# Block-Randomized Stochastic Proximal Gradient for Low-Rank Tensor Factorization

Xiao Fu, *Member, IEEE*, Shahana Ibrahim, *Student Member, IEEE*, Hoi-To Wai, *Member, IEEE*, Cheng Gao, and Kejun Huang, *Member, IEEE*



***Abstract*—This work considers the problem of computing the *canonical polyadic decomposition* (CPD) of large tensors. Prior works leverage data sparsity to handle this problem, which is not suitable for handling dense tensors that often arise in applications such as medical imaging, computer vision, and remote sensing. Stochastic optimization is known for its low memory cost and per-iteration complexity when handling dense data. However, existing stochastic CPD algorithms are not flexible to incorporate a variety of constraints/regularization terms that are of interest in signal and data analytics. Convergence properties of many such algorithms are also unclear. In this work, we propose a stochastic optimization framework for large-scale CPD with constraints/regularization terms. The framework works under a doubly randomized fashion, and can be regarded as a judicious combination of *randomized block coordinate descent* (BCD) and *stochastic proximal gradient* (SPG). The algorithm enjoys lightweight updates and small memory footprint. This framework entails considerable flexibility—many frequently used regularizers and constraints can be readily handled. The approach is supported by convergence analysis. Numerical results on large-scale dense tensors are presented to showcase the effectiveness of the proposed approach.**

***Index Terms*— Large-scale tensor decomposition, canonical polyadic decomposition, stochastic gradient, Adagrad**

## I. Introduction

*Canonical polyadic decomposition* (CPD) [previously known as parallel factor analysis (PARAFAC)] [1]–[3] is arguably the most popular low-rank tensor decomposition model. CPD has found applications in various fields, such as analytical chemistry [4], social network mining [5], hyperspectral imaging [6], topic modeling [7], and time series analysis [8]; also see [9]–[11] for more applications in communications.

Computing the CPD of a tensor, however, is a challenging optimization problem [12]. Many algorithms have been proposed through the years [3], [13]–[15]. To keep pace with the ever growing volume of available data, one pressing challenge is to compute CPD at scale. The classic alternating least squares (ALS) algorithm [3] has an elegant algorithmic structure, but suffers from a number of numerical issues [16], [17] and is hardly scalable. In recent years, many new CPD algorithms have appeared, triggered by the advances in big data analytics and first-order optimization [13], [14], [18]–[20]. Many of these algorithms leverage data sparsity to scale up CPD—by cleverly using the zero elements in huge tensors, computationally costly key operations in ALS (e.g., the *matricized tensor times Khatri-Rao product* (MTTKRP) operation) can be significantly simplified. Consequently, the classic ALS algorithm can be modified to handle CPD of huge and sparse tensors.

However, when the tensor to be factored is *dense*—i.e., when most entries of the tensor are nonzero—the sparsity-enabled efficient algorithms [6], [13], [14], [18], [19] are no longer applicable. Note that large and dense tensors arise in many timely and important applications such as medical imaging [21], hyperspectral imaging [6], and computer vision [22]. In fact, since big dense tensors typically cost a lot of memory (e.g., a dense tensor with a size of $2,000 \times 2,000 \times 2,000$ occupies 57.52GB memory if saved as double-precision numbers), it is even hard to load them into the RAM of laptops, desktops, or servers.

Stochastic gradient (SG) method is a powerful tool for handling optimization problems involving a large amount of data, which is known for its low per-iteration memory and computational complexities [23]. A number of stochastic optimization based CPD algorithms have been proposed in the literature [24]–[26]. Specifically, The works in [24], [25] work in an iterative manner. In each iteration, the algorithm samples a random subset of the tensor entries and update the corresponding parts of the latent factors using the sampled data. The algorithms have proven quite effective in practice, and feature distributed implementation [25]. The challenge here is that every tensor entry only contains information of a certain *row* of the latent factors, and updating the entire latent factors may require a high complexity. This may lead to slow improvement of the latent factor estimation accuracy. More importantly, this update strategy loses the opportunity to incorporate constraints/regularization terms on the whole latent factors, since the sampled entries only contain partial information of them. This is undesired in practice, since prior information on the latent factors are critical for enhancing performance, especially in noisy cases.

This work is supported in part by National Science Foundation under Projects NSF ECCS-1608961, ECCS-1808159, III-1910118, the Army Research Office (ARO) under Proejct ARO W911NF-19-1-0247, and by the Chinese University of Hong Kong under the CUHK Direct Grant 4055113.

X. Fu and S. Ibrahim are with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, United States. email (xiao.fu,ibrahish)@oregonstate.edu

H.-T. Wai is with the Department of Systems Engineering and Engineering Management, Shatin, Hong Kong. email: htwai@se.cuhk.edu.hk

C. Gao was with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, United States. He is now with the University of Missouri - Columbia, Columbia, MO 65211. email: gaoche@oregonstate.edu

K. Huang is with the Department of Computer and Information Science and Engineering, Gainesville, FL 32611, University of Florida. email: kejun.huang@ufl.edu

Recently, a stochastic algorithm that ensures updating one entire latent factor in every iteration was proposed in [26]. Instead of sampling tensor entries, the algorithm works via sampling *tensor fibers* that contain information of the whole latent factors. This algorithm exhibits very good empirical performance when the tensor rank is low. However, this algorithm works with at least as many fibers as the tensor rank, which in some cases gives rise to high per-iteration complexity. In addition, the algorithm in [26] did not explicitly offer implementations that take into considerations of constraints or regularization terms—although this can be fixed with some modifications. Lastly, convergence properties of many stochastic CPD algorithms such as those in [24], [26] are unclear.

**Contributions** In this work, we propose a new stochastic algorithmic framework for computing the CPD of large-scale dense tensors. Our contributions include:

• **A Doubly Randomized Computational Framework for Large-Scale CPD**. We propose an efficient and flexible computational framework for CPD of large dense tensors. Our method is a judicious combination of randomized block coordinate descent (BCD) [27], [28] and stochastic proximal gradient (SPG) [29], [30]. In each iteration, our algorithm first samples a mode from all modes of the tensor. Then, the algorithm samples some fibers of this mode and updates the corresponding latent factor via stochastic proximal operations. Such a combination exhibits an array of attractive features: It admits smaller per-iteration memory and computational complexities relative to the existing fiber sampling based method in [26], particularly in high-rank cases. More importantly, it is very flexible in terms of incorporating regularization terms and constraints on the latent factors.

• **Convergence Analysis**. Both BCD and SPG are well studied topics in the optimization literature [27], [28], [31]. However, convergence properties of the proposed framework is not immediately clear due to the nonconvex nature of CPD. The existing block-randomized SGD (BR-SGD) framework [32] only considers convex optimization. Another work [33] considers nonconvex optimization, which adopts a Gauss-Seidel type block updating strategy without randomization. The conditions for convergence in [33] are not easy to check or guarantee in the context of tensor factorization. In contrast, we offer tailored convergence analyses of the proposed algorithm leveraging block randomization, and show that the proposed optimization strategy features sub-sequence convergence to a stationary point—which is a *necessary condition* for attaining local or global optima.

• **Implementation-friendly Adaptive Stepsize Scheduling**. One of the most challenging aspects in stochastic optimization is selecting a proper stepsize schedule. To make the proposed algorithms friendly to use by practitioners, we propose a practical and adaptive stepsize schedule based on the celebrated `Adagrad` algorithm [34]. `Adagrad` is an adaptive stepsize selection method devised for single-block gradient descent. Nonetheless, we find through extensive simulations that it largely helps reducing the agonizing pain of tuning stepsize when implementing our multi-block algorithm for CPD. In addition, we show that the adaptive stepsize-based algorithm converges to a stationary point almost surely.

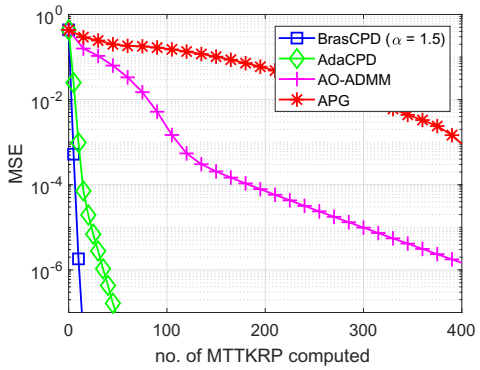



Fig. 1. The proposed algorithms (`AdaCPD` and `BrasCPD`) exhibit low complexity for achieving high accuracy of the estimated latent factors. The tensor under test has a size of $100 \times 100 \times 100$ and the rank is 10. The latent factors are constrained to be nonnegative. The baselines are two state-of-art constrained CPD algorithms `AO-ADMM` [13] and `APG` [14].

A quick demonstration of the effectiveness of the proposed algorithms is shown in Fig. 1, where the average mean squared error (MSE) of the estimated latent factors [cf. Eq. (22)] against the number of MTTKRP computed (which dominates the complexity) is plotted. One can see that the proposed algorithm largely outperforms a couple of state-of–the-art algorithms for constrained CPD. More thorough numerical results can be seen in Sec. VI.

Part of the work was submitted to ICASSP 2019 [35]. In this new version, we have included detailed convergence proofs and the new adaptive stepsize based algorithm. More extensive simulations and real-data experiments are also included.

**Notation.** We follow the established conventions in signal processing. $x$, $\boldsymbol{x}$, $\boldsymbol{X}$, and $\underline{\boldsymbol{X}}$ denote scalar, vector, matrix, and tensor, respectively; $\|\cdot\|$ denotes the Euclidean norm, i.e., $\|\boldsymbol{x}\|_2$ and $\|\boldsymbol{X}\|_F$, respectively; $\circ$, $\odot$, and $\circledast$ denote outer product, Khatri-Rao product, and Hadamard product, respectively, unless otherwise specified; $\mathrm{vec}(\boldsymbol{X})$ denotes the vectorization operator that concatenates the columns of $\boldsymbol{X}$; $\boldsymbol{X} \geq \boldsymbol{0}$ means that all the entries of $\boldsymbol{X}$ are nonnegative; $^\top$ and $^\dagger$ denote transpose and pseudo-inverse, respectively; $|\mathcal{C}|$ denotes the cardinality of set $\mathcal{C}$; $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of a matrix. Unless otherwise specified, we denote the total expectation by the subscript-less operator $\mathbb{E}[\cdot]$.

## II. Background

We first introduce some notions used in tensor algebra.

### A. Tensors and CPD

An $N$th order tensor is an array whose entries are indexed by $N$ coordinates; i.e., $\underline{\boldsymbol{X}}(i_1, \ldots, i_N)$ denotes an element of the tensor $\underline{\boldsymbol{X}}$ with a size of $I_1 \times I_2 \times \ldots \times I_N$. Like matrices, tensors can be represented as sum of rank-one components:

$$\underline{\boldsymbol{X}} = \sum_{f=1}^{F} \boldsymbol{A}_{(1)}(:, f) \circ \boldsymbol{A}_{(2)}(:, f) \circ \ldots \circ \boldsymbol{A}_{(N)}(:, f), \qquad (1)$$

where "$\circ$" denotes the outer product of vectors, and $\boldsymbol{A}_{(n)}$ is an $I_n \times F$ matrix that is often referred to as the *mode-$n$ latent factor*. When $F$ is the minimal integer that satisfies

the expression in (1), the right hand side in (1) is called the *canonical* polyadic decomposition of the tensor $\underline{\boldsymbol{X}}$. At the entry level, the CPD can be expressed as

$$\underline{\boldsymbol{X}}(i_1,\ldots,i_N)=\sum_{f=1}^{F}\prod_{n=1}^{N}\boldsymbol{A}_{(n)}(i_n,f), \tag{2}$$

for $i_n\in\{1,\ldots,I_n\}$. The CPD of a tensor is essentially unique under mild conditions[1]. The CPD of a tensor can be obtained via minimizing a certain criterion as follows:

$$\underset{\{\boldsymbol{A}_{(n)}\}_{n=1}^{N}}{\text{minimize}}\ f(\boldsymbol{A}_{(1)},\ldots,\boldsymbol{A}_{(N)}). \tag{3}$$

A common optimization criterion for CPD in the literature is the *least squares* (LS) fitting criterion [3], [13], [14]:

$$f(\boldsymbol{A}_{(1)},\ldots,\boldsymbol{A}_{(N)})=\Big\|\underline{\boldsymbol{X}}-\sum_{f=1}^{F}\boldsymbol{A}_{(1)}(:,f)\circ\ldots\circ\boldsymbol{A}_{(N)}(:,f)\Big\|_F^2.$$

In the sequel, we will often use the shorthand notation $f(\boldsymbol{\theta})$ to denote $f(\boldsymbol{A}_{(1)},\ldots,\boldsymbol{A}_{(N)})$, where

$$\boldsymbol{\theta}=[\mathrm{vec}(\boldsymbol{A}_{(1)})^\top,\ldots,\mathrm{vec}(\boldsymbol{A}_{(N)}^\top)]^\top.$$

Other criteria have also been considered, e.g., the Kullback-Leibler (KL) divergence [36] and robust fitting [37], [38] criteria—which serve for different purposes.

### *B. Unfolding, ALS and MTTKRP*

The matricization operation, or *matrix unfolding* of a tensor, has proven very useful in designing tensor factorization algorithms. The mode-$n$ unfolding of a tensor is a $J_n\times I_n$ matrix where

$$\underline{\boldsymbol{X}}(i_1,\ldots,i_N)=\boldsymbol{X}_{(n)}(j,i_n),$$

and we have $j=1+\sum_{k=1,k\neq n}^{N}(i_k-1)J_k$ and $J_k=\prod_{m=1,m\neq n}^{k-1}I_m$ [1]. The CPD representation in Eq. (1) can be expressed as

$$\boldsymbol{X}_{(n)}=\boldsymbol{H}_{(n)}\boldsymbol{A}_{(n)}^\top, \tag{4}$$

where the $J_n\times F$ matrix $\boldsymbol{H}_{(n)}$ is defined as

$$\boldsymbol{H}_{(n)}=\boldsymbol{A}_{(1)}\odot\boldsymbol{A}_{(n-1)}\odot\boldsymbol{A}_{(n+1)}\odot\ldots\odot\boldsymbol{A}_{(N)}=\odot_{i=1,i\neq n}^{N}\boldsymbol{A}_{(i)}.$$

The elegant form of the unfoldings has enabled the famous *alternating least squares* (ALS) algorithm [3] for handling Problem (3) with the LS objective. Specifically, ALS solves the following cyclically for $n=1,\ldots,N$:

$$\boldsymbol{A}_{(n)}\leftarrow\arg\min_{\boldsymbol{A}}\ \big\|\boldsymbol{X}_{(n)}-\boldsymbol{H}_{(n)}\boldsymbol{A}^\top\big\|_F^2. \tag{5}$$

Problem (5) is nothing but a least squares problem that admits the following closed-form solution:

$$\boldsymbol{A}_{(n)}\leftarrow\Big((\boldsymbol{H}_{(n)}^\top\boldsymbol{H}_{(n)})^{-1}\boldsymbol{H}_{(n)}^\top\boldsymbol{X}_{(n)}\Big)^\top,$$

if $\mathrm{rank}(\boldsymbol{H}_{(n)})=F$. Note that $(\boldsymbol{H}_{(n)}^\top\boldsymbol{H}_{(n)})^{-1}$ is not difficult to compute by exploiting the Khatri-Rao structure of $\boldsymbol{H}_{(n)}$ [1], [2], [13]. However, when the problem dimension is large (which often happens in applications such as medical imaging, remote sensing, and computer vision), solving the seemingly simple problem in (5) can be computationally prohibitive. The reason is that both $\boldsymbol{X}_{(n)}\in\mathbb{R}^{(\prod_{j=1,j\neq n}^{N}I_j)\times I_n}$ and $\boldsymbol{H}_{(n)}\in\mathbb{R}^{(\prod_{j=1,j\neq n}^{N}I_j)\times F}$ can be very large matrices. In particular, the so-called *matricized tensor times Khatri-Rao product* (MTTKRP) operation, i.e.,

$$\mathsf{MTTKRP}:\qquad \boldsymbol{H}_{(n)}^\top\boldsymbol{X}_{(n)}$$

that happens in every iteration of ALS costs $\mathcal{O}(\prod_{n=1}^{N}I_nF)$ flops (or, $\mathcal{O}(I^NF)$ if $I_n=I$). This is quite costly even if $I_n$ is moderately large. Many works have considered fast algorithms for computing MTTKRP, but these methods are mainly enabled by judiciously exploiting sparsity of the tensor data [18], [39]. Computing MTTKRP for dense tensors has also been considered. Nonetheless, these works are often concerned with practical implementation schemes such as parallelization and memory-efficient computation strategies, but the number of computational flops required is naturally high for the dense tensor case; see, e.g., [40], [41].

