OnlineBTD: Streaming Algorithms to Track the Block Term Decomposition of Large Tensors

Ekta Gujral
University of California, Riverside
egujr001@ucr.edu

Evangelos E.Papalexakis
University of California, Riverside
epapalex@cs.ucr.edu

***Abstract*—In data mining, block term tensor decomposition (BTD) is a relatively under-explored but very powerful multi-layer factor analysis method that is ideally suited for modeling for batch processing of data which is either low or multi-linear rank, e.g., EEG/ECG signals, that extract ”rich” structures ($> rank - 1$) from tensor data while still maintaining a lot of the desirable properties of popular tensor decompositions methods such as the interpretability, uniqueness, and etc. These days data, however, is constantly changing which hinders its use for large data. The tracking of the BTD decomposition for the dynamic tensors is a very pivotal and challenging task due to the variability of incoming data and lack of efficient online algorithms in terms of accuracy, time and space.**

In this paper, we fill this gap by proposing an efficient method OnlineBTD to compute the BTD decomposition of streaming tensor datasets containing millions of entries. In terms of effectiveness, our proposed method shows comparable results with the prior work, BTD, while being computationally much more efficient. We evaluate OnlineBTD on six synthetic and three diverse real datasets, indicatively, our proposed method shows $10 - 60\%$ speedup and saves $40 - 70\%$ memory usage over the traditional baseline methods and is capable of handling larger tensor streams for which the classic BTD fails to run. To the best of our knowledge, OnlineBTD is the first approach to track streaming block term decomposition while not only being able to provide stable decompositions but also provides better performance in terms of efficiency and scalability.

I. INTRODUCTION

Tensor decomposition methods are very vital tool in various applications like biomedical imaging [32], social networks [38], and recommender systems [21] to solve various challenging problems. Tensors are higher-order matrix generalization that can reveal more details compared to matrix data, while maintaining most of the computational efficiencies. Each such order of the data is an impression of the same underlying phenomenon e.g, the formation of friendship in social networks or the evolution of communities over time. A main task of the tensor analysis is to decompose the multi-modal data into its latent factors, which is widely known as CANDECOMP/PARAFAC (CP) [3], [16] and Tucker Decomposition [35] in the literature. The CP decomposition has found many applications in machine learning [27], statistical learning [4] and computational neuroscience [30] to understand brain generated signals.

Motivating example: Given the importance of tensor analysis for large-scale data science applications, there has been a growing interest in scaling up these methods to handle large real-world data [39], [12], [33], [13]. However, the CP and Tucker decomposition make a strong assumption on the factors, namely that they are rank-1 (see def. 5). In various application domains, like biomedical images [25], text data [11], it is often controversial whether this assumption is satisfied for all the modes of the problem. The low or multi-linear rank may be a better approximation of the real-world applications [11], [36]. For example, fetal electrocardiogram (fECG) tracking is extremely important for evaluating fetal health and analyzing fetal heart conditions during pregnancy. The electrical activity of the fetal heart is recorded during labor between 38 and 41 weeks of gestation by electrodes (non-invasive) placed on mother's abdomen or an electrode attached to the fetal scalp (invasive, not routinely used) while the cervix is dilated (i.e. during delivery). This signal is produced from a small heart, therefore the amplitude of the signal is low and quite similar to the adult ECG, with a higher heart rate. The recorded signal also has interference from muscle noise, motion artifacts and etc. The fetal and maternal ECG have temporal and spectral overlap. The separation of an accurate fetal electrocardiogram signal from the abdominal mixed signals is complicated but very crucial for many reasons like early detecting fetal distress to avoid painful emergency caesarean delivery ($> 9.6\%$) [15] and reduce brain damage or cerebral palsy in new-born. How to identify and separate fetal ECG over time which can signify a potential issues from noise? How to monitor the dynamic fetal signal behavior? In [1], the Tucker decomposition is used for fECG extraction. Because of the fetal low-amplitude signal compared to the mother's heart signal, the drawback of this method lies in the use of only the fetal periodicity constraints to get rank-1 components. Such limitations can be resolved by Block Term Decomposition (BTD) [9]. BTD helps to tensorize (low multi-linear blocks) abdominal mixed signals into separate subspaces of the mother, fetus and noise signals. The block term decomposition writes a given tensor as a sum of terms with a low multi-linear rank, without rank-1 being required.

Previous Works: The Block Term Decomposition (BTD) unifies the CP and Tucker Decomposition. The BTD framework offers a coherent viewpoint on how to generalize the basic concept of rank from matrices to tensors. The author [19] presented an application of BTD where epileptic seizures pattern was non-stationary, such a trilinear signal model is insufficient. The epilepsy patients suffer from recurring un-

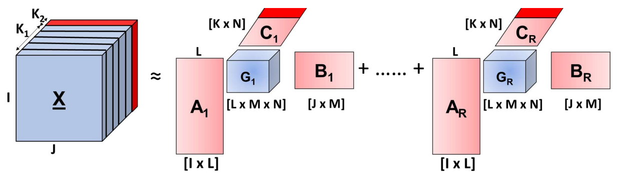


Fig. 1: OnlineBTD procedure. Left: the original tensor is extended with an extra slice (red) in the third mode. Right: the BTD of the tensor is updated by adding a new vector (red) to the factor matrix in the third mode and modifying the existing factor matrices (pink).

provoked seizures, which is a cause and a symptom of abrupt upsurges. They showed the robustness of BTD against these sudden upsurges with various model parameter settings. The author [4] used a higher-order BTD for the first time in fMRI analysis. Through extensive simulation, they demonstrated its effectiveness in handling strong instances of noise. A deterministic block term tensor decomposition (BTD) - based Blind Source Separation [8], [30] method was proposed and offered promising results in analyzing the atrial activity (AA) in short fixed segments of an AF ECG signals. The paper [10] extends the work [8] for better temporal stability. The paper [25] proposed doubly constrained block-term tensor decomposition to extract fetal signals from maternal abdominal signals. Recently, the paper [11] proposed ADMM based constrained BTD method to find structures of communities within social network data. However, these classic methods and applications of BTD are limited to small-sized static dense data ($1K \times 1K \times 100$) and require a large amount of resources (time and space) to process big data. In this era, data is growing very fast and a recipe for handling the limitations is to adapt existing approaches using online techniques. For example, health monitoring data like fECG represents a large number of vibration responses measured over time by many sensors attached at different parts of abdomen. In such applications, a naive approach would be to recompute the decomposition from scratch for each new incoming data. Therefore, this would become impractical and computationally expensive.

Challenges. For many different applications like sensor network monitoring or evolving social network, the data stream is an important model. Streaming decomposition is a challenging task due to the following reasons. First, **accuracy**: high-accuracy (competitive to decomposing the full tensor) using significantly fewer computations than the full decomposition calls for innovation. Second, **speed**: the velocity of incoming data into the system is very high and require real-time execution. Third, **space**: operating on the full ambient space of data, as the tensor is being updated online, leads to increase in space complexity, rendering offline approaches hard to scale, and calling for efficient methods that work on memory spaces which are significantly smaller than the original ambient data dimensions. Lastly, **beyond rank-1 data** [11]: there are certain instances wherein rank-1 decomposition (CP or Tucker) can not be useful, for example, EEG signals [19] needed to be modeled as a sum of exponentially damped sinusoids and allow the retrieval of poles by singular value decomposition. The rank-one terms can only model components of data that are proportional along columns and rows, and it may not be realistic to assume this. Alternatively, it can be handled with blocks of decomposition.

Mitigating Challenges: Motivated by the above challenges, the objective of our work is to develop an algorithm for large multi-aspect or multi-way data analysis that is scalable and amenable to incremental computation for continuously incoming data. In this paper, we propose a method to decompose online or streaming tensors based on BTD decomposition. Our goal is, given an already computed BTD decomposition, to *track* the BTD decomposition of an online tensor, as it receives streaming updates, a) *efficiently*, being much quicker than recomputing the entire decomposition from scratch after each update, and using less memory, and b) *accurately*, obtaining an approximation error which is as similar to the complete tensor decomposition as possible. Answering the above questions, we propose OnlineBTD framework (Figure 1). Our OnlineBTD achieves the best of both worlds in terms of accuracy, speed and memory efficiency: 1) in all cases considered for real and synthetic data (Table V) it is faster than a highly optimized baseline method, achieving up to $10-60\%$ performance improvement; 2) simultaneously, the proposed method is more robust and scalable, since it can execute large problem instances in a reasonable time whereas the baseline fails due to excessive memory consumption (Figure 4).

To our best knowledge, there is no work in the literature that deals with streaming or online Block Term Decomposition. To fill the gap, we propose a scalable and efficient method to find the BTD decomposition for streaming large-scale high-order multi-aspect or temporal data and maintain comparable accuracy. Our contributions can be summarized as:

- **Novel and Efficient Algorithm**: We introduce OnlineBTD, a novel and efficient algorithm for tracking the BTD decompositions of streaming tensors that admits an accelerated implementation described in Section III. We do not limit to three-mode tensors, our algorithm can easily handle higher-order tensor decompositions.
- **Stable Decomposition**: Based on the empirical analysis (Section IV) on synthetic datasets, our algorithm produces similar or more stable decompositions than to existing offline approaches, as well as a better scalability.
- **Real-World Utility**: We performed a case study of applying OnlineBTD on the dataset by ANT-Social Network which consists of > 1 million interactions over 41 days and EEG signal data to understand the human body movement.

