf{A}}, \mathbf{H}, \mathbf{D}^k_{\mathbf{B}}$ and $\mathbf{B}$ (for $1 \leq k \leq K$)

$[\mathbf{A}_0, \mathbf{D}^k_{\mathbf{A}_0}, \mathbf{H}_0, \mathbf{D}^k_{\mathbf{B}_0}, \mathbf{B}_0] = \mathrm{ALS}(\boldsymbol{\mathcal{X}}, [R, S])$
while stopping criteria not met do
  parfor $p = 1, \cdots, P$ do ▷ This is a parallel ''for'' loop
    select $\mathbf{F}_p$ randomly from $\{\mathbf{A}, \mathbf{D}^k_{\mathbf{A}}, \mathbf{H}, \mathbf{D}^k_{\mathbf{B}}, \mathbf{B}\}$
    if $\mathbf{F}_p = \mathbf{A}$ then
      replace $\mathbf{F}_p$ with a random factor
      $[\mathbf{A}_p, \mathbf{D}^k_{\mathbf{A}_p}, \mathbf{H}_p, \mathbf{D}^k_{\mathbf{B}_p}, \mathbf{B}_p] = \mathrm{ALS}\big(\boldsymbol{\mathcal{X}}, [R, S], [\![\mathbf{F}_p, \mathbf{D}^k_{\mathbf{A}_0}, \mathbf{H}_0, \mathbf{D}^k_{\mathbf{B}_0}, \mathbf{B}_0]\!]\big)$
    else if $\mathbf{F}_p = \mathbf{D}^k_{\mathbf{A}}$ then
      replace $\mathbf{F}_p$ with a random factor
      $[\mathbf{A}_p, \mathbf{D}^k_{\mathbf{A}_p}, \mathbf{H}_p, \mathbf{D}^k_{\mathbf{B}_p}, \mathbf{B}_p] = \mathrm{ALS}\big(\boldsymbol{\mathcal{X}}, [R, S], [\![\mathbf{A}_0, \mathbf{F}_p, \mathbf{H}_0, \mathbf{D}^k_{\mathbf{B}_0}, \mathbf{B}_0]\!]\big)$
    else if $\mathbf{F}_p = \mathbf{H}$ then
      replace $\mathbf{F}_p$ with a random factor
      $[\mathbf{A}_p, \mathbf{D}^k_{\mathbf{A}_p}, \mathbf{H}_p, \mathbf{D}^k_{\mathbf{B}_p}, \mathbf{B}_p] = \mathrm{ALS}\big(\boldsymbol{\mathcal{X}}, [R, S], [\![\mathbf{A}_0, \mathbf{D}^k_{\mathbf{A}_0}, \mathbf{F}_p, \mathbf{D}^k_{\mathbf{B}_0}, \mathbf{B}_0]\!]\big)$
    else if $\mathbf{F}_p = \mathbf{D}^k_{\mathbf{B}}$ then
      replace $\mathbf{F}_p$ with a random factor
      $[\mathbf{A}_p, \mathbf{D}^k_{\mathbf{A}_p}, \mathbf{H}_p, \mathbf{D}^k_{\mathbf{B}_p}, \mathbf{B}_p] = \mathrm{ALS}\big(\boldsymbol{\mathcal{X}}, [R, S], [\![\mathbf{A}_0, \mathbf{D}^k_{\mathbf{A}_0}, \mathbf{H}_0, \mathbf{F}_p, \mathbf{B}_0]\!]\big)$
    else if $\mathbf{F}_p = \mathbf{B}$ then
      replace $\mathbf{F}_p$ with a random factor
      $[\mathbf{A}_p, \mathbf{D}^k_{\mathbf{A}_p}, \mathbf{H}_p, \mathbf{D}^k_{\mathbf{B}_p}, \mathbf{B}_p] = \mathrm{ALS}\big(\boldsymbol{\mathcal{X}}, [R, S], [\![\mathbf{A}_0, \mathbf{D}^k_{\mathbf{A}_0}, \mathbf{H}_0, \mathbf{D}^k_{\mathbf{B}_0}, \mathbf{F}_p]\!]\big)$
    end if
  end parfor
  $[\mathbf{A}_0, \mathbf{D}^k_{\mathbf{A}_0}, \mathbf{H}_0, \mathbf{D}^k_{\mathbf{B}_0}, \mathbf{B}_0] = \underset{\{\mathbf{A}_p, \mathbf{D}^k_{\mathbf{A}_p}, \mathbf{H}_p, \mathbf{D}^k_{\mathbf{B}_p}, \mathbf{B}_p\}_{p=0}^{P}}{\arg\min} ||\boldsymbol{\mathcal{X}} - [\![\mathbf{A}_p, \mathbf{D}^k_{\mathbf{A}_p}, \mathbf{H}_p, \mathbf{D}^k_{\mathbf{B}_p}, \mathbf{B}_p]\!]||^2_F$
  ▷ p starts from 0 and not 1
end while
return $\mathbf{A}_0, \mathbf{D}^k_{\mathbf{A}_0}, \mathbf{H}_0, \mathbf{D}^k_{\mathbf{B}_0}$ and $\mathbf{B}_0$

