Randomshot, a fast nonnegative Tucker factorization approach

Abraham Traoré, Maxime Berar, Alain Rakotomamonjy

Randomshot, a fast nonnegative Tucker factorization approach

Abraham Traoré, Maxime Berar, and Alain Rakotomamonjy

University of Rouen-Normandie,76000 Rouen, France
abraham.traore@etu.univ-rouen.fr
{maxime.berar,alain.rakotomamonjy}@univ-rouen.fr

Abstract. Nonnegative **Tucker** decomposition is a powerful tool for the extraction of nonnegative and meaningful latent components from a positive multidimensional data (or tensor) while preserving the natural multilinear structure. However, as a tensor data has multiple modes, the existing approaches suffer from a high complexity in terms of computation time since they involve intermediate computations that can be time consuming. Besides, most of the existing approaches for nonnegative **Tucker** decomposition, inspired from well-established Nonnegative Matrix Factorization techniques, do not actually address the convergence issue. Most methods with a convergence rate guarantee assume restrictive conditions for their theoretical analyses (e.g. strong convexity, update of all of the block variables at least once within a fixed number of iterations). Thus, there still exists a theoretical vacuum for the convergence rate problem under very mild conditions.
To address these practical (computation time) and theoretical (convergence rate under mild conditions) challenges , we propose a new iterative approach named **Randomshot**, which principle is to update one latent factor per iteration with a theoretical guarantee: we prove, under mild conditions, the convergence of our approach to the set of minimizers with high probability at the rate $\mathcal{O}\left(\frac{1}{k}\right)$, k being the iteration number.
The effectiveness of the approach in terms of both running time and solution quality is proven via experiments on real and synthetic data.

Keywords: nonnegative **Tucker** · convergence rate · proximal gradient

1 Introduction

The recovery of information-rich and task-relevant variables hidden behind data (commonly referred to as latent variables) is a fundamental task that has been extensively studied in machine learning [19],[6],[10]. In many applications, the dataset we are dealing with naturally presents different modes (or dimensions) and thus, can be naturally represented by multidimensional arrays (also called tensors). The recent interest for efficient techniques to deal with such datasets is motivated by the fact that the methodologies that matricize the data and then apply well-known matrix factorization techniques give a flattened view of the data and often cause a loss of the internal structure information [6], [19]. Hence,

to mitigate the extent of this loss, it is more favourable to process a multimodal data set in its own domain, i.e. tensor domain, to obtain a multiple perspective view of the data rather than a flattened one [19].

Tensor decomposition techniques are promising tools for exploratory analysis of multidimensional data in diverse disciplines including signal processing [2], social networks analysis [8]. **In this paper, we focus on a specific tensor factorization, that is, the Tucker decomposition with nonnegativity constraints**. The **Tucker** decomposition, established by *Tucker* [16], is one of the most common decompositions used for tensor analysis with many real applications [17], [12].

Most of the approaches for the nonnegative **Tucker** problem are derived from some existing standard techniques for the *Nonnegative Matrix Factorization NMF* (i.e. they are natural extensions of these techniques) and do not provide any convergence guarantee (i.e. theoretical convergence proof). An iterative approach, inspired from the so-called *Hierarchical Least Squares* has been proposed in [12] . The idea is about employing "local learning rules", in the sense that the rows of the loading matrices are updated sequentially one by one. The main limitation of this method is the lack of convergence guarantee and the update of all of the variables, which can be time consuming. The class of *NMF*-based methods emcompasses other approaches such as those set up in [10], [6], etc.
Alongside the *NMF*-based approaches, some *Block Coordinate Descent*-type algorithms have been proposed. Roughly speaking, the idea is to cyclically update the variables (i.e. **update of all of the block variables** per iteration or in every fixed number of iterations in a deterministic or random order). A method proposed in [17] ensures a convergence rate (to a critical point) and imposes for the convergence analysis, the strong convexity assumption of each block-wise function (i.e. a multivariate function considered as a function of only one variable) for some of the update schemes proposed. This method can be time-consuming due to the update of **all of the block variables** per iteration. Besides, the block-wise strong convexity is not always verified [3]. A second approach established in [18], ensures a convergence rate (to a critical point) with no convexity assumption, but for the convergence analysis, it requires for **all of the block variables** to be updated (in a cyclic order or with a random shuffling) in every fixed number of iterations. Thus, there is still a theoretical vacuum for approaches yielding convergence rates under loose conditions (i.e. without any convexity-type assumption [17], any constraint for all of the variables to be updated per iteration [17] or at least once within a fixed number of iterations [18], any full-rankness [3] assumption).

The method set up in this paper, based on the update per iteration of a single block variable picked randomly, can be classified among the *Randomized Proximal Coordinate Gradient*-type methods [9]: we propose an algorithm for which each update is performed via a single iteration of *Projected Gradient Descent* with the

descent step defined via a carefully chosen minimization problem . Our approach is different from the *Block Coordinate Descent*, which idea is to update all of the variables (per iteration [17], [3] or in every fixed number of iterations [18]). With regards to the existing works for nonnegative **Tucker** decomposition, our contributions are the following ones:

- Proposition of a fast randomized algorithm for the nonnegative **Tucker** problem, named **Randomshot** for which each update stage can be parallelized,
- Theoretical guarantee: we prove with high probability, under fairly loose conditions (i.e. with no convexity assumption, no constraint for all of the variables to be updated and no rank-fullness assumption), the convergence to the set of minimizers at the rate $\mathcal{O}\left(\frac{1}{k}\right)$, k being the iteration number.
- Numerical experiments are performed to prove the efficiency of our approach in terms of running time and solution quality.

2 Notations

A N−order tensor is denoted by a boldface Euler script letter $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. The entries of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times .. \times I_N}$ are denoted by $\boldsymbol{\mathcal{X}}_{i_1,..,i_N}$. The matrices are denoted by bold capital letters (e.g. $\mathbf{A}$). The identity matrix is denoted by $\mathbf{Id}$. The jth row of a matrix $\mathbf{A} \in \mathbb{R}^{J \times L}$ is denoted by $\mathbf{A}_{j,:}$ and the transpose of a matrix $\mathbf{A}$ by $\mathbf{A}^\top$. The Hadamard product (or component-wise product) of two matrices of the same dimensions $\mathbf{A}$ and $\mathbf{B}$ is denoted by $\mathbf{A} \odot B$. Matricization is the process of reordering all the elements of a tensor into a matrix. The mode-n matricization of a tensor $[\boldsymbol{\mathcal{X}}]^{(n)}$ arranges the mode-n fibers to be the columns of the resulting matrix $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times (\prod_{m \neq n} I_m)}$. The mode-$n$ product of a tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{I_n \times J_n}$ denoted by $\boldsymbol{\mathcal{G}} \times_n \mathbf{A}$ yields a tensor of the same order $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times \cdots J_{n-1} \times I_n \times J_{n+1} \cdots \times J_N}$ whose mode-n matricized form is defined by: $\mathbf{B}^{(n)} = \mathbf{A}\mathbf{G}^{(n)}$. For a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, its i_n^{th} subtensor with respect to the mode n is denoted by $\boldsymbol{\mathcal{X}}^n_{i_n} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times 1 \times I_{n+1} \times \cdots \times I_N}$ and defined via the mapping between its n-mode matricization $\left[\boldsymbol{\mathcal{X}}^n_{i_n}\right]^{(n)}$ and the i_n^{th} row of $\mathbf{X}^{(n)}$, i.e. the tensor $\boldsymbol{\mathcal{X}}^n_{i_n}$ is obtained by reshaping the i_n^{th} row of $\mathbf{X}^{(n)}$, with the target shape $(I_1, .., I_{n-1}, 1, I_{n+1}, .., I_N)$ (e.g. the second subtensor with respect to the third mode of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{20 \times 30 \times 40}$ is the tensor $\boldsymbol{\mathcal{X}}^3_2 \in \mathbb{R}^{20 \times 30}$ with $\left(\boldsymbol{\mathcal{X}}^3_2\right)_{i,j} = \boldsymbol{\mathcal{X}}_{i,j,2}, 1 \leq i \leq 20, 1 \leq j \leq 30$). The $N - order$ identity tensor of size R is denoted by $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{R \times \ldots \times R}$ and is defined by: $\boldsymbol{\mathcal{I}}_{i_1,i_2,..,i_N} = 1$ if $i_1 = .. = i_N$ and 0 otherwise.
For writing simplicity, we introduce the following notations:

- The set of integers from n to N (with n and N included) is denoted by $\mathbf{I}_N^n = \{n, ., N\}$. If $n = 1$, it is simply denoted by $\mathbf{I}_N = \{1, .., N\}$.
- We denote by $\mathbf{I}^p_{N \neq n} = \{p, .., n-1, n+1, .., N\}$ the set of integers from p to N with n excluded. If $p = 1$, this set will be simply denoted by $\mathbf{I}_{N \neq n}$.

