Nesterov-based Alternating Optimization for Nonnegative Tensor Completion: Algorithm and Parallel Implementation

Georgios Lourakis and Athanasios P. Liavas
School of Electrical and Computer Engineering, Technical University of Crete, Greece
e-mail: {glourakis, aliavas}@isc.tuc.gr

***Abstract*—We consider the problem of nonnegative tensor completion. Our aim is to derive an efficient algorithm that is also suitable for parallel implementation. We adopt the alternating optimization framework and solve each nonnegative matrix completion problem via a Nesterov-type algorithm for smooth convex problems. We describe a parallel implementation of the algorithm and measure the attained speedup in a multi-core computing environment. It turns out that the derived algorithm is an efficient candidate for the solution of very large-scale sparse nonnegative tensor completion problems.**

***Index Terms*—tensors, nonnegative tensor completion, optimal first-order optimization algorithms, parallel algorithms.**

I. Introduction

Tensors have recently gained great popularity due to their ability to model multiway data dependencies [1], [2], [3], [4]. Tensor factorizations into latent factors are very important for numerous tasks, such as feature selection, dimensionality reduction, compression, data visualization, interpretation and completion, and are usually computed as solutions of optimization problems [1], [2]. Alternating Optimization (AO), All-at-Once Optimization (AOO), and Multiplicative Updates (MUs) are among the most commonly used techniques for tensor factorization [2], [5].

The problem of tensor completion arises in many modern applications, such as machine learning, signal processing, and scientific computing, where our aim is to estimate missing values in multi-way data, using only the available elements and structural properties of the data. Matrix completion problems are closely related to recommendation problems, which can be viewed as completing a partially observable matrix whose entries are ratings. Matrix factorization was empirically shown to be a better model than traditional nearest-neighbour based approaches in the Netflix Prize competition [6]. In many real world applications, when nonnegativity constraints are imposed to the factorization/completion, the results have more natural interpretations. In this work, we focus on multiway data and Nonnegative Tensor Completion (NTC). Similar to the matrix case, we employ factorization techniques to provide accurate recommendations. Other approaches for contextual recommendations use context as a means to pre-filter or post-filter the recommendations made [7].

Recent work for constrained tensor factorization/completion includes, among others, [8], [9], [10], and [11]. In [8], several Nonnegative Tensor Factorization (NTF) algorithms and a detailed convergence analysis have been developed. A general framework for joint matrix/tensor factorization/completion has been developed in [9]. In [11], the authors consider constrained matrix/tensor factorization/completion problems. They adopt the AO framework as outer loop and use the Alternating Direction Method of Multipliers (ADMM) for solving the inner constrained optimization problem for one matrix factor conditioned on the rest. The ADMM offers significant flexibility, due to its ability to efficiently handle a wide range of constraints.

Recent tensor applications, such as social network analysis, movie recommendation systems and targeted advertising, need to handle large-scale tensors, making the implementation of efficient parallel algorithms the only viable solution. In [12], two parallel algorithms for unconstrained tensor factorization/completion have been developed and results concerning the speedup attained by their Message Passing Interface (MPI) implementations on a multi-core system have been reported. Related work on parallel algorithms for unconstrained sparse tensor factorization includes [13] and [14]. Recently, the nonnegative case of the tensor factorization problem in a parallel environment has been addressed in [15] and [16].

A. Contribution

In this work, we focus on very large sparse NTC problems. Our aim is to derive an efficient NTC algorithm that is suitable for parallel implementation. We adopt the AO framework and solve each matrix Nonnegative Matrix Completion (NMC) problem via a first-order optimal (Nesterov-type) algorithm for L-smooth convex problems. Then, we describe in detail an MPI implementation of the AO NTC algorithm and measure the speedup attained in a multi-core environment. Throughout this work we focus on third-order tensors, but the generalization of the results in more dimensions is straightforward. We conclude that the proposed algorithm is an efficient candidate for the solution of very large sparse NTC problems.