the configuration that yields the lowest error. In cases where none of the configurations outperforms the previous iteration's error, we maintain the solution from the last iteration, signifying the start of a new round of R&R. The aforementioned detailed steps are integrated into Algorithm 2.

3.3. Convergence

In our case and for the ParaTuck decomposition, it should be noted that for the plain vanilla ALS, the solution is fully determined by the initial conditions (the starting factors for the algorithm) and the parameter values (e.g., the maximum number of iterations). However, the solution will also result from inherent randomness in the proposed stochastic ALS. This means that the same set of parameters and initializations will lead to different results. This makes the theoretical analysis of the stochastic algorithm inherently more complex than deterministic algorithm analysis, as is the case for almost all stochastic algorithms. We can cite the works in [31, 32], where the authors have demonstrated their algorithms' efficiency through numerical experiments. However, we can base ourselves on the ALS process, how our randomness is introduced, and the algorithm's design to state some guarantees for the convergence of the SALS.

In analyzing the convergence properties of Algorithm 2, it is essential to recognize that the update steps mirror those of Algorithm 1. Specifically, we address the optimization problem by iteratively fixing all factors but one and solving a linear least squares problem. This iterative process is replicated across all factors and for multiple iterations. Substituting the multilinear least squares problem with a linear counterpart inherently assures a non-increasing trend [33] in the objective function delineated in equation (5). Before the randomized step, this property ensures that the algorithm exhibits convergence characteristics akin to traditional ALS, where there is no guarantee of reaching a global minimum. Yet, the objective function is bound to decrease or, in the worst case, remain constant with successive iterations. The randomization step is essential to the algorithm's stochastic nature, particularly highlighted in step 22 of Algorithm 2. Here, the replacement of random factors is executed under the stringent condition that any potential update to the solution must not elevate the objective function. In practice, this implies that the updated solution is, at the very least, on par with its predecessor, ensuring that the objective function is subject to either a reduction or stabilization at its current value.

Considering these elements collectively, we postulate that Algorithm 2 is predisposed to a convergence trajectory that leads to a solution where the objective function plateaus, ceasing to decrease further. This tentative conclusion suggests that the stochastic algorithm possesses an intrinsic mechanism that favors convergence, albeit without a formal guarantee of finding the global optimum. However, it is worth noting that the randomness introduced may help in avoiding the entrapment in local minima, a common pitfall of deterministic algorithms.

One should note that even if we can not state guarantees for convergence to a global optimum solution, our results remain important. They show that the SALS will not diverge over the iterations. It should also be noted that this result resembles the convergence guarantee offered by the workhorse ALS algorithm in the CPD case.