The product of $\boldsymbol{\mathcal{G}}$ with the matrices $\mathbf{A}^{(m)}$ will also be alternatively expressed by: $\boldsymbol{\mathcal{G}} \underset{m \in \mathbf{I}_N}{\times_m} \mathbf{A}^{(m)} = \boldsymbol{\mathcal{G}} \underset{m \in \mathbf{I}_{n-1}}{\times_m} \mathbf{A}^{(m)} \times_n \mathbf{A}^{(n)} \underset{q \in \mathbf{I}_N^{n+1}}{\times_q} \mathbf{A}^{(q)} = \boldsymbol{\mathcal{G}} \underset{m \in \mathbf{I}_{N \neq n}}{\times_m} \mathbf{A}^{(m)} \times_n \mathbf{A}^{(n)}$

The Frobenius norm of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, denoted by $\|\boldsymbol{\mathcal{X}}\|_F$ is defined by: $\|\boldsymbol{\mathcal{X}}\|_F = \left(\sum_{1 \leq i_n \leq I_n, 1 \leq n \leq N} \boldsymbol{\mathcal{X}}^2_{i_1,\cdots,i_N}\right)^{\frac{1}{2}}$. The same definition holds for matrices.
The ℓ_1 norm for tensor, denoted by $\|\cdots\|_1$ is defined by: $\|\boldsymbol{\mathcal{X}}\|_1 = \sum_{i_1,..,i_N} |\boldsymbol{\mathcal{X}}_{i_1,..,i_N}|$.
The maximum and minimum functions are respectively denoted by max and min.
We denote the positive part of a tensor by $(\boldsymbol{\mathcal{X}})_+$, i.e. $(\boldsymbol{\mathcal{X}})_+ = \max(\boldsymbol{\mathcal{X}}, 0)$ with $\max(., 0)$ applied component-wise. The same definition holds for matrices.
The proximal operator of a function f, denoted $\mathbf{Prox}_f$ is defined by:
$\mathbf{Prox}_f(x) = \arg\min_u \left(f(u) + \frac{1}{2}\|u - x\|_F^2\right)$
The indicator function and the complement of a set $\mathbb{A}$ are denoted by $\mathbf{1}_{\mathbb{A}}$ and $\mathbb{A}^c$.
The notation $\mathbb{R}^I_+$ represent the elements of $\mathbb{R}^I$ with positive components.

3 Fast nonnegative Tucker

3.1 Nonnegative *Tucker* problem

Given a positive tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ (i.e. with nonnegative entries), the nonnegative **Tucker** decomposition aims at the following approximation:

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \underset{m \in \mathbf{I}_N}{\times_m} \mathbf{A}^{(m)}, \boldsymbol{\mathcal{G}} \in \mathbb{R}_+^{J_1 \times \ldots \times J_N}, \mathbf{A}^{(m)} \in \mathbb{R}_+^{I_m \times J_m}$$

The tensor $\boldsymbol{\mathcal{G}}$ is generally called the core tensor and the matrices $\mathbf{A}^{(m)}$ the loading matrices: we keep these denotations for the remainder of the paper. A natural way to tackle this problem is to infer $\boldsymbol{\mathcal{G}}$ and $\mathbf{A}^{(m)}$ in such a way that the discrepancy between $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{G}} \underset{m \in \mathbf{I}_N}{\times_m} \mathbf{A}^{(m)}$ is low. Thus, a relevant problem is:

$$\min_{\boldsymbol{\mathcal{G}} \geq 0, \mathbf{A}^{(1)} \geq 0, \cdots, \mathbf{A}^{(N)} \geq 0} \left\{ f\left(\boldsymbol{\mathcal{G}}, \mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)}\right) \triangleq \frac{1}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \underset{m \in \mathbf{I}_N}{\times_m} \mathbf{A}^{(m)}\|_F^2 \right\} \quad (1)$$

3.2 Randomshot, a randomized nonnegative Tucker approach

Contrary to the existing approaches for nonnegative **Tucker** problem, based either on *NMF* extension [12] or *Block Coordinate Descent* (update of all of the variables per iteration [17] or within a fixed number of iterations [18]), the idea of our approach named **Randomshot**, is based on the update of only one variable per iteration picked randomly while fixing the others at their last updated values. Each update is performed via a single iteration of *Projected Gradient Descent*. To avoid an unnecessary distinction between the core tensor $\boldsymbol{\mathcal{G}}$ and the loading matrices $\mathbf{A}^{(m)}$, we rename the variables $\boldsymbol{\mathcal{G}}, \mathbf{A}^{(1)}, ...\mathbf{A}^{(N)}$ by $x_1, x_2, ..., x_{N+1}$:

$$x_1 = \boldsymbol{\mathcal{G}}, x_{j+1} = \mathbf{A}^{(j)}, 1 \leq j \leq N \quad (2)$$

Besides, we denote by x_j^k the value of the variable x_j at the kth iteration. With these notations and by denoting the derivative of the function f with respect to x_i by $\partial_{x_i} f$, the principle of **Randomshot** is summarised by **Algorithm 1**.

Remark 1. *The equality in **Algorithm 1** is due to the fact that* $\mathbf{Prox}_{\mathbf{1}_{\mathbb{A}}}$ *with* $\mathbb{A}$ *being a convex set corresponds to the projection on* $\mathbb{A}$ *[17].*

Algorithm 1 Randomshot

Inputs: $\boldsymbol{\mathcal{X}}$: tensor of interest, n: splitting mode, $x_1^0 \equiv \boldsymbol{\mathcal{G}}_0$: initial core tensor, $\left\{x_{m+1}^0 \equiv \mathbf{A}_0^{(m)}\right\}_{1\leq m\leq N}$: initial loading matrices.
Output: $\boldsymbol{\mathcal{G}} \in \mathbb{R}_+^{J_1\times\ldots\times J_N}$, $\mathbf{A}^{(m)} \in \mathbb{R}_+^{I_m\times J_m}, 1 \leq m \leq N$
Initialization: $k = 0$

while a predefined stopping criterion is not met **do**
  Choose randomly index $i \in \{1, ..., N+1\}$ a with a probability $p_i > 0$
  Compute optimal step η_k^i via the problem defined by the equation (7).
  Block variable update

$$x_i^{k+1} = \left(x_i^k - \eta_k^i \partial_{\mathbf{x}_i} f(x_1^k, ..., x_{N+1}^k)\right)_+ = \mathbf{Prox}_{\mathbf{1}_{\mathbb{R}_+^I}}\left(x_i^k - \eta_k^i \partial_{\mathbf{x}_i} f(x_1^k, ..., x_{N+1}^k)\right)$$

  with I being the dimension of x_i, $\mathbf{1}_{\mathbb{R}_+^I}$ the indicator function of $\mathbb{R}_+^I$
  $x_j^{k+1} = x_j^k, \forall j \neq i$
  $k \leftarrow k+1$
end while