B. Notation

Vectors, matrices, and tensors are denoted by small, capital, and calligraphic capital bold letters, respectively; for example, $\mathbf{x}$, $\mathbf{X}$, and $\boldsymbol{\mathcal{X}}$. $\mathbb{R}_+^{I\times J\times K}$ denotes the set of $(I \times J \times K)$ real

978-1-5386-3512-4/18/$31.00 ©2018 IEEE

nonnegative tensors, while $\mathbb{R}_+^{I\times J}$ denotes the set of $(I \times J)$ real nonnegative matrices. $\|\cdot\|_F$ denotes the Frobenius norm of the tensor or matrix argument, $\mathbf{I}$ denotes the identity matrix of appropriate dimensions, and $(\mathbf{A})_+$ denotes the projection of matrix $\mathbf{A}$ onto the set of element-wise nonnegative matrices. The outer product of vectors $\mathbf{a} \in \mathbb{R}^{I\times 1}$, $\mathbf{b} \in \mathbb{R}^{J\times 1}$, and $\mathbf{c} \in \mathbb{R}^{K\times 1}$ is the rank-one tensor $\mathbf{a} \circ \mathbf{b} \circ \mathbf{c} \in \mathbb{R}^{I\times J\times K}$ with elements $(\mathbf{a} \circ \mathbf{b} \circ \mathbf{c})(i, j, k) = \mathbf{a}(i)\mathbf{b}(j)\mathbf{c}(k)$. The Kronecker product of compatible matrices $\mathbf{A}$ and $\mathbf{B}$ is denoted as $\mathbf{A}\otimes\mathbf{B}$, the Khatri-Rao (columnwise Kronecker) product is denoted as $\mathbf{A} \odot \mathbf{B}$ and the Hadamard (elementwise) product is denoted as $\mathbf{A} \circledast \mathbf{B}$. Inequality $\mathbf{A} \succeq \mathbf{B}$ means that matrix $\mathbf{A} - \mathbf{B}$ is positive semidefinite. At some points, we use matlab notation; for example, if $\mathbf{A}$ is a matrix, then $\mathbf{A}(i, :)$ denotes its i-th row.

C. Structure

In Section II, we briefly describe the NTC problem. In Section III, we present a Nesterov-type algorithm for the NMC problem. In Section IV, we present the associated AO NTC algorithm and in Section V we describe, in detail, a parallel implementation of the AO NTC algorithm. In Section VI, we use numerical experiments and measure the speedup of the proposed algorithm in a parallel computing environment.

II. Nonnegative tensor Completion

Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ be an incomplete tensor, and $\Omega \subseteq \{1 \dots I\} \times \{1 \dots J\} \times \{1 \dots K\}$ be the set of indices of its known entries. That is $\boldsymbol{\mathcal{X}}(i, j, k)$ is known if $(i, j, k) \in \Omega$. Also, let $\boldsymbol{\mathcal{M}}$ be a tensor with the same size as $\boldsymbol{\mathcal{X}}$, with elements $\boldsymbol{\mathcal{M}}(i, j, k)$ equal to one or zero based on the availability of the corresponding element of $\boldsymbol{\mathcal{X}}$. That is

$$\boldsymbol{\mathcal{M}}(i, j, k) = \begin{cases} 1, & \text{if } (i, j, k) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

We consider the nonnegative tensor completion problem

$$\begin{aligned} &\min f_\Omega(\mathbf{A}, \mathbf{B}, \mathbf{C}) + \tfrac{\lambda}{2}\|\mathbf{A}\|_F^2 + \tfrac{\lambda}{2}\|\mathbf{B}\|_F^2 + \tfrac{\lambda}{2}\|\mathbf{C}\|_F^2 \\ &\text{subject to } \mathbf{A} \geq \mathbf{0}, \mathbf{B} \geq \mathbf{0}, \mathbf{C} \geq \mathbf{0}, \end{aligned} \tag{2}$$

where $\mathbf{A} = [\mathbf{a}_1 \ \cdots \ \mathbf{a}_R] \in \mathbb{R}_+^{I\times R}$, $\mathbf{B} = [\mathbf{b}_1 \ \cdots \ \mathbf{b}_R] \in \mathbb{R}_+^{J\times R}$, $\mathbf{C} = [\mathbf{c}_1 \ \cdots \ \mathbf{c}_R] \in \mathbb{R}_+^{K\times R}$, and

$$f_\Omega(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \frac{1}{2}\left\|\boldsymbol{\mathcal{M}} \circledast (\boldsymbol{\mathcal{X}} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!])\right\|_F^2,$$

with

$$[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r.$$

We can derive matrix-based equivalent expressions of f_Ω as

$$\begin{aligned} f_\Omega(\mathbf{A}, \mathbf{B}, \mathbf{C}) &= \frac{1}{2}\left\|\mathbf{M}_\mathbf{A} \circledast \left(\mathbf{X}_\mathbf{A} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\right)\right\|_F^2 \\ &= \frac{1}{2}\left\|\mathbf{M}_\mathbf{B} \circledast \left(\mathbf{X}_\mathbf{B} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T\right)\right\|_F^2 \\ &= \frac{1}{2}\left\|\mathbf{M}_\mathbf{C} \circledast \left(\mathbf{X}_\mathbf{C} - \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T\right)\right\|_F^2, \end{aligned}$$