3.4. Computational complexity

Regarding the computational complexity inherent to Stochastic ALS compared to Alg. 1, it is imperative to acknowledge that both algorithms substantially mitigate the complexity of the multilinear least squares problem in (5) by converting it into a linear least squares problem. This relaxation is achieved by holding all factors fixed except one, thereby necessitating the invocation of the pseudo-inverse at various stages within the algorithmic flow. As elucidated by Alg. 1, for a single iteration, the computation involves pseudo-inversing matrices of dimensions $(R \times JK)$, $(IJK \times SR)$, $(S \times IK)$, in addition to K-times pseudo-inverse of matrices sized $(IJ \times R)$ and $(S \times IK)$. Considering the computational equivalence of the pseudo-inverse to the Singular Value Decomposition (SVD), and bearing in mind that the complexity of SVD scales as $\mathcal{O}(mn^2)$ [34] for a matrix of size $(m \times n)$ where $m \geq n$, the predominant computational cost of an iteration is thereby $\mathcal{O}\left((SR)^2 IJK\right)$ if we consider that $IJK \geq SR$. If we fix M as the maximum number of iterations for the ALS and N as the maximum number of R&R rounds, we can state that the overall complexity of the SALS is bounded by $\mathcal{O}\left((NP+1)M(SR)^2 IJK\right)$ if the P ALS runs are not executed in parallel, and $\mathcal{O}\left((N+1)M(SR)^2 IJK\right)$ otherwise. However, apart from the randomness introduction, the difference between the methods lies in the methodology for distributing the computational budget, *i.e.*, allocating iterations to each step to achieve a good tradeoff between convergence and computational cost, ensuring efficient use of resources while maintaining accuracy.. This

methodology of complexity budget allocation is further investigated in the next section.

4. Simulations

In this section, we conduct extensive simulations to evaluate the performance of the Stochastic ALS against the conventional ALS method. To ensure a comprehensive comparison, we generate a dataset of $N = 200$ PTDs with randomly generated factors, following a standard normal distribution. Uniquely, the factor $\mathbf{H}$ entries are generated from a uniform distribution ranging from 0.1 to 1.1. This specific range is selected to respect the essential uniqueness conditions for PTD. Ensuring non-zero and varied entries in $\mathbf{H}$ is crucial, as it impacts the decomposition results' structure and interpretability. A critical aspect of our simulation involves ensuring that the generated matrices are well-conditioned. A well-conditioned matrix is defined as one whose conditioning number is below 10. This condition is significant for the uniqueness and stability of the PTD. The condition number of a matrix is defined as the ratio between its maximal and minimal singular values. A well-conditioned matrix has a condition number close to 1, indicating that the matrix is far from being rank-deficient. Conversely, an ill-conditioned matrix has a condition number tending toward infinity, suggesting that the matrix is rank deficient or is close to rank deficiency. Since the rank of a matrix is linked to its nonzero singular values, the condition number is a meaningful measure to determine if a matrix is near or far from rank deficiency. This measure is crucial for ensuring the uniqueness of the PTD, as well-conditioned full-rank factor matrices contribute to the identifiability of the decomposition.

For the plain vanilla ALS, we use 200 different initializations for each ALS run, with the maximum number of iterations set to 2000 for each run. In contrast, the Stochastic ALS algorithm operates under different parameter settings. Initially, we fix $P = 4$ and limit the first ALS run to 500 iterations. Subsequently, we introduce a total of 2000 R&R rounds, with each round allowed a maximum of 50 iterations. This configuration ensures that both algorithms undergo a similar potential maximum number of iterations, approximately $400,000$ iterations for each PTD, enabling a fair and meaningful comparison of their convergence capabilities. One should note that, compared to the ALS in the case of CPD, the chosen numbers for initializations and iterations may seem high. Generally, in a non-noisy case of

the CPD model, a few iterations are sufficient for convergence. However, in our context, this is not the case for the PTD model. We have observed that more iterations and R&R rounds are generally needed to achieve convergence, depending on the initialization. This suggests that the choice of these parameters can cause more computational difficulties in the case of large tensors.