II. BACKGROUND

A. Notations

In this Section, we provide the necessary background for notations and tensor operations. Then, we briefly discuss the classical method for BTD for tensor factorization. Table I contains the symbols used throughout the paper.

Symbols	Definition
$\underline{\mathbf{X}}, \mathbf{X}, \mathbf{x}, x$	Tensor, Matrix, Column vector, Scalar
$\mathbf{X}^T, \mathbf{X}^{-1}, \mathbf{X}^\dagger$	Transpose, Inverse, Pseudo-inverse
$\underline{\mathbf{X}}_k$	shorthand for $\underline{\mathbf{X}}(:,:,k)$ (k-th frontal slice of $\underline{\mathbf{X}}$)
$\underline{\mathbf{X}}^{(n)}, \mathbf{X}^{(n)}$	mode-n matricization of $\underline{\mathbf{X}}$, matrix $\mathbf{X}$ at mode-n
$[\mathbf{X}_r], \mathbf{X}_r$	set of R block matrices , r^{th} block matrix $\mathbf{X}$
$\|\mathbf{A}\|_F, \|\mathbf{a}\|_2$	Frobenius norm, ℓ_2 norm
$\circ, \bullet, \circledast, \otimes$	Outer, Inner, Hadmard, Kronecker[20] product
$\odot, \odot_c$	Partition-wise Kronecker product and Column-wise Khatri-Rao product
$bd(\mathbf{X})$ or $blockdiag(\mathbf{X})$	a block diagonal representation of matrix $\mathbf{X}$

TABLE I: Table of symbols and their description

B. Basic definitions

Definition 1: ***Tensor***[20], [28]: A tensor is a higher order generalization of a matrix. An N-mode tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ is the outer product of N vectors, as given in equation 1,

$$\underline{\mathbf{X}} = \mathbf{a}_1 \circ \mathbf{a}_2 \cdots \circ \mathbf{a}_N \tag{1}$$

essentially indexed by N variables i.e. $(\mathbf{a}_1, \mathbf{a}_2 \ldots, \mathbf{a}_N)$. The outer product 3-mode tensor $\underline{\mathbf{X}}$ of vectors $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ can be written as $x_{ijk} = a_i b_j c_k$ for all values of the indices. We refer the interested reader to surveys of tensor applications [20], [28] for more details.

Definition 2: ***Kronecker product***[28] is denoted by symbol $\otimes$ and the Kronecker product of two matrices $\mathbf{A} \in \mathbb{R}^{I \times L}$ and $\mathbf{B} \in \mathbb{R}^{J \times L}$ results in matrix size of $(IJ \times L^2)$ and it is defined as:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \ldots & a_{1L}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \ldots & a_{2L}\mathbf{B} \\ \vdots & \vdots & & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \ldots & a_{IL}\mathbf{B} \end{bmatrix} \tag{2}$$

Definition 3: ***Column-wise Khatri-Rao product***[28] : It is denoted by symbol $\odot_c$ and the column-wise Khatri-Rao product of two matrices $\mathbf{A} \in \mathbb{R}^{I \times L}$ and $\mathbf{B} \in \mathbb{R}^{J \times L}$, $\mathbf{A} \odot_c \mathbf{B} \in \mathbb{R}^{IJ \times L}$ is defined as:

$$\mathbf{A} \odot_c \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \ldots \mathbf{a}_L \otimes \mathbf{b}_L \end{bmatrix} \tag{3}$$

In case of a and b are in vector form, then the Kronecker and column-wise Khatri-Rao products are same, i.e., $\mathbf{a} \otimes \mathbf{b} = \mathbf{a} \odot_c \mathbf{b}$.

Definition 4: ***Partition-wise Kronecker product*** [22], [6] : Let $\mathbf{A} = [\mathbf{A}_1 \quad \mathbf{A}_2 \ldots \mathbf{A}_R] \in \mathbb{R}^{I \times LR}$ and $\mathbf{B} = [\mathbf{B}_1 \quad \mathbf{B}_2 \ldots \mathbf{B}_R] \in \mathbb{R}^{I \times LR}$ are two partitioned matrices. The partition-wise Kronecker product is defined by:

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{A}_1 \otimes \mathbf{B}_1 & \mathbf{A}_2 \otimes \mathbf{B}_2 & \ldots & \mathbf{A}_R \otimes \mathbf{B}_R \end{bmatrix} \tag{4}$$

Definition 5: ***Rank-1*** A N-mode tensor is of rank-1 if it can be strictly decomposed into the outer product of N vectors. Therefore, we add different scaling of a sub-tensor as we introduce more modes when reconstructing the full tensor. A rank-1 of 3-mode tensor can be written as $\underline{\mathbf{X}} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$.
The rank R of a tensor $\underline{\mathbf{X}}$ is defined as the minimum number of rank-1 tensors which are needed to produce $\underline{\mathbf{X}}$ as their sum.

C. Classic BTD Decomposition

De Lathauwer et al. [6], [7], [9] introduced a new type of tensor decomposition that unifies the Tucker and the CP decomposition and referred as Block Term Decomposition (BTD). The BTD of a 3-mode tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, shown in figure 2, is a sum of rank-(L, M, N) terms is a represented as:

$$\underline{\mathbf{X}} \approx \sum_{r=1}^{R} \mathcal{G}_r \bullet_1 \mathbf{A}_r \bullet_2 \mathbf{B}_r \bullet_3 \mathbf{C}_r \tag{5}$$

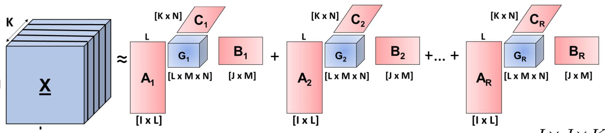


Fig. 2: BTD -(L, M, N) for a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$.

The factor matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ are defined as $\mathbf{A} = [\mathbf{A}_1 \quad \mathbf{A}_2 \ldots \mathbf{A}_R] \in \mathbb{R}^{I \times LR}$, $\mathbf{B} = [\mathbf{B}_1 \quad \mathbf{B}_2 \ldots \mathbf{B}_R] \in \mathbb{R}^{J \times MR}$ and $\mathbf{C} = [\mathbf{C}_1 \quad \mathbf{C}_2 \ldots \mathbf{C}_R] \in \mathbb{R}^{K \times NR}$. The small core tensors $\mathcal{G}_r \in \mathbb{R}^{L \times M \times N}$ are full rank-(L, M, N). If R=1, then Block-term and Tucker decompositions are same. In terms of the standard matrix representations of $\underline{\mathbf{X}}$, (5) can be written as:

$$\begin{aligned} \mathbf{X}^{(1)}_{I \times JK} &\approx \mathbf{A}.(blockdiag(\mathcal{G}^{(1)}_1 \ldots \mathcal{G}^{(1)}_R)).(\mathbf{C} \odot \mathbf{B})^T \\ \mathbf{X}^{(2)}_{J \times IK} &\approx \mathbf{B}.(blockdiag(\mathcal{G}^{(2)}_1 \ldots \mathcal{G}^{(2)}_R)).(\mathbf{C} \odot \mathbf{A})^T \\ \mathbf{X}^{(3)}_{K \times IJ} &\approx \mathbf{C}.(blockdiag(\mathcal{G}^{(3)}_1 \ldots \mathcal{G}^{(3)}_R)).(\mathbf{B} \odot \mathbf{A})^T \end{aligned} \tag{6}$$

In terms of the $(IJK \times 1)$ vector representation of $\underline{\mathbf{X}}$, the decomposition can be written as:

$$\mathbf{x}_{IJK} \approx ((\mathbf{C} \odot \mathbf{B}) \odot \mathbf{A}) \begin{bmatrix} (\mathcal{G}_1)_{LMN} \\ \ldots \\ (\mathcal{G}_R)_{LMN} \end{bmatrix} \tag{7}$$

Algorithm: The direct fitting of Eq. (7) is difficult. A various types of fitting algorithms [37], [20] have been derived and discussed in the literature for tensor decompositions. The most common fitting for tensor decomposition is by using ALS (Alternating Least Squares) and approximation loss can be written as:

$$\mathcal{LS}(\underline{\mathbf{X}}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \underline{\mathcal{G}}) = \underset{\mathbf{A},\mathbf{B},\mathbf{C},\underline{\mathcal{G}}}{\operatorname{argmin}} ||\underline{\mathbf{X}} - \sum_{r=1}^{R} [\![\mathbf{A}_r, \mathbf{B}_r, \mathbf{C}_r, \underline{\mathcal{G}}_r]\!]||^2_F \tag{8}$$

The main idea behind alternating least squares (ALS) is to fix all the factor matrices except for one, then the problem reduces to a linear least squares which can be solved optimally. Here, we used the ALS fitting of BTD Decomposition [9] and to promote reproducibility, we provide clean implementation of the method as described in Algorithm (1) in supplementary section (A.3).