where $\mathbf{M}_\mathbf{A}$, $\mathbf{M}_\mathbf{B}$, $\mathbf{M}_\mathbf{C}$ are the matrix unfoldings of $\boldsymbol{\mathcal{M}}$ and $\mathbf{X}_\mathbf{A}$, $\mathbf{X}_\mathbf{B}$, and $\mathbf{X}_\mathbf{C}$ are the matrix unfoldings of $\boldsymbol{\mathcal{X}}$, with respect to the first, second, and third mode, respectively.

III. Nonnegative Matrix Completion

In this section, we consider the NMC problem, whose solution will be the building block for the solution of the AO NTC problem. Let $\mathbf{X} \in \mathbb{R}^{m\times n}$, $\mathbf{A} \in \mathbb{R}^{m\times r}$, $\mathbf{B} \in \mathbb{R}^{n\times r}$. Also, let $\Omega \subseteq \{1 \dots m\} \times \{1 \dots n\}$ be the set of indices of the known entries of $\mathbf{X}$, and $\mathbf{M}$ be a matrix with the same size as $\mathbf{X}$, with elements $\mathbf{M}(i, j)$ equal to one or zero based on the availability of the corresponding element of $\mathbf{X}$. We consider the problem

$$\min_{\mathbf{A}\geq\mathbf{0}} f_\Omega(\mathbf{A}) := \frac{1}{2}\left\|\mathbf{M} \circledast \left(\mathbf{X} - \mathbf{A}\mathbf{B}^T\right)\right\|_F^2 + \frac{\lambda}{2}\|\mathbf{A}\|_F^2. \tag{3}$$

The gradient and the Hessian of f_Ω, at point $\mathbf{A}$, are given by

$$\nabla f_\Omega(\mathbf{A}) = -\left(\mathbf{M} \circledast \mathbf{X} - \mathbf{M} \circledast \mathbf{A}\mathbf{B}^T\right)\mathbf{B} + \lambda\mathbf{A}, \tag{4}$$

and

$$\nabla^2 f_\Omega(\mathbf{A}) = \left(\mathbf{B}^T \otimes \mathbf{I}\right)\mathtt{diag}^2\left(\mathtt{vec}\left(\mathbf{M}\right)\right)\left(\mathbf{B} \otimes \mathbf{I}\right) + \lambda\mathbf{I}. \tag{5}$$

We solve problem (3) with the first-order optimal (Nesterov-type) algorithm presented in Algorithm 1 (for a detailed exposition of first-order optimal optimization methods see [17, Chapter 2]).

Algorithm 1: Nesterov-type algorithm for NMC

Input: $\mathbf{X}, \mathbf{M} \in \mathbb{R}^{m\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times r}$, $\mathbf{A}_* \in \mathbb{R}^{m\times r}$, λ, μ, L

$\mathbf{W} = -(\mathbf{M} \circledast \mathbf{X})\mathbf{B}$
$q = \frac{\mu+\lambda}{L+\lambda}$
$\mathbf{A}_0 = \mathbf{Y}_0 = \mathbf{A}_*$
$\alpha_0 = 1$, $l = 0$
while (1) **do**
  $\nabla f_\Omega(\mathbf{Y}_l) = \mathbf{W} + (\mathbf{M} \circledast \mathbf{Y}_l\mathbf{B}^T)\mathbf{B} + \lambda\mathbf{Y}_l$
  if (term_cond is TRUE) **then**
    break
  else
    $\mathbf{A}_{l+1} = \left(\mathbf{Y}_l - \frac{1}{L+\lambda}\nabla f_\Omega(\mathbf{Y}_l)\right)_+$
    $\alpha_{l+1}^2 = (1 - \alpha_{l+1})\alpha_l^2 + q\alpha_{l+1}$
    $\beta_{l+1} = \frac{\alpha_l(1-\alpha_l)}{\alpha_l^2+\alpha_{l+1}}$
    $\mathbf{Y}_{l+1} = \mathbf{A}_{l+1} + \beta_{l+1}\left(\mathbf{A}_{l+1} - \mathbf{A}_l\right)$
    $l = l + 1$
return $\mathbf{A}_l$.