In addition to the maximum number of iterations, we have incorporated two critical convergence criteria into our simulations. The first criterion involves tracking the total normalized mean squared error (NMSE) between the original PTD and its estimation, defined as $\frac{||\boldsymbol{\mathcal{X}}-\hat{\boldsymbol{\mathcal{X}}}||_F^2}{||\boldsymbol{\mathcal{X}}||_F^2}$. If the NMSE falls below the machine precision threshold, we terminate the iterations, signifying convergence in overall reconstruction quality. The second convergence criterion is based on the difference between two successive NMSE values, calculated as $\frac{|\text{NMSE}^{(t)}-\text{NMSE}^{(t-1)}|}{\text{NMSE}^{(t-1)}}$. Suppose this relative change in NMSE between consecutive iterations is smaller than the machine precision. In that case, it indicates that the optimization process has reached a point where further improvement is negligible, prompting the algorithm to stop.

Table 1 illustrates the comparison between the ALS and the proposed stochastic ALS algorithms over 200 random PTD experiments. The Stochastic ALS significantly outperforms ALS in terms of convergence across various tensor dimensions and rank configurations, as evidenced by its higher number of converging experiments in all listed scenarios.

Parameters	ALS [21]	Stochastic ALS
$(I, J, K) = (4, 4, 10)$ $(R, S) = (2, 2)$	58	187
$(I, J, K) = (10, 10, 40)$ $(R, S) = (3, 2)$	49	166
$(I, J, K) = (15, 15, 40)$ $(R, S) = (2, 3)$	67	180
$(I, J, K) = (15, 15, 40)$ $(R, S) = (3, 3)$	53	170

Table 1: Number of converging experiments over 200 random PTDs.

To evaluate the performance of both algorithms in all cases, including non-converging experiments, we plot, in Figs. 1 and 2, the empirical cumulative distribution function (CDF). Fig. 1 shows the CFD of the NMSE. For

each NMSE value, the CDF illustrates the probability that each algorithm would be at or below that given level. The x-axis is on a logarithmic scale, showing the NMSE ranging from the machine's precision to 1. The Stochastic ALS curve is almost a step function, reaching a CDF close to 1 very quickly, indicating that this latter consistently achieves a low NMSE. On the other hand, the ALS curve is more gradual, suggesting more variance in the NMSE achieved by this algorithm. The graph suggests that the Stochastic ALS algorithm outperforms the ALS in achieving a lower NMSE for the 200 PTDs in the same conditions as in Table 1. In Fig. 2, we present an evalua-

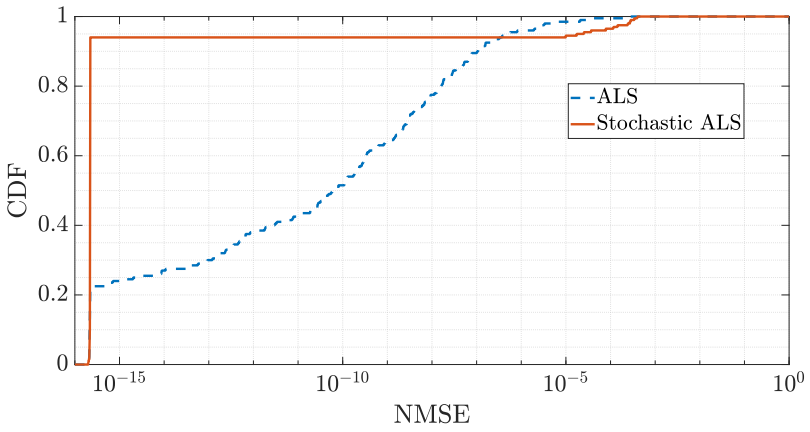

Figure 1: CDF for Stochastic ALS and ALS regarding 200 PTDs, with $(I, J, K) = (4, 4, 10)$, $(R, S) = (2, 2)$ and $P = 5$.

tion of the influence of parameter P on the CDF. It is worth noting that we fixed the same configuration for all experiments, and we only vary P. Notably, the algorithm rapidly reaches 90% in most cases. Interestingly, cases where $P \geq 3$ exhibit better performance, suggesting a potential advantage compared to cases where $P \leq 2$. This result suggests that a comprehensive evaluation of all $P = 5$ configurations of the factors may not be imperative. Instead, our results indicate that in the case of randomly generated PTD tensors, assessing only 3 configurations, for example, is often sufficient to get the same result, which opens avenues for optimizing resource allocation in future studies.