In a lot of applications, some prior knowledge on the latent factors is known—e.g., in image processing, $\boldsymbol{A}_{(n)}$'s are normally assumed to be nonnegative [6]; in statistical machine learning, sometimes the columns of $\boldsymbol{A}_{(n)}$ are assumed to be constrained within the probability simplex [36], [42]; i.e.,

$$\mathbf{1}^\top\boldsymbol{A}_{(n)}=\mathbf{1}^\top,\ \boldsymbol{A}_{(n)}\geq\mathbf{0}. \tag{6}$$

In those cases, the following criterion is often of interest:

$$\begin{aligned}&\underset{\{\boldsymbol{A}_{(n)}\}_{n=1}^{N}}{\text{minimize}}\ f(\boldsymbol{A}_{(1)},\ldots,\boldsymbol{A}_{(N)})\\&\text{subject to }\boldsymbol{A}_{(n)}\in\mathcal{A}_n.\end{aligned} \tag{7}$$

Compared to the unconstrained version, Problem (7) is even harder to handle. Some recent methods combine first-order constrained optimization and ALS [13], [14] to make the tensor factorization algorithms more flexible in handling constraints and regularization terms—but the complexity orders of those algorithms often scale similarly as that of ALS, since these algorithms do not avoid computing $\boldsymbol{H}_{(n)}^\top\boldsymbol{X}_{(n)}$ that is the bottleneck for computing CPD.

### *C. Stochastic Optimization*

When the tensor is large and dense, working with the entire dataset could be computationally and memory-wise expensive. A popular workaround is to apply *stochastic optimization*—i.e., sampling parts of the data at random and use the sampled piece to update the latent factors. Using Eq. (2), Problem (3) with the LS objective is equivalent to the following:

$$\underset{\{\boldsymbol{A}_{(n)}\}}{\text{minimize}}\ (1/T)\sum_{i_1,\ldots,i_N}f_{i_1,\ldots,i_N}(\boldsymbol{\theta}), \tag{8}$$

where $T=\prod_{n=1}^{N}I_n$ and

$$f_{i_1,\ldots,i_N}(\boldsymbol{\theta})=\left(\underline{\boldsymbol{X}}(i_1,\ldots,i_N)-\sum_{f=1}^{F}\prod_{n=1}^{N}\boldsymbol{A}_{(n)}(i_n,f)\right)^2.$$

The objective function in (8) can be understood as an empirical risk [23]. Using this observation, the algorithms in [24], [25]

[1]The latent factors $\boldsymbol{A}_{(n)}$'s that constitute the data $\underline{\boldsymbol{X}}$ are unique up to some trivial ambiguities like column permutations and scalings [2].

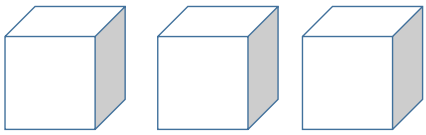


Fig. 2. From left to right: mode-$1, 2, 3$ tensor fibers of a third-order tensor, respectively.

randomly sample a subset of entries indexed by $\{(i_1, \ldots, i_N)\}$ and update the pertinent parts of the latent factors (note that the $(i_1, \ldots, i_N)$th entry of tensor contains the information of $\boldsymbol{A}_{(n)}(i_n, :)$ for $n = 1, \ldots, N$) using the sampled entries of the tensor. For example, [25] uses a stochastic gradient (SG) based approach and update the $\boldsymbol{A}_{(n)}(i_n, :)$'s that are associated with the sampled entries. The sampling method in [24] is similar, while the update is not gradient-based but Gauss-Newton or ALS applied to the sampled set of entries (or, sub-tensors, to be precise). The upshot of this line of work is that the per-iteration complexity can be quite low.

Despite of such favorable complexity savings, the approaches in [24], [25] have a couple of limitations. One challenge is that many useful prior information cannot be incorporated in the algorithm. The reason is that these algorithms update part of the *rows* of $\boldsymbol{A}_{(n)}$'s, while many useful priors are defined w.r.t. the *columns* of the latent factors, e.g., the probability simplex constraint in (6) and the total variation constraint that is heavily used in image processing. For example, the algorithm in [24] samples subtensors $\underline{\boldsymbol{X}}_{\mathsf{sub}} = \underline{\boldsymbol{X}}(\mathcal{S}_1, \ldots, \mathcal{S}_N)$ (where $\mathcal{S}_n \subset \{1, \ldots, I_n\}$) to update the corresponding $\boldsymbol{A}_{(n)}(\mathcal{S}_n, :)$'s. Under such a scheme, it is hard to handle constraints like $\mathbf{1}^\top \boldsymbol{A}_{(n)} = \mathbf{1}^\top, \boldsymbol{A}_{(n)} \geq \mathbf{0}$ that are critical in statistical learning [42]–[45]. Third, convergence properties of these methods are often unclear.

An alternative [26] is to leverage the tensor data structure by considering randomly sampled *fibers* of tensors. Note that a mode-$n$ fiber of $\underline{\boldsymbol{X}}$ (cf. Fig. 2) is a row of the mode-$n$ unfolding $\boldsymbol{X}_{(n)}$ [1]. Now, assuming that one samples a set of mode-$n$ fibers indexed by $\mathcal{F}_n \subset \{1, ..., J_n\}$, then $\boldsymbol{A}_{(n)}$ can be updated by solving a 'sketched version' of Problem (5):

$$\boldsymbol{A}_{(n)} \leftarrow \arg\min_{\boldsymbol{A}} \ \left\| \boldsymbol{X}_{(n)}(\mathcal{F}_n, :) - \boldsymbol{H}_{(n)}(\mathcal{F}_n, :)\boldsymbol{A}^\top \right\|_F^2, \quad (9)$$

If $|\mathcal{F}_n| \geq F$, then the sketched system of linear equations $\boldsymbol{X}_{(n)}(\mathcal{F}_n, :) \approx \boldsymbol{H}_{(n)}(\mathcal{F}_n, :)\boldsymbol{A}_{(n)}^\top$ can be over-determined. Hence, one can update $\boldsymbol{A}_{(n)}$ by solving the $|\mathcal{F}_n|$ dimensional linear system

$$\boldsymbol{A}_{(n)}^\top \leftarrow \boldsymbol{H}_{(n)}(\mathcal{F}_n, :)^\dagger \boldsymbol{X}_{(n)}(\mathcal{F}_n, :). \quad (10)$$

Similar to the ALS algorithm, after updating $\boldsymbol{A}_{(n)}$, the algorithm moves to mode-$(n+1)$ fibers and repeats the same for updating $\boldsymbol{A}_{(n+1)}$. The downside of this method is that it needs to sample at least $F$ fibers for each update, and $F$ can be larger than $I_n$ in tensor decomposition. In addition, the work in [26] focused on unconstrained cases while did not offer implementations for constrained/regularized cases. This can be compensated by replacing (10) with a constrained least squares solver or certain constraint enforcing operations. However, convergence properties of doing so are also unclear.

## III. Proposed Algorithm

In this work, we propose a new stochastic optimization strategy for CPD. Our method combines the insights from ALS and fiber sampling, but allows $|\mathcal{F}_n| \ll F$. This is instrumental in practice, since it is the key for achieving low per-iteration complexity. The proposed algorithm can easily handle a variety of constraints and regularizations that are commonly used in signal processing and data analytics—which is reminiscent of *stochastic proximal gradient* (SPG) [30], [46]. In addition, we provide convergence analyses to back up the proposed approach.

### A. Basic Idea: Unconstrained Case

We first consider Problem (3). Our idea is to apply SA while exploiting the tensor fiber structure. Specifically, at each iteration, we sample a set of mode-$n$ fibers for a certain $n$ as the method in [26] does. However, instead of exactly solving the least squares subproblems (5) for all the modes following a Gauss-Seidel manner in each iteration, we update $\boldsymbol{A}_{(n)}$ using a doubly stochastic procedure. To be more precise, at iteration $r$, we first randomly sample a mode index $n \in \{1, ..., N\}$. Then, we randomly sample a set of mode-$n$ fibers that is indexed by $\mathcal{F}_n \subset \{1, ..., J_n\}$. Let $\boldsymbol{G}^{(r)} \in \mathbb{R}^{(I_1 + \ldots + I_N) \times F}$ such that

$$\boldsymbol{G}^{(r)} = [(\boldsymbol{G}_{(1)}^{(r)})^\top, \ldots, (\boldsymbol{G}_{(N)}^{(r)})^\top]^\top,$$

where we have

$$\boldsymbol{G}_{(n)}^{(r)} = \frac{1}{|\mathcal{F}_n|}\Big(\boldsymbol{A}_{(n)}^{(r)} \boldsymbol{H}_{(n)}^\top(\mathcal{F}_n)\boldsymbol{H}_{(n)}(\mathcal{F}_n) - \boldsymbol{X}_{(n)}^\top(\mathcal{F}_n)\boldsymbol{H}_{(n)}(\mathcal{F}_n)\Big)$$
$$\boldsymbol{G}_{(n')}^{(r)} = \mathbf{0}, \quad n' \neq n, \quad (11)$$

and we used the shorthand notations

$$\boldsymbol{X}_{(n)}(\mathcal{F}_n) = \boldsymbol{X}_{(n)}(\mathcal{F}_n, :), \quad \boldsymbol{H}_{(n)}(\mathcal{F}_n) = \boldsymbol{H}_{(n)}(\mathcal{F}_n, :).$$

The latent variables are updated by

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)} \boldsymbol{G}_{(n)}^{(r)}, \quad n = 1, ..., N. \quad (12)$$

Observe that $\boldsymbol{G}_{(n)}^{(r)}$ is an estimate of the gradient applied to $f(\boldsymbol{A}_{(1)}, \ldots, \boldsymbol{A}_{(N)})$ taken w.r.t. the mode-$n$ variable $\boldsymbol{A}_{(n)}$, and the update is an iteration of the SG algorithm with a minibatch size $|\mathcal{F}_n|$ for solving the problem in (5).

The proposed update is very efficient, since the most resource-consuming update $\boldsymbol{H}_{(n)}^\top \boldsymbol{X}_{(n)}$ in algorithms such as those in [13], [14] is avoided. The corresponding part $\boldsymbol{X}_{(n)}^\top(\mathcal{F}_n, :)\boldsymbol{H}_{(n)}(\mathcal{F}_n, :)$ costs only $\mathcal{O}(|\mathcal{F}_n| F I_n)$ flops—and $|\mathcal{F}_n|$ is under our control. Note that the first step in this procedure is different from standard ALS-type algorithms that update the block variables $\boldsymbol{A}_{(n)}$ cyclically instead of updating a randomly sampled block. As we will show, this modification greatly simplifies our convergence analysis.

### B. Constrained and Regularized Case

As mentioned, there are many cases in practice where considering regularizations or constraints on $\boldsymbol{A}_{(n)}$'s can benefit the associated tasks. Since our framework updates an entire $\boldsymbol{A}_{(n)}$ in each iteration, it is friendly for incorporating a

**Algorithm 1:** `BrasCPD`

**input :** $N$-way tensor $\underline{\boldsymbol{X}} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$; rank $F$; sample size $B$, initialization $\{\boldsymbol{A}_{(n)}^{(0)}\}$, step size $\{\alpha^{(r)}\}_{r=0,\ldots}$

$r \leftarrow 0$;
**repeat**
  uniformly sample $n$ from $\{1,\ldots,N\}$, then sample $\mathcal{F}_n$ uniformly from $\{1,\ldots,J_n\}$ with $|\mathcal{F}_n| = B$;
  form the stochastic gradient $\boldsymbol{G}^{(r)} \leftarrow$ (11);
  update $\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow$ (14a), $\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}$ for $n' \neq n$;
  $r \leftarrow r+1$;
**until** *some stopping criterion is reached*;
**output:** $\{\boldsymbol{A}_{(n)}^{(r)}\}_{n=1}^{N}$

large variety of commonly used constraints/regularizations—which is more flexible relative to the entry sampling based approaches in [24], [25]. The algorithm can be easily extended to handle the constrained/regularized case. Consider:

$$\underset{\{\boldsymbol{A}_{(n)}\}_{n=1}^{N}}{\text{minimize}} \; f(\boldsymbol{\theta}) + \sum_{n=1}^{N} h_n(\boldsymbol{A}_{(n)}) \tag{13}$$

where $h_n(\boldsymbol{A}_{(n)})$ denotes a structure-promoting regularizer on $\boldsymbol{A}_{(n)}$. Note that if $\boldsymbol{A}_{(n)} \in \mathcal{A}_n$ is desired, we can write $h_n(\cdot)$ is defined as the indicator function of set $\mathcal{A}_n$:

$$h_n(\boldsymbol{A}) = \mathcal{I}(\mathcal{A}_n) = \begin{cases} 0, & \boldsymbol{A} \in \mathcal{A}_n \\ \infty, & \text{otherwise.} \end{cases}$$

Using the same fiber sampling strategy as in the previous subsection, we update $\boldsymbol{A}_{(n)}$ by

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \arg\min_{\boldsymbol{A}_{(n)}} \; \left\|\boldsymbol{A}_{(n)} - \left(\boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)}\boldsymbol{G}_{(n)}^{(r)}\right)\right\|_F^2 + h_n\left(\boldsymbol{A}_{(n)}\right), \tag{14a}$$

$$\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}, \quad n' \neq n. \tag{14b}$$

If $h_n(\cdot)$ is a closed proper convex function, the update (14a) can be solved by applying the proximal operator of $h_n(\cdot)$, which is often denoted as

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \mathsf{Prox}_{h_n}\left(\boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)}\boldsymbol{G}_{(n)}^{(r)}\right). \tag{15}$$

Many $h_n(\cdot)$'s admit simple closed-form solutions for their respective proximal operators, e.g., when $h_n(\cdot)$ is the indicator function of the nonnegative orthant and $h_n(\cdot) = \|\cdot\|_1$; see Table I and more details in [13], [47]. The complexity of computing (15) is similar to that of the plain update in (12), and thus is also computationally efficient. An overview of the proposed algorithm can be found in Algorithm 1, which we name ***B**lock-**Ra**ndomized **S**GD for **CPD*** (`BrasCPD`).