A crucial part of the algorithm is the assignment of values to parameters μ and L. If we denote the optimal values as μ^* and L^*, then it turns out that $\mu^* + \lambda$ and $L^* + \lambda$ are, respectively, equal to the smallest and the largest eigenvalue of $\nabla^2 f_\Omega$. As the size of the problem grows, the computation of μ^* and L^* becomes very demanding. An approximation is to set $\mu = 0$ and $L = \max(\mathrm{eig}(\mathbf{B}^T\mathbf{B}))$, where $\mathbf{B}^T\mathbf{B}$ is the Hessian of the problem with no missing entries and can be easily computed, especially in the cases of small r. We have observed that, in practice, our choice for μ is very accurate for very sparse problems, while our choice for L is an efficiently computed good upper bound of L^* (in most of our experiments, L is

about two times L^*). For notational convenience, we denote Algorithm 1 as

$$\mathbf{A}_{\text{opt}} = \text{N\_NMC}(\mathbf{X}, \mathbf{M}, \mathbf{B}, \mathbf{A}_*).$$

IV. Nesterov Based AO NTC

In Algorithm 2, we present the Nesterov-based AO NTC. We start from an initial point $(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$ and solve, in a circular manner, MNC problems, based on the previous estimates.

Algorithm 2: Nesterov-based AO NTC

Input: $\boldsymbol{\mathcal{X}}$, Ω, $\mathbf{A}_0 > \mathbf{0}$, $\mathbf{B}_0 > \mathbf{0}$, $\mathbf{C}_0 > \mathbf{0}$.

1 $l = 0$
2 **while** (1) **do**
3 $\quad \mathbf{A}_{l+1} = \text{N\_NMC}(\mathbf{X}_\mathbf{A}, \mathbf{M}_\mathbf{A}, (\mathbf{C}_l \odot \mathbf{B}_l), \mathbf{A}_l)$
4 $\quad \mathbf{B}_{l+1} = \text{N\_NMC}(\mathbf{X}_\mathbf{B}, \mathbf{M}_\mathbf{B}, (\mathbf{C}_l \odot \mathbf{A}_{l+1}), \mathbf{B}_l)$
5 $\quad \mathbf{C}_{l+1} = \text{N\_NMC}(\mathbf{X}_\mathbf{C}, \mathbf{M}_\mathbf{C}, (\mathbf{A}_{l+1} \odot \mathbf{B}_{l+1}), \mathbf{C}_l)$
6 $\quad$ **if** (term_cond is TRUE) **then** break; **endif**
7 $\quad l = l + 1$
8 **return** $\mathbf{A}_l$, $\mathbf{B}_l$, $\mathbf{C}_l$.

The most demanding computations during the update of matrix $\mathbf{A}_l$ via the N_NMC algorithm are

1) in line 1 of Algorithm 1, where we compute

$$\mathbf{W}_\mathbf{A} = (\mathbf{M}_\mathbf{A} \circledast \mathbf{X}_\mathbf{A})(\mathbf{C} \odot \mathbf{B}), \tag{6}$$

2) in line 6 of Algorithm 1, where we compute

$$\mathbf{Z}_\mathbf{A} = (\mathbf{M}_\mathbf{A} \circledast \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T)(\mathbf{C} \odot \mathbf{B}). \tag{7}$$

Of course, analogous quantities must be computed for the updates of $\mathbf{B}_l$ and $\mathbf{C}_l$. In the sequel, we describe efficient ways for the computation of $\mathbf{W}_\mathbf{A}$ and $\mathbf{Z}_\mathbf{A}$.

1) Computation of $\mathbf{W}_\mathbf{A}$: The i-th row of $\mathbf{W}_\mathbf{A}$, for $i = 1, \ldots, I$, is computed as

$$\mathbf{W}_\mathbf{A}(i,:) = \Big(\mathbf{M}_\mathbf{A}(i,:) \circledast \mathbf{X}_\mathbf{A}(i,:)\Big)(\mathbf{C} \odot \mathbf{B}). \tag{8}$$

This computation involves the multiplication of a $(1 \times JK)$ row vector and a $(JK \times R)$ matrix. A direct implementation of this multiplication may be prohibitive since, in many applications, the values of I, J, and K may be of the order of millions or even billions. In order to reduce the computational complexity, we must exploit the sparsity of $\boldsymbol{\mathcal{X}}$.