In Fig. 3, we adhere to a fixed computational complexity budget while varying the distribution between the outer loop (Ruin & Recreate rounds) and the inner loop (ALS iterations). Interestingly, none of the configurations

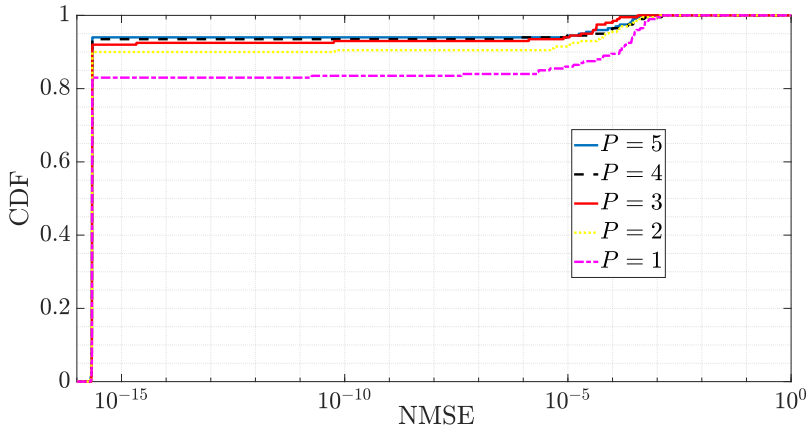

Figure 2: CDFs for different values of P.

significantly outperforms the others. However, the curve corresponding to an equal distribution of iterations closely follows the best-performing configurations, indicating a balanced approach can also be effective within the given computational budget.

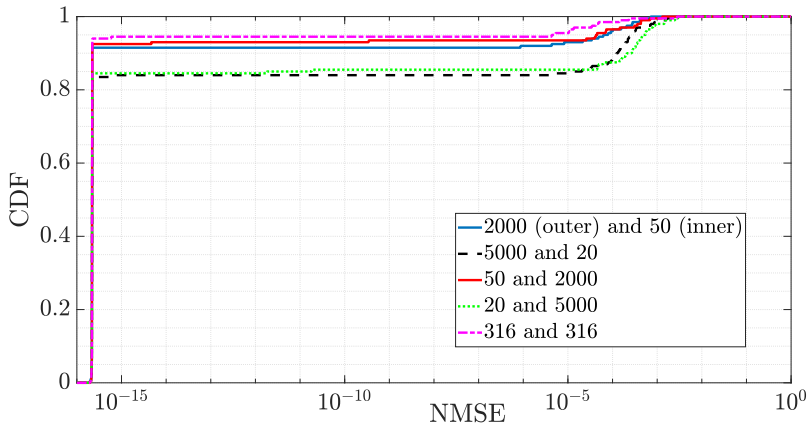

Figure 3: CDFs for different configurations of the iterations.

In Fig. 4, we investigate the efficacy of allowing restarts in the stochastic ALS algorithm for cases where convergence is not initially achieved. For this simulation, we permitted up to two restarts for non-converging experiments, delving into the convergence behavior in a noiseless environment and

attempting to reach 100% convergence. The results depicted in the figure are compelling; in the first algorithm run, convergence was attained in 183 out of 200 PTD experiments. Introducing a first restart led to convergence in an additional 15 cases, and a second restart resolved the remaining 2 cases, culminating in complete convergence across all experiments. This outcome demonstrates the powerful capability of restarts in overcoming local minima or poor initializations that might hinder convergence, ensuring that every PTD reaches a solution when given additional opportunities to escape sub-optimal points in the solution space.