3.3 Efficient computation of the gradient via parallelization

For the computation of the block-wise derivatives, we propose a divide-and-conquer approach: we split the data tensor into independent sub-parts and determine the gradients via the computation of independent terms involving these sub-parts. For a fixed integer $1 \leq n \leq N$ (in the sequel, n will be referred to as the splitting mode), this is achieved via the following reformulation of the objective function in terms of subtensors drawn with respect to the nth mode (see **Property 1** in the supplementary material):

$$f(\boldsymbol{\mathcal{G}}, \mathbf{A}^{(1)}, .., \mathbf{A}^{(N)}) = \sum_{i_n=1}^{I_n} \frac{1}{2} \|\boldsymbol{\mathcal{X}}_{i_n}^n - \boldsymbol{\mathcal{G}} \underset{m\in\mathbf{I}_{N\neq n}}{\times_m} \mathbf{A}^{(m)} \times_n \mathbf{A}_{i_n,:}^{(n)}\|_F^2 \tag{3}$$

The tensor $\boldsymbol{\mathcal{X}}_{i_n}^n$ is the i_n^{th} subtensor with respect to the splitting mode n.
From the equation (3), each derivative in **Algorithm 1 can be computed via a parallelization process**. To observe this, we distinguish three cases: derivative with respect to the core tensor $\boldsymbol{\mathcal{G}}$, derivative with respect to $\mathbf{A}^{(p)}, p \neq n$, derivative with respect to $\mathbf{A}^{(n)}$ with n being the splitting mode.
First case: derivative with respect to the core. The derivative with respect to the core is given by (see *Property 5* in the supplementary material):

$$\partial_{\boldsymbol{\mathcal{G}}} f\left(\boldsymbol{\mathcal{G}}_k, \mathbf{A}_k^{(1)}, .., \mathbf{A}_k^{(N)}\right) = \sum_{i_n=1}^{I_n} \underbrace{\boldsymbol{\mathcal{R}}_{i_n} \underset{m\in\mathbf{I}_{N\neq n}}{\times_m} \left(\mathbf{A}_k^{(m)}\right)^\top \times_n \left(\left(\mathbf{A}_k^{(n)}\right)_{i_n,:}\right)^\top}_{\theta_{i_n}} \tag{4}$$

with $\boldsymbol{\mathcal{R}}_{i_n} = -\boldsymbol{\mathcal{X}}_{i_n}^n + \boldsymbol{\mathcal{G}}_k \underset{m\in\mathbf{I}_{N\neq n}}{\times_m} \mathbf{A}_k^{(m)} \times_n \left(\mathbf{A}_k^{(n)}\right)_{i_n,:}$
The derivative with respect to the core given by the equation (4) being the sum

of I_n independent terms, it can be computed via a parallelization process.
Second case: derivative with respect to $\mathbf{A}^{(p)}, p \neq n$. The derivative with respect to $\mathbf{A}^{(p)}$ is given by (see *Property 7* in the supplementary material):

$$\partial_{\mathbf{A}^{(p)}} f\left(\boldsymbol{\mathcal{G}}_k, \mathbf{A}_k^{(1)}, .., \mathbf{A}_k^{(N)}\right) = \sum_{i_n=1}^{I_n} \left(-\left(\mathbf{X}_{i_n}^n\right)^{(p)} + \mathbf{A}_k^{(p)} \mathbf{B}_{i_n}^{(p)}\right) \left(\mathbf{B}_{i_n}^{(p)}\right)^\top \tag{5}$$

The matrices $(\mathbf{X}_{i_n}^n)^{(p)}$ and $\mathbf{B}_{i_n}^{(p)}$ represent respectively the mode-p matricized forms of the i_n^{th} subtensor $\boldsymbol{\mathcal{X}}_{i_n}^n$ and the tensor $\boldsymbol{\mathcal{B}}_{i_n}$ is defined by :

$$\boldsymbol{\mathcal{B}}_{i_n} = \boldsymbol{\mathcal{G}}_k \underset{m \in \mathbf{I}_{p-1}}{\times_m} \mathbf{A}_k^{(m)} \times_p \mathbf{Id} \underset{q \in \mathbf{I}_{N \neq n}^{p+1}}{\times_q} \mathbf{A}_k^{(q)} \times_n \left(\mathbf{A}_k^{(n)}\right)_{i_n,:}, \mathbf{Id} \in \mathbb{R}^{J_p \times J_p} : \text{identity}$$

This derivative being the sum of I_n independent terms, its computation can also be parallelized.
Third case: derivative with respect to $\mathbf{A}^{(n)}$. The derivative with respect to $\mathbf{A}^{(n)}$ can be computed via the row-wise stacking of I_n independent terms, that are the derivatives with respect to the rows $\mathbf{A}_{j,:}^{(n)}$, which allows to determine it via a parallelization process. Given the expression of f given by (3), $\partial_{\mathbf{A}_{j,:}^{(n)}} f$ depends on a single subtensor $\boldsymbol{\mathcal{X}}_j^n$ and is given by (see *Property 6* in the supplementary):

$$\partial_{\mathbf{A}_{j,:}^{(n)}} f\left(\boldsymbol{\mathcal{G}}_k, \mathbf{A}_k^{(1)}, ., \mathbf{A}_k^{(N)}\right) = -\left((\mathbf{X}_j^n)^{(n)} - \left(\mathbf{A}_k^{(n)}\right)_{j,:} \mathbf{B}^{(n)}\right) \mathbf{B}^{(n)\top} \tag{6}$$

The matrices $(\mathbf{X}_j^n)^{(n)} \in \mathbb{R}^{1 \times \prod_{k \neq n} I_k}$ and $\mathbf{B}^{(n)}$ respectively represent the mode-n matricized form of the tensors $\boldsymbol{\mathcal{X}}_j^n$ and $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{G}}_k \underset{m \in \mathbf{I}_{N \neq n}}{\times_m} \mathbf{A}_k^{(m)}$.

4 Framework of the theoretical analysis

For the analysis, we consider a framework where a single variable is updated per iteration. The justification of this theoretical analysis with respect to existing analyses performed for the *Randomized Proximal Coordinate Gradient* approaches stems from the fact that either they assume the convexity of the objective function [1] or present different algorithmic settings [9], which make their analyses non applicable to our setting. We recall that our objective in this paper is not to compete with *Randomized Proximal Coordinate Gradient* methods, but to fill the theoretical vaccum for the nonnegative **Tucker** problem. For simplicity purpose, we consider the alternative notations given by the equation (2) as well as the following notations:
A. The objective function evaluated at $\left\{x_1^k, .., x_{N+1}^k\right\}$ (i.e. the value of the variables at the iteration k) is denoted by $f(x^k)$. The same notation holds for any function of the variables $\left\{x_1^k, .., x_{N+1}^k\right\}$, with x_j^k being the value of x_j at the iteration k

B. The objective function evaluated at $\left\{x_1^k,..,x_{i-1}^k,x_i^{k+1},x_{i+1}^k,..,x_{N+1}^k\right\}$ will be denoted by $f(x_1^k,..,x_{i-1}^k,x_i^{k+1},x_{i+1}^k,..,x_{N+1}^k)$.
C. $\Gamma_i = \max_{\mathbb{D}_1\times\mathbb{D}_2\times...\mathbb{D}_{N+1}} \|\partial_{x_i} f(x_1,..,x_{N+1})\|_F$: supremum of $\|\partial_{x_i} f\|_F$ on $\mathbb{D}_1 \times ...\mathbb{D}_{N+1}$ with $\times$ referring to the Cartesian product and the set $\mathbb{D}_j$ defined in the subsection 4.2. Γ_i is well defined since $\|\partial_{x_i} f\|_F$ is continuous and the finite product of compact sets is a compact set.
D. $\Gamma = \max(\Gamma_i, 1 \leq i \leq N+1)$

Our theoretical analysis is mainly based on a careful definition of the descent steps (see section 4.1) as well as some natural assumptions (see section 4.2).