*1) **Uniqueness**:* It was provided in [7] that the BTD is essentially unique up to scaling, permutation and the simultaneous post multiplication of $\mathbf{A}_r$ by a non-singular matrix $\mathbf{F}_r \in \mathbb{R}^{L \times L}$, $\mathbf{A}_r$ by a non-singular matrix $\mathbf{G}_r \in \mathbb{R}^{M \times M}$, and $\mathbf{C}_r$ by a non-singular matrix $\mathbf{H}_r \in \mathbb{R}^{N \times N}$, provided that $\mathcal{G}_r$ is replaced by $\mathcal{G}_r \bullet_1 \mathbf{F}^{-1}_r \bullet_2 \mathbf{G}^{-1}_r \bullet_3 \mathbf{H}^{-1}_r$ and the matrices $[\mathbf{A}_1 \quad \mathbf{A}_2 \ldots \mathbf{A}_R]$ and $[\mathbf{B}_1 \quad \mathbf{B}_2 \ldots \mathbf{B}_R]$ are full column rank.

Proof of Uniqueness Suppose that the conditions a) $N > L + M - 2$ and b) $k^{'}_{\mathbf{A}} + k^{'}_{\mathbf{B}} + k^{'}_{\mathbf{C}} \geq 2R + 2$, hold and that we have an alternative decomposition of $\underline{\mathbf{X}}$, represented by $(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\mathbf{C}}, \overline{\underline{\mathbf{D}}})$, with $k_{\overline{\mathbf{A}}'}$ and $k_{\overline{\mathbf{B}}'}$ maximal under the given dimensionality constraints. $\overline{\mathbf{A}} = \mathbf{A}\Delta\Pi_a\Delta\Delta_a$ and $\overline{\mathbf{B}} = \mathbf{B}\Delta\Pi_b\Delta\Delta_b$, in which Π is a block permutation matrix and Δ is a square non-singular block-diagonal matrix, compatible with the structure of factor matrix.

It suffices to prove it for $\mathbf{A}$. The result for $\mathbf{B}$ and $\mathbf{C}$ can be obtained by switching modes. Let $\omega(\mathbf{x})$ denote the number of nonzero entries of a vector $\mathbf{x}$.

From [7], Lemma 5.2 (i), Upper-bound on $\omega^{'}(\mathbf{x^T\overline{A}})$: The constraint on $k_{\overline{\mathbf{A}}^{'}}$ implies that $k^{'}_{\overline{\mathbf{A}}} \geq k^{'}_{\mathbf{A}}$. Hence, if $\omega^{'}(\mathbf{x^T\overline{A}}) \leq R - k^{'}_{\overline{\mathbf{A}}} + 1$ then

$$\omega^{'}(\mathbf{x^T\overline{A}}) \leq R - k^{'}_{\overline{\mathbf{A}}} + 1 \leq R - k^{'}_{\mathbf{A}} + 1 \leq k^{'}_{\mathbf{B}} + k^{'}_{\mathbf{C}} - (R+1)$$

where the last inequality corresponds to condition (a).

From [7], Lemma 5.2 (ii), Lower-bound on $\omega(\mathbf{x^T\overline{A}})$: After columns are sampled in the column space of the corresponding sub-matrix of $\mathbf{B}$ and $\mathbf{C}$, lower bound is

$$\omega^{'}(\mathbf{x^T\overline{A}}) \geq min(\gamma, k^{'}_{\mathbf{B}}) + min(\gamma, k^{'}_{\mathbf{C}}) - \gamma$$

From [7], Lemma 5.2 (iii), Combination of the two bounds.

$$min(\gamma, k^{'}_{\mathbf{B}}) + min(\gamma, k^{'}_{\mathbf{C}}) - \gamma \leq \omega^{'}(\mathbf{x^T\overline{A}}) \leq k^{'}_{\mathbf{B}} + k^{'}_{\mathbf{C}} - (R+1)$$

If matrix $\mathbf{A}$ or $\mathbf{B}$ or $\mathbf{C}$ is tall and full column rank, then its essential uniqueness implies essential uniqueness of the overall tensor decomposition. We call the decomposition essentially unique when it is subject only to these trivial indeterminacies i.e $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \underline{\mathbf{D}})$ and $(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\mathbf{C}}, \overline{\underline{\mathbf{D}}})$ are equal.

III. Our Approach

In this section, we present our proposed method to track the BTD decomposition of streaming tensor data in an incremental setting. Initially a case of the third-order will be discussed for simplicity of presentation. Further, we expand further to more general conditions, where our proposed algorithm can handle tensors with a higher modes. Formally, the problem that we solve is the following:

Problem 1: **Given** (a) an existing set of BTD decomposition i.e. $\mathbf{A}_{old}, \mathbf{B}_{old}$ and $\mathbf{C}_{old}$ factor matrices, having (L_r, M_r, N_r) latent components, that approximate tensor $\underline{\mathbf{X}}_{old} \in \mathbb{R}^{I\times J\times K_1}$ with Rank R at time t , (b) new incoming slice (s) in form of tensor $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I\times J\times K_2}$ at any time Δt,
Find updates of $\mathbf{A}_{new}, \mathbf{B}_{new}$ and $\mathbf{C}_{new}$ **incrementally** to approximate BTD tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I\times J\times (K_1+K_2)}$ after appending new slice(s) at $t = t_1 + \Delta t$ in last mode while maintaining a comparable accuracy with running the full BTD decompositon on the entire updated tensor $\underline{\mathbf{X}}$.

A. The Principle of OnlineBTD

To address the online BTD problem, our proposed method follows the same alternating update schema as ALS, such that only one factor matrix is updated at one time by fixing all others. Our proposed method is the first work to do an online BTD algorithm, so we need to make some assumptions that will ground the problem. Practically, all other online decomposition (CP/Tucker) based works [39], [13], [26], [12] presume the same thing, either implicitly or explicitly.

Assumptions:

- We assume the last mode of a tensor growing, while the size of the other modes remain unchanged with time.
- The factor matrices $\mathbf{A}_{old}, \mathbf{B}_{old}$ and $\mathbf{C}_{old}$ and core tensor $\mathcal{G}_{old}$ for old data ($\underline{\mathbf{X}}_{old}$) at time stamp t_1 is available.
- The tensor $\underline{\mathbf{X}}$ rank R and Block rank L_r, M_r, N_r are available where $r \in [1, R]$.

1) Update Temporal Mode: Consider first the update of factor $\mathbf{C}_{new}$ obtained after fixing $\mathbf{A}_{old}$, $\mathbf{B}_{old}$ and $\mathcal{G}_{old}$, and solving the corresponding minimization in Equ 9.

$$\mathbf{C}_{new} \leftarrow \underset{\mathbf{C}}{\text{argmin}}\, ||\mathbf{X}^{(3)} - \mathbf{C}.bd(\mathcal{G}).(\mathbf{B}\odot\mathbf{A})^T||_2^F \quad (9)$$

The above Equ. (9) resembles similar to solving problem for CP [20], [39] but it is different and more challenging problem because offline/online CP has rank-1 latent factors (single block) for decomposition without any core tensor. In BTD, we deals with beyond rank-1 and decomposition consists of R number of blocks. Each block is solved with partition-wise Kronecker product (See def. (4)) instead of column-wise Khatri-Rao product (See def. (3)). Further Equ. (9) can be written as:

$$\begin{aligned}\mathbf{C}_{new} &= \underset{\mathbf{C}}{\text{argmin}}\, ||\begin{bmatrix}\mathbf{X}^{(3)}_{old}\\ \mathbf{X}^{(3)}_{new}\end{bmatrix} - \begin{bmatrix}\mathbf{C}_{r_{old}}\\ \widetilde{\mathbf{C}}_r\end{bmatrix}.[\mathcal{G}^{(3)}_r(\mathbf{B}_r\otimes\mathbf{A}_r)^T]||_2^F\\ &= \underset{\mathbf{C}}{\text{argmin}}\, ||\begin{bmatrix}\mathbf{X}^{(3)}_{old} - \mathbf{C}_{r_{old}}.[\mathcal{G}^{(3)}_r(\mathbf{B}_r\otimes\mathbf{A}_r)^T]\\ \mathbf{X}^{(3)}_{new} - \widetilde{\mathbf{C}}_r.[\mathcal{G}^{(3)}_r(\mathbf{B}_r\otimes\mathbf{A}_r)^T]\end{bmatrix}||_2^F\end{aligned} \quad (10)$$

where $r \in [1, R]$ and the above equation presents that the first part is minimized with respect to $\mathbf{C}_{r_{old}}$, since $\mathbf{A}_r$, $\mathbf{B}_r$ and $\mathcal{G}^{(3)}_r$ are fixed as $\mathbf{A}_{r_{old}}$, $\mathbf{B}_{r_{old}}$ and $\mathcal{G}^{(3)}_{r_{old}}$ from the last time stamp. The $\widetilde{\mathbf{C}}_r$ can be obtained after minimizing above equation as :

$$\begin{aligned}\widetilde{\mathbf{C}} &= \mathbf{X}^{(3)}_{new} * [\mathcal{G}_{r_{old}}(\mathbf{B}_{r_{old}}\otimes\mathbf{A}_{r_{old}})^T]^{\dagger} \quad \forall r\in[1,R]\\ &= \mathbf{X}^{(3)}_{new} * (\mathcal{G}_{1_{old}}.(\mathbf{B}_{1_{old}}\otimes\mathbf{A}_{1_{old}})^T \dots \mathcal{G}_{R_{old}}.(\mathbf{B}_{R_{old}}\otimes\mathbf{A}_{R_{old}})^T)^{\dagger}\end{aligned} \quad (11)$$