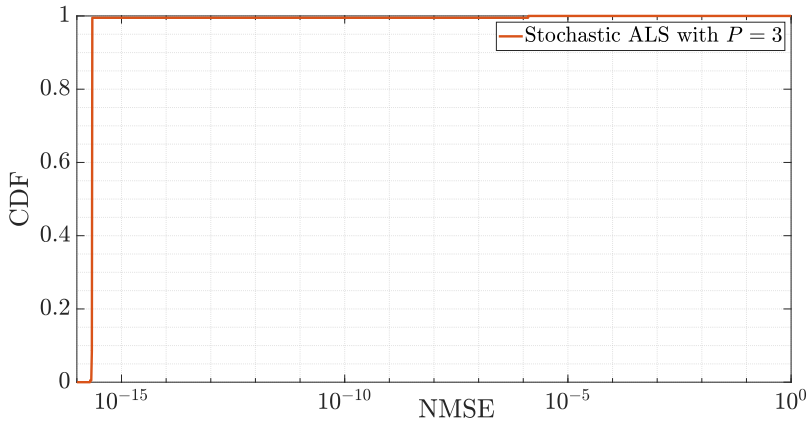

Figure 4: CDFs for Stochastic ALS with restarts.

In Fig. 5, the relationship between the signal-to-noise ratio (SNR) and the NMSE is depicted for the Stochastic ALS compared to the plain vanilla ALS. In this figure, we plot the median value of the NMSE over 200 Monte-Carlo runs to avoid the effect of ill-converging experiments. In this experiment, similar to the first experiments, the plain vanilla ALS algorithm was not run only once, but rather 200 times, each with a different initialization and with a maximum of 1000 iterations each time. The best result over these runs is considered. One should note that in this noisy context, both algorithms exhibit the same robustness. This result is not surprising, given that if we neglect the effect of pathological cases where we do not have convergence, both algorithms minimize the cost function with the same update formulas. The NMSE decreases consistently as the SNR increases, which indicates that the Stochastic ALS is robust to noise, maintaining a high-quality tensor decomposition across a wide range of SNR levels. Particularly noteworthy is

the algorithm's performance at low SNR levels, where despite the high noise presence, the NMSE remains within an acceptable range, showcasing the algorithm's effectiveness in noisy environments. This experiment substantiates the potential applicability of the Stochastic ALS algorithm in real-world scenarios where noise is an inevitable factor.

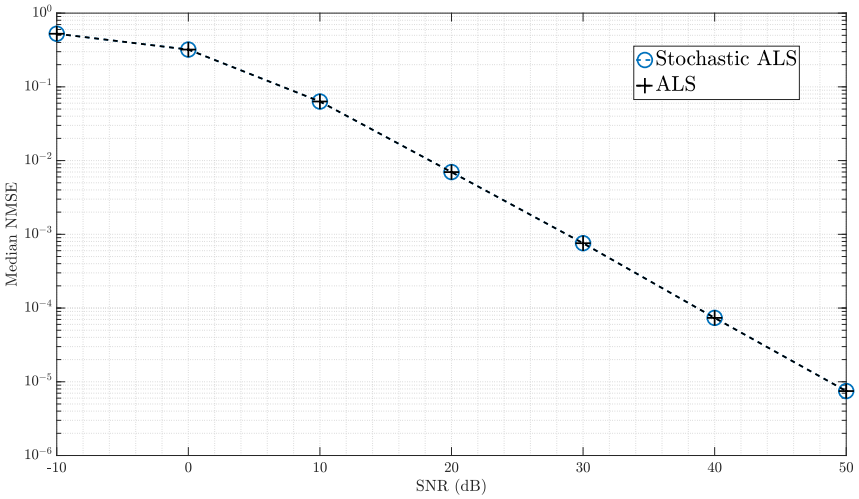

Figure 5: Median NMSE for 200 Monte-Carlo runs, with $(I, J, K) = (4, 4, 10)$, $(R, S) = (2, 2)$ and $P = 3$.

5. Conclusion and perspectives

This paper proposed a novel method to address the performance limitations of ParaTuck decomposition algorithms by implementing a stochastic ALS scheme. The effectiveness of the proposed algorithm was validated through extensive numerical simulations. Our results demonstrate that our scheme consistently reduces, or at the very least maintains, the approximation error at every ALS iteration. Future work includes a detailed study of the algorithm's convergence properties, adaptation to constrained decompositions, and applications to real-world signal processing and communications problems, where the ParaTuck models have been extensively used. The first possible application for which PTD holds promising is parametric channel estimation for intelligent reflecting surface (IRS) assisted MIMO systems [35]. The second application is in blind source separation, where the second-order

statistics can be linked to the PTD in certain scenarios [36]. Finally, we can cite neural network learning with flexible activation functions [13] as another promising application. From a methodological point of view, tackling tensor completion in the context of ParaTuck decomposition is also an interesting perspective for future work.