TABLE I
PROXIMAL/PROJECTION OPERATOR OF SOME FREQUENTLY USED REGULARIZATIONS AND CONSTRAINTS.

| $h(\cdot)$ | prox./proj. solution | complexity |
|---|---|---|
| $\|\cdot\|_1$ | soft-thresholding | $\mathcal{O}(d)$ |
| $\|\cdot\|_2$ | re-scale | $\mathcal{O}(d)$ |
| $\|\cdot\|_{2,1}$ | block soft-thresholding | $\mathcal{O}(d)$ |
| $\|\cdot\|_0$ | hard-thresholding | $\mathcal{O}(d)$ |
| $\mathcal{I}(\Delta)$ | randomized pivot search [48] | $\mathcal{O}(d)$ in expectation |
| $\mathcal{I}(\mathbb{R}_+)$ | max | $\mathcal{O}(d)$ |
| monotonic | monotone regression [49] | $\mathcal{O}(d)$ |
| unimodal | unimodal regression [50] | $\mathcal{O}(d^2)$ |

†In the table, $d$ is the number of optimization variables.

## IV. Convergence Properties

In this section, we offer tailored convergence analyses for `BrasCPD`. To this end, two most relevant works from the optimization literature are [14] and [32]. The work in [32] considers block-randomized SGD, but only for the convex case—while our problem is nonconvex. The work in [14] considers the Gauss-Seidel type block SGD (i.e., cyclically updating the blocks), instead of the block-randomized version as `BrasCPD` uses. There, convergence is established using a number of assumptions that are not easy to check or guarantee in the context of CPD, e.g., that the bias of the stochastic oracle is bounded. We will show that, by using the block-randomization strategy and the proposed stochastic oracle construction, such an assumption can be circumvented[2].

To facilitate our discussions, let us define $\xi^{(r)} \in \{1,...,N\}$ and $\boldsymbol{\zeta}^{(r)} \subseteq \{1,...,J_{\xi_{(r)}}\}$ as the random variables (r.v.) responsible for selecting the mode and fibers in iteration $r$, respectively. These r.v.s are distributed as

$$\mathsf{Pr}(\xi^{(r)} = n) = \frac{1}{N}, \; \mathsf{Pr}(\boldsymbol{\zeta}^{(r)} = \mathcal{S} \mid \xi^{(r)} = n) = \frac{1}{\binom{J_n}{B}}, \tag{16}$$

where $n \in \{1,...,N\}$, $\mathcal{S} \in \Sigma$ such that $\Sigma$ is the set of all subsets of $\{1,...,J_{\xi_{(r)}}\}$ with size $B$. We observe

**Fact 1** *Denote $\mathcal{B}^{(r)}$ as the filtration generated by the r.v.s $\{\xi^{(1)}, \boldsymbol{\zeta}^{(1)}, \ldots, \xi^{(r-1)}, \boldsymbol{\zeta}^{(r-1)}\}$ such that the $r$th iterate $\boldsymbol{\theta}^{(r)}$ is determined conditioned on $\mathcal{B}^{(r)}$. The stochastic gradient in* (11) *is an unbiased estimate for the full gradient w.r.t. $\boldsymbol{A}_{(\xi^{(r)})}$*

$$\mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[\boldsymbol{G}_{(\xi^{(r)})}^{(r)} \mid \mathcal{B}^{(r)}, \xi^{(r)}\right] = \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}). \tag{17}$$

The proof of the above is straightforward and thus skipped. Fact 1 says that our block stochastic gradient is an unbiased estimation for the "block gradient" $\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})$. This fact will prove quite handy in establishing convergence. The two-level sampling strategy (i.e., block sampling and fiber sampling, respectively) makes the gradient estimation w.r.t. $\boldsymbol{\theta}$ unbiased up to a scaling factor (see Appendix A). This connection intuitively suggests that the proposed algorithm should behave similarly as an SG algorithm.

### A. Unconstrained Case

We will use the following assumptions:

**Assumption 1** *The stepsize schedule follows the Robbins-Monro rule [51]:*

$$\textstyle\sum_{r=0}^{\infty} \alpha^{(r)} = \infty, \quad \sum_{r=0}^{\infty} (\alpha^{(r)})^2 < \infty.$$

**Assumption 2** *The updates $\boldsymbol{A}_{(n)}^{(r)}$ are bounded for all $n, r$.*

[2]We should note that the major motivation for using block randomization strategy is theoretical guarantees—since our goal is a convergence-guaranteed algorithmic framework; in practice, we observe cyclically updating the latent factors works as well.

Assumption 1 is a principle for stepsize scheduling, which is commonly used in SG algorithms. Assumption 2 is considered a relatively strong assumption. In practice, there are several simple ways to make $\boldsymbol{A}_{(n)}^{(r)}$'s bounded. One pragmatic modification is to change the objective to $f(\boldsymbol{\theta}) + \sum_{n=1}^{N} h_n(\boldsymbol{A}_{(n)} + \sum_{n=1}^{N} \lambda_n \|\boldsymbol{A}_{(n)}\|_F^2$. Another method is as mentioned in [13], [31]. At iteration $r$, one may add a proximal term $\lambda_n \|\boldsymbol{A}_{(n)} - \boldsymbol{A}_{(n)}^{(r)}\|_F^2$ to the cost function, which will effectively prevent $\boldsymbol{A}_{(n)}^{(r+1)}$ from being unbounded. Following both ways, the updates can still be handled by simple proximal operations for the $h$ functions in Table I.

There are an array of problem structures that are useful for studying convergence of the algorithm.

**Fact 2** *For any $\boldsymbol{\theta}, \bar{\boldsymbol{\theta}}$ and mode $n \in \{1, ..., N\}$, there exists a constant $\bar{L}_{(n)}$ such that*

$$f(\boldsymbol{\theta}) \leq f(\bar{\boldsymbol{\theta}}) + \langle \nabla_{\boldsymbol{A}_{(n)}} f(\bar{\boldsymbol{\theta}}), \boldsymbol{A} - \bar{\boldsymbol{A}}_{(n)} \rangle + \frac{\bar{L}_{(n)}}{2} \|\boldsymbol{A} - \bar{\boldsymbol{A}}_{(n)}\|_F^2, \quad (18)$$

*where $\bar{\boldsymbol{A}}_{(n)}$ and $\bar{\boldsymbol{H}}_{(n)}$ are extracted/constructed from $\bar{\boldsymbol{\theta}}$ following the respective definitions.*

Eq. (18) holds because the objective function $f(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{A}_{(n)}$ is a plain least squares fitting criterion, which is known to have a Lipschitz continuous gradient—and the smallest Lipschitz constant is $\lambda_{\max}(\bar{\boldsymbol{H}}_{(n)}^{\top} \bar{\boldsymbol{H}}_{(n)})$.

We have the following convergence property for `BrasCPD` in the unconstrained case:

**Proposition 1** *Consider the case where $h_n(\cdot) = 0$ for all $n$ and Assumptions 1-2 hold. The solution sequence produced by `BrasCPD` satisfies:*

$$\liminf_{r \to \infty} \mathbb{E}\left[ \left\| \nabla f(\boldsymbol{\theta}^{(r)}) \right\|^2 \right] = 0.$$

The proof is relegated to Appendix B. The above proposition implies that there exists a subsequence of the solution sequence that converges to a stationary point in expectation. The use of the expectation notion is due to the randomness in the algorithm. We should mention that the SG update and the block sampling step are essential for establishing convergence—and using the exact solution to (9) as in [26] may not have such convergence properties.

### *B. Constrained/Regularized Case*

When $h_n(\cdot) \neq 0$, the gradient of the objective function of (13) may be undefined. In this case, a solution $\boldsymbol{\theta}^{(r)}$ is stationary if $\boldsymbol{P}_{(n)}^{(r)} = \boldsymbol{0}$, $\forall\ n$, where

$$\boldsymbol{P}_{(n)}^{(r)} = \frac{1}{\alpha^{(r)}} \left( \boldsymbol{A}_{(n)}^{(r+1)} - \mathsf{Prox}_{h_n} \left( \boldsymbol{A}_{(n)}^{(r)} - \alpha^{(r)} \nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)}) \right) \right);$$

i.e., the condition is satisfied in a blockwise fashion [31], [33]. Hence, our goal of this section is to show $\mathbb{E}[\|\boldsymbol{P}_{(n)}^{(r)}\|^2]$ vanishes for all $n$ as $r \to \infty$. Consider the following assumption:

**Assumption 3** *There exists a sequence $\{\sigma^{(r)}\}_{r \geq 0}$ such that*

$$\mathbb{E}_{\boldsymbol{\zeta}^{(r)}} \left[ \left\| \boldsymbol{G}_{(\xi^{(r)})}^{(r)} - \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\|^2 \middle| \mathcal{B}^{(r)}, \xi^{(r)} \right] \leq (\sigma^{(r)})^2,$$

$$\sum_{r=0}^{\infty} (\sigma^{(r)})^2 < \infty, \quad \sum_{r=0}^{\infty} \alpha^{(r)} (\sigma^{(r)})^2 < \infty, \quad (19)$$

*where $\{\alpha^{(r)}\}_{r \geq 0}$ is the stepsize sequence following Assumption 1.*

The `BrasCPD` produces a convergent solution subsequence:

**Proposition 2** *Assume that Assumptions 1-3 hold. Also assume that $h_n(\cdot)$ is a closed proper convex function. Then, the solution sequence produced by `BrasCPD` satisfies*

$$\liminf_{r \to \infty} \mathbb{E}\left[ \left\| \boldsymbol{P}_{(n)}^{(r)} \right\|^2 \right] = 0, \ \forall\ n.$$

**Remark 1** *The convergence result in Proposition 2 inherits a drawback from single-block stochastic proximal gradient algorithms for nonsmooth nonconvex optimization. To be specific, the relatively strong assumption 3 is required to ensure convergence. Assumption 3 implies that the variance of the gradient estimation error $\boldsymbol{\delta}_{(\xi^{(r)})}^{(r)} = \boldsymbol{G}_{(\xi^{(r)})}^{(r)} - \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})$ converges to zero. This is not entirely trivial. One way to fulfill this assumption is to increase the minibatch size along the iterations, e.g., by setting [30], [33]:*

$$|\mathcal{F}_n^{(r)}| = \mathcal{O}(\lceil r^{1+\epsilon} \rceil), \quad \forall \epsilon > 0.$$

*Another popular way for achieving* (19) *is to use some advanced variance reduction techniques such as SVRG [46]—which may go beyond the scope of this paper and thus is left out of the discussion. Also notice that as the convergence analysis is pessimistic, in practice constant minibatch size works fairly well—as we will see soon.*

## V. An Adaptive Stepsize Scheme

One may have noticed that the convergence theories in Propositions 1-2 do not specify the sequence $\alpha^{(r)}$ except for the two constraints in Assumption 1. This often gives rise to agonizing tuning experience for practitioners when implementing stochastic algorithms.

Recently, a series of algorithms were proposed in the machine learning community for adaptive stepsize scheduling when training deep neural networks [52]–[54]. Most of these works are variants of the `Adagrad` algorithm [34]. The insight of `Adagrad` can be understood as follows: If one optimization variable has been heavily updated before, then it is given a smaller stepsize for the current iteration (and a larger stepsize otherwise). This way, all the optimization variables can be updated in a balanced manner. `Adagrad` was proposed for single-block algorithms, and this simple strategy admits many provable benefits under the context of convex optimization [34]. For our multi-block nonconvex problem, we extend the idea and propose the following updating rule:

**Algorithm 2:** `AdaCPD`

**input :** $N$-way tensor $\underline{\boldsymbol{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$; rank $F$; sample size $B$ , initialization $\{\boldsymbol{A}_{(n)}^{(0)}\}$

$r \leftarrow 0$;
**repeat**
  uniformly sample $n$ from $\{1, \ldots, N\}$, then sample $\mathcal{F}_n$ uniformly from $\{1, \ldots, J_n\}$ with $|\mathcal{F}_n| = B$;
  form the stochastic gradient $\boldsymbol{G}^{(r)} \leftarrow$ (11);
  determine the step size $\boldsymbol{\eta}_{(n)}^{(r)} \leftarrow$ (20a)
  update $\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow$ (20b), $\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}$ for $n' \neq n$;
  $r \leftarrow r + 1$;
**until** *some stopping criterion is reached*;
**output:** $\{\boldsymbol{A}_{(n)}^{(r)}\}_{n=1}^{N}$

In iteration $r$, if $\xi^{(r)} = n$, then, for all $i \in \{1, \ldots, I_n\}$ and all $f \in \{1, \ldots, F\}$, we have

$$[\boldsymbol{\eta}_{(n)}^{(r)}]_{i,f} \leftarrow \frac{\eta}{\left(b + \sum_{t=1}^{r} [\boldsymbol{G}_{(n)}^{(t)}]_{i,f}^2\right)^{1/2+\epsilon}}, \tag{20a}$$

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \mathsf{Prox}_{h_n}\left(\boldsymbol{A}_{(n)}^{(r)} - \boldsymbol{\eta}_{(n)}^{(r)} \circledast \boldsymbol{G}_{(n)}^{(r)}\right), \tag{20b}$$

$$\boldsymbol{A}_{(n')}^{(r+1)} \leftarrow \boldsymbol{A}_{(n')}^{(r)}, \quad n' \neq n, \tag{20c}$$

where $\eta, b, \epsilon > 0$. We note that $b > 0$, $\epsilon > 0$ are technical conditions used for establishing theoretical convergence. In practice, setting $b = \epsilon = 0$ does not hurt the performance and we also observe a slight gain in runtime performance when $\epsilon = 0$. The `Adagrad` version of block-randomized CPD algorithm is very simple to implement. The algorithm is summarized in Algorithm 2, which is named `AdaCPD`.

As one will soon see, such a simple stepsize strategy is very robust to a large number of scenarios under test—i.e., in most of the cases, `AdaCPD` performs well without tuning the stepsize schedule. In addition, the `AdaCPD` algorithm works well for both the constrained and unconstrained case.

Proving convergence for nonconvex `Adagrad`-like algorithms is quite challenging [55], [56]. We show that:

**Proposition 3** *Assume $h_n(\cdot) = 0$ for all $n$, and that $\mathsf{Pr}(\xi^{(r)} = n) = 1/N$ for all $r$ and $n$. Under the Assumptions 1-2, the solution sequence produced by `AdaCPD` satisfies*

$$\mathsf{Pr}\left(\liminf_{r\to\infty} \|\nabla f(\boldsymbol{\theta}^{(r)})\|^2 = 0\right) = 1.$$

Proposition 3 asserts that the algorithm converges almost surely. The proof is relegated to Appendix D. Our proof extends the idea from a recent paper [55] that focuses on using `Adagrad` for solving single-block nonconvex problems. As mentioned, our two-level sampling strategy makes our algorithm very similar to single-block SGD with a scaled gradient estimation (cf. Appendix A), and thus with careful modifications, the key proof techniques in [55] goes through. Nevertheless, we detail the proof for being self-containing.

## VI. Numerical Results

In this section, we use simulations and real-data experiments to showcase the effectiveness of the proposed algorithm.

### A. Synthetic Data Simulations

*1) Data Generation:* Throughout this subsection, we use synthetic third-order tensors (i.e., $N = 3$) whose latent factors are drawn from i.i.d. uniform distribution between 0 and 1—unless otherwise specified. This way, large and *dense* tensors can be created. For simplicity, we set $I_n = I$ for all $n$ and test the algorithms on tensors having different $I_n$'s and $F$'s. In some simulations, we also consider CPD for noisy tensors, i.e., factoring data tensors that have the following signal model:

$$\underline{\boldsymbol{Y}} = \underline{\boldsymbol{X}} + \underline{\boldsymbol{N}},$$

where $\underline{\boldsymbol{X}}$ is the noiseless low-rank tensor and $\underline{\boldsymbol{N}}$ denotes the additive noise. We use zero-mean i.i.d. Gaussian noise with variance $\sigma_N^2$ in our simulations, and the signal-to-noise ratio (SNR) (in dB) is defined as $\mathsf{SNR} = 10\log_{10}\left(\frac{\frac{1}{\prod_{n=1}^{N} I_n}\|\underline{\boldsymbol{X}}\|^2}{\sigma_N^2}\right)$.