Observation 1: The classic Matricized Tensor Kronecker product is expensive process because of high computations during Kronecker product, henceforth referred as *classic MTTKRONP*. Thus the *accelerated MTTKRONP* is required. Consider matrix $\mathbf{A} \in \mathbb{R}^{I\times L}$ and $\mathbf{B} \in \mathbb{R}^{J\times M}$ and its Kronecker product $\mathbf{AB} \in \mathbb{R}^{IJ\times LM}$. To speed up the process, we avoid use of def (2) and also avoid explicit allocation of memory by following steps:

- Reshaping matrix into 4-D array : $\mathbf{A} \in \mathbb{R}^{1\times I\times 1\times L}$; $\mathbf{B} \in \mathbb{R}^{J\times 1\times M\times 1}$;
- Multiplies 4-D array $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$ by multiplying corresponding elements; $\mathbf{Kr} = \underline{\mathbf{A}}.*\underline{\mathbf{B}}$
- Reshape $\mathbf{Kr}$ as $\in \mathbb{R}^{IJ\times LM}$ and multiple its transpose with matrix form of given core tensor.

The above method saves $\approx 30\%$ (average) of computational time as provided in Table II, when compared to classic

I=J,Rank	Classic (sec)	Accelerated (sec)	Improvement
$1000, 5$	0.16	0.09	43%
$1500, 15$	0.73	0.54	26%
$5000, 25$	37.55	28.91	23%

TABLE II: Computational gain of accelerated vs classic MTTKRONP.

(product of tensor and output of kron) method available in MATLAB [23].

The factor matrix $\mathbf{C}_{new} \in \mathbb{R}^{(K_1+K_2)\times N}$ is updated by appending the projection $\mathbf{C}_{old} \in \mathbb{R}^{K_1\times N}$ of previous time stamp, to $\widetilde{\mathbf{C}} \in \mathbb{R}^{K_2\times N}$ of new time stamp, i.e.,

$$\mathbf{C}_{new} = \begin{bmatrix}\mathbf{C}_{old}\\ \widetilde{\mathbf{C}}\end{bmatrix} = \begin{bmatrix}\mathbf{C}_{old}\\ \mathbf{X}^{(3)}_{new} * [\mathcal{G}_{r_{old}} * \mathbf{Kr}^T_r]^{\dagger}\end{bmatrix} \tag{12}$$

where $r \in [1, R]$ and the accelerated MTTKRONP is efficiently calculated in linear complexity to the number of nonzeros.

*2) **Update Non-Temporal Mode**:* We update $\mathbf{A}_{new}$ by fixing $\mathbf{B}_{old}$, $\mathcal{G}_{old}$ and $\mathbf{C}_{new}$. We set derivative of the loss $\mathcal{LS}$ w.r.t. $\mathbf{A}$ to zero to find local minima as :

$$\frac{\delta([\mathbf{X}^{(1)}_{new} - \mathbf{A}_{r_{new}}.[bd(\mathcal{G}_{r_{old}}).(\mathbf{C}_{r_{new}} \otimes \mathbf{B}_{r_{old}})]^T]}{\delta A_{r_{new}}} = 0 \tag{13}$$

By solving above equation, we obtain:

$$\begin{aligned}\mathbf{A}_{new} &= (\begin{bmatrix}\mathbf{X}^{(1)}_{old}\\ \mathbf{X}^{(1)}_{new}\end{bmatrix} * [\mathcal{G}_{r_{old}}.(\begin{bmatrix}\mathbf{C}_{r_{old}}\\ \widetilde{\mathbf{C}}_r\end{bmatrix} \otimes \mathbf{B}_{r_{old}})^T]^{\dagger}\\ &= \mathbf{X}^{(1)}_{new} * [\mathcal{G}_{r_{old}}.(\widetilde{\mathbf{C}_r} \otimes \mathbf{B}_{r_{old}})^T]^{\dagger} + \mathbf{X}^{(1)}_{old} * [\mathcal{G}_{r_{old}}.(\mathbf{C}_{r_{old}} \otimes \mathbf{B}_{r_{old}})^T]^{\dagger}\\ &= \mathbf{X}^{(1)}_{new} * \mathbf{G}^{\dagger} + \mathbf{A}_{old}, \quad \mathbf{G} = [\mathcal{G}_{r_{old}}(\widetilde{\mathbf{C}_r} \otimes \mathbf{B}_{r_{old}})^T]\end{aligned} \tag{14}$$

In this way, the factor update equation consists of two parts: the historical part; and the new data part that makes computation fast using *accelerated MTTKRONP*. The $\mathbf{A}_{new} \in \mathbb{R}^{I\times LR}$ is then partitioned into block matrices using corresponding rank (i.e. L) per block.

Similarly, $\mathbf{B}_{new}$ can be updated for mode-2 as :

$$\mathbf{B}_{new} = \mathbf{X}^{(2)}_{new} * \mathbf{G}^{\dagger} + \mathbf{B}_{old}, \quad \mathbf{G} = [\mathcal{G}_{r_{old}}(\widetilde{\mathbf{C}_r} \otimes \mathbf{A}_{r_{new}})^T] \tag{15}$$

The $\mathbf{B}_{new} \in \mathbb{R}^{J\times MR}$ is then partitioned into block matrices using corresponding rank (i.e. M) per block.

*3) **Update core tensor**:* The updated core tensor is obtained from updated factors $\mathbf{A}_{new}$, $\mathbf{B}_{new}$ and $\widetilde{\mathbf{C}}$ using following equation:

$$\begin{aligned}\begin{bmatrix}(\mathcal{G}_1)_{LMN}\\ \cdots\\ (\mathcal{G}_R)_{LMN}\end{bmatrix} &= \begin{bmatrix}(\widetilde{\mathbf{C}}_{1_{new}} \otimes \mathbf{B}_{1_{new}} \otimes \mathbf{A}_{1_{new}})\\ \cdots\\ (\widetilde{\mathbf{C}}_{R_{new}} \otimes \mathbf{B}_{R_{new}} \otimes \mathbf{A}_{R_{new}}\end{bmatrix}^{\dagger} .\underline{\mathbf{X}}_{new}(:)\\ &= \mathbf{H}^{\dagger}.\underline{\mathbf{X}}_{new}(:)\end{aligned} \tag{16}$$

Observation 2: The above pseudo-inverse ($\mathbf{H}^{\dagger}$) or generalized inverse is very expensive in terms of time and space. This can be accelerated using reverse order law [29], [5] and modified LU Factorization (provided in Algorithm 1) and equation can be re-written as:

$$\begin{aligned}&\mathbf{L} = LU_{modified}(\mathbf{H})\\ &\mathcal{G}_{new} = (\mathbf{L}(\mathbf{L}^T\mathbf{L})^{-1}(\mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T\mathbf{H}^T)\underline{\mathbf{X}}_{new}(:)\end{aligned} \tag{17}$$

The main reason to use LU factorization over traditional pseudo-inverse is because a back tracing error is lower[17] as:

$$E_{forward} \leq cond(\mathbf{H}) \times E_{backward} \tag{18}$$

where $E_{forward}$ is forward error, $E_{backward}$ backward error and $cond$ represents condition number of matrix. Since condition number does not depend on an algorithm used to solve given problem, so choosing LU algorithm gives smaller backward error and it will lead to lower forward error. Also, in this we are dealing with triangular matrices (L and U), which can be solved directly by forward and backward substitution without using the Gaussian elimination (Gauss - Jordan) process[18] used in pseudo-inverse. The $\mathcal{G}_{new} \in \mathbb{R}^{RLMN\times 1}$ is then partitioned into R blocks and reshaped using corresponding rank (i.e. $[L, M, N]$) per block.