References

[1] R. A. Harshman, M. E. Lundy, PARAFAC: Parallel factor analysis, Computational Statistics and Data Analysis 18 (1) (1994) 39–72. doi:https://doi.org/10.1016/0167-9473(94)90132-5.

[2] F. L. Hitchcock, Multiple invariants and generalized rank of a p-way matrix or tensor, Journal of Mathematics and Physics 7 (1-4) (1928) 39–79. doi:https://doi.org/10.1002/sapm19287139.

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

[4] I. V. Oseledets, Tensor-Train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317. doi:10.1137/090752286.

[5] R. Bro, PARAFAC. tutorial and applications, Chemometrics and Intelligent Laboratory Systems 38 (2) (1997) 149–171. doi:https://doi.org/10.1016/S0169-7439(97)00032-4.

[6] P. Comon, X. Luciani, A. L. F. De Almeida, Tensor decompositions, alternating least squares and other tales, Journal of Chemometrics: A Journal of the Chemometrics Society 23 (7-8) (2009) 393–405.

[7] L. De Lathauwer, B. De Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM Journal on Matrix Analysis and Applications 21 (4) (2000) 1253–1278. doi:10.1137/S0895479896305696.

[8] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM Review 51 (3) (2009) 455–500. doi:10.1137/07070111X.

[9] Y. Zniyed, Breaking the curse of dimensionality based on tensor train : models and algorithms, Ph.D. thesis (2019).
URL `http://www.theses.fr/2019SACLS330`

[10] Y. Zniyed, R. Boyer, A. L. de Almeida, G. Favier, High-order tensor estimation via trains of coupled third-order CP and Tucker decompositions, Linear Algebra and its Applications 588 (2020) 304–337. doi:https://doi.org/10.1016/j.laa.2019.11.005.

[11] Y. Zniyed, R. Boyer, A. L. F. de Almeida, G. Favier, A TT-based hierarchical framework for decomposing high-order tensors, SIAM Journal on Scientific Computing 42 (2) (2020) A822–A848. doi:10.1137/18M1229973.

[12] G. Favier, A. de Almeida, Overview of constrained PARAFAC models, EURASIP Journal on Advances in Signal Processing 2014 (1) (2014) 142. doi:10.1186/1687-6180-2014-142.

[13] K. Usevich, Y. Zniyed, M. Ishteva, P. Dreesen, A. L. F. de Almeida, Tensor-based two-layer decoupling of multivariate polynomial maps, in: 2023 31st European Signal Processing Conference (EUSIPCO), 2023, pp. 655–659. doi:10.23919/EUSIPCO58844.2023.10289900.

[14] A. L. F. de Almeida, G. Favier, J. P. C. L. da Costa, J. C. M. Mota, Overview of tensor decompositions with applications to communications, in: R. Coelho, V. Nascimento, R. de Queiroz, J. Romano, C. Cavalcante (Eds.), Signals and Images: Advances and Results in Speech, Estimation, Compression, Recognition, Filtering, and Processing, no. Chapter 12, CRC-Press, 2016, pp. 325–356.

[15] Space–time spreading–multiplexing for MIMO wireless communication systems using the PARATUCK-2 tensor model, Signal Processing 89 (11) (2009) 2103–2116.

[16] A. L. F. de Almeida, G. Favier, L. R. Ximenes, Space-time-frequency (STF) MIMO communication systems with blind receiver based on a generalized PARATUCK2 model, IEEE Transactions on Signal Processing 61 (8) (2013) 1895–1909. doi:10.1109/TSP.2013.2238534.

[17] H. Xi, A. L. F. de Almeida, Y. Zhen, Channel estimation for MIMO multi-relay systems using a tensor approach, EURASIP Journal on Advances in Signal Processing 2014 (163) (2014) 1–14.