4.1 Definition of the descent step η_k^i at the $(k+1)^{th}$ iteration

By assuming that at the iteration $k+1$, the ith variable has been selected, we introduce the following definition of η_i^k:

$$\eta_k^i = \underset{\eta\in[\frac{\delta_1}{\sqrt{K}},\frac{\delta_2}{\sqrt{K}}]}{\arg\min} \left(\eta - \frac{\delta_1}{\sqrt{K}}\right) \max\left(\Phi\left(\eta\right), \Psi\left(\eta\right), \theta(\eta)\right) \tag{7}$$

$$\Phi\left(\eta\right) = f(x_1^k,..,x_{i-1}^k,\left(x_i^k - \eta\partial_{x_i} f(x^k)\right)_+, x_{i+1}^k,..,x_{N+1}^k) - f(x^k)\left(1 - \frac{f(x^k)}{2}\right) \tag{8}$$

$$\Psi\left(\eta\right) = \|\left(x_i^k - \eta\partial_{x_i} f\left(x^k\right)\right)_+ - x_i^k\|_F^2 - \eta^2\|\partial_{x_i} f(x^k)\|_F^2 \tag{9}$$

$$\begin{aligned}\theta\left(\eta\right) &= \langle\partial_{x_i} f(x^k), (x_i^k - \eta\partial_{x_i} f(x^k))_+ - x_i^k\rangle \\ &+\lambda f\left(x_1^k,..,x_{i-1}^k,\left(x_i^k - \eta\partial_{x_i} f(x^k)\right)_+, x_{i+1}^k,..,x_{N+1}^k\right)\end{aligned} \tag{10}$$

The parameters $\lambda > 0, \delta_2 > \delta_1 > 0$ represent user-defined parameters, K represents the maximum number of iterations. The problem (7) is well defined since it corresponds to the minimization of a continuous function on a compact set, by **Assumption 4** (presented in the section 4.2) and by the fact that all of the factors $\left\{x_1^k,..,x_{N+1}^k\right\}$ are already known at the $(k+1)^{th}$ iteration. Besides, it corresponds to the minimization of a unimodal function for which there are several resolution heuristics such as the *Golden section* method.

4.2 Assumptions

Besides of the definition (7), we consider the following five assumptions:

Assumption 1. *the nth subtensors are uniformly bounded: $\|\boldsymbol{\mathcal{X}}_j^n\|_F \leq \sigma, \forall j$.*

Assumption 2. *we consider the domain $\boldsymbol{\mathcal{G}} \in \mathbb{D}_1, \mathbf{A}^{(m)} \in \mathbb{D}_{m+1}$ with:*

$$\mathbb{D}_1 = \left\{\boldsymbol{\mathcal{G}}_a \in \mathbb{R}^{J_1\times...\times J_N} | \|\boldsymbol{\mathcal{G}}_a\|_F \leq \alpha\right\}, \mathbb{D}_{m+1} = \left\{\mathbf{A}_a^{(m)} \in \mathbb{R}^{I_m\times J_m} | \|\mathbf{A}_a^{(m)}\|_F \leq \alpha\right\}$$

Assumption 3. *the solution of the problem* (7) *is not attained at* $\frac{\delta_1}{\sqrt{K}}$*:*

$$\frac{\delta_1}{\sqrt{K}} < \eta_k^i \leq \frac{\delta_2}{\sqrt{K}} \tag{11}$$

Assumption 4. *non-vanishing gradient*

$$\partial_{x_i} f(x^k) \neq 0 \tag{12}$$

Assumption 5. *minimizer not attained at the initial point*

$$f(x_1^0, .., x_{N+1}^0) = f(x^0) \neq f_{min}, f_{min} : \textit{minimum of } f \textit{ on } \mathbb{D}_1 \times ... \times \mathbb{D}_{N+1} \tag{13}$$

4.3 Theoretical result

Under the definition of the minimization problem given by (7) and the assumptions laid out in the section 4.2, the following inequality holds:
$\forall k > 2\max\left(k_0, \frac{2}{\epsilon}\left(1 + \log(\frac{1}{\rho})\right) - \frac{2}{\Delta_0} + 2\right)$ with $0 < \rho < 1$, $0 < \epsilon < \min(2, \Delta_0)$:

$$1 - \rho \leq \mathbb{P}\left(\Delta_k \leq \frac{2\epsilon}{k - 2k_0} + \frac{\alpha^{2N}\delta_2^2 \Gamma^2}{\lambda k}\right) \tag{14}$$

with log being the logarithmic function, $k_0 = \frac{1}{\log(1+\lambda)} \log\left(\frac{1}{\log(1+\lambda)}\right)$ ($\lambda > 0$ being the parameter that intervenes in the problem (7)), $\Delta_k = f\left(x^k\right) - f_{min} \geq 0$, f_{min} being the minimum value of f on $\mathbb{D}_1 \times \mathbb{D}_2 \times .. \times \mathbb{D}_{N+1}$: f_{min} well defined since f is continuous and a finite product of compact sets is a compact set. The probability $\mathbb{P}$ is the sampling distribution of the block variables in **Algorithm 1**. This result states that the random sequence $x^k = \left\{x_1^k, .., x_{N+1}^k\right\}$ converges to the set of minimizers of the function f (given by (1)) at the rate $\mathcal{O}(\frac{1}{k})$ with a probability at least $1 - \rho$.
To establish this result, we consider a sequence of properties that are Property 1, Property 2, Property 3, Property 4, Property 5. The proof of Property 4 and the Lipschitz character of the block-wise derivatives (used by Property 2) are postponed in the supplementary (since they are perfectly straightforward) as well as some inequalities for Property 1 (to avoid arguments redundancy).

Property 1. *By considering that the* i^{th} *variable has been picked at the* $(k+1)^{th}$ *iteration, the following inequalities hold:*

$$f(x^{k+1}) \leq f(x^k)\left(1 - \frac{f(x^k)}{2}\right) \tag{15}$$

$$\|x_i^{k+1} - x_i^k\|_F^2 \leq (\eta_k^i)^2 \|\partial_{x_i} f(x^k)\|_F^2 \tag{16}$$

$$\langle \partial_{x_i} f(x^k), x_i^{k+1} - x_i^k \rangle \leq -\lambda f(x^{k+1}) \tag{17}$$

The inequality (15) ensures the decreasing of the objective function after each update. The intuition of the inequality (16) is that it yields an estimation of the Lipschitz parameter of the function $x \to \partial_{x_i} f(x_1^k, ., x_{i-1}^k, x, x_{i+1}^k, ., x_{N+1}^k)$. The inequality (17) controls the random character of the gradient by carefully choosing the descent direction $x_i^{k+1} - x_i^k$.