Algorithm 1: Modified LU Factorization for OnlineBTD

```
Input: $\mathbf{A} \in \mathbb{R}^{n\times n}$
Output: Lower matrix $\mathbf{L}$, Upper matrix $\mathbf{U}$, Permutation matrix $\mathbf{P}$
1: $\mathbf{L} = eye(n)$; $\mathbf{P} = \mathbf{L}$; $\mathbf{U} = \mathbf{A}$;
2: for $k \leftarrow 1$ to $n$ do
3:     $[val \quad m] = max(abs(\mathbf{U}(k:n,k)));;m = m + k - 1$
4:     if m $\neq$ k then
5:         Interchange rows $m$ and $k$ in $\mathbf{U}$ and $\mathbf{P}$
6:         if k $\geq$ 2 then
7:             Interchange rows $m$ and $k$ in $\mathbf{L}$ for $(k-1)$ columns
8:         end if
9:     end if
10:    for $j \leftarrow (k+1)$ to $n$ do
11:        $\mathbf{L}(j,k) = \mathbf{U}(j,k)/\mathbf{U}(k,k)$;
           $\mathbf{U}(j,:) = \mathbf{U}(j,:) - \mathbf{L}(j,k) * \mathbf{U}(k,:)$;
12:    end for
13:    $\mathbf{L}(:,k) = \mathbf{L}(:,k) * \sqrt{\mathbf{A}(k,k) - \mathbf{L}(k,1:k-1) * \mathbf{L}(k,1:k-1)^T}$;
14: end for
15: return $\mathbf{L}$, $\mathbf{U}$, $\mathbf{P}$
```

Summary: For a 3-mode tensor that grows with time or at its 3^{rd} mode, we propose an efficient algorithm for tracking its BTD decomposition on the fly. We name this algorithm as OnlineBTD, comprising the following two stages:

1) Initialization stage (Algorithm (1) in supplementary section (A.3)): in case factors from old tensor $\underline{\mathbf{X}}_{old}$ are not available, then we obtain its BTD decomposition as ($\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathcal{G}$)

2) Update stage (Algorithm 2): for each new incoming data $\underline{\mathbf{X}}_{new}$, it is processed as:

- For the temporal mode 3, $\mathbf{C}$ is updated using Equ. (12)
- For non-temporal modes 1 and 2, $\mathbf{A}$ and $\mathbf{B}$ is updated using Equ. (14) and Equ. (15), respectively.
- For core-tensor, $\mathcal{G}$ is updated using Equ. (16) and accelerated using Equ. (17).

B. Extending to Higher order tensors

We now show how our approach is extended to higher-order cases. Consider N-mode tensor $\underline{\mathbf{X}}_{old} \in \mathbb{R}^{I_1\times I_2\times\cdots\times K_1}$. The

Algorithm 2: OnlineBTD Update Framework

```
Input: $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times K_2}$, old data factors
    $(\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)}), \underline{\mathbf{D}}$ , Rank $R$ and $L$.
Output: Updated factor matrices
    $(\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)}, \underline{\mathbf{D}})$,
1: Update temporal modes of tensor $\underline{\mathbf{X}}$ as:
   $\mathbf{A}^{(N)} \leftarrow \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \mathbf{X}^{(N)}_{new} * [\mathcal{G}_{r_{old}} . (\otimes_{i=1}^{N-1} \mathbf{A}^{(i)})^T]^{\dagger} \end{bmatrix} \quad \forall r \in [1, R]$
2: for $n \leftarrow 1$ to $N-1$ do
3:     Update other modes of tensor $\underline{\mathbf{X}}$ as: $\mathbf{A}^{(i)} \leftarrow \mathbf{X}^{(i)}_{new} *$
       $[\mathcal{G}_{r_{old}} (\otimes_{i \neq n}^{N} \mathbf{A}_r^{(i)})^T]^{\dagger} + \mathbf{A}^{(i)}_{old} \quad \forall i \in [1, N] \quad \forall r \in [1, R]$
4: end for
5: Update core tensor using $\underline{\mathbf{X}}_{new}$
   $\underline{\mathbf{D}} \leftarrow \begin{bmatrix} \otimes_{i=1}^{N} \mathbf{A}_1^{(i)} \\ \cdots \\ \otimes_{i=1}^{N} \mathbf{A}_R^{(i)} \end{bmatrix}^{\dagger} * \underline{\mathbf{X}}_{new}(:) \quad \forall i \in [1, N] \quad \forall r \in [1, R]$
6: return Updated $(\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)}, \underline{\mathbf{D}})$
```

factor matrices are $(\mathbf{A}^{(1)}_{old}, \mathbf{A}^{(2)}_{old}, \ldots, \mathbf{A}^{(N-1)}_{old}, \mathbf{A}^{(N)}_{old})$ for BTD decomposition with N^{th} mode as new incoming data. A new tensor $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times K_2}$ is added to $\underline{\mathbf{X}}_{old}$ to form new tensor of $\mathbb{R}^{I_1 \times I_2 \times \cdots \times K}$ where $K = K_1 + K_2$. The subscript $i \neq n$ indicated the n^{th} matrix is not included in the operation.

The Temporal mode can be updated as :

$$\mathbf{A}^{(N)} = \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \mathbf{A}^{(N)}_{new} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \mathbf{X}^{(N)}_{new} * [\mathcal{G}_{r_{old}} . (\otimes_{i=1}^{N-1} \mathbf{A}^{(i)})^T]^{\dagger} \end{bmatrix} \tag{19}$$

The Non-Temporal modes can be updated as:

$$\begin{aligned} \mathbf{A}^{(i)} &= \mathbf{X}^{(i)}_{new} * [\mathcal{G}_{r_{old}} (\otimes_{i \neq n}^{N} \mathbf{A}_r^{(i)})^T]^{\dagger} + \mathbf{A}^{(i)}_{old} \\ &= \mathbf{X}^{(i)}_{new} * [\mathcal{G}_{r_{old}} (Kr_r^{(n)})^T]^{\dagger} + \mathbf{A}^{(i)}_{old} \end{aligned} \tag{20}$$

where $i \in [1, N-1]$ and we denote the Kronecker(Kr) product of the first (N- 1) but the n^{th} loading matrices, $(\otimes_{i \neq n}^{N} \mathbf{A}_r^{(i)})$ as $Kr_r^{(n)}$.

The core tensor of BTD decomposition can be updated as:

$$\underline{\mathbf{D}} = \begin{bmatrix} \otimes_{i=1}^{N} \mathbf{A}_1^{(i)} \\ \cdots \\ \otimes_{i=1}^{N} \mathbf{A}_R^{(i)} \end{bmatrix}^{\dagger} * \underline{\mathbf{X}}_{new}(:) \tag{21}$$

Avoid duplicate Kronecker product: To avoid duplicate Kronecker product in above calculations, we use a dynamic programming method adapted from [39] to compute all the Kronecker products in one run as shown in Figure (3). The process uses intermediate results to avoid duplicate Kronecker product. The method goes through the factor matrix from both sides, until it reaches the results of first and last Kronecker product in a block and repeats the process for all the blocks.

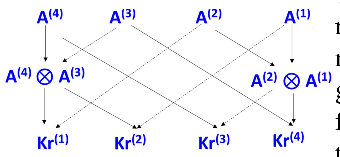


Fig. 3: Kronecker products for the 5th-order.

Accelerated computation summary: we accelerate computation in OnlineBTD via a) using *accelerated MTTKRONP* instead of *classic MTTKRONP*, b) using modified-LU factorization instead of classic pesudo-inverse, and c) by avoiding duplicate Kronecker product for higher order tensors. Finally, by putting everything together, we obtain the general version of our OnlineBTD algorithm, as presented in Algorithm 2.

Dataset	Statistics (K: Thousands M: Millions)			
	$I = J$	**K**	**[L,M,N]**	***Batch***
I	100	500	$[5, 6, 7]$	50
II	250	$1K$	$[5, 6, 7]$	50
III	$1K$	$1K$	$[5, 6, 7]$	10
IV	$1K$	$10K$	$[5, 6, 7]$	10
V	$1K$	$100K$	$[3, 4, 5]$	10
VI	$1K$	$1M$	$[3, 4, 5]$	10

TABLE III: Details for the synthetic datasets.

Dataset	I	J	K	[L,M,N]	*Batch*
ANT-Network [31]	822	822	41	$[3, 3, 3]$	10
EU Core [38]	1004	1004	526	$[8, 8, 8]$	10
EEG Signal [32]	109	896	$20K$	$[5, 5, 5]$	20

TABLE IV: Details for the real datasets.

IV. EMPIRICAL ANALYSIS

We design experiments to answer the following questions: **(Q1)** How fast, accurate and memory efficient are updates in OnlineBTD compared to classic BTD algorithm? **(Q2)** How does the running time of OnlineBTD increase as tensor data grow (in time mode)? **(Q3)** What is the influence of parameters on OnlineBTD? **(Q4)** How OnlineBTD used in real-world scenarios?

A. Experimental Setup

*1) **Synthetic Data**:* We provide the datasets used for evaluation in Table III. For all synthetic data we use rank $R = 3$. In the entries of factor matrix **A**, **B** and **C** Gaussian noise is added such that the signal-to-noise ratio is $10dB$.

*2) **Real Data**:* We provide the datasets used for evaluation in Table IV. Rank determination in the experiments is performed with the aid of the Core Consistency Diagnostic (CorConDia) method [2], [27] and the triangle method ,implemented by Tensorlab 3.0[37].

- **EU-Core**[38]: consists of e-mail data from a large European research institution over 526 days.
- **Ant Social Network**[31]: This dataset consists of information of all social interaction among ants, their behavior and spatial movement from all ants in six colonies over 41 days.
- **EEG Signal**[32]: This dataset includes 109 subjects who performed 14 experimental different motor/imagery tasks while 64-channel EEG were recorded using the BCI2000 system.