[18] J. Du, C. Yuan, P. Tian, H. Lin, Channel estimation for multi-input multi-output relay systems using the PARATUCK2

tensor model, IET Communications 10 (9) (2016) 995–1002. doi:https://doi.org/10.1049/iet-com.2015.0907.

[19] P. Marinho R. de Oliveira, C. A. R. Fernandes, G. Favier, R. Boyer, PARATUCK semi-blind receivers for relaying multi-hop MIMO systems, Digital Signal Processing 92 (2019) 127–138. doi:https://doi.org/10.1016/j.dsp.2019.05.011.

[20] R. A. Harshman, M. E. Lundy, Uniqueness proof for a family of models sharing features of Tucker's three-mode factor analysis and parafac/candecomp, Psychometrika 61 (1) (1996) 133–154. doi:10.1007/BF02296963.

[21] R. Bro, Multi-way analysis in the food industry: models, algorithms, and applications, Ph.D. thesis, Universiteit van Amsterdam (1998).

[22] K. Usevich, An algebraic algorithm for rank-2 ParaTuck-2 decomposition, preprint (Feb. 2023).
URL `https://hal.science/hal-03966869`

[23] T. G. Kolda, D. Hong, Stochastic gradients for large-scale tensor decomposition, SIAM Journal on Mathematics of Data Science 2 (4) (2020) 1066–1095. doi:10.1137/19M1266265.

[24] R. Ge, F. Huang, C. Jin, Y. Yuan, Escaping from saddle points — online stochastic gradient for tensor decomposition (2015). arXiv:1503.02101.

[25] L. Souquières, C. Prissette, S. Maire, N. Thirion-Moreau, A parallel strategy for an evolutionary stochastic algorithm: application to the CP decomposition of nonnegative N-th order tensors, in: 2020 28th European Signal Processing Conference (EUSIPCO), pp. 1956–1960. doi:10.23919/Eusipco47968.2020.9287389.

[26] S. K. Johannes Josef Schneider, Stochastic Optimization, 1st Edition, 1434-8322, Springer Berlin, Heidelberg, 2006.

[27] L. Bottou, Stochastic Learning, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 146–168.

[28] J. H. Holland, Genetic algorithms and the optimal allocation of trials, SIAM Journal on Computing 2 (2) (1973) 88–105. doi:10.1137/0202009.

[29] H.-P. Schwefel, Evolutionsstrategien für die numerische Optimierung, Birkhäuser Basel, Basel, 1977, pp. 123–176.

[30] P. M. Ferrante Neri, Carlos Cotta (Ed.), Handbook of Memetic Algorithms, 1st Edition, Springer Berlin, Heidelberg, 2011.

[31] X. T. Vu, S. Maire, C. Chaux, N. Thirion-Moreau, A new stochastic optimization algorithm to decompose large nonnegative tensors, IEEE Signal Processing Letters 22 (10) (2015) 1713–1717.

[32] T. Maehara, K. Hayashi, K.-i. Kawarabayashi, Expected tensor decomposition with stochastic gradient descent, Proceedings of the AAAI Conference on Artificial Intelligence 30 (1) (Feb. 2016). doi:10.1609/aaai.v30i1.10292.

[33] J. D. Carroll, J.-J. Chang, Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young"decomposition, Psychometrika 35 (3) (1970) 283–319. doi:10.1007/BF02310791.

[34] G. H. Golub, C. F. Van Loan, Matrix Computations, 3rd Edition, The Johns Hopkins University Press, 1996.

[35] G. T. de Araújo, A. L. F. de Almeida, R. Boyer, Channel estimation for intelligent reflecting surface assisted MIMO systems: A tensor modeling approach, IEEE Journal of Selected Topics in Signal Processing 15 (3) (2021) 789–802.

[36] D. Lahat, C. Jutten, A new link between joint blind source separation using second order statistics and the Canonical Polyadic decomposition, in: Y. Deville, S. Gannot, R. Mason, M. D. Plumbley, D. Ward (Eds.), Latent Variable Analysis and Signal Separation, Springer International Publishing, Cham, 2018, pp. 171–180.