*2) Baselines:* A number of baseline algorithms are employed as benchmarks. Specifically, we mainly use the `AO-ADMM` algorithm [57] and the `APG` algorithm [14] as our baselines since they are the most flexible algorithms with the ability of handling many different regularizations and constraints. We also present the results output by the `CPRAND` algorithm [26]. Note that we are preliminarily interested in constrained/regularized CPD. Because `CPRAND` operates without constraints, the comparison is not entirely fair (e.g., `CPRAND` can potentially attain smaller cost values since it has a much larger feasible set if other algorithms operate with constraints). Nevertheless, we employ it as a benchmark since it uses the same fiber sampling strategy as ours. To make the comparisons more comprehensive, we also offer a simple modification for `CPRAND` to incorporate constraints/regularization terms. Specifically, we apply the proximal operators associated with the constraint/regularization terms to the original `CPRAND` updates, and we denote this baseline as `CPRAND-Prox`. All the algorithms are initialized with the same random initialization; i.e., $\boldsymbol{A}_{(0)}$'s entries follow the uniform distribution between 0 and 1.

*3) Parameter Setting:* For `BrasCPD`, we set

$$\alpha^{(r)} = \frac{\alpha}{r^{\beta}}, \tag{21}$$

where $r$ is the number of iterations, $\beta = 10^{-6}$ and $\alpha$ typically takes a value in between 0.001 and 0.1, and we try multiple choices of $\alpha$ in our simulations. Our experience is that, under such settings, $\alpha$ is the main tuning parameter that affects the performance of `BrasCPD`. The batch size $|\mathcal{F}_n|$ is typically set to be below 25 throughout this section. For `AdaCPD`, we fix $b = 10^{-6}$, $\epsilon = 0$, and $\eta = 1$ for all the simulations. For `CPRAND`, we follow the instruction in the original paper [26] and sample $10F\log_2 F$ fibers for each update.

*4) Performance Metrics:* To measure the performance, we employ two metrics. We mainly use the estimation accuracy for the latent factors, $\boldsymbol{A}_{(n)}$ for $n = 1, \ldots, N$, as our performance indicator. The accuracy is measured by the *mean*

*squared error* (MSE) which is as defined in [58], [59]:

$$\text{MSE} = \min_{\pi(f)\in\{1,\ldots,F\}} \frac{1}{F}\sum_{f=1}^{F}\left\| \frac{\boldsymbol{A}_{(n)}(:,\pi(f))}{\|\boldsymbol{A}_{(n)}(:,\pi(f))\|_2} - \frac{\widehat{\boldsymbol{A}}_{(n)}(:,f)}{\|\widehat{\boldsymbol{A}}_{(n)}(:,f)\|_2}\right\|_2^2 \tag{22}$$

where $\widehat{\boldsymbol{A}}_{(n)}$ denotes the estimate of $\boldsymbol{A}_{(n)}$ and $\pi(f)$'s are under the constraint $\{\pi(1),\ldots,\pi(F)\} = \{1,\ldots,F\}$—which is used to fix the intrinsic column permutation in CPD. We also use the cost function, i.e., cost $= (1/\prod_{n=1}^{N} I_n) \times f(\boldsymbol{\theta}^{(r)})$ as a reference in some of the simulation tables.

Since the algorithms under test have very different operations and subproblem-solving strategies, it may be challenging to find an exactly unified complexity measure. In this section, we show the peformance of the algorithms against the number of MTTKRP operations $\boldsymbol{H}_{(n)}^{\top}\boldsymbol{X}_{(n)}$ used, since $\boldsymbol{H}_{(n)}^{\top}\boldsymbol{X}_{(n)}$ is the most costly step that dominates the complexity of all the algorithms under comparison. For the stochastic optimization/sketching based algorithms, we also test the MSE/cost value of the algorithms against runtime and the number of sampled entries for updating the latent factors. The latter is particularly meaningful under the stochastic settings, since it affects the communication overhead between the data storage units (e.g., the hard disks) and the computation units, e.g., CPUs and GPUs. All the simulations are conducted in Matlab. The results in this section are obtained from 50 trials with different randomly generated tensors.

### B. Results

Fig. 1 in Sec. I has shown the MSE performance of the algorithms in a relatively small-size example, where $I_n = I = 100$, $F = 10$ and the nonnegativity constraints are used in the algorithms. In that simulation, we use $|\mathcal{F}_n| = 20$ so that every 500 iterations of the proposed algorithm compute a full MTTKRP. One can see that for this relatively easy case, all the algorithms can reach a good estimation accuracy for the latent factors. Nevertheless, the proposed methods exhibit remarkably higher efficiency.

Fig. 3 shows the MSEs of the estimated latent factors by the algorithms under $I = 300$ and $F = 10$ benchmarked by more baselines. The result is the *median* of 50 Monte Carlo trials; we use median here since mean is dominated by outlying trials, even if there is only one outlying trial. We set $|\mathcal{F}_n| = 18$ so that the proposed algorithms use 5,000 iterations to compute a full MTTKRP. All the algorithms use nonnegativity constraints except `CPRAND`. One can see that under this setting, the most competitive algorithms are `CPRAND`, `CPRAND-Prox`, and `AdaCPD`.

In Fig. 4, we increase the rank to be $F = 200$. There are several observations in order: First, the stochastic algorithms (i.e., `BrasCPD`, `AdaCPD`, `CPRAND` and `CPRAND-Prox`) are much more efficient relative to the deterministic algorithms (`AO-ADMM` and `APG`). After 60 MTTKRPs computed, the stochastic algorithms often have reached a reasonable level of MSE. This is indeed remarkable, since 60 MTTKRPs are roughly equivalent to 20 iterations of `AO-ADMM` and `APG`.

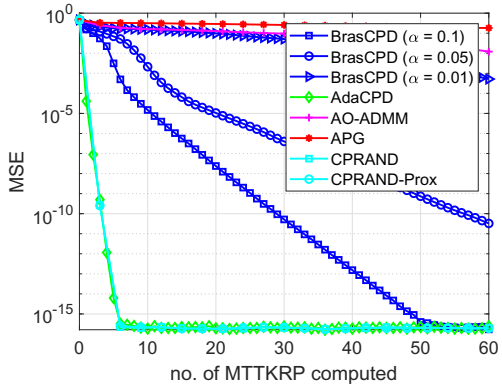



Fig. 3. Median of MSEs against MTTKRP. $I_1 = I_2 = I_3 = 300$ and $F = 10$. $\boldsymbol{A}_{(n)} \geq \boldsymbol{0}$.

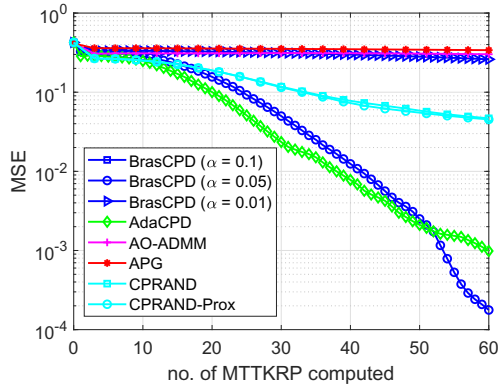



Fig. 4. Median of MSEs against MTTKRP. $I_1 = I_2 = I_3 = 300$ and $F = 200$. $\boldsymbol{A}_{(n)} \geq \boldsymbol{0}$.

Second, `CPRAND` and `CPRAND-Prox` are not as competitive in this high-rank regime. In particular, `BrasCPD` with $\alpha = 0.1$ gives the most promising performance. However, the performance of `BrasCPD` is affected a bit significantly by the parameters $\alpha$. One can see that using $\alpha = 0.05$ and $\alpha = 0.01$, the algorithm does not give promising results under this setting. Third, `AdaCPD` yields the second lowest MSEs after 60 MTTKRPs—while not using any parameter tuning.

Figs 5-6 show the MSEs of the stochastic algorithms (i.e., `BrasCPD`, `AdaCPD`, `CPRAND` and `CPRAND-Prox`) against the number of sampled entries and runtime, respectively. Here, we set $I_1 = I_2 = I_3 = 300$ and $F = 100$. One can see that `BrasCPD` and `AdaCPD` do not use many data samples to reach a good MSE level. This indicates that the communication overhead performance of the proposed algorithms is promising. In terms of runtime, `CPRAND` and `CPRAND-Prox` start with quick decrease of the MSE—they reach MSE$\approx 10^{-2}$ within 50 seconds, while the proposed approaches reach this level of accuracy after 100 seconds. Nevertheless, the MSEs of `CPRAND` and `CPRADN-Prox` are somehow stuck at this level, but the proposed algorithms can reach a much better accuracy for estimating the latent factors. We would like to remark that the runtime performance of stochastic algorithms are affected by the programming language used (i.e., `Matlab` in this case).

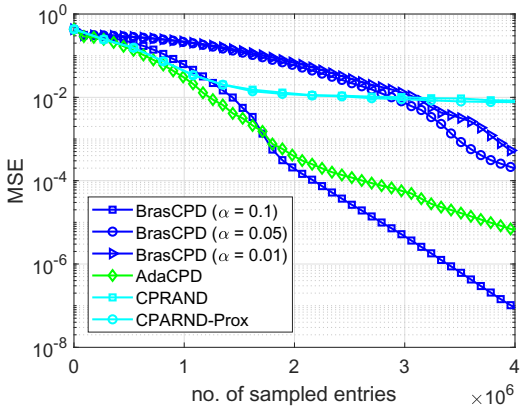



Fig. 5. MSE against the number of sampled entries for $I_1 = I_2 = I_3 = 300$, $F = 100$.

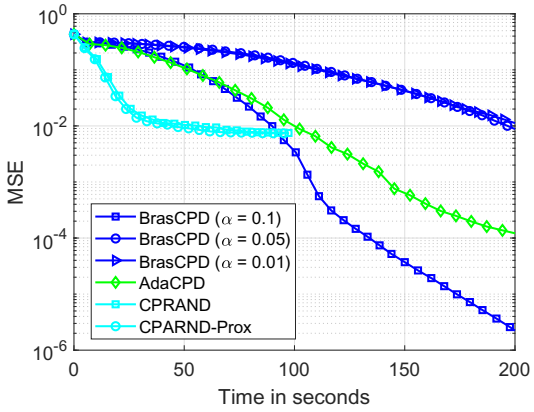



Fig. 6. MSE against runtime (sec.) for $I_1 = I_2 = I_3 = 300$, $F = 100$.

Typically, interpreted languages (e.g., `Matlab` and `Python`) are not specialized for handling "for" loops, which is heavily used in stochastic algorithms (especially when $|\mathcal{F}_n|$ is small). Hence, real-system implementations for these algorithms could be much faster.

Table II shows the mean and median of the MSEs and cost values output by the algorithms when the tensor rank varies under $I = 300$. All the algorithms are stopped after 60 full MTTKRPs are used. One can see that `BrasCPD` exhibits a quite competitive MSE performance if a proper $\alpha$ is chosen, under the employed stepsize schedule in (21). However, one can see that when $F$ changes, there is a risk that `BrasCPD` runs into numerical issues and yields unbounded solutions. This suggests that `BrasCPD` may need extra care for tuning its stepsize. In principle, when the problem setting changes, the "best" $\alpha$ of `BrasCPD` also changes. Our experience is that when $|\mathcal{F}_n|$ increases, using a properly scaled up $\alpha$ may help accelerate convergence. On the other hand, `AdaCPD` always outputs reasonably good results. More importantly, `AdaCPD` runs without tuning the stepsize parameters—which shows the power of the adaptive stepsize scheduling strategy. Another remark is that `CPRAND` and `CPRAND-Prox` work well when $F = 10$ and $F = 50$. In particular, `CPRAND-Prox` largely outperforms `CPRAND` when $F = 50$, showing the benefit of

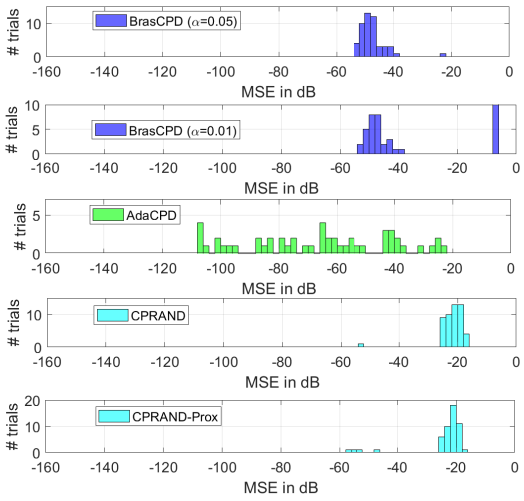



Fig. 7. Histograms of the algorithms; $I = 300, F = 100$.

explicitly considering constraints.

Table III shows the performance under different $I$'s when $F = 100$. In general, when $I$ increases, the performance of all the algorithms improves—with a fixed $F$, a larger $I$ means more data and more "degrees of freedom" available, which normally leads to better performance. Again, `BrasCPD` with a proper $\alpha$ and `AdaCPD` in general outperform the baselines. One particular observation is that, although the mean and median MSEs of `AdaCPD` are both low, the median is sometimes much better than the mean (cf. the case when $I = 400$), which indicates that there exist outlying trials (i.e., trials where `AdaCPD` does not produce very low MSE results). The median-mean gap is less often observed for other algorithms, e.g., `BrasCPD` and `CPRAND`. Fig. 7 may better illustrate the situation, where the histograms of the MSE (in dB) of the algorithms are shown. One can see that, although the worst-case result of `AdaCPD` is still acceptable (with MSE$< 10^{-4}$), the MSE of `AdaCPD` clearly has a larger variance compared to `BrasCPD` with $\alpha = 0.1$. This shows a trade-off between the easiness of stepsize scheduling and the risk of converging to less accurate solutions.

Table IV shows the median of the MSEs after 450 MTTKRPs of under same simulation setting (which is roughly equivalent to 150 iterations of AO-ADMM and APG). One can see that the performance of all the algorithms have improved. Note that due to resource limitations, we could not run 450 MTTKRPs for every table in this section and had to stop at 60 MTTKRPs. Hence, the capabilities of all the algorithms may not have fully shown up in those tables. Nonetheless, the 60-MTTKRP results can serve as a reference under limited time and computational resources.