*3) **Baseline method**:* In this experiment, two baselines have been used as the competitors to evaluate the performance.

- **BTD-ALS** : an implementation of standard ALS fitting algorithm BTD. Implementation of BTD-ALS is not provided in Tensorlab [37]. Hence, we provide an efficient implementation of BTD-ALS to promote reproducibility.
- **BTD-NLS** [37] : an implementation of standard NLS fitting algorithm BTD with noisy initialization.

Note that there is no method in literature for online or incremental BTD tensors. Hence, we compare our proposed method against algorithms that decompose full tensor. Also, extending BTD-NLS to online settings is challenging and requires further research both to find the best fit and to interpret the role of the independent variables used in various inherit methods. It faces difficulties in fitting due to the narrow boundaries on the model and less flexibility.

*4) **Evaluation Metrics**:* We evaluate OnlineBTD and the baselines using three criteria: 1) **approximation loss**, 2) **CPU time** in second, and 3) **memory usage** in Megabytes. These measures provide a quantitative way to compare the performance of our method. For all criterion, lower is better.

B. Experimental Results

*1) **Accurate, Fast and Memory Efficient**:* First, as shown in Table V, we remark that OnlineBTD is both more memory-efficient and faster than the baseline methods and at the same time to a large extent comparable in terms of accuracy. In particular, the baseline methods fail to execute in the large problems i.e. SYN-III to SYN-VI for given target rank due to out of memory problems during the formation of core tensor $\underline{\mathcal{G}}$. This improvement stems from the fact that baseline methods attempt to decompose in full tensor, whereas OnlineBTD can do the same in streaming mode or on the fly, thus having higher immunity to large data volumes in short time intervals and small memory space with comparable accuracy.

For OnlineBTD we use $1-10\%$ of the time-stamp data in each dataset as existing old tensor data. The results for qualitative measure for data are provided in Table V. For each of the tensor data, the best performance is shown in bold. All state-of-art methods address the issue very well for small datasets. Compared with BTD-ALS and BTD-NLS, OnlineBTD gives lower or similar approximation loss and reduce the mean CPU running time by up to avg. 45% times for big tensor data. For all datasets, BTD-NLS's loss is lower than all methods. But it is able to handle up to $1K \times 1K \times 100$ size only. Most importantly, however, OnlineBTD performed very well on SYN-V and SYN-VI datasets, arguably the hardest of the six synthetic datasets we examined *where none of the baselines was able to run efficiently (under 48-72 hours)*. It significantly saved $10-60\%$ of computation time and saved $40-80\%$ memory space compared to baseline methods as shown in Table V. Hence, OnlineBTD is comparable to state-of-art methods for the small datasets and outperformed them for the large datasets. These results answer Q1 as the OnlineBTD has better qualitative measures to other methods.

*2) **Scalability Evaluation**:* To evaluate the scalability of our method, firstly, a dense tensor $\underline{\mathbf{X}}$ of small slice size $I = J = 100$ but longer 3^{rd} dimension ($K \in [10^2 - 10^8]$) is created. Its first $1-10\%$ timestamps of data is used for $\underline{\mathbf{X}}_{old}$ and each method's running time for processing batch of 100 data slices at each timestamp is measured. We decomposed it using fixed target rank $R = 3$ and fixed block rank $L = M = N = 5$. The baseline approach consumes more time as we increase the K. The baseline method runs up to 10^9 non-zero elements or 10^5 slices and runs out of memory for further data. However, our proposed method, successfully decomposed the tensor in reasonable time as shown in Figure 4. Overall, our method achieves up to 27% (average) speed-up regarding the time required and average 76% gain over memory saving. This answers our Q2. In terms of batch size, it is observed that the time consumed by our method is linearly increasing as the batch size grows. However, their slopes vary with different rank used. The analysis is included in the supplementary section (B) due to the limitation of space here.

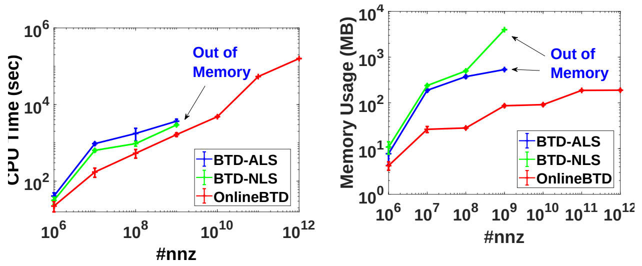


Fig. 4: CPU time (sec) and Memory (MB) used for processing slices to tensor $\underline{\mathbf{X}}$ incrementing in its time mode. The time and space consumption increases linearly. The mean approximation error is $\leq 10\%$ for all experiments. $\#nnz$: Number of non-zero elements.

*3) **Sensitivity of OnlineBTD**:* We extensively evaluate sensitivity of OnlineBTD w.r.t. target rank R, block rank (L, M, N) and noise added during initialization process. For all experiments, we use tensor $\underline{\mathbf{X}} \in \mathbb{R}^{250 \times 250 \times 10^5}$ and batch size of 10 slices at a time.

Sensitivity w.r.t tensor rank-R: We fixed the block rank $(L = 5, M = 6, N = 7)$ with initialization factor noise is fixed at 10dB.The number of blocks play an important role in OnlineBTD. We see in Figure 5 (a) that increasing number of blocks result in decrease of approximation error of reconstructed tensor. The CPU Time and Memory (MB) is linearly (slope 1.03) increased as shown in figure 5 (b) and 5 (c).

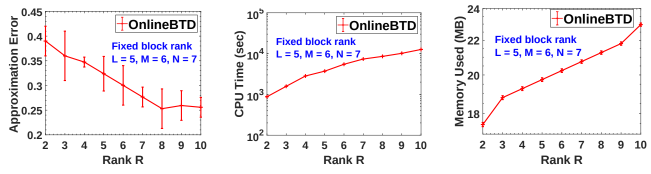


Fig. 5: The average approximation error, time and memory usage for varying target rank 'R' on different datasets.

Sensitivity w.r.t block rank (L, M, N): To evaluate the impact of block rank - (L, M, N), we fixed tensor rank $R = 5$ and noise added to $10dB$. We can see that with higher values of the (L, M, N), approximation error is improved as shown in Figure 6 (a) and become saturated when original block rank is achieved. Higher the block size, more memory and time is required to compute them as shown in figure 6.

*4) **Comparison to Online Tucker and Online CP**:* We also evaluate OnlineBTD performance in terms of computation time with other CP and Tucker online methods as given below:

- **Online Tucker Decomposition** [34]: *STA* is a streaming tensor analysis method, which provides a fast, stream-

SYN	Approximation Loss			CPU Time (sec)			Memory Usage (MBytes)		
	BTD-ALS	BTD-NLS	OnlineBTD	BTD-ALS	BTD-NLS	OnlineBTD	BTD-ALS	BTD-NLS	OnlineBTD
I	0.12 ± 0.01	**0.04 ± 0.03**	0.07 ± 0.02	79.62 ± 4.5	**12.7 ± 3.1**	13.36 ± 6.31	7.725 ± 0.01	14.7 ± 0.02	**4.23 ± 0.01**
II	0.16 ± 0.04	**0.09 ± 0.03**	**0.09 ± 0.01**	466.91 ± 23.6	325.3 ± 34.9	**106.99 ± 12.61**	157.2 ± 0.01	219.4 ± 4.5	**32.70 ± 0.05**
III	[OoM]	[OoM]	**0.11 ± 0.01**	[OoM]	[OoM]	**831.52 ± 43.82**	[OoM]	[OoM]	**315.11 ± 0.01**
IV	[OoM]	[OoM]	**0.13 ± 0.06**	[OoM]	[OoM]	**2858.45 ± 59.45**	[OoM]	[OoM]	**314.21 ± 0.01**
V	[OoM]	[OoM]	**0.16 ± 0.03**	[OoM]	[OoM]	**7665.23 ± 89.81**	[OoM]	[OoM]	**316.45 ± 0.01**
VI	[OoM]	[OoM]	**0.39 ± 0.11**	[OoM]	[OoM]	**89349.62 ± 253.06**	[OoM]	[OoM]	**312.21 ± 0.01**

TABLE V: Experimental results for approximation error, CPU Time in seconds and Memory Used in MB for synthetic tensor. We see that OnlineBTD gives stable decomposition in reasonable time and space as compared to classic BTD method. The boldface means the best results.