Let nnz_i be the number of known entries in the i-th horizontal slice of $\boldsymbol{\mathcal{X}}$. Also, let these known entries have indices $(i, j_q, k_q) \in \Omega$, for $q = 1, \ldots, nnz_i$. When we matricize with respect to the first mode, every known element $\boldsymbol{\mathcal{X}}(i, j_q, k_q)$ is mapped to $\mathbf{X}_\mathbf{A}(i, k_q J + j_q)$, for $q = 1, \ldots, nnz_i$. Also, the $(k_q J + j_q)$-th row of the Khatri-Rao product corresponds to $\mathbf{C}(k_q,:) \circledast \mathbf{B}(j_q,:)$. Thus, the computation of the i-th row of $\mathbf{W}_\mathbf{A}$ reduces to

$$\mathbf{W}_\mathbf{A}(i,:) = \sum_{q=1}^{nnz_i} \boldsymbol{\mathcal{X}}(i, j_q, k_q)\, \mathbf{C}(k_q,:) \circledast \mathbf{B}(j_q,:). \tag{9}$$

2) Computation of $\mathbf{Z}_\mathbf{A}$: Following similar arguments, it can be shown that the i-th row of $\mathbf{Z}_\mathbf{A}$ can be efficiently computed as

$$\mathbf{Z}_\mathbf{A}(i,:) = \sum_{q=1}^{nnz_i} \Big(\mathbf{A}(i,:)\big(\mathbf{C}(k_q,:) \circledast \mathbf{B}(j_q,:)\big)^T\Big) \times \big(\mathbf{C}(k_q,:) \circledast \mathbf{B}(j_q,:)\big). \tag{10}$$

3) Terminating condition: We can use various termination conditions for the AO NTC algorithm based, for example, on the maximum number of iterations, or the relative factor change (RFC) defined as

$$\texttt{RFC}_\mathbf{M} := \frac{\|\mathbf{M}_{l+1} - \mathbf{M}_l\|_F}{\|\mathbf{M}_l\|_F}, \text{ for } \mathbf{M} = \mathbf{A}, \mathbf{B}, \mathbf{C}.$$

V. Distributed memory implementation

Motivated by the coarse-grained approach of [18], we consider the implementation of the Nesterov-based AO NTC algorithm on a linear array with N_p processing elements.

1) Partitionings of the matrix factors and the tensor matricizations: We partition the factor matrix $\mathbf{A}_l$ as

$$\mathbf{A}_l = \begin{bmatrix} \left(\mathbf{A}_l^1\right)^T & \cdots & \left(\mathbf{A}_l^{N_p}\right)^T \end{bmatrix}^T, \tag{11}$$

with $\mathbf{A}_l^n \in \mathbb{R}^{\frac{I}{N_p} \times R}$, for $n = 1, \ldots, N_p$. We partition accordingly the matricization $\mathbf{X}_\mathbf{A}$ as

$$\mathbf{X}_\mathbf{A} = \begin{bmatrix} \left(\mathbf{X}_\mathbf{A}^1\right)^T & \cdots & \left(\mathbf{X}_\mathbf{A}^{N_p}\right)^T \end{bmatrix}^T, \tag{12}$$

with $\mathbf{X}_\mathbf{A}^n \in \mathbb{R}^{\frac{I}{N_p} \times JK}$, for $n = 1, \ldots, N_p$.

Analogous partitions are defined for $\mathbf{B}_l$, $\mathbf{X}_\mathbf{B}$, $\mathbf{C}_l$, and $\mathbf{X}_\mathbf{C}$.

2) Data allocation and distributed memory algorithm implementation: We allocate $\mathbf{X}_\mathbf{A}^n$, $\mathbf{X}_\mathbf{B}^n$, $\mathbf{X}_\mathbf{C}^n$ to the n-th processing element, for $n = 1, \ldots, N_p$. Each processing element is responsible for the update of the corresponding part of the matrix factors, that is, the n-th processing element computes $\mathbf{A}_{l+1}^n$, $\mathbf{B}_{l+1}^n$, and $\mathbf{C}_{l+1}^n$, for $n = 1, \ldots, N_p$. An MPI-pseudocode for the distributed-memory implementation of the NTC is given in Algorithm 3.