Tables V-VI show the estimation accuracy of the latent factors by the algorithms under different SNRs. Here, we use a low-rank case where $I_1 = I_2 = I_3 = 100$ and $F = 20$, under which `CPRAND` and `CPRAND-Prox` are more competitive. In a noisy environment, the ability of handling constraints/regularizations is essential for a CPD algorithm,

TABLE II
MSEs OF THE ESTIMATED LATENT FACTORS BY THE ALGORITHMS UNDER DIFFERENT $F$; $I = 300$; ALL THE ALGORITHMS ARE STOPPED AFTER COMPUTING 60 MTTKRPS. "NAN" MEANS THE ALGORITHM OUTPUTS UNBOUNDED SOLUTIONS. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$.

| Algorithm | Metric | | $F$ = 10 | 50 | 100 | 200 |
|---|---|---|---|---|---|---|
| BrasCPD ($\alpha = 0.1$) | MSE | Mean | 1.97E-16 | 1.98E-11 | 6.12E-10 | NaN |
| | | Median | 1.70E-16 | 5.66E-12 | 3.82E-10 | NaN |
| BrasCPD ($\alpha = 0.05$) | MSE | Mean | 1.02E-10 | 1.33E-06 | 1.06E-04 | 4.08E-04 |
| | | Median | 3.26E-11 | 7.20E-07 | 1.46E-05 | 1.77E-04 |
| BrasCPD ($\alpha = 0.01$) | MSE | Mean | 2.80E-03 | 0.1108 | 0.1968 | 0.2599 |
| | | Median | 5.18E-04 | 0.1104 | 0.0000 | 0.2604 |
| AdaCPD | MSE | Mean | 2.30E-16 | 1.27E-04 | 3.54E-04 | 0.0175 |
| | | Median | 2.44E-16 | 5.43E-15 | 2.96E-07 | 9.86E-04 |
| AO-ADMM | MSE | Mean | 2.25E-02 | 0.1900 | 0.2667 | 0.3018 |
| | | Median | 1.21E-02 | 0.1886 | 0.2664 | 0.3018 |
| APG | MSE | Mean | 1.80E-01 | 0.2993 | 0.3254 | 0.3399 |
| | | Median | 1.83E-01 | 0.2995 | 0.3255 | 0.3398 |
| CPRAND | MSE | Mean | 2.00E-16 | 0.0037 | 0.0082 | 0.0475 |
| | | Median | 1.92E-16 | 0.0027 | 0.0075 | 0.0468 |
| CPRAND-Prox | MSE | Mean | 2.02E-16 | 0.0026 | 0.0073 | 0.0459 |
| | | Median | 1.92E-16 | 3.10E-11 | 0.0074 | 0.0453 |
| BrasCPD ($\alpha = 0.1$) | Cost | Mean | 1.66E-19 | 2.63E-12 | 1.64E-10 | NaN |
| | | Median | 4.31E-20 | 7.23E-13 | 1.03E-10 | NaN |
| BrasCPD ($\alpha = 0.05$) | Cost | Mean | 3.52E-12 | 1.84E-07 | 2.78E-05 | 3.01E-04 |
| | | Median | 1.15E-12 | 1.00E-07 | 3.65E-06 | 8.99E-05 |
| BrasCPD ($\alpha = 0.01$) | Cost | Mean | 1.49E-04 | 0.0226 | 0.0698 | 0.1620 |
| | | Median | 2.89E-05 | 0.0226 | 0.0701 | 0.1620 |
| AdaCPD | Cost | Mean | 9.45E-31 | 4.95E-05 | 1.64E-04 | 0.0142 |
| | | Median | 1.95E-31 | 8.86E-16 | 1.05E-07 | 8.43E-04 |
| AO-ADMM | Cost | Mean | 1.10E-03 | 0.0361 | 0.0869 | 0.1791 |
| | | Median | 8.10E-04 | 0.0359 | 0.0868 | 0.1790 |
| APG | Cost | Mean | 2.77E-02 | 0.2665 | 0.5636 | 1.1709 |
| | | Median | 2.79E-02 | 0.2659 | 0.5637 | 1.1715 |
| CPRAND | Cost | Mean | 1.88E-31 | 1.79E-10 | 0.0023 | 0.0329 |
| | | Median | 1.87E-31 | 1.59E-12 | 0.0027 | 0.0319 |
| CPRAND-Prox | Cost | Mean | 1.89E-31 | 0.0168 | 0.2128 | 129.4668 |
| | | Median | 1.88E-31 | 4.73E-12 | 0.1279 | 120.7575 |

TABLE III
PERFORMANCE OF THE ALGORITHMS UNDER VARIOUS $I$'S, $F = 100$. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$; ALL THE ALGORITHMS ARE STOPPED AFTER COMPUTING 60 MTTKRPS. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$.

| Algorithm | Metric | | $I$ = 100 | 200 | 300 |
|---|---|---|---|---|---|
| BrasCPD ($\alpha = 0.1$) | MSE | Mean | 0.1567 | 4.41E-04 | 6.12E-10 |
| | | Median | 0.1586 | 8.54E-05 | 3.82E-10 |
| BrasCPD ($\alpha = 0.05$) | MSE | Mean | 0.2378 | 0.0297 | 1.06E-04 |
| | | Median | 0.2364 | 0.0282 | 1.46E-05 |
| BrasCPD ($\alpha = 0.01$) | MSE | Mean | 0.2861 | 0.2786 | 0.1968 |
| | | Median | 0.2857 | 0.2786 | 0.0000 |
| AdaCPD | MSE | Mean | 0.1312 | 0.0040 | 3.54E-04 |
| | | Median | 0.1281 | 1.66E-04 | 2.96E-07 |
| AO-ADMM | MSE | Mean | 0.2396 | 0.2586 | 0.2667 |
| | | Median | 0.2395 | 0.2585 | 0.2664 |
| APG | MSE | Mean | 0.2961 | 0.3162 | 0.3254 |
| | | Median | 0.2961 | 0.3163 | 0.3255 |
| CPRAND | MSE | Mean | 0.1901 | 0.0139 | 0.0082 |
| | | Median | 0.1914 | 0.0134 | 0.0075 |
| CPRAND-Prox | MSE | Mean | 0.1873 | 0.0141 | 0.0073 |
| | | Median | 0.1879 | 0.0146 | 0.0074 |
| BrasCPD ($\alpha = 0.1$) | Cost | Mean | 0.0561 | 1.39E-04 | 1.64E-10 |
| | | Median | 0.0556 | 2.22E-05 | 1.03E-10 |
| BrasCPD ($\alpha = 0.05$) | Cost | Mean | 0.0752 | 0.0119 | 2.78E-05 |
| | | Median | 0.0744 | 0.0115 | 3.65E-06 |
| BrasCPD ($\alpha = 0.01$) | Cost | Mean | 0.1019 | 0.0881 | 0.0698 |
| | | Median | 0.1010 | 0.0881 | 0.0701 |
| AdaCPD | Cost | Mean | 0.0499 | 0.0018 | 1.64E-04 |
| | | Median | 0.0499 | 5.01E-05 | 1.05E-07 |
| AO-ADMM | Cost | Mean | 0.0736 | 0.0833 | 0.0869 |
| | | Median | 0.0735 | 0.0833 | 0.0868 |
| APG | Cost | Mean | 0.4248 | 0.5249 | 0.5636 |
| | | Median | 0.4251 | 0.5251 | 0.5637 |
| CPRAND | Cost | Mean | 0.0585 | 0.0046 | 0.0023 |
| | | Median | 0.0575 | 0.0048 | 0.0027 |
| CPRAND-Prox | Cost | Mean | 15.0076 | 1.2778 | 0.2128 |
| | | Median | 12.3764 | 0.9751 | 0.1279 |

TABLE IV
MEDIAN OF THE MSES UNDER VARIOUS $I$'S, $F = 100$. $\boldsymbol{A}_n \geq 0$; ALL THE ALGORITHMS ARE STOPPED AFTER COMPUTING 450 MTTKRPS.

| Algorithm | Metric | $I$ = 100 | 200 | 300 |
|---|---|---|---|---|
| BrasCPD ($\alpha = 0.1$) | MSE | 3.76E-06 | 4.74E-17 | 9.56E-17 |
| BrasCPD ($\alpha = 0.05$) | MSE | 0.0016 | 1.01E-14 | 8.75E-17 |
| BrasCPD ($\alpha = 0.01$) | MSE | 0.1957 | 6.13E-04 | 8.44E-08 |
| AdaCPD | MSE | 9.51E-05 | 1.60E-07 | 2.43E-06 |
| AO-ADMM | MSE | 3.45E-04 | 1.59E-04 | 1.44E-04 |
| APG | MSE | 0.2615 | 0.2873 | 0.3006 |
| CPRAND | MSE | 0.0230 | 0.0070 | 0.0073 |
| CPRAND-Prox | MSE | 0.0260 | 0.0069 | 0.0065 |

TABLE V
PERFORMANCE OF THE ALGORITHMS UNDER VARIOUS SNRS; ALL THE ALGORITHMS ARE STOPPED AFTER COMPUTING 60 MTTKRPS. $I_1 = I_2 = I_3 = 100$, $F = 20$. $\boldsymbol{A}_{(n)} \geq \mathbf{0}$.

| Algorithm | Metric | | SNR 10 | 20 | 30 | 40 |
|---|---|---|---|---|---|---|
| BrasCPD ($\alpha = 0.1$) | MSE | Mean | 0.0433 | 0.0240 | 0.0271 | 0.0281 |
| | | Median | 0.0453 | 0.0195 | 0.0263 | 0.0257 |
| BrasCPD ($\alpha = 0.05$) | MSE | Mean | 0.1168 | 0.1135 | 0.1133 | 0.1093 |
| | | Median | 0.1232 | 0.1149 | 0.1121 | 0.1090 |
| BrasCPD ($\alpha = 0.01$) | MSE | Mean | 0.2135 | 0.2114 | 0.2104 | 0.2087 |
| | | Median | 0.2141 | 0.2121 | 0.2102 | 0.2078 |
| AdaCPD | MSE | Mean | 0.0179 | 0.0040 | 0.0017 | 1.39E-04 |
| | | Median | 0.0168 | 0.0036 | 7.02E-04 | 1.24E-04 |
| AO-ADMM | MSE | Mean | 0.1012 | 0.0925 | 0.0874 | 0.0905 |
| | | Median | 0.1009 | 0.0921 | 0.0870 | 0.0899 |
| APG | MSE | Mean | 0.2326 | 0.2291 | 0.2282 | 0.2284 |
| | | Median | 0.2319 | 0.2286 | 0.2279 | 0.2291 |
| CPRAND | MSE | Mean | 0.2049 | 0.0088 | 0.0021 | 0.0020 |
| | | Median | 0.2046 | 0.0077 | 7.36E-04 | 7.33E-05 |
| CPRAND-Prox | MSE | Mean | 0.1737 | 0.0108 | 0.0024 | 0.0017 |
| | | Median | 0.1715 | 0.0075 | 7.32E-04 | 7.44E-05 |

TABLE VI
PERFORMANCE OF THE ALGORITHMS UNDER VARIOUS SNRS AFTER COMPUTING 30 MTTKRPS. $I_1 = I_2 = I_3 = 100$, $F = 20$. $\mathbf{1}^\top \boldsymbol{A}_{(n)} = \rho \mathbf{1}^\top$, $\boldsymbol{A}_{(n)} \geq \mathbf{0}$. $\rho = 100$.

| Algorithm | Metric | | SNR 10 | 20 | 30 | 40 |
|---|---|---|---|---|---|---|
| BrasCPD ($\alpha = 0.1$) | MSE | Mean | 0.3946 | 0.3950 | 0.3939 | 0.4032 |
| | | Median | 0.3987 | 0.3993 | 0.3942 | 0.4062 |
| BrasCPD ($\alpha = 0.05$) | MSE | Mean | 0.0705 | 0.0067 | 6.80E-04 | 6.82E-05 |
| | | Median | 0.0705 | 0.0067 | 6.81E-04 | 6.80E-05 |
| BrasCPD ($\alpha = 0.01$) | MSE | Mean | 0.0169 | 0.0054 | 0.0017 | 0.0026 |
| | | Median | 0.0156 | 0.0017 | 2.34E-04 | 7.75E-05 |
| AdaCPD | MSE | Mean | 0.0202 | 0.0019 | 3.94E-04 | 6.13E-05 |
| | | Median | 0.0193 | 0.0019 | 3.95E-04 | 6.11E-05 |
| AO-ADMM | MSE | Mean | 0.0884 | 0.0759 | 0.0824 | 0.0772 |
| | | Median | 0.0881 | 0.0763 | 0.0823 | 0.0805 |
| APG | MSE | Mean | 0.2186 | 0.2149 | 0.2144 | 0.2146 |
| | | Median | 0.2188 | 0.2152 | 0.2157 | 0.2148 |
| CPRAND | MSE | Mean | 0.1937 | 0.0097 | 0.0034 | 0.0032 |
| | | Median | 0.1944 | 0.0077 | 7.44E-04 | 7.37E-05 |
| CPRAND-Prox | MSE | Mean | 0.1509 | 0.0670 | 0.0599 | 0.0602 |
| | | Median | 0.1519 | 0.0663 | 0.0591 | 0.0582 |

since prior information on the latent factors can help improve estimation accuracy. Table V and Table VI test the cases where $\boldsymbol{A}_{(n)}$ is elementwise nonnegative and the columns of $\boldsymbol{A}_{(n)}$ reside in a scaled version of the probability simplex, respectively. One can see from the two tables that both `BrasCPD` (with a proper $\alpha$) and `AdaCPD` work very well. In Table VI, one can see that `BrasCPD` again shows its sensitivity to the choice of $\alpha$, with $\alpha = 0.1$ and $0.05$ actually not working. One can also see that, in the low-SNR regime (SNR=10 and 20dB), `BrasCPD` with proper $\alpha$ and `AdaCPD` outperform the baselines. `CPRAND-Prox` also works well for the nonnegativity constraint case (especially for SNR=30 and 40dB), but not as promising for the simplex constraints.

### C. Real-Data Experiment

In this subsection, we test our algorithm on a constrained tensor decomposition problem; i.e., we apply the proposed `BrasCPD` and `AdaCPD` to factor hyperspectral images. Hyperspectral images (HSIs) are special images with pixels measured at a large number of wavelengths. Hence, an HSI is usually stored as a third-order tensor with two spatial coordinates and one spectral coordinate. HSIs are dense tensors and thus are suitable for testing the proposed algorithms. We use sub-images of the Indian Pines dataset that has a size of $145 \times 145 \times 220$ and the Pavia University dataset[3] that has a

[3]Both datasets are available online: http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes

size of $610 \times 340 \times 103$. We apply the stochastic algorithms to the datasets by fixing $|\mathcal{F}_n| = 500$ in this section.

TABLE VII
PERFORMANCE OF THE ALGORITHMS ON THE INDIAN PINES DATASET UNDER DIFFERENT $F$'S.

| Algorithm | Metric | $F$ | | | |
|---|---|---|---|---|---|
| | | 10 | 20 | 30 | 40 |
| BrasCPD ($\alpha$=4) | Cost | 6.82E-04 | 4.68E-04 | 3.46E-04 | 3.58E-04 |
| BrasCPD ($\alpha$=3) | Cost | 6.86E-04 | 4.52E-04 | 3.54E-04 | 3.35E-04 |
| BrasCPD ($\alpha$=2) | Cost | 6.88E-04 | 6.11E-04 | 5.38E-04 | 4.11E-04 |
| AdaCPD | Cost | 6.23E-04 | 4.56E-04 | 5.40E-04 | 5.23E-04 |
| AO-ADMM | Cost | 7.70E-04 | 5.28E-04 | 5.07E-04 | 4.91E-04 |
| APG | Cost | 0.0019 | 0.0019 | 0.0018 | 0.0019 |
| CPRAND | Cost | 6.74E-04 | 4.82E-04 | 4.35E-04 | 4.08E-04 |
| CPRAND-Prox | Cost | 0.1116 | 0.0021 | 0.0020 | 0.0021 |

TABLE VIII
PERFORMANCE OF THE ALGORITHMS ON THE PAVIA UNIVERSITY DATASET UNDER DIFFERENT $F$'S.

| Algorithm | Metric | $F$ | | |
|---|---|---|---|---|
| | | 50 | 100 | 200 |
| BrasCPD ($\alpha$=4) | Cost | 0.0031 | 0.0027 | 0.0013 |
| BrasCPD ($\alpha$=3) | Cost | 0.0033 | 0.0053 | 0.0031 |
| BrasCPD ($\alpha$=2) | Cost | 0.0044 | 0.0067 | 0.0059 |
| AdaCPD | Cost | 0.0022 | 0.0013 | 0.0008 |
| AO-ADMM | Cost | 0.0378 | 0.0425 | 0.0053 |
| APG | Cost | 0.0074 | 0.0073 | 0.0031 |
| CPRAND | Cost | 0.0033 | 0.0028 | 0.0034 |
| CPRAND-Prox | Cost | 0.0103 | 0.0109 | 8.46E+03 |

Tables VII-VIII show the cost values of the nonnegativity constrained optimization algorithms under different ranks, after computing 120 MTTKRPs for all three modes, which corresponds to 120 iterations for `AO-ADMM` and `APG` (we use this "all-mode MTTKRP" in this section since the tensors are unsymmetrical and thus single-mode MTTKRPs cannot be directly translated to iterations in batch algorithms). One can see that the proposed algorithms show the same merits as we have seen in the simulations: `BrasCPD` can exhibit very competitive performance when $\alpha$ is properly chosen (e.g., when $F = 10$ and $\alpha = 5$ for the Indian Pines dataset); in addition, `AdaCPD` gives consistently good performance without tuning the stepsize manually. Particularly, on the Pavia University dataset, `AdaCPD` gives much lower cost values compared to other algorithms. We also present the results of `CPRAND` and `CPRAND-Prox`. Note that since `CPRAND` does not have constraints, its cost value is naturally lower than the other methods. Hence, this baseline is only for reference. Both `CPRAND` and `CPRAND-Prox` work reasonably well for the two datasets, especially on Indian Pines under low rank. However, we note that for Pavia University, `CPRAND-Prox` does not converge well for the $F = 200$ case.