R	REAL	Approximation Loss			CPU Time (sec)			Memory Usage (MBytes)		
		BTD-ALS	BTD-NLS	OnlineBTD	BTD-ALS	BTD-NLS	OnlineBTD	BTD-ALS	BTD-NLS	OnlineBTD
5	A	0.36 ± 0.13	**0.34 ± 0.14**	0.37 ± 0.13	385.51 ± 67.52	432.47 ± 46.89	**127.32 ± 13.31**	353.35 ± 0.01	211.34 ± 0.02	**45.37 ± 0.01**
	B	[OoM]	[OoM]	**0.31 ± 0.09**	[OoM]	[OoM]	**742.08 ± 65.37**	[OoM]	[OoM]	**312.34 ± 0.01**
	C	[OoM]	[OoM]	**0.34 ± 0.03**	[OoM]	[OoM]	**4834.3 ± 169.42**	[OoM]	[OoM]	**572.7 ± 0.01**
10	A	0.22 ± 0.11	**0.21 ± 0.09**	0.23 ± 0.02	593.1 ± 53.5	602.4 ± 77.34	**283.45 ± 43.47**	490.3 ± 0.03	381.59 ± 0.01	**90.43 ± 0.01**
	B	[OoM]	[OoM]	**0.23 ± 0.14**	[OoM]	[OoM]	**1267.4 ± 121.3**	[OoM]	[OoM]	**583.4 ± 0.01**
	C	[OoM]	[OoM]	**0.27 ± 0.05**	[OoM]	[OoM]	**8639.7 ± 392.36**	[OoM]	[OoM]	**845.5 ± 0.01**

TABLE VI: A $\leftarrow$ ANT-Network; B $\leftarrow$ EU-Core; C $\leftarrow$ EEG Signals dataset. The average and standard deviation of memory usage and time metric comparison on real world dataset using two different target for five random initialization. For EU-Core and EEG signal datasets, baselines are unable to create intermediate core tensor. The baseline method has less approximation loss as compared to our proposed method. However, the time and memory saving ($> 50\%$) with OnlineBTD is significant.

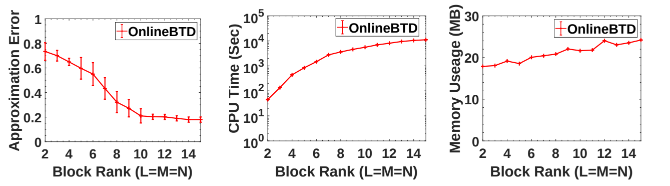


Fig. 6: The average approximation error, CPU time in seconds and memory usage in MBytes for varying block rank - (L,M,N) of $\underline{\mathbf{X}}$ with original $L = M = N = 10$. As the rank increases, lower approximation error is achieved. Increase in time and memory consumption is expected behaviour.

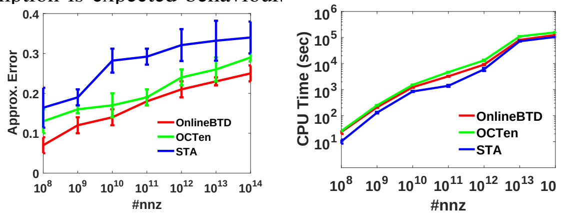


Fig. 7: The CPU time in seconds and approximation loss for CP/Tucker/BTD online tensor decomposition.

ing approximation method that continuously track the changes of projection matrices using the online PCA technique.

- **Online CP Decomposition** [14]: *OCTen* is a compression-based online parallel implementation for the CP decomposition.

Here, we use dense tensor $\underline{\mathbf{X}}$ of slice size $I = J = 1000$ but longer 3^{rd} dimension ($K \in [10^2 - 10^8]$) for evaluation. For CP and Tucker decomposition, we use rank $R = 10$ and for OnlineBTD we use $L = 10$ and $R = 1$. We see in Figure 7, OnlineBTD performance is better than OCTen, however, STA outperforms the all the methods. In terms of Fitness, OnlineBTD is better ($> 35\%$) than all the baseline methods.

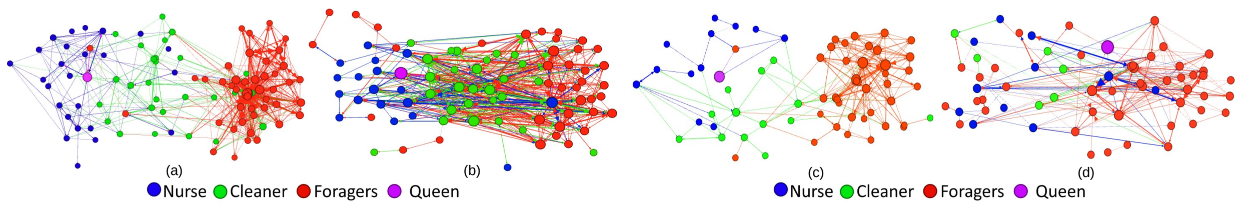
C. Effectiveness on Real-world data

We evaluate the performance of OnlineBTD approach for the real datasets as well. The empirical results are given in Table (VI).

*1) **Ant Social Network**:* Here, we discuss the usefulness of OnlineBTD towards extracting meaningful communities or clusters of ants from Ant Social Network[31] data. It is well known fact that ants live in highly organized societies with a specific division of labor among workers, such as cleaning the nest, caring for the eggs, or foraging, but little is known or researched about how these division of labor are generated. This network consists of more than 1 million interactions within 41 days between 822 ants. The challenge in community detection of such large data is to capture the functional behavioral based on spatial and temporal distribution of interactions. Therefore, there is a serious need to learn more useful representation.

Qualitative Analysis: For this case study, we decompose tensor in batch of 10 days to extract communities. We observed that the ant organization in this dataset is type Pleometrosis where multiple egg laying queen create there own colonies within organization. There are three communities [24] namely nurse (N), cleaner (C) and forager (F) present in each colony. We compute *F1-score* to evaluate the communities quality.

We focus our analysis on communities within each colony of ants in this dataset. The word "ant colony" refers to groups of workers, fertile individuals, and brood that non-aggressively stay together and work jointly. Our proposed method helped us to track temporal changes among the groups by performing community detection analyses on the batches of 10-day periods. Figure 8(e) shows behavioral trajectories of three communities of ants over the 41 days. We observed that ants exhibit preferred behavioral trajectory i.e. move from nursing (located near the queen) to cleaning (move throughout the colony) to foraging (moving in and out of the colony) as they age. The most common transition among them was from cleaner to forager. Conceptually, those ants share similar behaviour in terms of movement and work load. As a result, it becomes a very important challenge to accurately cluster them. Nevertheless, OnlineBTD tracked the behaviour changes very well and achieves significantly good performance in terms of $F1 - score$ i.e ≈ 0.79 as compared to baseline (max F1-Score ≈ 0.63). The communities of colony 1 for day 11, 21, 31 and 41 is shown in Figure 8(a)-(d). The heatmap



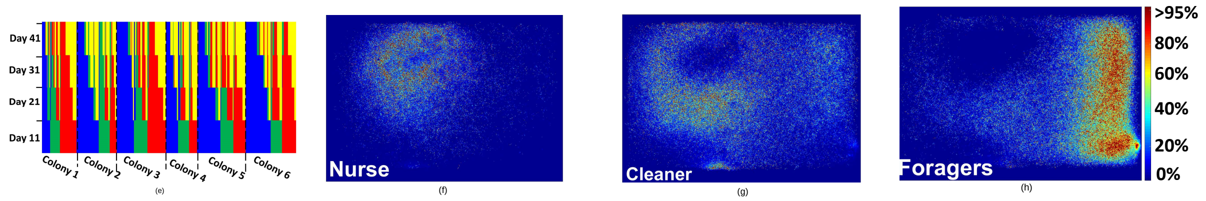


Fig. 8: (a)-(d) Community detection results of colony 1 on days 11, 21, 31 and 41 of the experiment; (e) The community profiling of the each ant for every 10-day period for all colonies. Blue: nurse community; green: cleaning community; red: foraging community. Ants that disappeared because they are lost or dead are indicated in yellow; (f)-(h) Spatial distribution of nurses, cleaners, and foragers.

(Figure 8(f)-(h)) provides spatial distribution of community interactions. As any of the baseline does not have capability to capture this temporal behaviour efficiently, our proposed method gives advantage to decompose the data in streaming fashion in reasonable time.

*2) **EU-Core [38]**:* : This is a temporal network dataset of communication via. emails between students, professor and staff members of the research institution during October 2003 to May 2005 (18 months) of 42 departments. EU-Core consists of multiple strongly connected communities corresponding to many instances of communication within department. However, we also find numerous re-occurred groups which indicate communication across departments. Interestingly, we note that between Oct,2004 -Jan,2005 one community of researchers established a continuous communication consisting of $80-85$ researchers who interacted each month from the same department. Suddenly, some members disappeared during Dec'04 and again in Jan'05 communication resumed back normally as shown in Figure 9. We believe that this may reflect the days of the week of Christmas and New year holidays.

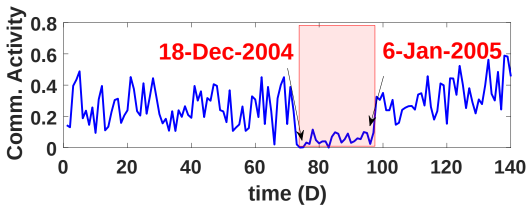


Fig. 9: A community activity profile of EU-Core.