Proof: by definition of η_k^i as a minimizer (equation (7)), we have:
$(\eta_k^i - \frac{\delta_1}{\sqrt{K}}) \max\left(\Phi\left(\eta_k^i\right), \Psi\left(\eta_k^i\right), \theta(\eta_k^i)\right) \leq (\frac{\delta_1}{\sqrt{K}} - \frac{\delta_1}{\sqrt{K}}) \max\left(\Phi\left(\frac{\delta_1}{\sqrt{K}}\right), \Psi\left(\frac{\delta_1}{\sqrt{K}}\right), \theta(\frac{\delta_1}{\sqrt{K}})\right)$
By **Assumption 3**, $\eta_k^i > \frac{\delta_1}{\sqrt{K}}$. Thus, the previous inequality yields:
$\max\left(\Phi\left(\eta_k^i\right), \Psi\left(\eta_k^i\right), \theta(\eta_k^i)\right) \leq 0 \Rightarrow \Phi\left(\eta_k^i\right) \leq 0$ and $\Psi\left(\eta_k^i\right) \leq 0$ and $\theta(\eta_k^i) \leq 0$.
The inequalities (15), (16), (17) stem from $\Phi\left(\eta_k^i\right) \leq 0$, $\Psi\left(\eta_k^i\right) \leq 0$, $\theta(\eta_k^i) \leq 0$.
Since the reasoning is identical for the three inequalities, we perform it only for the first one (see *Property 8* in the supplementary for the remaining ones).
Given that $\Phi\left(\eta_k^i\right) \leq 0$, we have:
$f(x_1^k, .., x_{i-1}^k, \left(x_i^k - \eta_k^i \partial_{x_i} f(x^k)\right)_+, x_{i+1}^k, .., x_{N+1}^k) - f(x^k)(1 - \frac{f(x^k)}{2}) \leq 0.$
Since $x_i^{k+1} = \left(x_i^k - \eta_k^i \partial_{x_i} f(x^k)\right)_+$ by definition, we have:

$$f(x_1^k, .., x_{i-1}^k, x_i^{k+1}, x_{i+1}^k, .., x_{N+1}^k) \leq f(x^k)\left(1 - \frac{(f(x^k)}{2}\right) \tag{18}$$

As $x_j^{k+1} = x_j^k$ for $j \neq i$ by definition, the inequality (18) yields the result:
$f(x^{k+1}) = f(x_1^{k+1}, .., x_{i-1}^{k+1}, x_i^{k+1}, x_{i+1}^{k+1}, .., x_{N+1}^{k+1}) \leq f(x^k)(1 - \frac{(f(x^k))}{2})$ □

Property 2. *The sequence $\Delta_k = f(x^k) - f_{min}$ verifies the inequality:*

$$\Delta_{k+1} \leq \frac{\Delta_k}{1+\lambda} + \frac{\alpha^{2N}\delta_2^2\Gamma^2}{2K(1+\lambda)} \tag{19}$$

Proof: let's consider x_i the variable selected at the $(k+1)^{th}$ iteration. The derivative $x \to \partial_{x_i} f(x_1^k, ., x_{i-1}^k, x, x_{i+1}^k, .., x_{N+1}^k)$ is Lipschitz (by *Property 4* and *Property 6* in the supplementary, each of them using **Assumptions 1 and 2**) with respect to the variable x_i with the parameter α^{2N}. Thus, we have by [13]
$f(x_1^k, .., x_{i-1}^k, x_i^{k+1}, x_{i+1}^k, .., x_{N+1}^k) \leq f(x_1^k, .., x_{i-1}^k, x_i^k, x_{i+1}^k, .., x_{N+1}^k)$
$+\langle \partial_{x_i} f(x^k), x_i^{k+1} - x_i^k \rangle + \frac{\alpha^{2N}}{2}\|x_i^{k+1} - x_i^k\|_F^2.$

Given that $x_j^{k+1} = x_j^k, \forall j \neq i$ (selection of x_i), the last inequality yields:

$$\underbrace{f(x^{k+1}) - f_{min}}_{\Delta_{k+1}} \leq \underbrace{f(x^k) - f_{min}}_{\Delta_k} + \underbrace{\langle \partial_{x_i} f(x^k), x_i^{k+1} - x_i^k \rangle}_{\text{can be bounded by Property 1}} + \underbrace{\frac{\alpha^{2N}}{2}\|x_i^{k+1} - x_i^k\|_F^2}_{\text{can be bounded by Property 1}}.$$

By *Property 1*, **Assumption 3**, the definitions of Γ_i (as the supremum of $\partial_{x_i} f$) and Γ (as $\max(\Gamma_1, .., \Gamma_N)$) respectively, the last inequality yields:
$\Delta_{k+1} \leq \Delta_k - \lambda f(x^{k+1}) + \frac{\alpha^{2N}}{2}\left(\eta_k^i\right)^2 \Gamma_i^2 \leq \Delta_k - \lambda f(x^{k+1}) + \frac{\alpha^{2N}\delta_2^2}{2K}\Gamma^2$
$\Rightarrow \Delta_{k+1} \leq \Delta_k - \lambda f(x^{k+1}) + \lambda f_{min} + \frac{\alpha^{2N}\delta_2^2}{2K}\Gamma^2 = \Delta_k - \lambda\Delta_{k+1} + \frac{\alpha^{2N}\delta_2^2}{2K}\Gamma^2$
$\Rightarrow (1+\lambda)\Delta_{k+1} \leq \Delta_k + \frac{\alpha^{2N}\delta_2^2}{2K}\Gamma^2$: which concludes the proof. □

Property 3. *Let's consider $\mathbb{P}$ the probability distribution with respect to which the block variables are drawn and $\mathbb{E}_{\mathbb{P}}$ the expectation associated. Let's consider the sequence $\Delta_k = f(x^k) - f_{min}$. For $k > \frac{2}{\epsilon}\left(1 + \log(\frac{1}{\rho})\right) - \frac{2}{\Delta_0} + 2$ (log being the logarithmic function) with $0 < \rho < 1, 0 < \epsilon < \min(2, \Delta_0)$. We have :*

$$\mathbb{P}\left(\Delta_k \leq \epsilon\right) \geq 1 - \rho \tag{20}$$

Proof: **Claim 1**: the sequence Δ_k is decreasing.
Justification of **Claim 1**: by *Property 1*, we have:
$\Delta_{k+1} = f(x^{k+1}) - f_{min} \leq f(x^k) - \frac{1}{2}\left(f(x^k)\right)^2 - f_{min} \leq f(x^k) - f_{min} = \Delta_k$
Claim 2: $\mathbb{E}_{\mathbb{P}}\left(\Delta_{k+1}|\Delta_k\right) \leq \Delta_k(1 - \frac{\Delta_k}{2})$, $\mathbb{E}_{\mathbb{P}}(|)$: the conditional expectation [14].
Justification of the **Claim 2**: by *Property 1* we have:
$f(x^{k+1}) \leq f(x^k) - \frac{1}{2}\left(f(x^k)\right)^2 \leq f(x^k) - \frac{\Delta_k^2}{2}$ (due to the fact $0 \leq \Delta_k \leq f(x^k)$)
$\Rightarrow \Delta_{k+1} \leq \Delta_k - \frac{\Delta_k^2}{2}$ (subtraction of f_{min} on both sides of the last inequality)
$\Rightarrow \mathbb{E}_{\mathbb{P}}\left(\Delta_{k+1}|\Delta_k\right) \leq \Delta_k(1 - \frac{\Delta_k}{2})$ (property of the conditional expectation)
Given $\Delta_0 > 0$ (by **Assumption 5**), $\Delta_k \geq 0$ (definition of f_{min}), **Claim 1**, **Claim 2** along with $0 < \epsilon < \Delta_0$ and $\epsilon < 2$, we have (**Theorem 1** in [14]):
$\mathbb{P}\left(\Delta_k \leq \epsilon\right) \geq 1 - \rho$ for $k > \frac{2}{\epsilon}\left(1 + \log(\frac{1}{\rho})\right) - \frac{2}{\Delta_0} + 2$ □

Property 4. *For p fixed and q such that $p + q \leq K$, we can show that:*

$$\Delta_{p+q} \leq \frac{\Delta_p}{(1+\lambda)^q} + \frac{\alpha^{2N}\delta_2^2\Gamma^2}{2K}\sum_{j=1}^{q}\frac{1}{(1+\lambda)^j} \leq \frac{\Delta_p}{(1+\lambda)^q} + \frac{\alpha^{2N}\delta_2^2\Gamma^2}{2K\lambda} \tag{21}$$

Proof: this is perfectly straightforward and done by a simple reasoning by induction on q (for p fixed) using *Property 2.* (see *property 9* in the supplementary material). □