Fig. 8 shows how the cost values change along with the iterations on the Pavia University data using $F = 100$. One can see that `BrasCPD` ($\alpha = 0.5$) and `AdaCPD` reduce the cost value quickly in this case. After 120 iterations (equivalent to 120 all-mode full MTTKRPs), the batch algorithm `APG` eventually reaches the same cost value level of those of `BrasCPD` ($\alpha = 0.5$) and `AdaCPD`.

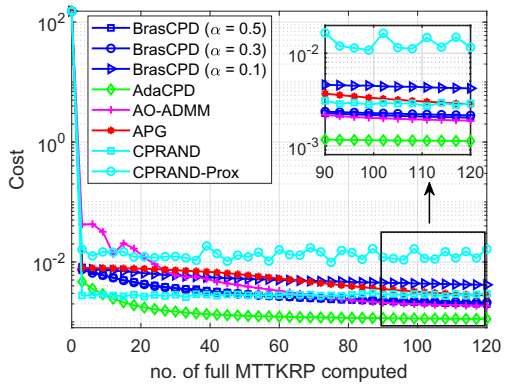



Fig. 8. Number of all-mode MTTKRPs v.s. cost values output by the algorithms when applied to the Pavia University dataset. $F = 100$. Nonnegativity constraint is added .

## VII. CONCLUSION

To conclude, we proposed a block-randomized stochastic proximal gradient based CPD algorithmic framework for large-scale dense tensors. The framework works under a doubly stochastic manner, which randomly selects a mode and then samples a set of fibers for updating the associated latent factor. The framework has a series of nice features including being able to quickly improve estimation accuracy of the latent factors, being flexible with incorporating constraints and regularizations, and having rigorous convergence guarantees. We also proposed a practical and effective adaptive stepsize scheduling method that is reminiscent of recent advances in neural network training algorithms. Simulations and real-data experiments show that the proposed algorithms outperform a number of state-of-art constrained CPD algorithms when dealing with large dense tensors.

## APPENDIX A
## CONNECTION BETWEEN $\nabla f(\boldsymbol{\theta}^{(r)})$ AND $\boldsymbol{G}^{(r)}$

Let $n$ be any integer from $\{1, ..., N\}$, consider the following conditional expectation:

$$\begin{aligned}
\overline{\boldsymbol{G}}_{(n)}^{(r)} &= \mathbb{E}_{\xi^{(r)},\boldsymbol{\zeta}^{(r)}}\left[\boldsymbol{G}_{(n)}^{(r)} \mid \mathcal{B}^{(r)}\right] \\
&\overset{(a)}{=} \mathbb{E}_{\xi^{(r)}}\left[\frac{1}{\binom{J_{\xi^{(r)}}}{B}}\left(\boldsymbol{A}_{(\xi^{(r)})}^{(r)}\boldsymbol{H}_{(\xi^{(r)})}^{\top}\boldsymbol{H}_{(\xi^{(r)})} - \boldsymbol{X}_{(\xi^{(r)})}^{\top}\boldsymbol{H}_{(\xi^{(r)})}\right)\right] \\
&\overset{(b)}{=} \sum_{n'=1}^{N}\frac{\delta(n'-n)}{N\binom{J_{n'}}{B}}\left(\boldsymbol{A}_{(n')}^{(r)}\boldsymbol{H}_{(n')}^{\top}\boldsymbol{H}_{(n')} - \boldsymbol{X}_{(n')}^{\top}\boldsymbol{H}_{(n')}\right) \\
&= \frac{1}{N\binom{J_n}{B}}\left(\boldsymbol{A}_{(n)}^{(r)}\boldsymbol{H}_{(n)}^{\top}\boldsymbol{H}_{(n)} - \boldsymbol{X}_{(n)}^{\top}\boldsymbol{H}_{(n)}\right) \qquad (23)
\end{aligned}$$

where $\delta(\cdot)$ is the Dirac function. In the above chain, (a) is due to Fact 1 and (b) is obtained by evaluating the expectation with respect to the possible modes $\xi^{(r)} = n'$. The last equality shows that $\overline{\boldsymbol{G}}_{(n)}^{(r)}$ is a scaled gradient of the objective function of (3) taken w.r.t. $\boldsymbol{A}_{(n)}^{(r)}$. The block sampling step together with fiber sampling entails us an easy way to estimate the *full gradient w.r.t. all the latent factors* in an unbiased manner.

## APPENDIX B
## PROOF OF PROPOSITION 1

To show Proposition 1, we will need the following [60, Lemma A.5]:

**Lemma 1** *Let $\{a_t\}_t$ and $\{b_t\}_t$ be two nonnegative sequences such that $b_t$ is bounded, $\sum_{t=0}^{\infty} a_t b_t$ converges and $\sum_{t=0}^{\infty} a_t$ diverges, then we have*

$$\liminf_{r\to\infty} b_t = 0.$$

Recall that $\xi^{(r)}$, $\boldsymbol{\zeta}^{(r)}$ are the random mode, fiber chosen at iteration $r$, respectively. Under Assumption 2, we have $\|\boldsymbol{H}^{(r)}_{(\xi^{(r)})}\|_2^2 \le L^{(r)}_{(\xi^{(r)})}$ where $\boldsymbol{H}^{(r)}_{(\xi^{(r)})} = \odot^N_{n'=1,n'\neq\xi^{(r)}} \boldsymbol{A}^{(r)}_{(n')}$ and $L^{(r)}_{(\xi^{(r)})} < \infty$. Combining with Fact 2, we observe:

$$\begin{aligned} f(\boldsymbol{\theta}^{(r+1)}) - f(\boldsymbol{\theta}^{(r)}) &\le \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{A}^{(r+1)}_{(\xi^{(r)})} - \boldsymbol{A}^{(r)}_{(\xi^{(r)})} \right\rangle \\ &\quad + \frac{L}{2}\left\| \boldsymbol{A}^{(r+1)}_{(\xi^{(r)})} - \boldsymbol{A}^{(r)}_{(\xi^{(r)})} \right\|^2 \\ &= -\alpha^{(r)} \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{G}^{(r)}_{(\xi^{(r)})} \right\rangle + \frac{(\alpha^{(r)})^2 L}{2}\left\| \boldsymbol{G}^{(r)}_{(\xi^{(r)})} \right\|^2, \end{aligned}$$

where we have denoted

$$L = \max_{r=0,\ldots,\infty} L^{(r)}_{(\xi^{(r)})} < \infty.$$

Taking expectation conditioned on the filtration $\mathcal{B}^{(r)}$ and the chosen mode index $\xi^{(r)}$, we have

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[ f(\boldsymbol{\theta}^{(r+1)}) \,|\, \mathcal{B}^{(r)}, \xi^{(r)} \right] - f(\boldsymbol{\theta}^{(r)}) \\ &\le -\alpha^{(r)} \left\| \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\|^2 \\ &\qquad + \frac{(\alpha^{(r)})^2 L}{2} \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[ \left\| \boldsymbol{G}^{(r)}_{(\xi^{(r)})} \right\|^2 \,|\, \mathcal{B}^{(r)}, \xi^{(r)} \right] \\ &\le -\alpha^{(r)} \left\| \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\|^2 + \frac{(\alpha^{(r)})^2 LM}{2}, \end{aligned} \tag{24}$$

where the first inequality used the assumption that $L^{(r)}_{(\xi^{(r)})} \le L$ and Fact 1, and the second inequality is a consequence of Assumption 2, as we observe:

$$\begin{aligned} \left\| \boldsymbol{G}^{(r)}_{(\xi^{(r)})} \right\| = \frac{1}{B} \Big\| &\boldsymbol{A}^{(r)}_{(\xi^{(r)})} (\boldsymbol{H}^{(r)}_{(\xi^{(r)})}(\boldsymbol{\zeta}^{(r)}))^\top \boldsymbol{H}^{(r)}_{(\xi^{(r)})}(\boldsymbol{\zeta}^{(r)}) \\ &- \boldsymbol{X}^\top_{(\xi^{(r)})}(\boldsymbol{\zeta}^{(r)}) \boldsymbol{H}^{(r)}_{(\xi^{(r)})}(\boldsymbol{\zeta}^{(r)}) \Big\|. \end{aligned} \tag{25}$$

As $\boldsymbol{X}_{(n)}$ is bounded for all $n$, and all the $\boldsymbol{A}^{(r)}_{(n)}$ are bounded under Assumption 2, we have $\|\boldsymbol{G}^{(r)}_{(\xi^{(r)})}\|^2 \le M$ for all $n, r$ and some $M < \infty$. Taking the expectation w.r.t. $\xi^{(r)}$ yields

$$\begin{aligned} &\mathbb{E}_{\xi^{(r)},\boldsymbol{\zeta}^{(r)}}\left[ f(\boldsymbol{\theta}^{(r+1)}) \,|\, \mathcal{B}^{(r)} \right] - f(\boldsymbol{\theta}^{(r)}) \\ &\le -\alpha^{(r)} \mathbb{E}_{\xi^{(r)}}\left[ \left\| \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\|^2 \right] + \frac{(\alpha^{(r)})^2 ML}{2}. \end{aligned} \tag{26}$$

Note that $\mathbb{E}_{\xi^{(r)}}[\|\nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\|^2] = \|\nabla f(\boldsymbol{\theta}^{(r)})\|^2$. Taking the total expectation (w.r.t. all random variables in $\mathcal{B}^{(r)}$) gives

$$\begin{aligned} &\mathbb{E}\left[ f(\boldsymbol{\theta}^{(r+1)}) \right] - \mathbb{E}\left[ f(\boldsymbol{\theta}^{(r)}) \right] \\ &\le -\alpha^{(r)} \mathbb{E}\left[ \left\| \nabla f(\boldsymbol{\theta}^{(r)}) \right\|^2 \right] + \frac{(\alpha^{(r)})^2 ML}{2}. \end{aligned} \tag{27}$$

Summing up (27) from $t = 0$ to $t = r$, we have

$$\begin{aligned} &\mathbb{E}\left[ f(\boldsymbol{\theta}^{(t+1)}) \right] - f(\boldsymbol{\theta}^{(0)}) \\ &\le \sum_{t=0}^{r} -\alpha^{(t)} \mathbb{E}\left[ \left\| \nabla f(\boldsymbol{\theta}^{(t)}) \right\|^2 \right] + \sum_{t=0}^{r} \frac{(\alpha^{(t)})^2 ML}{2}. \end{aligned}$$

Taking $r \to \infty$, the above implies that

$$\begin{aligned} &\sum_{r=0}^{\infty} \alpha^{(r)} \mathbb{E}\left[ \left\| \nabla f(\boldsymbol{\theta}^{(r)}) \right\|^2 \right] \\ &\qquad \le f(\boldsymbol{\theta}^{(0)}) - f(\boldsymbol{\theta}^{(\star)}) + \sum_{r=0}^{\infty} \frac{(\alpha^{(r)})^2 ML}{2}, \end{aligned} \tag{28}$$

where we have used $f(\boldsymbol{\theta}) \ge f(\boldsymbol{\theta}^{(\star)})$, and $f(\boldsymbol{\theta}^{(\star)})$ denotes the global optimal value. Note that the right hand side above is bounded from above because $\sum_{r=0}^{\infty} (\alpha^{(r)})^2 < \infty$. Hence, using Lemma 1 we conclude:

$$\liminf_{r\to\infty} \mathbb{E}\big[ \|\nabla f(\boldsymbol{\theta}^{(r)})\|^2 \big] = 0.$$

## APPENDIX C
## PROOF OF PROPOSITION 2

### A. Preliminaries

For the constrained case, let us denote $\Phi(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \sum_{n=1}^{N} h_n(\boldsymbol{\theta})$ as the objective function. Unlike the unconstrained case where we measure convergence via observing if the gradient vanishes, the optimality condition of the constrained case is a bit more complicated. Consider the following optimization problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \;\; f(\boldsymbol{\theta}) + h(\boldsymbol{\theta}),$$

where $f(\boldsymbol{\theta})$ is continuously differentiable while $h$ is convex but possibly nonsmooth. The deterministic proximal gradient algorithm for handling this problem is as follows:

$$\boldsymbol{\theta}^{(r+1)} \leftarrow \mathsf{Prox}_h\left( \boldsymbol{\theta}^{(r)} - \alpha^{(r)} \nabla f(\boldsymbol{\theta}^{(r)}) \right).$$

Define $\boldsymbol{P}^{(r)} = \frac{1}{\alpha^{(r)}}\left( \boldsymbol{\theta}^{(r+1)} - \boldsymbol{\theta}^{(r)} \right)$, the update can also be represented as $\boldsymbol{\theta}^{(r+1)} \leftarrow \boldsymbol{\theta}^{(r)} - \alpha^{(r)} \boldsymbol{P}^{(r)}$, which is analogous to the gradient descent algorithm. It can be shown that $\boldsymbol{P}^{(r)} = \boldsymbol{0}$ implies that the necessary optimality condition is satisfied, and thus $\boldsymbol{P}^{(r)}$ can be considered as a "generalized gradient". In the multi-block setting of (13), we define:

$$\begin{aligned} &\boldsymbol{P}^{(r)}_{(n)} \\ &= \frac{1}{\alpha^{(r)}}\left( \boldsymbol{A}^{(r+1)}_{(n)} - \mathsf{Prox}_{h_n}\left( \boldsymbol{A}^{(r)}_{(n)} - \alpha^{(r)} \nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)}) \right) \right). \end{aligned}$$

To show that the `BrasCPD` algorithm finds a stationary point, our goal is to show the subsequence convergence of $\mathbb{E}\left[ \boldsymbol{P}^{(r)}_{(n)} \right]$ to zero for all $n$ as $r \to \infty$.