*3) **EEG Signals [32]**:* : The challenge in movement detection of EEG signal data is to capture behaviour regarding the spatio-temporal representations of raw streams. EEG signals usually consist of various noises e.g. cardiac signal. Apart from the systems noises, such as power line interference etc. EEG signals consist from some unavoidable noises like eye blinks, heart beat and muscle activity, all harm to collecting high signal-to-noise ratio EEG signals. It is difficult to make sure that the subjects concentrate on the performing tasks during the whole experiment period. The offline tensor decomposition model assumes that the subject maintains the same spectral structure and topography within the observed window.

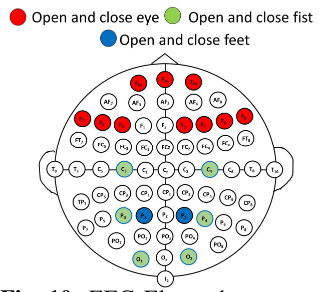


Fig. 10: EEG Electrode map.

However, these are typically characterised by evolving repetitive sharp waves. Our proposed method allows more variability and more interaction between the factors in order to capture such non-stationarities. We find that frontal electrode lobes i.e F_2 through F_8 and front polar electrode lobes F_{P1} and F_{P2} gives the better separations results to differentiate the motor movements tasks between eye open and eye closed. The parietal (P_3, P_4), central (C_3, C_4) and occipital (O_1, O_2) electrode lobes as shown in Figure 10 gives results at temporal scales for open and close left or right fist. This movement capturing over temporal mode is beneficial to users with severe disabilities.

The results in Table VI and above qualitative analysis shows the effectiveness of the decomposition and confirms that the OnlineBTD can be used for various types of data analysis and this answers **Q4**.

Reproducibility: To promote reproducibility, we make our MATLAB implementation publicly available at Link[1].

V. CONCLUSIONS AND FUTURE WORK

We proposed OnlineBTD, a novel online method to learn Block Term Decomposition (BTD). The performance of the proposed method is assessed via experiments on six synthetic as well as three real-world networks. We summarize our contribution as:

- The proposed framework effectively identify the beyond rank-1 latent factors of incoming slice(s) to achieve online block term tensor decompositions. To further enhance the capability, we also tailor our general framework towards higher-order online tensors.

[1]http://www.cs.ucr.edu/~egujr001/ucr/madlab/src/OnlineBTD.zip

- Through experimental evaluation on multiple datasets, we show that OnlineBTD provides stable decompositions and have significant improvement in terms of run time and memory usage.
- Utility: we provide a clean and effective implementation of BTD-ALS and all accelerated supporting implementations along with OnlineBTD source code.

There is still room for improving our method. One direction is to explore online BTD for NLS (Non-Linear Square). Another direction is to further extend it for more general dynamic tensors that may be changed on any modes so that our method can be more suitable for applications such as computer vision.

VI. Acknowledgements

Research was supported by the Department of the Navy, Naval Engineering Education Consortium under award no. N00174-17-1-0005, the National Science Foundation Grant no. 1901379 and an Adobe Data Science Research Faculty Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

[1] Hassan Akbari, Mohammad B Shamsollahi, and Ronald Phlypo. Fetal ecg extraction using πtucker decomposition. In *2015 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 174–178. IEEE, 2015.

[2] Rasmus Bro and Henk AL Kiers. A new efficient method for determining the number of components in parafac models. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 2003.

[3] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of “eckart-young” decomposition. *Psychometrika*, 35(3):283–319, 1970.

[4] Christos Chatzichristos, Eleftherios Kofidis, Yiannis Kopsinis, Manuel Morante Moreno, and Sergios Theodoridis. Higher-order block term decomposition for spatially folded fmri data. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 3–15. Springer, 2017.

[5] Pierre Courrieu. Fast computation of moore-penrose inverse matrices. *arXiv preprint arXiv:0804.4809*, 2008.

[6] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms—part i: Lemmas for partitioned matrices. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1022–1032, 2008.

[7] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms—part ii: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1033–1066, 2008.

[8] Lieven De Lathauwer. Blind separation of exponential polynomials and the decomposition of a tensor in rank-(l_r,l_r,1) terms. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1451–1474, 2011.

[9] Lieven De Lathauwer and Dimitri Nion. Decompositions of a higher-order tensor in block terms—part iii: Alternating least squares algorithms. *SIAM journal on Matrix Analysis and Applications*, 30(3):1067–1083, 2008.

[10] Pedro Marinho Ramos de Oliveira and Vicente Zarzoso. Block term decomposition of ecg recordings for atrial fibrillation analysis: Temporal and inter-patient variability. *Journal of Communication and Information Systems*, 34(1):111–119, 2019.

[11] Ekta Gujral, Ravdeep Pasricha, and Evangelos Papalexakis. Beyond rank-1: Discovering rich community structure in multi-aspect graphs. In *Proceedings of The Web Conference 2020*, 2020.

[12] Ekta Gujral, Ravdeep Pasricha, and Evangelos E Papalexakis. Sambaten: Sampling-based batch incremental tensor decomposition. In *Proceedings of the 2018 SIAM SDM*, pages 387–395. SIAM, 2018.

[13] Ekta Gujral, Ravdeep Pasricha, Tianxiong Yang, and Evangelos E Papalexakis. Octen: Online compression-based tensor decomposition. *arXiv preprint arXiv:1807.01350*, 2018.

[14] Ekta Gujral, Ravdeep Pasricha, Tianxiong Yang, and Evangelos E Papalexakis. Octen: Online compression-based tensor decomposition. In *2019 IEEE 8th International Workshop on CAMSAP*. IEEE, 2019.

[15] Maimoona Hafeez, Asthma Yasin, Nazia Badar, Muhammad Irfan Pasha, Nishat Akram, and Bushra Gulzar. Prevalence and indications of caesarean section in a teaching hospital. *JIMSA*, 27(1):15–6, 2014.

[16] R.A. Harshman. Foundations of the parafac procedure: Models and conditions for an” explanatory” multimodal factor analysis. 1970.

[17] Nicholas J Higham. *Accuracy and stability of numerical algorithms*, volume 80. Siam, 2002.

[18] Francis Begnaud Hildebrand. *Introduction to numerical analysis*. Courier Corporation, 1987.

[19] Borbála Hunyadi, Daan Camps, Laurent Sorber, Wim Van Paesschen, Maarten De Vos, Sabine Van Huffel, and Lieven De Lathauwer. Block term decomposition for modelling epileptic seizures. *EURASIP Journal on Advances in Signal Processing*, 2014.

[20] T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM review*, 2009.

[21] Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. The dynamics of viral marketing. *ACM TWEB*, pages 5–es, 2007.

[22] Na Li. *Variants of ALS on Tensor Decompositions and Applications*. PhD thesis, Clarkson University, 2013.

[23] MATLAB. Kron product. https://www.mathworks.com/help/matlab/ref/kron.html?s_tid=mwa_osa_a.

[24] Danielle P Mersch, Alessandro Crespi, and Laurent Keller. Tracking individuals shows spatial fidelity is a key regulator of ant social organization. *Science*, 340(6136):1090–1093, 2013.

[25] Iman Mousavian, Mohammad Bagher Shamsollahi, and Emad Fatemizadeh. Noninvasive fetal ecg extraction using doubly constrained block-term decomposition. *Mathematical biosciences and engineering: MBE*, 17(1):144–159, 2019.

[26] D. Nion and N.D. Sidiropoulos. Adaptive algorithms to track the parafac decomposition of a third-order tensor. *Signal Processing, IEEE Transactions on*, 57(6):2299–2310, 2009.

[27] Evangelos E Papalexakis. Automatic unsupervised tensor mining with quality assessment. In *Proceedings of the 2016 SIAM Int. Conf. on Data Mining*. SIAM, 2016.

[28] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM TIST*, 8(2):16, 2016.

[29] Medhat A Rakha. On the moore–penrose generalized inverse matrix. *Applied Mathematics and Computation*, 158(1):185–200, 2004.

[30] Lucas N Ribeiro, Antonio R Hidalgo-Muñoz, Gérard Favier, João Cesar M Mota, André LF De Almeida, and Vicente Zarzoso. A tensor decomposition approach to noninvasive atrial activity extraction in atrial fibrillation ecg. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 2576–2580. IEEE, 2015.

[31] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[32] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004.

[33] Shaden Smith, Kejun Huang, Nicholas D Sidiropoulos, and George Karypis. Streaming tensor factorization for infinite data sources. In *Proceedings of the 2018 SIAM SDM*, pages 81–89. SIAM, 2018.

[34] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383, 2006.

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

[36] M. Alex O. Vasilescu and Eric Kim. Compositional hierarchical tensor factorization: Representing hierarchical intrinsic and extrinsic causal factors. Aug. 2019.

[37] Nico Vervliet, Otto Debals, and Lieven De Lathauwer. Tensorlab 3.0—numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization. In *2016 50th Asilomar Conference*, pages 1733–1738. IEEE, 2016.

[38] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD Int. Conf. on KDD*, pages 555–564. ACM, 2017.

[39] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. Accelerating online cp decompositions for higher order tensors. In *Proceedings of the 22nd ACM SIGKDD Int. Conf. on KDD*, pages 1375–1384, 2016.