Property 5. *For $\lambda > 0$ and $k > k_0 = \frac{1}{log(1+\lambda)} log\left(\frac{1}{log(1+\lambda)}\right), \frac{1}{(1+\lambda)^k} \leq \frac{1}{k-k_0}$ with* log *being the logarithmic function .*

Proof: let's consider the univariate function $\ell(x) = x - e^{x\log(1+\lambda)} - k_0$ defined on the domain $x > k_0$, e being the exponential function (inverse of the log function). The derivative is given by $\ell^{'}(x) = 1 - (\log(1+\lambda))e^{x \times ln(1+\lambda)}$.
By definition of e and given that $x > k_0$, $\log(1+\lambda) > 0$ (since $\lambda > 0$), we have: $x\log(1+\lambda) \geq \log\left(\frac{1}{\log(1+\lambda)}\right) \Rightarrow (\log(1+\lambda))\, e^{x\log(1+\lambda)} \geq 1 \Rightarrow \ell^{'}(x) \leq 0$: this implies that ℓ is a decreasing function on the domain $x > k_0$.
Thus, for an integer $k > k_0$, we have:
$\ell(k) = k - e^{k\log(1+\lambda)} - k_0 \leq \ell(k_0) = -e^{k_0\log(1+\lambda)} < 0$
$\Rightarrow 0 < k - k_0 < e^{k\log(1+\lambda)} = (1+\lambda)^k$: this concludes the proof. □

Now, we establish the convergence to a minimizer with high probability.
Proof of our result given by the inequality (14)
We consider $\epsilon, k_0, \rho, \Delta_k$ as for (14) and $\Lambda = \max\left(k_0, \frac{2}{\epsilon}\left(1 + \log(\frac{1}{\rho})\right) - \frac{2}{\Delta_0} + 2\right)$.
Without loss of generality, we consider that $\Lambda < \frac{K}{2}$ (we can choose K sufficiently large such that this inequality is verified since Λ is fixed in advance). Let's choose k such that $\Lambda < k \leq \frac{K}{2}$ (such k exists, e.g. $k = \frac{K}{2}$). We have by *Property 4*:
$\Delta_{2k} \leq \frac{\Delta_k}{(1+\lambda)^k} + \frac{\alpha^{2N}\delta_2^2\Gamma^2}{2\lambda K} \leq \frac{\Delta_k}{k-k_0} + \frac{\alpha^{2N}\delta_2^2\Gamma^2}{2\lambda k}$ (since $k \leq K$ and *Property 5*: applicable since $k > k_0$ due to the fact that $k > \Lambda \geq k_0$)
Given the implication $\Delta_k \leq \epsilon \Rightarrow \Delta_{2k} \leq \frac{\epsilon}{k-k_0} + \frac{\alpha^{2N}\delta_2^2\Gamma^2}{2\lambda k}$, we have:

$\mathbb{P}(\Delta_k \leq \epsilon) \leq \mathbb{P}\left(\Delta_{2k} \leq \frac{\epsilon}{k-k_0} + \frac{\alpha^{2N}\delta_2^2\Gamma^2}{2\lambda k}\right)$
Since $k > \frac{2}{\epsilon}\left(1+\log(\frac{1}{\rho})\right) - \frac{2}{\Delta_0} + 2$ (due to the fact that $k > \Lambda$), $0 < \epsilon < \min(\Delta_0, 2), 0 < \rho < 1$, we have by *Property 3*:
$1-\rho \leq \mathbb{P}(\Delta_k \leq \epsilon) \leq \mathbb{P}(\Delta_{2k} \leq \frac{\epsilon}{k-k_0} + \frac{\alpha^{2N}\delta_2^2\Gamma^2}{2\lambda k})$
Let's denote $h = 2k$. Thus, $\forall h > 2\max\left(k_0, \frac{2}{\epsilon}\left(1+\log(\frac{1}{\rho})\right) - \frac{2}{\Delta_0} + 2\right)$,$0 < \epsilon < \min(2, \Delta_0), 0 < \rho < 1$, the following inequality holds:

$$1-\rho \leq \mathbb{P}\left(\Delta_h \leq \frac{2\epsilon}{h-2k_0} + \frac{\alpha^{2N}\delta_2^2\Gamma^2}{\lambda h}\right)$$

Remark 2. *Contrary to the existing analyses for nonnegative* ***Tucker****, our proof does not use any convexity assumption as in [17], does not impose for all of the block variables to be updated per iteration [17] or at least once within a fixed number of iterations as in [18] and does not assume any rank-fullness as in [3]. Besides, we prove the convergence to a minimizer for nonnegative* ***Tucker*** *instead of a critical point as it is the case for the state-of-the-art methods [17],[18], [3].*

5 Numerical experiments

The objective of these experiments is to prove that our approach, based on the update of one variable picked randomly per iteration, yields competitive results both in terms of running time and solution quality compared to some state-of-the-art *NMF*-based and *Block Coordinate Descent*-type (with all of the variables updated per iteration in a predefined order) approaches for the nonnegative **Tucker** problem (**which justifies the choice of the competitors presented in the section 5.1**). This demonstration is performed via some benchmark tasks: the objective is not to establish new techniques for these tasks, but to prove the efficiency of our approach in terms of computation time and solution quality.

5.1 Experimental setting

Randomshot is compared to the following five state-of-the-art methods:
1. TuckerLRA (**Algorithm 1** in [19]): this method is based on the multiplicative update rules for the *NMF* and the replacement of the original tensor by a noise-reduced version.
2. TuckerHALS [19]: this approach is an extension of the *Hierarchical Least Squares* for *NMF*.
3. TuckerSparse [6]: this is an approach inspired from the multiplicative update rules set up for the *NMF* problem.
4. TuckerCCD [17]: this approach proposes a *Block Coordinate Descent* method for the nonnegative **Tucker** problem with a convergence rate (to a critical point). Each block variable update is performed via an extrapolation operation.
5. CPCCD [17]: this is a standard method for the tensor completion problem with nonnegativity constraints and convergence guarantee (to a critical point).

For all of the methods, the code has been done with **Tensorly** [7] except for **TuckerCCD** and **CPCCD** for which we recover the matlab code made available on the author's [17] website for fairness purpose. For **Randomshot**, the derivatives computation is performed via the multiprocessing package of *Python* and the minimisation problems for the steps are solved by the *Golden Section* method. As in [9], the variables are picked according to a uniform distribution on $\{1, .., N+1\}$, N: tensor order and only one variable is updated per iteration. For each experiment, the splitting mode n is fixed to 1, i.e. we consider subtensors with respect to the first mode (changing n has no effect on **Randomshot** output: it simply changes the way to compute the gradients, but not their final values).

5.2 Synthetic experiment

The task considered is the denoising problem of a three-order noisy tensor $\boldsymbol{\mathcal{X}}_n \in \mathbb{R}^{300\times200\times100}$ defined by: $\boldsymbol{\mathcal{X}}_n = \boldsymbol{\mathcal{X}}_{real} + 2 \times \boldsymbol{\mathcal{N}}_{oise}$ with $\boldsymbol{\mathcal{N}}_{oise} = (\mathcal{N})_+$ and $\boldsymbol{\mathcal{X}}_{real} = (\mathcal{T})_+$. The entries of $\mathcal{T}$ and $\mathcal{N}$ are drawn from a standard Gaussian distribution. For fairness purpose, the initial points and the stopping criterion are defined identically: each algorithm is stopped when the relative error is inferior to a predefined threshold or a maximum number of iterations is reached.
The evaluation criteria are the running time and the reconstruction error $\mathcal{R}_e$ defined by:
$\mathcal{R}_e = \|\boldsymbol{\mathcal{X}}_{real} - \boldsymbol{\mathcal{G}}_{out} \times_1 \mathbf{A}^{(1)}_{out} \times_2 \mathbf{A}^{(2)}_{out} \times_3 \mathbf{A}^{(3)}_{out}\|_F^2$ with $\boldsymbol{\mathcal{G}}_{out}, \left\{\mathbf{A}^{(n)}_{out}, 1 \leq n \leq 3\right\}$ being the latent factors inferred from the decomposition of $\boldsymbol{\mathcal{X}}_n$.
For $R \in \{5, 7, 10\}$, $\{\delta_1, \delta_2\} = \left\{10^{-6}, 10^{-5}\right\}$ and for the remaining values of R, δ_1 and δ_2 are fixed to 10^{-7} and 10^{-6}. The value of λ is fixed to 10^{-1} for all R.
Figure 1 portrays two expected behaviors of randomized-coordinate type algorithms, i.e. the decreasing of the objective function and the decreasing of the reconstruction error with respect to the number of epochs (one epoch= m consecutive iterations with m being the number of block variables [4]). From Table 1 and Figure 1, our approach outperforms its competitors with less running time (other numerical results are provided in the section 6.2 of the supplementary).