### B. Proof

Our update is equivalent to the following:

$$\boldsymbol{A}_{(n)}^{(r+1)} \leftarrow \arg\min_{\boldsymbol{A}_{(n)}} \left\langle \boldsymbol{G}_{(n)}^{(r)}, \boldsymbol{A}_{(n)} - \boldsymbol{A}_{(n)}^{(r)} \right\rangle + \frac{1}{2\alpha^{(r)}} \left\| \boldsymbol{A}_{(n)} - \boldsymbol{A}_{(n)}^{(r)} \right\|^2 + h_n(\boldsymbol{A}_{(n)}) \tag{29}$$

for a randomly selected $n$, i.e., the above is the proximal operator. For a given $\xi^{(r)}$, we have

$$\begin{aligned} & h_{\xi^{(r)}}\left(\boldsymbol{A}_{(\xi^{(r)})}^{(r+1)}\right) - h_{\xi^{(r)}}\left(\boldsymbol{A}_{(\xi^{(r)})}^{(r)}\right) \\ \leq & -\left\langle \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right\rangle - \frac{1}{2\alpha^{(r)}} \left\| \boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right\|^2 \end{aligned}$$

by the optimality of $\boldsymbol{A}_{(\xi^{(r)})}^{(r+1)}$ for solving Problem (29).

By the block Lipschitz continuity of the smooth part (cf. Fact 2), we have

$$\begin{aligned} f(\boldsymbol{\theta}^{(r+1)}) - f(\boldsymbol{\theta}^{(r)}) \leq & \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right\rangle \\ & + \frac{L_{(\xi^{(r)})}^{(r)}}{2} \left\| \boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right\|^2, \end{aligned}$$

where $f$ denotes the smooth part in the objective function and

$$L_{(\xi^{(r)})}^{(r)} = \lambda_{\max}\left( \left(\boldsymbol{H}_{(\xi^{(r)})}^{(r)}\right)^\top \boldsymbol{H}_{(\xi^{(r)})}^{(r)} \right) \leq L.$$

Combining the two inequalities, we have

$$\begin{aligned} \Phi(\boldsymbol{\theta}^{(r+1)}) \leq & \Phi(\boldsymbol{\theta}^{(r)}) \\ & - \alpha^{(r)} \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\rangle \\ & + \left( \frac{L(\alpha^{(r)})^2}{2} - \frac{\alpha^{(r)}}{2} \right) \|\boldsymbol{p}_{(\xi^{(r)})}^{(r)}\|^2, \end{aligned} \tag{30}$$

where we have defined:

$$\boldsymbol{p}_{(\xi^{(r)})}^{(r)} = \frac{1}{\alpha^{(r)}} \left( \boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right).$$

The inequality in (30) can be further written as

$$\begin{aligned} & \Phi(\boldsymbol{\theta}^{(r+1)}) - \Phi(\boldsymbol{\theta}^{(r)}) \\ & \leq -\alpha^{(r)} \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{p}_{(\xi^{(r)})}^{(r)} - \boldsymbol{P}_{(\xi^{(r)})}^{(r)} \right\rangle \\ & \quad - \alpha^{(r)} \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{P}_{(\xi^{(r)})}^{(r)} \right\rangle \\ & \quad + \left( \frac{L(\alpha^{(r)})^2}{2} - \frac{\alpha^{(r)}}{2} \right) \|\boldsymbol{p}_{(\xi^{(r)})}^{(r)}\|^2. \end{aligned} \tag{31}$$

Again, taking expectation conditioning on the filtration $\mathcal{B}^{(r)}$ and $\xi^{(r)}$, we can upper bound $\frac{1}{\alpha^{(r)}}\left(\mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[\Phi(\boldsymbol{\theta}^{(r+1)})|\mathcal{B}^{(r)}, \xi^{(r)}\right] - \Phi(\boldsymbol{\theta}^{(r)})\right)$ by

$$\begin{aligned} & \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[ \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{P}_{(\xi^{(r)})}^{(r)} - \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\rangle | \mathcal{B}^{(r)}, \xi^{(r)} \right] \\ & \quad + \left( \frac{L\alpha^{(r)}}{2} - \frac{1}{2} \right) \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[ \left\| \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\|^2 \Big| \mathcal{B}^{(r)}, \xi^{(r)} \right], \end{aligned} \tag{32}$$

i.e., the second term on the right hand side of (31) becomes zero because of Fact 1. The first term of (32) can be bounded via the following chain of inequalities:

$$\begin{aligned} & \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[ \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}, \boldsymbol{P}_{(\xi^{(r)})}^{(r)} - \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\rangle \Big| \mathcal{B}^{(r)}, \xi^{(r)} \right] \\ & \overset{(a)}{\leq} \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[ \left\| \boldsymbol{\delta}^{(r)} \right\| \left\| \boldsymbol{P}_{(\xi^{(r)})}^{(r)} - \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\| \;\middle|\; \mathcal{B}^{(r)}, \xi^{(r)} \right] \\ & \overset{(b)}{\leq} \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[ \|\boldsymbol{\delta}^{(r)}\|^2 \mid \mathcal{B}^{(r)}, \xi^{(r)} \right] \leq (\sigma^{(r)})^2 \end{aligned} \tag{33}$$

where (a) is due to the Cauchy-Schwartz inequality, and (b) is a consequence of the non-expansiveness of the proximal operator of convex $h_n(\cdot)$. Taking the total expectation, we have

$$\begin{aligned} & \mathbb{E}\left[\Phi(\boldsymbol{\theta}^{(r+1)})\right] - \mathbb{E}\left[\Phi(\boldsymbol{\theta}^{(r)})\right] \\ & \leq \alpha^{(r)}(\sigma^{(r)})^2 + \left( \frac{L(\alpha^{(r)})^2}{2} - \frac{\alpha^{(r)}}{2} \right) \mathbb{E}\left[ \left\| \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\|^2 \right]. \end{aligned} \tag{34}$$

Summing up the inequality from $t = 0$ to $t = r - 1$,

$$\begin{aligned} & \mathbb{E}\left[\Phi(\boldsymbol{\theta}^{(r)})\right] - \Phi(\boldsymbol{\theta}^{(0)}) \\ & \leq \sum_{t=0}^{r} \alpha^{(t)}(\sigma^{(t)})^2 + \sum_{t=0}^{r} \left( \frac{L(\alpha^{(t)})^2}{2} - \frac{\alpha^{(t)}}{2} \right) \mathbb{E}\left[ \left\| \boldsymbol{p}_{(\xi^{(t)})}^{(t)} \right\|^2 \right]. \end{aligned} \tag{35}$$

Since $\alpha^{(r)} < 1/L$, we have $\frac{L(\alpha^{(r)})^2}{2} - \frac{\alpha^{(r)}}{2} < 0$, therefore,

$$\begin{aligned} & \sum_{t=0}^{r} \left( \frac{\alpha^{(t)}}{2} - \frac{L(\alpha^{(t)})^2}{2} \right) \mathbb{E}\left[ \left\| \boldsymbol{p}_{(\xi^{(t)})}^{(t)} \right\|^2 \right] \\ & \leq \Phi(\boldsymbol{\theta}^{(0)}) - \Phi(\boldsymbol{\theta}^{\star}) + \sum_{t=0}^{r} \alpha^{(t)}(\sigma^{(t)})^2, \end{aligned} \tag{36}$$

such that $\boldsymbol{\theta}^{\star} \in \arg\min_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta})$. Taking $r \to \infty$, and by the assumption that $\sum_{r=0}^{\infty} \alpha^{(r)}(\sigma^{(r)})^2 < \infty$, we can conclude that

$$\liminf_{r\to\infty} \mathbb{E}\left[ \left\| \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\|^2 \right] = 0,$$

using Lemma 1.

To complete the proof, we observe that

$$\begin{aligned} & \frac{1}{2}\mathbb{E}\left[ \left\| \boldsymbol{P}_{(\xi^{(r)})}^{(r)} \right\|^2 \right] \leq \mathbb{E}\left[ \left\| \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\|^2 \right] + \mathbb{E}\left[ \left\| \boldsymbol{p}_{(\xi^{(r)})}^{(r)} - \boldsymbol{P}_{(\xi^{(r)})}^{(r)} \right\|^2 \right] \\ & \leq \mathbb{E}\left[ \left\| \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\|^2 \right] \\ & \quad + \mathbb{E}_{\xi^{(r)}, \mathcal{B}^{(r)}}\left[ \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[ \left\| \boldsymbol{G}_{(\xi^{(r)})}^{(r)} - \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\|^2 \Big| \mathcal{B}^{(r)}, \xi^{(r)} \right] \right] \\ & \leq \mathbb{E}\left[ \left\| \boldsymbol{p}_{(\xi^{(r)})}^{(r)} \right\|^2 \right] + (\sigma^{(r)})^2. \end{aligned} \tag{37}$$

where the last inequality is obtained via applying the nonexpansive property again. Note that both terms on the right hand side converge to zero. Hence, this relationship implies that

$$\liminf_{r\to\infty} \mathbb{E}\left[ \left\| \boldsymbol{P}_{(\xi^{(r)})}^{(r)} \right\|^2 \right] = 0.$$

Note that by our sampling strategy, we have

$$\mathbb{E}\left[ \left\| \boldsymbol{P}_{(\xi^{(r)})}^{(r)} \right\|^2 \right] = \mathbb{E}_{\xi^{(r)}, \mathcal{B}^{(r)}}\left[ \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[ \left\| \boldsymbol{P}_{(\xi^{(r)})}^{(r)} \right\|^2 \Big| \mathcal{B}^{(r)}, \xi^{(r)} \right] \right].$$

However, since $\boldsymbol{P}_{(\xi^{(r)})}^{(r)}$ is independent of the random seed $\zeta^{(r)}$, we have

$$\mathbb{E}\left[\left\|\boldsymbol{P}_{(\xi^{(r)})}^{(r)}\right\|^2\right] = \mathbb{E}_{\mathcal{B}^{(r)}}\left[\mathbb{E}_{\xi^{(r)}}\left[\left\|\boldsymbol{P}_{(\xi^{(r)})}^{(r)}\right\|^2\Big|\mathcal{B}^{(r)}\right]\right] = \mathbb{E}_{\mathcal{B}^{(r)}}\left[\sum_{n=1}^{N}\frac{1}{N}\left\|\boldsymbol{P}_{(n)}^{(r)}\right\|^2\right].$$

This proves the proposition.

## Appendix D
## Proof of Proposition 3

The insight of the proof largely follows the technique for single-block `Adagrad` [55], with some careful modifications to multiple block updates. One will see that the block sampling strategy and the block-wise unbiased gradient estimation are key to apply the proof techniques developed in [55] to our case. To show convergence, let us first consider:

**Lemma 2** *[55] Let $a_0 > 0$, $a_i \geq 0$, $i = 1, \ldots, T$ and $\beta > 1$. Then, we have*

$$\sum_{t=1}^{T}\frac{a_t}{(a_0 + \sum_{i=1}^{t} a_i)^{\beta}} \leq \frac{1}{(\beta-1)a_0^{\beta-1}}.$$

The proof is simple and elegant; see [55, Lemma 4].

**Lemma 3** *[55] Consider a random variable $X$. If $\mathbb{E}[X] < \infty$, then*

$$\mathsf{Pr}(X < \infty) = 1.$$

Let us consider the block-wise again:

$$f(\boldsymbol{\theta}^{(r+1)}) \leq f(\boldsymbol{\theta}^{(r)}) + \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)} \right\rangle + \frac{L_{\xi^{(r)}}^{(r)}}{2}\left\|\boldsymbol{A}_{(\xi^{(r)})}^{(r+1)} - \boldsymbol{A}_{(\xi^{(r)})}^{(r)}\right\|^2. \tag{38}$$

Plugging in our update rule under `AdaCPD`, one can see that

$$\begin{aligned} f(\boldsymbol{\theta}^{(r+1)}) &\leq f(\boldsymbol{\theta}^{(r)}) + \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), -\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)} \right\rangle + \frac{L_{\xi^{(r)}}^{(r)}}{2}\left\|\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right\|^2 \\ &= f(\boldsymbol{\theta}^{(r)}) - \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\rangle \\ &\quad + \left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \left(\nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) - \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right) \right\rangle \\ &\quad + \frac{L_{\xi^{(r)}}^{(r)}}{2}\left\|\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right\|^2. \end{aligned} \tag{39}$$

Taking expectation w.r.t. $\boldsymbol{\zeta}^{(r)}$ (the random seed that is responsible for selecting fibers) conditioning on the filtration $\mathcal{B}^{(r)}$ and the selected block $\xi^{(r)}$, the middle term is zero—since the block stochastic gradient is unbiased [cf. Fact 1]. Hence, we have reached the following

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[f(\boldsymbol{\theta}^{(r+1)})|\mathcal{B}^{(r)}, \xi^{(r)}\right] \leq \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[f(\boldsymbol{\theta}^{(r)})|\mathcal{B}^{(r)}, \xi^{(r)}\right] \\ &- \mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\rangle |\mathcal{B}^{(r)}, \xi^{(r)}\right] \\ &+ \frac{L_{\xi^{(r)}}^{(r)}}{2}\mathbb{E}_{\boldsymbol{\zeta}^{(r)}}\left[\left\|\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right\|^2\Big|\mathcal{B}^{(r)}, \xi^{(r)}\right]. \end{aligned} \tag{40}$$

Taking total expectation on both sides, we have

$$\begin{aligned} \mathbb{E}\left[f(\boldsymbol{\theta}^{(r+1)})\right] &\leq \mathbb{E}\left[f(\boldsymbol{\theta}^{(r)})\right] \\ &\quad - \mathbb{E}\left[\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\rangle\right] \\ &\quad + \mathbb{E}\left[\frac{L_{\xi^{(r)}}^{(r)}}{2}\left\|\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right\|^2\right]. \end{aligned} \tag{41}$$

From the above inequality and the assumption that $L_{(n)}^{(r)}$ is bounded from above by $L$, we can conclude that

$$\begin{aligned} &\sum_{r=0}^{R}\mathbb{E}\left[\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}) \right\rangle\right] \\ &\leq f(\boldsymbol{\theta}^{(0)}) - f(\boldsymbol{\theta}^{(\star)}) + \sum_{r=0}^{R}\frac{L}{2}\mathbb{E}\left[\left\|\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right\|^2\right] \end{aligned}$$

by summing up all the inequalities in (40) from $r = 0$ to $R$.