3) Detailed description of a factor update: The algorithmic steps for the computation of $\mathbf{A}_{l+1}$ are as follows. All processing elements work in parallel. The n-th processing element uses its local data $\mathbf{X}_\mathbf{A}^n$, as well as the whole matrices $\mathbf{B}_l$ and $\mathbf{C}_l$, and computes the n-th block row of matrix $\mathbf{A}_{l+1}$, $\mathbf{A}_{l+1}^n$, via the N_NMC algorithm. Then, each processing element broadcasts the part of the updated factor it has computed to all others, via the MPI statement `MPI_Allgather`. At the end of this step, all processing elements possess $\mathbf{A}_{l+1}$. At this point, we are ready to proceed to the computation of $\mathbf{B}_{l+1}$ and then of $\mathbf{C}_{l+1}$, completing one (outer) iteration of the AO NTC algorithm. The algorithm continues until convergence.

4) Communication complexity: The communication requirements of this implementation consist of one `Allgather` operation per NMC, implying gathering of terms with IF, JF, and KF elements, per outer iteration.

We compute the communication cost during the update of $\mathbf{A}_l$, $\mathcal{C}^{\mathbf{A}}$. We assume that an m-word message is transferred from one process to another with communication cost $t_s + t_w m$, where t_s is the latency, or startup time for the data transfer, and t_w is the word transfer time [19].

All processors learn the updated $\mathbf{A}_{l+1}$ through an `Allgather` operation on its updated parts, each of dimension $\frac{I}{N_p} \times R$, with communication cost [19, §4.2]

$$\mathcal{C}^{\mathbf{A}} = \left(t_s + t_w \frac{IR}{N_p} \right) (N_p - 1) .$$

When we are dealing with large messages, the t_w terms dominate the communication cost. Thus, if we ignore the startup time, the total communication time is

$$\mathcal{C}^{\mathbf{A}} \approx t_w IR.$$

Analogous results hold for the updates of $\mathbf{B}_l$ and $\mathbf{C}_l$.

VI. Numerical Experiments

In this section, we present results obtained from the MPI implementation of the AO NTC. The program is executed on a DELL PowerEdge R820 system with SandyBridge - Intel(R) Xeon(R) CPU E5 − 4650v2 (in total, 16 nodes with 40 cores each at 2.4 Gz) and 512 GB RAM per node. The matrix operations are implemented using routines of the C++ library Eigen [20]. We test the behavior of our implementation using both synthetic and real data.

The real data we used is the MovieLens 10M dataset [21], which contains time-stamped ratings of movies. Binning the time into seven-day-wide bins, results in a tensor of size $71567 \times 65133 \times 171$. The number of samples is 8000044 (99.99% sparsity). In order to distribute the known entries as uniformly as possible across the N_p processors and resolve load imbalance issues, we first perform a random permutation on our data. The tensor with synthetic data is of the same size and sparsity level as the MovieLens 10M dataset, whose true rank is $R = 10$ and the true latent factors have independent and identically distributed elements, uniformly distributed in $[0, 1]$.

The performance metric we compute is the attained speedup. For the computation of the speedup, we measure the execution time for 10 outer iterations; for each NMC problem, we perform 100 (inner) iterations.

In Figure 1, we plot the speedup for the MovieLens 10M dataset, for $N_p = 1, 5, 20, 171$ and rank $R = 10$. In Figure 2, we plot the speedup for the synthetic data for the same number of processors and rank as in Figure 1.

We observe that, in both cases, we attain significant speedup. Comparing the speedup achieved with real and synthetic data, we observe that we attain greater speedup with the synthetic data, We attribute this phenomenon to the more even distribution of the known entries across the processing elements. Thus, more advanced techniques for load imbalance issues should be considered in the future.

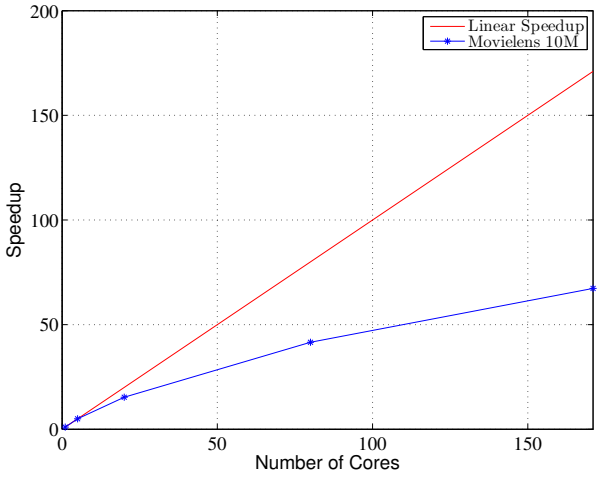

Fig. 1. Speedup achieved for the MovieLens 10M dataset of size $71567 \times 65133 \times 171$ with p cores, for $N_p = 1, 5, 20, 171$.