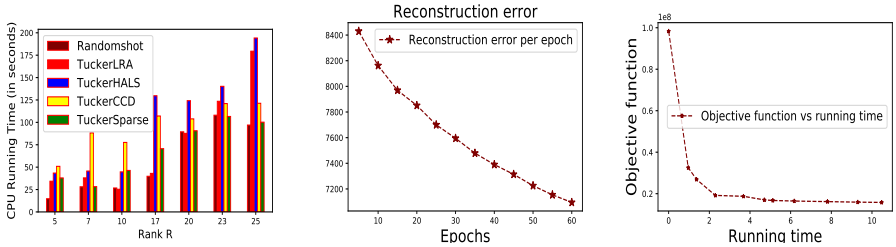

Fig. 1. Left: Running time (mean over 5 runs), Center: Reconstruction error with respect to the number of epochs, Right: Objective function with respect to time

Approximation error: $\mathcal{G} \in \mathbb{R}^{R\times R\times R}$					
R \ **Methods**	**Randomshot**	**TuckerHALS**	**TuckerCCD**	**TuckerLRA**	**TuckerSparse**
5	**2047**	2409	2421	2118	2564
7	**2224**	2418	2422	12919	2856
10	2300	2480	**2180**	31417	3970
17	2467	5974	2425	12402	**1935**
20	**2213**	30550	2427	21930	3412
23	**2163**	4246	2428	7613	2423
25	**1806**	6266	2428	23762	2181

Table 1. Reconstruction error (averaged over 5 independent runs)

5.3 Application to the impainting problem

We compare our approach to its competitors for the inpainting task, which aims to infer missing pixels in an image (we consider two images of size 256×256 and 512×512: see Figure **2**). For this application, we build a training tensor $\mathcal{S}_{train} \in \mathbb{R}^{N_p\times 8\times 8}$ by stacking along the first mode overlapping patches $\boldsymbol{\mathcal{P}}_t$ of size 8×8 (the number of patches being $N_p \in \{32, 64\}$ for the two images) constructed from the input image (with missing pixels). The core tensor as well as the loading matrices $\mathbf{A}_l \in \mathbb{R}^{8\times R}, \mathbf{A}_w \in \mathbb{R}^{8\times R}$, $\mathbf{A}_e \in \mathbb{R}^{N_p\times R}$ (R being an integer whose value has to be defined) are learned from $\mathcal{S}_{train}$ through a nonnegative decomposition. Each patch is then reconstructed by estimating the sparse coefficients through the projection of the non-missing pixels on $\mathbf{A}_l$ and $\mathbf{A}_w$ matrices, i.e. $\boldsymbol{\mathcal{P}}_t^r = \boldsymbol{\mathcal{G}}_t \times_1 \mathbf{A}_l \times_2 \mathbf{A}_w$ with the components of $\boldsymbol{\mathcal{G}}_t$ (defined as in [11]) representing the sparse coefficients. The evaluation criteria are the running time for the decomposition of $\mathcal{S}_{train}$ (i.e. the inference of the four latent factors $\boldsymbol{\mathcal{G}}, \mathbf{A}_l, \mathbf{A}_w, \mathbf{A}_e$) and the PSNR defined by:
$PSNR = 10\log_{10}\left(\frac{m\times n\times 255^2}{\|\mathbf{I}_{real}-\mathbf{I}_{rec}\|_F^2}\right)$ with m, n representing respectively the image sizes and $\mathbf{I}_{real}, \mathbf{I}_{rec}$ the real and the reconstructed images. Each evaluation criterion is averaged over 5 independent selections of the non-missing pixels
Again, we notice as expected, that our algorithm learns better with the number of epochs (one epoch being defined as in [4]). Our approach achieves greater PSNR within less time (Figure **2**) due its parallel and randomized natures. This proves that **Randomshot**, based on the update of a single variable picked randomly per iteration, can be competitive with respect to *NMF*-based and *Block Coordinate Descent*-type approaches for nonnegative **Tucker** problem.

5.4 Application to tensor completion problem

In this section, we consider a completion problem via **Tucker** with the core fixed to the identity tensor $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{R\times R\times R}$. The problem of interest is given by [17]:

$$\min_{\mathbf{A}^{(1)}\in\mathbb{R}^{I_1\times R}\geq 0,..,\mathbf{A}^{(N)}\in\mathbb{R}^{I_N\times R}\geq 0} \|\mathcal{P}_\Omega(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{I}} \underset{m\in\mathbf{I}_N}{\times_m} \mathbf{A}^{(m)})\|_F^2 \tag{22}$$

with $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n\times R}$, $\Omega \subset [I_1] \times ... \times [I_N]$, $[I_n]$ representing the set of consecutive integers from 1 to I_n and $\times$ the Cartesian product. For a given tensor, the

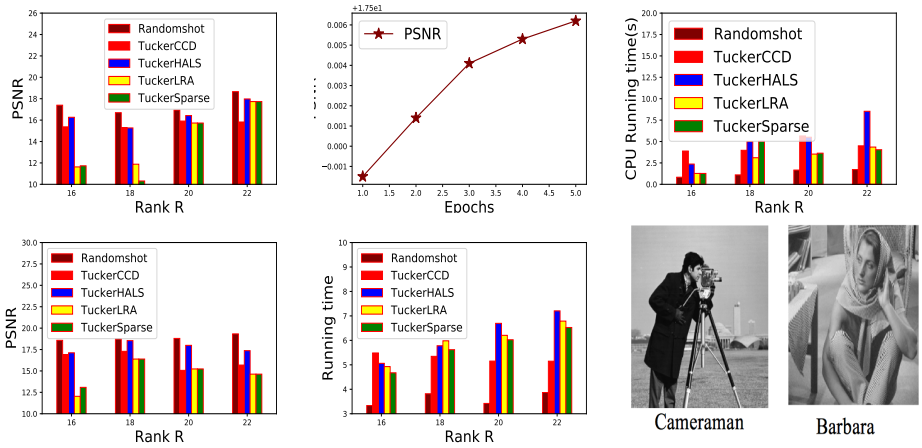

Fig. 2. Top: results for Cameraman (image of size 256×256), Bottom: results for Barbara (image of size 512×512). Top left: PSNR , Top center: PSNR with respect to the number of epochs, Top right: Running time. Bottom left: PSNR, Bottom center: Running time (average of 5 independent runs). For **Cameraman** and **Barbara**, the values of N_p (number of patches) are fixed to 32 and 64. See supplementary for $\delta_1, \delta_2, \lambda$

operator $\mathcal{P}_\Omega$ keeps the entries which indexes belong to Ω and sets to zero the remaining ones. We consider an equivalent formulation of (22) given by [17]:

$$\min_{\boldsymbol{\mathcal{Y}}, \mathbf{A}^{(1)} \geq 0, .., \mathbf{A}^{(N)} \geq 0, \mathcal{P}_\Omega(\boldsymbol{\mathcal{X}}) = \boldsymbol{\mathcal{Y}}} \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{I}} \underset{m \in \mathbf{I}_N}{\times_m} \mathbf{A}^{(m)}\|_F^2 \tag{23}$$