Taking $R \to \infty$ and observe that:

$$\begin{aligned} &\mathbb{E}\left[\sum_{r=0}^{\infty}\left\|\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right\|^2\right] \\ &= \sum_{r=0}^{\infty}\mathbb{E}\left[\left\|\left(\boldsymbol{\eta}_{(\xi^{(r)})}^{(r+1)} + \boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} - \boldsymbol{\eta}_{(\xi^{(r)})}^{(r+1)}\right) \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right\|^2\right] \\ &= \sum_{r=0}^{\infty}\mathbb{E}\left[\left\|\boldsymbol{\eta}_{(\xi^{(r)})}^{(r+1)} \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right\|^2\right] \\ &\quad + \sum_{r=0}^{\infty}\mathbb{E}\left[\left\|\left(\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} - \boldsymbol{\eta}_{(\xi^{(r)})}^{(r+1)}\right) \circledast \boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right\|^2\right]. \end{aligned} \tag{42}$$

$$\tag{43}$$

Note that we have exchanged the order of the limits and expectations, since the expectation is taking on nonnegative terms. Using Lemma 2, one can easily show the first term above is bounded from above by $\frac{C_1}{2\epsilon\beta^{2\epsilon}}$, where $0 < C_1 < \infty$ is a constant. To see the second term is bounded, observe

$$\begin{aligned} &\mathbb{E}\left[\sum_{r=0}^{\infty}\sum_{i=1}^{J_{\xi^{(r)}}}\sum_{f=1}^{F}\left(\left[\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)}\right]_{i,f}^2 - \left[\boldsymbol{\eta}_{(\xi^{(r)})}^{(r+1)}\right]_{i,f}^2\right)\left[\boldsymbol{G}_{(\xi^{(r)})}^{(r)}\right]_{i,f}^2\right] \\ &= \widetilde{\mathbb{E}}\left[\sum_{r=0}^{\infty}\sum_{f=1}^{F}\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{J_n}\left(\left[\boldsymbol{\eta}_{(n)}^{(r)}\right]_{i,f}^2 - \left[\boldsymbol{\eta}_{(n)}^{(r+1)}\right]_{i,f}^2\right)\left[\boldsymbol{G}_{(n)}^{(r)}\right]_{i,f}^2\right] \\ &\leq \widetilde{\mathbb{E}}\left[\sum_{n=1}^{N}\sum_{i=1}^{J_n}\sum_{f=1}^{F}\frac{1}{N}\max_{r\geq 0}\left[\boldsymbol{G}_{(n)}^{(r)}\right]_{i,f}^2\sum_{r=0}^{\infty}\left(\left[\boldsymbol{\eta}_{(n)}^{(r)}\right]_{i,f}^2 - \left[\boldsymbol{\eta}_{(n)}^{(r+1)}\right]_{i,f}^2\right)\right] \\ &\leq \widetilde{\mathbb{E}}\left[\sum_{n=1}^{N}\sum_{i=1}^{J_n}\sum_{f=1}^{F}\frac{1}{N}\max_{r\geq 0}\left[\boldsymbol{G}_{(n)}^{(r)}\right]_{i,f}^2\left[\boldsymbol{\eta}_{(n)}^{(0)}\right]_{i,f}^2\right] \\ &\leq \sum_{n=1}^{N}\sum_{i=1}^{J_n}\sum_{f=1}^{F}\frac{2}{N}\left[\boldsymbol{\eta}_{(n)}^{(0)}\right]_{i,f}^2\widetilde{\mathbb{E}}\left[\max_{r\geq 0}\left[\left[\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})\right]_{i,f}^2 + \left(\left[\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})\right]_{i,f} - \left[\boldsymbol{G}_{(n)}^{(r)}\right]_{i,f}\right)^2\right]\right], \end{aligned}$$

where $\widetilde{\mathbb{E}}$ means taking expectation w.r.t. all the random variables except for $\xi^{(r)}$ for $r = 0, \ldots, \infty$, and the second inequality is due to the effect of the telescope summation.

Since we have assumed that $\boldsymbol{A}_{(n)}^{(r)}$'s are bounded, the right hand side is bounded from above. Therefore, we have reached the conclusion

$$\widetilde{\mathbb{E}}\left[\sum_{r=0}^{\infty}\left\langle \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)}), \boldsymbol{\eta}_{(\xi^{(r)})}^{(r)} \circledast \nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right\rangle\right] < \infty.$$

Applying Lemma 3, one can see that

$$\mathsf{Pr}\left(\sum_{r=0}^{\infty}\left[\boldsymbol{\eta}_{(\xi^{(r)})}^{(r)}\right]_{i,f}\left[\nabla_{\boldsymbol{A}_{(\xi^{(r)})}} f(\boldsymbol{\theta}^{(r)})\right]_{i,f}^{2} < \infty\right) = 1.$$

Since $\mathsf{Pr}(\xi^{(r)} = n) > 0$, one immediate result is that any $n$ appears infinitely many times in the sequence $r = 0, \ldots, \infty$, according to the second Borel-Cantelli lemma. This leads to

$$\mathsf{Pr}\left(\sum_{j=1}^{\infty}\left[\boldsymbol{\eta}_{(n)}^{(r_j(n))}\right]_{i,f}\left[\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r_j(n))})\right]_{i,f}^{2} < \infty\right) = 1,$$

holds for $n = 1, \ldots, N$, where $r_1(n), \ldots, r_j(n), \ldots$ is the subsequence of $\{r\}$ such that block $n$ is sampled for updating. Hence, with probability one there exists a subsequence $r_1(n), \ldots, r_\infty(n)$ such that at the corresponding iterations block $n$ is sampled for updating. It is not hard to show that

$$\sum_{j=1}^{\infty}\left[\boldsymbol{\eta}_{(n)}^{(r_j(n))}\right]_{i,f} = \infty,$$

by the assumption that $\boldsymbol{A}_{(n)}^{(r)}$ are all bounded. This directly implies that $\sum_{r=1}^{\infty}\left[\boldsymbol{\eta}_{(n)}^{(r)}\right]_{i,f} = \infty$. Together with Lemma 1, we have

$$\mathsf{Pr}\left(\liminf_{r\to\infty}\ [\nabla_{\boldsymbol{A}_{(n)}} f(\boldsymbol{\theta}^{(r)})]_{i,f}^{2} = 0\right) = 1, \quad \forall\ i, f.$$

## REFERENCES


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

[2] 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 Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, 2017.

[3] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[4] B. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," *IEEE Trans. Signal Process.*, vol. 47, no. 1, pp. 187 –200, jan 1999.

[5] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: dynamic tensor analysis," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 374–383.

[6] C. I. Kanatsoulis, X. Fu, N. D. Sidiropoulos, and W.-K. Ma, "Hyperspectral super-resolution: A coupled tensor factorization approach," *IEEE Trans. Signal Process.* to appear, 2018.

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

[8] A. Anandkumar, D. Hsu, and S. M. Kakade, "A method of moments for mixture models and hidden markov models," in *Conference on Learning Theory*, 2012, pp. 33–1.

[9] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 2377–2388, Aug. 2000.

[10] N. D. Sidiropoulos and X.-Q. Liu, "Identifiability results for blind beamforming in incoherent multipath with small delay spread," *IEEE Trans. Signal Process.*, vol. 49, no. 1, pp. 228–236, Jan. 2001.

[11] X. Fu, N. D. Sidiropoulos, J. H. Tranter, and W.-K. Ma, "A factor analysis framework for power spectra separation and emitter localization," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6581–6594, 2015.

[12] C. J. Hillar and L.-H. Lim, "Most tensor problems are np-hard," *Journal of the ACM (JACM)*, vol. 60, no. 6, p. 45, 2013.

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

[14] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on imaging sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.

[15] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 63, no. 20, pp. 5450–5463, 2015.

[16] C. Navasca, L. De Lathauwer, and S. Kindermann, "Swamp reducing technique for tensor decomposition." in *EUSIPCO*, 2008, pp. 1–5.

[17] P. Comon, X. Luciani, and A. L. De Almeida, "Tensor decompositions, alternating least squares and other tales," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 23, no. 7-8, pp. 393–405, 2009.

[18] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, "Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries," in *Proc. ACM SIGKDD 2012*, 2012, pp. 316–324.

[19] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Parcube: Sparse parallelizable tensor decompositions," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 521–536.

[20] D. Mitchell, N. Ye, and H. De Sterck, "Nesterov acceleration of alternating least squares for canonical tensor decomposition," *arXiv preprint arXiv:1810.05846*, 2018.

[21] A. L. Alexander, J. E. Lee, M. Lazar, and A. S. Field, "Diffusion tensor imaging of the brain," *Neurotherapeutics*, vol. 4, no. 3, pp. 316–329, 2007.

[22] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 792–799.

[23] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[24] N. Vervliet and L. De Lathauwer, "A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 2, pp. 284–295, 2016.

[25] A. Beutel, P. P. Talukdar, A. Kumar, C. Faloutsos, E. E. Papalexakis, and E. P. Xing, "Flexifact: Scalable flexible factorization of coupled tensors on hadoop," in *Proc. SIAM SDM 2014*. SIAM, 2014, pp. 109–117.

[26] 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.

[27] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM journal on Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.

[28] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[29] S. Ghadimi and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming," *Mathematical Programming*, vol. 156, no. 1-2, pp. 59–99, 2016.

[30] ——, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[31] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[32] H. Wang and A. Banerjee, "Randomized block coordinate descent for online and stochastic optimization," *arXiv preprint arXiv:1407.0107*, 2014.

[33] Y. Xu and W. Yin, "Block stochastic gradient iteration for convex and nonconvex optimization," *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1686–1716, 2015.

[34] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[35] X. Fu, C. Gao, H.-T. Wai, and K. Huang, "Block-randomized stochastic proximal gradient for constrained low-rank tensor factorization," in submitted to *IEEE ICASSP 2019*, 2019.

[36] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.

[37] S. A. Vorobyov, Y. Rong, N. D. Sidiropoulos, and A. B. Gershman, "Robust iterative fitting of multilinear models," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2678–2689, Aug 2005.

[38] X. Fu, K. Huang, W.-K. Ma, N. Sidiropoulos, and R. Bro, "Joint tensor factorization and outlying slab suppression with applications," *IEEE Trans. Signal Process.*, vol. 63, no. 23, pp. 6315–6328, 2015.

[39] E. E. Papalexakis, U. Kang, C. Faloutsos, N. D. Sidiropoulos, and A. Harpale, "Large Scale Tensor Decompositions: Algorithmic Developments and Applications," *IEEE Data Engineering Bulletin, Special Issue on Social Media and Data Analysis*, vol. 36, no. 3, pp. 59–66, Sep. 2013.

[40] N. Ravindran, N. D. Sidiropoulos, S. Smith, and G. Karypis, "Memory-efficient parallel computation of tensor and matrix products for big tensor decomposition," in *2014 48th Asilomar Conference on Signals, Systems and Computers*, Nov 2014, pp. 581–585.

[41] J. Li, C. Battaglino, I. Perros, J. Sun, and R. Vuduc, "An input-adaptive and in-place approach to dense tensor-times-matrix multiply," in *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2015, pp. 1–12.

[42] N. Kargas, N. D. Sidiropoulos, and X. Fu, "Tensors, learning, and'kolmogorov extension'for finite-alphabet random vectors," *arXiv preprint arXiv:1712.00205*, 2017.

[43] S. Ibrahim and X. Fu, "Stochastic optimization for coupled tensor decomposition with applications in statistical learning," in *Proc. IEEE DSW 2019*, 2019.

[44] N. Kargas and N. D. Sidiropoulos, "Learning mixtures of smooth product distributions: Identifiability and algorithm," *arXiv preprint arXiv:1904.01156*, 2019.

[45] P. A. Traganitis, A. Pags-Zamora, and G. B. Giannakis, "Blind multiclass ensemble classification," *IEEE Trans. Signal Process.*, vol. 66, no. 18, pp. 4737–4752, Sep. 2018.

[46] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 2057–2075, 2014.

[47] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 123–231, 2013.

[48] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the $\ell_1$-ball for learning in high dimensions," in *Proc. the 25th international conference on Machine learning*. ACM, 2008, pp. 272–279.

[49] J. B. Kruskal, "Nonmetric multidimensional scaling: a numerical method," *Psychometrika*, vol. 29, no. 2, pp. 115–129, 1964.

[50] R. Bro and N. D. Sidiropoulos, "Least squares algorithms under unimodality and non-negativity constraints," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 12, no. 4, pp. 223–247, 1998.

[51] H. Robbins and S. Monro, "A stochastic approximation method," in *Herbert Robbins Selected Papers*. Springer, 1985, pp. 102–109.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[53] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[54] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[55] X. Li and F. Orabona, "On the convergence of stochastic gradient descent with adaptive stepsizes," *arXiv preprint arXiv:1805.08114*, 2018.

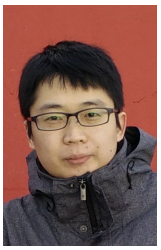
[56] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of adam-type algorithms for non-convex optimization," *arXiv preprint arXiv:1808.02941*, 2018.

[57] K. Huang, N. Sidiropoulos, E. Papalexakis, C. Faloutsos, P. Talukdar, and T. Mitchell, "Principled neuro-functional connectivity discovery," in *Proc. SIAM SDM 2015*, 2015.

[58] X. Fu, W.-K. Ma, K. Huang, and N. D. Sidiropoulos, "Blind separation of quasi-stationary sources: Exploiting convex geometry in covariance domain," *IEEE Trans. Signal Process.*, vol. 63, no. 9, pp. 2306–2320, May 2015.

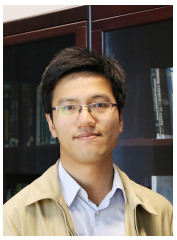
[59] L. D. Lathauwer and J. Castaing, "Blind identification of underdetermined mixtures by simultaneous matrix diagonalization," *IEEE Trans. Signal Process.*, vol. 56, no. 3, pp. 1096 –1105, Mar. 2008.

[60] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *The Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

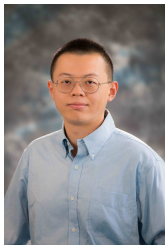
**Xiao Fu** (S'12-M'15) is an Assistant Professor in the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, Oregon, United States. He received his Ph.D. degree in Electronic Engineering from The Chinese University of Hong Kong (CUHK), Hong Kong, in 2014. He was a Postdoctoral Associate in the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, United States, from 2014 to 2017. His research interests include the broad area of signal processing and machine learning. He received a Best Student Paper Award at ICASSP 2014. Two of his co-authored papers received Best Student Paper Awards at IEEE CAMSAP 2015 and IEEE MLSP 2019, respectively. He serves as the treasurer of IEEE Signal Processing Society Oregon Chapter. He also serves as a member of the EURASIP Technical Area Committee in Signal Processing for Multisensor Systems for the term of 2020-2023. He is a member of IEEE.

**Shahana Ibrahim** received her B.Tech. degree in Electronics and Communication Engineering from National Institute of Technology, Calicut, India, in 2012. She had been working as System Validation Engineer at Texas Instruments, Bengaluru, India, from 2012 to 2017. She received her M.S. degree in Electrical Engineering from Oregon State University, Corvallis, Oregon, United States, in 2019. She is currently pursuing her PhD degree in Electrical Engineering at Oregon State University, Corvallis, Oregon, United States. Her research interests are in the broad areas of statistical machine learning and signal processing.



**Hoi-To Wai** (S11M18) received his PhD degree from Arizona State University (ASU) in Electrical Engineering in Fall 2017, B. Eng. (with First Class Honor) and M. Phil. degrees in Electronic Engineering from The Chinese University of Hong Kong (CUHK) in 2010 and 2012, respectively. He is an Assistant Professor in the Department of Systems Engineering & Engineering Management at CUHK. He has held research positions at ASU, UC Davis, Telecom ParisTech, Ecole Polytechnique, LIDS, MIT. Hoi-To's research interests are in the broad area of signal processing, machine learning and distributed optimization, with a focus of their applications to network science. His dissertation has received the 2017's Dean's Dissertation Award from the Ira A. Fulton Schools of Engineering of ASU and he is a recipient of a Best Student Paper Award at ICASSP 2018.

**Cheng Gao** is a Computer Science PhD student at University of Missouri, Columbia, Missouri, United States. He is currently focusing his research on sparse learning and bioinformatics. He earned his Master's degree in Electrical and Computer Engineering from Oregon State University, Corvallis, Oregon, United States, in 2019 with research focus on large-scale tensor decomposition algorithms. He received his Bachelors degree in Automation from Wuhan University of Technology, China in 2016.



**Kejun Huang** received his B.Eng. degree in communication engineering from the Nanjing University of Information Science and Technology, China, in 2010 and his Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2016. He is an assistant professor in the Department of Computer and Information Science and Engineering at the University of Florida, Gainesville. He was a postdoctoral associate in the Department of Electrical and Computer Engineering at the University of Minnesota, Minneapolis, from 2016 to 2018. His research interests include machine learning, signal processing, optimization, and statistics. He is a Member of the IEEE.