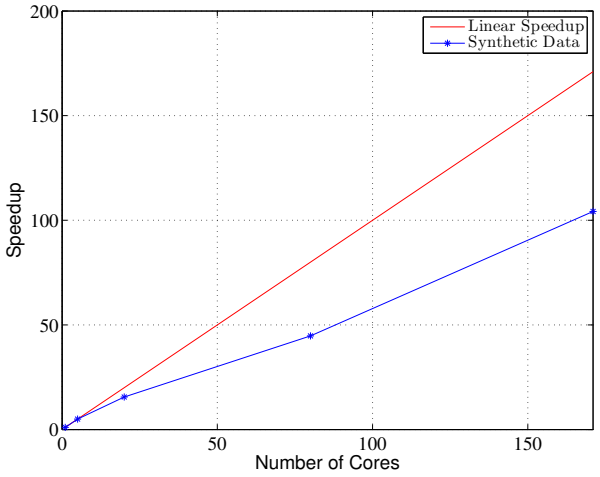

Fig. 2. Speedup achieved for a $71567 \times 65133 \times 171$ tensor with N_p cores, for $N_p = 1, 5, 20, 171$.

In order to check the applicability of our method, we test the accuracy of the predictions. For the MovieLens 10M dataset, we test the completion accuracy by measuring the relative mean squared error of 2×10^6 known ratings with our predictions. For the computation of the accuracy, we use latent factors $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, computed by the AO NTC algorithm, using a terminating condition based on the relative factor change (`RFC` < 0.1); for each NMC problem, we perform 400 (inner) iterations.

For the n-th known rating, for $n = 1, \ldots, 2 \times 10^6$, with indices (i_n, j_n, k_n), we compute our prediction after rounding the quantity $\sum_{r=1}^{R} \mathbf{A}(i_n, :) \circledast \mathbf{B}(j_n, :) \circledast \mathbf{C}(k_n, :)$ to the closest integer. The relative mean squared error we achieved is 0.0033, making our predictions quite accurate.

Algorithm 3: Parallel AO NTC

Input: Processing element n, for $n = 1, \ldots, N_p$, knows $\mathbf{X}^n_{\mathbf{A}}$, $\mathbf{X}^n_{\mathbf{B}}$, $\mathbf{X}^n_{\mathbf{C}}$, Ω. All processing elements know $\mathbf{B}_0$, $\mathbf{C}_0$.

Set $l = 0$

while (terminating condition is FALSE) **do**

  In parallel, for $n = 1, \ldots, N_p$, **do**

    Compute $\mathbf{A}^n_{l+1}$ with N_NMC. Compute line 1 and 6 of N_NMC, for $i = 1, \ldots, I/N_p$ and $(i, j_q, k_q) \in \Omega$, using:

    $\mathbf{W}^n_{\mathbf{A}}(i,:) = \sum_{q=1}^{nnz_i} \boldsymbol{\mathcal{X}}^n \left(\frac{I}{N_p}(n-1) + i, j_q, k_q \right) \mathbf{C}_l(k_q,:) \circledast \mathbf{B}_l(j_q,:)$

    $\mathbf{Z}^n_{\mathbf{A}}(i,:) = \sum_{q=1}^{nnz_i} \left(\mathbf{A}_l \left(\frac{I}{N_p}(n-1) + i, : \right) \left(\mathbf{C}_l(k_q,:) \circledast \mathbf{B}_l(j_q,:) \right)^T \right) \left(\mathbf{C}_l(k_q,:) \circledast \mathbf{B}_l(j_q,:) \right)$

    MPI_Allgather($\mathbf{A}^n_{l+1}$)

  In parallel, for $n = 1, \ldots, N_p$, **do**

    Compute $\mathbf{B}^n_{l+1}$ with N_NMC. Compute line 1 and 6 of N_NMC, for $j = 1, \ldots, J/N_p$ and $(i_q, j, k_q) \in \Omega$, using:

    $\mathbf{W}^n_{\mathbf{B}}(j,:) = \sum_{q=1}^{nnz_j} \boldsymbol{\mathcal{X}}^n \left(i_q, \frac{J}{N_p}(n-1) + j, k_q \right) \mathbf{C}_l(k_q,:) \circledast \mathbf{A}_{l+1}(i_q,:)$