To solve the problem (23), we propose an extension of **Algorithm 1** named **CPRandomshotcomp**. The idea is to replace in **Algorithm 1** the tensor $\boldsymbol{\mathcal{X}}$ by an intermediate variable $\boldsymbol{\mathcal{Y}}_k$ in the definition of the partial gradient and the steps η_k^i (see **section 5** in the supplementary material fore more details). The variable $\boldsymbol{\mathcal{Y}}_k$ is updated just after the **line 5** in **Algorithm 1** via the equality:

$$\boldsymbol{\mathcal{Y}}_{k+1} = \mathcal{P}_\Omega(\boldsymbol{\mathcal{Y}}_k) + \mathcal{P}_{\Omega^c}\left(\boldsymbol{\mathcal{I}} \underset{m \in \mathbf{I}_N}{\times_m} \mathbf{A}_{k+1}^{(m)}\right) \tag{24}$$

with k being to the iteration number. The difference between **CPRandomshotcomp** and **CPCCD** is the update scheme: for the first one, only one variable is updated per iteration. For the completion task, we consider two images, each being a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{500 \times 500 \times 3}$ (see figure 3) provided by [5]. The tensor at hand is split into a training and a test sets: 25% of entries are used for the inference of the latent factors and 75% are used for the test. For the evaluation criteria, we consider besides of the running time, two standard measures for a tensor completion problem that are the **Relative Standard Error** (RSE: the less the better) [15] and the **Tensor Completion Score** (TCS: the less the better) [15]. Each evaluation criteria is averaged over 5 independent train−test splits.

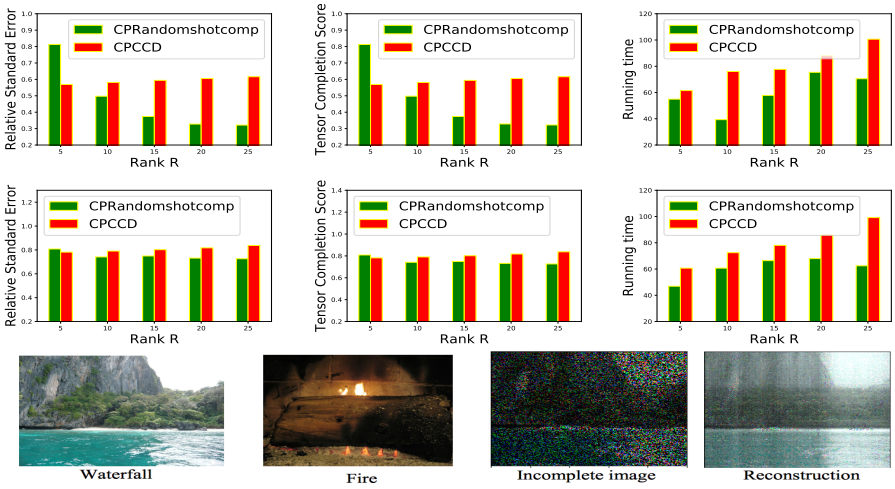

Fig. 3. Top: results for **Waterfall**. Center: results for **Fire**. From left to right: Relative Standard Error, Tensor completion Score and Running time with respect to the rank R. Bottom: images used and example of reconstruction for **Waterfall** by our approach

Our approach **CPRandomshotcomp** performs better than its competitor in terms of both solution quality and running time (see Figure 3). This proves numerically that the random selection of the variables along with the parallelization of the partial gradients can be competitive with respect to cyclic (deterministic) *Block Coordinate Descent* in terms in solution quality within less running time (other numerical results are provided in the section 6.2 of the supplementary).

5.5 Additional experiments

In the section 7 of the supplementary, the robustness of our approach with respect to the step is proven, i.e. we demonstrate that if we fix the step instead of solving the problem (7), our approach still achieves competitive results. Besides, in the section 4 of the supplementary, we demonstrate that the non-vanishing gradient assumption (**Assumption 4** in the section 4.2) is verified in practice.

6 Conclusion

In this paper, we propose a new algorithm for the nonnegative **Tucker** problem for which we establish a convergence rate of $\mathcal{O}\left(\frac{1}{k}\right)$ with high probability and prove that it achieves competitive results compared to some state-of-the-art nonnegative **Tucker** approaches. Besides, we have proven numerically the robustness with respect to the descent steps. Our future work entails the application of our approach principle to other types of decomposition different from Tucker as well the investigation of accelerated versions yielding better convergence rates for the nonnegative **Tucker** problem.

References

1. Alacaoglu, A., Tran-Dinh, Q., Fercoq, O., Cevher, V.: Smooth primal-dual coordinate descent algorithms for nonsmooth convex optimization. arXiv (2017)
2. Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.i.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. Wiley Publishing (2009)
3. Friedlander, M.P., Hatz, K.: Computing non-negative tensor factorizations. Optimization Methods Software **23**(4), 631–647 (2008)
4. Gürbüzbalaban, M., Ozdaglar, A., Parrilo, P.A., Vanli, N.D.: When cyclic coordinate descent outperforms randomized coordinate descent. In: NIPS'17. pp. 7002–7010 (2017)
5. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: Proceedings of the 10th European Conference on Computer Vision: Part I. pp. 304–317 (2008)
6. Kim, Y.D., Choi, S.: Nonnegative tucker decomposition. 2007 IEEE Conference on Computer Vision and Pattern Recognition pp. 1–8 (2007)
7. Kossaifi, J., Panagakis, Y., Pantic, M.: Tensorly: Tensor learning in python. arXiv (2018)
8. Lin, C.Y., Cao, N., Xia Liu, S., Papadimitriou, S., Sun, J., Yan, X.: Smallblue: Social network analysis for expertise search and collective intelligence. ICDE pp. 1483 – 1486 (2009)
9. Lin, Q., Lu, Z., Xiao, L.: An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. SIAM Journal on Optimization **25**, 2244–2273 (2015)
10. Liu, J., Liu, J., Wonka, P., Ye, J.: Sparse non-negative tensor factorization using columnwise coordinate descent. Pattern Recogn. **45**(1), 649–656 (2012)
11. Lu, C., Shi, J., Jia, J.: Online robust dictionary learning. In: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 415–422. CVPR '13 (2013)
12. Phan, A.H., Cichocki, A.: Extended hals algorithm for nonnegative tucker decomposition and its applications for multiway analysis and classification. Neurocomput. **74**(11), 1956–1969 (2011)
13. Reddi, S.J., Hefny, A., Sra, S., Póczós, B., Smola, A.: Stochastic variance reduction for nonconvex optimization. pp. 314–323. ICML'16 (2016)
14. Richtárik, P., Takáuaź, M.: Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Math. Program. **144**(1-2), 1–38 (2014)
15. Song, Q., Ge, H., Caverlee, J., Hu, X.: Tensor completion algorithms in big data analytics. ACM Transactions on Knowledge Discovery from Data **13** (2017)
16. Tucker, L.R.: Implications of factor analysis of three-way matrices for measurement of change. C.W. Harris (Ed.), Problems in Measuring Change, University of Wisconsin Press pp. 122–137 (1963)
17. Xu, Y., Yin, W.: A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM J. Imaging Sciences **6**, 1758–1789 (2013)
18. Xu, Y., Yin, W.: A globally convergent algorithm for nonconvex optimization based on block coordinate update. J. Sci. Comput. **72**, 700–734 (2017)
19. Zhou, G., Cichocki, A., Zhao, Q., Xie, S.: Efficient nonnegative tucker decompositions: Algorithms and uniqueness. IEEE Transactions on Image Processing **24**(12), 4990–5003 (2015)