    $\mathbf{Z}^n_{\mathbf{B}}(j,:) = \sum_{q=1}^{nnz_j} \left(\mathbf{B}_l \left(\frac{J}{N_p}(n-1) + j, : \right) \left(\mathbf{C}_l(k_q,:) \circledast \mathbf{A}_{l+1}(i_q,:) \right)^T \right) \left(\mathbf{C}_l(k_q,:) \circledast \mathbf{A}_{l+1}(i_q,:) \right)$

    MPI_Allgather($\mathbf{B}^n_{l+1}$)

  In parallel, for $n = 1, \ldots, N_p$, **do**

    Compute $\mathbf{C}^n_{l+1}$ with N_NMC. Compute line 1 and 6 of N_NMC, for $k = 1, \ldots, K/N_p$ and $(i_q, j_q, k) \in \Omega$, using:

    $\mathbf{W}^n_{\mathbf{C}}(k,:) = \sum_{q=1}^{nnz_k} \boldsymbol{\mathcal{X}}^n \left(i_q, j_q, \frac{K}{N_p}(n-1) + k \right) \mathbf{B}_{l+1}(j_q,:) \circledast \mathbf{A}_{l+1}(i_q,:)$

    $\mathbf{Z}^n_{\mathbf{C}}(k,:) = \sum_{q=1}^{nnz_k} \left(\mathbf{C}_l \left(\frac{K}{N_p}(n-1) + k, : \right) \left(\mathbf{B}_{l+1}(j_q,:) \circledast \mathbf{A}_{l+1}(i_q,:) \right)^T \right) \left(\mathbf{B}_{l+1}(j_q,:) \circledast \mathbf{A}_{l+1}(i_q,:) \right)$

    MPI_Allgather($\mathbf{C}^n_{l+1}$)

  $l = l + 1$

return $\mathbf{A}_l$, $\mathbf{B}_l$, $\mathbf{C}_l$.

VII. Conclusion

We considered the NTC problem. We adopted the AO framework and solved each NMC problem via a Nesterov-type algorithm for smooth convex problems. We described in detail a parallel implementation of the algorithm, which attained significant speedup. Thus, our algorithm is a strong candidate for the solution of very large-scale sparse NTC problems.

References

[1] P. M. Kroonenberg, *Applied Multiway Data Analysis*. Wiley-Interscience, 2008.

[2] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorizations*. Wiley, 2009.

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

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

[5] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *Signal Processing Magazine, IEEE*, vol. 32, no. 2, pp. 145–163, 2015.

[6] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Proceedings of the International Conference on Algorithmic Aspects in Information and Management*, 2008.

[7] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," *Recommender systems handbook*, pp. 217–253, 2011.

[8] 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.

[9] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion." *IEEE Journal on Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 586–600, 2015.

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

[11] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, "A flexible and efficient framework for constrained matrix and tensor factorization," *IEEE Transactions on Signal Processing*, accepted for publication, May 2016.

[12] L. Karlsson, D. Kressner, and A. Uschmajew, "Parallel algorithms for tensor completion in the CP format," *Parallel Computing*, 2015.

[13] S. Smith and G. Karypis, "A medium-grained algorithm for distributed sparse tensor factorization," *30th IEEE International Parallel & Distributed Processing Symposium*, 2016.

[14] O. Kaya and B. Uçar, "Scalable sparse tensor decompositions in distributed memory systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 77.

[15] A. P. Liavas, G. Kostoulas, G. Lourakis, K. Huang, and N. D. Sidiropoulos, "Nesterov-based alternating optimization for nonnegative tensor factorization: Algorithm and parallel implementations," *IEEE Transactions on Signal Processing*, vol. 66, no. 4, pp. 944–953, Feb. 2018.

[16] ——, "Nesterov-based parallel algorithm for large-scale nonnegative tensor factorization," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, USA, March 5-9, 2017*. IEEE, 2017.

[17] Y. Nesterov, *Introductory lectures on convex optimization*. Kluwer Academic Publishers, 2004.

[18] K. Shin and U. Kang, "Distributed methods for high-dimensional and large-scale tensor factorization," in *IEEE International Conference on Data Mining, ICDM*, 2014, pp. 989–994.

[19] A. Grama, G. Karypis, V. Kumar, and A. Gupta, *Introduction to Parallel Computing (2nd Edition)*. Pearson, 2003.

[20] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[21] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, pp. 1–19, Dec. 2015.